# PIC18F2480/2580/4480/4580
# Data Sheet

28/40/44-Pin
Enhanced Flash Microcontrollers
with ECAN™ Technology, 10-Bit A/D
and nanoWatt Technology

© 2009 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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, PIC$^{32}$ logo, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock 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.

All other trademarks mentioned herein are property of their respective companies.

© 2009, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

*Microchip received ISO/TS-16949:2002 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.*


© 2009 Microchip Technology Inc.

# MICROCHIP PIC18F2480/2580/4480/4580

## 28/40/44-Pin Enhanced Flash Microcontrollers with ECAN™ Technology, 10-Bit A/D and nanoWatt Technology

### Power-Managed Modes:

- Run: CPU on, Peripherals on
- Idle: CPU off, Peripherals on
- Sleep: CPU off, Peripherals off
- Idle mode Currents Down to 6.1 μA Typical
- Sleep mode Current Down to 0.2 μA Typical
- Timer1 Oscillator: 1 μA, 32 kHz, 2V
- Watchdog Timer: 1.7 μA
- Two-Speed Oscillator Start-up

### Flexible Oscillator Structure:

- Four Crystal modes, up to 40 MHz
- 4x Phase Lock Loop (PLL) – Available for Crystal and Internal Oscillators)
- Two External RC modes, up to 4 MHz
- Two External Clock modes, up to 40 MHz
- Internal Oscillator Block:
  - Fast wake from Sleep and Idle, 1 μs typical
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - Provides a complete range of clock speeds, from 31 kHz to 32 MHz when used with PLL
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if peripheral clock stops

### Special Microcontroller Features:

- C Compiler Optimized Architecture with Optional Extended Instruction Set
- 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
- 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

### Peripheral Highlights:

- High-Current Sink/Source 25 mA/25 mA
- Three External Interrupts
- One Capture/Compare/PWM (CCP) module
- Enhanced Capture/Compare/PWM (ECCP) module (40/44-pin devices only):
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Master Synchronous Serial Port (MSSP) module Supporting 3-Wire SPI (all 4 modes) and $I^2C$™ Master and Slave modes
- Enhanced Addressable USART module
  - Supports RS-485, RS-232 and LIN/J2602
  - RS-232 operation using internal oscillator block
  - Auto-wake-up on Start bit
  - Auto-Baud Detect
- 10-Bit, up to 11-Channel Analog-to-Digital Converter (A/D) module, up to 100 ksps
  - Auto-acquisition capability
  - Conversion available during Sleep
- Dual Analog Comparators with Input Multiplexing

### ECAN Technology Module Features:

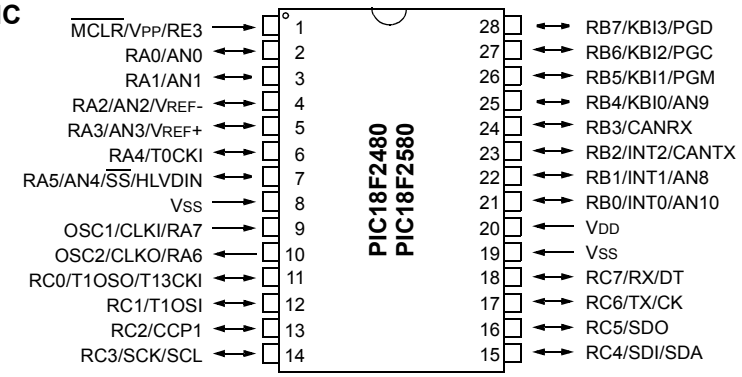
- Message Bit Rates up to 1 Mbps
- Conforms to CAN 2.0B Active Specification
- Fully Backward Compatible with PIC18XXX8 CAN modules
- Three Modes of Operation:
  - Legacy, Enhanced Legacy, FIFO
- Three Dedicated Transmit Buffers with Prioritization
- Two Dedicated Receive Buffers
- Six Programmable Receive/Transmit Buffers
- Three Full 29-Bit Acceptance Masks
- 16 Full 29-Bit Acceptance Filters w/Dynamic Association
- DeviceNet™ Data Byte Filter Support
- Automatic Remote Frame Handling
- Advanced Error Management Features

| Device | Program Memory | | Data Memory | | I/O | 10-Bit A/D (ch) | CCP/ ECCP (PWM) | MSSP | | EUSART | Comp. | Timers 8/16-bit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Flash (bytes) | # Single-Word Instructions | SRAM (bytes) | EEPROM (bytes) | | | | SPI | Master $I^2C$™ | | | |
| PIC18F2480 | 16K | 8192 | 768 | 256 | 25 | 8 | 1/0 | Y | Y | 1 | 0 | 1/3 |
| PIC18F2580 | 32K | 16384 | 1536 | 256 | 25 | 8 | 1/0 | Y | Y | 1 | 0 | 1/3 |
| PIC18F4480 | 16K | 8192 | 768 | 256 | 36 | 11 | 1/1 | Y | Y | 1 | 2 | 1/3 |
| PIC18F4580 | 32K | 16384 | 1536 | 256 | 36 | 11 | 1/1 | Y | Y | 1 | 2 | 1/3 |

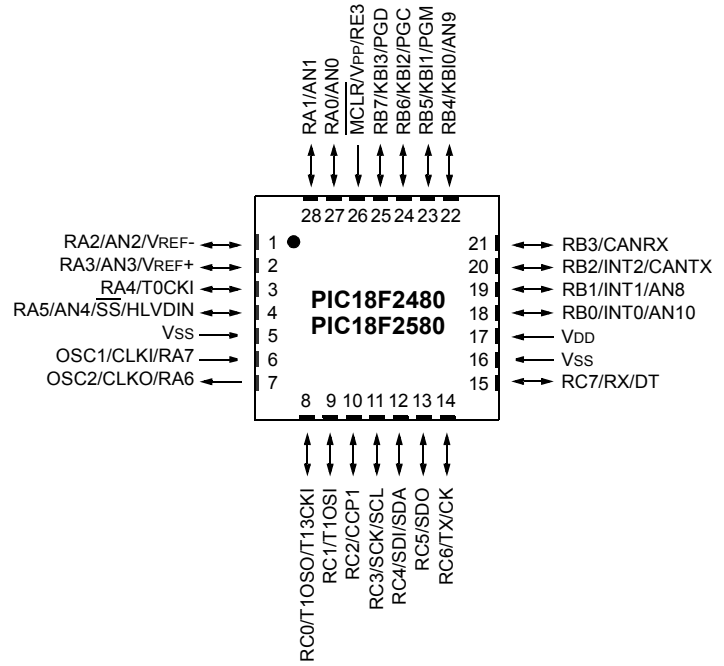
© 2009 Microchip Technology Inc.

## Pin Diagrams

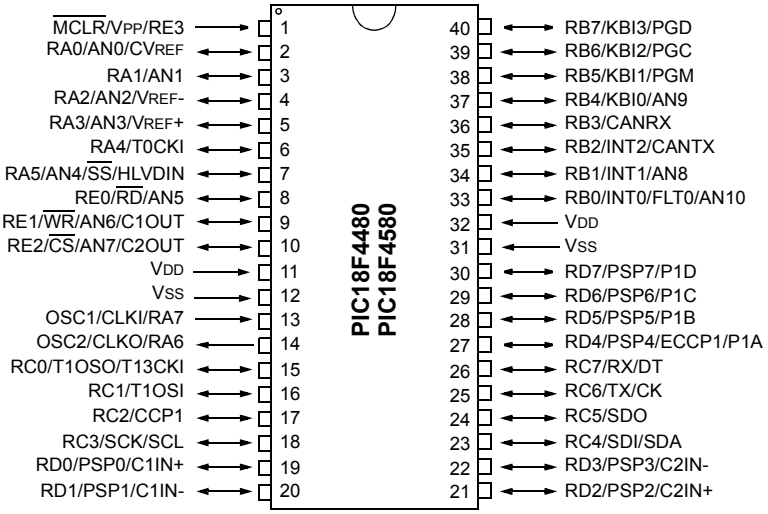
### 28-Pin SPDIP, SOIC




### 28-Pin QFN




### 40-Pin PDIP

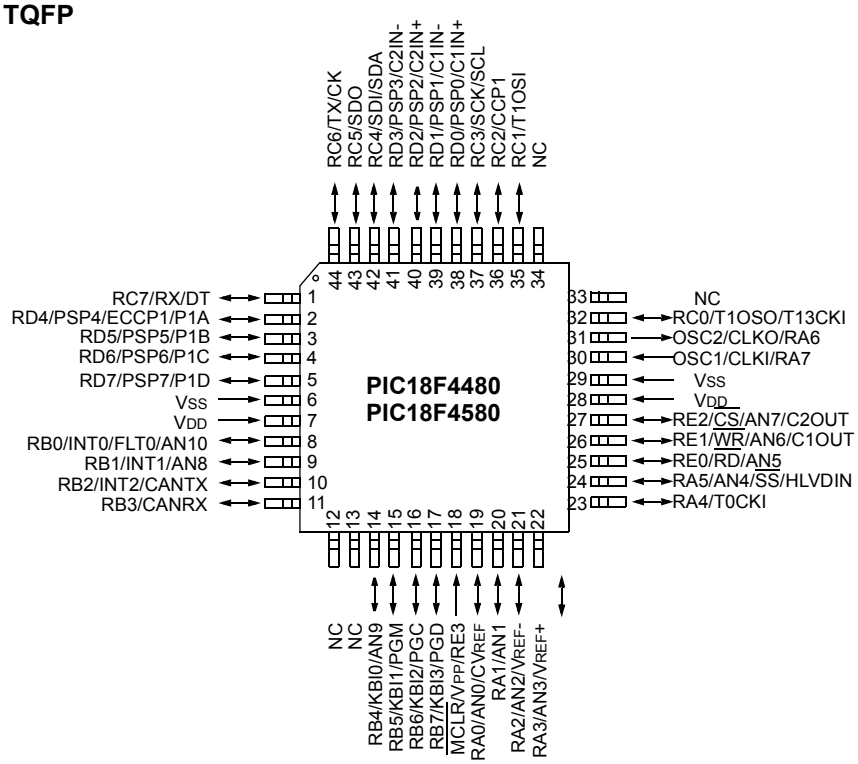




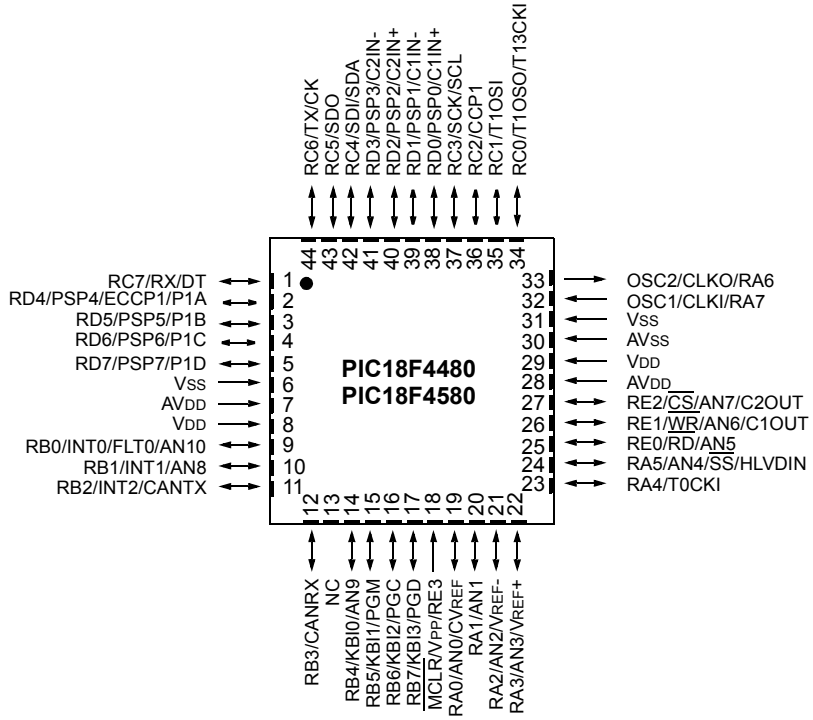
© 2009 Microchip Technology Inc.

## Pin Diagrams (Continued)

### 44-Pin TQFP




### 44-Pin QFN(1)




**Note 1:** For the QFN package, it is recommended that the bottom pad be connected to VSS.

## Table of Contents

1.0 Device Overview ..... 9
2.0 Guidelines for Getting Started with PIC18F Microcontrollers ..... 25
3.0 Oscillator Configurations ..... 29
4.0 Power-Managed Modes ..... 39
5.0 Reset ..... 47
6.0 Memory Organization ..... 67
7.0 Flash Program Memory ..... 101
8.0 Data EEPROM Memory ..... 111
9.0 8 x 8 Hardware Multiplier ..... 117
10.0 Interrupts ..... 119
11.0 I/O Ports ..... 135
12.0 Timer0 Module ..... 151
13.0 Timer1 Module ..... 155
14.0 Timer2 Module ..... 161
15.0 Timer3 Module ..... 163
16.0 Capture/Compare/PWM (CCP) Modules ..... 167
17.0 Enhanced Capture/Compare/PWM (ECCP) Module ..... 177
18.0 Master Synchronous Serial Port (MSSP) Module ..... 191
19.0 Enhanced Universal Synchronous Receiver Transmitter (EUSART) ..... 231
20.0 10-Bit Analog-to-Digital Converter (A/D) Module ..... 253
21.0 Comparator Module ..... 263
22.0 Comparator Voltage Reference Module ..... 269
23.0 High/Low-Voltage Detect (HLVD) ..... 273
24.0 ECAN Module ..... 279
25.0 Special Features of the CPU ..... 349
26.0 Instruction Set Summary ..... 367
27.0 Development Support ..... 417
28.0 Electrical Characteristics ..... 421
29.0 Packaging Information ..... 459
Appendix A: Revision History ..... 471
Appendix B: Device Differences ..... 471
Appendix C: Conversion Considerations ..... 472
Appendix D: Migration from Baseline to Enhanced Devices ..... 472
Appendix E: Migration From Mid-Range to Enhanced Devices ..... 473
Appendix F: Migration From High-End to Enhanced Devices ..... 473
The Microchip Web Site ..... 487
Customer Change Notification Service ..... 487
Customer Support ..... 487
Reader Response ..... 488
PIC18F2480/2580/4480/4580 Product Identification System ..... 489


© 2009 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@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

### Most Current Data Sheet

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

http://www.microchip.com

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

### Errata

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

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

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, 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/cn** to receive the most current information on all of our products.

**NOTES:**


© 2009 Microchip Technology Inc.

# 1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F2480
- PIC18F2580
- PIC18F4480
- PIC18F4580

This family of devices 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. In addition to these features, the PIC18F2480/2580/4480/4580 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 nanoWatt TECHNOLOGY

All of the devices in the PIC18F2480/2580/4480/4580 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 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.
- **Extended Instruction Set:** In addition to the standard 75 instructions of the PIC18 instruction set, PIC18F2480/2580/4480/4580 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

### 1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F2480/2580/4480/4580 family offer ten 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 6 user-selectable clock frequencies, between 125 kHz to 4 MHz, for a total of 8 clock frequencies. This option frees the two oscillator pins for use as additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the high-speed crystal and internal oscillator modes, which allows clock speeds of up to 40 MHz. Used with the internal oscillator, the PLL gives users a complete selection of clock speeds, from 31 kHz to 32 MHz – all without using an external crystal or clock circuit.

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.

## 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.
- **Extended Instruction Set:** The PIC18F2480/2580/4480/4580 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **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 Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN/J2602 bus protocol. Other enhancements include automatic baud rate detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **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 131 seconds, that is stable across operating voltage and temperature.

## 1.3 Details on Individual Family Members

Devices in the PIC18F2480/2580/4480/4580 family are available in 28-pin (PIC18F2X80) and 40/44-pin (PIC18F4X80) packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in six ways:

1. Flash program memory (16 Kbytes for PIC18FX480 devices; 32 Kbytes for PIC18FX580 devices).
2. A/D channels (8 for PIC18F2X80 devices; 11 for PIC18F4X80 devices).
3. I/O ports (3 bidirectional ports and 1 input only port on PIC18F2X80 devices; 5 bidirectional ports on PIC18F4X80 devices).
4. CCP and Enhanced CCP implementation (PIC18F2X80 devices have 1 standard CCP module; PIC18F4X80 devices have one standard CCP module and one ECCP module).
5. Parallel Slave Port (present only on PIC18F4X80 devices).
6. PIC18F4X80 devices provide two comparators.

All other features for devices in this family are identical. These are summarized in Table 1-1.

The pinouts for all devices are listed in Table 1-2 and Table 1-3.

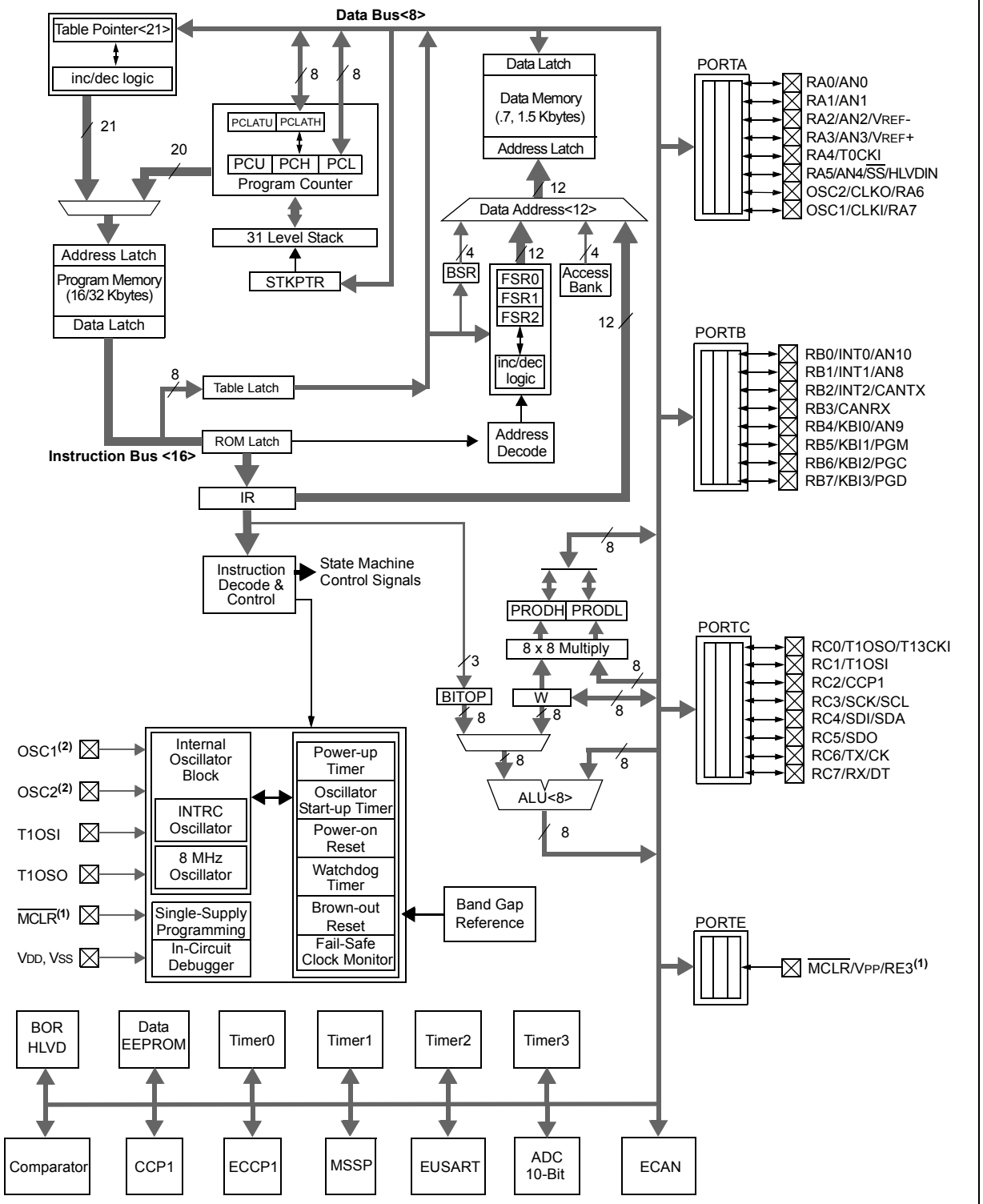
Like all Microchip PIC18 devices, members of the PIC18F2480/2580/4480/4580 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an “F” in the part number (such as PIC18**F**2580), accommodate an operating $V_{DD}$ range of 4.2V to 5.5V. Low-voltage parts, designated by “LF” (such as PIC18**LF**2580), function over an extended $V_{DD}$ range of 2.0V to 5.5V.


© 2009 Microchip Technology Inc.

**TABLE 1-1: DEVICE FEATURES**

| Features | PIC18F2480 | PIC18F2580 | PIC18F4480 | PIC18F4580 |
|---|---|---|---|---|
| Operating Frequency | DC – 40 MHz | DC – 40 MHz | DC – 40 MHz | DC – 40 MHz |
| Program Memory (Bytes) | 16384 | 32768 | 16384 | 32768 |
| Program Memory (Instructions) | 8192 | 16384 | 8192 | 16384 |
| Data Memory (Bytes) | 768 | 1536 | 768 | 1536 |
| Data EEPROM Memory (Bytes) | 256 | 256 | 256 | 256 |
| Interrupt Sources | 19 | 19 | 20 | 20 |
| I/O Ports | Ports A, B, C, (E) | Ports A, B, C, (E) | Ports A, B, C, D, E | Ports A, B, C, D, E |
| Timers | 4 | 4 | 4 | 4 |
| Capture/Compare/PWM Modules | 1 | 1 | 1 | 1 |
| Enhanced Capture/ Compare/PWM Modules | 0 | 0 | 1 | 1 |
| ECAN Module | 1 | 1 | 1 | 1 |
| Serial Communications | MSSP, Enhanced USART | MSSP, Enhanced USART | MSSP, Enhanced USART | MSSP, Enhanced USART |
| Parallel Communications (PSP) | No | No | Yes | Yes |
| 10-Bit Analog-to-Digital Module | 8 Input Channels | 8 Input Channels | 11 Input Channels | 11 Input Channels |
| Comparators | 0 | 0 | 2 | 2 |
| 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 | 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 High/ Low-Voltage Detect | Yes | Yes | Yes | Yes |
| Programmable Brown-out Reset | Yes | Yes | Yes | Yes |
| Instruction Set | 75 Instructions; 83 with Extended Instruction Set Enabled | 75 Instructions; 83 with Extended Instruction Set Enabled | 75 Instructions; 83 with Extended Instruction Set Enabled | 75 Instructions; 83 with Extended Instruction Set Enabled |
| Packages | 28-pin SPDIP 28-pin SOIC 28-pin QFN | 28-pin SPDIP 28-pin SOIC 28-pin QFN | 40-pin PDIP 44-pin QFN 44-pin TQFP | 40-pin PDIP 44-pin QFN 44-pin TQFP |

**FIGURE 1-1: PIC18F2480/2580 (28-PIN) BLOCK DIAGRAM**

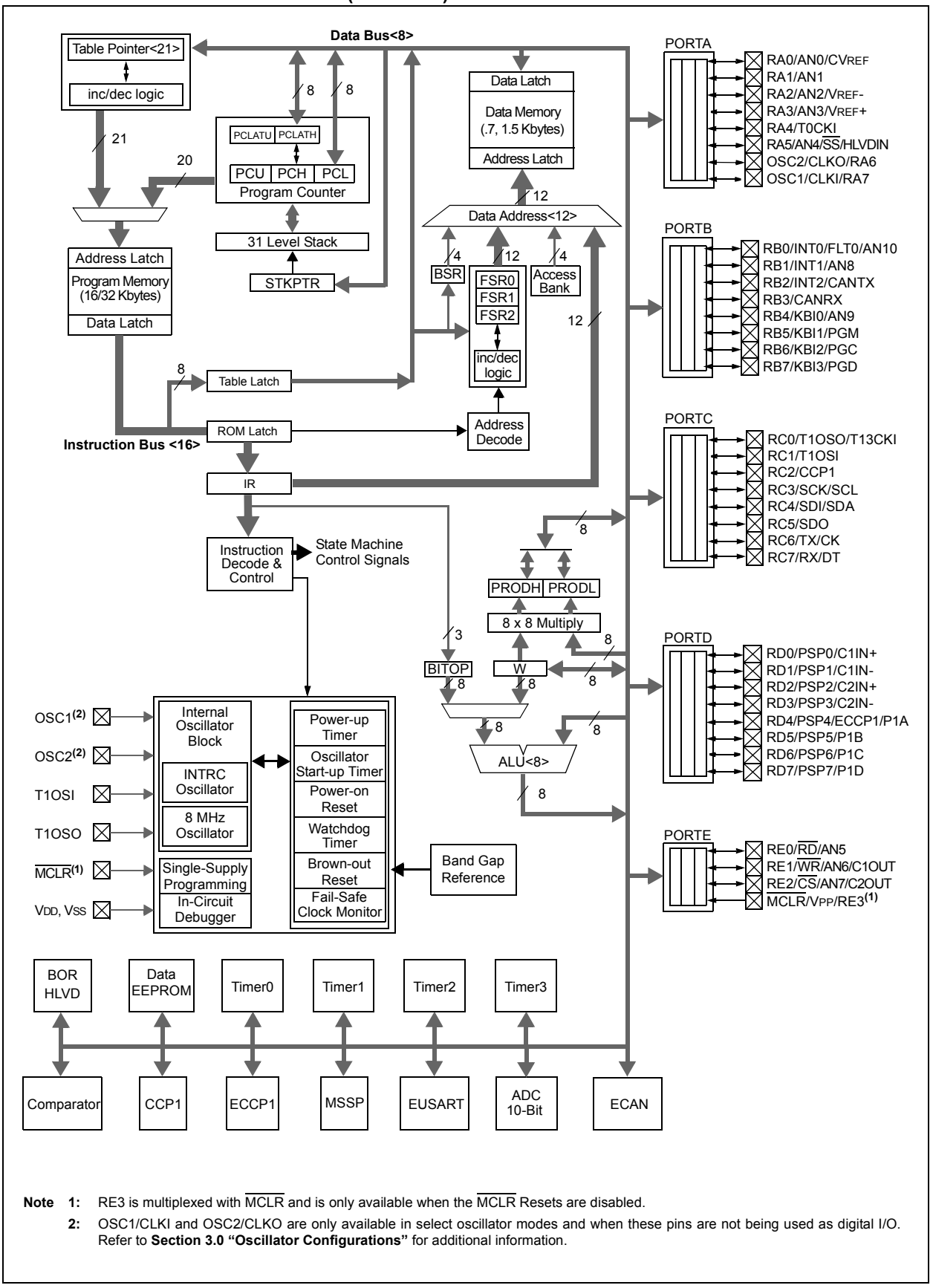



**Note 1:** RE3 is multiplexed with $\overline{MCLR}$ and is only available when the $\overline{MCLR}$ Resets are disabled.

**2:** OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 3.0 "Oscillator Configurations"** for additional information.


© 2009 Microchip Technology Inc.

**FIGURE 1-2: PIC18F4480/4580 (40/44-PIN) BLOCK DIAGRAM**




**Note 1:** RE3 is multiplexed with $\overline{\text{MCLR}}$ and is only available when the $\overline{\text{MCLR}}$ Resets are disabled.

**2:** OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 3.0 "Oscillator Configurations"** for additional information.

**TABLE 1-2: PIC18F2480/2580 PINOUT I/O DESCRIPTIONS**

| Pin Name | Pin Number | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | SPDIP, SOIC | QFN | | | |
| $\overline{MCLR}$/VPP/RE3 | 1 | 26 | | | Master Clear (input) or programming voltage (input). |
| $\overline{MCLR}$ | | | I | ST | Master Clear (Reset) input. This pin is an active-low Reset to the device. |
| VPP | | | P | | Programming voltage input. |
| RE3 | | | I | ST | Digital input. |
| OSC1/CLKI/RA7 | 9 | 6 | | | 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 | TTL | General purpose I/O pin. |
| OSC2/CLKO/RA6 | 10 | 7 | | | Oscillator crystal or clock output. |
| OSC2 | | | O | — | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| CLKO | | | O | — | In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. |
| RA6 | | | I/O | TTL | General purpose I/O pin. |

**Legend:** TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
$I^2C$ = $I^2C$™/SMBus input buffer


© 2009 Microchip Technology Inc.

**TABLE 1-2: PIC18F2480/2580 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | SPDIP, SOIC | QFN | | | |
| | | | | | PORTA is a bidirectional I/O port. |
| RA0/AN0 | 2 | 27 | | | |
| RA0 | | | I/O | TTL | Digital I/O. |
| AN0 | | | I | Analog | Analog Input 0. |
| RA1/AN1 | 3 | 28 | | | |
| RA1 | | | I/O | TTL | Digital I/O. |
| AN1 | | | I | Analog | Analog Input 1. |
| RA2/AN2/VREF- | 4 | 1 | | | |
| RA2 | | | I/O | TTL | Digital I/O. |
| AN2 | | | I | Analog | Analog Input 2. |
| VREF- | | | I | Analog | A/D reference voltage (low) input. |
| RA3/AN3/VREF+ | 5 | 2 | | | |
| RA3 | | | I/O | TTL | Digital I/O. |
| AN3 | | | I | Analog | Analog Input 3. |
| VREF+ | | | I | Analog | A/D reference voltage (high) input. |
| RA4/T0CKI | 6 | 3 | | | |
| RA4 | | | I/O | TTL | Digital I/O. |
| T0CKI | | | I | ST | Timer0 external clock input. |
| RA5/AN4/$\overline{SS}$/ HLVDIN | 7 | 4 | | | |
| RA5 | | | I/O | TTL | Digital I/O. |
| AN4 | | | I | Analog | Analog Input 4. |
| $\overline{SS}$ | | | I | TTL | SPI slave select input. |
| HLVDIN | | | I | Analog | High/Low-Voltage Detect input. |
| RA6 | | | | | See the OSC2/CLKO/RA6 pin. |
| RA7 | | | | | See the OSC1/CLKI/RA7 pin. |

**Legend:** TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
$I^2C$ = $I^2C$™/SMBus input buffer

**TABLE 1-2: PIC18F2480/2580 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | SPDIP, SOIC | QFN | | | |
| | | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/INT0/ AN10 | 21 | 18 | | | |
| RB0 | | | I/O | TTL | Digital I/O. |
| INT0 | | | I | ST | External Interrupt 0. |
| AN10 | | | I | Analog | Analog Input 10. |
| RB1/INT1/AN8 | 22 | 19 | | | |
| RB1 | | | I/O | TTL | Digital I/O. |
| INT1 | | | I | ST | External Interrupt 1. |
| AN8 | | | I | Analog | Analog Input 8. |
| RB2/INT2/CANTX | 23 | 20 | | | |
| RB2 | | | I/O | TTL | Digital I/O. |
| INT2 | | | I | ST | External Interrupt 2. |
| CANTX | | | O | TTL | CAN bus TX. |
| RB3/CANRX | 24 | 21 | | | |
| RB3 | | | I/O | TTL | Digital I/O. |
| CANRX | | | I | TTL | CAN bus RX. |
| RB4/KBI0/AN9 | 25 | 22 | | | |
| RB4 | | | I/O | TTL | Digital I/O. |
| KBI0 | | | I | TTL | Interrupt-on-change pin. |
| AN9 | | | I | Analog | Analog Input 9. |
| RB5/KBI1/PGM | 26 | 23 | | | |
| RB5 | | | I/O | TTL | Digital I/O. |
| KBI1 | | | I | TTL | Interrupt-on-change pin. |
| PGM | | | I/O | ST | Low-Voltage ICSP™ Programming enable pin. |
| RB6/KBI2/PGC | 27 | 24 | | | |
| RB6 | | | I/O | TTL | Digital I/O. |
| KBI2 | | | I | TTL | Interrupt-on-change pin. |
| PGC | | | I/O | ST | In-Circuit Debugger and ICSP programming clock pin. |
| RB7/KBI3/PGD | 28 | 25 | | | |
| RB7 | | | I/O | TTL | Digital I/O. |
| KBI3 | | | I | TTL | Interrupt-on-change pin. |
| PGD | | | I/O | ST | In-Circuit Debugger and ICSP programming data pin. |

**Legend:** TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
$I^2C$ = $I^2C$™/SMBus input buffer


© 2009 Microchip Technology Inc.

**TABLE 1-2: PIC18F2480/2580 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | SPDIP, SOIC | QFN | | | |
| | | | | | PORTC is a bidirectional I/O port. |
| RC0/T1OSO/T13CKI | 11 | 8 | | | |
| RC0 | | | I/O | ST | Digital I/O. |
| T1OSO | | | O | — | Timer1 oscillator output. |
| T13CKI | | | I | ST | Timer1/Timer3 external clock input. |
| RC1/T1OSI | 12 | 9 | | | |
| RC1 | | | I/O | ST | Digital I/O. |
| T1OSI | | | I | CMOS | Timer1 oscillator input. |
| RC2/CCP1 | 13 | 10 | | | |
| RC2 | | | I/O | ST | Digital I/O. |
| CCP1 | | | I/O | ST | Capture 1 input/Compare 1 output/PWM1 output. |
| RC3/SCK/SCL | 14 | 11 | | | |
| RC3 | | | I/O | ST | Digital I/O. |
| SCK | | | I/O | ST | Synchronous serial clock input/output for SPI mode. |
| SCL | | | I/O | $I^2C$ | Synchronous serial clock input/output for $I^2C$™ mode. |
| RC4/SDI/SDA | 15 | 12 | | | |
| RC4 | | | I/O | ST | Digital I/O. |
| SDI | | | I | ST | SPI data in. |
| SDA | | | I/O | $I^2C$ | $I^2C$ data I/O. |
| RC5/SDO | 16 | 13 | | | |
| RC5 | | | I/O | ST | Digital I/O. |
| SDO | | | O | — | SPI data out. |
| RC6/TX/CK | 17 | 14 | | | |
| RC6 | | | I/O | ST | Digital I/O. |
| TX | | | O | — | EUSART asynchronous transmit. |
| CK | | | I/O | ST | EUSART synchronous clock (see related RX/DT). |
| RC7/RX/DT | 18 | 15 | | | |
| RC7 | | | I/O | ST | Digital I/O. |
| RX | | | I | ST | EUSART asynchronous receive. |
| DT | | | I/O | ST | EUSART synchronous data (see related TX/CK). |
| RE3 | — | — | — | — | See $\overline{MCLR}$/$V_{PP}$/RE3 pin. |
| $V_{SS}$ | 8, 19 | 5, 16 | P | — | Ground reference for logic and I/O pins. |
| $V_{DD}$ | 20 | 17 | P | — | Positive supply for logic and I/O pins. |

**Legend:** TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
$I^2C$ = $I^2C$™/SMBus input buffer

**TABLE 1-3: PIC18F4480/4580 PINOUT I/O DESCRIPTIONS**

| Pin Name | Pin Number | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | PDIP | QFN | TQFP | | | |
| $\overline{MCLR}$/VPP/RE3 | 1 | 18 | 18 | | | Master Clear (input) or programming voltage (input). |
| $\overline{MCLR}$ | | | | I | ST | Master Clear (Reset) input. This pin is an active-low Reset to the device. |
| VPP | | | | P | | Programming voltage input. |
| RE3 | | | | I | ST | Digital input. |
| OSC1/CLKI/RA7 | 13 | 32 | 30 | | | 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 | TTL | General purpose I/O pin. |
| OSC2/CLKO/RA6 | 14 | 33 | 31 | | | Oscillator crystal or clock output. |
| OSC2 | | | | O | — | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| CLKO | | | | O | — | In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. |
| RA6 | | | | I/O | TTL | General purpose I/O pin. |

**Legend:** TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
$I^2C$ = $I^2C$™/SMBus input buffer


© 2009 Microchip Technology Inc.

**TABLE 1-3: PIC18F4480/4580 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | PDIP | QFN | TQFP | | | |
| | | | | | | PORTA is a bidirectional I/O port. |
| RA0/AN0/CVREF | 2 | 19 | 19 | | | |
| RA0 | | | | I/O | TTL | Digital I/O. |
| AN0 | | | | I | Analog | Analog Input 0. |
| CVREF | | | | O | Analog | Analog comparator reference output. |
| RA1/AN1 | 3 | 20 | 20 | | | |
| RA1 | | | | I/O | TTL | Digital I/O. |
| AN1 | | | | I | Analog | Analog Input 1. |
| RA2/AN2/VREF- | 4 | 21 | 21 | | | |
| RA2 | | | | I/O | TTL | Digital I/O. |
| AN2 | | | | I | Analog | Analog Input 2. |
| VREF- | | | | I | Analog | A/D reference voltage (low) input. |
| RA3/AN3/VREF+ | 5 | 22 | 22 | | | |
| RA3 | | | | I/O | TTL | Digital I/O. |
| AN3 | | | | I | Analog | Analog Input 3. |
| VREF+ | | | | I | Analog | A/D reference voltage (high) input. |
| RA4/T0CKI | 6 | 23 | 23 | | | |
| RA4 | | | | I/O | TTL | Digital I/O. |
| T0CKI | | | | I | ST | Timer0 external clock input. |
| RA5/AN4/$\overline{SS}$/ HLVDIN | 7 | 24 | 24 | | | |
| RA5 | | | | I/O | TTL | Digital I/O. |
| AN4 | | | | I | Analog | Analog Input 4. |
| $\overline{SS}$ | | | | I | TTL | SPI slave select input. |
| HLVDIN | | | | I | Analog | High/Low-Voltage Detect input. |
| RA6 | | | | | | See the OSC2/CLKO/RA6 pin. |
| RA7 | | | | | | See the OSC1/CLKI/RA7 pin. |

**Legend:** TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
$I^2C$ = $I^2C$™/SMBus input buffer

**TABLE 1-3: PIC18F4480/4580 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | PDIP | QFN | TQFP | | | |
| | | | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/INT0/FLT0/ AN10 | 33 | 9 | 8 | | | |
| RB0 | | | | I/O | TTL | Digital I/O. |
| INT0 | | | | I | ST | External Interrupt 0. |
| FLT0 | | | | I | ST | Enhanced PWM Fault input (ECCP1 module). |
| AN10 | | | | I | Analog | Analog input 10. |
| RB1/INT1/AN8 | 34 | 10 | 9 | | | |
| RB1 | | | | I/O | TTL | Digital I/O. |
| INT1 | | | | I | ST | External Interrupt 1. |
| AN8 | | | | I | Analog | Analog input 8. |
| RB2/INT2/CANTX | 35 | 11 | 10 | | | |
| RB2 | | | | I/O | TTL | Digital I/O. |
| INT2 | | | | I | ST | External Interrupt 2. |
| CANTX | | | | O | TTL | CAN bus TX. |
| RB3/CANRX | 36 | 12 | 11 | | | |
| RB3 | | | | I/O | TTL | Digital I/O. |
| CANRX | | | | I | TTL | CAN bus RX. |
| RB4/KBI0/AN9 | 37 | 14 | 14 | | | |
| RB4 | | | | I/O | TTL | Digital I/O. |
| KBI0 | | | | I | TTL | Interrupt-on-change pin. |
| AN9 | | | | I | Analog | Analog Input 9. |
| RB5/KBI1/PGM | 38 | 15 | 15 | | | |
| RB5 | | | | I/O | TTL | Digital I/O. |
| KBI1 | | | | I | TTL | Interrupt-on-change pin. |
| PGM | | | | I/O | ST | Low-Voltage ICSP™ Programming enable pin. |
| RB6/KBI2/PGC | 39 | 16 | 16 | | | |
| RB6 | | | | I/O | TTL | Digital I/O. |
| KBI2 | | | | I | TTL | Interrupt-on-change pin. |
| PGC | | | | I/O | ST | In-Circuit Debugger and ICSP programming clock pin. |
| RB7/KBI3/PGD | 40 | 17 | 17 | | | |
| RB7 | | | | I/O | TTL | Digital I/O. |
| KBI3 | | | | I | TTL | Interrupt-on-change pin. |
| PGD | | | | I/O | ST | In-Circuit Debugger and ICSP programming data pin. |

**Legend:** TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
O = Output
$I^2C$ = $I^2C$™/SMBus input buffer
CMOS = CMOS compatible input or output
I = Input
P = Power


© 2009 Microchip Technology Inc.

**TABLE 1-3: PIC18F4480/4580 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | PDIP | QFN | TQFP | | | |
| | | | | | | PORTC is a bidirectional I/O port. |
| RC0/T1OSO/T13CKI | 15 | 34 | 32 | | | |
| RC0 | | | | I/O | ST | Digital I/O. |
| T1OSO | | | | O | — | Timer1 oscillator output. |
| T13CKI | | | | I | ST | Timer1/Timer3 external clock input. |
| RC1/T1OSI | 16 | 35 | 35 | | | |
| RC1 | | | | I/O | ST | Digital I/O. |
| T1OSI | | | | I | CMOS | Timer1 oscillator input. |
| RC2/CCP1 | 17 | 36 | 36 | | | |
| RC2 | | | | I/O | ST | Digital I/O. |
| CCP1 | | | | I/O | ST | Capture 1 input/Compare 1 output/PWM1 output. |
| RC3/SCK/SCL | 18 | 37 | 37 | | | |
| RC3 | | | | I/O | ST | Digital I/O. |
| SCK | | | | I/O | ST | Synchronous serial clock input/output for SPI mode. |
| SCL | | | | I/O | $I^2C$ | Synchronous serial clock input/output for $I^2C$™ mode. |
| RC4/SDI/SDA | 23 | 42 | 42 | | | |
| RC4 | | | | I/O | ST | Digital I/O. |
| SDI | | | | I | ST | SPI data in. |
| SDA | | | | I/O | $I^2C$ | $I^2C$ data I/O. |
| RC5/SDO | 24 | 43 | 43 | | | |
| RC5 | | | | I/O | ST | Digital I/O. |
| SDO | | | | O | — | SPI data out. |
| RC6/TX/CK | 25 | 44 | 44 | | | |
| RC6 | | | | I/O | ST | Digital I/O. |
| TX | | | | O | — | EUSART asynchronous transmit. |
| CK | | | | I/O | ST | EUSART synchronous clock (see related RX/DT). |
| RC7/RX/DT | 26 | 1 | 1 | | | |
| RC7 | | | | I/O | ST | Digital I/O. |
| RX | | | | I | ST | EUSART asynchronous receive. |
| DT | | | | I/O | ST | EUSART synchronous data (see related TX/CK). |

**Legend:** TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
$I^2C$ = $I^2C$™/SMBus input buffer

**TABLE 1-3: PIC18F4480/4580 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | PDIP | QFN | TQFP | | | |
| | | | | | | PORTD is a bidirectional I/O port or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when the PSP module is enabled. |
| RD0/PSP0/C1IN+ | 19 | 38 | 38 | | | |
| RD0 | | | | I/O | ST | Digital I/O. |
| PSP0 | | | | I/O | TTL | Parallel Slave Port data. |
| C1IN+ | | | | I | Analog | Comparator 1 input (+). |
| RD1/PSP1/C1IN- | 20 | 39 | 39 | | | |
| RD1 | | | | I/O | ST | Digital I/O. |
| PSP1 | | | | I/O | TTL | Parallel Slave Port data. |
| C1IN- | | | | I | Analog | Comparator 1 input (-) |
| RD2/PSP2/C2IN+ | 21 | 40 | 40 | | | |
| RD2 | | | | I/O | ST | Digital I/O. |
| PSP2 | | | | I/O | TTL | Parallel Slave Port data. |
| C2IN+ | | | | I | Analog | Comparator 2 input (+). |
| RD3/PSP3/C2IN- | 22 | 41 | 41 | | | |
| RD3 | | | | I/O | ST | Digital I/O. |
| PSP3 | | | | I/O | TTL | Parallel Slave Port data. |
| C2IN- | | | | I | Analog | Comparator 2 input (-). |
| RD4/PSP4/ECCP1/ P1A | 27 | 2 | 2 | | | |
| RD4 | | | | I/O | ST | Digital I/O. |
| PSP4 | | | | I/O | TTL | Parallel Slave Port data. |
| ECCP1 | | | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P1A | | | | O | TTL | ECCP1 PWM Output A. |
| RD5/PSP5/P1B | 28 | 3 | 3 | | | |
| RD5 | | | | I/O | ST | Digital I/O. |
| PSP5 | | | | I/O | TTL | Parallel Slave Port data. |
| P1B | | | | O | TTL | ECCP1 PWM Output B. |
| RD6/PSP6/P1C | 29 | 4 | 4 | | | |
| RD6 | | | | I/O | ST | Digital I/O. |
| PSP6 | | | | I/O | TTL | Parallel Slave Port data. |
| P1C | | | | O | TTL | ECCP1 PWM Output C. |
| RD7/PSP7/P1D | 30 | 5 | 5 | | | |
| RD7 | | | | I/O | ST | Digital I/O. |
| PSP7 | | | | I/O | TTL | Parallel Slave Port data. |
| P1D | | | | O | TTL | ECCP1 PWM Output D. |

**Legend:** TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power
$I^2C$ = $I^2C$™/SMBus input buffer


© 2009 Microchip Technology Inc.

**TABLE 1-3: PIC18F4480/4580 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | PDIP | QFN | TQFP | | | |
| | | | | | | PORTE is a bidirectional I/O port. |
| RE0/$\overline{RD}$/AN5 | 8 | 25 | 25 | | | |
| RE0 | | | | I/O | ST | Digital I/O. |
| $\overline{RD}$ | | | | I | TTL | Read control for Parallel Slave Port (see also $\overline{WR}$ and $\overline{CS}$ pins). |
| AN5 | | | | I | Analog | Analog Input 5. |
| RE1/$\overline{WR}$/AN6/C1OUT | 9 | 26 | 26 | | | |
| RE1 | | | | I/O | ST | Digital I/O. |
| $\overline{WR}$ | | | | I | TTL | Write control for Parallel Slave Port (see $\overline{CS}$ and $\overline{RD}$ pins). |
| AN6 | | | | I | Analog | Analog Input 6. |
| C1OUT | | | | O | TTL | Comparator 1 output. |
| RE2/$\overline{CS}$/AN7/C2OUT | 10 | 27 | 27 | | | |
| RE2 | | | | I/O | ST | Digital I/O. |
| $\overline{CS}$ | | | | I | TTL | Chip select control for Parallel Slave Port (see related $\overline{RD}$ and $\overline{WR}$). |
| AN7 | | | | I | Analog | Analog Input 7. |
| C2OUT | | | | O | TTL | Comparator 2 output. |
| RE3 | — | — | — | — | — | See $\overline{MCLR}$/VPP/RE3 pin. |
| VSS | 12, 31 | 6, 30, 31 | 6, 29 | P | — | Ground reference for logic and I/O pins. |
| VDD | 11, 32 | 7, 8, 28, 29 | 7, 28 | P | — | Positive supply for logic and I/O pins. |
| NC | — | 13 | 12, 13, 33, 34 | — | — | No connect. |

**Legend:** TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power; $I^2C$ = $I^2C$™/SMBus input buffer

**NOTES:**


© 2009 Microchip Technology Inc.

## 2.0 GUIDELINES FOR GETTING STARTED WITH PIC18F MICROCONTROLLERS

### 2.1 Basic Connection Requirements

Getting started with the PIC18F2480/2580/4480/4580 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All VDD and VSS pins
  (see **Section 2.2 "Power Supply Pins"**)
- All AVDD and AVSS pins, regardless of whether or not the analog device features are used
  (see **Section 2.2 "Power Supply Pins"**)
- $\overline{\text{MCLR}}$ pin
  (see **Section 2.3 "Master Clear (MCLR) Pin"**)

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

- PGC/PGD pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes
  (see **Section 2.4 "ICSP Pins"**)
- OSCI and OSCO pins when an external oscillator source is used
  (see **Section 2.5 "External Oscillator Pins"**)

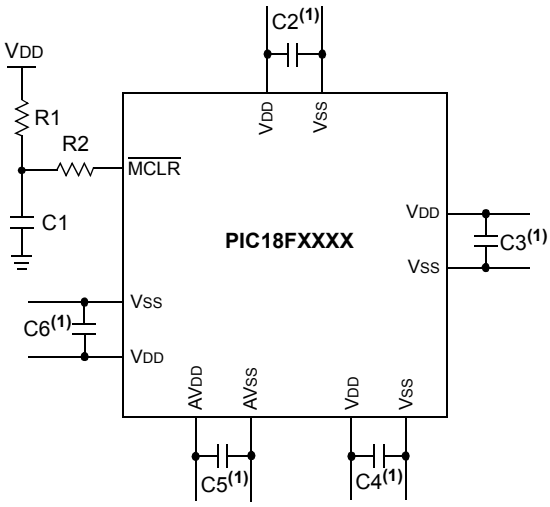
Additionally, the following pins may be required:

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

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

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

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




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

C1 through C6: 0.1 µF, 20V ceramic

R1: 10 kΩ

R2: 100Ω to 470Ω

**Note 1:** The example shown is for a PIC18F device with five VDD/VSS and AVDD/AVSS pairs. Other devices may have more or less pairs; adjust the number of decoupling capacitors appropriately.

## 2.2 Power Supply Pins

### 2.2.1 DECOUPLING CAPACITORS

The use of decoupling capacitors on every pair of power supply pins, such as VDD, VSS, AVDD and AVSS, is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** A 0.1 μF (100 nF), 10-20V capacitor is recommended. The capacitor should be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the printed circuit board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of 0.01 μF to 0.001 μF. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1 μF in parallel with 0.001 μF).
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

### 2.2.2 TANK CAPACITORS

On boards with power traces running longer than six inches in length, it is suggested to use a tank capacitor for integrated circuits, including microcontrollers, to supply a local power source. The value of the tank capacitor should be determined based on the trace resistance that connects the power supply source to the device, and the maximum current drawn by the device in the application. In other words, select the tank capacitor so that it meets the acceptable voltage sag at the device. Typical values range from 4.7 μF to 47 μF.

### 2.2.3 CONSIDERATIONS WHEN USING BOR

When the Brown-out Reset (BOR) feature is enabled, a sudden change in VDD may result in a spontaneous BOR event. This can happen when the microcontroller is operating under normal operating conditions, regardless of what the BOR set point has been programmed to, and even if VDD does not approach the set point. The precipitating factor in these BOR events is a rise or fall in VDD with a slew rate faster than 0.15V/μs.

An application that incorporates adequate decoupling between the power supplies will not experience such rapid voltage changes. Additionally, the use of an electrolytic tank capacitor across VDD and VSS, as described above, will be helpful in preventing high slew rate transitions.

If the application has components that turn on or off, and share the same VDD circuit as the microcontroller, the BOR can be disabled in software by using the SBOREN bit before switching the component. Afterwards, allow a small delay before re-enabling the BOR. By doing this, it is ensured that the BOR is disabled during the interval that might cause high slew rate changes of VDD.

| | |
|---|---|
| **Note:** | Not all devices incorporate software BOR control. See **Section 5.0 "Reset"** for device-specific information. |


© 2009 Microchip Technology Inc.

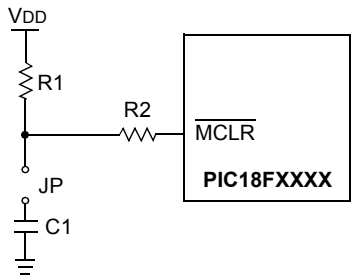
## 2.3 Master Clear ($\overline{MCLR}$) Pin

The $\overline{MCLR}$ pin provides two specific device functions: Device Reset, and Device Programming and Debugging. If programming and debugging are not required in the end application, a direct connection to VDD may be all that is required. The addition of other components, to help increase the application's resistance to spurious Resets from voltage sags, may be beneficial. A typical configuration is shown in Figure 2-1. Other circuit designs may be implemented, depending on the application's requirements.

During programming and debugging, the resistance and capacitance that can be added to the pin must be considered. Device programmers and debuggers drive the $\overline{MCLR}$ pin. Consequently, specific voltage levels (VIH and VIL) and fast signal transitions must not be adversely affected. Therefore, specific values of R1 and C1 will need to be adjusted based on the application and PCB requirements. For example, it is recommended that the capacitor, C1, be isolated from the $\overline{MCLR}$ pin during programming and debugging operations by using a jumper (Figure 2-2). The jumper is replaced for normal run-time operations.

Any components associated with the $\overline{MCLR}$ pin should be placed within 0.25 inch (6 mm) of the pin.

**FIGURE 2-2: EXAMPLE OF $\overline{MCLR}$ PIN CONNECTIONS**




**Note 1:** R1 ≤ 10 kΩ is recommended. A suggested starting value is 10 kΩ. Ensure that the $\overline{MCLR}$ pin VIH and VIL specifications are met.

**2:** R2 ≤ 470Ω will limit any current flowing into $\overline{MCLR}$ from the external capacitor, C, in the event of $\overline{MCLR}$ pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS). Ensure that the $\overline{MCLR}$ pin VIH and VIL specifications are met.

## 2.4 ICSP Pins

The PGC and PGD pins are used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor is recommended, with the value in the range of a few tens of ohms, not to exceed 100Ω.

Pull-up resistors, series diodes, and capacitors on the PGC and PGD pins are not recommended as they will interfere with the programmer/debugger communications to the device. If such discrete components are an application requirement, they should be removed from the circuit during programming and debugging. Alternatively, refer to the AC/DC characteristics and timing requirements information in the respective device Flash programming specification for information on capacitive loading limits and pin input voltage high (VIH) and input low (VIL) requirements.

For device emulation, ensure that the "Communication Channel Select" (i.e., PGCx/PGDx pins) programmed into the device matches the physical connections for the ICSP to the Microchip debugger/emulator tool.

For more information on available Microchip development tools connection requirements, refer to **Section 27.0 "Development Support"**.

## 2.5 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to **Section 3.0 "Oscillator Configurations"** for details).

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

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

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

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

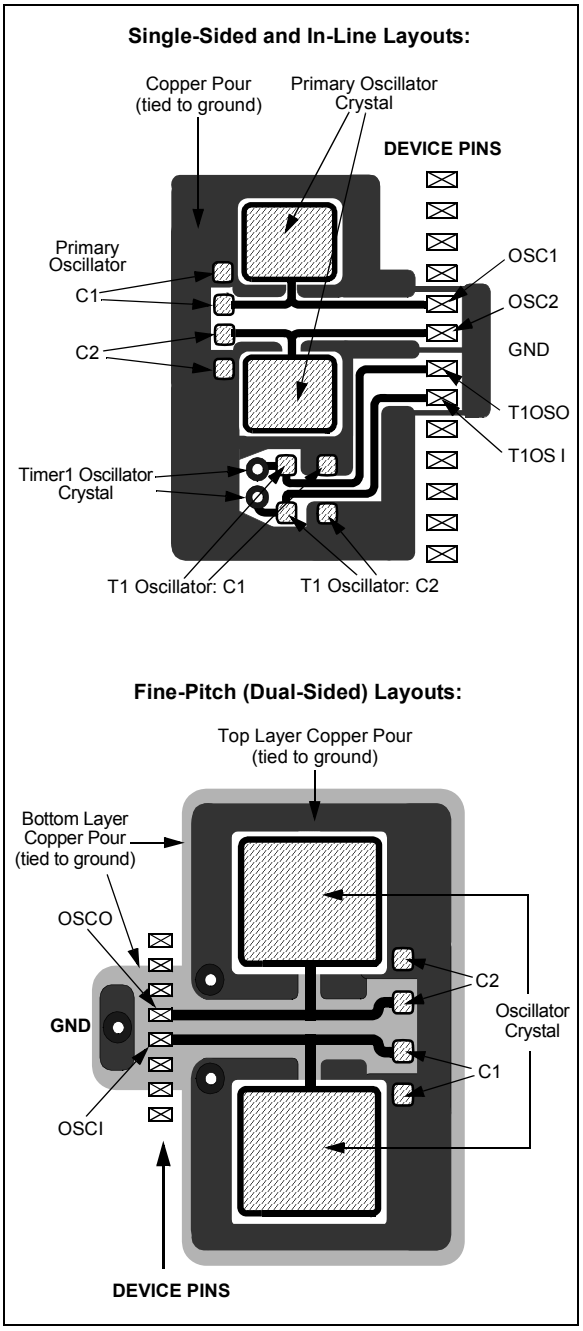
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate web site (www.microchip.com):

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

## 2.6 Unused I/Os

Unused I/O pins should be configured as outputs and driven to a logic low state. Alternatively, connect a 1 kΩ to 10 kΩ resistor to VSS on unused pins and drive the output to logic low.

**FIGURE 2-3: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT**





© 2009 Microchip Technology Inc.

# 3.0 OSCILLATOR CONFIGURATIONS

## 3.1 Oscillator Types

PIC18F2480/2580/4480/4580 devices can be operated in ten different oscillator modes. The user can program the Configuration bits, FOSC<3:0>, 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

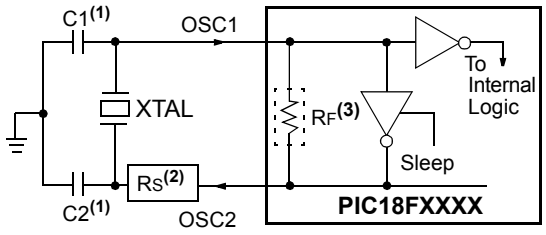
## 3.2 Crystal Oscillator/Ceramic Resonators

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 3-1 shows the pin connections.

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

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

**FIGURE 3-1: CRYSTAL/CERAMIC RESONATOR OPERATION (XT, LP, HS OR HSPLL CONFIGURATION)**




**Note 1:** See Table 3-1 and Table 3-2 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 3-1: 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 on page 30 for additional information.

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

| | |
|---|---|
| **Note:** | When using resonators with frequencies above 3.5 MHz, the use of HS mode, rather than XT mode, is recommended. HS mode may be used at any VDD for which the controller is rated. If HS is selected, it is possible that the gain of the oscillator will overdrive the resonator. Therefore, a series resistor should be placed between the OSC2 pin and the resonator. As a good starting point, the recommended value of RS is 330Ω. |

**TABLE 3-2: 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 the 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.

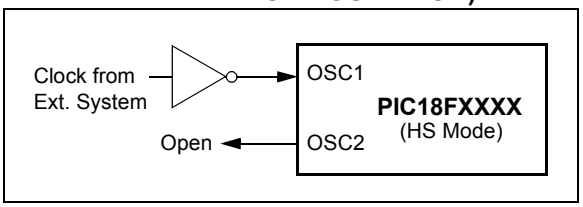
**3:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

**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 3-2.

**FIGURE 3-2: EXTERNAL CLOCK INPUT OPERATION (HS OSCILLATOR CONFIGURATION)**

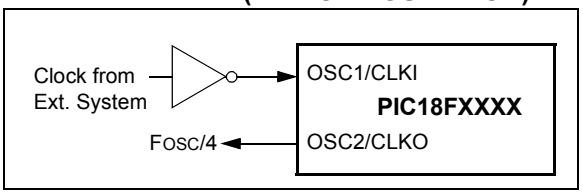



## 3.3 External Clock Input

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.

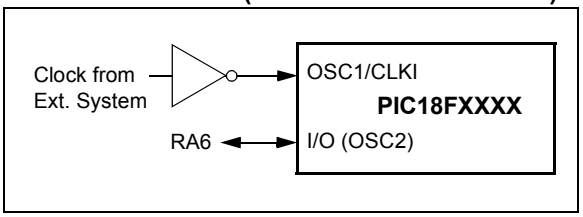
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 3-3 shows the pin connections for the EC Oscillator mode.

**FIGURE 3-3: 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 3-4 shows the pin connections for the ECIO Oscillator mode.

**FIGURE 3-4: EXTERNAL CLOCK INPUT OPERATION (ECIO CONFIGURATION)**





© 2009 Microchip Technology Inc.

## 3.4 RC Oscillator

For timing insensitive applications, the “RC” and “RCIO” device options offer additional cost savings. The actual oscillator frequency is a function of several factors:

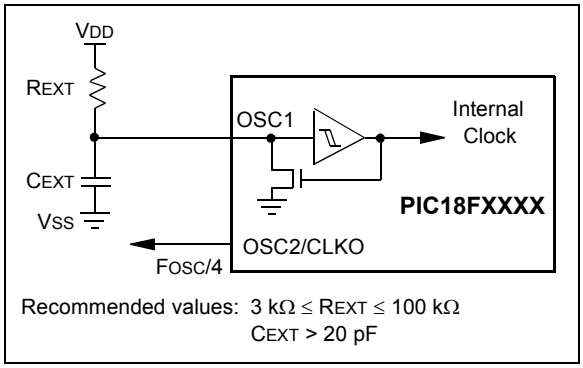
- supply voltage
- values of the external resistor (REXT) and capacitor (CEXT)
- operating temperature

Given the same device, operating voltage and temperature and component values, there will also be unit-to-unit frequency variations. These are due to factors such as:

- normal manufacturing variation
- difference in lead frame capacitance between package types (especially for low CEXT values)
- variations within the tolerance of limits of REXT and CEXT

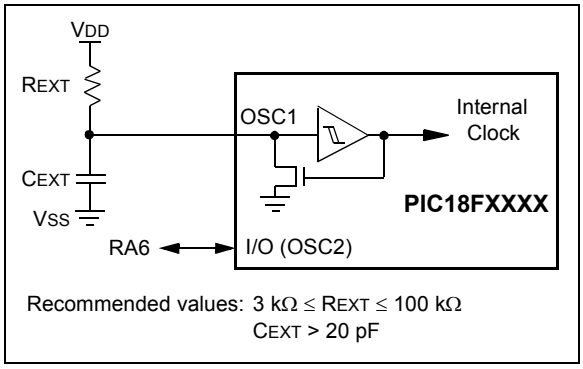
In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 3-5 shows how the R/C combination is connected.

**FIGURE 3-5: RC OSCILLATOR MODE**




The RCIO Oscillator mode (Figure 3-6) 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 3-6: RCIO OSCILLATOR MODE**




## 3.5 PLL Frequency Multiplier

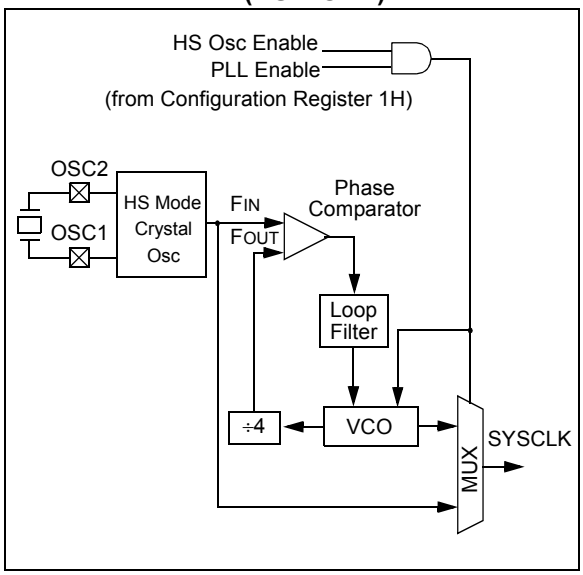
A Phase Locked Loop (PLL) circuit is provided as an option for users who wish to use a lower frequency oscillator circuit or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals or users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

### 3.5.1 HSPLL OSCILLATOR MODE

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 only available to the crystal oscillator when the FOSC<3:0> Configuration bits are programmed for HSPLL mode (= 0110).

**FIGURE 3-7: PLL BLOCK DIAGRAM (HS MODE)**




### 3.5.2 PLL AND INTOSC

The PLL is also available to the internal oscillator block in selected oscillator modes. In this configuration, the PLL is enabled in software and generates a clock output of up to 32 MHz. The operation of INTOSC with the PLL is described in **Section 3.6.4 “PLL in INTOSC Modes”**.

## 3.6 Internal Oscillator Block

The PIC18F2480/2580/4480/4580 devices include an internal oscillator block which generates two different clock signals; either can be used as the microcontroller's clock source. This may 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 device clock. It also drives a postscaler, which can provide a range of clock frequencies from 31 kHz to 4 MHz. The INTOSC output is enabled when a clock frequency from 125 kHz to 8 MHz is selected.

The other clock source is the internal RC oscillator (INTRC), which provides a nominal 31 kHz output. INTRC is enabled if it is selected as the device clock source; it is also enabled automatically 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 25.0 "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 3-2).

### 3.6.1 INTIO MODES

Using the internal oscillator as the clock source eliminates 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.

### 3.6.2 INTOSC OUTPUT FREQUENCY

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 8.0 MHz.

The INTRC oscillator operates independently of the INTOSC source. Any changes in INTOSC across voltage and temperature are not necessarily reflected by changes in INTRC and vice versa.

### 3.6.3 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 3-1).

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). Code execution continues during this shift. There is no indication that the shift has occurred.

The OSCTUNE register also implements the INTSRC and PLLEN bits, which control certain features of the internal oscillator block. The INTSRC bit allows users to select which internal oscillator provides the clock source when the 31 kHz frequency option is selected. This is covered in greater detail in **Section 3.7.1 "Oscillator Control Register"**.

The PLLEN bit controls the operation of the frequency multiplier, PLL, in internal oscillator modes.

### 3.6.4 PLL IN INTOSC MODES

The 4x frequency multiplier can be used with the internal oscillator block to produce faster device clock speeds than are normally possible with an internal oscillator. When enabled, the PLL produces a clock speed of up to 32 MHz.

Unlike HSPLL mode, the PLL is controlled through software. The control bit, PLLEN (OSCTUNE<6>), is used to enable or disable its operation. If PLL is enabled and a Two-Speed Start-up from wake is performed, execution is delayed until the PLL starts.

The PLL is available when the device is configured to use the internal oscillator block as its primary clock source (FOSC<3:0> = 1001 or 1000). Additionally, the PLL will only function when the selected output frequency is either 4 MHz or 8 MHz (OSCCON<6:4> = 111 or 110). If both of these conditions are not met, the PLL is disabled.

The PLLEN control bit is only functional in those internal oscillator modes where the PLL is available. In all other modes, it is forced to '0' and is effectively unavailable.

### 3.6.5 INTOSC FREQUENCY DRIFT

The factory calibrates the internal oscillator block output (INTOSC) for 8 MHz. 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. This has no effect on the INTRC clock source frequency.

Tuning the INTOSC source requires knowing when to make the adjustment, in which direction it should be made, and in some cases, how large a change is needed. Three compensation techniques are discussed in **Section 3.6.5.1 "Compensating with the EUSART"**, **Section 3.6.5.2 "Compensating with the Timers"** and **Section 3.6.5.3 "Compensating with the CCP Module in Capture Mode"**, but other techniques may be used.


© 2009 Microchip Technology Inc.

**REGISTER 3-1: OSCTUNE: OSCILLATOR TUNING REGISTER**

| R/W-0 | R/W-0(1) | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| INTSRC | PLLEN(1) | — | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **INTSRC:** Internal Oscillator Low-Frequency Source Select bit
1 = 31.25 kHz device clock derived from 8 MHz INTOSC source (divide-by-256 enabled)
0 = 31 kHz device clock derived directly from INTRC internal oscillator

bit 6 **PLLEN:** Frequency Multiplier PLL for INTOSC Enable bit(1)
1 = PLL enabled for INTOSC (4 MHz and 8 MHz only)
0 = PLL disabled

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

bit 4-0 **TUN<4:0>:** Frequency Tuning bits
01111 = Maximum frequency
• •
• •
00001
00000 = Center frequency. Oscillator module is running at the calibrated frequency.
11111
• •
• •
10000 = Minimum frequency

**Note 1:** Available only in certain oscillator configurations; otherwise, this bit is unavailable and reads as '0'. See text for details.

#### 3.6.5.1 Compensating with the EUSART

An adjustment may be required when the EUSART begins to generate framing errors or receives data with errors while in Asynchronous mode. Framing errors indicate that the device clock frequency is too high. To adjust for this, decrement the value in OSCTUNE to reduce the clock frequency. On the other hand, errors in data may suggest that the clock speed is too low. To compensate, increment OSCTUNE to increase the clock frequency.

#### 3.6.5.2 Compensating with the Timers

This technique compares device 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. To adjust for this, decrement the OSCTUNE register.

#### 3.6.5.3 Compensating with the CCP Module 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. To compensate, decrement the OSCTUNE register. If the measured time is much less than the calculated time, the internal oscillator block is running too slow. To compensate, increment the OSCTUNE register.

## 3.7 Clock Sources and Oscillator Switching

Like previous PIC18 devices, the PIC18F2480/2580/4480/4580 family includes a feature that allows the device clock source to be switched from the main oscillator to an alternate low-frequency clock source. PIC18F2480/2580/4480/4580 devices offer two alternate clock sources. When an alternate clock source is enabled, the various power-managed operating modes are available.

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 by the FOSC<3:0> Configuration bits. 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.

PIC18F2480/2580/4480/4580 devices offer 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 (RTC).

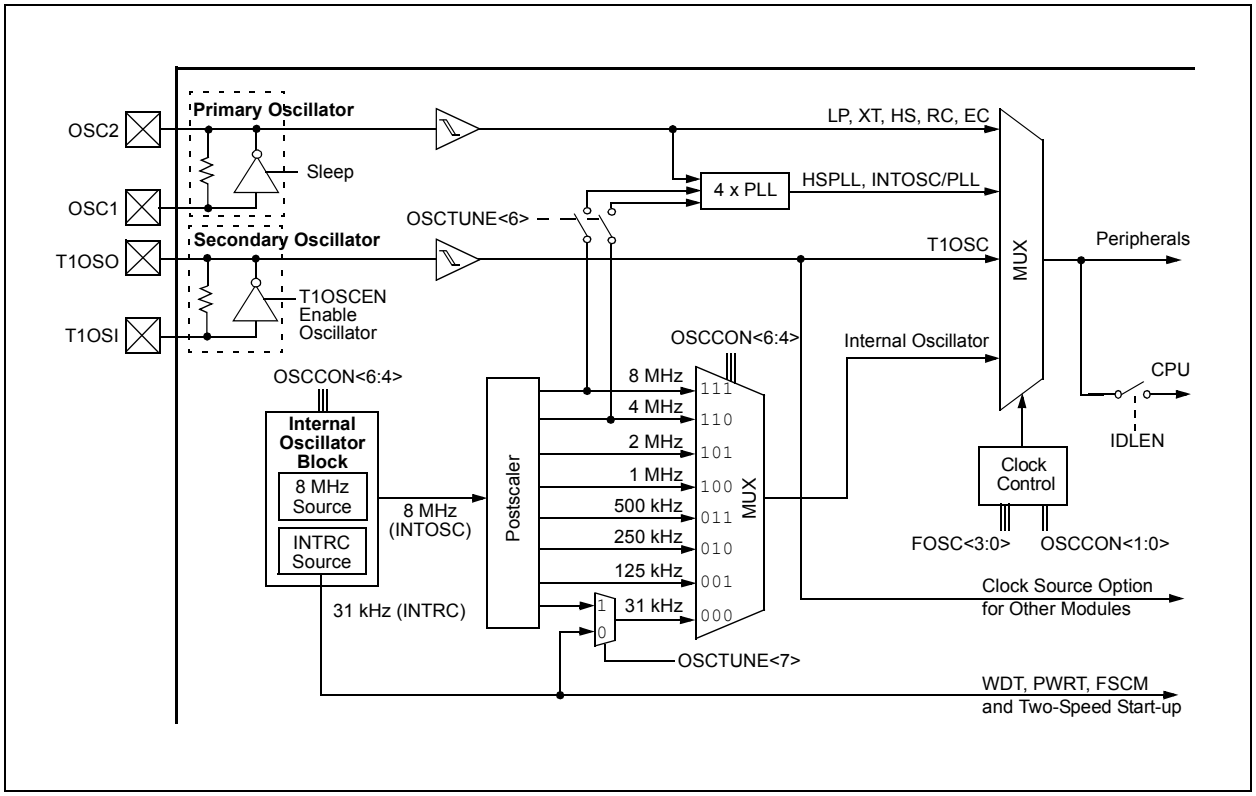
Most often, a 32.768 kHz watch crystal is connected between the RC0/T1OSO/T13CKI and RC1/T1OSI pins. Like the LP Oscillator mode circuit, loading capacitors are also connected from each pin to ground.

The Timer1 oscillator is discussed in greater detail in **Section 13.3 "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 PIC18F2480/2580/4480/4580 devices are shown in Figure 3-8. See **Section 25.0 "Special Features of the CPU"** for Configuration register details.

**FIGURE 3-8: PIC18F2480/2580/4480/4580 CLOCK DIAGRAM**





© 2009 Microchip Technology Inc.

### 3.7.1 OSCILLATOR CONTROL REGISTER

The OSCCON register (Register 3-2) controls several aspects of the device clock's operation, both in full-power operation and in power-managed modes.

The System Clock Select bits, SCS<1:0>, select the clock source. The available clock sources are the primary clock (defined by the FOSC<3:0> Configuration bits), the secondary clock (Timer1 oscillator) and the internal oscillator block. The clock source changes immediately after one or more of the bits is written to, following a brief clock transition interval. The SCS bits are cleared on all forms of Reset.

The Internal Oscillator Frequency Select bits, IRCF<2:0>, select the frequency output of the internal oscillator block to drive the device clock. The choices are the INTRC source, the INTOSC source (8 MHz) or one of the frequencies derived from the INTOSC postscaler (31 kHz to 4 MHz). If the internal oscillator block is supplying the device clock, changing the states of these bits will have an immediate change on the internal oscillator's output. On device Resets, the default output frequency of the internal oscillator block is set at 1 MHz.

When an output frequency of 31 kHz is selected (IRCF<2:0> = 000), users may choose which internal oscillator acts as the source. This is done with the INTSRC bit in the OSCTUNE register (OSCTUNE<7>). Setting this bit selects INTOSC as a 31.25 kHz clock source by enabling the divide-by-256 output of the INTOSC postscaler. Clearing INTSRC selects INTRC (nominally 31 kHz) as the clock source.

This option allows users to select the tunable and more precise INTOSC as a clock source, while maintaining power savings with a very low clock speed. Regardless of the setting of INTSRC, INTRC always remains the clock source for features such as the Watchdog Timer and the Fail-Safe Clock Monitor.

The OSTS, IOFS and T1RUN bits indicate which clock source is currently providing the device clock. The OSTS bit indicates that the Oscillator Start-up Timer (OST) has timed out and the primary clock is providing the device clock in primary clock modes. The IOFS bit indicates when the internal oscillator block has stabilized and is providing the device clock in RC Clock modes. The T1RUN bit (T1CON<6>) indicates when the Timer1 oscillator is providing the device 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 clock or the internal oscillator block has just started and is not yet stable.

The IDLEN bit determines if the device goes into Sleep mode or one of the Idle modes when the SLEEP instruction is executed.

The use of the flag and control bits in the OSCCON register is discussed in more detail in **Section 4.0 "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. |

### 3.7.2 OSCILLATOR TRANSITIONS

PIC18F2480/2580/4480/4580 devices contain circuitry to prevent clock "glitches" when switching between clock sources. A short pause in the device clock occurs during the clock switch. The length of this pause is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Clock transitions are discussed in greater detail in **Section 4.1.2 "Entering Power-Managed Modes"**.

**REGISTER 3-2: OSCCON: OSCILLATOR CONTROL REGISTER**

| R/W-0 | R/W-1 | R/W-0 | R/W-0 | R[(1)] | R-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IDLEN | IRCF2 | IRCF1 | IRCF0 | OSTS | IOFS | SCS1 | SCS0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **IDLEN:** Idle Enable bit
1 = Device enters Idle mode on SLEEP instruction
0 = Device enters Sleep mode on SLEEP instruction

bit 6-4 **IRCF<2:0>:** Internal Oscillator Frequency Select bits
111 = 8 MHz (INTOSC drives clock directly)
110 = 4 MHz
101 = 2 MHz
100 = 1 MHz[(3)]
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz (from either INTOSC/256 or INTRC directly)[(2)]

bit 3 **OSTS:** Oscillator Start-up Timer Time-out Status bit[(1)]
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 and the frequency is provided by one of the RC modes
0 = INTOSC frequency is not stable

bit 1-0 **SCS<1:0>:** System Clock Select bits
1x = Internal oscillator block
01 = Timer1 oscillator
00 = Primary oscillator

**Note 1:** Depends on state of the IESO Configuration bit.
**2:** Source selected by the INTSRC bit (OSCTUNE<7>), see text.
**3:** Default output frequency of INTOSC on Reset.


© 2009 Microchip Technology Inc.

## 3.8 Effects of Power-Managed Modes on the Various Clock Sources

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 device 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 device clock source. The 31 kHz INTRC output can be used directly to provide the clock and may be enabled to support various special features, regardless of the power-managed mode (see **Section 25.2 "Watchdog Timer (WDT)"**, **Section 25.3 "Two-Speed Start-up"** and **Section 25.4 "Fail-Safe Clock Monitor"** for more information on WDT, Two-Speed Start-up and Fail-Safe Clock Monitor. The INTOSC output at 8 MHz may be used directly to clock the device or may be divided down by the postscaler. The INTOSC output is disabled if the clock is provided directly from the INTRC output. The INTOSC output is enabled for Two-Speed Start-up at 1 MHz after a Reset.

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 (RTC). Other features may be operating that do not require a device clock source (i.e., MSSP slave, PSP, INTx pins and others). Peripherals that may add significant current consumption are listed in **Section 28.2 "DC Characteristics: Power Down and Supply Current".**

## 3.9 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 **Section 5.5 "Device Reset Timers"**.

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up (parameter 33, Table 28-10). It is enabled by clearing (= 0) the PWRTEN Configuration bit.

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 interval, TCSD (parameter 38, Table 28-10), 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 3-3: OSC1 AND OSC2 PIN STATES IN SLEEP MODE**

| OSC 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 5-2 in **Section 5.0 "Reset"**, for time-outs due to Sleep and MCLR Reset.

**NOTES:**


© 2009 Microchip Technology Inc.

## 4.0 POWER-MANAGED MODES

PIC18F2480/2580/4480/4580 devices offer a total of seven operating modes for more efficient power management. These modes 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:

- Run modes
- Idle modes
- Sleep mode

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 internal oscillator block); the Sleep mode does not use a clock source.

The power-managed modes include several power-saving features offered on previous PIC® devices. One is the clock switching feature, offered in other PIC18 devices, allowing the controller to use the Timer1 oscillator in place of the primary oscillator. Also included is the Sleep mode, offered by all PIC devices, where all device clocks are stopped.

### 4.1 Selecting Power-Managed Modes

Selecting a power-managed mode requires two decisions: if the CPU is to be clocked or not and the selection of a clock source. The IDLEN bit (OSCCON<7>) controls CPU clocking, while the SCS<1:0> bits (OSCCON<1:0>) select the clock source. The individual modes, bit settings, clock sources and affected modules are summarized in Table 4-1.

#### 4.1.1 CLOCK SOURCES

The SCS<1:0> bits allow the selection of one of three clock sources for power-managed modes. They are:

- The primary clock, as defined by the FOSC<3:0> Configuration bits
- The secondary clock (the Timer1 oscillator)
- The internal oscillator block (for RC modes)

#### 4.1.2 ENTERING POWER-MANAGED MODES

Switching from one power-managed mode to another begins by loading the OSCCON register. The SCS<1:0> bits select the clock source and determine which Run or Idle mode is to be used. Changing these bits causes an immediate switch to the new clock source, assuming that it is running. The switch may also be subject to clock transition delays. These are discussed in **Section 4.1.3 "Clock Transitions and Status Indicators"** and subsequent sections.

Entry to the power-managed Idle or Sleep modes is triggered by the execution of a `SLEEP` instruction. The actual mode that results depends on the status of the IDLEN bit.

Depending on the current mode and the mode being switched to, a change to a power-managed mode does not always require setting all of these bits. Many transitions may be done by changing the oscillator select bits, or changing the IDLEN bit, prior to issuing a `SLEEP` instruction. If the IDLEN bit is already configured correctly, it may only be necessary to perform a `SLEEP` instruction to switch to the desired mode.

**TABLE 4-1: POWER-MANAGED MODES**

| Mode | OSCCON<7,1:0> | | Module Clocking | | Available Clock and Oscillator Source |
|---|---|---|---|---|---|
| | IDLEN(1) | SCS<1:0> | CPU | Peripherals | |
| Sleep | 0 | N/A | Off | Off | None – All clocks are disabled |
| PRI_RUN | N/A | 00 | Clocked | Clocked | Primary – LP, XT, HS, HSPLL, RC, EC, INTRC(2): This is the normal full-power execution mode. |
| SEC_RUN | N/A | 01 | Clocked | Clocked | Secondary – Timer1 Oscillator |
| RC_RUN | N/A | 1x | Clocked | Clocked | Internal Oscillator Block(2) |
| 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(2) |

**Note 1:** IDLEN reflects its value when the `SLEEP` instruction is executed.

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

### 4.1.3 CLOCK TRANSITIONS AND STATUS INDICATORS

The length of the transition between clock sources is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Three bits indicate the current clock source and its status. They are:

- OSTS (OSCCON<3>)
- IOFS (OSCCON<2>)
- T1RUN (T1CON<6>)

In general, only one of these bits will be set while in a given power-managed mode. When the OSTS bit is set, the primary clock is providing the device clock. When the IOFS bit is set, the INTOSC output is providing a stable 8 MHz clock source to a divider that actually drives the device clock. When the T1RUN bit is set, the Timer1 oscillator is providing the clock. If none of these bits are set, then either the INTRC clock source is clocking the device, or the INTOSC source is not yet stable.

If the internal oscillator block is configured as the primary clock source by the FOSC<3:0> Configuration bits, 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 another RC power-managed mode at the 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. It acts as the trigger to place the controller into either the Sleep mode, or one of the Idle modes, depending on the setting of the IDLEN bit. |

### 4.1.4 MULTIPLE SLEEP COMMANDS

The power-managed mode that is invoked with the SLEEP instruction is determined by the setting of the IDLEN bit at the time the instruction is executed. If another SLEEP instruction is executed, the device will enter the power-managed mode specified by IDLEN at that time. If IDLEN has changed, the device will enter the new power-managed mode specified by the new setting.

Upon resuming normal operation after waking form Sleep or Idle, the internal state machines require at least one TCY delay before another SLEEP instruction can be executed. If two back-to-back SLEEP instructions need to be executed, the process shown in Example 4-1 should be used.

**EXAMPLE 4-1: EXECUTING BACK-TO-BACK SLEEP INSTRUCTIONS**

```
SLEEP
NOP     ; Wait at least 1 Tcy before
executing another SLEEP instruction
SLEEP
```

## 4.2 Run Modes

In the Run modes, clocks to both the core and peripherals are active. The difference between these modes is the clock source.

### 4.2.1 PRI_RUN MODE

The PRI_RUN mode is the normal, full-power execution mode of the microcontroller. This is also the default mode upon a device Reset, unless Two-Speed Start-up is enabled (see **Section 25.3 "Two-Speed Start-up"** for details). In this mode, the OSTS bit is set. The IOFS bit may be set if the internal oscillator block is the primary clock source (see **Section 3.7.1 "Oscillator Control Register"**).

### 4.2.2 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 setting the SCS<1:0> bits to '01'. The device clock source is switched to the Timer1 oscillator (see Figure 4-1), 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 SCS<1:0> bits are set to '01', entry to SEC_RUN mode will not occur. If the Timer1 oscillator is enabled but not yet running, device clocks will be delayed until the oscillator has started. In such situations, initial oscillator operation is far from stable and unpredictable operation may result. |


© 2009 Microchip Technology Inc.

On transitions from SEC_RUN mode to PRI_RUN mode, 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 switch back to the primary clock occurs (see Figure 4-2). When the clock switch is complete, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the clock. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run.

**FIGURE 4-1: TRANSITION TIMING FOR ENTRY TO SEC_RUN MODE**

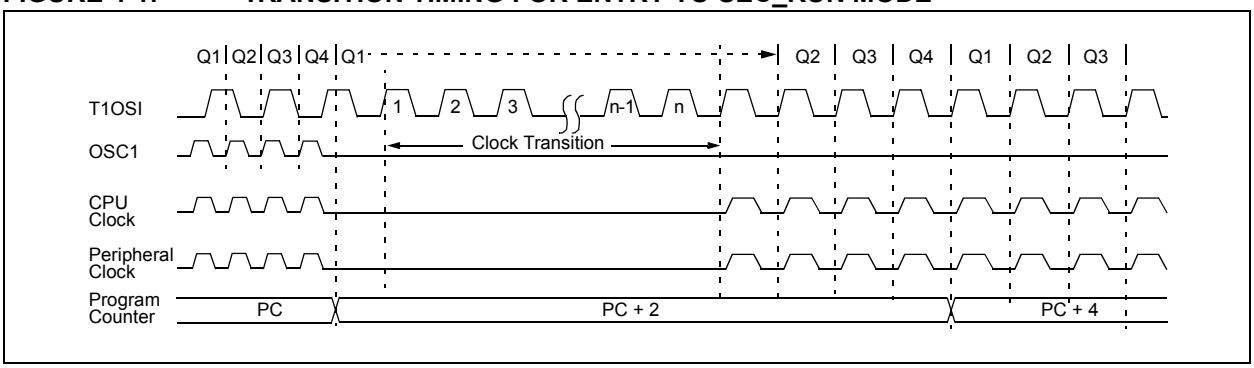
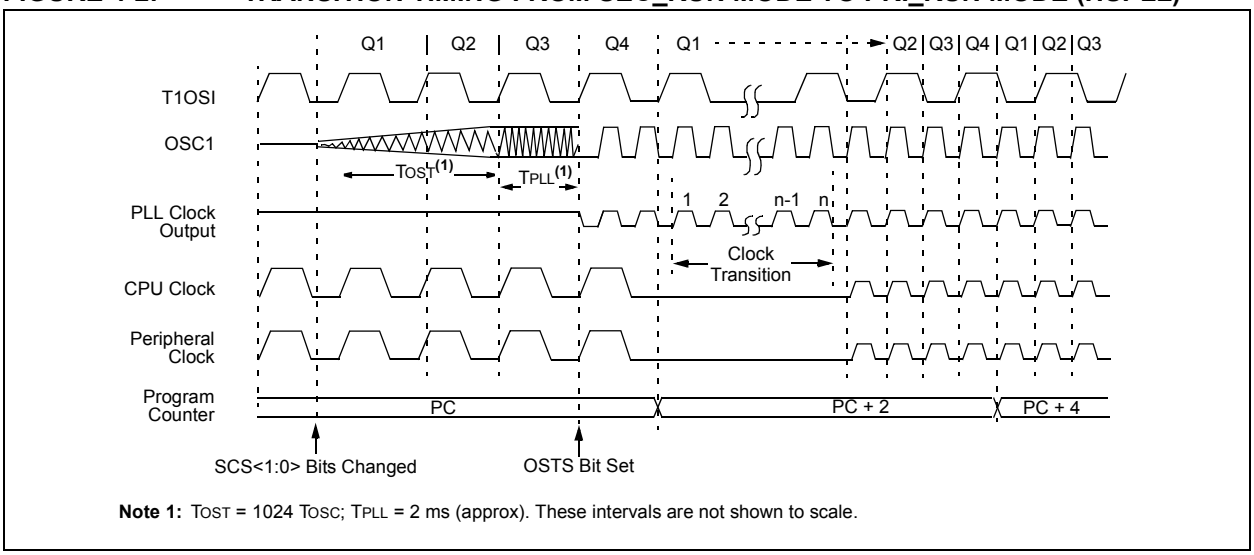



**FIGURE 4-2: TRANSITION TIMING FROM SEC_RUN MODE TO PRI_RUN MODE (HSPLL)**




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

### 4.2.3 RC_RUN MODE

In RC_RUN mode, the CPU and peripherals are clocked from the internal oscillator block using the INTOSC multiplexer; 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 INTRC or INTOSC), 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 setting SCS1 to '1'. Although it is ignored, it is recommended that SCS0 also be cleared; this is to maintain software compatibility with future devices. When the clock source is switched to the INTOSC multiplexer (see Figure 4-3), 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 clock speed.

| **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 and the INTSRC bit 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 device clocks.

If the IRCF bits are changed from all clear (thus, enabling the INTOSC output) or if INTSRC is set, the IOFS bit becomes set after the INTOSC output becomes stable. Clocks to the device continue while the INTOSC source stabilizes after an interval of TIOBST.

If the IRCF bits were previously at a non-zero value or if INTSRC was set before setting SCS1 and the INTOSC source was already stable, the IOFS bit will remain set.

On transitions from RC_RUN mode to PRI_RUN mode, the device 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 4-4). When the clock switch is complete, the IOFS bit is cleared, the OSTS bit is set and the primary clock is providing the device clock. The IDLEN and SCS bits are not affected by the switch. The INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

**FIGURE 4-3: TRANSITION TIMING TO RC_RUN MODE**

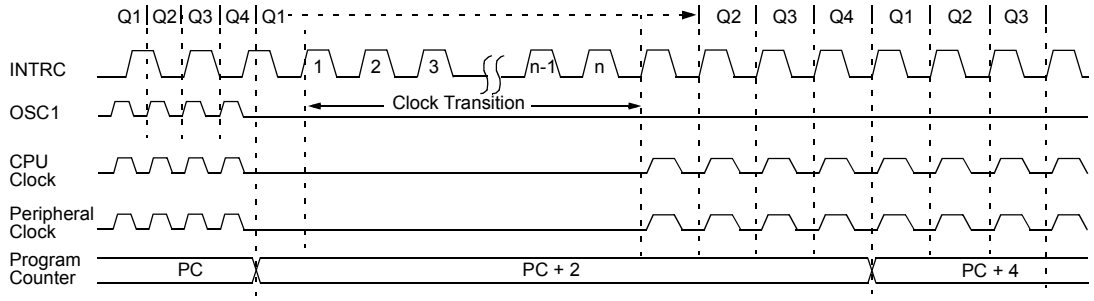
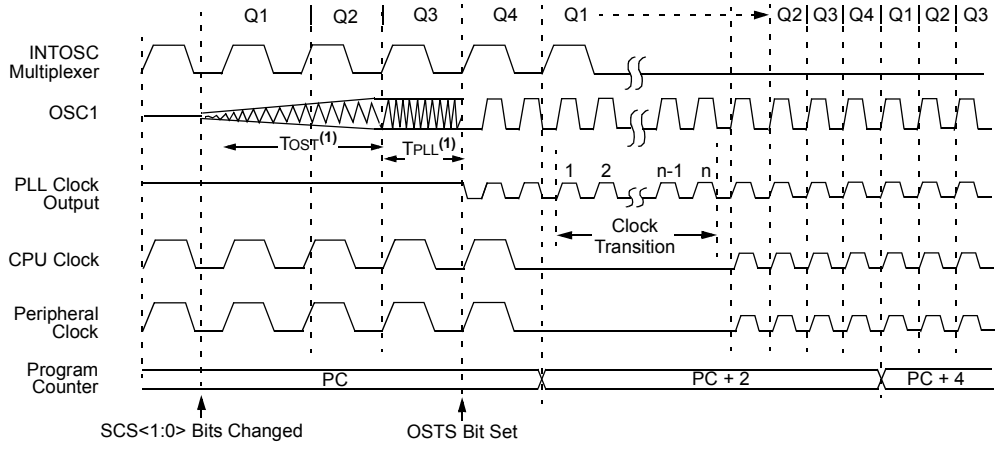



**FIGURE 4-4: TRANSITION TIMING FROM RC_RUN MODE TO PRI_RUN MODE**




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


© 2009 Microchip Technology Inc.

## 4.3 Sleep Mode

The power-managed Sleep mode in the PIC18F2480/2580/4480/4580 devices is identical to the legacy Sleep mode offered in all other PIC devices. It is entered by clearing the IDLEN bit (the default state on device Reset) and executing the SLEEP instruction. This shuts down the selected oscillator (Figure 4-5). All clock source status bits are cleared.

Entering the Sleep mode from any other mode does not require a clock switch. This is because no clocks are needed once the controller has entered Sleep. If the WDT is selected, the INTRC source will continue to operate. If the Timer1 oscillator is enabled, it will also continue to run.

When a wake event occurs in Sleep mode (by interrupt, Reset or WDT time-out), the device will not be clocked until the clock source selected by the SCS1:SCS0 bits becomes ready (see Figure 4-6), 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 25.0 "Special Features of the CPU"**). In either case, the OSTS bit is set when the primary clock is providing the device clocks. The IDLEN and SCS bits are not affected by the wake-up.

## 4.4 Idle Modes

The Idle modes allow the controller's CPU to be selectively shut down while the peripherals continue to operate. Selecting a particular Idle mode allows users to further manage power consumption.

If the IDLEN bit is set to '1' when a SLEEP instruction is executed, the peripherals will be clocked from the clock source selected using the SCS<1:0> bits; however, the CPU will not be clocked. The clock source status bits are not affected. Setting IDLEN and executing a SLEEP instruction provides a quick method of switching from a given Run mode to its corresponding Idle mode.

If the WDT is selected, the INTRC source will continue to operate. If the Timer1 oscillator is enabled, it will also continue to run.

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 by an interval of TCSD (parameter 38, Table 28-10) while it becomes ready to execute code. When the CPU begins executing code, it resumes with the same clock source for the current Idle mode. For example, when waking from RC_IDLE mode, the internal oscillator block will clock the CPU and peripherals (in other words, RC_RUN mode). The IDLEN and SCS bits are not affected by the wake-up.

While in any Idle mode or Sleep mode, a WDT time-out will result in a WDT wake-up to the Run mode currently specified by the SCS<1:0> bits.

**FIGURE 4-5: TRANSITION TIMING FOR ENTRY TO SLEEP MODE**

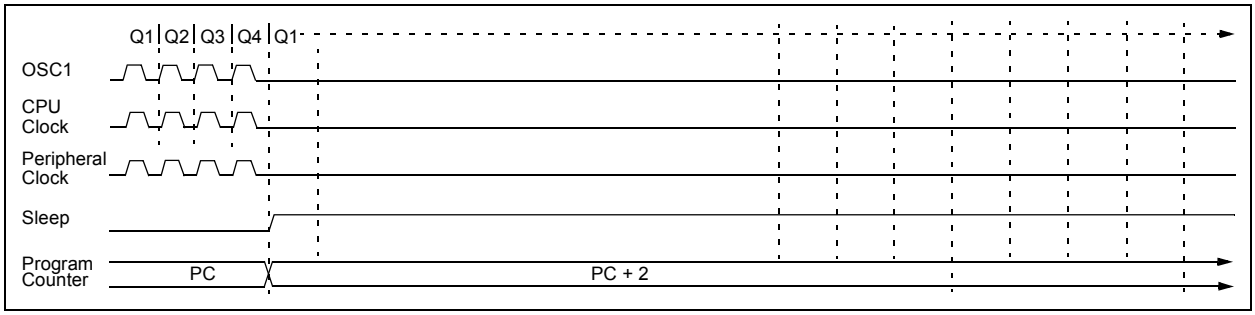
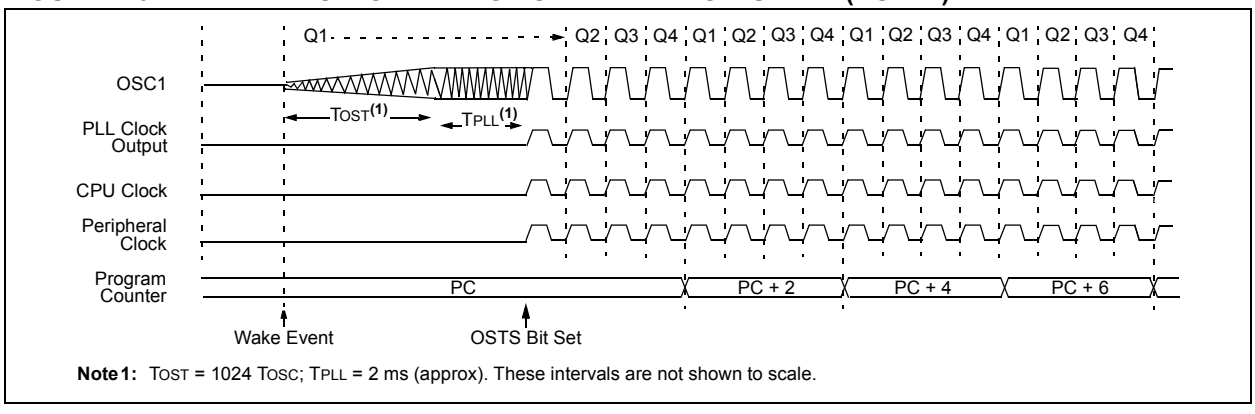



**FIGURE 4-6: TRANSITION TIMING FOR WAKE FROM SLEEP (HSPLL)**




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

### 4.4.1 PRI_IDLE MODE

This mode is unique among the three low-power Idle modes, in that it does not disable the primary device 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 from PRI_RUN mode by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set IDLEN first, then clear the SCS bits and execute SLEEP. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified by the FOSC<3:0> Configuration bits. The OSTS bit remains set (see Figure 4-7).

When a wake event occurs, the CPU is clocked from the primary clock source. A delay of interval TCSD is required between the wake event and when 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 4-8).

### 4.4.2 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 from SEC_RUN by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set the IDLEN bit first, then set the SCS<1:0> bits to ‘01’ and execute SLEEP. When the clock source is switched to the Timer1 oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the T1RUN bit is set.

When a wake event occurs, the peripherals continue to be clocked from the Timer1 oscillator. After an interval of TCSD following the wake event, the CPU begins executing code being clocked by the Timer1 oscillator. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run (see Figure 4-8).

| | |
|---|---|
| **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. |

**FIGURE 4-7: TRANSITION TIMING FOR ENTRY TO IDLE MODE**

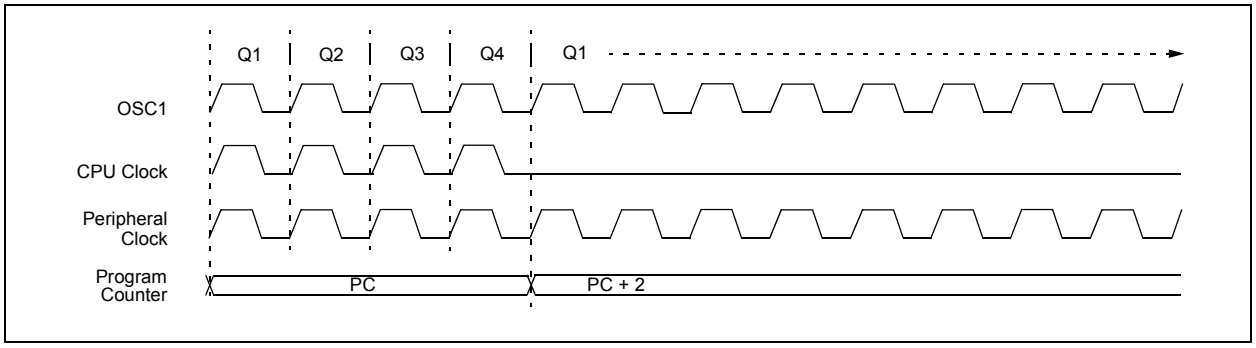
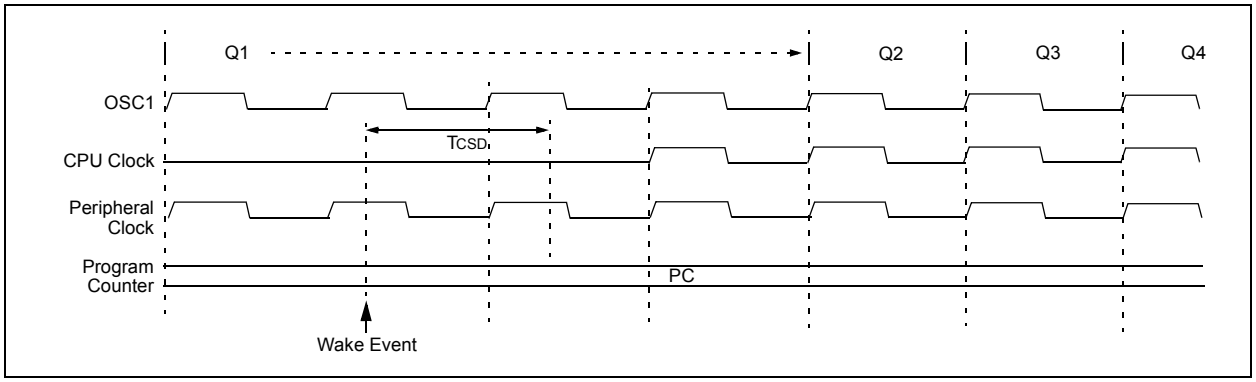



**FIGURE 4-8: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE**





© 2009 Microchip Technology Inc.

### 4.4.3 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.

From RC_RUN, this mode is entered by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, first set IDLEN, then set the SCS1 bit and execute SLEEP. Although its value is ignored, it is recommended that SCS0 also be cleared; this is to maintain software compatibility with future devices. 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, the primary oscillator is shut down and the OSTS bit is cleared.

If the IRCF bits are set to any non-zero value or the INTSRC bit is set, the INTOSC output is enabled. The IOFS bit becomes set, after the INTOSC output becomes stable, after an interval of TIOBST (parameter 39, Table 28-10). Clocks to the peripherals continue while the INTOSC source stabilizes. If the IRCF bits were previously at a non-zero value, or INTSRC was set before the SLEEP instruction was executed and the INTOSC source was already stable, the IOFS bit will remain set. If the IRCF bits and INTSRC are all clear, the INTOSC output will not be enabled, the IOFS bit will remain clear and 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 delay of TCSD following the wake event, the CPU begins executing code being clocked by the INTOSC multiplexer. 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.

## 4.5 Exiting Idle and Sleep Modes

An exit from Sleep mode or any of the Idle 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 **Section 4.2 “Run Modes”, Section 4.3 “Sleep Mode”** and **Section 4.4 “Idle Modes”**).

### 4.5.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit from an Idle mode or the Sleep mode to a Run mode. 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 Idle or Sleep modes 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 10.0 “Interrupts”**).

A fixed delay of interval, TCSD, 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.

### 4.5.2 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 an exit from the power-managed mode (see **Section 4.2 “Run Modes”** and **Section 4.3 “Sleep Mode”**). If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see **Section 25.2 “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 device clock source.

### 4.5.3 EXIT BY RESET

Normally, the device is held in Reset by the Oscillator Start-up Timer (OST) until the primary clock becomes ready. At that time, the OSTS bit is set and the device begins executing code. If the internal oscillator block is the new clock source, the IOFS bit is set instead.

The exit delay time from Reset to the start of code execution depends on both the clock sources before and after the wake-up and the type of oscillator if the new clock source is the primary clock. Exit delays are summarized in Table 4-2.

Code execution can begin before the primary clock becomes ready. If either the Two-Speed Start-up (see **Section 25.3 “Two-Speed Start-up”**) or Fail-Safe Clock Monitor (see **Section 25.4 “Fail-Safe Clock Monitor”**) is enabled, 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. 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.

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

Certain exits from power-managed modes do not invoke the OST at all. There are two cases:

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

In these instances, 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 of interval, TCSD, following the wake event is still required when leaving Sleep and Idle modes to allow the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

**TABLE 4-2: EXIT DELAY ON WAKE-UP BY RESET FROM SLEEP MODE OR ANY IDLE MODE (BY CLOCK SOURCES)**

| Clock Source Before Wake-up | Clock Source After Wake-up | Exit Delay | Clock Ready Status bit (OSCCON) |
|---|---|---|---|
| Primary Device Clock (PRI_IDLE mode) | LP, XT, HS | $T_{CSD}$(2) | OSTS |
| | HSPLL | | |
| | EC, RC | | |
| | INTRC(1) | | — |
| | INTOSC(3) | | IOFS |
| T1OSC or INTRC(1) | LP, XT, HS | $T_{OST}$(4) | OSTS |
| | HSPLL | $T_{OST} + t_{rc}$(4) | |
| | EC, RC | $T_{CSD}$(2) | |
| | INTRC(1) | | — |
| | INTOSC(3) | $T_{IOBST}$(5) | IOFS |
| INTOSC(3) | LP, XT, HS | $T_{OST}$(5) | OSTS |
| | HSPLL | $T_{OST} + t_{rc}$(4) | |
| | EC, RC | $T_{CSD}$(2) | |
| | INTRC(1) | | — |
| | INTOSC(3) | None | IOFS |
| None (Sleep mode) | LP, XT, HS | $T_{OST}$(4) | OSTS |
| | HSPLL | $T_{OST} + t_{rc}$(4) | |
| | EC, RC | $T_{CSD}$(2) | |
| | INTRC(1) | | — |
| | INTOSC(3) | $T_{IOBST}$(5) | IOFS |

**Note 1:** In this instance, refers specifically to the 31 kHz INTRC clock source.

**2:** TCSD (parameter 38) is a required delay when waking from Sleep and all Idle modes and runs concurrently with any other required delays (see **Section 4.4 "Idle Modes"**).

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

**4:** TOST is the Oscillator Start-up Timer (parameter 32). $t_{rc}$ is the PLL Lock-out Timer (parameter F12); it is also designated as TPLL.

**5:** Execution continues during TIOBST (parameter 39), the INTOSC stabilization period.


© 2009 Microchip Technology Inc.

## 5.0 RESET

The PIC18F2480/2580/4480/4580 devices differentiate between various kinds of Reset:

a) Power-on Reset (POR)
b) $\overline{\text{MCLR}}$ Reset during normal operation
c) $\overline{\text{MCLR}}$ Reset during power-managed modes
d) Watchdog Timer (WDT) Reset (during execution)
e) Programmable Brown-out Reset (BOR)
f) RESET Instruction
g) Stack Full Reset
h) Stack Underflow Reset

This section discusses Resets generated by $\overline{\text{MCLR}}$, POR and BOR, and covers the operation of the various start-up timers. Stack Reset events are covered in **Section 6.1.2.4 "Stack Full and Underflow Resets"**. WDT Resets are covered in **Section 25.2 "Watchdog Timer (WDT)"**.

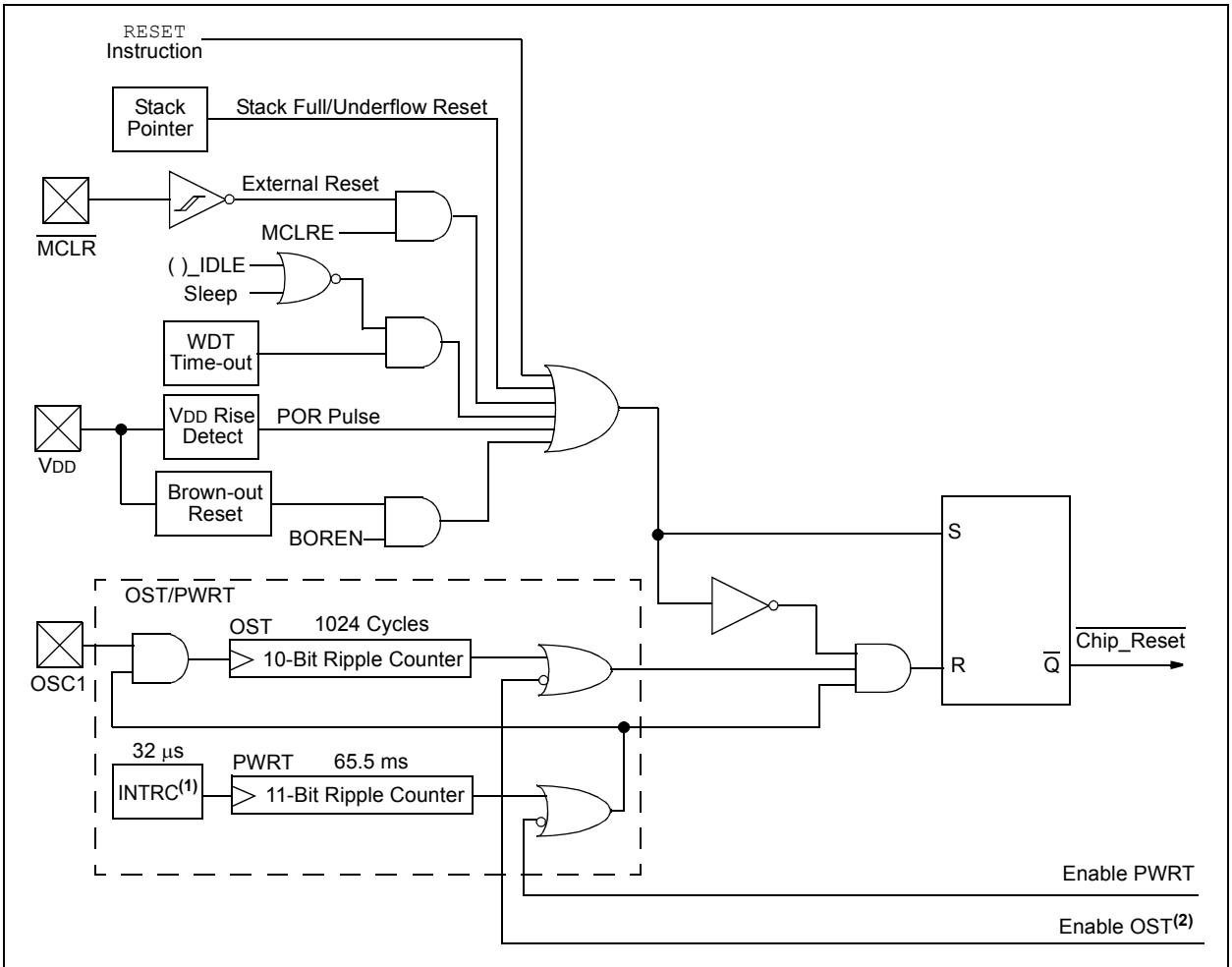
A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 5-1.

### 5.1 RCON Register

Device Reset events are tracked through the RCON register (Register 5-1). The lower five bits of the register indicate that a specific Reset event has occurred. In most cases, these bits can only be cleared by the event and must be set by the application after the event. The state of these flag bits, taken together, can be read to indicate the type of Reset that just occurred. This is described in more detail in **Section 5.6 "Reset State of Registers"**.

The RCON register also has control bits for setting interrupt priority (IPEN) and software control of the BOR (SBOREN). Interrupt priority is discussed in **Section 10.0 "Interrupts"**. BOR is covered in **Section 5.4 "Brown-out Reset (BOR)"**.

**FIGURE 5-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**




**Note 1:** This is the INTRC source from the internal oscillator block and is separate from the RC oscillator of the CLKI pin.
**2:** See Table 5-2 for time-out situations.

**REGISTER 5-1: RCON: RESET CONTROL REGISTER**

| R/W-0 | R/W-1(1) | U-0 | R/W-1 | R-1 | R-1 | R/W-0(2) | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | SBOREN | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **SBOREN:** BOR Software Enable bit(1)
If BOREN<1:0> = 01:
1 = BOR is enabled
0 = BOR is disabled
If BOREN<1:0> = 00, 10 or 11:
Bit is disabled and reads as '0'.

bit 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 = Set by execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit(2)
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:** If SBOREN is enabled, its Reset state is '1'; otherwise, it is '0'.
**2:** The actual Reset value of $\overline{POR}$ is determined by the type of device Reset. See the notes following this register and **Section 5.6 "Reset State of Registers"** for additional information.

**Note 1:** 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.
**2:** Brown-out Reset is said to have occurred when $\overline{BOR}$ is '0' and $\overline{POR}$ is '1' (assuming that $\overline{POR}$ was set to '1' by software immediately after a Power-on Reset).


© 2009 Microchip Technology Inc.

## 5.2 Master Clear Reset ($\overline{\text{MCLR}}$)

The $\overline{\text{MCLR}}$ pin provides a method for triggering an external Reset of the device. A Reset is generated by holding the pin low. These devices have a noise filter in the $\overline{\text{MCLR}}$ Reset path which detects and ignores small pulses.

The $\overline{\text{MCLR}}$ pin is not driven low by any internal Resets, including the WDT.

In PIC18F2480/2580/4480/4580 devices, the $\overline{\text{MCLR}}$ input can be disabled with the MCLRE Configuration bit. When $\overline{\text{MCLR}}$ is disabled, the pin becomes a digital input. See **Section 11.5 "PORTE, TRISE and LATE Registers"** for more information.

## 5.3 Power-on Reset (POR)

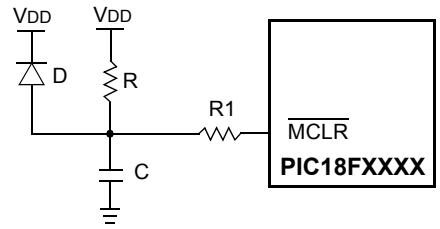
A Power-on Reset pulse is generated on-chip whenever VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.

To take advantage of the POR circuitry, tie the $\overline{\text{MCLR}}$ pin through a resistor (1 kΩ 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 5-2.

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.

POR events are captured by the $\overline{\text{POR}}$ bit (RCON<1>). The state of the bit is set to '0' whenever a Power-on Reset occurs; it does not change for any other Reset event. $\overline{\text{POR}}$ is not reset to '1' by any hardware event. To capture multiple events, the user manually resets the bit to '1' in software following any Power-on Reset.

**FIGURE 5-2: 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 $\overline{\text{MCLR}}$ from external capacitor C, in the event of $\overline{\text{MCLR}}$/VPP pin breakdown, due to Electro-static Discharge (ESD) or Electrical Overstress (EOS).

## 5.4 Brown-out Reset (BOR)

PIC18F2480/2580/4480/4580 devices implement a BOR circuit that provides the user with a number of configuration and power-saving options. The BOR is controlled by the BORV<1:0> and BOREN<1:0> Configuration bits. There are a total of four BOR configurations which are summarized in Table 5-1.

The BOR threshold is set by the BORV<1:0> bits. If BOR is enabled (any values of BOREN<1:0>, except '00'), any drop of VDD below VBOR (parameter D005) for greater than TBOR (parameter 35) will reset the device. A Reset may or 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.

BOR and the Power-on Timer (PWRT) are independently configured. Enabling a Brown-out Reset does not automatically enable the PWRT.

### 5.4.1 SOFTWARE ENABLED BOR

When BOREN<1:0> = 01, the BOR can be enabled or disabled by the user in software. This is done with the control bit, SBOREN (RCON<6>). Setting SBOREN enables the BOR to function as previously described. Clearing SBOREN disables the BOR entirely. The SBOREN bit operates only in this mode; otherwise it is read as '0'.

Placing the BOR under software control gives the user the additional flexibility of tailoring the application to its environment without having to reprogram the device to change BOR configuration. It also allows the user to tailor device power consumption in software by eliminating the incremental current that the BOR consumes. While the BOR current is typically very small, it may have some impact in low-power applications.

| **Note:** | Even when BOR is under software control, the Brown-out Reset voltage level is still set by the BORV<1:0> Configuration bits. It cannot be changed in software. |
|---|---|

### 5.4.2 DETECTING BOR

When Brown-out Reset is enabled, the $\overline{\text{BOR}}$ bit always resets to '0' on any Brown-out Reset or Power-on Reset event. This makes it difficult to determine if a Brown-out Reset event has occurred just by reading the state of $\overline{\text{BOR}}$ alone. A more reliable method is to simultaneously check the state of both $\overline{\text{POR}}$ and $\overline{\text{BOR}}$. This assumes that the $\overline{\text{POR}}$ bit is reset to '1' in software immediately after any Power-on Reset event. IF $\overline{\text{BOR}}$ is '0' while $\overline{\text{POR}}$ is '1', it can be reliably assumed that a Brown-out Reset event has occurred.

### 5.4.3 DISABLING BOR IN SLEEP MODE

When BOREN<1:0> = 10, the BOR remains under hardware control and operates as previously described. Whenever the device enters Sleep mode, however, the BOR is automatically disabled. When the device returns to any other operating mode, BOR is automatically re-enabled.

This mode allows for applications to recover from brown-out situations, while actively executing code, when the device requires BOR protection the most. At the same time, it saves additional power in Sleep mode by eliminating the small incremental BOR current.

**TABLE 5-1: BOR CONFIGURATIONS**

| BOR Configuration | | Status of SBOREN (RCON<6>) | BOR Operation |
|---|---|---|---|
| BOREN1 | BOREN0 | | |
| 0 | 0 | Unavailable | BOR disabled; must be enabled by reprogramming the Configuration bits. |
| 0 | 1 | Available | BOR enabled in software; operation controlled by SBOREN. |
| 1 | 0 | Unavailable | BOR enabled in hardware in Run and Idle modes, disabled during Sleep mode. |
| 1 | 1 | Unavailable | BOR enabled in hardware; must be disabled by reprogramming the Configuration bits. |


© 2009 Microchip Technology Inc.

## 5.5 Device Reset Timers

PIC18F2480/2580/4480/4580 devices incorporate three separate on-chip timers that help regulate the Power-on Reset process. Their main function is to ensure that the device clock is stable before code is executed. These timers are:

- Power-up Timer (PWRT)
- Oscillator Start-up Timer (OST)
- PLL Lock Time-out

### 5.5.1 POWER-UP TIMER (PWRT)

The Power-up Timer (PWRT) of the PIC18F2480/2580/4480/4580 devices is an 11-bit counter which uses the INTRC source as the clock input. This yields an approximate time interval of 2048 x 32 μs = 65.6 ms. While the PWRT is counting, the device is held in Reset.

The power-up time delay depends on the INTRC clock and will vary from chip-to-chip due to temperature and process variation. See DC parameter 33 for details.

The PWRT is enabled by clearing the $\overline{\text{PWRTEN}}$ Configuration bit.

### 5.5.2 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 power-managed modes.

### 5.5.3 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 separate 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 ($T_{PLL}$) is typically 2 ms and follows the oscillator start-up time-out.

### 5.5.4 TIME-OUT SEQUENCE

On power-up, the time-out sequence is as follows:

1. After the POR pulse has cleared, PWRT time-out is invoked (if enabled).
2. Then, the OST is activated.

The total time-out will vary based on oscillator configuration and the status of the PWRT. Figure 5-3, Figure 5-4, Figure 5-5, Figure 5-6 and Figure 5-7 all depict time-out sequences on power-up, with the Power-up Timer enabled and the device operating in HS Oscillator mode. Figures 5-3 through 5-6 also apply to devices operating in XT or LP modes. For devices in RC mode and with the PWRT disabled, on the other hand, there will be no time-out at all.

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

**TABLE 5-2: TIME-OUT IN VARIOUS SITUATIONS**

| Oscillator Configuration | Power-up[2] and Brown-out | | Exit from Power-Managed Mode |
|---|---|---|---|
| | $\overline{\text{PWRTEN}}$ = 0 | $\overline{\text{PWRTEN}}$ = 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] | — | — |
| RC, RCIO | 66 ms[1] | — | — |
| INTIO1, INTIO2 | 66 ms[1] | — | — |

**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 PLL to lock.

**FIGURE 5-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ TIED TO $V_{DD}$, $V_{DD}$ RISE < $T_{PWRT}$)**

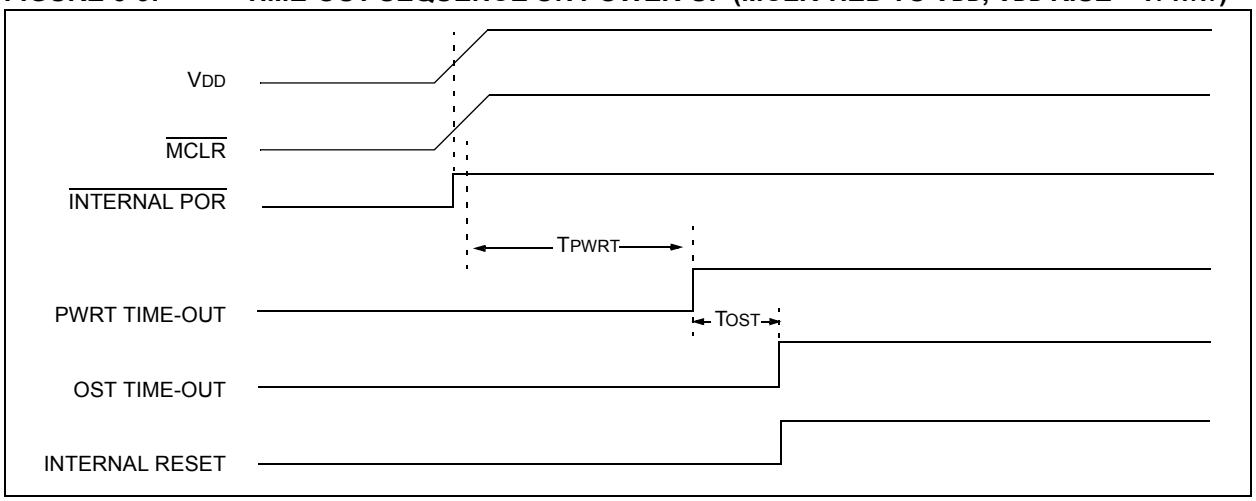



**FIGURE 5-4: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ NOT TIED TO $V_{DD}$): CASE 1**

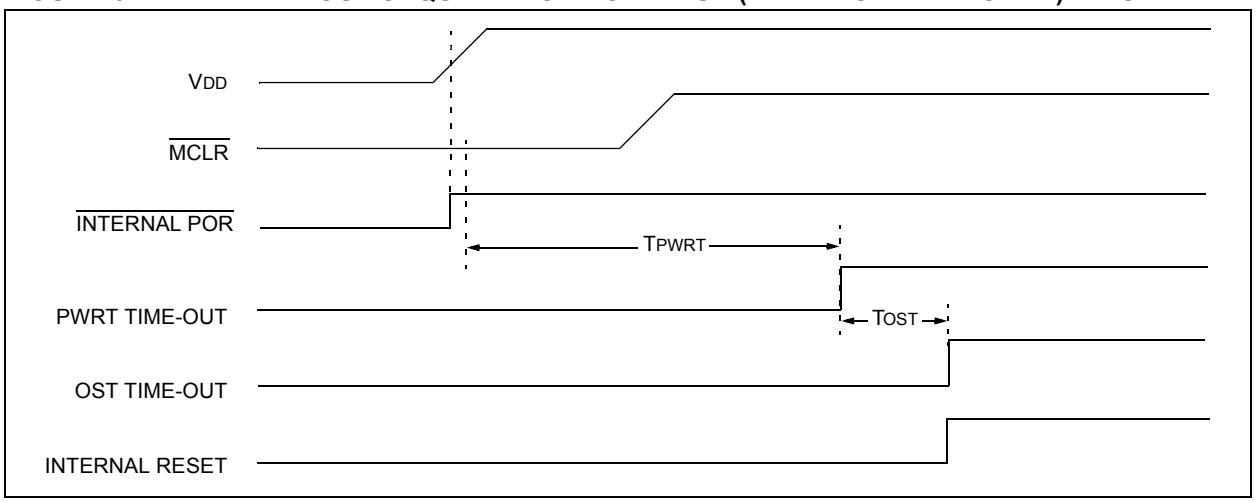
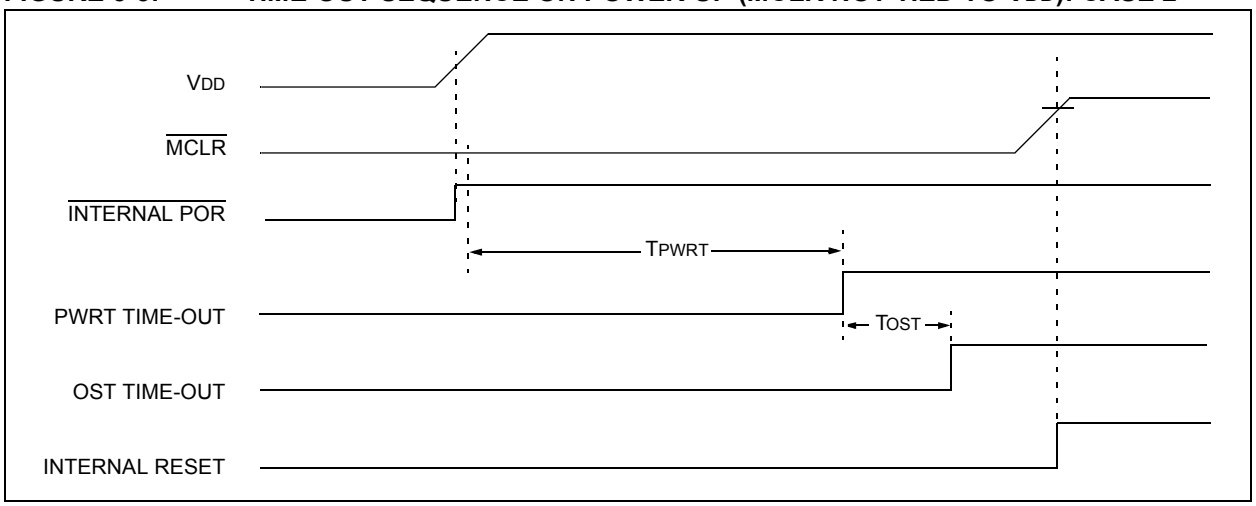



**FIGURE 5-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{MCLR}$ NOT TIED TO $V_{DD}$): CASE 2**





© 2009 Microchip Technology Inc.

**FIGURE 5-6: SLOW RISE TIME ($\overline{MCLR}$ TIED TO $V_{DD}$, $V_{DD}$ RISE > $T_{PWRT}$)**

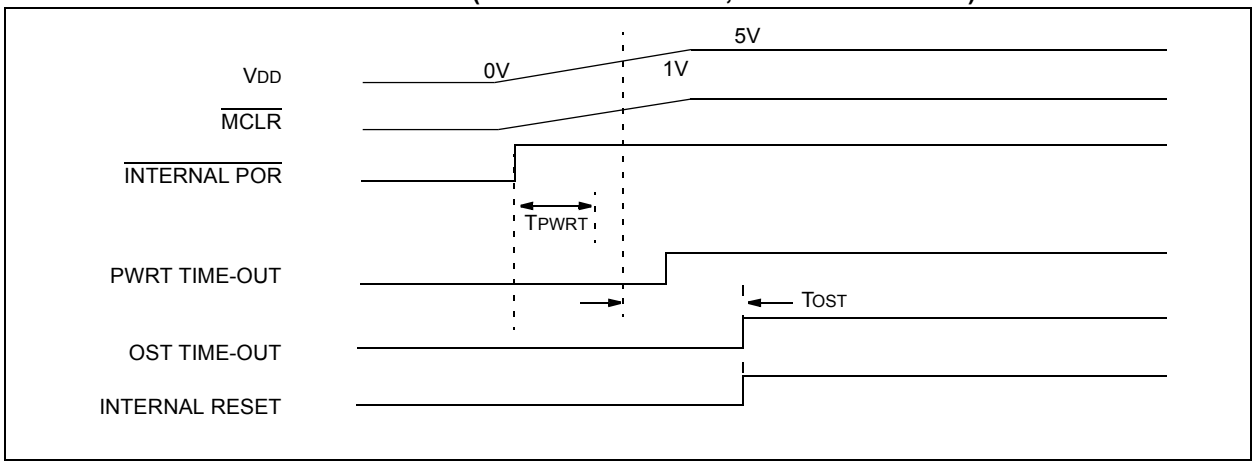
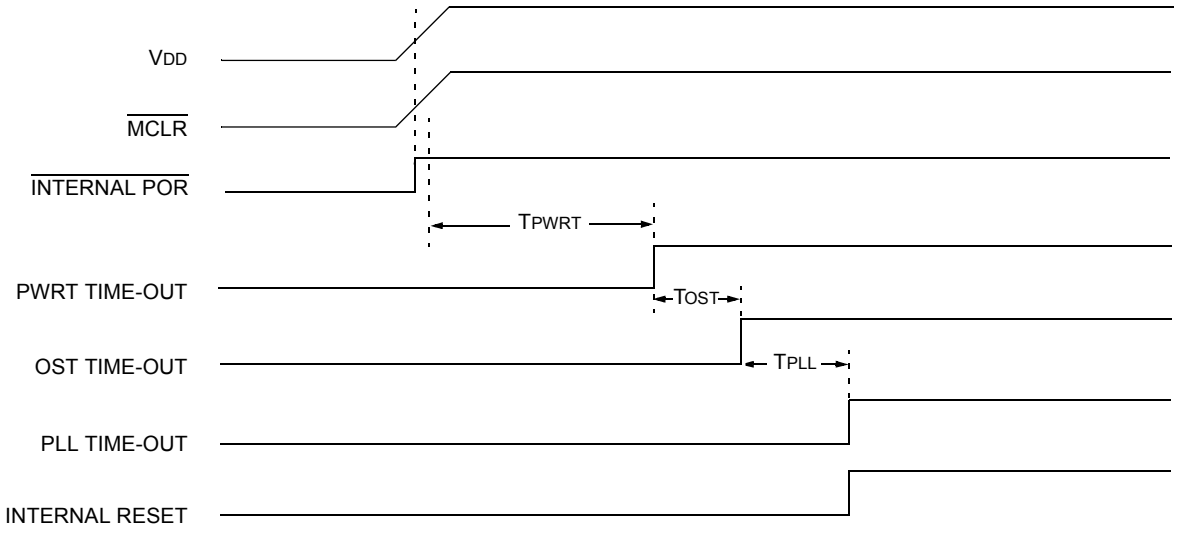



**FIGURE 5-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED ($\overline{MCLR}$ TIED TO $V_{DD}$)**




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

## 5.6 Reset State of Registers

Most registers are unaffected by a Reset. Their status is unknown on a Power-on Reset 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, $\overline{RI}$, $\overline{TO}$, $\overline{PD}$, $\overline{POR}$ and $\overline{BOR}$, are set or cleared differently in different Reset situations, as indicated in Table 5-3. These bits are used in software to determine the nature of the Reset.

Table 5-4 describes the Reset states for all of the Special Function Registers. These are categorized by Power-on and Brown-out Resets, Master Clear and WDT Resets and WDT wake-ups.

**TABLE 5-3: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER**

| Condition | Program Counter(1) | RCON Register | | | | | | STKPTR Register | |
|---|---|---|---|---|---|---|---|---|---|
| | | SBOREN | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | STKFUL | STKUNF |
| Power-on Reset | 0000h | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| RESET Instruction | 0000h | u(2) | 0 | u | u | u | u | u | u |
| Brown-out Reset | 0000h | u(2) | 1 | 1 | 1 | u | 0 | u | u |
| $\overline{MCLR}$ Reset during Power-Managed Run modes | 0000h | u(2) | u | 1 | u | u | u | u | u |
| $\overline{MCLR}$ Reset during Power-Managed Idle modes and Sleep mode | 0000h | u(2) | u | 1 | 0 | u | u | u | u |
| WDT Time-out during Full Power or Power-Managed Run modes | 0000h | u(2) | u | 0 | u | u | u | u | u |
| $\overline{MCLR}$ Reset during Full-Power execution | 0000h | u(2) | u | u | u | u | u | u | u |
| Stack Full Reset (STVREN = 1) | 0000h | u(2) | u | u | u | u | u | 1 | u |
| Stack Underflow Reset (STVREN = 1) | 0000h | u(2) | u | u | u | u | u | u | 1 |
| Stack Underflow Error (not an actual Reset, STVREN = 0) | 0000h | u(2) | u | u | u | u | u | u | 1 |
| WDT Time-out during Power-Managed Idle or Sleep modes | PC + 2 | u(2) | u | 0 | 0 | u | u | u | u |
| Interrupt Exit from Power-Managed modes | PC + 2 | u(2) | u | u | 0 | u | u | u | u |

**Legend:** u = unchanged

**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 (008h or 0018h).

**2:** Reset state is '1' for POR and unchanged for all other Resets when software BOR is enabled (BOREN<1:0> Configuration bits = 01 and SBOREN = 1); otherwise, the Reset state is '0'.


© 2009 Microchip Technology Inc.

**TABLE 5-4: 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 | 2480 | 2580 | 4480 | 4580 | ---0 0000 | ---0 0000 | ---0 uuuu(3) |
| TOSH | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| TOSL | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| STKPTR | 2480 | 2580 | 4480 | 4580 | 00-0 0000 | uu-0 0000 | uu-u uuuu(3) |
| PCLATU | 2480 | 2580 | 4480 | 4580 | ---0 0000 | ---0 0000 | ---u uuuu |
| PCLATH | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCL | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | PC + 2(2) |
| TBLPTRU | 2480 | 2580 | 4480 | 4580 | --00 0000 | --00 0000 | --uu uuuu |
| TBLPTRH | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TBLPTRL | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TABLAT | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PRODH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PRODL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INTCON | 2480 | 2580 | 4480 | 4580 | 0000 000x | 0000 000u | uuuu uuuu(1) |
| INTCON2 | 2480 | 2580 | 4480 | 4580 | 1111 -1-1 | 1111 -1-1 | uuuu -u-u(1) |
| INTCON3 | 2480 | 2580 | 4480 | 4580 | 11-0 0-00 | 11-0 0-00 | uu-u u-uu(1) |
| INDF0 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| POSTINC0 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| POSTDEC0 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| PREINC0 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| PLUSW0 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| FSR0H | 2480 | 2580 | 4480 | 4580 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR0L | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| WREG | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INDF1 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| POSTINC1 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| POSTDEC1 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| PREINC1 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| PLUSW1 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| FSR1H | 2480 | 2580 | 4480 | 4580 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR1L | 2480 | 2580 | 4480 | 4580 | 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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.

**TABLE 5-4: 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 | 2480 | 2580 | 4480 | 4580 | ---- 0000 | ---- 0000 | ---- uuuu |
| INDF2 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| POSTINC2 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| POSTDEC2 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| PREINC2 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| PLUSW2 | 2480 | 2580 | 4480 | 4580 | N/A | N/A | N/A |
| FSR2H | 2480 | 2580 | 4480 | 4580 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR2L | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| STATUS | 2480 | 2580 | 4480 | 4580 | ---x xxxx | ---u uuuu | ---u uuuu |
| TMR0H | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TMR0L | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T0CON | 2480 | 2580 | 4480 | 4580 | 1111 1111 | 1111 1111 | uuuu uuuu |
| OSCCON | 2480 | 2580 | 4480 | 4580 | 0100 q000 | 0100 00q0 | uuuu uuqu |
| HLVDCON | 2480 | 2580 | 4480 | 4580 | 0-00 0101 | 0-00 0101 | 0-uu uuuu |
| WDTCON | 2480 | 2580 | 4480 | 4580 | ---- ---0 | ---- ---0 | ---- ---u |
| RCON(4) | 2480 | 2580 | 4480 | 4580 | 0q-1 11q0 | 0q-q qquu | uq-u qquu |
| TMR1H | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR1L | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T1CON | 2480 | 2580 | 4480 | 4580 | 0000 0000 | u0uu uuuu | uuuu uuuu |
| TMR2 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PR2 | 2480 | 2580 | 4480 | 4580 | 1111 1111 | 1111 1111 | 1111 1111 |
| T2CON | 2480 | 2580 | 4480 | 4580 | -000 0000 | -000 0000 | -uuu uuuu |
| SSPBUF | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| SSPADD | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPSTAT | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON1 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSPCON2 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ADRESH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADRESL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADCON0 | 2480 | 2580 | 4480 | 4580 | --00 0000 | --00 0000 | --uu uuuu |
| ADCON1 | 2480 | 2580 | 4480 | 4580 | --00 0qqq | --00 0qqq | --uu 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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.


© 2009 Microchip Technology Inc.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| ADCON2 | 2480 | 2580 | 4480 | 4580 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| CCPR1H | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR1L | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP1CON | 2480 | 2580 | 4480 | 4580 | --00 0000 | --00 0000 | --uu uuuu |
| ECCPR1H | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ECCPR1L | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ECCP1CON | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| BAUDCON | 2480 | 2580 | 4480 | 4580 | 01-0 0-00 | 01-0 0-00 | --uu uuuu |
| ECCP1DEL | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP1AS | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CVRCON | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CMCON | 2480 | 2580 | 4480 | 4580 | 0000 0111 | 0000 0111 | uuuu uuuu |
| TMR3H | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR3L | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T3CON | 2480 | 2580 | 4480 | 4580 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| SPBRGH | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SPBRG | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RCREG | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXREG | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXSTA | 2480 | 2580 | 4480 | 4580 | 0000 0010 | 0000 0010 | uuuu uuuu |
| RCSTA | 2480 | 2580 | 4480 | 4580 | 0000 000x | 0000 000x | uuuu uuuu |
| EEADR | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| EEDATA | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| EECON2 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | 0000 0000 |
| EECON1 | 2480 | 2580 | 4480 | 4580 | xx-0 x000 | uu-0 u000 | uu-0 u000 |
| IPR3 | 2480 | 2580 | 4480 | 4580 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIR3 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PIE3 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| IPR2 | 2480 | 2580 | 4480 | 4580 | 11-1 1111 | 11-1 1111 | uu-u uuuu |
| | 2480 | 2580 | 4480 | 4580 | 1--1 111- | 1--1 111- | u--u uuu- |

**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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| PIR2 | 2480 | 2580 | 4480 | 4580 | 00-0 0000 | 00-0 0000 | uu-u uuuu(1) |
| | 2480 | 2580 | 4480 | 4580 | 0--0 000- | 0--0 000- | u--u uuu-(1) |
| PIE2 | 2480 | 2580 | 4480 | 4580 | 00-0 0000 | 00-0 0000 | uu-u uuuu |
| | 2480 | 2580 | 4480 | 4580 | 0--0 000- | 0--0 000- | u--u uuu- |
| IPR1 | 2480 | 2580 | 4480 | 4580 | 1111 1111 | 1111 1111 | uuuu uuuu |
| | 2480 | 2580 | 4480 | 4580 | -111 1111 | -111 1111 | -uuu uuuu |
| PIR1 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu(1) |
| | 2480 | 2580 | 4480 | 4580 | -000 0000 | -000 0000 | -uuu uuuu |
| PIE1 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| | 2480 | 2580 | 4480 | 4580 | -000 0000 | -000 0000 | -uuu uuuu |
| OSCTUNE | 2480 | 2580 | 4480 | 4580 | --00 0000 | --00 0000 | --uu uuuu |
| TRISE | 2480 | 2580 | 4480 | 4580 | 0000 -111 | 0000 -111 | uuuu -uuu |
| TRISD | 2480 | 2580 | 4480 | 4580 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISC | 2480 | 2580 | 4480 | 4580 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISB | 2480 | 2580 | 4480 | 4580 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISA(5) | 2480 | 2580 | 4480 | 4580 | 1111 1111(5) | 1111 1111(5) | uuuu uuuu(5) |
| LATE | 2480 | 2580 | 4480 | 4580 | ---- -xxx | ---- -uuu | ---- -uuu |
| LATD | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATC | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATB | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATA(5) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx(5) | uuuu uuuu(5) | uuuu uuuu(5) |
| PORTE | 2480 | 2580 | 4480 | 4580 | ---- x000 | ---- x000 | ---- uuuu |
| PORTD | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTC | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTB | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTA(5) | 2480 | 2580 | 4480 | 4580 | xx0x 0000(5) | uu0u 0000(5) | uuuu uuuu(5) |
| ECANCON | 2480 | 2580 | 4480 | 4580 | 0001 0000 | 0001 0000 | uuuu uuuu |
| TXERRCNT | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXERRCNT | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| COMSTAT | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CIOCON | 2480 | 2580 | 4480 | 4580 | --00 ---- | --00 ---- | --uu ---- |

**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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.


© 2009 Microchip Technology Inc.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| BRGCON3 | 2480 | 2580 | 4480 | 4580 | 00-- -000 | 00-- -000 | uu-- -uuu |
| BRGCON2 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| BRGCON1 | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CANCON | 2480 | 2580 | 4480 | 4580 | 1000 000- | 1000 000- | uuuu uuu- |
| CANSTAT | 2480 | 2580 | 4480 | 4580 | 100- 000- | 100- 000- | uuu- uuu- |
| RXB0D7 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D6 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D5 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D4 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D3 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D2 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D1 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0D0 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0DLC | 2480 | 2580 | 4480 | 4580 | -xxx xxxx | -uuu uuuu | -uuu uuuu |
| RXB0EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0SIDL | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| RXB0SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB0CON | 2480 | 2580 | 4480 | 4580 | 000- 0000 | 000- 0000 | uuu- uuuu |
| RXB1D7 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D6 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D5 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D4 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D3 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D2 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D1 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1D0 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1DLC | 2480 | 2580 | 4480 | 4580 | -xxx xxxx | -uuu uuuu | -uuu uuuu |
| RXB1EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1SIDL | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | uuuu u-uu |

**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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| RXB1SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXB1CON | 2480 | 2580 | 4480 | 4580 | 000- 0000 | 000- 0000 | uuu- uuuu |
| TXB0D7 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D6 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D5 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D4 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D3 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D2 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D1 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0D0 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0DLC | 2480 | 2580 | 4480 | 4580 | -x-- xxxx | -u-- uuuu | -u-- uuuu |
| TXB0EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| TXB0SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| TXB0SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB0CON | 2480 | 2580 | 4480 | 4580 | 0000 0-00 | 0000 0-00 | uuuu u-uu |
| TXB1D7 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D6 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D5 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D4 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D3 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D2 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D1 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1D0 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1DLC | 2480 | 2580 | 4480 | 4580 | -x-- xxxx | -u-- uuuu | -u-- uuuu |
| TXB1EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB1SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- uu-u |
| TXB1SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| TXB1CON | 2480 | 2580 | 4480 | 4580 | 0000 0-00 | 0000 0-00 | uuuu u-uu |
| TXB2D7 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | 0uuu 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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.


© 2009 Microchip Technology Inc.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| TXB2D6 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | 0uuu uuuu |
| TXB2D5 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | 0uuu uuuu |
| TXB2D4 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | 0uuu uuuu |
| TXB2D3 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | 0uuu uuuu |
| TXB2D2 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | 0uuu uuuu |
| TXB2D1 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | 0uuu uuuu |
| TXB2D0 | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | 0uuu uuuu |
| TXB2DLC | 2480 | 2580 | 4480 | 4580 | -x-- xxxx | -u-- uuuu | -u-- uuuu |
| TXB2EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXB2SIDL | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | -uuu uuuu |
| TXB2SIDH | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| TXB2CON | 2480 | 2580 | 4480 | 4580 | 0000 0-00 | 0000 0-00 | uuuu u-uu |
| RXM1EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM1EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM1SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXM1SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM0EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM0EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXM0SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXM0SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF5EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF5EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF5SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF5SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF4EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF4EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF4SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF4SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF3EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF3EIDH | 2480 | 2580 | 4480 | 4580 | 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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| RXF3SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF3SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF2EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF2EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF2SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF2SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF1EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF1EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF1SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF1SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF0EIDL | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF0EIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF0SIDL | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF0SIDH | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5D7(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5D6(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5D5(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5D4(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5D3(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5D2(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5D1(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5D0(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5DLC(6) | 2480 | 2580 | 4480 | 4580 | -xxx xxxx | -uuu uuuu | -uuu uuuu |
| B5EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B5SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| B5SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| B5CON(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| B4D7(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4D6(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4D5(6) | 2480 | 2580 | 4480 | 4580 | 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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.


© 2009 Microchip Technology Inc.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| B4D4(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4D3(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4D2(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4D1(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4D0(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4DLC(6) | 2480 | 2580 | 4480 | 4580 | -xxx xxxx | -uuu uuuu | -uuu uuuu |
| B4EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| B4SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B4CON(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| B3D7(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3D6(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3D5(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3D4(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3D3(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3D2(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3D1(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3D0(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3DLC(6) | 2480 | 2580 | 4480 | 4580 | -xxx xxxx | -uuu uuuu | -uuu uuuu |
| B3EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| B3SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B3CON(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| B2D7(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2D6(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2D5(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2D4(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2D3(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2D2(6) | 2480 | 2580 | 4480 | 4580 | 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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| B2D1(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2D0(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2DLC(6) | 2480 | 2580 | 4480 | 4580 | -xxx xxxx | -uuu uuuu | -uuu uuuu |
| B2EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| B2SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B2CON(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| B1D7(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1D6(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1D5(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1D4(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1D3(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1D2(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1D1(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1D0(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1DLC(6) | 2480 | 2580 | 4480 | 4580 | -xxx xxxx | -uuu uuuu | -uuu uuuu |
| B1EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| B1SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B1CON(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| B0D7(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0D6(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0D5(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0D4(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0D3(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0D2(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0D1(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0D0(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0DLC(6) | 2480 | 2580 | 4480 | 4580 | -xxx xxxx | -uuu uuuu | -uuu 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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.


© 2009 Microchip Technology Inc.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| B0EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx x-xx | uuuu u-uu | uuuu u-uu |
| B0SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| B0CON(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXBIE(6) | 2480 | 2580 | 4480 | 4580 | ---0 00-- | ---u uu-- | ---u uu-- |
| BIE0(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| BSEL0(6) | 2480 | 2580 | 4480 | 4580 | 0000 00-- | 0000 00-- | uuuu uu-- |
| MSEL3(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MSEL2(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MSEL1(6) | 2480 | 2580 | 4480 | 4580 | 0000 0101 | 0000 0101 | uuuu uuuu |
| MSEL0(6) | 2480 | 2580 | 4480 | 4580 | 0101 0000 | 0101 0000 | uuuu uuuu |
| SDFLC(6) | 2480 | 2580 | 4480 | 4580 | ---0 0000 | ---0 0000 | -u-- uuuu |
| RXFCON1(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXFCON0(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXFBCON7(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXFBCON6(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXFBCON5(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXFBCON4(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXFBCON3(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXFBCON2(6) | 2480 | 2580 | 4480 | 4580 | 0001 0001 | 0001 0001 | uuuu uuuu |
| RXFBCON1(6) | 2480 | 2580 | 4480 | 4580 | 0001 0001 | 0001 0001 | uuuu uuuu |
| RXFBCON0(6) | 2480 | 2580 | 4480 | 4580 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RXF15EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF15EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF15SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF15SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF14EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF14EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF14SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF14SIDH(6) | 2480 | 2580 | 4480 | 4580 | 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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.

**TABLE 5-4: 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 |
|---|---|---|---|---|---|---|---|
| RXF13EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF13EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF13SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF13SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF12EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF12EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF12SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF12SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF11EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF11EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF11SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | uuu- u-uu |
| RXF11SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RXF10EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF10EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF10SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | -uuu uuuu |
| RXF10SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF9EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF9EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF9SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | -uuu uuuu |
| RXF9SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF8EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF8EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF8SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | -uuu uuuu |
| RXF8SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF7EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF7EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF7SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | -uuu uuuu |
| RXF7SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF6EIDL(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF6EIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu uuuu |
| RXF6SIDL(6) | 2480 | 2580 | 4480 | 4580 | xxx- x-xx | uuu- u-uu | -uuu uuuu |
| RXF6SIDH(6) | 2480 | 2580 | 4480 | 4580 | xxxx xxxx | uuuu uuuu | -uuu 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 5-3 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:** This register reads all '0's until ECAN™ technology is set up in Mode 1 or Mode 2.


© 2009 Microchip Technology Inc.

## 6.0 MEMORY ORGANIZATION

There are three types of memory in PIC18 Enhanced microcontroller devices:

- Program Memory
- Data RAM
- Data EEPROM

As Harvard architecture devices, the data and program memories use separate busses; this allows for concurrent access of the two memory spaces. The data EEPROM, for practical purposes, can be regarded as a peripheral device, since it is addressed and accessed through a set of control registers.

Additional detailed information on the operation of the Flash program memory is provided in **Section 7.0 "Flash Program Memory"**. Data EEPROM is discussed separately in **Section 8.0 "Data EEPROM Memory"**.

### 6.1 Program Memory Organization

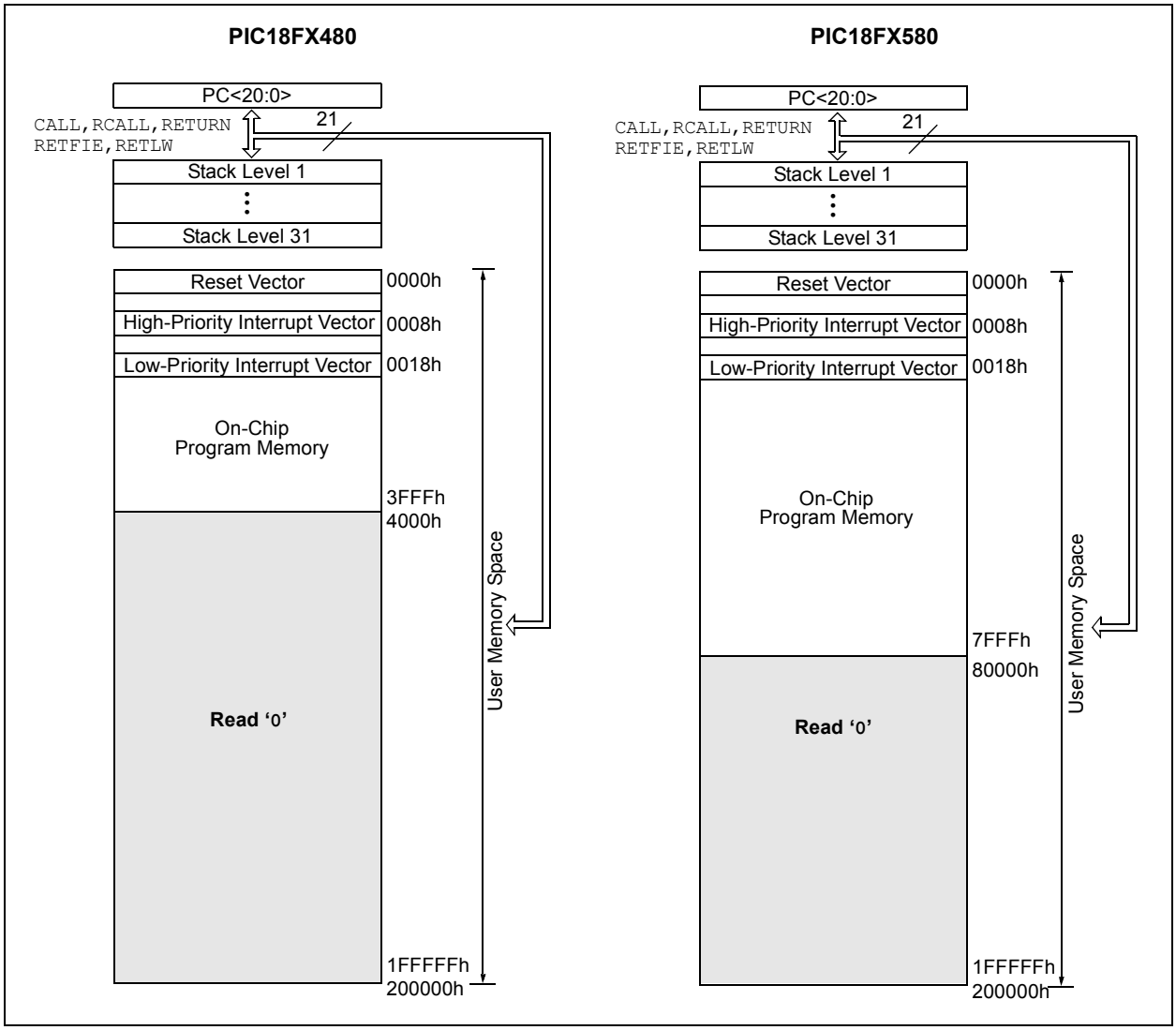
PIC18 microcontrollers implement a 21-bit program counter, which is capable of addressing a 2-Mbyte program memory space. Accessing a location between upper boundary of the physically implemented memory and the 2-Mbyte address will return all '0's (a NOP instruction).

The PIC18F2480 and PIC18F4480 each have 16 Kbytes of Flash memory and can store up to 8,192 single-word instructions. The PIC18F2580 and PIC18F4580 each have 32 Kbytes of Flash memory and can store up to 16,384 single-word instructions.

PIC18 devices have two interrupt vectors. The Reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

The program memory maps for PIC18FX480 and PIC18FX580 devices are shown in Figure 6-1.

**FIGURE 6-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F2480/2580/4480/4580 DEVICES**



### 6.1.1 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. 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; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 6.1.4.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by 2 to address sequential instructions in the program memory.

The CALL, RCALL and GOTO 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.

### 6.1.2 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC 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, STKPTR. 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 (TOF) Special File Registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL type instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

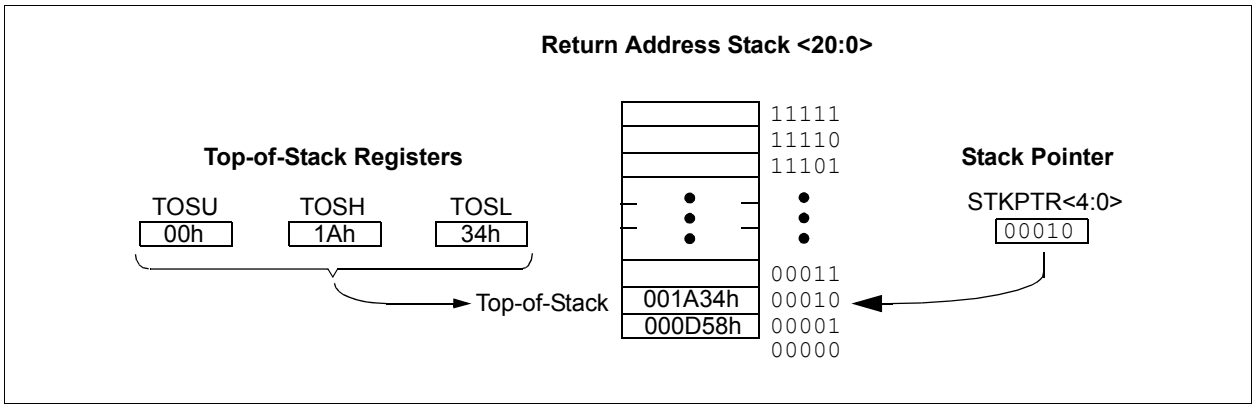
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits indicate if the stack is full or has overflowed or has underflowed.

#### 6.1.2.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 6-2). 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:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the global interrupt enable bits while accessing the stack to prevent inadvertent stack corruption.

**FIGURE 6-2: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**





© 2009 Microchip Technology Inc.

#### 6.1.2.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 6-1) 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. On 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 STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 25.1 "Configuration Bits"** for a description of the device Configuration bits.) If STVREN 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 STVREN 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 until 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. |
|---|---|

#### 6.1.2.3 PUSH and POP Instructions

Since the Top-of-Stack 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 feature. The PIC18 instruction set includes two instructions, PUSH and POP, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The PUSH instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The POP instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

**REGISTER 6-1: STKPTR: STACK POINTER REGISTER**

| R/C-0 | R/C-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| STKFUL(1) | STKUNF(1) | — | SP4 | SP3 | SP2 | SP1 | SP0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | C = Clearable bit | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 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.

#### 6.1.2.4 Stack Full and Underflow Resets

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

### 6.1.3 FAST REGISTER STACK

A Fast Register Stack is provided for the STATUS, WREG and BSR registers, to provide a “fast return” option for interrupts. Each stack is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers, if the `RETFIE, FAST` instruction is used to return from the interrupt.

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. In these cases, 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 6-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

**EXAMPLE 6-1: 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
```

### 6.1.4 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed `GOTO`
- Table Reads

#### 6.1.4.1 Computed `GOTO`

A computed `GOTO` is accomplished by adding an offset to the program counter. An example is shown in Example 6-2.

A look-up table can be formed with an `ADDWF PCL` instruction and a group of `RETLW nn` instructions. The W register 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 nn` instructions, that returns the value ‘`nn`’ 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 6-2: COMPUTED `GOTO` USING AN OFFSET VALUE**

```
        MOVF   OFFSET, W
        CALL   TABLE
ORG     nn00h
TABLE   ADDWF  PCL
        RETLW  nnh
        RETLW  nnh
        RETLW  nnh
        .
        .
        .
```

#### 6.1.4.2 Table Reads and 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 or from program memory one byte at a time.

Table read and table write operations are discussed further in **Section 7.1 “Table Reads and Table Writes”**.


© 2009 Microchip Technology Inc.

## 6.2 PIC18 Instruction Cycle

### 6.2.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the Program Counter (PC) is incremented on every Q1; the instruction is fetched from the program memory and latched into the Instruction Register (IR) during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 6-3.

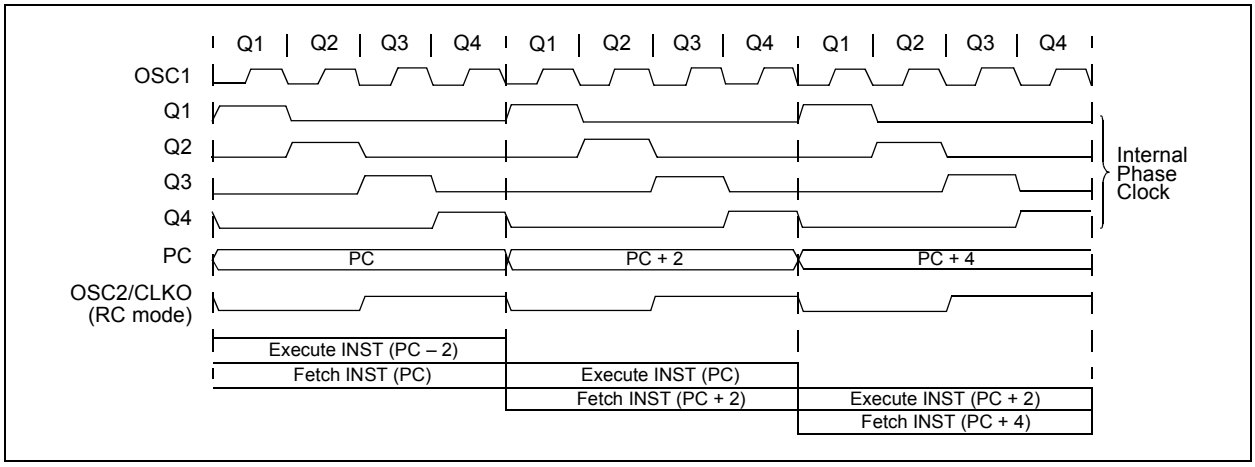
### 6.2.2 INSTRUCTION FLOW/PIPELINING

An “Instruction Cycle” consists of four Q cycles: Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take 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 6-3).

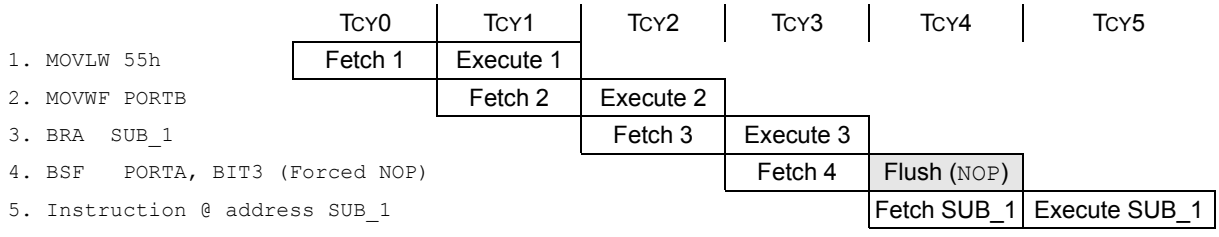
A fetch cycle begins with the program counter incrementing in Q1.

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 6-3: CLOCK/INSTRUCTION CYCLE**




**EXAMPLE 6-3: INSTRUCTION PIPELINE FLOW**




**Note:** 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.

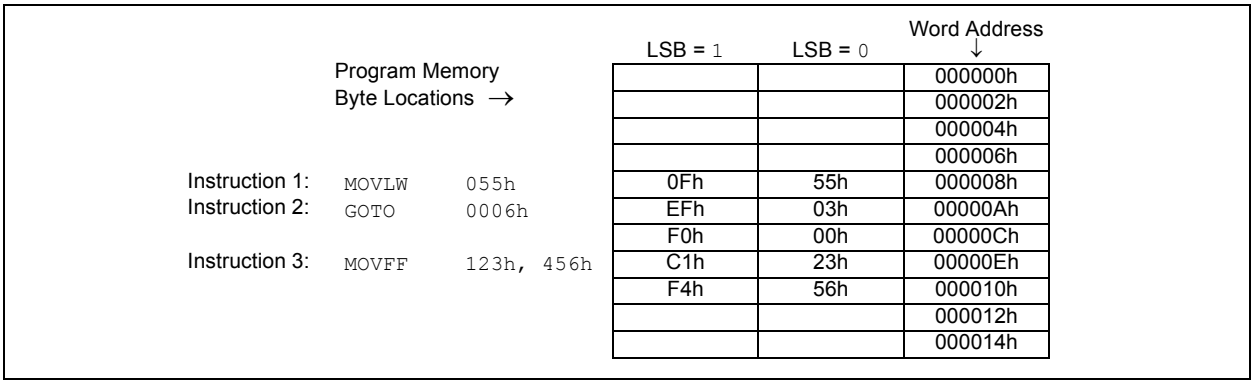
### 6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

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). To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see **Section 6.1.1 "Program Counter"**).

Figure 6-4 shows an example of how instruction words are stored in the program memory.

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 6-4 shows how the instruction, GOTO 0006h, 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 26.0 "Instruction Set Summary"** provides further details of the instruction set.

**FIGURE 6-4: INSTRUCTIONS IN PROGRAM MEMORY**




### 6.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four, two-word instructions: CALL, MOVFF, GOTO and LSFR. In all cases, the second word of the instructions always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the 4 MSbs of an instruction specifies a special form of NOP. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a NOP is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 6-4 shows how this works.

| **Note:** | See **Section 6.5 "Program Memory and the Extended Instruction Set"** for information on two-word instructions in the extended instruction set. |
|---|---|

**EXAMPLE 6-4: 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 |


© 2009 Microchip Technology Inc.

## 6.3 Data Memory Organization

**Note:** The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See **Section 6.6 "Data Memory and the Extended Instruction Set"** for more information.

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each; PIC18F2480/2580/4480/4580 devices implement all 16 banks. Figure 6-6 shows the data memory organization for the PIC18F2480/2580/4480/4580 devices.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '0's.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this subsection.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to SFRs and the lower portion of GPR Bank 0 without using the BSR. **Section 6.3.2 "Access Bank"** provides a detailed description of the Access RAM.

### 6.3.1 BANK SELECT REGISTER (BSR)

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 16 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 12-bit address, or an 8-bit low-order address and a 4-bit Bank Pointer.

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the 4 Most Significant bits of a location's address; the instruction itself includes the 8 Least Significant bits. Only the four lower bits of the BSR are implemented (BSR<3:0>). The upper four bits are unused; they will always read '0' and cannot be written to. The BSR can be loaded directly by using the MOVLB instruction.

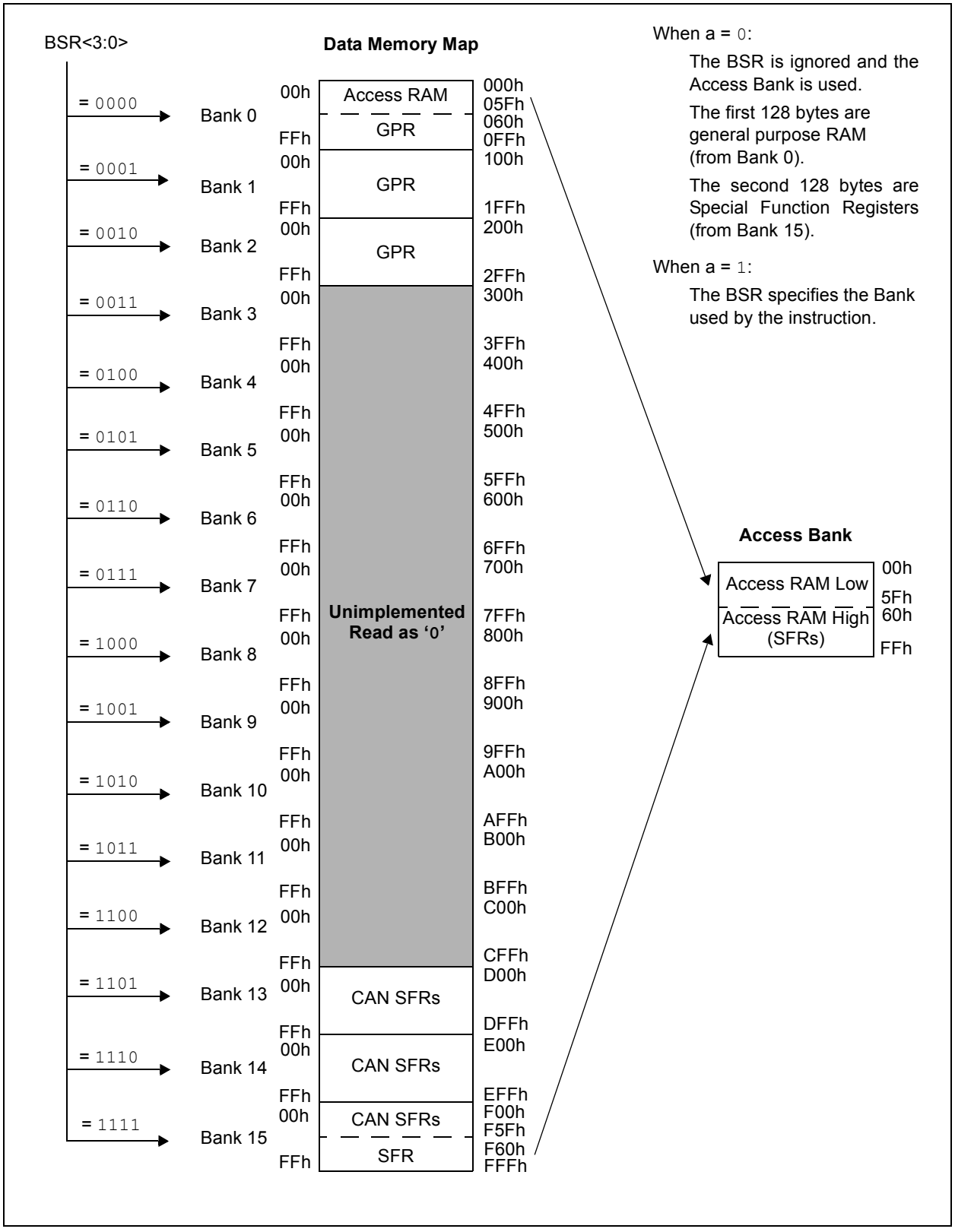
The value of the BSR indicates the bank in data memory; the 8 bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 6-7.

Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h, while the BSR is 0Fh will end up resetting the Program Counter.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory map in Figure 6-6 indicates which banks are implemented.

In the core PIC18 instruction set, only the MOVFF instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the Access Bank to locate their target registers.

**FIGURE 6-5: DATA MEMORY MAP FOR PIC18F2480/4480 DEVICES**





© 2009 Microchip Technology Inc.

**FIGURE 6-6: DATA MEMORY MAP FOR PIC18F2580/4580 DEVICES**

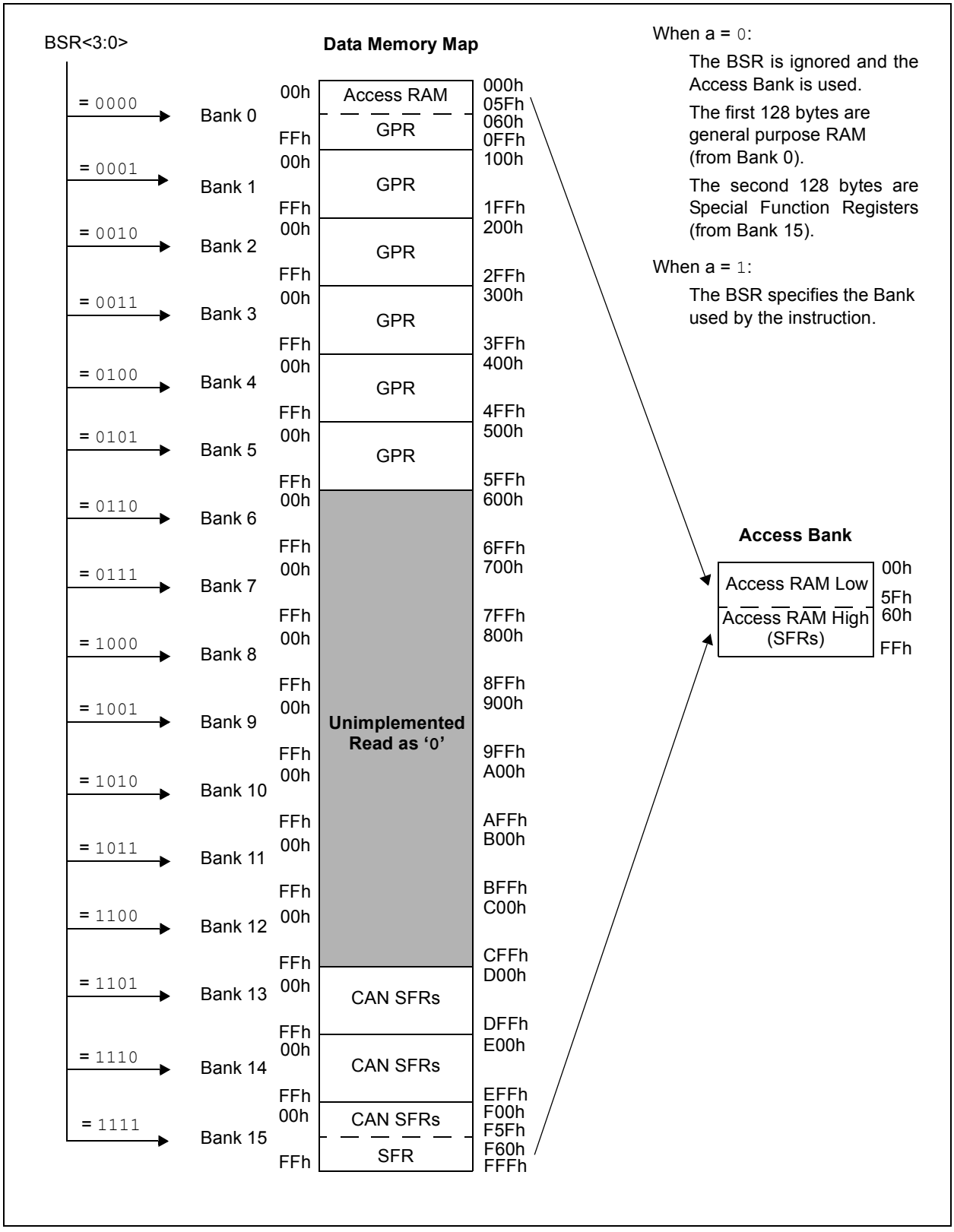
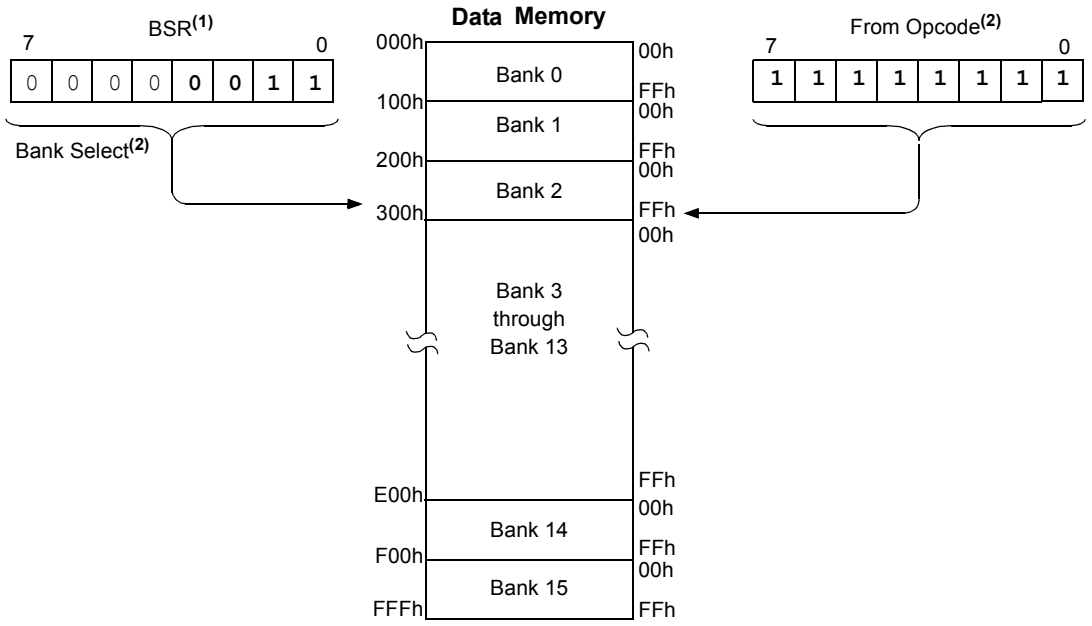


**FIGURE 6-7: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)**




**Note 1:** The Access RAM 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.

**2:** The `MOVFF` instruction embeds the entire 12-bit address in the instruction.

### 6.3.2 ACCESS BANK

While the use of the BSR with an embedded 8-bit address allows users to address the entire range of data memory, it also means that the user must always ensure that the correct bank is selected. Otherwise, data may be read from or written to the wrong location. This can be disastrous if a GPR is the intended target of an operation, but an SFR is written to instead. Verifying and/or changing the BSR for each read or write to data memory can become very inefficient.

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 128 bytes of memory (00h-7Fh) in Bank 0 and the last 128 bytes of memory (80h-FFh) in Block 15. The lower half is known as the “Access RAM” and is composed of GPRs. The upper half is where the device’s SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figure 6-6).

The Access Bank is used by core PIC18 instructions that include the Access RAM bit (the ‘a’ parameter in the instruction). When ‘a’ is equal to ‘`1`’, the instruction uses the BSR and the 8-bit address included in the opcode for the data memory address. When ‘a’ is ‘`0`’ however, the instruction is forced to use the Access Bank address map; the current value of the BSR is ignored entirely.

Using this “forced” addressing allows the instruction to operate on a data address in a single cycle, without updating the BSR first. For 8-bit addresses of 80h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 80h is a good place for data values that the user might need to access rapidly, such as immediate computational results or common program variables. Access RAM also allows for faster and more code efficient context saving and switching of variables.

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = `1`). This is discussed in more detail in **Section 6.6.3 “Mapping the Access Bank in Indexed Literal Offset Mode”**.

### 6.3.3 GENERAL PURPOSE REGISTER FILE

PIC18 devices may have banked memory in the GPR area. This is data RAM, which is available for use by all instructions. GPRs start at the bottom of Bank 0 (address 000h) and grow upwards towards the bottom of the SFR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.


© 2009 Microchip Technology Inc.

### 6.3.4 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. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top half of Bank 15 (F80h to FFFh). A list of these registers is given in Table 6-1 and Table 6-2.

The SFRs can be classified into two sets: those associated with the "core" device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The reset and interrupt registers are described in their respective chapters, while the ALU's STATUS register is described later in this section. Registers related to the operation of a peripheral feature are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as '0's.

**TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2480/2580/4480/4580 DEVICES**

| Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| FFFh | TOSU | FDFh | INDF2(3) | FBFh | ECCPR1H | F9Fh | IPR1 |
| FFEh | TOSH | FDEh | POSTINC2(3) | FBEh | ECCPR1L | F9Eh | PIR1 |
| FFDh | TOSL | FDDh | POSTDEC2(3) | FBDh | CCP1CON | F9Dh | PIE1 |
| FFCh | STKPTR | FDCh | PREINC2(3) | FBCh | CCPR2H(1) | F9Ch | — |
| FFBh | PCLATU | FDBh | PLUSW2(3) | FBBh | CCPR2L(1) | F9Bh | OSCTUNE |
| FFAh | PCLATH | FDAh | FSR2H | FBAh | ECCP1CON(1) | F9Ah | — |
| FF9h | PCL | FD9h | FSR2L | FB9h | — | F99h | — |
| FF8h | TBLPTRU | FD8h | STATUS | FB8h | BAUDCON | F98h | — |
| FF7h | TBLPTRH | FD7h | TMR0H | FB7h | ECCP1DEL | F97h | — |
| FF6h | TBLPTRL | FD6h | TMR0L | FB6h | ECCP1AS(1) | F96h | TRISE(1) |
| FF5h | TABLAT | FD5h | T0CON | FB5h | CVRCON(1) | F95h | TRISD(1) |
| FF4h | PRODH | FD4h | — | FB4h | CMCON | F94h | TRISC |
| FF3h | PRODL | FD3h | OSCCON | FB3h | TMR3H | F93h | TRISB |
| FF2h | INTCON | FD2h | HLVDCON | FB2h | TMR3L | F92h | TRISA |
| FF1h | INTCON2 | FD1h | WDTCON | FB1h | T3CON | F91h | — |
| FF0h | INTCON3 | FD0h | RCON | FB0h | SPBRGH | F90h | — |
| FEFh | INDF0(3) | FCFh | TMR1H | FAFh | SPBRG | F8Fh | — |
| FEEh | POSTINC0(3) | FCEh | TMR1L | FAEh | RCREG | F8Eh | — |
| FEDh | POSTDEC0(3) | FCDh | T1CON | FADh | TXREG | F8Dh | LATE(1) |
| FECh | PREINC0(3) | FCCh | TMR2 | FACh | TXSTA | F8Ch | LATD(1) |
| FEBh | PLUSW0(3) | FCBh | PR2 | FABh | RCSTA | F8Bh | LATC |
| FEAh | FSR0H | FCAh | T2CON | FAAh | — | F8Ah | LATB |
| FE9h | FSR0L | FC9h | SSPBUF | FA9h | EEADR | F89h | LATA |
| FE8h | WREG | FC8h | SSPADD | FA8h | EEDATA | F88h | — |
| FE7h | INDF1(3) | FC7h | SSPSTAT | FA7h | EECON2(3) | F87h | — |
| FE6h | POSTINC1(3) | FC6h | SSPCON1 | FA6h | EECON1 | F86h | — |
| FE5h | POSTDEC1(3) | FC5h | SSPCON2 | FA5h | IPR3 | F85h | — |
| FE4h | PREINC1(3) | FC4h | ADRESH | FA4h | PIR3 | F84h | PORTE |
| FE3h | PLUSW1(3) | FC3h | ADRESL | FA3h | PIE3 | F83h | PORTD(1) |
| FE2h | FSR1H | FC2h | ADCON0 | FA2h | IPR2 | F82h | PORTC |
| FE1h | FSR1L | FC1h | ADCON1 | FA1h | PIR2 | F81h | PORTB |
| FE0h | BSR | FC0h | ADCON2 | FA0h | PIE2 | F80h | PORTA |

**Note 1:** Registers available only on PIC18F4X80 devices; otherwise, the registers read as '0'.
**2:** When any TX_ENn bit in RX_TX_SELn is set, then the corresponding bit in this register has transmit properties.
**3:** This is not a physical register.

**TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2480/2580/4480/4580 DEVICES (CONTINUED)**

| Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| F7Fh | — | F5Fh | CANCON_RO0 | F3Fh | CANCON_RO2 | F1Fh | RXM1EIDL |
| F7Eh | — | F5Eh | CANSTAT_RO0 | F3Eh | CANSTAT_RO2 | F1Eh | RXM1EIDH |
| F7Dh | — | F5Dh | RXB1D7 | F3Dh | TXB1D7 | F1Dh | RXM1SIDL |
| F7Ch | — | F5Ch | RXB1D6 | F3Ch | TXB1D6 | F1Ch | RXM1SIDH |
| F7Bh | — | F5Bh | RXB1D5 | F3Bh | TXB1D5 | F1Bh | RXM0EIDL |
| F7Ah | — | F5Ah | RXB1D4 | F3Ah | TXB1D4 | F1Ah | RXM0EIDH |
| F79h | — | F59h | RXB1D3 | F39h | TXB1D3 | F19h | RXM0SIDL |
| F78h | — | F58h | RXB1D2 | F38h | TXB1D2 | F18h | RXM0SIDH |
| F77h | ECANCON | F57h | RXB1D1 | F37h | TXB1D1 | F17h | RXF5EIDL |
| F76h | TXERRCNT | F56h | RXB1D0 | F36h | TXB1D0 | F16h | RXF5EIDH |
| F75h | RXERRCNT | F55h | RXB1DLC | F35h | TXB1DLC | F15h | RXF5SIDL |
| F74h | COMSTAT | F54h | RXB1EIDL | F34h | TXB1EIDL | F14h | RXF5SIDH |
| F73h | CIOCON | F53h | RXB1EIDH | F33h | TXB1EIDH | F13h | RXF4EIDL |
| F72h | BRGCON3 | F52h | RXB1SIDL | F32h | TXB1SIDL | F12h | RXF4EIDH |
| F71h | BRGCON2 | F51h | RXB1SIDH | F31h | TXB1SIDH | F11h | RXF4SIDL |
| F70h | BRGCON1 | F50h | RXB1CON | F30h | TXB1CON | F10h | RXF4SIDH |
| F6Fh | CANCON | F4Fh | CANCON_RO1 | F2Fh | CANCON_RO3 | F0Fh | RXF3EIDL |
| F6Eh | CANSTAT | F4Eh | CANSTAT_RO1 | F2Eh | CANSTAT_RO3 | F0Eh | RXF3EIDH |
| F6Dh | RXB0D7 | F4DH | TXB0D7 | F2Dh | TXB2D7 | F0Dh | RXF3SIDL |
| F6Ch | RXB0D6 | F4Ch | TXB0D6 | F2Ch | TXB2D6 | F0Ch | RXF3SIDH |
| F6Bh | RXB0D5 | F4Bh | TXB0D5 | F2Bh | TXB2D5 | F0Bh | RXF2EIDL |
| F6Ah | RXB0D4 | F4Ah | TXB0D4 | F2Ah | TXB2D4 | F0Ah | RXF2EIDH |
| F69h | RXB0D3 | F49h | TXB0D3 | F29h | TXB2D3 | F09h | RXF2SIDL |
| F68h | RXB0D2 | F48h | TXB0D2 | F28h | TXB2D2 | F08h | RXF2SIDH |
| F67h | RXB0D1 | F47h | TXB0D1 | F27h | TXB2D1 | F07h | RXF1EIDL |
| F66h | RXB0D0 | F46h | TXB0D0 | F26h | TXB2D0 | F06h | RXF1EIDH |
| F65h | RXB0DLC | F45h | TXB0DLC | F25h | TXB2DLC | F05h | RXF1SIDL |
| F64h | RXB0EIDL | F44h | TXB0EIDL | F24h | TXB2EIDL | F04h | RXF1SIDH |
| F63h | RXB0EIDH | F43h | TXB0EIDH | F23h | TXB2EIDH | F03h | RXF0EIDL |
| F62h | RXB0SIDL | F42h | TXB0SIDL | F22h | TXB2SIDL | F02h | RXF0EIDH |
| F61h | RXB0SIDH | F41h | TXB0SIDH | F21h | TXB2SIDH | F01h | RXF0SIDL |
| F60h | RXB0CON | F40h | TXB0CON | F20h | TXB2CON | F00h | RXF0SIDH |

**Note 1:** Registers available only on PIC18F4X80 devices; otherwise, the registers read as '0'.
**2:** When any TX_ENn bit in RX_TX_SELn is set, then the corresponding bit in this register has transmit properties.
**3:** This is not a physical register.


© 2009 Microchip Technology Inc.

**TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2480/2580/4480/4580 DEVICES (CONTINUED)**

| Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| EFFh | — | EDFh | — | EBFh | — | E9Fh | — |
| EFEh | — | EDEh | — | EBEh | — | E9Eh | — |
| EFDh | — | EDDh | — | EBDh | — | E9Dh | — |
| EFCh | — | EDCh | — | EBCh | — | E9Ch | — |
| EFBh | — | EDBh | — | EBBh | — | E9Bh | — |
| EFAh | — | EDAh | — | EBAh | — | E9Ah | — |
| EF9h | — | ED9h | — | EB9h | — | E99h | — |
| EF8h | — | ED8h | — | EB8h | — | E98h | — |
| EF7h | — | ED7h | — | EB7h | — | E97h | — |
| EF6h | — | ED6h | — | EB6h | — | E96h | — |
| EF5h | — | ED5h | — | EB5h | — | E95h | — |
| EF4h | — | ED4h | — | EB4h | — | E94h | — |
| EF3h | — | ED3h | — | EB3h | — | E93h | — |
| EF2h | — | ED2h | — | EB2h | — | E92h | — |
| EF1h | — | ED1h | — | EB1h | — | E91h | — |
| EF0h | — | ED0h | — | EB0h | — | E90h | — |
| EEFh | — | ECFh | — | EAFh | — | E8Fh | — |
| EEEh | — | ECEh | — | EAEh | — | E8Eh | — |
| EEDh | — | ECDh | — | EADh | — | E8Dh | — |
| EECh | — | ECCh | — | EACh | — | E8Ch | — |
| EEBh | — | ECBh | — | EABh | — | E8Bh | — |
| EEAh | — | ECAh | — | EAAh | — | E8Ah | — |
| EE9h | — | EC9h | — | EA9h | — | E89h | — |
| EE8h | — | EC8h | — | EA8h | — | E88h | — |
| EE7h | — | EC7h | — | EA7h | — | E87h | — |
| EE6h | — | EC6h | — | EA6h | — | E86h | — |
| EE5h | — | EC5h | — | EA5h | — | E85h | — |
| EE4h | — | EC4h | — | EA4h | — | E84h | — |
| EE3h | — | EC3h | — | EA3h | — | E83h | — |
| EE2h | — | EC2h | — | EA2h | — | E82h | — |
| EE1h | — | EC1h | — | EA1h | — | E81h | — |
| EE0h | — | EC0h | — | EA0h | — | E80h | — |

**Note 1:** Registers available only on PIC18F4X80 devices; otherwise, the registers read as '0'.
**2:** When any TX_ENn bit in RX_TX_SELn is set, then the corresponding bit in this register has transmit properties.
**3:** This is not a physical register.

**TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2480/2580/4480/4580 DEVICES (CONTINUED)**

| Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| E7Fh | CANCON_RO4 | E6Fh | CANCON_RO5 | E5Fh | CANCON_RO6 | E4Fh | CANCON_RO7 |
| E7Eh | CANSTAT_RO4 | E6Eh | CANSTAT_RO5 | E5Eh | CANSTAT_RO6 | E4Eh | CANSTAT_RO7 |
| E7Dh | B5D7(2) | E6Dh | B4D7(2) | E5Dh | B3D7(2) | E4Dh | B2D7(2) |
| E7Ch | B5D6(2) | E6Ch | B4D6(2) | E5Ch | B3D6(2) | E4Ch | B2D6(2) |
| E7Bh | B5D5(2) | E6Bh | B4D5(2) | E5Bh | B3D5(2) | E4Bh | B2D5(2) |
| E7Ah | B5D4(2) | E6Ah | B4D4(2) | E5Ah | B3D4(2) | E4Ah | B2D4(2) |
| E79h | B5D3(2) | E69h | B4D3(2) | E59h | B3D3(2) | E49h | B2D3(2) |
| E78h | B5D2(2) | E68h | B4D2(2) | E58h | B3D2(2) | E48h | B2D2(2) |
| E77h | B5D1(2) | E67h | B4D1(2) | E57h | B3D1(2) | E47h | B2D1(2) |
| E76h | B5D0(2) | E66h | B4D0(2) | E56h | B3D0(2) | E46h | B2D0(2) |
| E75h | B5DLC(2) | E65h | B4DLC(2) | E55h | B3DLC(2) | E45h | B2DLC(2) |
| E74h | B5EIDL(2) | E64h | B4EIDL(2) | E54h | B3EIDL(2) | E44h | B2EIDL(2) |
| E73h | B5EIDH(2) | E63h | B4EIDH(2) | E53h | B3EIDH(2) | E43h | B2EIDH(2) |
| E72h | B5SIDL(2) | E62h | B4SIDL(2) | E52h | B3SIDL(2) | E42h | B2SIDL(2) |
| E71h | B5SIDH(2) | E61h | B4SIDH(2) | E51h | B3SIDH(2) | E41h | B2SIDH(2) |
| E70h | B5CON (2) | E60h | B4CON(2) | E50h | B3CON(2) | E40h | B2CON(2) |
| E3Fh | CANCON_RO8 | E2Fh | CANCON_RO9 | E1Fh | — | E0Fh | — |
| E3Eh | CANSTAT_RO8 | E2Eh | CANSTAT_RO9 | E1Eh | — | E0Eh | — |
| E3Dh | B1D7(2) | E2Dh | B0D7(2) | E1Dh | — | E0Dh | — |
| E3Ch | B1D6(2) | E2Ch | B0D6(2) | E1Ch | — | E0Ch | — |
| E3Bh | B1D5(2) | E2Bh | B0D5(2) | E1Bh | — | E0Bh | — |
| E3Ah | B1D4(2) | E2Ah | B0D4(2) | E1Ah | — | E0Ah | — |
| E39h | B1D3(2) | E29h | B0D3(2) | E19h | — | E09h | — |
| E38h | B1D2(2) | E28h | B0D2(2) | E18h | — | E08h | — |
| E37h | B1D1(2) | E27h | B0D1(2) | E17h | — | E07h | — |
| E36h | B1D0(2) | E26h | B0D0(2) | E16h | — | E06h | — |
| E35h | B1DLC(2) | E25h | B0DLC(2) | E15h | — | E05h | — |
| E34h | B1EIDL(2) | E24h | B0EIDL(2) | E14h | — | E04h | — |
| E33h | B1EIDH(2) | E23h | B0EIDH(2) | E13h | — | E03h | — |
| E32h | B1SIDL(2) | E22h | B0SIDL(2) | E12h | — | E02h | — |
| E31h | B1SIDH(2) | E21h | B0SIDH(2) | E11h | — | E01h | — |
| E30h | B1CON(2) | E20h | B0CON(2) | E10h | — | E00h | — |

**Note 1:** Registers available only on PIC18F4X80 devices; otherwise, the registers read as '0'.
**2:** When any TX_ENn bit in RX_TX_SELn is set, then the corresponding bit in this register has transmit properties.
**3:** This is not a physical register.


© 2009 Microchip Technology Inc.

**TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2480/2580/4480/4580 DEVICES (CONTINUED)**

| Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| DFFh | — | DDFh | — | DBFh | — | D9Fh | — |
| DFEh | — | DDEh | — | DBEh | — | D9Eh | — |
| DFDh | — | DDDh | — | DBDh | — | D9Dh | — |
| DFCh | TXBIE | DDCh | — | DBCh | — | D9Ch | — |
| DFBh | — | DDBh | — | DBBh | — | D9Bh | — |
| DFAh | BIE0 | DDAh | — | DBAh | — | D9Ah | — |
| DF9h | — | DD9h | — | DB9h | — | D99h | — |
| DF8h | BSEL0 | DD8h | SDFLC | DB8h | — | D98h | — |
| DF7h | — | DD7h | — | DB7h | — | D97h | — |
| DF6h | — | DD6h | — | DB6h | — | D96h | — |
| DF5h | — | DD5h | RXFCON1 | DB5h | — | D95h | — |
| DF4h | — | DD4h | RXFCON0 | DB4h | — | D94h | — |
| DF3h | MSEL3 | DD3h | — | DB3h | — | D93h | RXF15EIDL |
| DF2h | MSEL2 | DD2h | — | DB2h | — | D92h | RXF15EIDH |
| DF1h | MSEL1 | DD1h | — | DB1h | — | D91h | RXF15SIDL |
| DF0h | MSEL0 | DD0h | — | DB0h | — | D90h | RXF15SIDH |
| DEFh | — | DCFh | — | DAFh | — | D8Fh | — |
| DEEh | — | DCEh | — | DAEh | — | D8Eh | — |
| DEDh | — | DCDh | — | DADh | — | D8Dh | — |
| DECh | — | DCCh | — | DACh | — | D8Ch | — |
| DEBh | — | DCBh | — | DABh | — | D8Bh | RXF14EIDL |
| DEAh | — | DCAh | — | DAAh | — | D8Ah | RXF14EIDH |
| DE9h | — | DC9h | — | DA9h | — | D89h | RXF14SIDL |
| DE8h | — | DC8h | — | DA8h | — | D88h | RXF14SIDH |
| DE7h | RXFBCON7 | DC7h | — | DA7h | — | D87h | RXF13EIDL |
| DE6h | RXFBCON6 | DC6h | — | DA6h | — | D86h | RXF13EIDH |
| DE5h | RXFBCON5 | DC5h | — | DA5h | — | D85h | RXF13SIDL |
| DE4h | RXFBCON4 | DC4h | — | DA4h | — | D84h | RXF13SIDH |
| DE3h | RXFBCON3 | DC3h | — | DA3h | — | D83h | RXF12EIDL |
| DE2h | RXFBCON2 | DC2h | — | DA2h | — | D82h | RXF12EIDH |
| DE1h | RXFBCON1 | DC1h | — | DA1h | — | D81h | RXF12SIDL |
| DE0h | RXFBCON0 | DC0h | — | DA0h | — | D80h | RXF12SIDH |

**Note 1:** Registers available only on PIC18F4X80 devices; otherwise, the registers read as '0'.
**2:** When any TX_ENn bit in RX_TX_SELn is set, then the corresponding bit in this register has transmit properties.
**3:** This is not a physical register.

**TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2480/2580/4480/4580 DEVICES (CONTINUED)**

| Address | Name |
|---|---|
| D7Fh | — |
| D7Eh | — |
| D7Dh | — |
| D7Ch | — |
| D7Bh | RXF11EIDL |
| D7Ah | RXF11EIDH |
| D79h | RXF11SIDL |
| D78h | RXF11SIDH |
| D77h | RXF10EIDL |
| D76h | RXF10EIDH |
| D75h | RXF10SIDL |
| D74h | RXF10SIDH |
| D73h | RXF9EIDL |
| D72h | RXF9EIDH |
| D71h | RXF9SIDL |
| D70h | RXF9SIDH |
| D6Fh | — |
| D6Eh | — |
| D6Dh | — |
| D6Ch | — |
| D6Bh | RXF8EIDL |
| D6Ah | RXF8EIDH |
| D69h | RXF8SIDL |
| D68h | RXF8SIDH |
| D67h | RXF7EIDL |
| D66h | RXF7EIDH |
| D65h | RXF7SIDL |
| D64h | RXF7SIDH |
| D63h | RXF6EIDL |
| D62h | RXF6EIDH |
| D61h | RXF6SIDL |
| D60h | RXF6SIDH |

**Note 1:** Registers available only on PIC18F4X80 devices; otherwise, the registers read as '0'.
**2:** When any TX_ENn bit in RX_TX_SELn is set, then the corresponding bit in this register has transmit properties.
**3:** This is not a physical register.


© 2009 Microchip Technology Inc.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580)**

| 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 | 55, 68 |
| TOSH | Top-of-Stack High Byte (TOS<15:8>) | | | | | | | | 0000 0000 | 55, 68 |
| TOSL | Top-of-Stack Low Byte (TOS<7:0>) | | | | | | | | 0000 0000 | 55, 68 |
| STKPTR | STKFUL | STKUNF | — | Return Stack Pointer | | | | | 00-0 0000 | 55, 69 |
| PCLATU | — | — | bit 21(1) | Holding Register for PC<20:16> | | | | | ---0 0000 | 55, 68 |
| PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000 0000 | 55, 68 |
| PCL | PC Low Byte (PC<7:0>) | | | | | | | | 0000 0000 | 55, 68 |
| TBLPTRU | — | — | bit 21 | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | --00 0000 | 55, 109 |
| TBLPTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 | 55, 109 |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 | 55, 109 |
| TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 | 55, 109 |
| PRODH | Product Register High Byte | | | | | | | | xxxx xxxx | 55, 117 |
| PRODL | Product Register Low Byte | | | | | | | | xxxx xxxx | 55, 117 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 55, 121 |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RBIP | 1111 -1-1 | 55, 122 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 11-0 0-00 | 55, 123 |
| INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) | | | | | | | | N/A | 55, 96 |
| POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | N/A | 55, 97 |
| POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) | | | | | | | | N/A | 55, 97 |
| PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) | | | | | | | | N/A | 55, 97 |
| PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register), value of FSR0 offset by W | | | | | | | | N/A | 55, 97 |
| FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0 High | | | | ---- xxxx | 55, 96 |
| FSR0L | Indirect Data Memory Address Pointer 0 Low Byte | | | | | | | | xxxx xxxx | 55, 96 |
| WREG | Working Register | | | | | | | | xxxx xxxx | 55 |
| INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) | | | | | | | | N/A | 55, 96 |
| POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | N/A | 55, 97 |
| POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) | | | | | | | | N/A | 55, 97 |
| PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) | | | | | | | | N/A | 55, 97 |
| PLUSW1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register), value of FSR1 offset by W | | | | | | | | N/A | 55, 97 |
| FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1 High | | | | ---- xxxx | 55, 96 |
| FSR1L | Indirect Data Memory Address Pointer 1 Low Byte | | | | | | | | xxxx xxxx | 55, 96 |
| BSR | — | — | — | — | Bank Select Register | | | | ---- 0000 | 56, 73 |
| INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) | | | | | | | | N/A | 56, 96 |
| POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | N/A | 56, 97 |
| POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) | | | | | | | | N/A | 56, 97 |
| PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) | | | | | | | | N/A | 56, 97 |
| PLUSW2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register), value of FSR2 offset by W | | | | | | | | N/A | 56, 97 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2 High | | | | ---- xxxx | 56, 96 |
| FSR2L | Indirect Data Memory Address Pointer 2 Low Byte | | | | | | | | xxxx xxxx | 56, 96 |
| STATUS | — | — | — | N | OV | Z | DC | C | ---x xxxx | 56, 94 |
| TMR0H | Timer0 Register High Byte | | | | | | | | 0000 0000 | 56, 153 |
| TMR0L | Timer0 Register Low Byte | | | | | | | | xxxx xxxx | 56, 153 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 1111 1111 | 56, 153 |
| OSCCON | IDLEN | IRCF2 | IRCF1 | IRCF0 | OSTS | IOFS | SCS1 | SCS0 | 0000 q000 | 36, 56 |
| HLVDCON | VDIRMAG | — | IRVST | HLVDEN | HLVDL3 | HLVDL2 | HLVDL1 | HLVDL0 | 0-00 0101 | 56, 273 |
| WDTCON | — | — | — | — | — | — | — | SWDTEN | --- ---0 | 56, 359 |
| RCON | IPEN | SBOREN(2) | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 0q-1 11q0 | 56, 133 |
| TMR1H | Timer1 Register High Byte | | | | | | | | xxxx xxxx | 56, 159 |
| TMR1L | Timer1 Register Low Byte | | | | | | | | 0000 0000 | 56, 159 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{T1SYNC}$ | TMR1CS | TMR1ON | 0000 0000 | 56, 155 |
| TMR2 | Timer2 Register | | | | | | | | 1111 1111 | 56, 162 |
| PR2 | Timer2 Period Register | | | | | | | | -000 0000 | 56, 159 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | 56, 161 |
| SSPBUF | MSSP Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | 56, 199 |
| SSPADD | MSSP Address Register in $I^2C$ Slave Mode. MSSP Baud Rate Reload Register in $I^2C$ Master Mode. | | | | | | | | 0000 0000 | 56, 199 |
| SSPSTAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 0000 0000 | 56, 201 |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 56, 202 |
| SSPCON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | 56, 203 |
| ADRESH | A/D Result Register High Byte | | | | | | | | xxxx xxxx | 56, 262 |
| ADRESL | A/D Result Register Low Byte | | | | | | | | xxxx xxxx | 56, 262 |
| ADCON0 | — | — | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{DONE}$ | ADON | --00 0000 | 56, 253 |
| ADCON1 | — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | --00 0qqq | 56, 254 |
| ADCON2 | ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 0-00 0000 | 57, 255 |
| CCPR1H | Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | 57, 172 |
| CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | 57, 172 |
| CCP1CON | — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | --00 0000 | 57, 167 |
| ECCPR1H(9) | Enhanced Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | 57, 171 |
| ECCPR1L(9) | Enhanced Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | 57, 171 |
| ECCP1CON(9) | EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 | 0000 0000 | 57, 172 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 01-0 0000 | 57, 234 |
| ECCP1DEL(9) | PRSEN | PDC6(3) | PDC5(3) | PDC4(3) | PDC3(3) | PDC2(3) | PDC1(3) | PDC0(3) | 0000 0000 | 57, 187 |
| ECCP1AS(9) | ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1(3) | PSSBD0(3) | 0000 0000 | 57, 187 |
| CVRCON(9) | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 0000 0000 | 57, 269 |
| CMCON(9) | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 0000 0000 | 57, 263 |
| TMR3H | Timer3 Register High Byte | | | | | | | | xxxx xxxx | 57, 165 |
| TMR3L | Timer3 Register Low Byte | | | | | | | | xxxx xxxx | 57, 165 |
| T3CON | RD16 | T3ECCP1(9) | T3CKPS1 | T3CKPS0 | T3CCP1(9) | $\overline{T3SYNC}$ | TMR3CS | TMR3ON | 0000 0000 | 57, 165 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.


© 2009 Microchip Technology Inc.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| SPBRGH | EUSART Baud Rate Generator High Byte | | | | | | | | 0000 0000 | 57, 236 |
| SPBRG | EUSART Baud Rate Generator | | | | | | | | 0000 0000 | 57, 236 |
| RCREG | EUSART Receive Register | | | | | | | | 0000 0000 | 57, 244 |
| TXREG | EUSART Transmit Register | | | | | | | | 0000 0000 | 57, 241 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 57, 243 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 57, 243 |
| EEADR | EEPROM Address Register | | | | | | | | 0000 0000 | 57, 111 |
| EEDATA | EEPROM Data Register | | | | | | | | 0000 0000 | 57, 111 |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | 0000 0000 | 57, 111 |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | xx-0 x000 | 57, 111 |
| IPR3<br>Mode 0 | IRXIP | WAKIP | ERRIP | TXB2IP | TXB1IP | TXB0IP | RXB1IP | RXB0IP | 1111 1111 | 57, 132 |
| IPR3<br>Mode 1, 2 | IRXIP | WAKIP | ERRIP | TXBnIP | TXB1IP(8) | TXB0IP(8) | RXBnIP | FIFOWMIP | 1111 1111 | 57, 132 |
| PIR3<br>Mode 0 | IRXIF | WAKIF | ERRIF | TXB2IF | TXB1IF | TXB0IF | RXB1IF | RXB0IF | 0000 0000 | 57, 126 |
| PIR3<br>Mode 1, 2 | IRXIF | WAKIF | ERRIF | TXBnIF | TXB1IF(8) | TXB0IF(8) | RXBnIF | FIFOWMIF | 0000 0000 | 57, 126 |
| PIE3<br>Mode 0 | IRXIE | WAKIE | ERRIE | TXB2IE | TXB1IE | TXB0IE | RXB1IE | RXB0IE | 0000 0000 | 57, 129 |
| PIE3<br>Mode 1, 2 | IRXIE | WAKIE | ERRIE | TXBnIE | TXB1IE(8) | TXB0IE(8) | RXBnIE | FIFOMWIE | 0000 0000 | 57, 129 |
| IPR2 | OSCFIP | CMIP(9) | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP(9) | 11-1 1111 | 57, 131 |
| PIR2 | OSCFIF | CMIF(9) | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF(9) | 00-0 0000 | 58, 125 |
| PIE2 | OSCFIE | CMIE(9) | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE(9) | 00-0 0000 | 58, 128 |
| IPR1 | PSPIP(3) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 58, 130 |
| PIR1 | PSPIF(3) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 58, 124 |
| PIE1 | PSPIE(3) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 58, 127 |
| OSCTUNE | INTSRC | PLLEN(4) | — | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 | 0q-0 0000 | 33, 58 |
| TRISE(3) | IBF | OBF | IBOV | PSPMODE | — | TRISE2 | TRISE1 | TRISE0 | 0000 -111 | 58, 146 |
| TRISD(3) | PORTD Data Direction Register | | | | | | | | 1111 1111 | 58, 143 |
| TRISC | PORTC Data Direction Register | | | | | | | | 1111 1111 | 58, 141 |
| TRISB | PORTB Data Direction Register | | | | | | | | 1111 1111 | 58, 138 |
| TRISA | TRISA7(6) | TRISA6(6) | PORTA Data Direction Register | | | | | | 1111 1111 | 58, 135 |
| LATE(3) | — | — | — | — | — | LATE2 | LATE1 | LATE0 | ---- -xxx | 58, 146 |
| LATD(3) | LATD Output Latch Register | | | | | | | | xxxx xxxx | 58, 143 |
| LATC | LATC Output Latch Register | | | | | | | | xxxx xxxx | 58, 141 |
| LATB | LATB Output Latch Register | | | | | | | | xxxx xxxx | 58, 138 |
| LATA | LATA7(6) | LATA6(6) | LATA Output Latch Register | | | | | | xxxx xxxx | 58, 135 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| PORTE(3) | — | — | — | — | RE3(5) | RE2(3) | RE1(3) | RE0(3) | ---- xxxx | 58, 150 |
| PORTD(3) | PORTD Data Direction Register | | | | | | | | xxxx xxxx | 58, 143 |
| PORTC | PORTC Data Direction Register | | | | | | | | xxxx xxxx | 58, 141 |
| PORTB | PORTB Data Direction Register | | | | | | | | xxxx xxxx | 58, 138 |
| PORTA | RA7(6) | RA6(6) | PORTA Data Direction Register | | | | | | xx00 0000 | 58, 135 |
| ECANCON | MDSEL1 | MDSEL0 | FIFOWM | EWIN4 | EWIN3 | EWIN2 | EWIN1 | EWIN0 | 0001 000 | 58, 286 |
| TXERRCNT | TEC7 | TEC6 | TEC5 | TEC4 | TEC3 | TEC2 | TEC1 | TEC0 | 0000 0000 | 58, 291 |
| RXERRCNT | REC7 | REC6 | REC5 | REC4 | REC3 | REC2 | REC1 | REC0 | 0000 0000 | 58, 299 |
| COMSTAT Mode 0 | RXB0OVFL | RXB1OVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN | 0000 0000 | 58, 287 |
| COMSTAT Mode 1 | — | RXBnOVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN | -000 0000 | 58, 287 |
| COMSTAT Mode 2 | $\overline{\text{FIFOEMPTY}}$ | RXBnOVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN | 0000 0000 | 58, 287 |
| CIOCON | — | — | ENDRHI | CANCAP | — | — | — | — | --00 ---- | 58, 320 |
| BRGCON3 | WAKDIS | WAKFIL | — | — | — | SEG2PH2 | SEG2PH1 | SEG2PH0 | 00-- -000 | 59, 319 |
| BRGCON2 | SEG2PHTS | SAM | SEG1PH2 | SEG1PH1 | SEG1PH0 | PRSEG2 | PRSEG1 | PRSEG0 | 0000 0000 | 59, 318 |
| BRGCON1 | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 | 0000 0000 | 59, 317 |
| CANCON Mode 0 | REQOP2 | REQOP1 | REQOP0 | ABAT | WIN2(7) | WIN1(7) | WIN0(7) | —(7) | 1000 000- | 59, 282 |
| CANCON Mode 1 | REQOP2 | REQOP1 | REQOP0 | ABAT | —(7) | —(7) | —(7) | —(7) | 1000 ---- | 59, 282 |
| CANCON Mode 2 | REQOP2 | REQOP1 | REQOP0 | ABAT | FP3(7) | FP2(7) | FP1(7) | FP0(7) | 1000 0000 | 59, 282 |
| CANSTAT Mode 0 | OPMODE2 | OPMODE1 | OPMODE0 | —(7) | ICODE3(7) | ICODE2(7) | ICODE1(7) | —(7) | 000- 0000 | 59, 283 |
| CANSTAT Modes 1, 2 | OPMODE2 | OPMODE1 | OPMODE0 | EICODE4(7) | EICODE3(7) | EICODE2(7) | EICODE1(7) | EICODE0(7) | 0000 0000 | 59, 283 |
| RXB0D7 | RXB0D77 | RXB0D76 | RXB0D75 | RXB0D74 | RXB0D73 | RXB0D72 | RXB0D71 | RXB0D70 | xxxx xxxx | 59, 298 |
| RXB0D6 | RXB0D67 | RXB0D66 | RXB0D65 | RXB0D64 | RXB0D63 | RXB0D62 | RXB0D61 | RXB0D60 | xxxx xxxx | 59, 298 |
| RXB0D5 | RXB0D57 | RXB0D56 | RXB0D55 | RXB0D54 | RXB0D53 | RXB0D52 | RXB0D51 | RXB0D50 | xxxx xxxx | 59, 298 |
| RXB0D4 | RXB0D47 | RXB0D46 | RXB0D45 | RXB0D44 | RXB0D43 | RXB0D42 | RXB0D41 | RXB0D40 | xxxx xxxx | 59, 298 |
| RXB0D3 | RXB0D37 | RXB0D36 | RXB0D35 | RXB0D34 | RXB0D33 | RXB0D32 | RXB0D31 | RXB0D30 | xxxx xxxx | 59, 298 |
| RXB0D2 | RXB0D27 | RXB0D26 | RXB0D25 | RXB0D24 | RXB0D23 | RXB0D22 | RXB0D21 | RXB0D20 | xxxx xxxx | 59, 298 |
| RXB0D1 | RXB0D17 | RXB0D16 | RXB0D15 | RXB0D14 | RXB0D13 | RXB0D12 | RXB0D11 | RXB0D10 | xxxx xxxx | 59, 298 |
| RXB0D0 | RXB0D07 | RXB0D06 | RXB0D05 | RXB0D04 | RXB0D03 | RXB0D02 | RXB0D01 | RXB0D00 | xxxx xxxx | 59, 298 |
| RXB0DLC | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 59, 298 |
| RXB0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 59, 297 |
| RXB0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 59, 297 |
| RXB0SIDL | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 59, 297 |
| RXB0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 59, 296 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.


© 2009 Microchip Technology Inc.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| RXB0CON Mode 0 | RXFUL | RXM1 | RXM0(7) | —(7) | RXRTRRO(7) | RXBODBEN(7) | JTOFF(7) | FILHIT0(7) | 000- 0000 | 59, 293 |
| RXB0CON Mode 1, 2 | RXFUL | RXM1 | RTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 59, 293 |
| RXB1D7 | RXB1D77 | RXB1D76 | RXB1D75 | RXB1D74 | RXB1D73 | RXB1D72 | RXB1D71 | RXB1D70 | xxxx xxxx | 59, 298 |
| RXB1D6 | RXB1D67 | RXB1D66 | RXB1D65 | RXB1D64 | RXB1D63 | RXB1D62 | RXB1D61 | RXB1D60 | xxxx xxxx | 59, 298 |
| RXB1D5 | RXB1D57 | RXB1D56 | RXB1D55 | RXB1D54 | RXB1D53 | RXB1D52 | RXB1D51 | RXB1D50 | xxxx xxxx | 59, 298 |
| RXB1D4 | RXB1D47 | RXB1D46 | RXB1D45 | RXB1D44 | RXB1D43 | RXB1D42 | RXB1D41 | RXB1D40 | xxxx xxxx | 59, 298 |
| RXB1D3 | RXB1D37 | RXB1D36 | RXB1D35 | RXB1D34 | RXB1D33 | RXB1D32 | RXB1D31 | RXB1D30 | xxxx xxxx | 59, 298 |
| RXB1D2 | RXB1D27 | RXB1D26 | RXB1D25 | RXB1D24 | RXB1D23 | RXB1D22 | RXB1D21 | RXB1D20 | xxxx xxxx | 59, 298 |
| RXB1D1 | RXB1D17 | RXB1D16 | RXB1D15 | RXB1D14 | RXB1D13 | RXB1D12 | RXB1D11 | RXB1D10 | xxxx xxxx | 59, 298 |
| RXB1D0 | RXB1D07 | RXB1D06 | RXB1D05 | RXB1D04 | RXB1D03 | RXB1D02 | RXB1D01 | RXB1D00 | xxxx xxxx | 59, 298 |
| RXB1DLC | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 59, 298 |
| RXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 59, 297 |
| RXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 59, 297 |
| RXB1SIDL | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx xxxx | 59, 297 |
| RXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 60, 296 |
| RXB1CON Mode 0 | RXFUL | RXM1 | RXM0(7) | —(7) | RXRTRRO(7) | FILHIT2(7) | FILHIT1(7) | FILHIT0(7) | 000- 0000 | 60, 293 |
| RXB1CON Mode 1, 2 | RXFUL | RXM1 | RTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 60, 293 |
| TXB0D7 | TXB0D77 | TXB0D76 | TXB0D75 | TXB0D74 | TXB0D73 | TXB0D72 | TXB0D71 | TXB0D70 | xxxx xxxx | 60, 290 |
| TXB0D6 | TXB0D67 | TXB0D66 | TXB0D65 | TXB0D64 | TXB0D63 | TXB0D62 | TXB0D61 | TXB0D60 | xxxx xxxx | 60, 290 |
| TXB0D5 | TXB0D57 | TXB0D56 | TXB0D55 | TXB0D54 | TXB0D53 | TXB0D52 | TXB0D51 | TXB0D50 | xxxx xxxx | 60, 290 |
| TXB0D4 | TXB0D47 | TXB0D46 | TXB0D45 | TXB0D44 | TXB0D43 | TXB0D42 | TXB0D41 | TXB0D40 | xxxx xxxx | 60, 290 |
| TXB0D3 | TXB0D37 | TXB0D36 | TXB0D35 | TXB0D34 | TXB0D33 | TXB0D32 | TXB0D31 | TXB0D30 | xxxx xxxx | 60, 290 |
| TXB0D2 | TXB0D27 | TXB0D26 | TXB0D25 | TXB0D24 | TXB0D23 | TXB0D22 | TXB0D21 | TXB0D20 | xxxx xxxx | 60, 290 |
| TXB0D1 | TXB0D17 | TXB0D16 | TXB0D15 | TXB0D14 | TXB0D13 | TXB0D12 | TXB0D11 | TXB0D10 | xxxx xxxx | 60, 290 |
| TXB0D0 | TXB0D07 | TXB0D06 | TXB0D05 | TXB0D04 | TXB0D03 | TXB0D02 | TXB0D01 | TXB0D00 | xxxx xxxx | 60, 290 |
| TXB0DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 60, 291 |
| TXB0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 60, 290 |
| TXB0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 60, 289 |
| TXB0SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 60, 289 |
| TXB0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 60, 289 |
| TXB0CON | TXBIF | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | 0000 0-00 | 60, 288 |
| TXB1D7 | TXB1D77 | TXB1D76 | TXB1D75 | TXB1D74 | TXB1D73 | TXB1D72 | TXB1D71 | TXB1D70 | xxxx xxxx | 60, 290 |
| TXB1D6 | TXB1D67 | TXB1D66 | TXB1D65 | TXB1D64 | TXB1D63 | TXB1D62 | TXB1D61 | TXB1D60 | xxxx xxxx | 60, 290 |
| TXB1D5 | TXB1D57 | TXB1D56 | TXB1D55 | TXB1D54 | TXB1D53 | TXB1D52 | TXB1D51 | TXB1D50 | xxxx xxxx | 60, 290 |
| TXB1D4 | TXB1D47 | TXB1D46 | TXB1D45 | TXB1D44 | TXB1D43 | TXB1D42 | TXB1D41 | TXB1D40 | xxxx xxxx | 60, 290 |
| TXB1D3 | TXB1D37 | TXB1D36 | TXB1D35 | TXB1D34 | TXB1D33 | TXB1D32 | TXB1D31 | TXB1D30 | xxxx xxxx | 60, 290 |
| TXB1D2 | TXB1D27 | TXB1D26 | TXB1D25 | TXB1D24 | TXB1D23 | TXB1D22 | TXB1D21 | TXB1D20 | xxxx xxxx | 60, 290 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| TXB1D1 | TXB1D17 | TXB1D16 | TXB1D15 | TXB1D14 | TXB1D13 | TXB1D12 | TXB1D11 | TXB1D10 | xxxx xxxx | 60, 290 |
| TXB1D0 | TXB1D07 | TXB1D06 | TXB1D05 | TXB1D04 | TXB1D03 | TXB1D02 | TXB1D01 | TXB1D00 | xxxx xxxx | 60, 290 |
| TXB1DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 60, 291 |
| TXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 60, 290 |
| TXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 60, 289 |
| TXB1SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 60, 289 |
| TXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 60, 289 |
| TXB1CON | TXBIF | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | 0000 0-00 | 60, 288 |
| TXB2D7 | TXB2D77 | TXB2D76 | TXB2D75 | TXB2D74 | TXB2D73 | TXB2D72 | TXB2D71 | TXB2D70 | xxxx xxxx | 60, 290 |
| TXB2D6 | TXB2D67 | TXB2D66 | TXB2D65 | TXB2D64 | TXB2D63 | TXB2D62 | TXB2D61 | TXB2D60 | xxxx xxxx | 61, 290 |
| TXB2D5 | TXB2D57 | TXB2D56 | TXB2D55 | TXB2D54 | TXB2D53 | TXB2D52 | TXB2D51 | TXB2D50 | xxxx xxxx | 61, 290 |
| TXB2D4 | TXB2D47 | TXB2D46 | TXB2D45 | TXB2D44 | TXB2D43 | TXB2D42 | TXB2D41 | TXB2D40 | xxxx xxxx | 61, 290 |
| TXB2D3 | TXB2D37 | TXB2D36 | TXB2D35 | TXB2D34 | TXB2D33 | TXB2D32 | TXB2D31 | TXB2D30 | xxxx xxxx | 61, 290 |
| TXB2D2 | TXB2D27 | TXB2D26 | TXB2D25 | TXB2D24 | TXB2D23 | TXB2D22 | TXB2D21 | TXB2D20 | xxxx xxxx | 61, 290 |
| TXB2D1 | TXB2D17 | TXB2D16 | TXB2D15 | TXB2D14 | TXB2D13 | TXB2D12 | TXB2D11 | TXB2D10 | xxxx xxxx | 61, 290 |
| TXB2D0 | TXB2D07 | TXB2D06 | TXB2D05 | TXB2D04 | TXB2D03 | TXB2D02 | TXB2D01 | TXB2D00 | xxxx xxxx | 61, 290 |
| TXB2DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 61, 291 |
| TXB2EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 61, 290 |
| TXB2EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 61, 289 |
| TXB2SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxxx x-xx | 61, 289 |
| TXB2SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxx- x-xx | 61, 289 |
| TXB2CON | TXBIF | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | 0000 0-00 | 61, 288 |
| RXM1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 61, 310 |
| RXM1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 61, 310 |
| RXM1SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 61, 310 |
| RXM1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 61, 310 |
| RXM0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 61, 310 |
| RXM0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 61, 310 |
| RXM0SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 61, 310 |
| RXM0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 61, 309 |
| RXF5EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 61, 309 |
| RXF5EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 61, 309 |
| RXF5SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 61, 308 |
| RXF5SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 61, 308 |
| RXF4EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 61, 309 |
| RXF4EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 61, 309 |
| RXF4SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 61, 308 |
| RXF4SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 61, 308 |
| RXF3EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 61, 309 |
| RXF3EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 61, 309 |
| RXF3SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 62, 308 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.


© 2009 Microchip Technology Inc.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| RXF3SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 62, 308 |
| RXF2EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 62, 309 |
| RXF2EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 62, 309 |
| RXF2SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 62, 308 |
| RXF2SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 62, 308 |
| RXF1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 62, 309 |
| RXF1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 62, 309 |
| RXF1SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 62, 308 |
| RXF1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 62, 308 |
| RXF0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 62, 309 |
| RXF0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 62, 309 |
| RXF0SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 62, 308 |
| RXF0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 62, 308 |
| B5D7(8) | B5D77 | B5D76 | B5D75 | B5D74 | B5D73 | B5D72 | B5D71 | B5D70 | xxxx xxxx | 62, 305 |
| B5D6(8) | B5D67 | B5D66 | B5D65 | B5D64 | B5D63 | B5D62 | B5D61 | B5D60 | xxxx xxxx | 62, 305 |
| B5D5(8) | B5D57 | B5D56 | B5D55 | B5D54 | B5D53 | B5D52 | B5D51 | B5D50 | xxxx xxxx | 62, 305 |
| B5D4(8) | B5D47 | B5D46 | B5D45 | B5D44 | B5D43 | B5D42 | B5D41 | B5D40 | xxxx xxxx | 62, 305 |
| B5D3(8) | B5D37 | B5D36 | B5D35 | B5D34 | B5D33 | B5D32 | B5D31 | B5D30 | xxxx xxxx | 62, 305 |
| B5D2(8) | B5D27 | B5D26 | B5D25 | B5D24 | B5D23 | B5D22 | B5D21 | B5D20 | xxxx xxxx | 62, 305 |
| B5D1(8) | B5D17 | B5D16 | B5D15 | B5D14 | B5D13 | B5D12 | B5D11 | B5D10 | xxxx xxxx | 62, 305 |
| B5D0(8) | B5D07 | B5D06 | B5D05 | B5D04 | B5D03 | B5D02 | B5D01 | B5D00 | xxxx xxxx | 62, 305 |
| B5DLC(8) Receive mode | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 62, 307 |
| B5DLC(8) Transmit mode | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 62, 307 |
| B5EIDL(8) | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 62, 305 |
| B5EIDH(8) | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 62, 304 |
| B5SIDL(8) Receive mode | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 62, 303 |
| B5SIDL(8) Transmit mode | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 62, 303 |
| B5SIDH(8) | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx x-xx | 62, 302 |
| B5CON(8) Receive mode | RXFUL | RXM1 | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 62, 301 |
| B5CON(8) Transmit mode | TXBIF | TXABT | TXLARB | TXERR | TXREQ | RTREN | TXPRI1 | TXPRI0 | 0000 0000 | 62, 301 |
| B4D7(8) | B4D77 | B4D76 | B4D75 | B4D74 | B4D73 | B4D72 | B4D71 | B4D70 | xxxx xxxx | 62, 305 |
| B4D6(8) | B4D67 | B4D66 | B4D65 | B4D64 | B4D63 | B4D62 | B4D61 | B4D60 | xxxx xxxx | 62, 305 |
| B4D5(8) | B4D57 | B4D56 | B4D55 | B4D54 | B4D53 | B4D52 | B4D51 | B4D50 | xxxx xxxx | 62, 305 |
| B4D4(8) | B4D47 | B4D46 | B4D45 | B4D44 | B4D43 | B4D42 | B4D41 | B4D40 | xxxx xxxx | 63, 305 |
| B4D3(8) | B4D37 | B4D36 | B4D35 | B4D34 | B4D33 | B4D32 | B4D31 | B4D30 | xxxx xxxx | 63, 305 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| B4D2(8) | B4D27 | B4D26 | B4D25 | B4D24 | B4D23 | B4D22 | B4D21 | B4D20 | xxxx xxxx | 63, 305 |
| B4D1(8) | B4D17 | B4D16 | B4D15 | B4D14 | B4D13 | B4D12 | B4D11 | B4D10 | xxxx xxxx | 63, 305 |
| B4D0(8) | B4D07 | B4D06 | B4D05 | B4D04 | B4D03 | B4D02 | B4D01 | B4D00 | xxxx xxxx | 62, 305 |
| B4DLC(8) Receive mode | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 63, 307 |
| B4DLC(8) Transmit mode | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 63, 307 |
| B4EIDL(8) | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 63, 305 |
| B4EIDH(8) | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 63, 304 |
| B4SIDL(8) Receive mode | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 63, 303 |
| B4SIDL(8) Transmit mode | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 63, 303 |
| B4SIDH(8) | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 63, 302 |
| B4CON(8) Receive mode | RXFUL | RXM1 | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 63, 301 |
| B4CON(8) Transmit mode | TXBIF | TXABT | TXLARB | TXERR | TXREQ | RTREN | TXPRI1 | TXPRI0 | 0000 0000 | 63, 301 |
| B3D7(8) | B3D77 | B3D76 | B3D75 | B3D74 | B3D73 | B3D72 | B3D71 | B3D70 | xxxx xxxx | 63, 305 |
| B3D6(8) | B3D67 | B3D66 | B3D65 | B3D64 | B3D63 | B3D62 | B3D61 | B3D60 | xxxx xxxx | 63, 305 |
| B3D5(8) | B3D57 | B3D56 | B3D55 | B3D54 | B3D53 | B3D52 | B3D51 | B3D50 | xxxx xxxx | 63, 305 |
| B3D4(8) | B3D47 | B3D46 | B3D45 | B3D44 | B3D43 | B3D42 | B3D41 | B3D40 | xxxx xxxx | 63, 305 |
| B3D3(8) | B3D37 | B3D36 | B3D35 | B3D34 | B3D33 | B3D32 | B3D31 | B3D30 | xxxx xxxx | 63, 305 |
| B3D2(8) | B3D27 | B3D26 | B3D25 | B3D24 | B3D23 | B3D22 | B3D21 | B3D20 | xxxx xxxx | 63, 305 |
| B3D1(8) | B3D17 | B3D16 | B3D15 | B3D14 | B3D13 | B3D12 | B3D11 | B3D10 | xxxx xxxx | 63, 305 |
| B3D0(8) | B3D07 | B3D06 | B3D05 | B3D04 | B3D03 | B3D02 | B3D01 | B3D00 | xxxx xxxx | 63, 305 |
| B3DLC(8) Receive mode | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 63, 307 |
| B3DLC(8) Transmit mode | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 63, 307 |
| B3EIDL(8) | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 63, 305 |
| B3EIDH(8) | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 63, 304 |
| B3SIDL(8) Receive mode | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 63, 303 |
| B3SIDL(8) Transmit mode | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 63, 303 |
| B3SIDH(8) | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 63, 302 |
| B3CON(8) Receive mode | RXFUL | RXM1 | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 63, 301 |
| B3CON(8) Transmit mode | TXBIF | TXABT | TXLARB | TXERR | TXREQ | RTREN | TXPRI1 | TXPRI0 | 0000 0000 | 63, 301 |
| B2D7(8) | B2D77 | B2D76 | B2D75 | B2D74 | B2D73 | B2D72 | B2D71 | B2D70 | xxxx xxxx | 63, 305 |
| B2D6(8) | B2D67 | B2D66 | B2D65 | B2D64 | B2D63 | B2D62 | B2D61 | B2D60 | xxxx xxxx | 63, 305 |
| B2D5(8) | B2D57 | B2D56 | B2D55 | B2D54 | B2D53 | B2D52 | B2D51 | B2D50 | xxxx xxxx | 63, 305 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.


© 2009 Microchip Technology Inc.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| B2D4(8) | B2D47 | B2D46 | B2D45 | B2D44 | B2D43 | B2D42 | B2D41 | B2D40 | xxxx xxxx | 63, 305 |
| B2D3(8) | B2D37 | B2D36 | B2D35 | B2D34 | B2D33 | B2D32 | B2D31 | B2D30 | xxxx xxxx | 63, 305 |
| B2D2(8) | B2D27 | B2D26 | B2D25 | B2D24 | B2D23 | B2D22 | B2D21 | B2D20 | xxxx xxxx | 63, 305 |
| B2D1(8) | B2D17 | B2D16 | B2D15 | B2D14 | B2D13 | B2D12 | B2D11 | B2D10 | xxxx xxxx | 64, 305 |
| B2D0(8) | B2D07 | B2D06 | B2D05 | B2D04 | B2D03 | B2D02 | B2D01 | B2D00 | xxxx xxxx | 64, 305 |
| B2DLC(8) Receive mode | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 64, 307 |
| B2DLC(8) Transmit mode | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 64, 307 |
| B2EIDL(8) | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 64, 305 |
| B2EIDH(8) | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 64, 304 |
| B2SIDL(8) Receive mode | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 64, 303 |
| B2SIDL(8) Transmit mode | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 64, 303 |
| B2SIDH(8) | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 64, 302 |
| B2CON(8) Receive mode | RXFUL | RXM1 | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 64, 301 |
| B2CON(8) Transmit mode | TXBIF | RXM1 | TXLARB | TXERR | TXREQ | RTREN | TXPRI1 | TXPRI0 | 0000 0000 | 64, 301 |
| B1D7(8) | B1D77 | B1D76 | B1D75 | B1D74 | B1D73 | B1D72 | B1D71 | B1D70 | xxxx xxxx | 64, 305 |
| B1D6(8) | B1D67 | B1D66 | B1D65 | B1D64 | B1D63 | B1D62 | B1D61 | B1D60 | xxxx xxxx | 64, 305 |
| B1D5(8) | B1D57 | B1D56 | B1D55 | B1D54 | B1D53 | B1D52 | B1D51 | B1D50 | xxxx xxxx | 64, 305 |
| B1D4(8) | B1D47 | B1D46 | B1D45 | B1D44 | B1D43 | B1D42 | B1D41 | B1D40 | xxxx xxxx | 64, 305 |
| B1D3(8) | B1D37 | B1D36 | B1D35 | B1D34 | B1D33 | B1D32 | B1D31 | B1D30 | xxxx xxxx | 64, 305 |
| B1D2(8) | B1D27 | B1D26 | B1D25 | B1D24 | B1D23 | B1D22 | B1D21 | B1D20 | xxxx xxxx | 64, 305 |
| B1D1(8) | B1D17 | B1D16 | B1D15 | B1D14 | B1D13 | B1D12 | B1D11 | B1D10 | xxxx xxxx | 64, 305 |
| B1D0(8) | B1D07 | B1D06 | B1D05 | B1D04 | B1D03 | B1D02 | B1D01 | B1D00 | xxxx xxxx | 64, 305 |
| B1DLC(8) Receive mode | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 64, 307 |
| B1DLC(8) Transmit mode | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 64, 307 |
| B1EIDL(8) | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 64, 305 |
| B1EIDH(8) | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 64, 304 |
| B1SIDL(8) Receive mode | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 64, 303 |
| B1SIDL(8) Transmit mode | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 64, 303 |
| B1SIDH(8) | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 64, 302 |
| B1CON(8) Receive mode | RXFUL | RXM1 | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 64, 301 |
| B1CON(8) Transmit mode | TXBIF | TXABT | TXLARB | TXERR | TXREQ | RTREN | TXPRI1 | TXPRI0 | 0000 0000 | 64, 301 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| B0D7(8) | B0D77 | B0D76 | B0D75 | B0D74 | B0D73 | B0D72 | B0D71 | B0D70 | xxxx xxxx | 64, 305 |
| B0D6(8) | B0D67 | B0D66 | B0D65 | B0D64 | B0D63 | B0D62 | B0D61 | B0D60 | xxxx xxxx | 64, 305 |
| B0D5(8) | B0D57 | B0D56 | B0D55 | B0D54 | B0D53 | B0D52 | B0D51 | B0D50 | xxxx xxxx | 64, 305 |
| B0D4(8) | B0D47 | B0D46 | B0D45 | B0D44 | B0D43 | B0D42 | B0D41 | B0D40 | xxxx xxxx | 64, 305 |
| B0D3(8) | B0D37 | B0D36 | B0D35 | B0D34 | B0D33 | B0D32 | B0D31 | B0D30 | xxxx xxxx | 64, 305 |
| B0D2(8) | B0D27 | B0D26 | B0D25 | B0D24 | B0D23 | B0D22 | B0D21 | B0D20 | xxxx xxxx | 64, 305 |
| B0D1(8) | B0D17 | B0D16 | B0D15 | B0D14 | B0D13 | B0D12 | B0D11 | B0D10 | xxxx xxxx | 64, 305 |
| B0D0(8) | B0D07 | B0D06 | B0D05 | B0D04 | B0D03 | B0D02 | B0D01 | B0D00 | xxxx xxxx | 64, 305 |
| B0DLC(8) Receive mode | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 64, 307 |
| B0DLC(8) Transmit mode | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 64, 307 |
| B0EIDL(8) | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 65, 305 |
| B0EIDH(8) | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 65, 304 |
| B0SIDL(8) Receive mode | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 65, 303 |
| B0SIDL(8) Transmit mode | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx- x-xx | 65, 303 |
| B0SIDH(8) | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 65, 302 |
| B0CON(8) Receive mode | RXFUL | RXM1 | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 | 0000 0000 | 64, 301 |
| B0CON(8) Transmit mode | TXBIF | TXABT | TXLARB | TXERR | TXREQ | RTREN | TXPRI1 | TXPRI0 | 0000 0000 | 64, 301 |
| TXBIE | — | — | — | TXB2IE | TXB1IE | TXB0IE | — | — | ---0 00-- | 65, 324 |
| BIE0 | B5IE | B4IE | B3IE | B2IE | B1IE | B0IE | RXB1IE | RXB0IE | 0000 0000 | 65, 324 |
| BSEL0 | B5TXEN | B4TXEN | B3TXEN | B2TXEN | B1TXEN | B0TXEN | — | — | 0000 00-- | 65, 307 |
| MSEL3 | FIL15_1 | FIL15_0 | FIL14_1 | FIL14_0 | FIL13_1 | FIL13_0 | FIL12_1 | FIL12_0 | 0000 0000 | 65, 316 |
| MSEL2 | FIL11_1 | FIL11_0 | FIL10_1 | FIL10_0 | FIL9_1 | FIL9_0 | FIL8_1 | FIL8_0 | 0000 0000 | 65, 315 |
| MSEL1 | FIL7_1 | FIL7_0 | FIL6_1 | FIL6_0 | FIL5_1 | FIL5_0 | FIL4_1 | FIL4_0 | 0000 0101 | 65, 314 |
| MSEL0 | FIL3_1 | FIL3_0 | FIL2_1 | FIL2_0 | FIL1_1 | FIL1_0 | FIL0_1 | FIL0_0 | 0101 0000 | 65, 313 |
| RXFBCON7 | F15BP_3 | F15BP_2 | F15BP_1 | F15BP_0 | F14BP_3 | F14BP_2 | F14BP_1 | F14BP_0 | 0000 0000 | 65, 312 |
| RXFBCON6 | F13BP_3 | F13BP_2 | F13BP_1 | F13BP_0 | F12BP_3 | F12BP_2 | F12BP_1 | F12BP_0 | 0000 0000 | 65, 312 |
| RXFBCON5 | F11BP_3 | F11BP_2 | F11BP_1 | F11BP_0 | F10BP_3 | F10BP_2 | F10BP_1 | F10BP_0 | 0000 0000 | 65, 312 |
| RXFBCON4 | F9BP_3 | F9BP_2 | F9BP_1 | F9BP_0 | F8BP_3 | F8BP_2 | F8BP_1 | F8BP_0 | 0000 0000 | 65, 312 |
| RXFBCON3 | F7BP_3 | F7BP_2 | F7BP_1 | F7BP_0 | F6BP_3 | F6BP_2 | F6BP_1 | F6BP_0 | 0000 0000 | 65, 312 |
| RXFBCON2 | F5BP_3 | F5BP_2 | F5BP_1 | F5BP_0 | F4BP_3 | F4BP_2 | F4BP_1 | F4BP_0 | 0001 0001 | 65, 312 |
| RXFBCON1 | F3BP_3 | F3BP_2 | F3BP_1 | F3BP_0 | F2BP_3 | F2BP_2 | F2BP_1 | F2BP_0 | 0001 0001 | 65, 312 |
| RXFBCON0 | F1BP_3 | F1BP_2 | F1BP_1 | F1BP_0 | F0BP_3 | F0BP_2 | F0BP_1 | F0BP_0 | 0000 0000 | 65, 312 |
| SDFLC | — | — | — | FLC4 | FLC3 | FLC2 | FLC1 | FLC0 | ---0 0000 | 65, 312 |
| RXFCON1 | RXF15EN | RXF14EN | RXF13EN | RXF12EN | RXF11EN | RXF10EN | RXF9EN | RXF8EN | 0000 0000 | 65, 311 |
| RXFCON0 | RXF7EN | RXF6EN | RXF5EN | RXF4EN | RXF3EN | RXF2EN | RXF1EN | RXF0EN | 0000 0000 | 65, 311 |
| RXF15EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 65, 309 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.


© 2009 Microchip Technology Inc.

**TABLE 6-2: REGISTER FILE SUMMARY (PIC18F2480/2580/4480/4580) (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: |
|---|---|---|---|---|---|---|---|---|---|---|
| RXF15EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 65, 309 |
| RXF15SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 65, 308 |
| RXF15SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 65, 309 |
| RXF14EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 65, 309 |
| RXF14EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 65, 309 |
| RXF14SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 65, 308 |
| RXF14SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 65, 309 |
| RXF13EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 66, 309 |
| RXF13EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 66, 309 |
| RXF13SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 66, 308 |
| RXF13SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 66, 309 |
| RXF12EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 66, 309 |
| RXF12EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 66, 309 |
| RXF12SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 66, 308 |
| RXF12SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 66, 309 |
| RXF11EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 66, 309 |
| RXF11EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 66, 309 |
| RXF11SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 66, 308 |
| RXF11SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 66, 309 |
| RXF10EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 66, 309 |
| RXF10EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 66, 309 |
| RXF10SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 66, 308 |
| RXF10SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 66, 309 |
| RXF9EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 66, 309 |
| RXF9EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 66, 309 |
| RXF9SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 66, 308 |
| RXF9SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 66, 309 |
| RXF8EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 66, 309 |
| RXF8EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 66, 309 |
| RXF8SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 66, 308 |
| RXF8SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 66, 309 |
| RXF7EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 66, 309 |
| RXF7EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 66, 309 |
| RXF7SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 66, 308 |
| RXF7SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 66, 309 |
| RXF6EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 66, 309 |
| RXF6EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 66, 309 |
| RXF6SIDL | SID2 | SID1 | SID0 | — | EXIDEN | — | EID17 | EID16 | xxx- x-xx | 66, 308 |
| RXF6SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 66, 309 |

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

**Note 1:** Bit 21 of the PC is only available in Test mode and Serial Programming modes.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**3:** These registers and/or bits are not implemented on PIC18F2X80 devices and are read as '0'. Reset values are shown for PIC18F4X80 devices; individual unimplemented bits should be interpreted as '—'.

**4:** The PLLEN bit is only available in specific oscillator configuration; otherwise, it is disabled and reads as '0'. See **Section 3.6.4 "PLL in INTOSC Modes"**.

**5:** The RE3 bit is only available when Master Clear Reset is disabled (CONFIG3H<7> = 0); otherwise, RE3 reads as '0'. This bit is read-only.

**6:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**7:** CAN bits have multiple functions depending on the selected mode of the CAN module.

**8:** This register reads all '0's until the ECAN™ technology is set up in Mode 1 or Mode 2.

**9:** These registers are available on PIC18F4X80 devices only.

### 6.3.5 STATUS REGISTER

The STATUS register, shown in Register 6-2, 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 26-2 and Table 26-3.

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

**REGISTER 6-2: STATUS REGISTER**

| U-0 | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | — | N | OV | Z | DC(1) | C(2) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| -n = Value at POR | ‘1’ = Bit is set | ‘0’ = Bit is cleared | x = Bit is unknown |

bit 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/borrow bit(1)
For `ADDWF`, `ADDLW`, `SUBLW` and `SUBWF` instructions:
`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(2)
For `ADDWF`, `ADDLW`, `SUBLW` and `SUBWF` instructions:
`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 bit 4 or bit 3 of the source register.

**2:** 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 or low-order bit of the source register.


© 2009 Microchip Technology Inc.

## 6.4 Data Addressing Modes

**Note:** The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 6.6 "Data Memory and the Extended Instruction Set"** for more information.

While the program memory can be addressed in only one way – through the program counter – information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in **Section 6.6.1 "Indexed Addressing with Literal Offset"**.

### 6.4.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include SLEEP, RESET and DAW.

Other instructions work in a similar way but require an additional explicit argument in the opcode. This is known as Literal Addressing mode because they require some literal value as an argument. Examples include ADDLW and MOVLW which, respectively, add or move a literal value to the W register. Other examples include CALL and GOTO, which include a 20-bit program memory address.

### 6.4.2 DIRECT ADDRESSING

Direct Addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 6.3.3 "General Purpose Register File"**) or a location in the Access Bank (**Section 6.3.2 "Access Bank"**) as the data source for the instruction.

The Access RAM bit 'a' determines how the address is interpreted. When 'a' is '1', the contents of the BSR (**Section 6.3.1 "Bank Select Register (BSR)"**) are used with the address to determine the complete 12-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as MOVFF, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation's results is determined by the destination bit, 'd'. When 'd' is '1', the results are stored back in the source register, overwriting its original contents. When 'd' is '0', the results are stored in the W register. Instructions without the 'd' argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

### 6.4.3 INDIRECT ADDRESSING

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations to be read or written to. Since the FSRs are themselves located in RAM as Special File Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for Indirect Addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in Example 6-5.

**EXAMPLE 6-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING**

```
        LFSR   FSR0, 100h  ;
NEXT    CLRF   POSTINC0    ; Clear INDF
                           ; register then
                           ; inc pointer
        BTFSS  FSR0H,1     ; All done with
                           ; Bank1?
        BRA    NEXT        ; NO, clear next
CONTINUE                   ; YES, continue
```

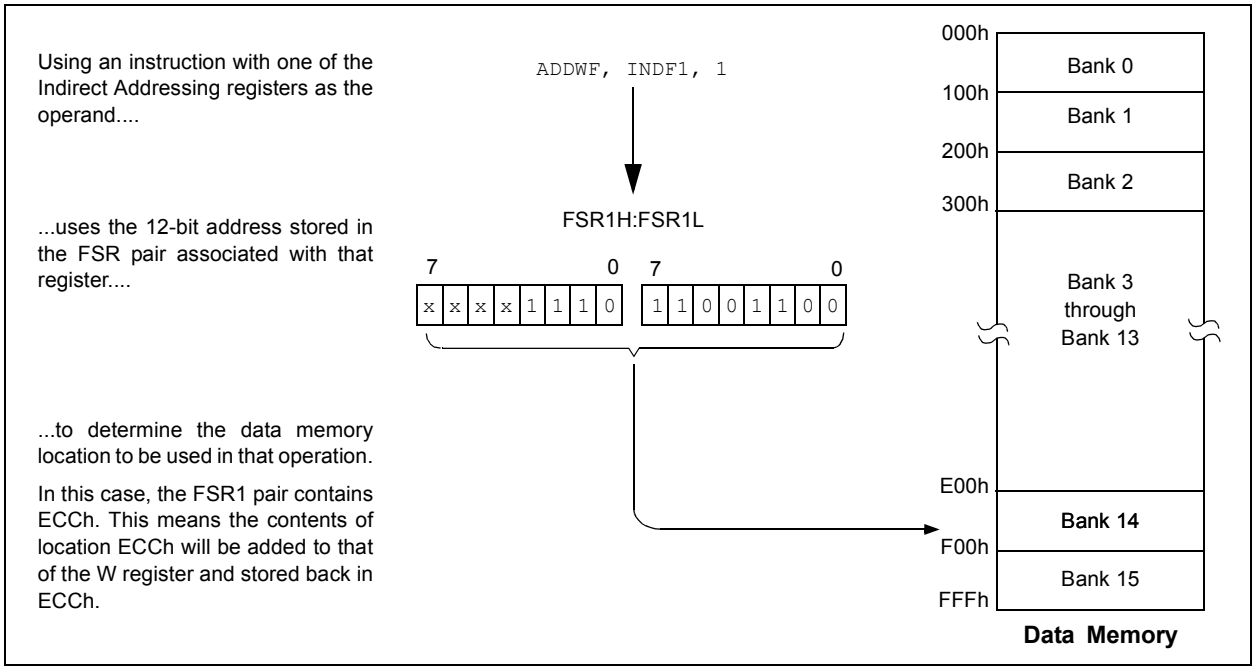
#### 6.4.3.1 FSR Registers and the INDF Operand

At the core of Indirect Addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers, FSRnH and FSRnL. The four upper bits of the FSRnH register are not used, so each FSR pair holds a 12-bit value. This represents a value that can address the entire range of the data memory in a linear fashion. The FSR register pairs, then, serve as pointers to data memory locations.

Indirect Addressing is accomplished with a set of Indirect File Operands, INDF0 through INDF2. These can be thought of as "virtual" registers: they are mapped in the SFR space, but are not physically implemented. Reading or writing to a particular INDF register actually accesses its corresponding FSR register pair. A read from INDF1, for example, reads the data at the address indicated by FSR1H:FSR1L. Instructions that use the INDF registers as operands actually use the contents of their corresponding FSR as a pointer to the instruction's target. The INDF operand is just a convenient way of using the pointer.

Because Indirect Addressing uses a full 12-bit address, data RAM banking is not necessary. Thus, the current contents of the BSR and the Access RAM bit have no effect on determining the target address.

**FIGURE 6-8: INDIRECT ADDRESSING**





© 2009 Microchip Technology Inc.

#### 6.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers that cannot be indirectly read or written to. Accessing these registers actually accesses the associated FSR register pair, but also performs a specific action on its stored value. They are:

- POSTDEC: accesses the FSR value, then automatically decrements it by 1 afterwards
- POSTINC: accesses the FSR value, then automatically increments it by 1 afterwards
- PREINC: increments the FSR value by 1, then uses it in the operation
- PLUSW: adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the new value in the operation.

In this context, accessing an INDF register, uses the value in the FSR registers without changing them. Similarly, accessing a PLUSW register gives the FSR value offset by that in the W register; neither value is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR registers.

Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, roll-overs of the FSRnL register from FFh to 00h carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (e.g., Z, N, OV, etc.).

The PLUSW register can be used to implement a form of Indexed Addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

#### 6.4.3.3 Operations by FSRs on FSRs

Indirect Addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specific case, assume that FSR0H:FSR0L contains FE7h, the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a `NOP`.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to INDF2 or POSTDEC2 will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, particularly if their code uses Indirect Addressing.

Similarly, operations by Indirect Addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

## 6.5 Program Memory and the Extended Instruction Set

The operation of program memory is unaffected by the use of the extended instruction set.

Enabling the extended instruction set adds eight additional two-word commands to the existing PIC18 instruction set: ADDFSR, ADDULNK, CALLW, MOVSF, MOVSS, PUSHL, SUBFSR and SUBULNK. These instructions are executed as described in **Section 6.2.4 "Two-Word Instructions"**.

## 6.6 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different. This is due to the introduction of a new addressing mode for the data memory space. This mode also alters the behavior of Indirect Addressing using FSR2 and its associated operands.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect Addressing with FSR0 and FSR1 also remains unchanged.

### 6.6.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair and its associated file operands. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented – instructions – can invoke a form of Indexed Addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced ('a' = 0); and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in Direct Addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer, specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

### 6.6.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use Direct Addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they use the Access Bank (Access RAM bit is '1'), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled in shown in Figure 6-9.

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 26.2.1 "Extended Instruction Syntax"**.


© 2009 Microchip Technology Inc.

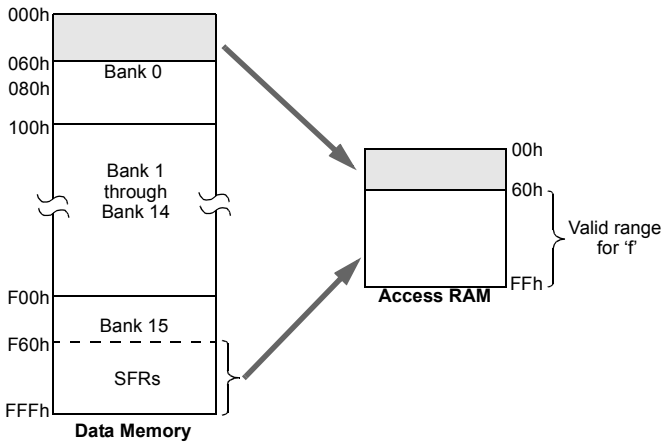
**FIGURE 6-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

**EXAMPLE INSTRUCTION:** `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

**When a = 0 and f ≥ 60h:**

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as the SFRs, or locations F60h to 0FFh (Bank 15) of data memory.

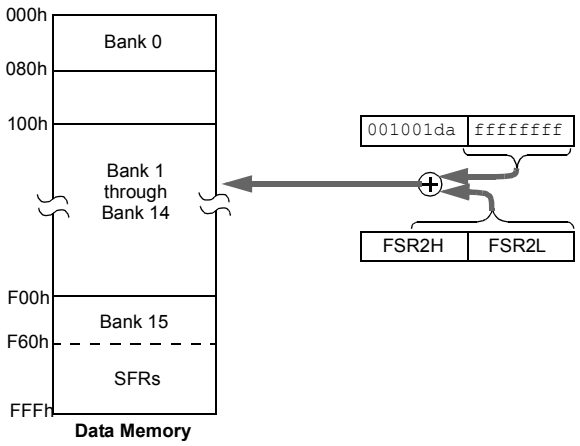
Locations below 60h are not available in this addressing mode.




**When a = 0 and f ≤ 5Fh:**

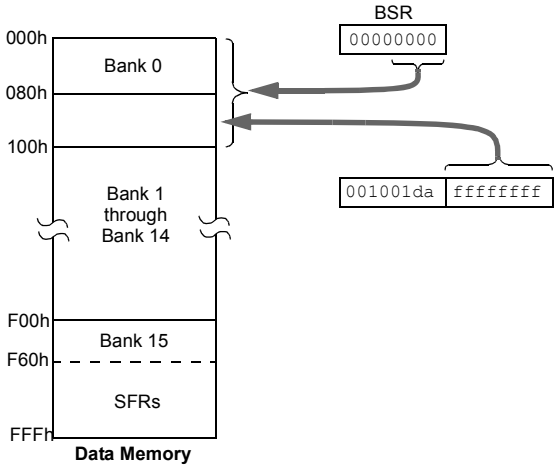
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now:
`ADDWF [k], d`
where 'k' is the same as 'f'.




**When a = 1 (all values of f):**

The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 16 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



### 6.6.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

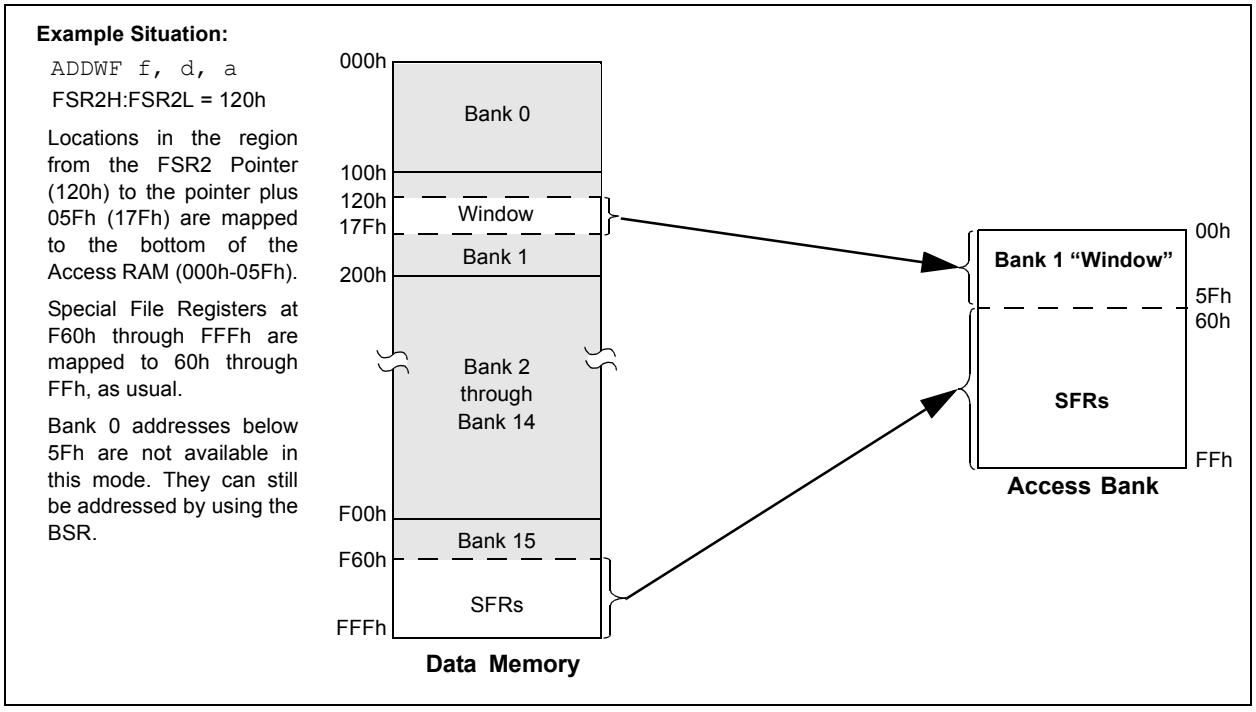
The use of Indexed Literal Offset Addressing mode effectively changes how the lower half of Access RAM (00h to 7Fh) is mapped. Rather than containing just the contents of the bottom half of Bank 0, this mode maps the contents from Bank 0 and a user-defined “window” that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see **Section 6.3.2 “Access Bank”**). An example of Access Bank remapping in this addressing mode is shown in Figure 6-10.

Remapping of the Access Bank applies *only* to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit is ‘1’) will continue to use Direct Addressing as before. Any indirect or indexed operation that explicitly uses any of the indirect file operands (including FSR2) will continue to operate as standard Indirect Addressing. Any instruction that uses the Access Bank, but includes a register address of greater than 05Fh, will use Direct Addressing and the normal Access Bank map.

### 6.6.4 BSR IN INDEXED LITERAL OFFSET MODE

Although the Access Bank is remapped when the extended instruction set is enabled, the operation of the BSR remains unchanged. Direct Addressing using the BSR to select the data memory bank operates in the same manner as previously described.

**FIGURE 6-10: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING**





© 2009 Microchip Technology Inc.

## 7.0 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.

Writing or erasing program memory will cease instruction fetches 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.

### 7.1 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)

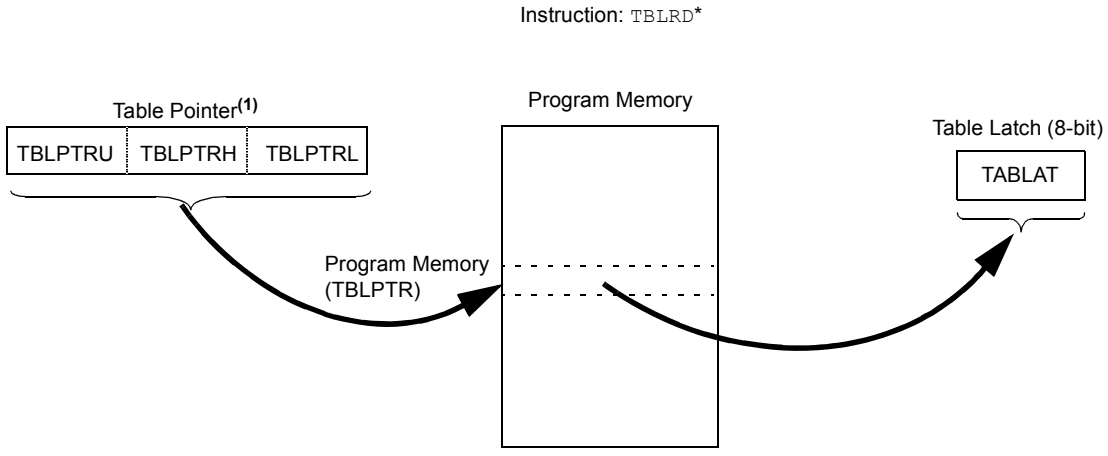
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 7-1 shows the operation of a table read with program memory and data RAM.

Table write operations store data from 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 7.5 "Writing to Flash Program Memory"**. Figure 7-2 shows the operation of a table write with program memory and data RAM.

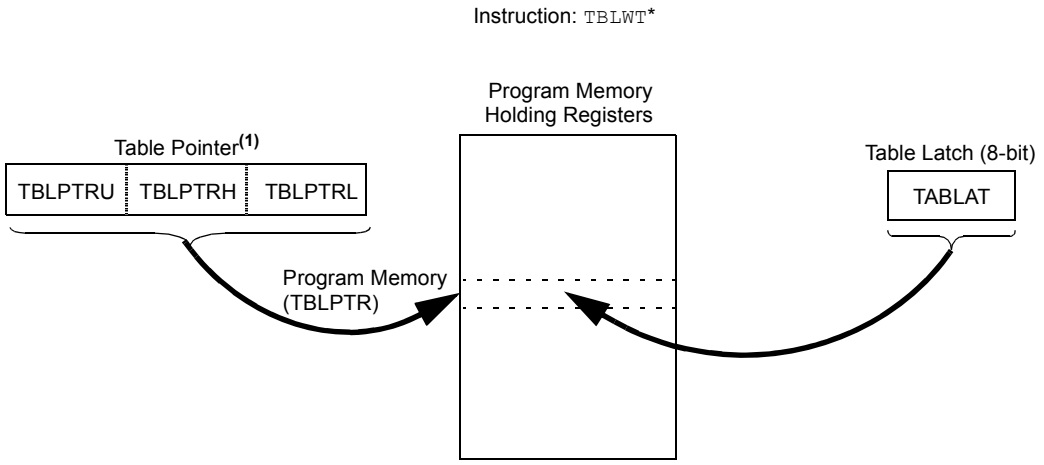
Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions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

**FIGURE 7-1: TABLE READ OPERATION**




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

**FIGURE 7-2: TABLE WRITE OPERATION**




**Note 1:** Table Pointer actually points to one of 32 holding registers, the address of which is determined by TBLPTRL<4:0>. The process for physically writing data to the program memory array is discussed in **Section 7.5 "Writing to Flash Program Memory"**.

## 7.2 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

### 7.2.1 EECON1 AND EECON2 REGISTERS

The EECON1 register (Register 7-1) is the control register for memory accesses. The EECON2 register is not a physical register; it is used exclusively in the memory write and erase sequences. Reading EECON2 will read all '0's.

The EEPGD control bit determines if the access will be a program or data EEPROM memory access. When clear, any subsequent operations will operate on the data EEPROM memory. When set, any subsequent operations will operate on the program memory.

The CFGS control bit determines if the access will be to the Configuration/Calibration registers or to program memory/data EEPROM memory. When set, subsequent operations will operate on Configuration registers regardless of EEPGD (see **Section 25.0 "Special Features of the CPU"**). When clear, memory selection access is determined by EEPGD.

The FREE bit, when set, will allow a program memory erase operation. When FREE is set, the erase operation is initiated on the next WR command. When FREE is clear, only writes are enabled.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set in hardware when the WREN bit is set and cleared when the internal programming timer expires and the write operation is complete.

| **Note:** | During normal operation, the WRERR is read as '1'. This can indicate that a write operation was prematurely terminated by a Reset, or a write operation was attempted improperly. |
|---|---|

The WR control bit initiates write operations. The bit cannot be cleared, only set, in software; it is cleared in hardware at the completion of the write operation.

| **Note:** | The EEIF Interrupt Flag bit (PIR2<4>) is set when the write is complete. It must be cleared in software. |
|---|---|


© 2009 Microchip Technology Inc.

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

| 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:** | S = Settable bit | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
- 1 = Access Configuration registers
- 0 = Access 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)
- 0 = Perform write only

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit(1)
- 1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation or an improper write attempt)
- 0 = The write operation completed

bit 2 **WREN:** Flash Program/Data EEPROM Write Enable bit
- 1 = Allows write cycles to Flash program/data EEPROM
- 0 = Inhibits write cycles to Flash program/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 to the EEPROM is complete

bit 0 **RD:** Read Control bit
- 1 = Initiates an EEPROM 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 or CFGS = 1.)
- 0 = Does not initiate an EEPROM read

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

### 7.2.2 TABLAT – TABLE LATCH REGISTER

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

### 7.2.3 TBLPTR – TABLE POINTER REGISTER

The Table Pointer (TBLPTR) register 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. The 22nd bit allows access to the Device ID, the user ID and the Configuration bits.

The Table Pointer, TBLPTR, 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 7-1. These operations on the TBLPTR only affect the low-order 21 bits.

### 7.2.4 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 TBLPTR determine which byte is read from program memory into TABLAT.

When a `TBLWT` is executed, the five LSbs of the Table Pointer register (TBLPTR<4:0>) determine which of the 32 program memory holding registers is written to. When the timed write to program memory begins (via the WR bit), the 16 MSbs of the TBLPTR (TBLPTR<21:6>) determine which program memory block of 32 bytes is written to. For more detail, see **Section 7.5 "Writing to Flash Program Memory"**.

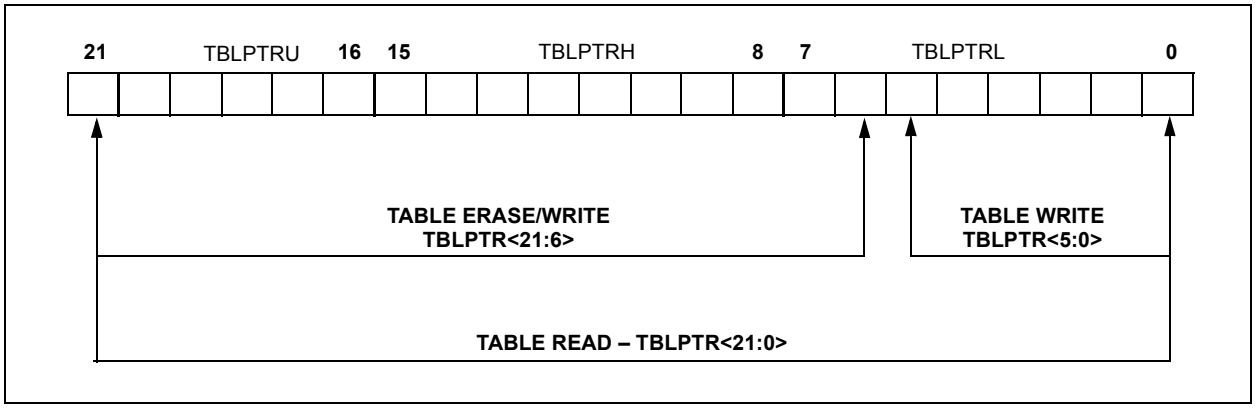
When an erase of program memory is executed, the 16 MSbs of the Table Pointer register (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

Figure 7-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

**TABLE 7-1: 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 7-3: TABLE POINTER BOUNDARIES BASED ON OPERATION**





© 2009 Microchip Technology Inc.

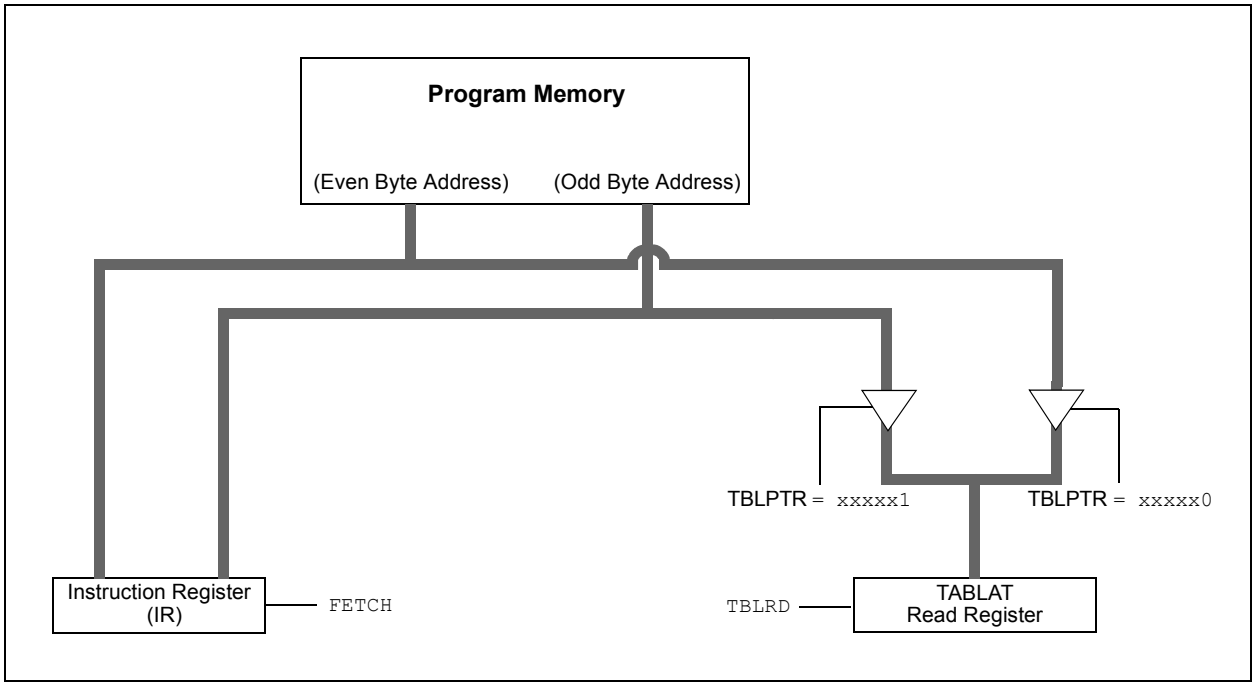
## 7.3 Reading the Flash Program Memory

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 TBLRD 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 7-4 shows the interface between the internal program memory and the TABLAT.

**FIGURE 7-4: READS FROM FLASH PROGRAM MEMORY**




**EXAMPLE 7-1: 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
            MOVF      TABLAT, W                 ; get data
            MOVWF     WORD_EVEN
            TBLRD*+                             ; read into TABLAT and increment
            MOVF      TABLAT, W                 ; get data
            MOVF      WORD_ODD
```

## 7.4 Erasing Flash Program Memory

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

When initiating an erase sequence from the microcontroller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:5> point to the block being erased. TBLPTR<4:0> are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

For protection, the write initiate sequence for 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.

### 7.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

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

1. Load Table Pointer register 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 0AAh 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. Re-enable interrupts.

**EXAMPLE 7-2: 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
                BCF     EECON1, CFGS      ; access Flash program memory
                BSF     EECON1, WREN      ; enable write to memory
                BSF     EECON1, FREE      ; enable Row Erase operation
                BCF     INTCON, GIE       ; disable interrupts
Required        MOVLW   55h
Sequence        MOVWF   EECON2            ; write 55h
                MOVLW   0AAh
                MOVWF   EECON2            ; write 0AAh
                BSF     EECON1, WR        ; start erase (CPU stall)
                BSF     INTCON, GIE       ; re-enable interrupts
```


© 2009 Microchip Technology Inc.

## 7.5 Writing to Flash Program Memory

The minimum programming block is 16 words or 32 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 32 holding registers used by the table writes for programming.

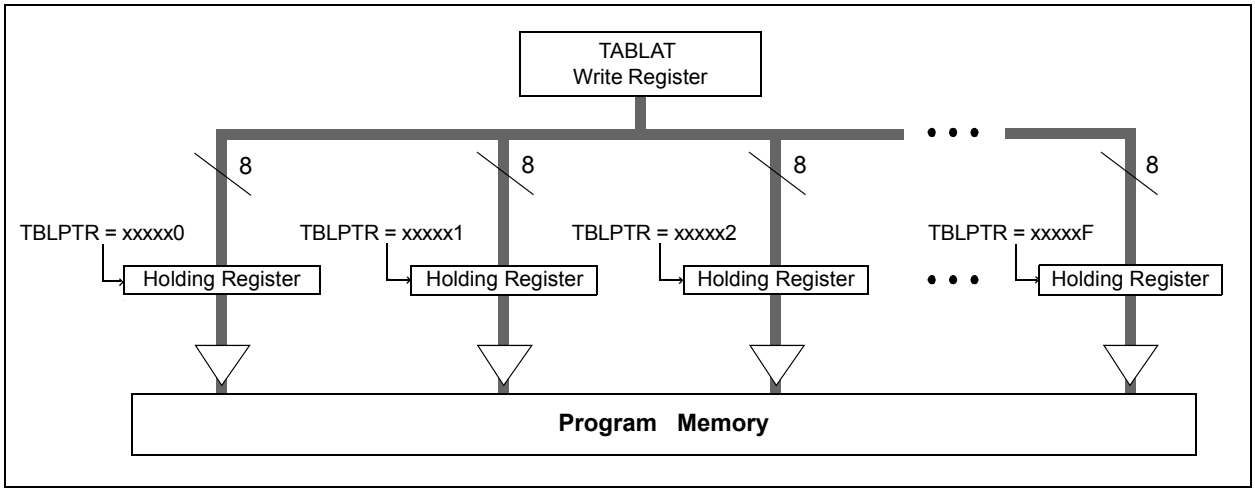
Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction may need to be executed 32 times for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. At the end of updating the 32 holding registers, the EECON1 register must be written to in order 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.

The EEPROM on-chip timer controls the write time. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device.

| **Note:** | The default value of the holding registers on device Resets and after write operations is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all 32 holding registers before executing a write operation. |
|---|---|

**FIGURE 7-5: TABLE WRITES TO FLASH PROGRAM MEMORY**




### 7.5.1 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 register with address being erased.
4. Execute the row erase procedure.
5. Load Table Pointer register with address of first byte being written.
6. Write the 32 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 to enable byte writes.
8. Disable interrupts.
9. Write 55h to EECON2.
10. Write 0AAh 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). After writing to the holding registers, it will be set to 0xFF.
13. Repeat the question three more times.
14. Re-enable interrupts.
15. Verify the memory (table read).

This procedure will require about 6 ms to update one row of 64 bytes of memory. An example of the required code is given in Example 7-3.

| **Note:** | Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the 32 bytes in the holding register. |
|---|---|

**EXAMPLE 7-3: 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
                    MOVWF   TBLPTRL
READ_BLOCK
                    TBLRD*+                         ; read into TABLAT, and inc
                    MOVF    TABLAT, W               ; get data
                    MOVWF   POSTINC0                ; store data
                    DECFSZ  COUNTER                 ; done?
                    BRA     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
                    MOVWF   POSTINC0
                    MOVLW   NEW_DATA_HIGH
                    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
                    MOVWF   TBLPTRL
                    BSF     EECON1, EEPGD           ; point to Flash program memory
                    BCF     EECON1, CFGS            ; access 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            MOVWF   EECON2                  ; write 55h
Sequence            MOVLW   0AAh
                    MOVWF   EECON2                  ; write 0AAh
                    BSF     EECON1, WR              ; start erase (CPU stall)
                    BSF     INTCON, GIE             ; re-enable interrupts
                    TBLRD*-                         ; dummy read decrement
                    MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
                    MOVWF   FSR0H
                    MOVLW   BUFFER_ADDR_LOW
                    MOVWF   FSR0L
                    MOVLW   D'4'
                    MOVWF   COUNTER1
WRITE_BUFFER_BACK
                    MOVLW   D'64                    ; number of bytes in holding register
                    MOVWF   COUNTER
WRITE_BYTE_TO_HREGS
                    MOVF    POSTINC0, W             ; get low byte of buffer data
                    MOVWF   TABLAT                  ; present data to table latch
                    TBLWT+*                         ; write data, perform a short write
                                                    ; to internal TBLWT holding register.
                    DECFSZ  COUNTER                 ; loop until buffers are full
                    BRA     WRITE_BYTE_TO_HREGS
```


© 2009 Microchip Technology Inc.

**EXAMPLE 7-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)**

```
PROGRAM_MEMORY
                BSF     EECON1,EEPGD          ; point to Flash program memory
                BCF     EECON1, CFGS          ; access Flash program memory
                BSF     EECON1, WREN          ; enable write to memory
                BCF     INTCON, GIE           ; disable interrupts
                MOVLW   55h
   Required     MOVWF   EECON2                ; write 55h
   Sequence     MOVLW   0AAh
                MOVWF   EECON2                ; write 0AAh
                BSF     EECON1, WR            ; start program (CPU stall)
                DECFSZ  COUNTER1
                BRA     WRITE_BUFFER_BACK
                BSF     INTCON, GIE           ; re-enable interrupts
                BCF     EECON1, WREN          ; disable write to memory
```

### 7.5.2 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.

### 7.5.3 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. If the write operation is interrupted by a $\overline{\text{MCLR}}$ Reset or a WDT time-out Reset during normal operation, the user can check the WRERR bit and rewrite the location(s) as needed.

### 7.5.4 PROTECTION AGAINST SPURIOUS WRITES

To protect against spurious writes to Flash program memory, the write initiate sequence must also be followed. See **Section 25.0 "Special Features of the CPU"** for more detail.

## 7.6 Flash Program Operation During Code Protection

See **Section 25.5 "Program Verification and Code Protection"** for details on code protection of Flash program memory.

**TABLE 7-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| TBLPTRU | — | — | bit21(3) | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | 55 |
| TBPLTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 55 |
| TBLPTRL | Program Memory Table Pointer High Byte (TBLPTR<7:0>) | | | | | | | | 55 |
| TABLAT | Program Memory Table Latch | | | | | | | | 55 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INTE | RBIE | TMR0IF | INTF | RBIF | 55 |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | 57 |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | 57 |
| IPR2 | OSCFIP | CMIP(2) | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP(1) | 57 |
| PIR2 | OSCFIF | CMIF(2) | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF(1) | 58 |
| PIE2 | OSCFIE | CMIE(2) | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE(1) | 58 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during Flash/EEPROM access.

**Note 1:** These bits are available in PIC18F4X80 devices only.

**2:** These bits are available in PIC18F4X80 devices and reserved in PIC18F2X80 devices.

**3:** This bit is available only in Test mode and Serial Programming mode.

**NOTES:**


© 2009 Microchip Technology Inc.

## 8.0 DATA EEPROM MEMORY

The data EEPROM is a nonvolatile memory array, separate from the data RAM and program memory, that is used for long-term storage of program data. It is not directly mapped in either the register file or program memory space, but is indirectly addressed through the Special Function Registers (SFRs). The EEPROM is readable and writable during normal operation over the entire VDD range.

Four SFRs are used to read and write to the data EEPROM, as well as the program memory. They are:

- EECON1
- EECON2
- EEDATA
- EEADR

The data EEPROM allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and the EEADR register holds the address of the EEPROM location being accessed.

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; it will vary with voltage and temperature, as well as from chip to chip. Please refer to parameter D122 (Table 28-1 in **Section 28.0 "Electrical Characteristics"**) for exact limits.

### 8.1 EEADR Register

The EEADR register is used to address the data EEPROM for read and write operations. The 8-bit range of the register can address a memory range of 256 bytes (00h to FFh).

### 8.2 EECON1 and EECON2 Registers

Access to the data EEPROM is controlled by two registers: EECON1 and EECON2. These are the same registers which control access to the program memory and are used in a similar manner for the data EEPROM.

The EECON1 register (Register 8-1) is the control register for data and program memory access. 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, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set in hardware when the WREN bit is set and cleared when the internal programming timer expires and the write operation is complete.

| | |
|---|---|
| **Note:** | During normal operation, the WRERR is read as '1'. This can indicate that a write operation was prematurely terminated by a Reset, or a write operation was attempted improperly. |

The WR control bit initiates write operations. The bit cannot be cleared, only set, in software; it is cleared in hardware at the completion of the write operation.

| | |
|---|---|
| **Note:** | The EEIF interrupt flag bit (PIR2<4>) is set when the write is complete. It must be cleared in software. |

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 7.1 "Table Reads and Table Writes"** regarding table reads.

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

**REGISTER 8-1: EECON1: DATA EEPROM CONTROL REGISTER 1**

| 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:** | S = Settable bit | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
1 = Access Configuration registers
0 = Access 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)
0 = Perform write only

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit(1)
1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation or an improper write attempt)
0 = The write operation completed

bit 2 **WREN:** Flash Program/Data EEPROM Write Enable bit
1 = Allows write cycles to Flash program/data EEPROM
0 = Inhibits write cycles to Flash program/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 to the EEPROM is complete

bit 0 **RD:** Read Control bit
1 = Initiates an EEPROM 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 or CFGS = 1.)
0 = Does not initiate an EEPROM read

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


© 2009 Microchip Technology Inc.

## 8.3 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 on 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).

The basic process is shown in Example 8-1.

## 8.4 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 8-2 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 0AAh 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.

## 8.5 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.

**EXAMPLE 8-1: DATA EEPROM READ**

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

**EXAMPLE 8-2: 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
                BCF     EECON1, CFGS       ; Access EEPROM
                BSF     EECON1, WREN       ; Enable writes

                BCF     INTCON, GIE        ; Disable Interrupts
                MOVLW   55h                ;
Required        MOVWF   EECON2             ; Write 55h
Sequence        MOVLW   0AAh               ;
                MOVWF   EECON2             ; Write 0AAh
                BSF     EECON1, WR         ; Set WR bit to begin write
                BTFSC   EECON1, WR         ; Wait for write to complete
                BRA     $-2
                BSF     INTCON, GIE        ; Enable Interrupts

                                           ; User code execution
                BCF     EECON1, WREN       ; Disable writes on write complete (EEIF set)
```

## 8.6 Operation During Code-Protect

Data EEPROM memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if code protection is 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 25.0 "Special Features of the CPU"** for additional information.

## 8.7 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 implemented. On power-up, the WREN bit is cleared. In addition, writes to the EEPROM are blocked during the Power-up Timer period (TPWRT, parameter 33).

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

## 8.8 Using the Data EEPROM

The data EEPROM is a high-endurance, byte address-able 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 8-3.

| **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 8-3: 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   0AAh                ;
        MOVWF   EECON2              ; Write 0AAh
        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
```


© 2009 Microchip Technology Inc.

**TABLE 8-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| EEADR | EEPROM Address Register | | | | | | | | 57 |
| EEDATA | EEPROM Data Register | | | | | | | | 57 |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | 57 |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | 57 |
| IPR2 | OSCFIP | CMIP(1) | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP(1) | 57 |
| PIR2 | OSCFIF | CMIF(1) | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF(1) | 58 |
| PIE2 | OSCFIE | CMIE(1) | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE(1) | 58 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during Flash/EEPROM access.

**Note 1:** These bits are available in PIC18F4X80 devices and reserved in PIC18F2X80 devices.

**NOTES:**


© 2009 Microchip Technology Inc.

# 9.0 8 x 8 HARDWARE MULTIPLIER

## 9.1 Introduction

All PIC18 devices include an 8 x 8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. The multiplier's operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 9-1.

## 9.2 Operation

Example 9-1 shows the instruction sequence for an 8 x 8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 9-2 shows the sequence to do an 8 x 8 signed multiplication. To account for the signed bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

**EXAMPLE 9-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE**

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

**EXAMPLE 9-2: 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
```

**TABLE 9-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS**

| 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 | 4.0 μs | 16.0 μs | 40 μs |

Example 9-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 9-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES3:RES0).

**EQUATION 9-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM**

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

**EXAMPLE 9-3: 16 x 16 UNSIGNED 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         ;
```

Example 9-4 shows the sequence to do a 16 x 16 signed multiply. Equation 9-2 shows the algorithm used. The 32-bit result is stored in four registers (RES3:RES0). To account for the signed bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

**EQUATION 9-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM**

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H\langle 7\rangle \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H\langle 7\rangle \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

**EXAMPLE 9-4: 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
   :
```


© 2009 Microchip Technology Inc.

## 10.0 INTERRUPTS

The PIC18F2480/2580/4480/4580 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, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

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.

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

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.

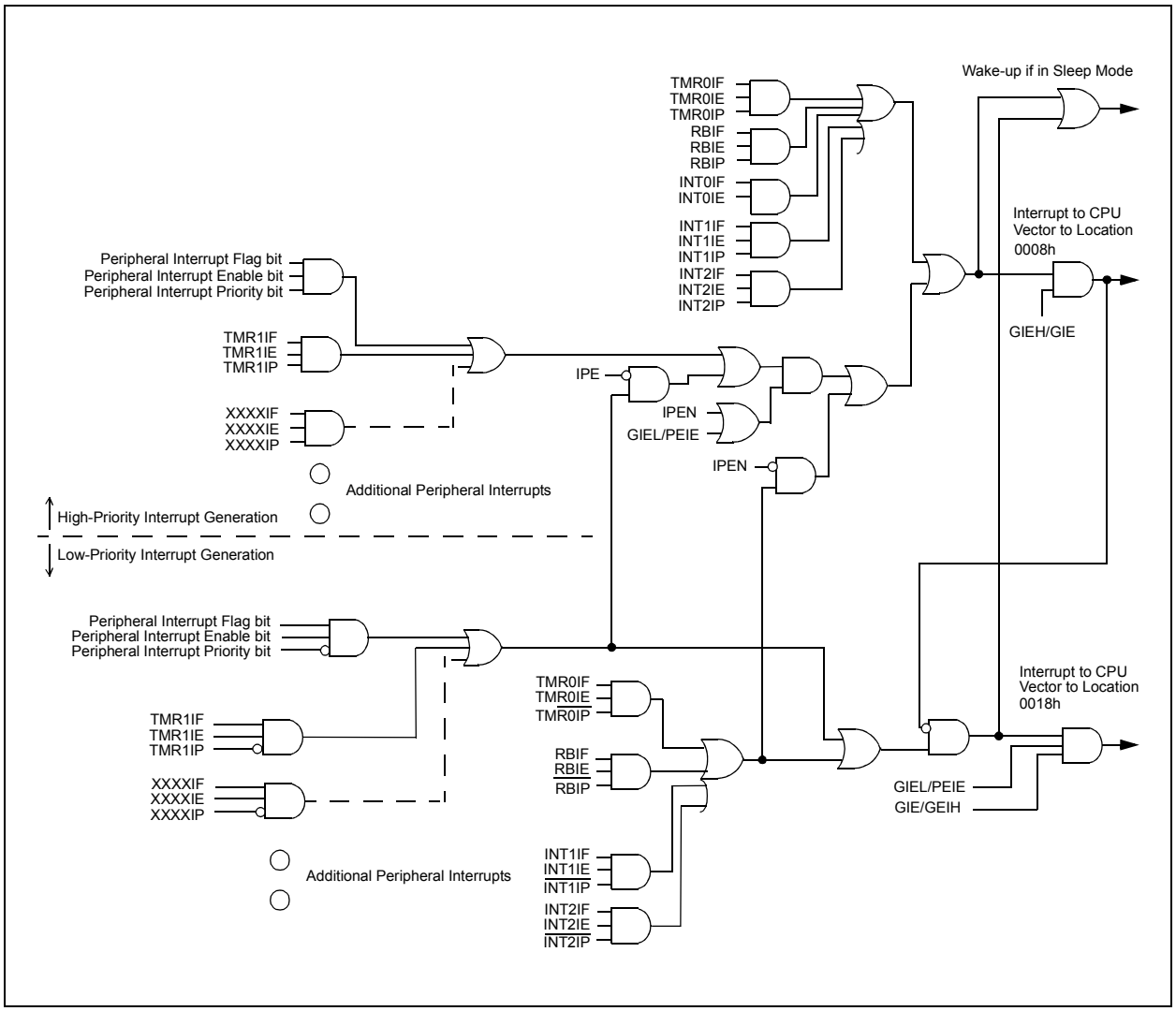
The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before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

**FIGURE 10-1: INTERRUPT LOGIC**





© 2009 Microchip Technology Inc.

## 10.1 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 10-1: INTCON: INTERRUPT CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-x |
|---|---|---|---|---|---|---|---|
| GIE/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' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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 high-priority 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:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
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.

**REGISTER 10-2: INTCON2: INTERRUPT CONTROL REGISTER 2**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | U-0 | R/W-1 | U-0 | R/W-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' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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.


© 2009 Microchip Technology Inc.

**REGISTER 10-3: INTCON3: INTERRUPT CONTROL REGISTER 3**

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

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 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.

## 10.2 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 10-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1**

| R/W-0 | R/W-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit(1)
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

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 **SSPIF:** Master Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

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

**Note 1:** This bit is reserved on PIC18F2X80 devices; always maintain this bit clear.


© 2009 Microchip Technology Inc.

**REGISTER 10-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2**

| R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| OSCFIF | CMIF(1) | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF(1) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 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 **CMIF:** Comparator Interrupt Flag bit(1)
1 = Comparator input has changed (must be cleared in software)
0 = Comparator input has not changed

bit 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 **BCLIF:** Bus Collision Interrupt Flag bit
1 = A bus collision occurred (must be cleared in software)
0 = No bus collision occurred

bit 2 **HLVDIF:** High/Low-Voltage Detect Interrupt Flag bit
1 = A low-voltage condition occurred (must be cleared in software)
0 = The device voltage is above the High/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 **ECCP1IF:** CCPx Interrupt Flag bit(1)
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.

**Note 1:** These bits are available in PIC18F4X80 and reserved in PIC18F2X80 devices.

**REGISTER 10-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3**

| **Mode 0** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|---|
| | IRXIF | WAKIF | ERRIF | TXB2IF | TXB1IF(1) | TXB0IF(1) | RXB1IF | RXB0IF |

| **Mode 1,2** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|---|
| | IRXIF | WAKIF | ERRIF | TXBnIF | TXB1IF(1) | TXB0IF(1) | RXBnIF | FIFOWMIF(1) |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **IRXIF:** CAN Invalid Received Message Interrupt Flag bit
1 = An invalid message has occurred on the CAN bus
0 = No invalid message on CAN bus

bit 6 **WAKIF:** CAN bus Activity Wake-up Interrupt Flag bit
1 = Activity on CAN bus has occurred
0 = No activity on CAN bus

bit 5 **ERRIF:** CAN bus Error Interrupt Flag bit
1 = An error has occurred in the CAN module (multiple sources)
0 = No CAN module errors

bit 4 When CAN is in Mode 0:
**TXB2IF:** CAN Transmit Buffer 2 Interrupt Flag bit
1 = Transmit Buffer 2 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 2 has not completed transmission of a message
When CAN is in Mode 1 or 2:
**TXBnIF:** Any Transmit Buffer Interrupt Flag bit
1 = One or more transmit buffers have completed transmission of a message and may be reloaded
0 = No transmit buffer is ready for reload

bit 3 **TXB1IF:** CAN Transmit Buffer 1 Interrupt Flag bit(1)
1 = Transmit Buffer 1 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 1 has not completed transmission of a message

bit 2 **TXB0IF:** CAN Transmit Buffer 0 Interrupt Flag bit(1)
1 = Transmit Buffer 0 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 0 has not completed transmission of a message

bit 1 When CAN is in Mode 0:
**RXB1IF:** CAN Receive Buffer 1 Interrupt Flag bit
1 = Receive Buffer 1 has received a new message
0 = Receive Buffer 1 has not received a new message
When CAN is in Mode 1 or 2:
**RXBnIF:** Any Receive Buffer Interrupt Flag bit
1 = One or more receive buffers has received a new message
0 = No receive buffer has received a new message

bit 0 When CAN is in Mode 0:
**RXB0IF:** CAN Receive Buffer 0 Interrupt Flag bit
1 = Receive Buffer 0 has received a new message
0 = Receive Buffer 0 has not received a new message
When CAN is in Mode 1:
**Unimplemented:** Read as '0'
When CAN is in Mode 2:
**FIFOWMIF:** FIFO Watermark Interrupt Flag bit(1)
1 = FIFO high watermark is reached
0 = FIFO high watermark is not reached

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.


© 2009 Microchip Technology Inc.

## 10.3 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 10-7: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit(1)
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

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 **SSPIE:** Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

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

**Note 1:** This bit is reserved on PIC18F2X80 devices; always maintain this bit clear.

**REGISTER 10-8: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2**

| R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| OSCFIE | CMIE(1) | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE(2) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **CMIE:** Comparator Interrupt Enable bit(1)
1 = Enabled
0 = Disabled

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

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

bit 3 **BCLIE:** Bus Collision Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **HLVDIE:** High/Low-Voltage Detect Interrupt Enable bit
1 = Enabled
0 = Disabled

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

bit 0 **ECCP1IE:** CCP1 Interrupt Enable bit(2)
1 = Enabled
0 = Disabled

**Note 1:** This bit is available in PIC18F4X80 devices and reserved in PIC18F2X80 devices.
**2:** This bit is available in PIC18F4X80 devices only.


© 2009 Microchip Technology Inc.

**REGISTER 10-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3**

| Mode 0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|---|
| | IRXIE | WAKIE | ERRIE | TXB2IE | TXB1IE(1) | TXB0IE(1) | RXB1IE | RXB0IE |

| Mode 1,2 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|---|
| | IRXIE | WAKIE | ERRIE | TXBnIE | TXB1IE(1) | TXB0IE(1) | RXBnIE | FIFOWMIE(1) |
| | bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **IRXIE:** CAN Invalid Received Message Interrupt Enable bit
1 = Enable invalid message received interrupt
0 = Disable invalid message received interrupt

bit 6 **WAKIE:** CAN bus Activity Wake-up Interrupt Enable bit
1 = Enable bus activity wake-up interrupt
0 = Disable bus activity wake-up interrupt

bit 5 **ERRIE:** CAN bus Error Interrupt Enable bit
1 = Enable CAN bus error interrupt
0 = Disable CAN bus error interrupt

bit 4 When CAN is in Mode 0:
**TXB2IE:** CAN Transmit Buffer 2 Interrupt Enable bit
1 = Enable Transmit Buffer 2 interrupt
0 = Disable Transmit Buffer 2 interrupt
When CAN is in Mode 1 or 2:
**TXBnIE:** CAN Transmit Buffer Interrupts Enable bit
1 = Enable transmit buffer interrupt; individual interrupt is enabled by TXBIE and BIE0
0 = Disable all transmit buffer interrupts

bit 3 **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 1 interrupt
0 = Disable Transmit Buffer 1 interrupt

bit 2 **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 0 interrupt
0 = Disable Transmit Buffer 0 interrupt

bit 1 When CAN is in Mode 0:
**RXB1IE:** CAN Receive Buffer 1 Interrupt Enable bit
1 = Enable Receive Buffer 1 interrupt
0 = Disable Receive Buffer 1 interrupt
When CAN is in Mode 1 or 2:
**RXBnIE:** CAN Receive Buffer Interrupts Enable bit
1 = Enable receive buffer interrupt; individual interrupt is enabled by BIE0
0 = Disable all receive buffer interrupts

bit 0 When CAN is in Mode 0:
**RXB0IE:** CAN Receive Buffer 0 Interrupt Enable bit
1 = Enable Receive Buffer 0 interrupt
0 = Disable Receive Buffer 0 interrupt
When CAN is in Mode 1:
**Unimplemented:** Read as '0'
When CAN is in Mode 2:
**FIFOWMIE:** FIFO Watermark Interrupt Enable bit(1)
1 = Enable FIFO watermark interrupt
0 = Disable FIFO watermark interrupt

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.

## 10.4 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 10-10: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **PSPIP:** Parallel Slave Port Read/Write Interrupt Priority bit(1)
1 = High priority
0 = Low priority

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 **SSPIP:** Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

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

**Note 1:** This bit is reserved on PIC18F2X80 devices; always maintain this bit set.


© 2009 Microchip Technology Inc.

**REGISTER 10-11: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2**

| R/W-1 | R/W-1 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| OSCFIP | CMIP(1) | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP(2) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **CMIP:** Comparator Interrupt Priority bit(1)
1 = High priority
0 = Low priority

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

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

bit 3 **BCLIP:** Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **HLVDIP:** High/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 **ECCP1IP:** CCP1 Interrupt Priority bit(2)
1 = High priority
0 = Low priority

**Note 1:** This bit is available in PIC18F4X80 devices and reserved in PIC18F2X80 devices.
**2:** This bit is available in PIC18F4X80 devices only.

**REGISTER 10-12: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 0** | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| | IRXIP | WAKIP | ERRIP | TXB2IP | TXB1IP(1) | TXB0IP(1) | RXB1IP | RXB0IP |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 1,2** | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| | IRXIP | WAKIP | ERRIP | TXBnIP | TXB1IP(1) | TXB0IP(1) | RXBnIP | FIFOWMIP |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **IRXIP:** CAN Invalid Received Message Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **WAKIP:** CAN bus Activity Wake-up Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **ERRIP:** CAN bus Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 When CAN is in Mode 0:
**TXB2IP:** CAN Transmit Buffer 2 Interrupt Priority bit
1 = High priority
0 = Low priority
When CAN is in Mode 1 or 2:
**TXBnIP:** CAN Transmit Buffer Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **TXB1IP:** CAN Transmit Buffer 1 Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 2 **TXB0IP:** CAN Transmit Buffer 0 Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 1 When CAN is in Mode 0:
**RXB1IP:** CAN Receive Buffer 1 Interrupt Priority bit
1 = High priority
0 = Low priority
When CAN is in Mode 1 or 2:
**RXBnIP:** CAN Receive Buffer Interrupts Priority bit
1 = High priority
0 = Low priority

bit 0 When CAN is in Mode 0:
**RXB0IP:** CAN Receive Buffer 0 Interrupt Priority bit
1 = High priority
0 = Low priority
When CAN is in Mode 1:
**Unimplemented:** Read as '0'
When CAN is in Mode 2:
**FIFOWMIP:** FIFO Watermark Interrupt Priority bit
1 = High priority
0 = Low priority

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.


© 2009 Microchip Technology Inc.

## 10.5 RCON Register

The RCON register contains flag bits which are used to determine the cause of the last Reset or wake-up from Idle or Sleep modes. RCON also contains the IPEN bit which enables interrupt priorities.

**REGISTER 10-13: RCON: RESET CONTROL REGISTER**

| R/W-0 | R/W-1(1) | U-0 | R/W-1 | R-1 | R-1 | R/W-0(2) | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | SBOREN | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **SBOREN:** BOR Software Enable bit(1)
For details of bit operation, see Register 5-1.

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

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
For details of bit operation, see Register 5-1.

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
For details of bit operation, see Register 5-1.

bit 2 **$\overline{PD}$:** Power-Down Detection Flag bit
For details of bit operation, see Register 5-1.

bit 1 **$\overline{POR}$:** Power-on Reset Status bit(2)
For details of bit operation, see Register 5-1.

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
For details of bit operation, see Register 5-1.

**Note 1:** If SBOREN is enabled, its Reset state is '1'; otherwise, it is '0'.
**2:** The actual Reset value of $\overline{POR}$ is determined by the type of device Reset. See Register 5-1 for additional information.

## 10.6 INTx Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1 and RB2/INT2 pins are edge-triggered. If the corresponding INTEDGx bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. 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 the power-managed modes, if bit INTxE was set prior to going into power-managed 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.

## 10.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set 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 14.0 "Timer2 Module"** for further details on the Timer0 module.

## 10.8 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>).

## 10.9 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 6.3 "Data Memory Organization"**), 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 10-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

**EXAMPLE 10-1: 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
```


© 2009 Microchip Technology Inc.

## 11.0 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.

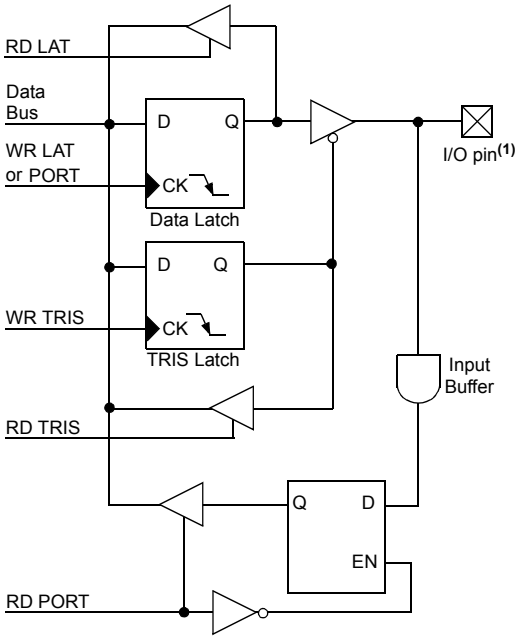
Each port has three registers for its operation. These registers are:

- TRIS register (Data Direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (Output Latch register)

The Output Latch register (LAT) 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 11-1.

**FIGURE 11-1: GENERIC I/O PORT OPERATION**




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

## 11.1 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 register 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 Output 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. 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 25.1 "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 comparator voltage reference output. The operation of pins, RA<3:0> and RA5 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, RA5 and RA<3:0> are configured as analog inputs and read as '0'. RA4 is configured as a digital input. |
|---|---|

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 11-1: INITIALIZING PORTA**

```
CLRF    PORTA   ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF    LATA    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0Fh     ; Configure A/D
MOVWF   ADCON1  ; for digital inputs
MOVWF   07h     ; Configure comparators
MOVWF   CMCON   ; for digital input
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISA   ; Set RA<3:0> as inputs
                ; RA<5:4> as outputs
```

**TABLE 11-1: PORTA I/O SUMMARY**

| Pin Name | Function | I/O | TRIS | Buffer | Description |
|---|---|---|---|---|---|
| RA0/AN0/CVREF | RA0 | OUT | 0 | DIG | LATA<0> data output. |
| | | IN | 1 | TTL | PORTA<0> data input. |
| | AN0 | IN | 1 | ANA | A/D Input Channel 0. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| | CVREF(1) | OUT | x | ANA | Comparator voltage reference analog output. Enabling this analog output overrides the digital I/O (read as clear – low level). |
| RA1/AN1 | RA1 | OUT | 0 | DIG | LATA<1> data output. |
| | | IN | 1 | TTL | PORTA<1> data input. |
| | AN1 | IN | 1 | ANA | A/D Input Channel 1. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| RA2/AN2/VREF- | RA2 | OUT | 0 | DIG | LATA<2> data output. |
| | | IN | 1 | TTL | PORTA<2> data input. |
| | AN2 | IN | 1 | ANA | A/D Input Channel 2. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| | VREF- | IN | 1 | ANA | A/D and comparator negative voltage analog input. |
| RA3/AN3/VREF+ | RA3 | OUT | 0 | DIG | LATA<3> data output. |
| | | IN | 1 | TTL | PORTA<3> data input. |
| | AN3 | IN | 1 | ANA | A/D Input Channel 3. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| | VREF+ | IN | 1 | ANA | A/D and comparator positive voltage analog input. |
| RA4/T0CKI | RA4 | OUT | 0 | DIG | LATA<4> data output. |
| | | IN | 1 | TTL | PORTA<4> data input. |
| | T0CKI | IN | 1 | ST | Timer0 clock input. |
| RA5/AN4/$\overline{SS}$/HLVDIN | RA5 | OUT | 0 | DIG | LATA<5> data output. |
| | | IN | 1 | TTL | PORTA<5> data input. |
| | AN4 | IN | 1 | ANA | A/D Input Channel 4. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| | $\overline{SS}$ | IN | 1 | TTL | Slave select input for MSSP. |
| | HLVDIN | IN | 1 | ANA | High/Low-Voltage Detect external trip point input. |
| OSC2/CLKO/RA6 | OSC2 | OUT | x | ANA | Output connection; selected by FOSC<3:0> Configuration bits. Enabling OSC2 overrides digital I/O. |
| | CLKO | OUT | x | DIG | Output connection; selected by FOSC<3:0> Configuration bits. Enabling CLKO overrides digital I/O (FOSC/4). |
| | RA6 | OUT | 0 | DIG | LATA<6> data output. |
| | | IN | 1 | TTL | PORTA<6> data input. |
| OSC1/CLKI/RA7 | OSC1 | IN | x | ANA | Main oscillator input connection determined by FOSC<3:0> Configuration bits. Enabling OSC1 overrides digital I/O. |
| | CLKI | IN | x | ANA | Main clock input connection determined by FOSC<3:0> Configuration bits. Enabling CLKI overrides digital I/O. |
| | RA7 | OUT | 0 | DIG | LATA<7> data output. |
| | | IN | 1 | TTL | PORTA<7> data input. |

**Legend:** OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input

**Note 1:** Available on 40/44-pin devices only.


© 2009 Microchip Technology Inc.

**TABLE 11-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTA | RA7(1) | RA6(1) | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 58 |
| LATA | LATA7(1) | LATA6(1) | LATA Output Latch Register | | | | | | 58 |
| TRISA | TRISA7(1) | TRISA6(1) | PORTA Data Direction Register | | | | | | 58 |
| ADCON1 | — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 56 |
| CVRCON(2) | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PORTA.

**Note 1:** RA<7:6> 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:** These registers are unimplemented on PIC18F2X80 devices.

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

Pins, RB2 through RB3, are multiplexed with the ECAN peripheral. Refer to **Section 24.0 "ECAN Module"** for proper settings of TRISB when CAN is enabled.

**EXAMPLE 11-2: INITIALIZING PORTB**

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0Eh     ; Set RB<4:0> as
MOVWF   ADCON1  ; digital I/O pins
                ; (required if config bit
                ; PBADEN is set)
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

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 all device resets.

| **Note:** | On a Power-on Reset, RB4, RB1 and RB0 are configured as analog inputs by default and read as '0'; RB<7:5> and RB<3:2> are configured as digital inputs. |
|---|---|

Four of the PORTB pins (RB<7:4>) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB<7:4> pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB<7:4>) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB<7:4> are ORed 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 (ANY), PORTB instruction). This will end the mismatch condition.
b) 1 TCY.
c) Clear flag bit, RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB and waiting 1 TCY 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.


© 2009 Microchip Technology Inc.

**TABLE 11-3: PORTB I/O SUMMARY**

| Pin Name | Function | I/O | TRIS | Buffer | Description |
|---|---|---|---|---|---|
| RB0/INT0/FLT0/AN10 | RB0 | OUT | 0 | DIG | LATB<0> data output. |
| | | IN | 1 | TTL | PORTB<0> data input. Weak pull-up available only in this mode. |
| | INT0 | IN | 1 | ST | External Interrupt 0 input. |
| | FLT0(1) | IN | 1 | ST | Enhanced PWM Fault input. |
| | AN10 | IN | 1 | ANA | A/D Input Channel 10. Enabled on POR, this analog input overrides the digital input (read as clear – low level). |
| RB1/INT1/AN8 | RB1 | OUT | 0 | DIG | LATB<1> data output. |
| | | IN | 1 | TTL | PORTB<1> data input. Weak pull-up available only in this mode. |
| | INT1 | IN | 1 | ST | External Interrupt 1 input. |
| | AN8 | IN | 1 | ANA | A/D Input Channel 8. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| RB2/INT2/CANTX | RB2 | OUT | x | DIG | LATB<2> data output. |
| | | IN | 1 | TTL | PORTB<2> data input. Weak pull-up available only in this mode. |
| | INT2 | IN | 1 | ST | External Interrupt 2 input. |
| | CANTX | OUT | 1 | DIG | CAN transmit signal output. The CAN interface overrides the TRIS<2> control when enabled. |
| RB3/CANRX | RB3 | OUT | 0 | DIG | LATB<3> data output. |
| | | IN | 1 | TTL | PORTB<3> data input. Weak pull-up available only in this mode. |
| | CANRX | IN | 1 | ST | CAN receive signal input. Pin must be configured as a digital input by setting TRISB<3>. |
| RB4/KBI0/AN9 | RB4 | OUT | 0 | DIG | LATB<4> data output. |
| | | IN | 1 | TTL | PORTB<4> data input. Weak pull-up available only in this mode. |
| | KBI0 | IN | 1 | TTL | Interrupt-on-pin change. |
| | AN9 | IN | 1 | ANA | A/D Input Channel 9. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| RB5/KBI1/PGM | RB5 | OUT | 0 | DIG | LATB<5> data output. |
| | | IN | 1 | TTL | PORTB<5> data input. Weak pull-up available only in this mode. |
| | KBI1 | IN | 1 | TTL | Interrupt-on-pin change. |
| | PGM | IN | x | ST | Low-Voltage Programming mode entry (ICSP™). Enabling this function overrides digital output. |
| RB6/KBI2/PGC | RB6 | OUT | 0 | DIG | LATB<6> data output. |
| | | IN | 1 | TTL | PORTB<6> data input. Weak pull-up available only in this mode. |
| | KBI2 | IN | 1 | TTL | Interrupt-on-pin change. |
| | PGC | IN | x | ST | Low-Voltage Programming mode entry (ICSP) clock input. |
| RB7/KBI3/PGD | RB7 | OUT | 0 | DIG | LATB<7> data output. |
| | | IN | 1 | TTL | PORTB<7> data input. Weak pull-up available only in this mode. |
| | KBI3 | IN | 1 | TTL | Interrupt-on-pin change. |
| | PGD | OUT | x | DIG | Low-Voltage Programming mode entry (ICSP) clock output. |
| | | IN | x | ST | Low-Voltage Programming mode entry (ICSP) clock input. |

**Legend:** OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input

**Note 1:** Available on 40/44-pin devices only.

**TABLE 11-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | 58 |
| LATB | LATB Output Latch Register | | | | | | | | 58 |
| TRISB | PORTB Data Direction Register | | | | | | | | 58 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RBIP | 55 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 55 |
| ADCON1 | — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 56 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PORTB.


© 2009 Microchip Technology Inc.

## 11.3 PORTC, TRISC and LATC Registers

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

The Output Latch register (LATC) is also memory mapped. Read-modify-write operations on the LATC register read and write the latched output value for PORTC.

PORTC is multiplexed with several peripheral functions (Table 11-5). The pins have Schmitt Trigger input buffers.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

| **Note:** | On a Power-on Reset, these pins are configured as digital inputs. |
|---|---|

The contents of the TRISC register are affected by peripheral overrides. Reading TRISC always returns the current contents, even though a peripheral device may be overriding one or more of the pins.

**EXAMPLE 11-3: INITIALIZING PORTC**

```
CLRF    PORTC   ; Initialize PORTC by
                ; clearing output
                ; data latches
CLRF    LATC    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISC   ; Set RC<3:0> as inputs
                ; RC<5:4> as outputs
                ; RC<7:6> as inputs
```

**TABLE 11-5: PORTC I/O SUMMARY**

| Pin Name | Function | I/O | TRIS | Buffer | Description |
|---|---|---|---|---|---|
| RC0/T1OSO/ T13CKI | RC0 | OUT | 0 | DIG | LATC<0> data output. |
| | | IN | 1 | ST | PORTC<0> data input. |
| | T1OSO | OUT | x | ANA | Timer1 oscillator output – overrides the TRIS<0> control when enabled. |
| | T13CKI | IN | 1 | ST | Timer1/Timer3 clock input. |
| RC1/T1OSI | RC1 | OUT | 0 | DIG | LATC<1> data output. |
| | | IN | 1 | ST | PORTC<1> data input. |
| | T1OSI | IN | x | ANA | Timer1 oscillator input – overrides the TRIS<1> control when enabled. |
| RC2/CCP1 | RC2 | OUT | 0 | DIG | LATC<2> data output. |
| | | IN | 1 | ST | PORTC<2> data input. |
| | CCP1 | OUT | 0 | DIG | CCP1 compare output. |
| | | IN | 1 | ST | CCP1 capture input. |
| RC3/SCK/SCL | RC3 | OUT | 0 | DIG | LATC<3> data output. |
| | | IN | 1 | ST | PORTC<3> data input. |
| | SCK | OUT | 0 | DIG | SPI clock output (MSSP module) – must have TRIS set to '1' to allow MSSP module to control the bidirectional communication. |
| | | IN | 1 | ST | SPI clock input (MSSP module). |
| | SCL | OUT | 0 | DIG | $I^2C$™/SM bus clock output (MSSP module) – must have TRIS set to '1' to allow MSSP module to control the bidirectional communication. |
| | | IN | 1 | $I^2C$™/SMB | $I^2C$/SM bus clock input. |
| RC4/SDI/SDA | RC4 | OUT | 0 | DIG | LATC<4> data output. |
| | | IN | 1 | ST | PORTC<4> data input. |
| | SDI | IN | 1 | ST | SPI data input (MSSP module). |
| | SDA | OUT | 1 | DIG | $I^2C$/SM bus data output (MSSP module) – must have TRIS set to '1' to allow MSSP module to control the bidirectional communication. |
| | | IN | 1 | $I^2C$/SMB | $I^2C$/SM bus data input (MSSP module) – must have TRIS set to '1' to allow MSSP module to control the bidirectional communication. |
| RC5/SDO | RC5 | OUT | 0 | DIG | LATC<5> data output. |
| | | IN | 1 | ST | PORTC<5> data input. |
| | SDO | OUT | 0 | DIG | SPI data output (MSSP module). |
| RC6/TX/CK | RC6 | OUT | 0 | DIG | LATC<6> data output. |
| | | IN | 1 | ST | PORTC<6> data input. |
| | TX | OUT | 0 | DIG | EUSART data output. |
| | CK | OUT | 1 | DIG | EUSART synchronous clock output – must have TRIS set to '1' to enable EUSART to control the bidirectional communication. |
| | | IN | 1 | ST | EUSART synchronous clock input. |
| RC7/RX/DT | RC7 | OUT | 0 | DIG | LATC<7> data output. |
| | | IN | 1 | ST | PORTC<7> data input. |
| | RX | IN | 1 | ST | EUSART asynchronous data input. |
| | DT | OUT | 1 | DIG | EUSART synchronous data output – must have TRIS set to '1' to enable EUSART to control the bidirectional communication. |
| | | IN | 1 | ST | EUSART synchronous data input. |

**Legend:** OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input

**TABLE 11-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | 58 |
| LATC | LATC Output Latch Register | | | | | | | | 58 |
| TRISC | PORTC Data Direction Register | | | | | | | | 58 |


© 2009 Microchip Technology Inc.

## 11.4 PORTD, TRISD and LATD Registers

| **Note:** | PORTD is only available on PIC18F4X80 devices. |
|---|---|

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

The Output Latch register (LATD) is also memory mapped. Read-modify-write operations on the LATD register read and write the latched output value for PORTD.

All pins on PORTD are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Four of the PORTD pins are multiplexed with outputs P1A, P1B, P1C and P1D of the Enhanced CCP module. The operation of these additional PWM output pins is covered in greater detail in **Section 17.0 "Enhanced Capture/Compare/PWM (ECCP) Module"**.

Four of the PORTD pins are multiplexed with the input pins of the comparators. The operation of these input pins is covered in greater detail in **Section 21.0 "Comparator Module"**.

| **Note:** | On a Power-on Reset, these pins are configured as analog inputs. |
|---|---|

PORTD can also be configured as an 8-bit wide microprocessor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL. See **Section 11.6 "Parallel Slave Port"** for additional information on the Parallel Slave Port (PSP).

**EXAMPLE 11-4: INITIALIZING PORTD**

```
CLRF    PORTD   ; Initialize PORTD by
                ; clearing output
                ; data latches
CLRF    LATD    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISD   ; Set RD<3:0> as inputs
                ; RD<5:4> as outputs
                ; RD<7:6> as inputs
```

**TABLE 11-7: PORTD I/O SUMMARY**

| Pin Name | Function | I/O | TRIS | Buffer | Description |
|---|---|---|---|---|---|
| RD0/PSP0/ C1IN+ | RD0 | OUT | 0 | DIG | LATD<0> data output. |
| | | IN | 1 | ST | PORTD<0> data input. |
| | PSP0 | OUT | x | DIG | Parallel Slave Port (PSP) data output (overrides the TRIS<0> control when enabled). |
| | | IN | x | TTL | Parallel Slave Port (PSP) data input (overrides the TRIS<0> control when enabled). |
| | C1IN+ | IN | 1 | ANA | Comparator 1 Positive Input B. Default on POR. This analog input overrides the digital input (read as clear – low level). |
| RD1/PSP1/ C1IN- | RD1 | OUT | 0 | DIG | LATD<1> data output. |
| | | IN | 1 | ST | PORTD<1> data input. |
| | PSP1 | OUT | x | DIG | Parallel Slave Port (PSP) data output (overrides the TRIS<1> control when enabled). |
| | | IN | x | TTL | Parallel Slave Port (PSP) data input (overrides the TRIS<1> control when enabled). |
| | C1IN- | IN | 1 | ANA | Comparator 1 negative input. Default on POR. This analog input overrides the digital input (read as clear – low level). |
| RD2/PSP2/ C2IN+ | RD2 | OUT | 0 | DIG | LATD<2> data output. |
| | | IN | 1 | ST | PORTD<2> data input. |
| | PSP2 | OUT | x | DIG | Parallel Slave Port (PSP) data output (overrides the TRIS<2> control when enabled). |
| | | IN | x | TTL | Parallel Slave Port (PSP) data input (overrides the TRIS<2> control when enabled). |
| | C2IN+ | IN | 1 | ANA | Comparator 2 positive input. Default on POR. This analog input overrides the digital input (read as clear – low level). |
| RD3/PSP3/ C2IN- | RD3 | OUT | 0 | DIG | LATD<3> data output. |
| | | IN | 1 | ST | PORTD<3> data input. |
| | PSP3 | OUT | x | DIG | Parallel Slave Port (PSP) data output (overrides the TRIS<3> control when enabled). |
| | | IN | x | TTL | Parallel Slave Port (PSP) data input (overrides the TRIS<3> control when enabled). |
| | C2IN- | IN | 1 | ANA | Comparator 2 negative input. Default input on POR. This analog input overrides the digital input (read as clear – low level). |
| RD4/PSP4/ ECCP1/P1A | RD4 | OUT | 0 | DIG | LATD<4> data output. |
| | | IN | 1 | ST | PORTD<4> data input. |
| | PSP4 | OUT | x | DIG | Parallel Slave Port (PSP) data output (overrides the TRIS<4> control when enabled). |
| | | IN | x | TTL | Parallel Slave Port (PSP) data input (overrides the TRIS<4> control when enabled). |
| | ECCP1 | OUT | 0 | DIG | ECCP1 compare output. |
| | | IN | 1 | ST | ECCP1 capture input. |
| | P1A | OUT | 0 | DIG | ECCP1 Enhanced PWM output, Channel A. |
| RD5/PSP5/ P1B | RD5 | OUT | 0 | DIG | LATD<5> data output. |
| | | IN | 1 | ST | PORTD<5> data input. |
| | PSP5 | OUT | X | DIG | Parallel Slave Port (PSP) data output (overrides the TRIS<5> control when enabled). |
| | | IN | x | TTL | Parallel Slave Port (PSP) data input (overrides the TRIS<5> control when enabled). |
| | P1B | OUT | 0 | DIG | ECCP1 Enhanced PWM output, Channel B. |
| RD6/PSP6/ P1C | RD6 | OUT | 0 | DIG | LATD<6> data output. |
| | | IN | 1 | ST | PORTD<6> data input. |
| | PSP6 | OUT | x | DIG | Parallel Slave Port (PSP) data output (overrides the TRIS<6> control when enabled). |
| | | IN | x | TTL | Parallel Slave Port (PSP) data input (overrides the TRIS<6> control when enabled). |
| | P1C | OUT | 0 | DIG | ECCP1 Enhanced PWM output, Channel C. |
| RD7/PSP7/ P1D | RD7 | OUT | 0 | DIG | LATD<7> data output. |
| | | IN | 1 | ST | PORTD<7> data input. |
| | PSP7 | OUT | x | DIG | Parallel Slave Port (PSP) data output (overrides the TRIS<7> control when enabled). |
| | | IN | x | TTL | Parallel Slave Port (PSP) data input (overrides the TRIS<7> control when enabled). |
| | P1D | OUT | 0 | DIG | ECCP1 Enhanced PWM output, channel D. |

**Legend:** OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input


© 2009 Microchip Technology Inc.

**TABLE 11-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTD(1) | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | 58 |
| LATD(1) | LATD Output Latch Register | | | | | | | | 58 |
| TRISD(1) | PORTD Data Direction Register | | | | | | | | 58 |
| TRISE(1) | IBF | OBF | IBOV | PSPMODE | — | TRISE2 | TRISE1 | TRISE0 | 58 |
| ECCP1CON(1) | EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PORTD.

**Note 1:** These registers are available on PIC18F4X80 devices only.

## 11.5 PORTE, TRISE and LATE Registers

Depending on the particular PIC18F2480/2580/4480/4580 device selected, PORTE is implemented in two different ways.

For PIC18F4X80 devices, PORTE is a 4-bit wide port. Three pins (RE0/$\overline{RD}$/AN5, RE1/$\overline{WR}$/AN6/C1OUT and RE2/$\overline{CS}$/AN7/C2OUT) are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. When selected as an analog input, these pins will read as '0's.

The corresponding Data Direction register is TRISE. Setting a TRISE bit (= 1) will make the corresponding PORTE pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISE bit (= 0) will make the corresponding PORTE pin an output (i.e., put the contents of the output latch on the selected pin).

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

| **Note:** | On a Power-on Reset, RE<2:0> are configured as analog inputs. |
|---|---|

The upper four bits of the TRISE register also control the operation of the Parallel Slave Port. Their operation is explained in Register 11-1.

The Output Latch register (LATE) is also memory mapped. Read-modify-write operations on the LATE register, read and write the latched output value for PORTE.

The fourth pin of PORTE ($\overline{MCLR}$/VPP/RE3) is an input only pin. Its operation is controlled by the MCLRE Configuration bit. 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, RE3 also functions as the programming voltage input during programming.

| **Note:** | On a Power-on Reset, RE3 is enabled as a digital input only if Master Clear functionality is disabled. |
|---|---|

**EXAMPLE 11-5: INITIALIZING PORTE**

```
CLRF    PORTE   ; Initialize PORTE by
                ; clearing output
                ; data latches
CLRF    LATE    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0Ah     ; Configure A/D
MOVWF   ADCON1  ; for digital inputs
MOVLW   03h     ; Value used to
                ; initialize data
                ; direction
MOVLW   07h     ; Turn off
MOVWF   CMCON   ; comparators
MOVWF   TRISC   ; Set RE<0> as inputs
                ; RE<1> as outputs
                ; RE<2> as inputs
```

### 11.5.1 PORTE IN 28-PIN DEVICES

For PIC18F2X80 devices, PORTE is only available when Master Clear functionality is disabled (MCLRE = 0). In these cases, PORTE is a single bit, input only port comprised of RE3 only. The pin operates as previously described.


© 2009 Microchip Technology Inc.

**REGISTER 11-1: TRISE REGISTER (PIC18F4X80 DEVICES ONLY)**

| R-0 | R-0 | R/W-0 | R/W-0 | U-0 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| IBF | OBF | IBOV | PSPMODE | — | TRISE2 | TRISE1 | TRISE0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **IBF:** Input Buffer Full Status bit
1 = A word has been received and waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF:** Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV:** Input Buffer Overflow Detect bit (in Microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE:** Parallel Slave Port Mode Select bit
1 = Parallel Slave Port mode
0 = General purpose I/O mode

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

bit 2 **TRISE2:** RE2 Direction Control bit
1 = Input
0 = Output

bit 1 **TRISE1:** RE1 Direction Control bit
1 = Input
0 = Output

bit 0 **TRISE0:** RE0 Direction Control bit
1 = Input
0 = Output

**TABLE 11-9: PORTE I/O SUMMARY**

| Pin Name | Function | I/O | TRIS | Buffer | Description |
|---|---|---|---|---|---|
| RE0/$\overline{RD}$/AN5 | RE0 | OUT | 0 | DIG | LATE<0> data output. |
| | | IN | 1 | ST | PORTE<0> data input. |
| | $\overline{RD}$ | IN | 1 | TTL | PSP read enable input. |
| | AN5 | IN | 1 | ANA | A/D Input Channel 5. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| RE1/$\overline{WR}$/AN6/C1OUT | RE1 | OUT | 0 | DIG | LATE<1> data output. |
| | | IN | 1 | ST | PORTE<1> data input. |
| | $\overline{WR}$ | IN | 1 | TTL | PSP write enable input. |
| | AN6 | IN | 1 | ANA | A/D Input Channel 6. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| | C1OUT | OUT | 0 | DIG | Comparator 1 output. |
| RE2/$\overline{CS}$/AN7/C2OUT | RE2 | OUT | 0 | DIG | LATE<2> data output. |
| | | IN | 1 | ST | PORTE<2> data input. |
| | $\overline{CS}$ | IN | 1 | TTL | PSP chip select input. |
| | AN7 | IN | 1 | ANA | A/D Input Channel 7. Enabled on POR; this analog input overrides the digital input (read as clear – low level). |
| | C2OUT | OUT | 0 | DIG | Comparator 2 output. |
| $\overline{MCLR}$/VPP/RE3 | $\overline{MCLR}$ | IN | x | ST | External Reset input. Disabled when MCLRE Configuration bit is '1'. |
| | VPP | IN | x | ANA | High-voltage detection; used by ICSP™ operation. |
| | RE3 | IN | 1 | ST | PORTE<3> data input. Disabled when MCLRE Configuration bit is '0'. |

**Legend:** PWR = Power Supply, OUT = Output, IN = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input

**TABLE 11-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTE(3) | — | — | — | — | RE3(1,2) | RE2 | RE1 | RE0 | 58 |
| LATE(2) | — | — | — | — | — | LATE Output Latch Register | | | 58 |
| TRISE(3) | IBF | OBF | IBOV | PSPMODE | — | TRISE2 | TRISE1 | TRISE0 | 58 |
| ADCON1 | — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 56 |
| CMCON(3) | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PORTE.

**Note 1:** Implemented only when Master Clear functionality is disabled (MCLRE Configuration bit = 0).

**2:** RE3 is the only PORTE bit implemented on both PIC18F2X80 and PIC18F4X80 devices. All other bits are implemented only when PORTE is implemented (i.e., PIC18F4X80 devices).

**3:** These registers are unimplemented on PIC18F2X80 devices.


© 2009 Microchip Technology Inc.

## 11.6 Parallel Slave Port

**Note:** The Parallel Slave Port is only available on PIC18F4X80 devices.

In addition to its function as a general I/O port, PORTD can also operate as an 8-bit wide Parallel Slave Port (PSP) or microprocessor port. PSP operation is controlled by the 4 upper bits of the TRISE register (Register 11-1). Setting control bit, PSPMODE (TRISE<4>), enables PSP operation, as long as the Enhanced CCP module is not operating in dual output or quad output PWM mode. In Slave mode, the port is asynchronously readable and writable by the external world.

The PSP can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an 8-bit latch. Setting the control bit PSPMODE enables the PORTE I/O pins to become control inputs for the microprocessor port. When set, port pin RE0 is the $\overline{RD}$ input, RE1 is the $\overline{WR}$ input and RE2 is the $\overline{CS}$ (Chip Select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (set). The A/D port Configuration bits, PFCG<3:0> (ADCON1<3:0>), must also be set to '1010'.

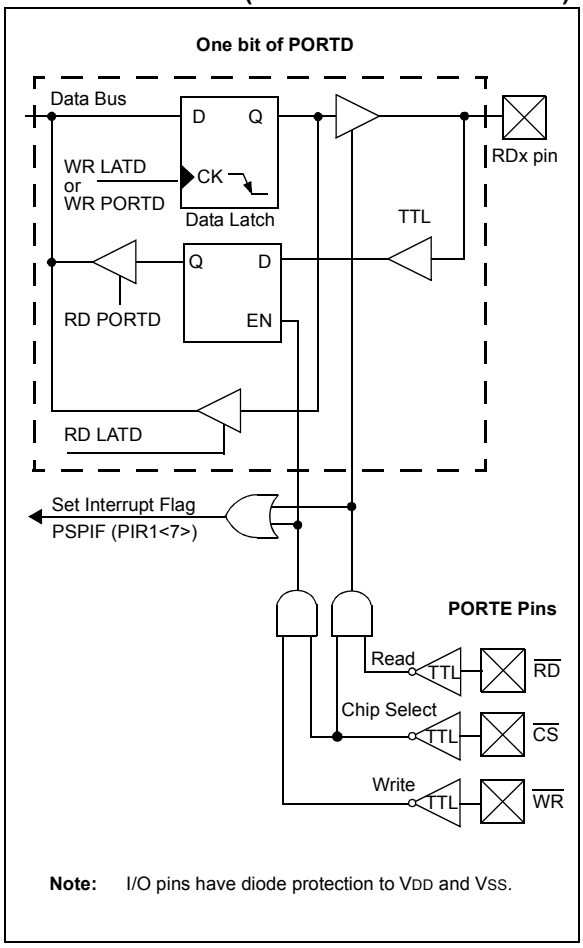
A write to the PSP occurs when both the $\overline{CS}$ and $\overline{WR}$ lines are first detected low and ends when either are detected high. The PSPIF and IBF flag bits are both set when the write ends.

A read from the PSP occurs when both the $\overline{CS}$ and $\overline{RD}$ lines are first detected low. The data in PORTD is read out and the OBF bit is set. If the user writes new data to PORTD to set OBF, the data is immediately read out; however, the OBF bit is not set.

When either the $\overline{CS}$ or $\overline{RD}$ lines are detected high, the PORTD pins return to the input state and the PSPIF bit is set. User applications should wait for PSPIF to be set before servicing the PSP; when this happens, the IBF and OBF bits can be polled and the appropriate action taken.

The timing for the control signals in Write and Read modes is shown in Figure 11-3 and Figure 11-4, respectively.

**FIGURE 11-2: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)**




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

**FIGURE 11-3: PARALLEL SLAVE PORT WRITE WAVEFORMS**

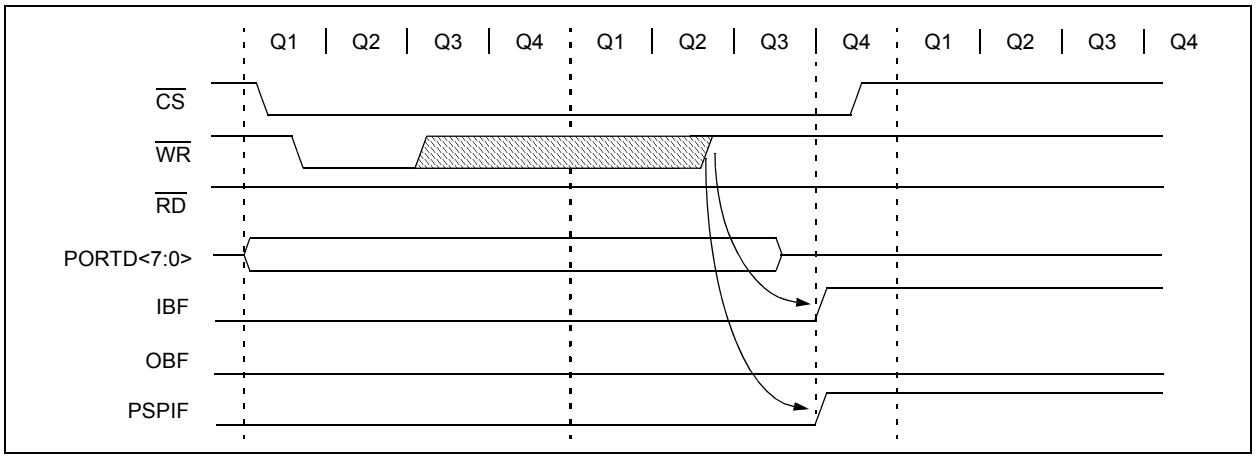



**FIGURE 11-4: PARALLEL SLAVE PORT READ WAVEFORMS**

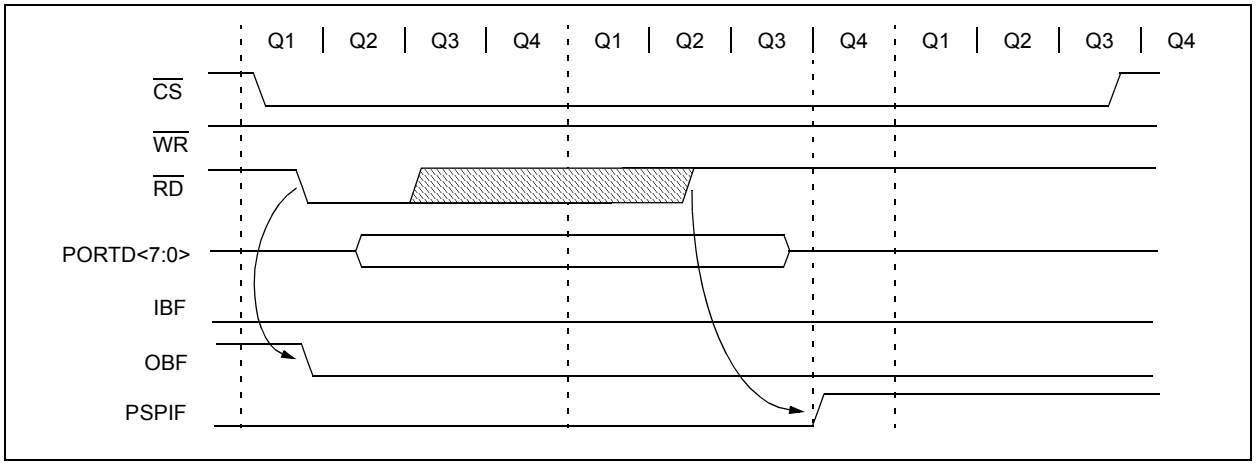



**TABLE 11-11: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTD(1) | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | 58 |
| LATD(1) | LATD Output Latch Register | | | | | | | | 58 |
| TRISD(1) | PORTD Data Direction Register | | | | | | | | 58 |
| PORTE(1) | — | — | — | — | RE3 | RE2 | RE1 | RE0 | 58 |
| LATE(1) | — | — | — | — | — | LATE Output Latch Register | | | 58 |
| TRISE(1) | IBF | OBF | IBOV | PSPMODE | — | TRISE2 | TRISE1 | TRISE0 | 58 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| ADCON1 | — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 56 |
| CMCON(1) | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Parallel Slave Port.

**Note 1:** These registers are available on PIC18F4X80 devices only.


© 2009 Microchip Technology Inc.

## 12.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software-selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 12-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 12-1. Figure 12-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

**REGISTER 12-1: T0CON: TIMER0 CONTROL REGISTER**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-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' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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 input edge
0 = Internal clock (FOSC/4)

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

## 12.1 Timer0 Operation

Timer0 can operate as either a timer or a counter; the mode is selected by clearing the T0CS bit (T0CON<5>). In Timer mode, the module increments on every clock by default unless a different prescaler value is selected (see **Section 12.3 "Prescaler"**). If the TMR0 register is written to, 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.

The Counter mode is selected by setting the T0CS bit (= 1). In Counter mode, Timer0 increments either on every rising or falling edge of pin, RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (T0CON<4>); clearing this bit selects the rising edge. Restrictions on the external clock input are discussed below.

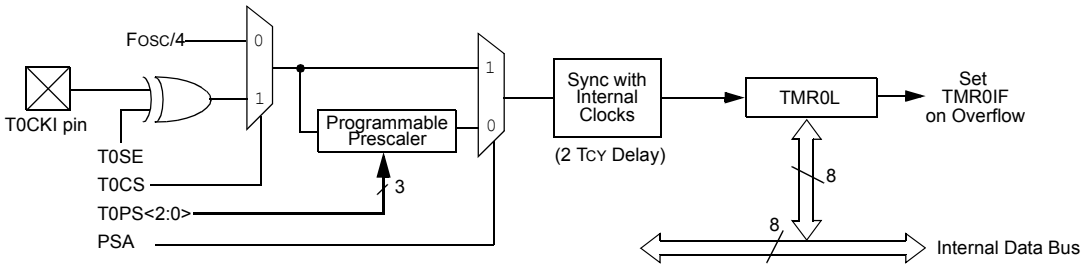
An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (TOSC). There is a delay between synchronization and the onset of incrementing the timer/counter.

## 12.2 Timer0 Reads and Writes in 16-Bit Mode

TMR0H is not the actual high byte of Timer0 in 16-bit mode; it is actually a buffered version of the real high byte of Timer0, which is not directly readable nor writable (refer to Figure 12-2). 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.

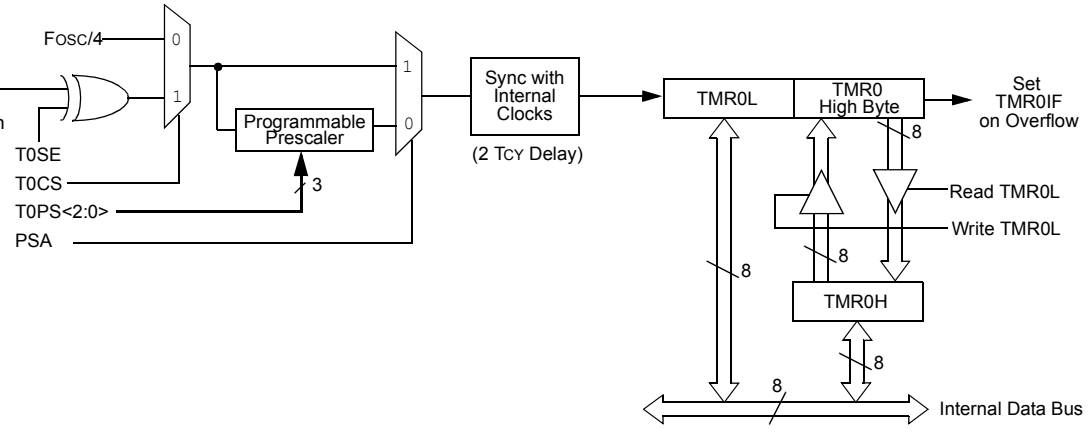
Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

**FIGURE 12-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)**




**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

**FIGURE 12-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)**




**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.


© 2009 Microchip Technology Inc.

## 12.3 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not directly readable or writable; its value is set by the PSA and T0PS<2:0> bits (T0CON<3:0>) which determine the prescaler assignment and prescale ratio.

Clearing the PSA bit assigns the prescaler to the Timer0 module. When it is assigned, prescale values from 1:2 through 1:256 in power-of-2 increments are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF TMR0`, `MOVWF TMR0`, `BSF TMR0`, etc.) 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. |
|---|---|

### 12.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control and can be changed “on-the-fly” during program execution.

## 12.4 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or from FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF flag bit. The interrupt can be masked by clearing the TMR0IE bit (INTCON<5>). Before re-enabling the interrupt, the TMR0IF bit must be cleared in software by the Interrupt Service Routine.

Since Timer0 is shut down in Sleep mode, the TMR0 interrupt cannot awaken the processor from Sleep.

**TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER0**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| TMR0L | Timer0 Register Low Byte | | | | | | | | 56 |
| TMR0H | Timer0 Register High Byte | | | | | | | | 56 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 56 |
| TRISA | TRISA7(1) | TRISA6(1) | PORTA Data Direction Register | | | | | | 58 |

**Legend:** — = unimplemented locations, read as ‘0’. Shaded cells are not used by Timer0.

**Note 1:** RA6/RA7 and their associated latch and direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as ‘0’.

**NOTES:**


© 2009 Microchip Technology Inc.

## 13.0 TIMER1 MODULE

The Timer1 timer/counter module incorporates these features:

- Software-selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR1H and TMR1L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Module Reset on CCP Special Event Trigger
- Device clock status flag (T1RUN)

A simplified block diagram of the Timer1 module is shown in Figure 13-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 13-2.

The module incorporates its own low-power oscillator to provide an additional clocking option. The Timer1 oscillator can also be used as a low-power clock source for the microcontroller in power-managed operation.

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.

Timer1 is controlled through the T1CON Control register (Register 13-1). It also contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

**REGISTER 13-1: T1CON: TIMER1 CONTROL REGISTER**

| R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-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' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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 = Device clock is derived from Timer1 oscillator
0 = Device 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 RC0/T1OSO/T13CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

## 13.1 Timer1 Operation

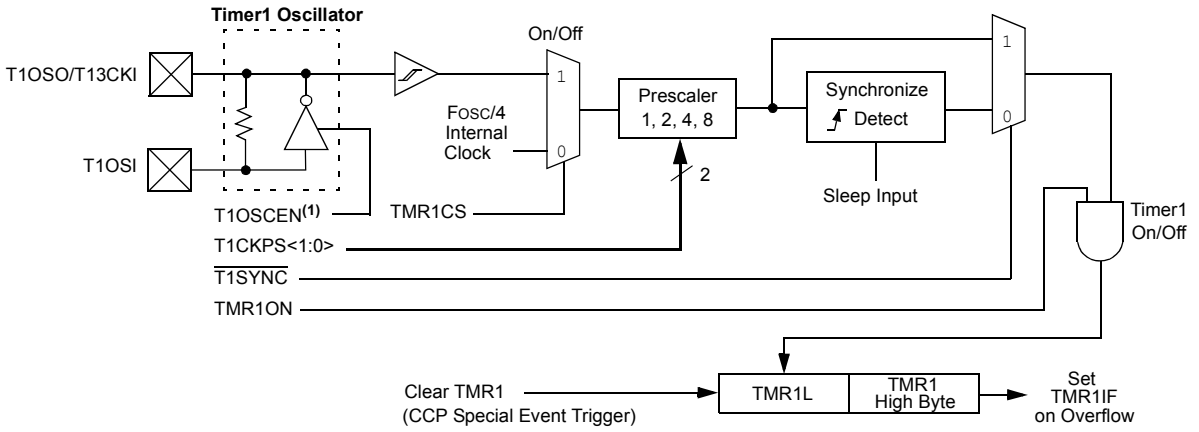
Timer1 can operate in one of these modes:

- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>). When TMR1CS is cleared (= 0), Timer1 increments on every internal instruction cycle (FOSC/4). When the bit is set, Timer1 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

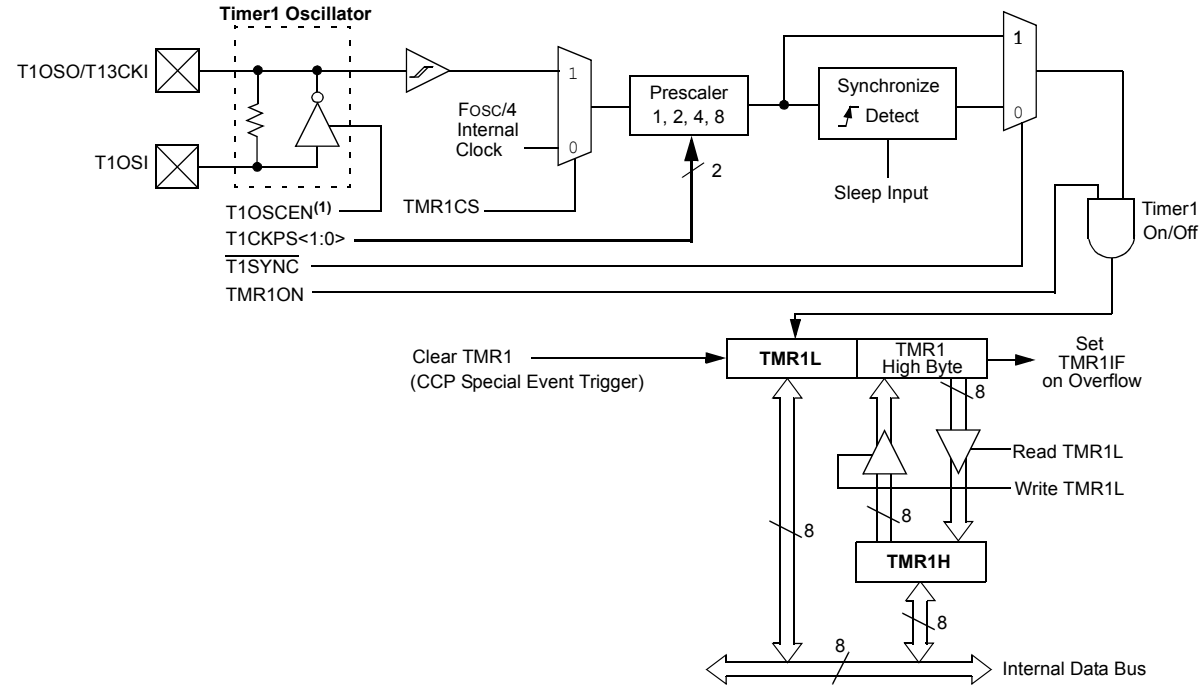
When Timer1 is enabled, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

**FIGURE 13-1: TIMER1 BLOCK DIAGRAM**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

**FIGURE 13-2: TIMER1 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.


© 2009 Microchip Technology Inc.

## 13.2 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 13-2). 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 register. 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, has become invalid due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H Buffer register. The 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.

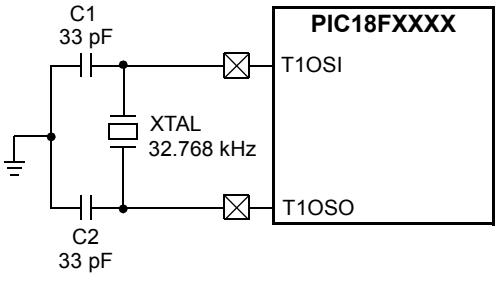
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

## 13.3 Timer1 Oscillator

An on-chip crystal oscillator circuit is incorporated between pins, T1OSI (input) and T1OSO (amplifier output). It is enabled by setting the Timer1 Oscillator Enable bit, T1OSCEN (T1CON<3>). The oscillator is a low-power circuit 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 13-3. Table 13-1 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.

**FIGURE 13-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR**




**Note:** See the Notes with Table 13-1 for additional information about capacitor selection.

**TABLE 13-1: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR(1-4)**

| Osc Type | Freq | C1 | C2 |
|---|---|---|---|
| LP | 32 kHz | 27 pF | 27 pF |

**Note 1:** Microchip suggests these values as a starting point in validating the oscillator circuit.

**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.

### 13.3.1 USING TIMER1 AS A CLOCK SOURCE

The Timer1 oscillator is also available as a clock source in power-managed modes. By setting the clock select bits, SCS<1:0> (OSCCON<1:0>), to '01', the device switches to SEC_RUN mode; both the CPU and peripherals are clocked from the Timer1 oscillator. If the IDLEN bit (OSCCON<7>) is cleared and a SLEEP instruction is executed, the device enters SEC_IDLE mode. Additional details are available in **Section 4.0 "Power-Managed Modes"**.

Whenever the Timer1 oscillator is providing the clock source, the Timer1 system clock status flag, T1RUN (T1CON<6>), is set. This can be used to determine the controller's current clocking mode. It can also indicate the clock source being currently used by the Fail-Safe Clock Monitor. If the Clock Monitor is enabled and the Timer1 oscillator fails while providing the clock, polling the T1RUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

### 13.3.2 LOW-POWER TIMER1 OPTION

The Timer1 oscillator can operate at two distinct levels of power consumption based on device configuration. When the LPT1OSC Configuration bit is set, the Timer1 oscillator operates in a low-power mode. When LPT1OSC is not set, Timer1 operates at a higher power level. Power consumption for a particular mode is relatively constant, regardless of the device's operating mode. The default Timer1 configuration is the higher power mode.

As the low-power Timer1 mode tends to be more sensitive to interference, high noise environments may cause some oscillator instability. The low-power option is, therefore, best suited for low noise applications where power conservation is an important design consideration.

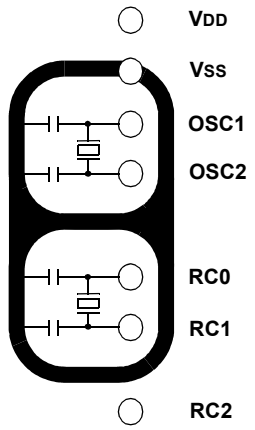
### 13.3.3 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

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 13-3, 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 13-4, may be helpful when used on a single-sided PCB or in addition to a ground plane.

**FIGURE 13-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING**




**Note:** Not drawn to scale.

## 13.4 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 or disabled by setting or clearing the Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

## 13.5 Resetting Timer1 Using the CCP Special Event Trigger

If either of the CCP modules is configured in Compare mode to generate a Special Event Trigger (CCP1M<3:0> or CCP2M<3:0> = 1011), this signal will reset Timer1. The trigger from ECCP1 will also start an A/D conversion if the A/D module is enabled (see **Section 16.3.4 "Special Event Trigger"** for more information.).

The module must be configured as either a timer or a synchronous counter to take advantage of this feature. When used this way, the CCPRH:CCPRL register pair effectively becomes a period register for Timer1.

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, the write operation will take precedence.

| **Note:** | The special event triggers from the ECCP1 module will not set the TMR1IF interrupt flag bit (PIR1<0>). |
|---|---|

## 13.6 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 13.3 "Timer1 Oscillator"**) 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 13-1, 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 2 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.


© 2009 Microchip Technology Inc.

**EXAMPLE 13-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE**

```
RTCinit
            MOVLW   80h             ; 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
```

**TABLE 13-2: 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 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| TMR1L | Timer1 Register Low Byte | | | | | | | | 56 |
| TMR1H | TImer1 Register High Byte | | | | | | | | 56 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 56 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

**Note 1:** These bits are unimplemented on PIC18F2X80 devices; always maintain these bits clear.

**NOTES:**


© 2009 Microchip Technology Inc.

## 14.0 TIMER2 MODULE

The Timer2 module timer incorporates the following features:

- 8-Bit Timer and Period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable postscaler (1:1 through 1:16)
- Interrupt on TMR2 to PR2 match
- Optional use as the shift clock for the MSSP module

The module is controlled through the T2CON register (Register 14-1), which enables or disables the timer and configures the prescaler and postscaler. Timer2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

A simplified block diagram of the module is shown in Figure 14-1.

### 14.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (FOSC/4). A 2-bit counter/prescaler on the clock input gives direct input, divide-by-4 and divide-by-16 prescale options; these are selected by the prescaler control bits, T2CKPS<1:0> (T2CON<1:0>). The value of TMR2 is compared to that of the Period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/postscaler (see **Section 14.2 "Timer2 Interrupt"**).

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, while the PR2 register initializes at FFh. Both the prescaler and postscaler counters are cleared on the following events:

- 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 14-1: T2CON: TIMER2 CONTROL REGISTER**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6-3 **T2OUTPS<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

## 14.2 Timer2 Interrupt

Timer2 also can generate an optional device interrupt. The Timer2 output signal (TMR2 to PR2 match) provides the input for the 4-bit output counter/post-scaler. This counter generates the TMR2 match interrupt flag which is latched in TMR2IF (PIR1<1>). The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE (PIE1<1>).

A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS<3:0> (T2CON<6:3>).

## 14.3 TMR2 Output

The unscaled output of TMR2 is available primarily to the CCP modules, where it is used as a time base for operations in PWM mode.

Timer2 can be optionally used as the shift clock source for the MSSP module operating in SPI mode. Additional information is provided in **Section 18.0 "Master Synchronous Serial Port (MSSP) Module"**.

**FIGURE 14-1: TIMER2 BLOCK DIAGRAM**

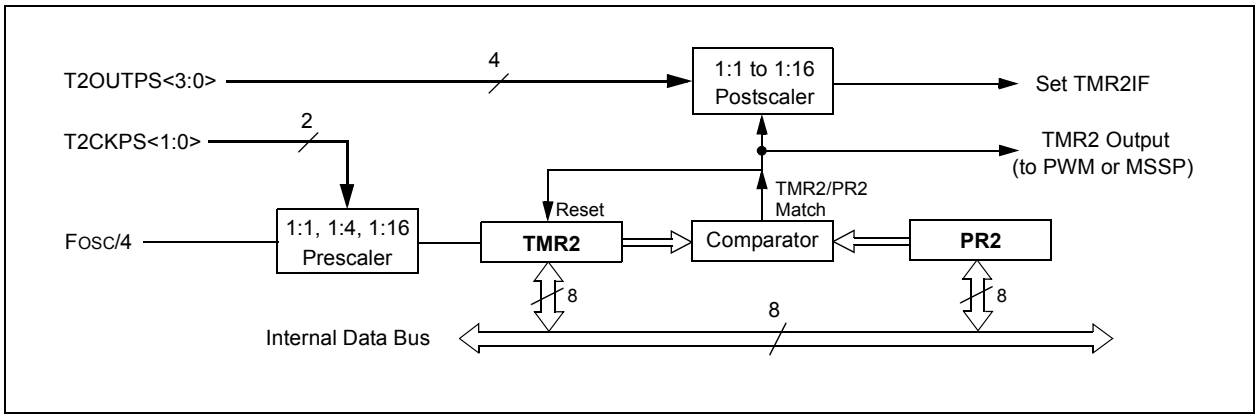



**TABLE 14-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| TMR2 | Timer2 Register | | | | | | | | 56 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | 56 |
| PR2 | Timer2 Period Register | | | | | | | | 56 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

**Note 1:** These bits are unimplemented on PIC18F2X80 devices; always maintain these bits clear.


© 2009 Microchip Technology Inc.

## 15.0 TIMER3 MODULE

The Timer3 module timer/counter incorporates these features:

- Software-selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR3H and TMR3L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Module Reset on CCP Special Event Trigger

A simplified block diagram of the Timer3 module is shown in Figure 15-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 15-2.

The Timer3 module is controlled through the T3CON register (Register 15-1). It also selects the clock source options for the CCP modules (see **Section 16.1.1 "CCP Modules and Timer Resources"** for more information).

**REGISTER 15-1: T3CON: TIMER3 CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| RD16 | T3ECCP1(1) | T3CKPS1 | T3CKPS0 | T3CCP1(1) | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 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 6,3 **T3ECCP1:T3CCP1:** Timer3 and Timer1 to CCP/ECCP Enable bits(1)
1x = Timer3 is the capture/compare clock source for both CCP and ECCP modules
01 = Timer3 is the capture/compare clock source for ECCP;
Timer1 is the capture/compare clock source for CCP
00 = Timer1 is the capture/compare clock source for both CCP and ECCP modules

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 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the device 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

**Note 1:** These bits and the ECCP module are available on PIC18F4X80 devices only.

## 15.1 Timer3 Operation

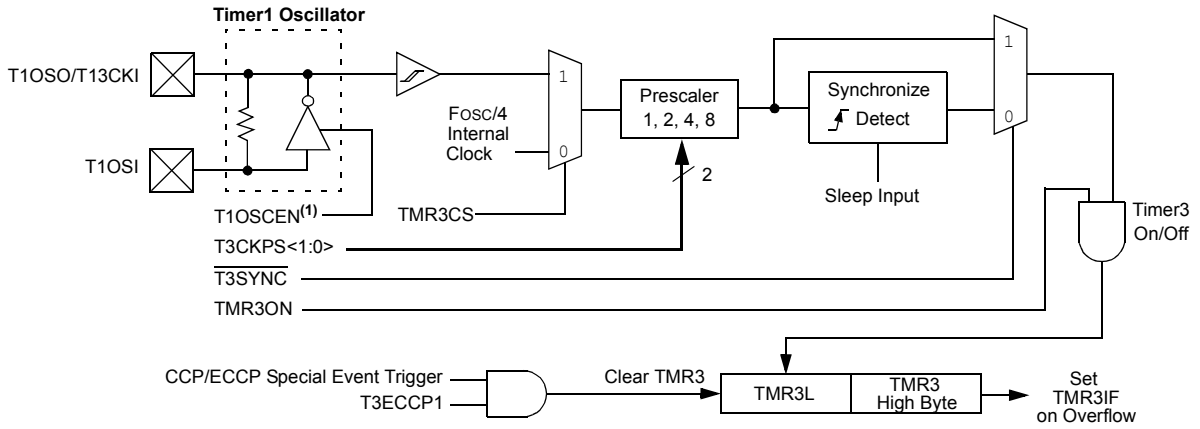
Timer3 can operate in one of three modes:

- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR3CS (T3CON<1>). When TMR3CS is cleared (= 0), Timer3 increments on every internal instruction cycle (Fosc/4). When the bit is set, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator if enabled.

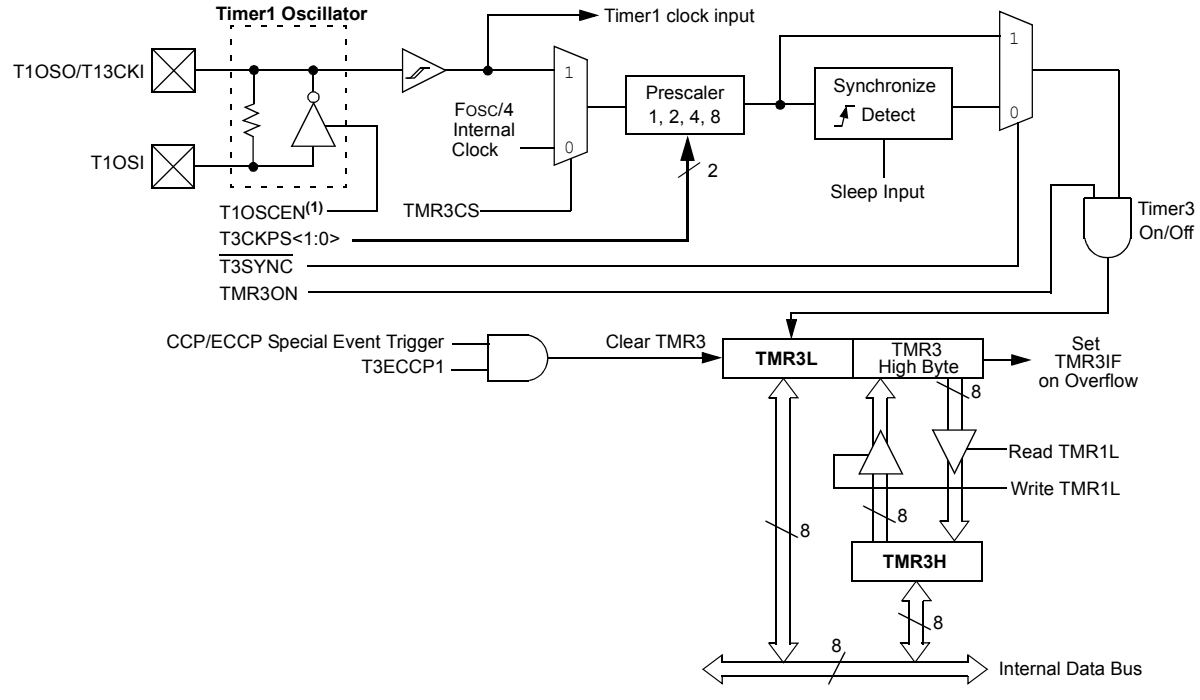
As with Timer1, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs when the Timer1 oscillator is enabled. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

**FIGURE 15-1: TIMER3 BLOCK DIAGRAM**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

**FIGURE 15-2: TIMER3 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.


© 2009 Microchip Technology Inc.

## 15.2 Timer3 16-Bit Read/Write Mode

Timer3 can be configured for 16-bit reads and writes (see Figure 15-2). When the RD16 control bit (T3CON<7>) is set, the address for TMR3H is mapped to a buffer register for the high byte of Timer3. A read from TMR3L will load the contents of the high byte of Timer3 into the Timer3 High Byte Buffer register. 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, has become invalid due to a rollover between reads.

A write to the high byte of Timer3 must also take place through the TMR3H Buffer register. The Timer3 high byte is updated with the contents of TMR3H when a write occurs to TMR3L. This allows a user to write all 16 bits to both the high and low bytes of Timer3 at once.

The high byte of Timer3 is not directly readable or writable in this mode. All reads and writes must take place through the Timer3 High Byte Buffer register.

Writes to TMR3H do not clear the Timer3 prescaler. The prescaler is only cleared on writes to TMR3L.

## 15.3 Using the Timer1 Oscillator as the Timer3 Clock Source

The Timer1 internal oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN (T1CON<3>) bit. To use it as the Timer3 clock source, the TMR3CS bit must also be set. As previously noted, this also configures Timer3 to increment on every rising edge of the oscillator source.

The Timer1 oscillator is described in **Section 13.0 "Timer1 Module"**.

## 15.4 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and overflows to 0000h. The Timer3 interrupt, if enabled, is generated on overflow and is latched in the interrupt flag bit, TMR3IF (PIR2<1>). This interrupt can be enabled or disabled by setting or clearing the Timer3 Interrupt Enable bit, TMR3IE (PIE2<1>).

## 15.5 Resetting Timer3 Using the CCP Special Event Trigger

If the ECCP1 module is configured to generate a special event trigger in Compare mode (ECCP1M<3:0> = 1011), this signal will reset Timer3. It will also start an A/D conversion if the A/D module is enabled (see **Section 16.3.4 "Special Event Trigger"** for more information.).

The module must be configured as either a timer or synchronous counter to take advantage of this feature. When used this way, the ECCPR2H:ECCPR2L register pair effectively becomes a period register for Timer3.

If Timer3 is running in Asynchronous Counter mode, the Reset operation may not work.

In the event that a write to Timer3 coincides with a Special Event Trigger from a CCP module, the write will take precedence.

| **Note:** | The special event triggers from the ECCP1 module will not set the TMR3IF interrupt flag bit (PIR1<0>). |
|---|---|

**TABLE 15-1: 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 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR2 | OSCFIF | CMIF(2) | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF(2) | 58 |
| PIE2 | OSCFIE | CMIE(2) | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE(2) | 58 |
| IPR2 | OSCFIP | CMIP(2) | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP(2) | 57 |
| TMR3L | Timer3 Register Low Byte | | | | | | | | 57 |
| TMR3H | Timer3 Register High Byte | | | | | | | | 57 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 56 |
| T3CON | RD16 | T3ECCP1(1) | T3CKPS1 | T3CKPS0 | T3CCP1(1) | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer3 module.

**Note 1:** These bits are available in PIC18F4X80 devices only.

**2:** These bits are available in PIC18F4X80 devices and reserved in PIC18F2X80 devices.

**NOTES:**


© 2009 Microchip Technology Inc.

## 16.0 CAPTURE/COMPARE/PWM (CCP) MODULES

PIC18F2480/2580 devices have one CCP module. PIC18F4480/4580 devices have two CCP (Capture/Compare/PWM) modules. CCP1, discussed in this chapter, implements standard Capture, Compare and Pulse-Width Modulation (PWM) modes.

ECCP1 implements an Enhanced PWM mode. The ECCP implementation is discussed in **Section 17.0 "Enhanced Capture/Compare/PWM (ECCP) Module"**.

The CCP1 module contains a 16-bit register which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register. For the sake of clarity, all CCP module operation in the following sections is described with respect to CCP1, but is equally applicable to ECCP1.

Capture and Compare operations described in this chapter apply to all standard and Enhanced CCP modules. The operations of PWM mode, described in **Section 16.4 "PWM Mode"**, apply to ECCP1 only.

**REGISTER 16-1: CCP1CON: CAPTURE/COMPARE/PWM CONTROL REGISTER**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **DC1B<1:0>**: CCP1 Module PWM Duty Cycle bit 1 and bit 0

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs (DC1B<9:2>) of the duty cycle are found in CCPR1L.

bit 3-0 **CCP1M<3:0>**: CCP1 Module Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCP1 module)
0001 = Reserved
0010 = Compare mode; toggle output on match (CCP1IF bit is set)
0011 = Reserved
0100 = Capture mode; every falling edge or CAN message received (time-stamp)(1)
0101 = Capture mode; every rising edge or CAN message received (time-stamp)(1)
0110 = Capture mode; every 4th rising edge or every 4th CAN message received (time-stamp)(1)
0111 = Capture mode; every 16th rising edge or every 16th CAN message received (time-stamp)(1)
1000 = Compare mode; initialize CCP1 pin low; on compare match, force CCP1 pin high (CCPIF bit is set)
1001 = Compare mode; initialize CCP pin high; on compare match, force CCP1 pin low (CCPIF bit is set)
1010 = Compare mode; generate software interrupt on compare match (CCP1IF bit is set, CCP1 pin reflects I/O state)
1011 = Compare mode; trigger special event; reset timer (TMR1 or TMR3, CCP1IF bit is set)
11xx = PWM mode

**Note 1:** Selected by CANCAP (CIOCON<4>) bit; overrides the CCP1 input pin source.

## 16.1 CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (generically, CCPxCON) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte). All registers are both readable and writable.

### 16.1.1 CCP MODULES AND TIMER RESOURCES

The CCP modules utilize Timers 1, 2 or 3, depending on the mode selected. Timer1 and Timer3 are available to modules in Capture or Compare modes, while Timer2 is available for modules in PWM mode.

**TABLE 16-1: CCP MODE – TIMER RESOURCE**

| CCP/ECCP Mode | Timer Resource |
|---|---|
| Capture | Timer1 or Timer3 |
| Compare | Timer1 or Timer3 |
| PWM | Timer2 |

The assignment of a particular timer to a module is determined by the Timer to CCP enable bits in the T3CON register (Register 15-1). Both modules may be active at any given time and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time. The interactions between the two modules are summarized in Figure 16-1 and Figure 16-2.

**TABLE 16-2: INTERACTIONS BETWEEN CCP1 AND ECCP1 FOR TIMER RESOURCES**

| CCP1 Mode | ECCP1 Mode | Interaction |
|---|---|---|
| Capture | Capture | Each module can use TMR1 or TMR3 as the time base. Time base can be different for each CCP. |
| Capture | Compare | CCP1 can be configured for the Special Event Trigger to reset TMR1 or TMR3 (depending upon which time base is used). Automatic A/D conversions on trigger event can also be done. Operation of CCP1 could be affected if it is using the same timer as a time base. |
| Compare | Capture | CCP1 can be configured for the Special Event Trigger to reset TMR1 or TMR3 (depending upon which time base is used). Operation of CCP1 could be affected if it is using the same timer as a time base. |
| Compare | Compare | Either module can be configured for the Special Event Trigger to reset the time base. Automatic A/D conversions on ECCP1 trigger event can be done. Conflicts may occur if both modules are using the same time base. |
| Capture | PWM(1) | None |
| Compare | PWM(1) | None |
| PWM(1) | Capture | None |
| PWM(1) | Compare | None |
| PWM(1) | PWM | Both PWMs will have the same frequency and update rate (TMR2 interrupt). |

**Note 1:** Includes standard and Enhanced PWM operation.


© 2009 Microchip Technology Inc.

## 16.2 Capture Mode

In Capture mode, the CCPR1H:CCPR1L register pair captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on the CCP1 pin (RB3 or RC1, depending on device configuration). 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 the mode select bits, CCP1M<3:0> (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit, CCP1IF (PIR2<1>), 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.

### 16.2.1 CCP1/ECCP1 PIN CONFIGURATION

In Capture mode, the appropriate CCP1/ECCP1 pin should be configured as an input by setting the corresponding TRIS direction bit.

| | |
|---|---|
| **Note:** | If RC2/CCP1 or RD4/PSP4/ECCP1/P1A is configured as an output, a write to the port can cause a capture condition. |

### 16.2.2 TIMER1/TIMER3 MODE SELECTION

The timers that are to be used with the capture feature (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 each CCP module is selected in the T3CON register (see **Section 16.1.1 "CCP Modules and Timer Resources"**).

### 16.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE interrupt enable bit clear to avoid false interrupts. The interrupt flag bit, CCPxIF, should also be cleared following any such change in operating mode.

### 16.2.4 CCP PRESCALER

There are four prescaler settings in Capture mode; they are specified as part of the operating mode selected by the mode select bits (CCP1M<3:0>). 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 16-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the "false" interrupt.

### 16.2.5 CAN MESSAGE TIME-STAMP

The CAN capture event occurs when a message is received in any of the receive buffers. When configured, the CAN module provides the trigger to the CCP1 module to cause a capture event. This feature is provided to "time-stamp" the received CAN messages.

This feature is enabled by setting the CANCAP bit of the CAN I/O Control register (CIOCON<4>). The message receive signal from the CAN module then takes the place of the events on RC2/CCP1.

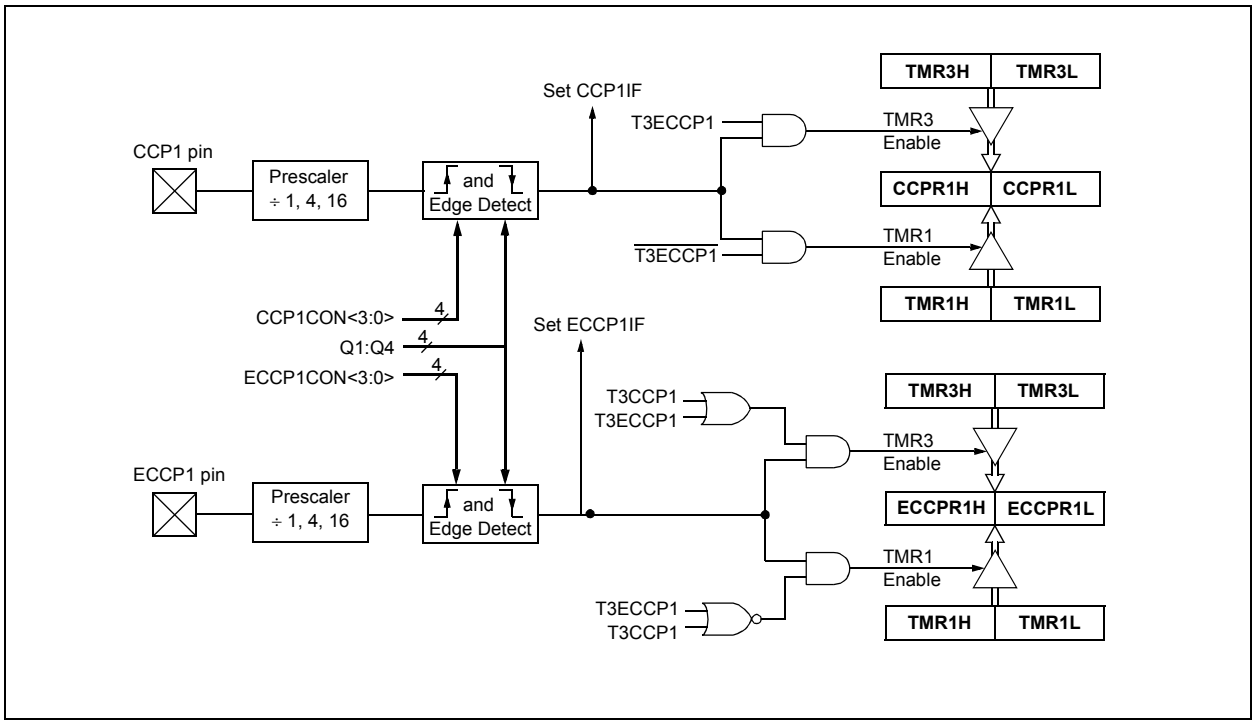
If this feature is selected, then four different capture options for CCP1M<3:0> are available:

- 0100 – every time a CAN message is received
- 0101 – every time a CAN message is received
- 0110 – every 4th time a CAN message is received
- 0111 – capture mode, every 16th time a CAN message is received

**EXAMPLE 16-1: 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 16-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**





© 2009 Microchip Technology Inc.

## 16.3 Compare Mode

In Compare mode, the 16-bit CCPR1 register value is constantly compared against either the TMR1 or TMR3 register pair value. When a match occurs, the CCP1 pin can be:

- driven high
- driven low
- toggled (high-to-low or low-to-high)
- remain unchanged (that is, reflects the state of the I/O latch)

The action on the pin is based on the value of the mode select bits (ECCP1M<3:0>). At the same time, the interrupt flag bit, ECCP1IF, is set.

### 16.3.1 CCP PIN CONFIGURATION

The user must configure the CCP1 pin as an output by clearing the appropriate TRIS bit.

| **Note:** | Clearing the CCP1CON register will force the RC2 compare output latch (depending on device configuration) to the default low level. This is not the PORTC I/O data latch. |
|---|---|

### 16.3.2 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.

### 16.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCP1M<3:0> = 1010), the CCP1 pin is not affected. Only a CCP interrupt is generated, if enabled, and the CCP1IE bit is set.

### 16.3.4 SPECIAL EVENT TRIGGER

Both CCP modules are equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode (CCP1M<3:0> = 1011).

For either CCP module, the Special Event Trigger resets the Timer register pair for whichever timer resource is currently assigned as the module's time base. This allows the CCPR1 registers to serve as a programmable period register for either timer.

**FIGURE 16-2: COMPARE MODE OPERATION BLOCK DIAGRAM**

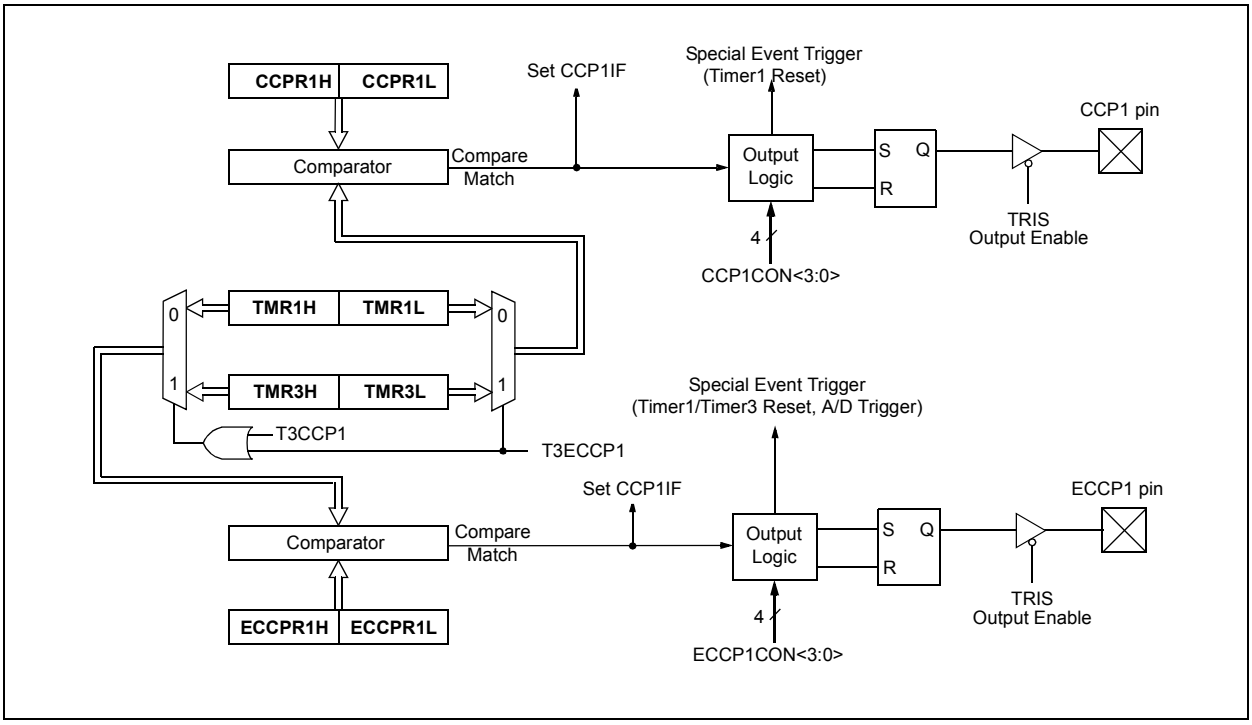


**TABLE 16-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| RCON | IPEN | SBOREN(3) | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 56 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR2 | OSCFIP | CMIP(2) | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP(2) | 58 |
| PIR2 | OSCFIF | CMIF(2) | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF(2) | 58 |
| PIE2 | OSCFIE | CMIE(2) | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE(2) | 57 |
| TRISB | PORTB Data Direction Register | | | | | | | | 58 |
| TRISC | PORTC Data Direction Register | | | | | | | | 58 |
| TMR1L | Timer1 Register Low Byte | | | | | | | | 56 |
| TMR1H | Timer1 Register High Byte | | | | | | | | 56 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{T1SYNC}$ | TMR1CS | TMR1ON | 56 |
| TMR3H | Timer3 Register High Byte | | | | | | | | 57 |
| TMR3L | Timer3 Register Low Byte | | | | | | | | 57 |
| T3CON | RD16 | T3ECCP1(1) | T3CKPS1 | T3CKPS0 | T3CCP1(1) | $\overline{T3SYNC}$ | TMR3CS | TMR3ON | 57 |
| CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | 57 |
| CCPR1H | Capture/Compare/PWM Register 1 High Byte | | | | | | | | 57 |
| CCP1CON | — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 57 |
| ECCPR1L(1) | Enhanced Capture/Compare/PWM Register 1 Low Byte | | | | | | | | 57 |
| ECCPR1H(1) | Enhanced Capture/Compare/PWM Register 1 High Byte | | | | | | | | 57 |
| ECCP1CON(1) | EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by capture, compare, Timer1 or Timer3.

**Note 1:** These bits or registers are available on PIC18F4X80 devices only.

**2:** These bits are available on PIC18F4X80 devices and reserved on PIC18F2X80 devices.

**3:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.


© 2009 Microchip Technology Inc.

## 16.4 PWM Mode

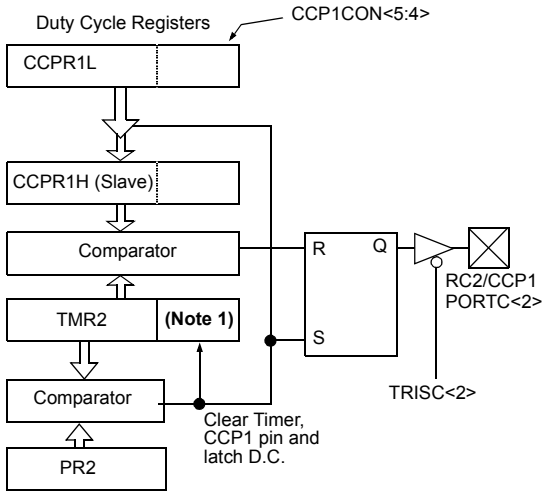
In Pulse-Width Modulation (PWM) mode, the CCP1 pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with a PORTB or PORTC data latch, the appropriate TRIS bit must be cleared to make the CCP1 pin an output.

| | |
|---|---|
| **Note:** | Clearing the CCP1CON register will force the RC2 output latch (depending on device configuration) to the default low level. This is not the PORTC I/O data latch. |

Figure 16-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see **Section 16.4.4 "Setup for PWM Operation"**.

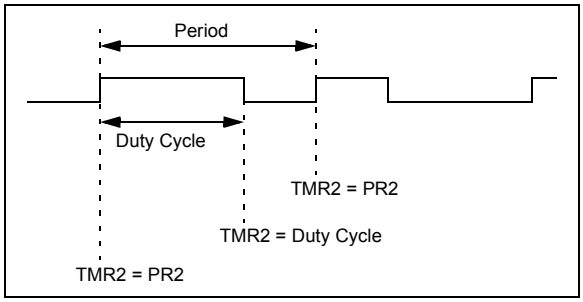
**FIGURE 16-3: SIMPLIFIED PWM BLOCK DIAGRAM**




**Note 1:** The 8-bit TMR2 value is concatenated with 2-bit internal Q clock, or 2 bits of the prescaler, to create the 10-bit time base.

A PWM output (Figure 16-4) has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

**FIGURE 16-4: PWM OUTPUT**




### 16.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 (PR4) register. The PWM period can be calculated using the following formula.

**EQUATION 16-1:**

$$\text{PWM Period} = (PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[PWM period].

When TMR1 (TMR3) is equal to PR2 (PR2), the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

| | |
|---|---|
| **Note:** | The Timer2 postscalers (see **Section 14.0 "Timer2 Module"**) are 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. |

### 16.4.2 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> bits contain the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time.

**EQUATION 16-2:**

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after 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 2 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 16-3:**

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{FOSC}{FPWM}\right)}{\log(2)} \text{ bits}$$

| **Note:** | If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared. |
|---|---|

**TABLE 16-4: 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 |

### 16.4.3 PWM AUTO-SHUTDOWN (ECCP1 ONLY)

The PWM auto-shutdown features of the Enhanced CCP module are available to ECCP1 in PIC18F4480/4580 (40/44-pin) devices. The operation of this feature is discussed in detail in **Section 17.4.7 "Enhanced PWM Auto-Shutdown"**.

Auto-shutdown features are not available for CCP1.

### 16.4.4 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.
3. Make the CCP1 pin an output by clearing the appropriate TRIS bit.
4. Set the TMR2 prescale value, then enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.


© 2009 Microchip Technology Inc.

**TABLE 16-5: REGISTERS ASSOCIATED WITH PWM AND TIMER2**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| RCON | IPEN | SBOREN(2) | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 56 |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| TRISB | PORTB Data Direction Register | | | | | | | | 58 |
| TRISC | PORTC Data Direction Register | | | | | | | | 58 |
| TMR2 | Timer2 Register | | | | | | | | 56 |
| PR2 | Timer2 Period Register | | | | | | | | 56 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | 56 |
| CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | 57 |
| CCPR1H | Capture/Compare/PWM Register 1 High Byte | | | | | | | | 57 |
| CCP1CON | — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 57 |
| ECCPR1L(1) | Enhanced Capture/Compare/PWM Register 1 Low Byte | | | | | | | | 57 |
| ECCPR1H(1) | Enhanced Capture/Compare/PWM Register 1 High Byte | | | | | | | | 57 |
| ECCP1CON(1) | EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PWM or Timer2.

**Note 1:** These registers are unimplemented on PIC18F2X80 devices.

**2:** The SBOREN bit is only available when CONFIG2L<1:0> = 01; otherwise, it is disabled and reads as '0'. See **Section 5.4 "Brown-out Reset (BOR)"**.

**NOTES:**


© 2009 Microchip Technology Inc.

## 17.0 ENHANCED CAPTURE/COMPARE/PWM (ECCP) MODULE

| **Note:** | The ECCP1 module is implemented only in PIC18F4X80 (40/44-pin) devices. |
|---|---|

In PIC18F4480/4580 devices, ECCP1 is implemented as a standard CCP module with Enhanced PWM capabilities. These include the provision for 2 or 4 output channels, user-selectable polarity, dead-band control and automatic shutdown and restart. The Enhanced features are discussed in detail in **Section 17.4 "Enhanced PWM Mode"**. Capture, Compare and single-output PWM functions of the ECCP module are the same as described for the standard CCP module.

The control register for the Enhanced CCP module is shown in Register 17-1. It differs from the CCP1CON register in PIC18F2480/2580 devices in that the two Most Significant bits are implemented to control PWM functionality.

**REGISTER 17-1: ECCP1CON REGISTER (ECCP1 MODULE, PIC18F4480/4580 DEVICES)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **EPWM1M<1:0>:** Enhanced PWM Output Configuration bits

If ECCP1M<3:2> = 00, 01, 10:
xx = P1A assigned as Capture/Compare input/output; P1B, P1C, P1D assigned as port pins

If ECCP1M<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 **EDC1B<1:0>**: ECCP1 Module PWM Duty Cycle bit 1 and bit 0

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in ECCPR1L.

bit 3-0 **ECCP1M<3:0>**: Enhanced CCP1 Mode Select bits

0000 = Capture/Compare/PWM off (resets ECCP1 module)
0001 = Reserved
0010 = Compare mode; toggle output on match
0011 = 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; initialize ECCP1 pin low; set output on compare match (set ECCP1IF)
1001 = Compare mode; initialize ECCP1 pin high; clear output on compare match (set ECCP1IF)
1010 = Compare mode; generate software interrupt only; ECCP1 pin reverts to I/O state
1011 = Compare mode; trigger special event (ECCP1 resets TMR1 or TMR3, sets ECCP1IF bit and starts the A/D conversion on ECCP1 match)
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

In addition to the expanded range of modes available through the CCP1CON register, the ECCP module has two additional registers associated with Enhanced PWM operation and auto-shutdown features. They are:

- ECCP1DEL (Dead-Band Delay)
- ECCP1AS (Auto-Shutdown Control)

## 17.1 ECCP Outputs and Configuration

The Enhanced CCP module may have up to four PWM outputs, depending on the selected operating mode. These outputs, designated P1A through P1D, are multiplexed with I/O pins on PORTC and PORTD. The outputs that are active depend on the CCP operating mode selected. The pin assignments are summarized in Table 17-1.

To configure the I/O pins as PWM outputs, the proper PWM mode must be selected by setting the EPWM1M<1:0> and CCP1M<3:0> bits. The appropriate TRISC and TRISD direction bits for the port pins must also be set as outputs.

### 17.1.1 ECCP MODULES AND TIMER RESOURCES

Like the standard CCP modules, the ECCP module can utilize Timers 1, 2 or 3, depending on the mode selected. Timer1 and Timer3 are available for modules in Capture or Compare modes, while Timer2 is available for modules in PWM mode. Interactions between the standard and Enhanced CCP modules are identical to those described for standard CCP modules. Additional details on timer resources are provided in **Section 16.1.1 "CCP Modules and Timer Resources"**.

## 17.2 Capture and Compare Modes

Except for the operation of the Special Event Trigger discussed below, the Capture and Compare modes of the ECCP1 module are identical in operation to that of CCP1. These are discussed in detail in **Section 16.2 "Capture Mode"** and **Section 16.3 "Compare Mode"**.

### 17.2.1 SPECIAL EVENT TRIGGER

The Special Event Trigger output of ECCP1 resets the TMR1 or TMR3 register pair, depending on which timer resource is currently selected. This allows the ECCP1 register to effectively be a 16-bit programmable period register for Timer1 or Timer3. The Special Event Trigger for ECCP1 can also start an A/D conversion. In order to start the conversion, the A/D Converter must be previously enabled.

## 17.3 Standard PWM Mode

When configured in Single Output mode, the ECCP module functions identically to the standard CCP module in PWM mode, as described in **Section 16.4 "PWM Mode"**. This is also sometimes referred to as "Compatible CCP" mode, as in Table 17-1.

| | |
|---|---|
| **Note:** | When setting up single output PWM operations, users are free to use either of the processes described in **Section 16.4.4 "Setup for PWM Operation"** or **Section 17.4.9 "Setup for PWM Operation"**. The latter is more generic, but will work for either single or multi-output PWM. |

**TABLE 17-1: PIN ASSIGNMENTS FOR VARIOUS ECCP MODES**

| ECCP Mode | CCP1CON Configuration | RD4 | RD5 | RD6 | RD7 |
|---|---|---|---|---|---|
| **All PIC18F4480/4580 Devices:** | | | | | |
| Compatible CCP | 00xx 11xx | CCP1 | RD5/PSP5 | RD6/PSP6 | RD7/PSP7 |
| Dual PWM | 10xx 11xx | P1A | P1B | RD6/PSP6 | RD7/PSP7 |
| Quad PWM | x1xx 11xx | P1A | P1B | P1C | P1D |

**Legend:** x = Don't care. Shaded cells indicate pin assignments not used by ECCP1 in a given mode.


© 2009 Microchip Technology Inc.

## 17.4 Enhanced PWM Mode

The Enhanced PWM mode provides additional PWM output options for a broader range of control applications. The module is a backward 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 EPWM1M<1:0> and CCP1M<3:0> bits of the ECCP1CON register.

Figure 17-1 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 ECCP PWM Dead-Band 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.

### 17.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following equation.

**EQUATION 17-1:**

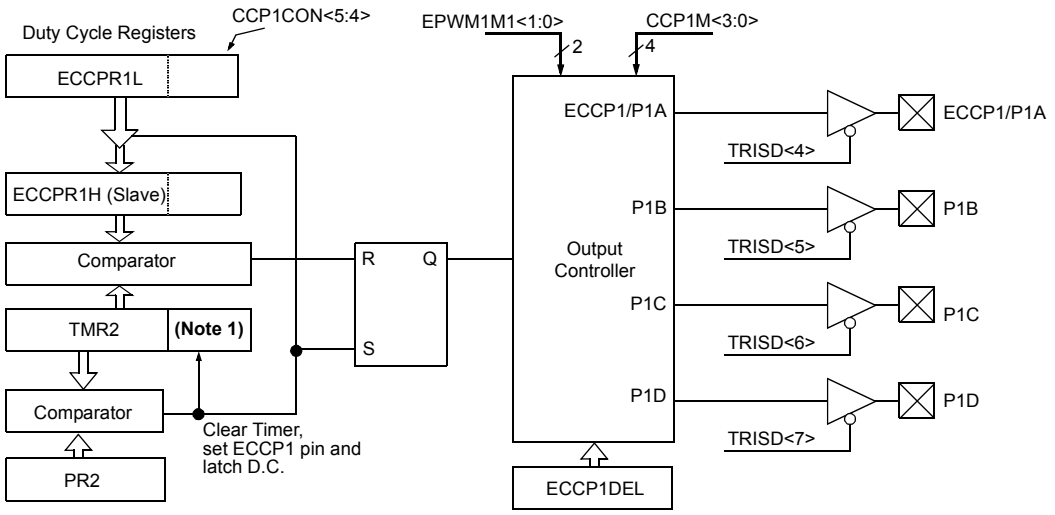
$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

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 ECCP1 pin is set (if PWM duty cycle = 0%, the ECCP1 pin will not be set)
- The PWM duty cycle is copied from ECCPR1L into ECCPR1H

**Note:** The Timer2 postscaler (see **Section 14.0 "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.

**FIGURE 17-1: SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODULE**




**Note:** The 8-bit TMR2 register is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler, to create the 10-bit time base.

### 17.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the ECCPR1L register and to the ECCP1CON<5:4> bits. Up to 10-bit resolution is available. The ECCPR1L contains the eight MSbs and the ECCP1CON<5:4> bits contain the two LSbs. This 10-bit value is represented by ECCPR1L:ECCP1CON<5:4>. The PWM duty cycle is calculated by the following equation.

**EQUATION 17-2:**

$$\text{PWM Duty Cycle} = (\text{ECCPR1L:ECCP1CON<5:4>} \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

ECCPR1L and ECCP1CON<5:4> can be written to at any time, but the duty cycle value is not copied into ECCPR1H until a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, ECCPR1H is a read-only register.

The ECCPR1H 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 ECCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or two bits of the TMR2 prescaler, the ECCP1 pin is cleared. The maximum PWM resolution (bits) for a given PWM frequency is given by the following equation.

**EQUATION 17-3:**

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

**Note:** If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

### 17.4.3 PWM OUTPUT CONFIGURATIONS

The EPWM1M<1:0> bits in the ECCP1CON 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 17.4 "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 17-2.

**TABLE 17-2: 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 |


© 2009 Microchip Technology Inc.

**FIGURE 17-2: PWM OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)**

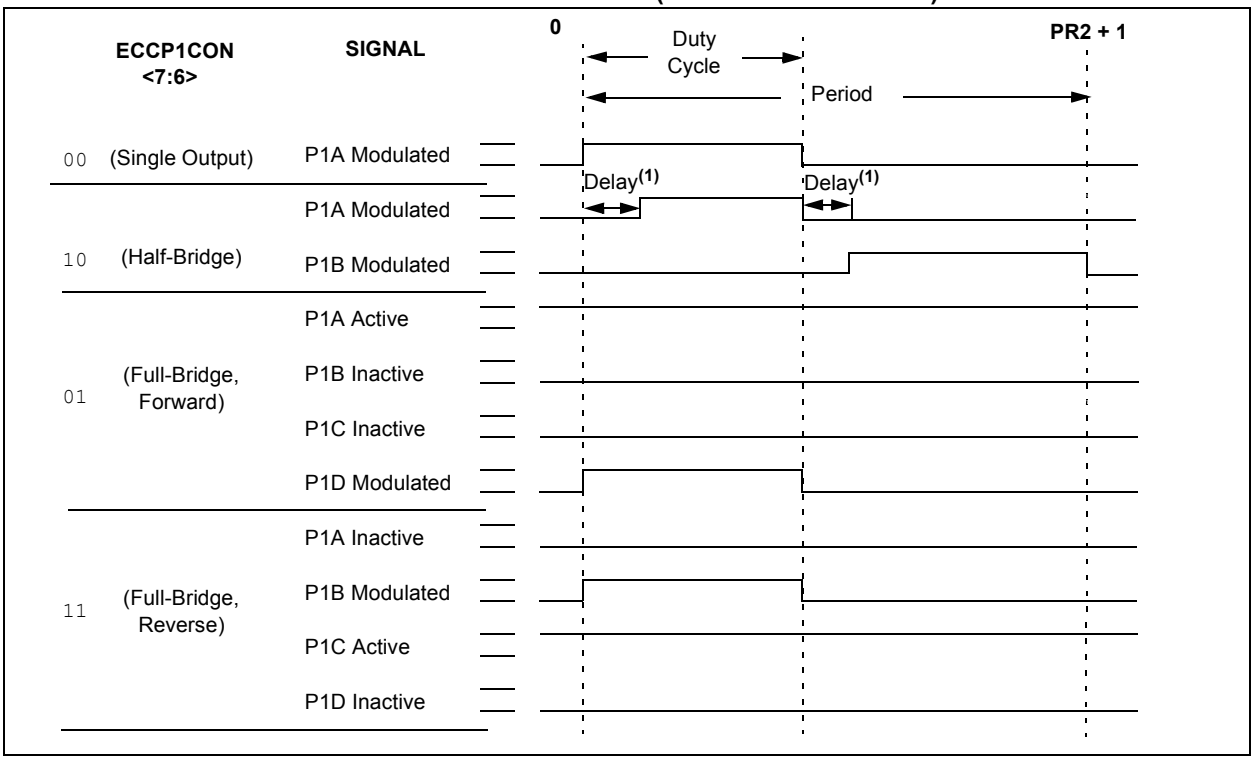
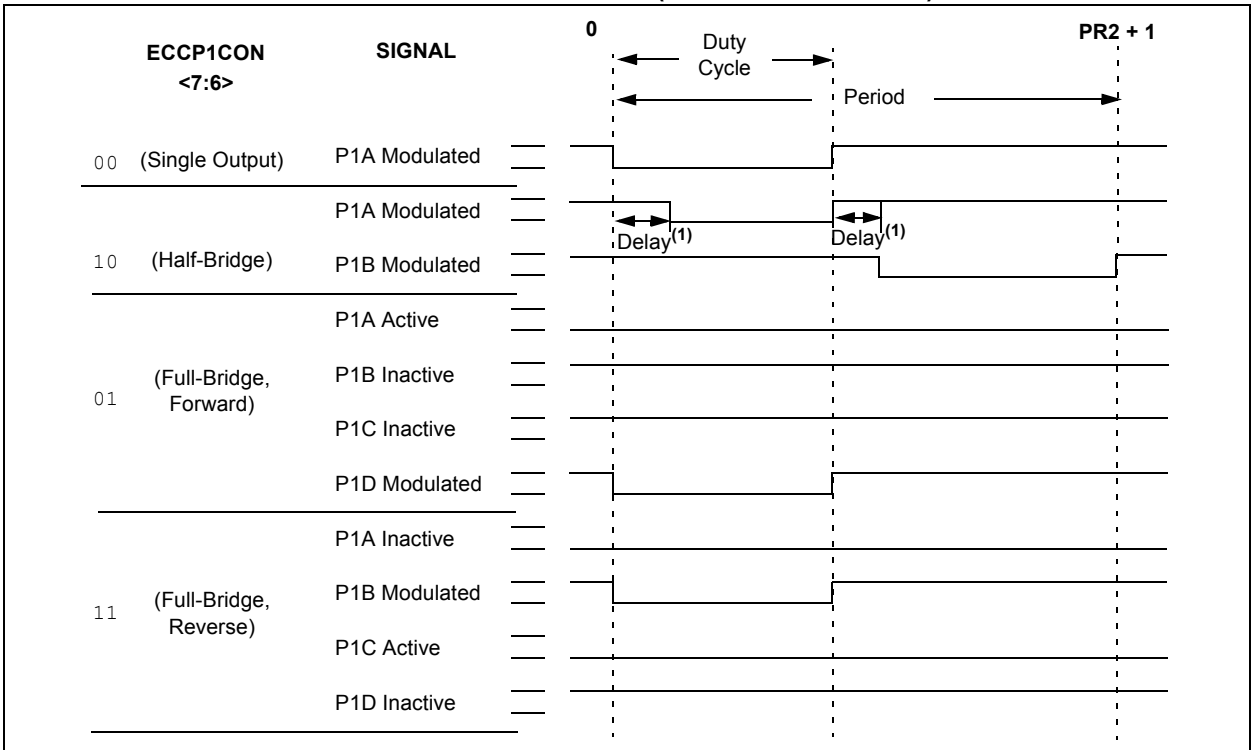



**FIGURE 17-3: PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)**




**Relationships:**

- Period = 4 * TOSC * (PR2 + 1) * (TMR2 Prescale Value)
- Duty Cycle = TOSC * (ECCPR1L<7:0>:ECCP1CON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * TOSC * (ECCP1DEL<6:0>)

**Note 1:** Dead-band delay is programmed using the ECCP1DEL register (**Section 17.4.6 "Programmable Dead-Band Delay"**).

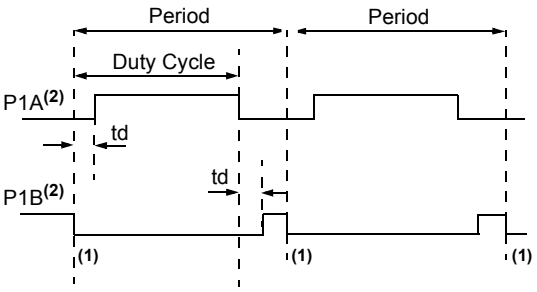
### 17.4.4 HALF-BRIDGE MODE

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 P1A pin, while the complementary PWM output signal is output on the P1B pin (Figure 17-4). This mode can be used for half-bridge applications, as shown in Figure 17-5, 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, PDC<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 17.4.6 "Programmable Dead-Band Delay"** for more details of the dead-band delay operations.

Since the P1A and P1B outputs are multiplexed with the PORTD<4> and PORTD<5> data latches, the TRISD<4> and TRISD<5> bits must be cleared to configure P1A and P1B as outputs.

**FIGURE 17-4: HALF-BRIDGE PWM OUTPUT**

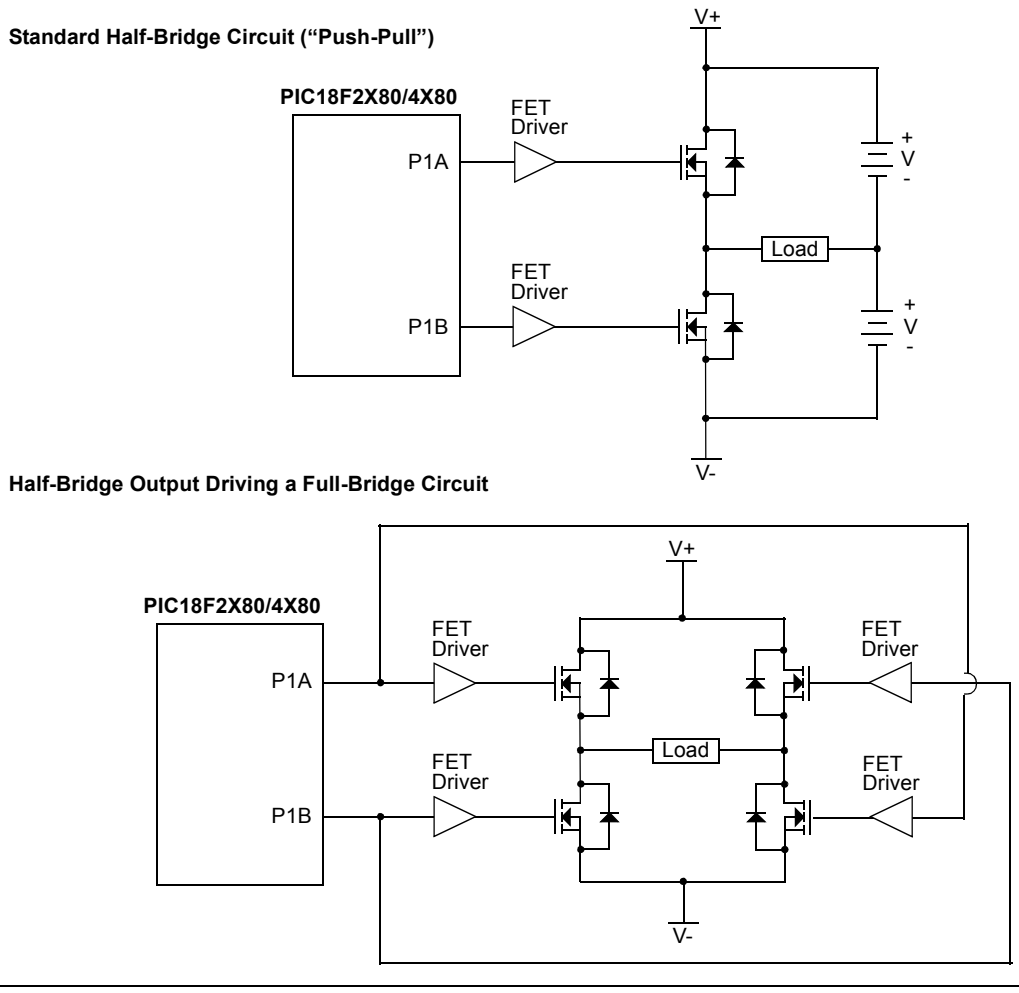



td = Dead-Band Delay

**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

**2:** Output signals are shown as active-high.

**FIGURE 17-5: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS**





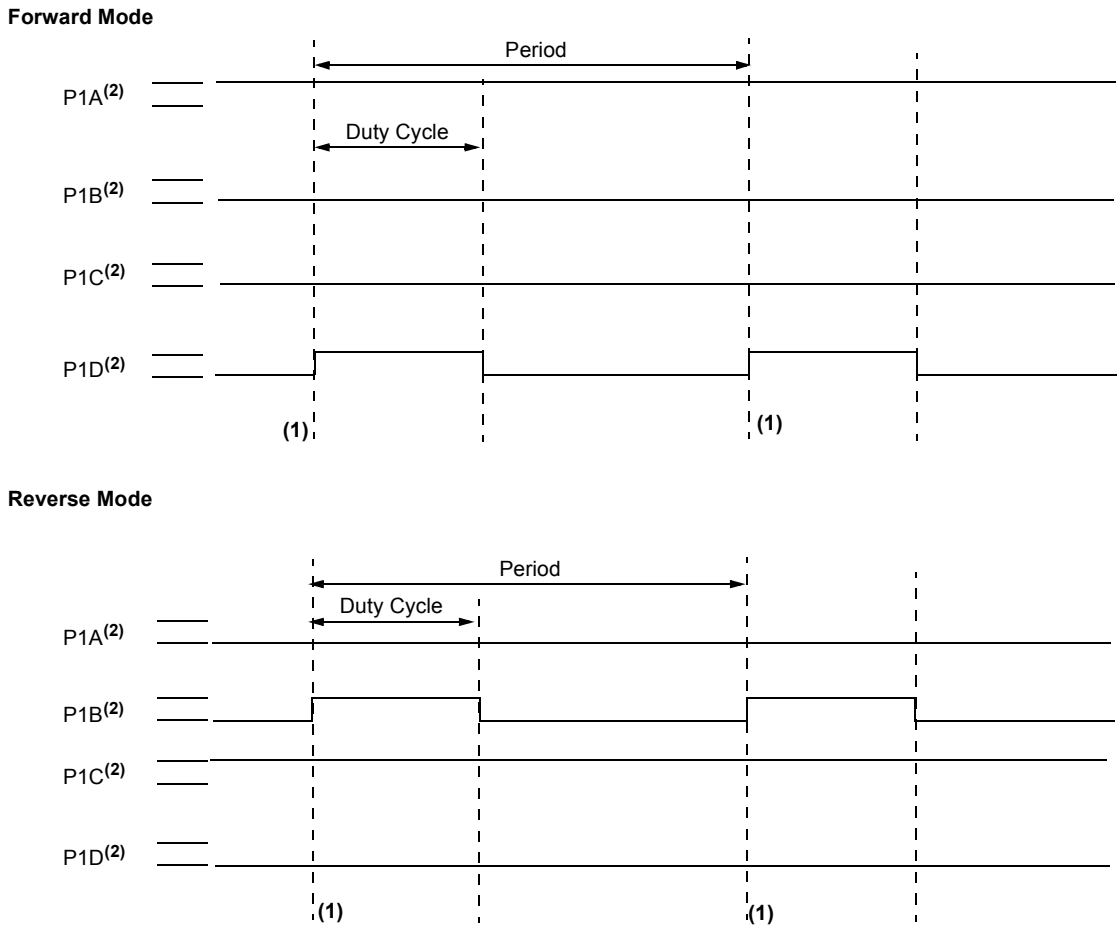
© 2009 Microchip Technology Inc.

### 17.4.5 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 P1A is continuously active and pin P1D is modulated. In the Reverse mode, pin P1C is continuously active and pin P1B is modulated. These are illustrated in Figure 17-6.

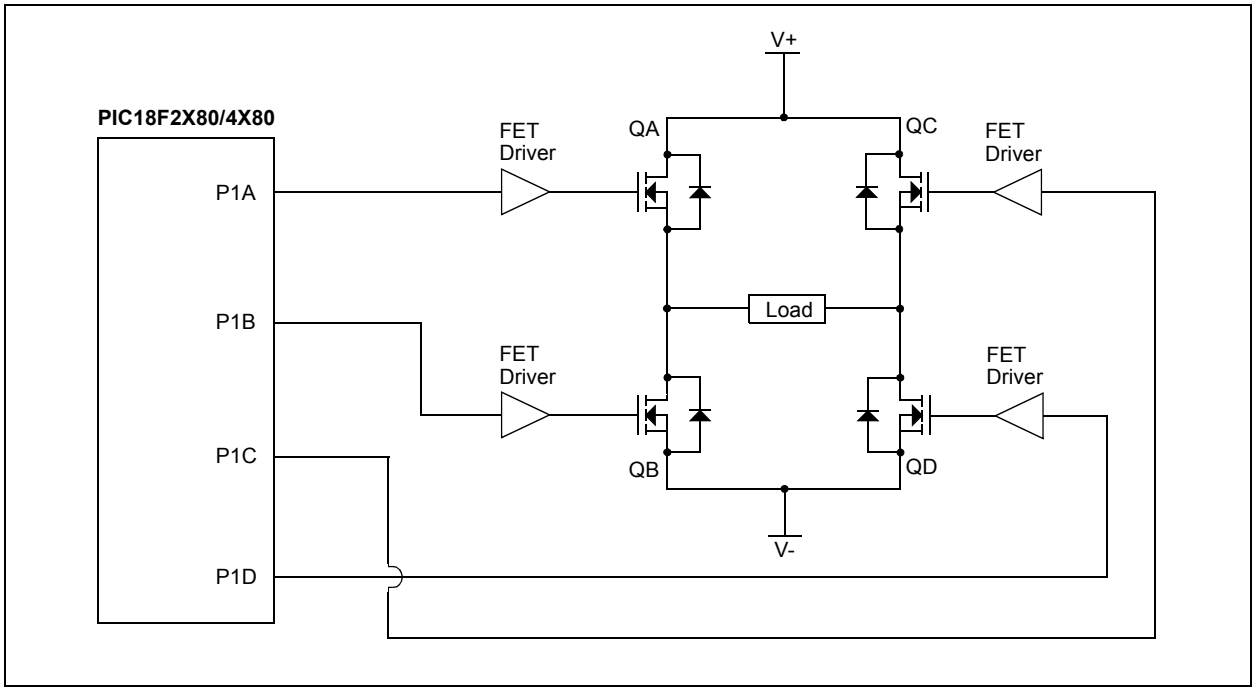
P1A, P1B, P1C and P1D outputs are multiplexed with the PORTD<4>, PORTD<5>, PORTD<6> and PORTD<7> data latches. The TRISD<4>, TRISD<5>, TRISD<6> and TRISD<7> bits must be cleared to make the P1A, P1B, P1C and P1D pins outputs.

**FIGURE 17-6: FULL-BRIDGE PWM OUTPUT**




**Note 1:** At this time, the TMR2 register is equal to the PR2 register.
**2:** Output signal is shown as active-high.

**FIGURE 17-7: EXAMPLE OF FULL-BRIDGE OUTPUT APPLICATION**




### 17.4.5.1 Direction Change in Full-Bridge Output Mode

In the Full-Bridge Output mode, the EPWM1M1 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 bits (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 17-8.

Note that in the Full-Bridge Output mode, the CCP1 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 17-9 shows an example where the PWM direction changes from forward to reverse at a near 100% duty cycle. At time t1, the outputs, 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 17-7), for the duration of ‘t’. The same phenomenon will occur to power devices, QA and QB, for PWM direction change from reverse to forward.

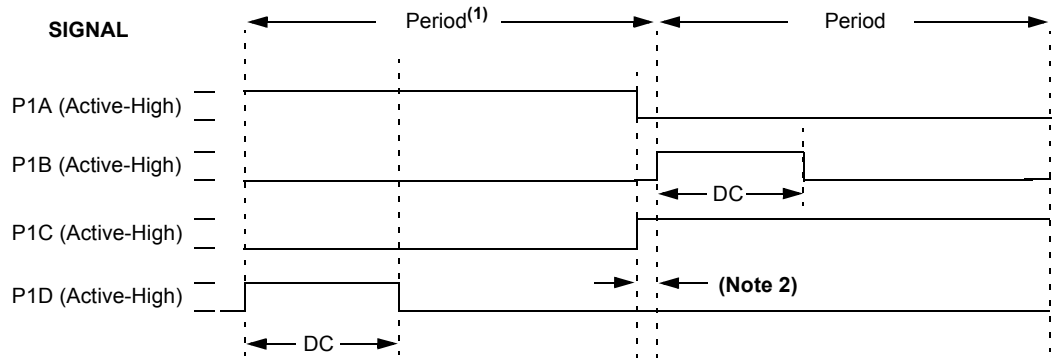
If changing PWM direction at high duty cycle is required for an application, one of the following requirements must be met:

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.


© 2009 Microchip Technology Inc.

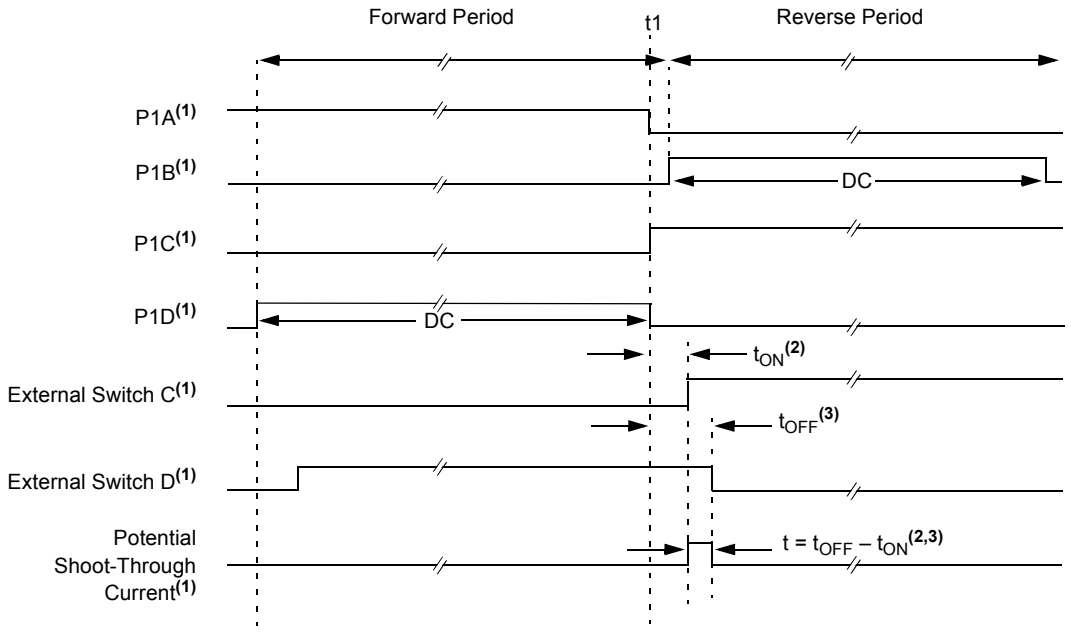
**FIGURE 17-8: PWM DIRECTION CHANGE**




**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 signals switch before the end of the current PWM cycle at intervals of 4 TOSC, 16 TOSC or 64 TOSC, depending on the Timer2 prescaler value. The modulated P1B and P1D signals are inactive at this time.

**FIGURE 17-9: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE**




**Note 1:** All signals are shown as active-high.

**2:** $t_{ON}$ is the turn-on delay of power switch QC and its driver.

**3:** $t_{OFF}$ is the turn-off delay of power switch QD and its driver.

### 17.4.6 PROGRAMMABLE DEAD-BAND DELAY

| | |
|---|---|
| **Note:** | Programmable dead-band delay is not implemented in PIC18F2X80 devices with standard CCP modules. |

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 17-4 for illustration). Bits, PDC<6:0< of the ECCP1DEL register (Register 17-2), set the delay period in terms of microcontroller instruction cycles (TCY or 4 TOSC). These bits are not available on PIC18F2X80 devices, as the standard CCP module does not support half-bridge operation.

### 17.4.7 ENHANCED PWM AUTO-SHUTDOWN

When the CCP1 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 either of the comparator modules, a low level on the RB0/INT0/FLT0/AN10 pin, or any combination of these three sources. The comparators may be used to monitor a voltage input proportional to a current being monitored in the bridge circuit. If the voltage exceeds a threshold, the comparator switches state and triggers a shutdown. Alternatively, a digital signal on the INT0 pin can also trigger a shutdown. 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 ECCPAS<2:0> bits (ECCP1AS<6:4>).

When a shutdown occurs, the output pins are asynchronously placed in their shutdown states, specified by the PSSAC<1:0> and PSS1BD<1:0> 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 (ECCP1AS<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. |

| | |
|---|---|
| **Note:** | If the dead-band delay value is increased after the dead-band time has elapsed, that new value takes effect immediately. This happens even if the PWM pulse is high and can appear to be a glitch. Dead-band values must be changed during the dead-band time or before ECCP is active |


© 2009 Microchip Technology Inc.

**REGISTER 17-2: ECCP1DEL: ECCP PWM DEAD-BAND DELAY REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PRSEN | PDC6(1) | PDC5(1) | PDC4(1) | PDC3(1) | PDC2(1) | PDC1(1) | PDC0(1) |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 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(1)

Delay time, in number of FOSC/4 (4 * TOSC) cycles, between the scheduled and actual time for a PWM signal to transition to active.

**Note 1:** Reserved on PIC18F2X80 devices; maintain these bits clear.

**REGISTER 17-3: ECCP1AS: ECCP AUTO-SHUTDOWN CONTROL REGISTER(1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1(1) | PSSBD0(1) |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **ECCPASE:** ECCP Auto-Shutdown Event Status bit
- 1 = A shutdown event has occurred; ECCP outputs are in shutdown state
- 0 = ECCP outputs are operating

bit 6-4 **ECCPAS<2:0>:** ECCP Auto-Shutdown Source Select bits
- 111 = RB0 or Comparator 1 or Comparator 2
- 110 = RB0 or Comparator 2
- 101 = RB0 or Comparator 1
- 100 = RB0
- 011 = Either Comparator 1 or 2
- 010 = Comparator 2 output
- 001 = Comparator 1 output
- 000 = Auto-shutdown is disabled

bit 3-2 **PSSAC<1:0>:** Pins, A and C, Shutdown State Control bits
- 1x = Pins, A and C, tri-state (PIC18F4X80 devices)
- 01 = Drive Pins, A and C, to '1'
- 00 = Drive Pins, A and C, to '0'

bit 1-0 **PSSBD<1:0>:** Pins, B and D, Shutdown State Control bits(1)
- 1x = Pins, B and D, tri-state
- 01 = Drive Pins, B and D, to '1'
- 00 = Drive Pins, B and D, to '0'

**Note 1:** Reserved on PIC18F2X80 devices; maintain these bits clear.

#### 17.4.7.1 Auto-Shutdown and Auto-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 ECCP1DEL register (ECCP1DEL<7>).

In Shutdown mode with PRSEN = 1 (Figure 17-10), the ECCPASE bit will remain set for as long as the cause of the shutdown continues. When the shutdown condition clears, the ECCP1ASE bit is cleared. If PRSEN = 0 (Figure 17-11), 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, if the auto-shutdown source is one of the comparators, the shutdown condition is a level. 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.

### 17.4.8 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 CCP1M<1:0> bits (ECCP1CON<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 17-10: PWM AUTO-SHUTDOWN (PRSEN = 1, AUTO-RESTART ENABLED)**

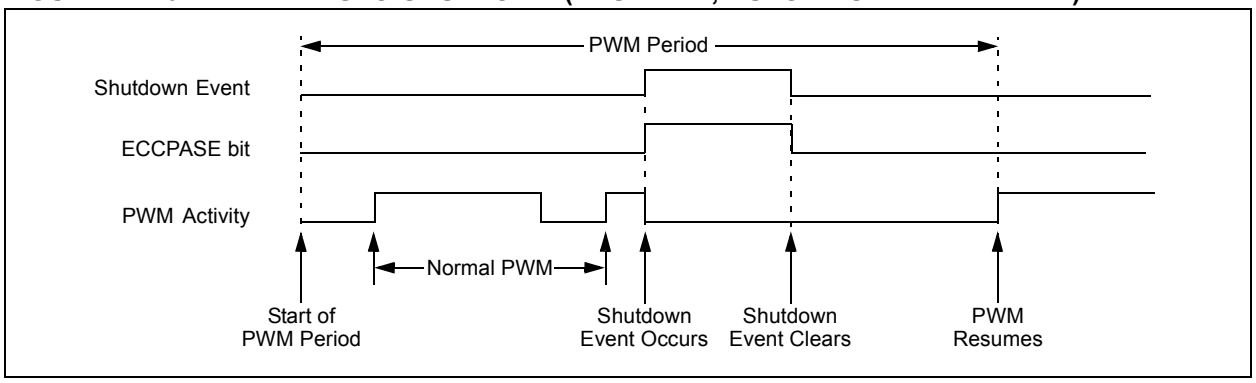
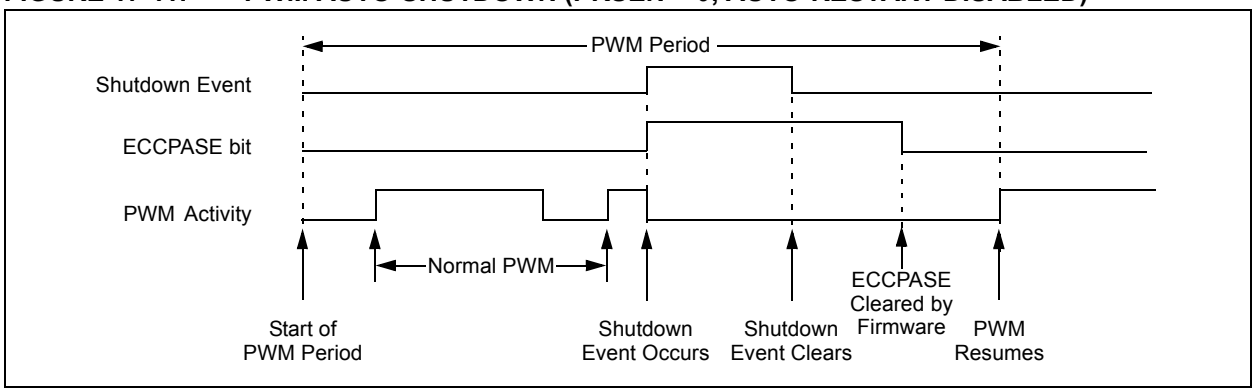



**FIGURE 17-11: PWM AUTO-SHUTDOWN (PRSEN = 0, AUTO-RESTART DISABLED)**





© 2009 Microchip Technology Inc.

### 17.4.9 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the ECCP module for PWM operation:

1. Configure the PWM pins, P1A and P1B (and P1C and P1D, if used), as inputs by setting the corresponding TRIS bits.
2. Set the PWM period by loading the PR2 register.
3. Configure the ECCP1 module for the desired PWM mode and configuration by loading the ECCP1CON register with the appropriate values:
   - Select one of the available output configurations and direction with the EPWM1M<1:0> bits.
   - Select the polarities of the PWM output signals with the ECCP1M<3:0> bits.
4. Set the PWM duty cycle by loading the ECCPR1L register and ECCP1CON<5:4> bits.
5. For Half-Bridge Output mode, set the dead-band delay by loading ECCP1DEL<6:0> with the appropriate value.
6. If auto-shutdown operation is required, load the ECCP1AS register:
   - Select the auto-shutdown sources using the ECCPAS<2:0> bits.
   - Select the shutdown states of the PWM output pins using PSSAC<1:0> and PSSBD<1:0> bits.
   - Set the ECCPASE bit (ECCP1AS<7>).
   - Configure the comparators using the CMCON register.
   - Configure the comparator inputs as analog inputs.
7. If auto-restart operation is required, set the PRSEN bit (ECCP1DEL<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 TMRn overflows (TMRnIF bit is set).
   - Enable the ECCP1/P1A, P1B, P1C and/or P1D pin outputs by clearing the respective TRIS bits.
   - Clear the ECCPASE bit (ECCP1AS<7>).

### 17.4.10 EFFECTS OF A RESET

Both Power-on Reset 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.

**TABLE 17-3: REGISTERS ASSOCIATED WITH ECCP1 MODULE AND TIMER1 TO TIMER3**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| RCON | IPEN | SBOREN | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 56 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR2 | OSCFIP | CMIP(3) | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP(3) | 57 |
| PIR2 | OSCFIF | CMIF(3) | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF(3) | 58 |
| PIE2 | OSCFIE | CMIE(3) | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE(3) | 58 |
| TRISB | PORTB Data Direction Register | | | | | | | | 58 |
| TRISC | PORTC Data Direction Register | | | | | | | | 58 |
| TRISD(1) | PORTD Data Direction Register | | | | | | | | 58 |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | 56 |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | 56 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{T1SYNC}$ | TMR1CS | TMR1ON | 56 |
| TMR2 | Timer2 Module Register | | | | | | | | 56 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | 56 |
| PR2 | Timer2 Period Register | | | | | | | | 56 |
| TMR3L | Holding Register for the Least Significant Byte of the 16-bit TMR3 Register | | | | | | | | 57 |
| TMR3H | Holding Register for the Most Significant Byte of the 16-bit TMR3 Register | | | | | | | | 57 |
| T3CON | RD16 | T3ECCP1(1) | T3CKPS1 | T3CKPS0 | T3CCP1(1) | $\overline{T3SYNC}$ | TMR3CS | TMR3ON | 57 |
| ECCPR1L(2) | Enhanced Capture/Compare/PWM Register 1 (LSB) | | | | | | | | 57 |
| ECCPR1H(2) | Enhanced Capture/Compare/PWM Register 1 (MSB) | | | | | | | | 57 |
| ECCP1CON(2) | EPWM1M1 | EPWM1M0 | EDC1B1 | EDC1B0 | ECCP1M3 | ECCP1M2 | ECCP1M1 | ECCP1M0 | 57 |
| ECCP1AS(2) | ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1(2) | PSSBD0(2) | 57 |
| ECCP1DEL(2) | PRSEN | PDC6(2) | PDC5(2) | PDC4(2) | PDC3(2) | PDC2(2) | PDC1(2) | PDC0(2) | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during ECCP operation.

**Note 1:** These bits are available on PIC18F4X80 devices only.

**2:** These bits or registers are unimplemented in PIC18F2X80 devices; always maintain these bit clear.

**3:** These bits are available on PIC18F4X80 and reserved on PIC18F2X80 devices.


© 2009 Microchip Technology Inc.

## 18.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

### 18.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D Converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit ($I^2C$)
  - Full Master mode
  - Slave mode (with general address call)

The $I^2C$ interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

### 18.2 Control Registers

The MSSP module has three associated registers. These include a status register (SSPSTAT) and two control registers (SSPCON1 and SSPCON2). The use of these registers and their individual configuration bits differ significantly depending on whether the MSSP module is operated in SPI or $I^2C$ mode.

Additional details are provided under the individual sections.

### 18.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

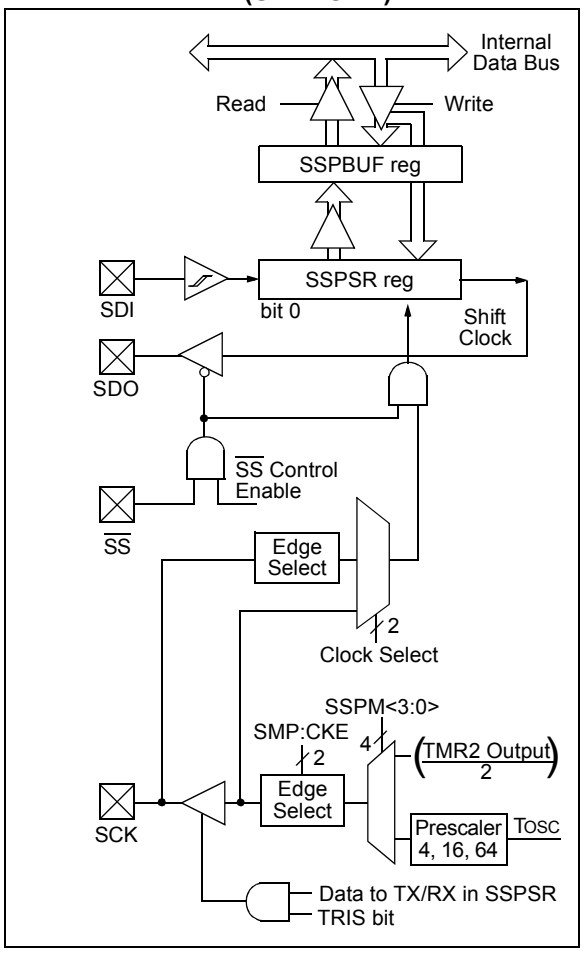
- Serial Data Out (SDO) – RC5/SDO
- Serial Data In (SDI) – RC4/SDI/SDA
- Serial Clock (SCK) – RC3/SCK/SCL

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{SS}$) – RA5/AN4/$\overline{SS}$/HLVDIN

Figure 18-1 shows the block diagram of the MSSP module when operating in SPI mode.

**FIGURE 18-1: MSSP BLOCK DIAGRAM (SPI MODE)**



### 18.3.1 REGISTERS

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPSR and SSPBUF together, create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

**REGISTER 18-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)**

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| -n = Value at POR | ‘1’ = Bit is set | ‘0’ = Bit is cleared | x = Bit is unknown |

bit 7 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Select bit
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state
Polarity of clock state is set by the CKP bit (SSPCON1<4>).

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit
Used in $I^2C$ mode only.

bit 4 **P:** Stop bit
Used in $I^2C$ mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** Start bit
Used in $I^2C$ mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{\text{Write}}$ Information bit
Used in $I^2C$ mode only.

bit 1 **UA:** Update Address bit
Used in $I^2C$ mode only.

bit 0 **BF:** Buffer Full Status bit (Receive mode only)
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty


© 2009 Microchip Technology Inc.

**REGISTER 18-2: SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| WCOL | SSPOV(1) | SSPEN(2) | CKP | SSPM3(3) | SSPM2(3) | SSPM1(3) | SSPM0(3) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit(1)
SPI Slave mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
0 = No overflow

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(2)
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins(2)
0 = Disables serial port and configures these pins as I/O port pins(2)

bit 4 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Master Synchronous Serial Port Mode Select bits(3)
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
0011 = SPI Master mode, clock = TMR2 output/2
0010 = SPI Master mode, clock = FOSC/64
0001 = SPI Master mode, clock = FOSC/16
0000 = SPI Master mode, clock = FOSC/4

**Note 1:** In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.
**2:** When enabled, these pins must be properly configured as input or output.
**3:** Bit combinations not specifically listed here are either reserved or implemented in $I^2C$™ mode only.

### 18.3.2 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

The MSSP consists of a Transmit/Receive Shift register (SSPSR) and a Buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready. Once the 8 bits of data have been received, that byte is moved to the SSPBUF register. Then, the Buffer Full detect bit, BF (SSPSTAT<0>), and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the write collision detect bit, WCOL (SSPCON1<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. The Buffer Full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. The SSPBUF must be read and/or written. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 18-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP Status register (SSPSTAT) indicates the various status conditions.

| **Note:** | The SSPBUF register cannot be used with read-modify-write instructions such as `BCF`, `BTFSC` and `COMF`, etc. |
|---|---|

| **Note:** | To avoid lost data in Master mode, a read of the SSPBUF must be performed to clear the Buffer Full (BF) detect bit (SSPSTAT<0>) between each transmission. |
|---|---|

**EXAMPLE 18-1: LOADING THE SSPBUF (SSPSR) REGISTER**

```
LOOP    BTFSS   SSPSTAT, BF     ;Has data been received (transmit complete)?
        BRA     LOOP            ;No
        MOVF    SSPBUF, W       ;WREG reg = contents of SSPBUF
        MOVWF   RXDATA          ;Save in user RAM, if data is meaningful
        MOVF    TXDATA, W       ;W reg = contents of TXDATA
        MOVWF   SSPBUF          ;New data to xmit
```


© 2009 Microchip Technology Inc.

### 18.3.3 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize the SSPCON registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> bit cleared
- SCK (Master mode) must have TRISC<3> bit cleared
- SCK (Slave mode) must have TRISC<3> bit set
- $\overline{SS}$ must have TRISF<7> bit set

Any serial port function that is not desired may be overridden by programming the corresponding Data Direction (TRIS) register to the opposite value.

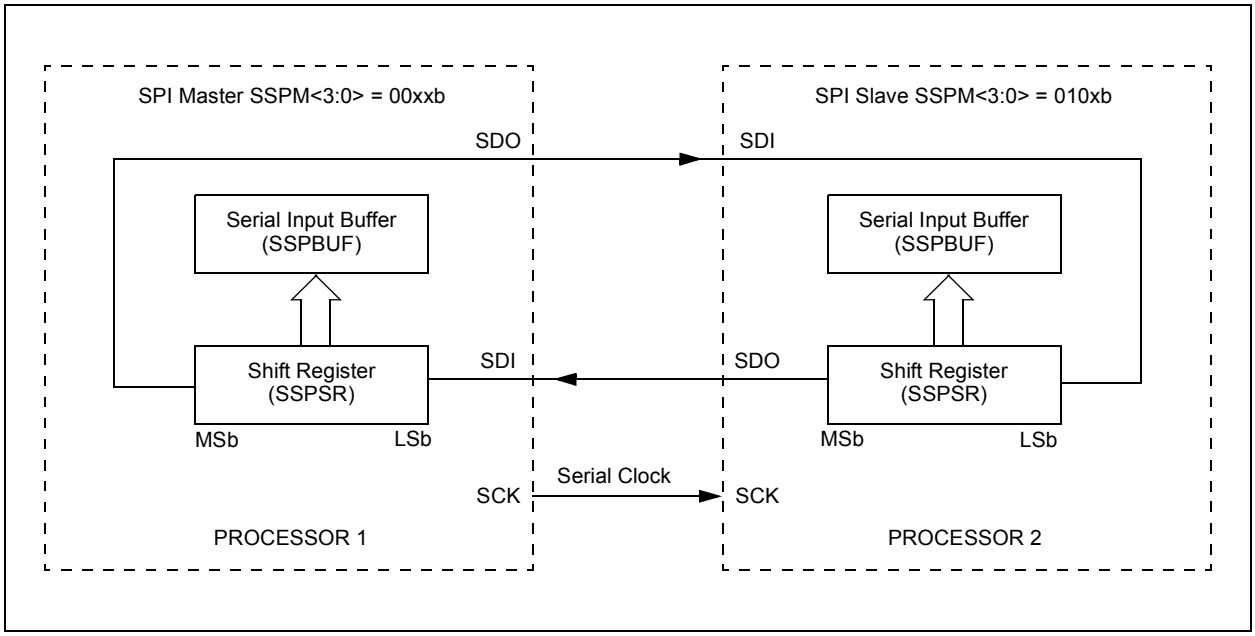
### 18.3.4 TYPICAL CONNECTION

Figure 18-2 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data – Slave sends dummy data
- Master sends data – Slave sends data
- Master sends dummy data – Slave sends data

| **Note:** | When the module is enabled and in Master mode (CKE, SSPSTAT<6> = 1), a small glitch of approximately half a TCY may be seen on the SCK pin. To resolve this, keep the SCK pin as an input while setting SPEN. Then, configure the SCK pin as an output (TRISC<3> = 0). |
|---|---|

**FIGURE 18-2: SPI MASTER/SLAVE CONNECTION**



### 18.3.5 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 18-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a “Line Activity Monitor” mode.

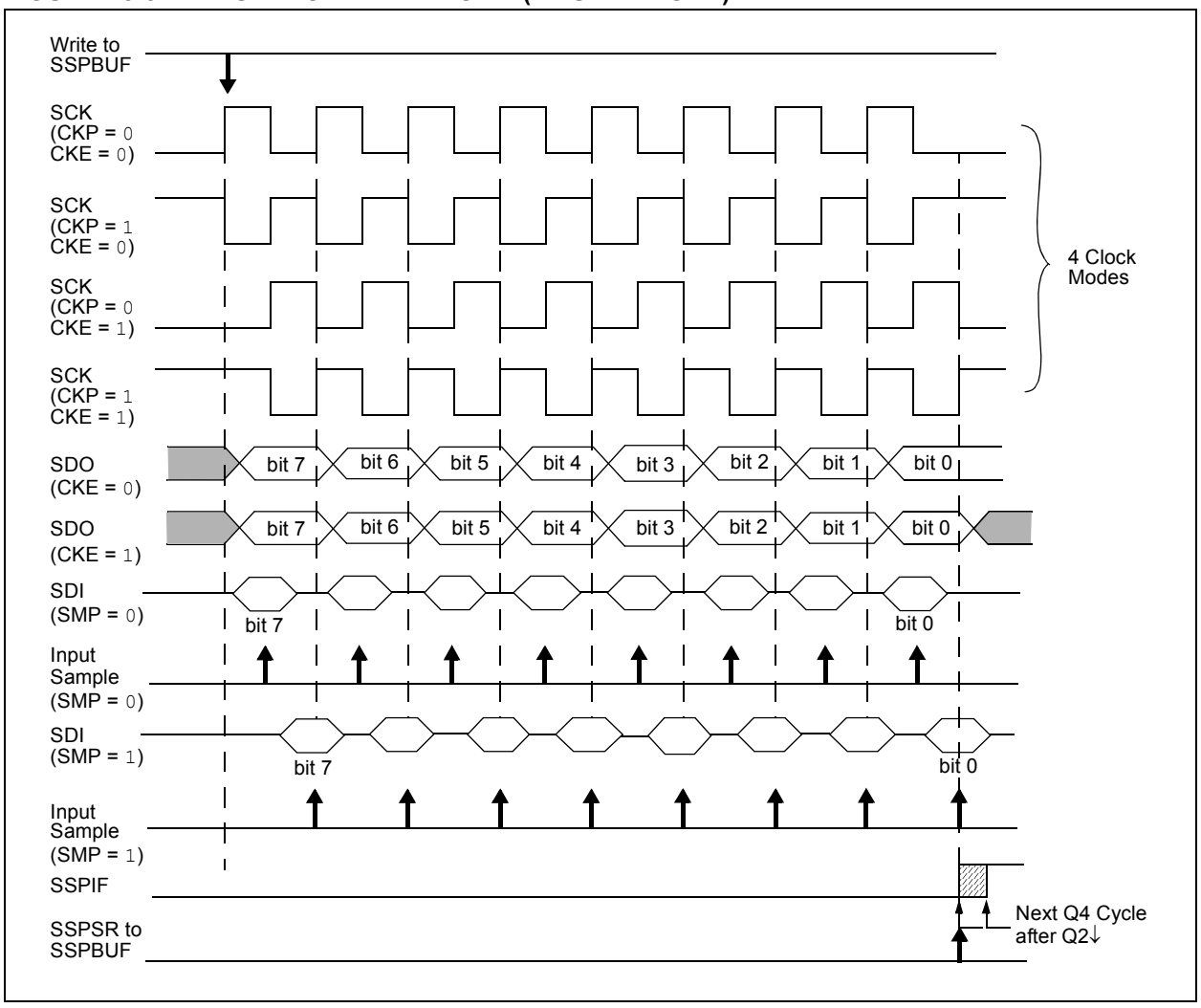
The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then, would give waveforms for SPI communication as shown in Figure 18-3, Figure 18-5 and Figure 18-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user-programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 18-3 shows the waveforms for Master mode. When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

**FIGURE 18-3: SPI MODE WAVEFORM (MASTER MODE)**





© 2009 Microchip Technology Inc.

### 18.3.6 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag bit is set.

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit (SSPCON1<4>).

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. When a byte is received, the device will wake-up from Sleep.

### 18.3.7 SLAVE SELECT SYNCHRONIZATION

The $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 04h). The pin must not be driven low for the $\overline{SS}$ pin to function as an input. The data latch must be high. When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven. When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

| | |
|---|---|
| **Note 1:** | When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPCON<3:0> = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to VDD. |
| **2:** | If the SPI is used in Slave mode with CKE set, then the $\overline{SS}$ pin control must be enabled. |

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the $\overline{SS}$ pin to a high level or clearing the SSPEN bit.

To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function) since it cannot create a bus conflict.

**FIGURE 18-4: SLAVE SYNCHRONIZATION WAVEFORM**

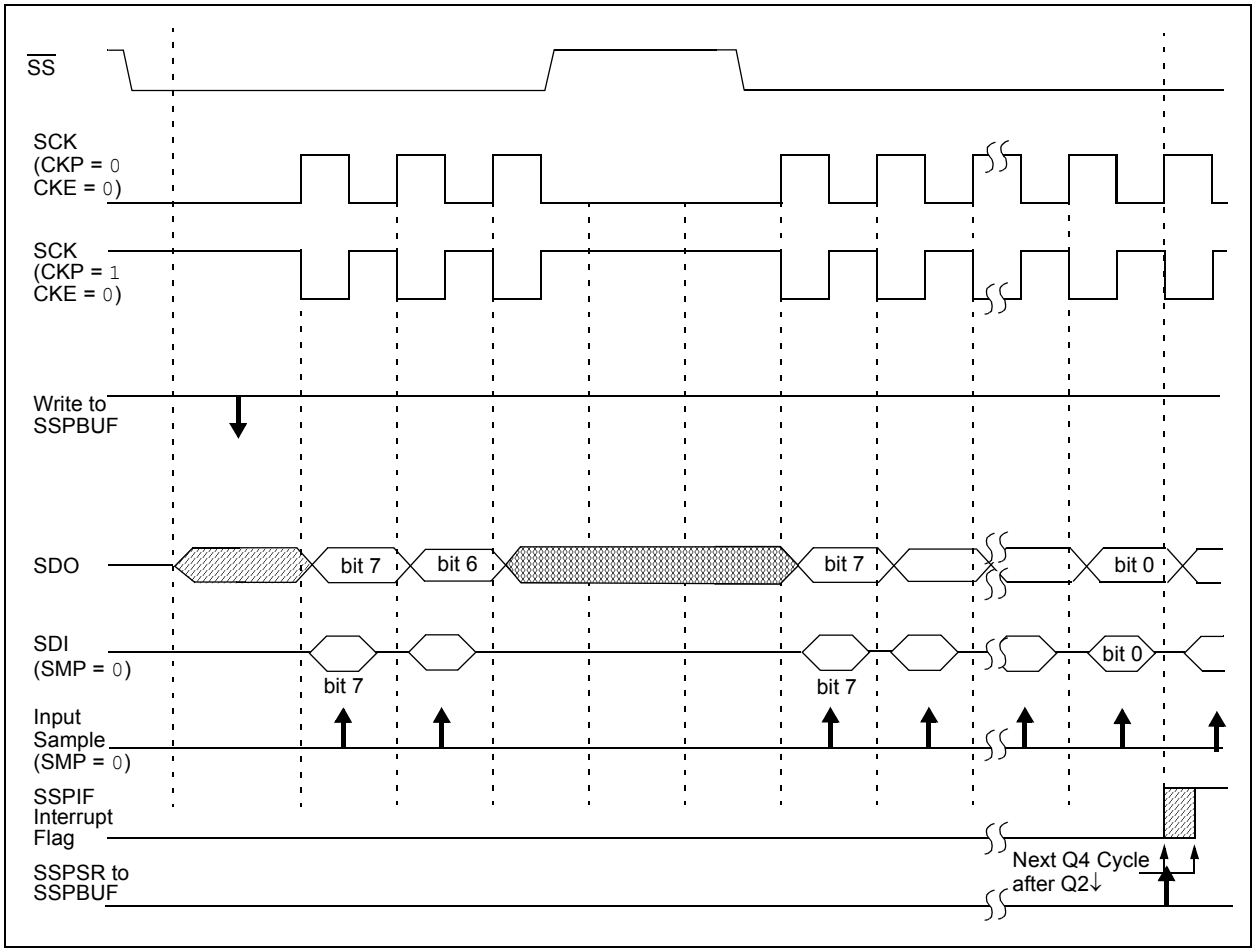


**FIGURE 18-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**

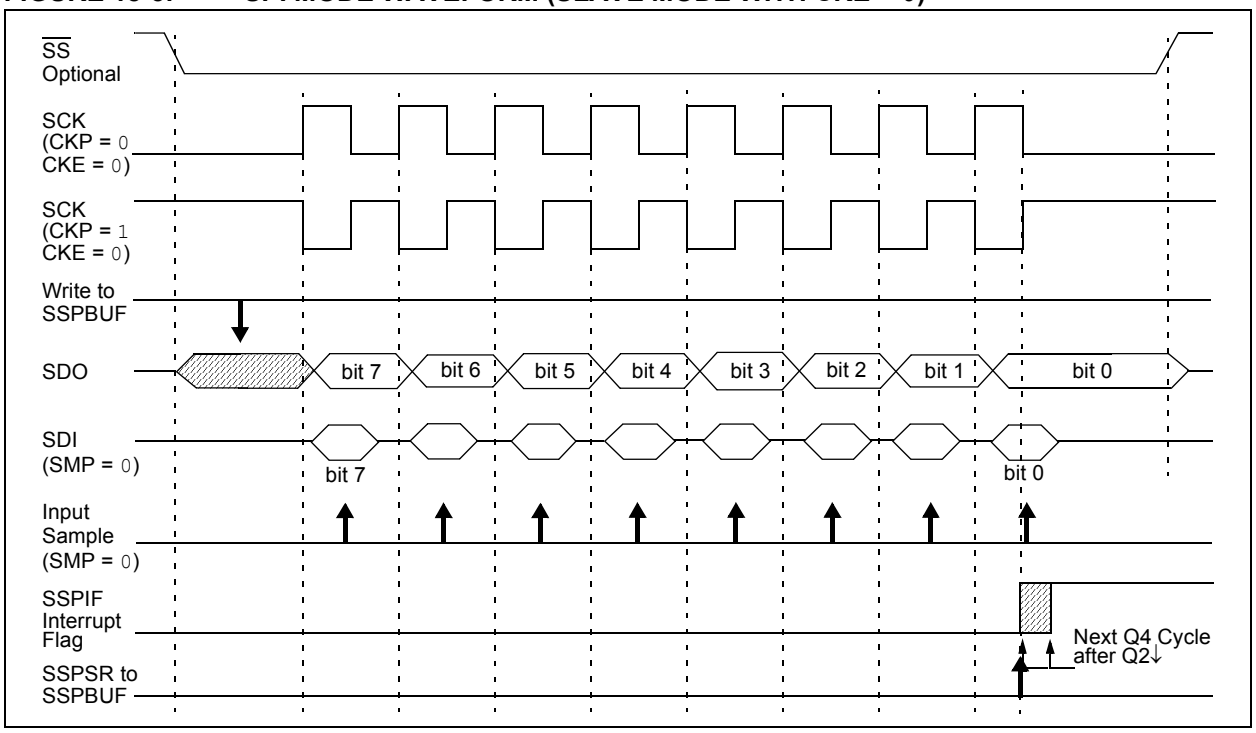
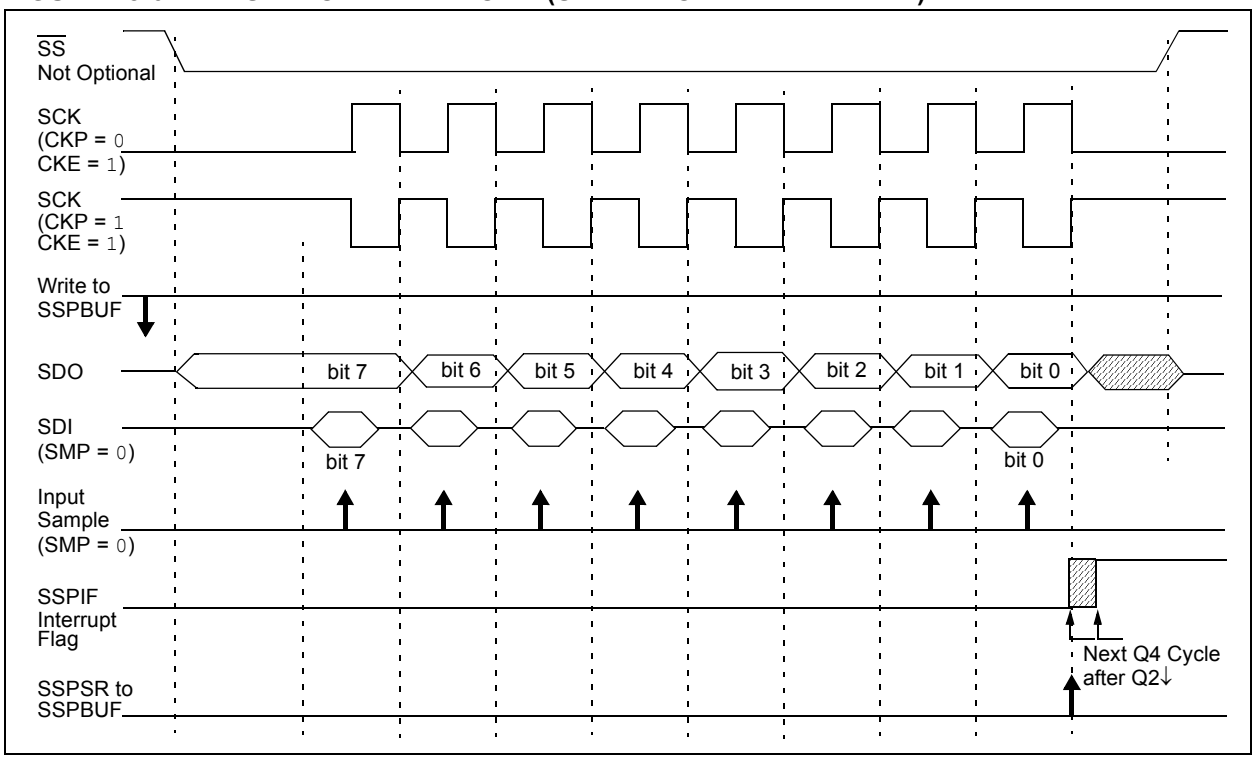



**FIGURE 18-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**





© 2009 Microchip Technology Inc.

### 18.3.8 OPERATION IN POWER-MANAGED MODES

In SPI Master mode, module clocks may be operating at a different speed than when in full-power mode; in the case of the Sleep mode, all clocks are halted.

In most power-managed modes, a clock is provided to the peripherals. That clock should be from the primary clock source, the secondary clock (Timer1 oscillator at 32.768 kHz) or the INTOSC source. See **Section 3.7 "Clock Sources and Oscillator Switching"** for additional information.

In most cases, the speed that the master clocks SPI data is not important; however, this should be evaluated for each system.

If MSSP interrupts are enabled, they can wake the controller from Sleep mode, or one of the Idle modes, when the master completes sending data. If an exit from Sleep or Idle mode is not desired, MSSP interrupts should be disabled.

If the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in any power-managed mode and data to be shifted into the SPI Transmit/Receive Shift register. When all 8 bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

### 18.3.9 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

### 18.3.10 BUS MODE COMPATIBILITY

Table 18-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

**TABLE 18-1: SPI BUS MODES**

| Standard SPI Mode Terminology | Control Bits State | |
|---|---|---|
| | CKP | CKE |
| 0, 0 | 0 | 1 |
| 0, 1 | 0 | 0 |
| 1, 0 | 1 | 1 |
| 1, 1 | 1 | 0 |

There is also a SMP bit which controls when the data is sampled.

**TABLE 18-2: REGISTERS ASSOCIATED WITH SPI OPERATION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| TRISA | PORTA Data Direction Register | | | | | | | | 58 |
| TRISC | PORTC Data Direction Register | | | | | | | | 58 |
| SSPBUF | Synchronous Serial Port Receive Buffer/Transmit Register | | | | | | | | 56 |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 56 |
| SSPSTAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 56 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the MSSP in SPI mode.

**Note 1:** These bits are unimplemented in PIC18F2X80 devices; always maintain these bits clear.

## 18.4 I²C Mode

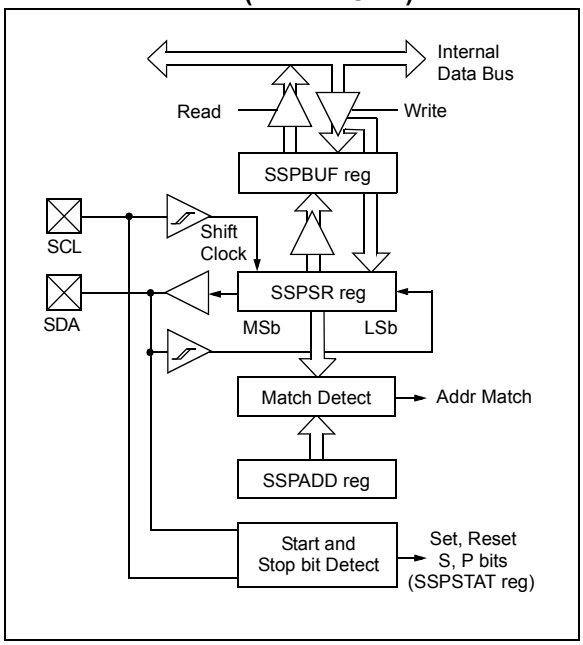
The MSSP module in I²C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial Clock (SCL) – RC3/SCK/SCL
- Serial Data (SDA) – RC4/SDI/SDA

The user must configure these pins as inputs or outputs through the TRISC<4:3> bits.

**FIGURE 18-7: MSSP BLOCK DIAGRAM (I²C™ MODE)**




### 18.4.1 REGISTERS

The MSSP module has six registers for I²C operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Control Register 2 (SSPCON2)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible
- MSSP Address Register (SSPADD)

SSPCON1, SSPCON2 and SSPSTAT are the control and status registers in I²C mode operation. The SSPCON1 and SSPCON2 registers are readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

SSPADD register holds the slave device address when the MSSP is configured in I²C Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPADD act as the Baud Rate Generator reload value.

In receive operations, SSPSR and SSPBUF together, create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.


© 2009 Microchip Technology Inc.

**REGISTER 18-3: SSPSTAT: MSSP STATUS REGISTER (I²C™ MODE)**

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SMP | CKE | D/$\overline{A}$ | P(1) | S(1) | R/$\overline{W}$(2,3) | UA | BF |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **SMP:** Slew Rate Control bit

In Master or Slave mode:

1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)

0 = Slew rate control enabled for High-Speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit

In Master or Slave mode:

1 = Enable SMBus specific inputs

0 = Disable SMBus specific inputs

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit

In Master mode:

Reserved.

In Slave mode:

1 = Indicates that the last byte received or transmitted was data

0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit(1)

1 = Indicates that a Stop bit has been detected last

0 = Stop bit was not detected last

bit 3 **S:** Start bit(1)

1 = Indicates that a Start bit has been detected last

0 = Start bit was not detected last

bit 2 **R/$\overline{W}$:** Read/$\overline{\text{Write}}$ Information bit (I²C mode only)(2,3)

In Slave mode:

1 = Read

0 = Write

In Master mode:

1 = Transmit is in progress

0 = Transmit is not in progress

bit 1 **UA:** Update Address bit (10-Bit Slave mode only)

1 = Indicates that the user needs to update the address in the SSPADD register

0 = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit

In Receive mode:

1 = Receive complete, SSPBUF is full

0 = Receive is not complete, SSPBUF is empty

In Transmit mode:

1 = Data transmit in progress (does not include the $\overline{\text{ACK}}$ and Stop bits), SSPBUF is full

0 = Data transmit complete (does not include the $\overline{\text{ACK}}$ and Stop bits), SSPBUF is empty

**Note 1:** This bit is cleared on Reset and when SSPEN is cleared.

**2:** This bit holds the R/$\overline{W}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{\text{ACK}}$ bit.

**3:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Idle mode.

**REGISTER 18-4: SSPCON1: MSSP CONTROL REGISTER 1 (I²C™ MODE)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| WCOL | SSPOV | SSPEN(1) | CKP | SSPM3(2) | SSPM2(2) | SSPM1(2) | SSPM0(2) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **WCOL:** Write Collision Detect bit

In Master Transmit mode:
1 = A write to the SSPBUF register was attempted while the I²C conditions were not valid for a transmission to be started (must be cleared in software)
0 = No collision

In Slave Transmit mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

In Receive mode (Master or Slave modes):
This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared in software)
0 = No overflow

In Transmit mode:
This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(1)
1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCK Release Control bit

In Slave mode:
1 = Releases clock
0 = Holds clock low (clock stretch), used to ensure data setup time

In Master mode:
Unused in this mode.

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits(2)
1111 = I²C Slave mode, 10-bit address with Start and Stop bit interrupts enabled
1110 = I²C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1011 = I²C Firmware Controlled Master mode (slave Idle)
1000 = I²C Master mode, clock = FOSC/(4 * (SSPADD + 1))
0111 = I²C Slave mode, 10-bit address
0110 = I²C Slave mode, 7-bit address

**Note 1:** When enabled, the SDA and SCL pins must be properly configured as input or output.
**2:** Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.


© 2009 Microchip Technology Inc.

**REGISTER 18-5: SSPCON2: MSSP CONTROL REGISTER 2 ($I^2C$™ MODE)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| GCEN | ACKSTAT | ACKDT(1) | ACKEN(2) | RCEN(2) | PEN(2) | RSEN(2) | SEN(2) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **GCEN:** General Call Enable bit (Slave mode only)
1 = Enables interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)(1)
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)(2)
1 = Initiates Acknowledge sequence on SDA and SCL pins and transmits the ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence Idle

bit 3 **RCEN:** Receive Enable bit (Master mode only)(2)
1 = Enables Receive mode for $I^2C$
0 = Receive Idle

bit 2 **PEN:** Stop Condition Enable bit (Master mode only)(2)
1 = Initiates Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enable bit (Master mode only(2)
1 = Initiates Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enable/Stretch Enable bit(2)
In Master mode:
1 = Initiates Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle
In Slave mode:
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is disabled

**Note 1:** Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
**2:** For bits, ACKEN, RCEN, PEN, RSEN, SEN: If the $I^2C$ module is not in the Idle mode, these bits may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

### 18.4.2 OPERATION

The MSSP module functions are enabled by setting the MSSP Enable bit, SSPEN (SSPCON<5>).

The SSPCON1 register allows control of the $I^2C$ operation. Four mode selection bits (SSPCON<3:0>) allow one of the following $I^2C$ modes to be selected:

- $I^2C$ Master mode, clock = (FOSC/4) x (SSPADD + 1)
- $I^2C$ Slave mode (7-bit address)
- $I^2C$ Slave mode (10-bit address)
- $I^2C$ Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- $I^2C$ Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- $I^2C$ Firmware Controlled Master mode, slave is Idle

Selection of any $I^2C$ mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain, provided these pins are programmed to inputs by setting the appropriate TRISC bits. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

### 18.4.3 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs (TRISC<4:3> set). The MSSP module will override the input state with the output data when required (slave-transmitter).

The $I^2C$ Slave mode hardware will always generate an interrupt on an address match. Through the mode select bits, the user can also choose to interrupt on Start and Stop bits

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge ($\overline{ACK}$) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

Any combination of the following conditions will cause the MSSP module not to give this $\overline{ACK}$ pulse:

- The Buffer Full bit, BF (SSPSTAT<0>), was set before the transfer was received.
- The overflow bit, SSPOV (SSPCON<6>), was set before the transfer was received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit, SSPIF (PIR1<3>), is set. The BF bit is cleared by reading the SSPBUF register, while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the $I^2C$ specification, as well as the requirement of the MSSP module, are shown in timing parameter 100 and parameter 101.

#### 18.4.3.1 Addressing

Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the 8-bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register, SSPSR<7:1>, is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

1. The SSPSR register value is loaded into the SSPBUF register.
2. The Buffer Full bit, BF, is set.
3. An $\overline{ACK}$ pulse is generated.
4. MSSP Interrupt Flag bit, SSPIF (PIR1<3>), is set (interrupt is generated, if enabled) on the falling edge of the ninth SCL pulse.

In 10-Bit Addressing mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit, R/$\overline{W}$ (SSPSTAT<2>), must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '11110 A9 A8 0', where 'A9' and 'A8' are the two MSbs of the address. The sequence of events for 10-bit addressing is as follows, with steps 7 through 9 for the slave-transmitter:

1. Receive first (high) byte of address (bits, SSPIF, BF and UA (SSPSTAT<1>), are set).
2. Update the SSPADD register with second (low) byte of address (clears bit, UA, and releases the SCL line).
3. Read the SSPBUF register (clears bit, BF) and clear flag bit, SSPIF.
4. Receive second (low) byte of address (bits, SSPIF, BF and UA, are set).
5. Update the SSPADD register with the first (high) byte of address. If match releases SCL line, this will clear bit, UA.
6. Read the SSPBUF register (clears bit BF) and clear flag bit, SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of address (bits, SSPIF and BF, are set).
9. Read the SSPBUF register (clears bit, BF) and clear flag bit, SSPIF.


© 2009 Microchip Technology Inc.

#### 18.4.3.2 Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and the SDA line is held low ($\overline{ACK}$).

When the address byte overflow condition exists, then the no Acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit, BF (SSPSTAT<0>), is set, or bit, SSPOV (SSPCON1<6>), is set.

An MSSP interrupt is generated for each data transfer byte. Flag bit, SSPIF (PIR1<3>), must be cleared in software. The SSPSTAT register is used to determine the status of the byte.

If SEN is enabled (SSPCON2<0> = 1), RC3/SCK/SCL will be held low (clock stretch) following each data transfer. The clock must be released by setting bit, CKP (SSPCON<4>). See **Section 18.4.4 "Clock Stretching"** for more details.

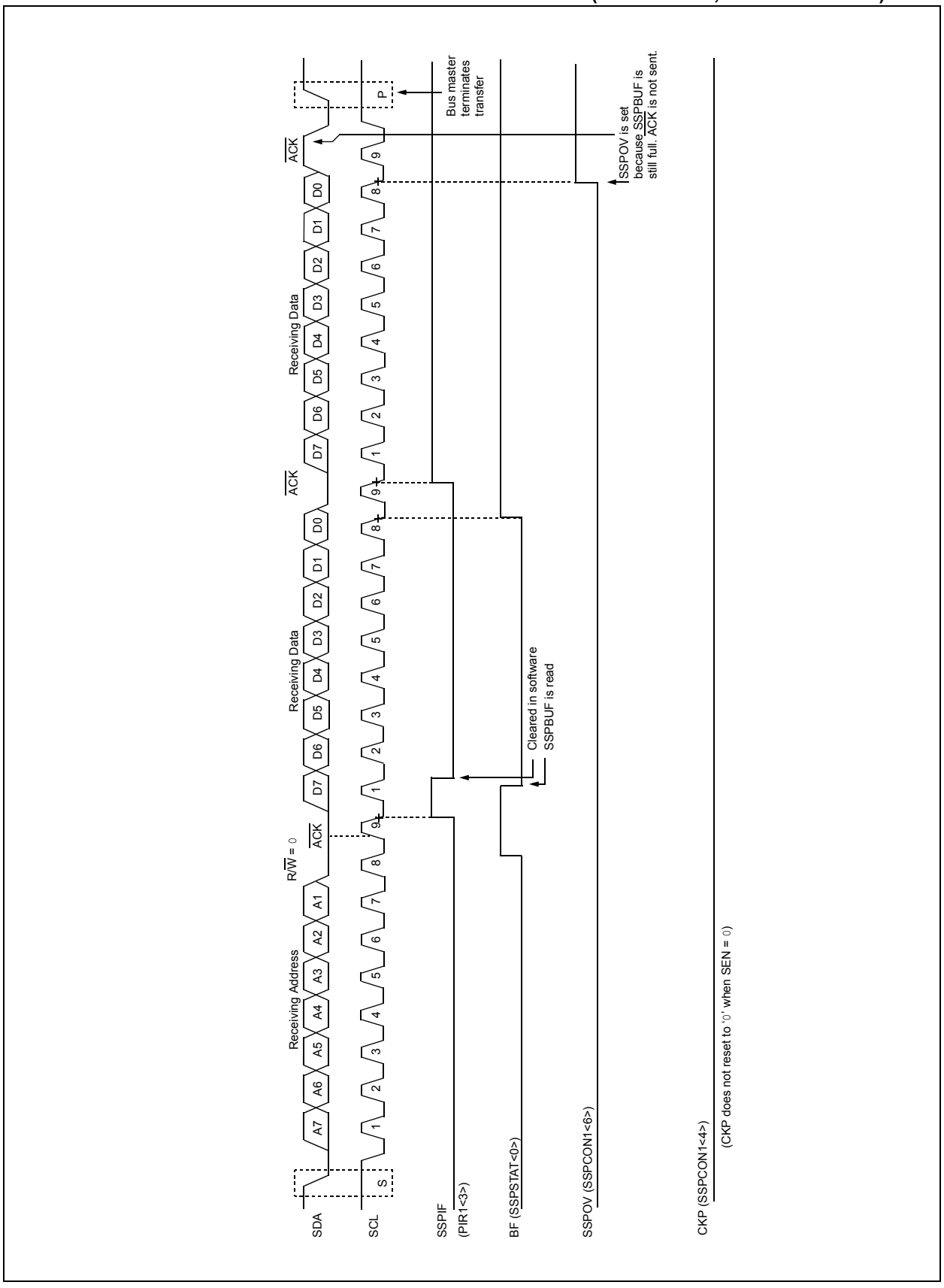
#### 18.4.3.3 Transmission

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{ACK}$ pulse will be sent on the ninth bit and pin RC3/SCK/SCL is held low regardless of SEN (see **Section 18.4.4 "Clock Stretching"** for more details). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPBUF register which also loads the SSPSR register. Then, the RC3/SCK/SCL pin should be enabled by setting bit, CKP (SSPCON1<4>). The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time (Figure 18-9).

The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not $\overline{ACK}$), then the data transfer is complete. In this case, when the $\overline{ACK}$ is latched by the slave, the slave logic is reset and the slave monitors for another occurrence of the Start bit. If the SDA line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPBUF register. Again, pin, RC3/SCK/SCL, must be enabled by setting bit, CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

**FIGURE 18-8: I²C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)**





© 2009 Microchip Technology Inc.

**FIGURE 18-9: I²C™ SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)**

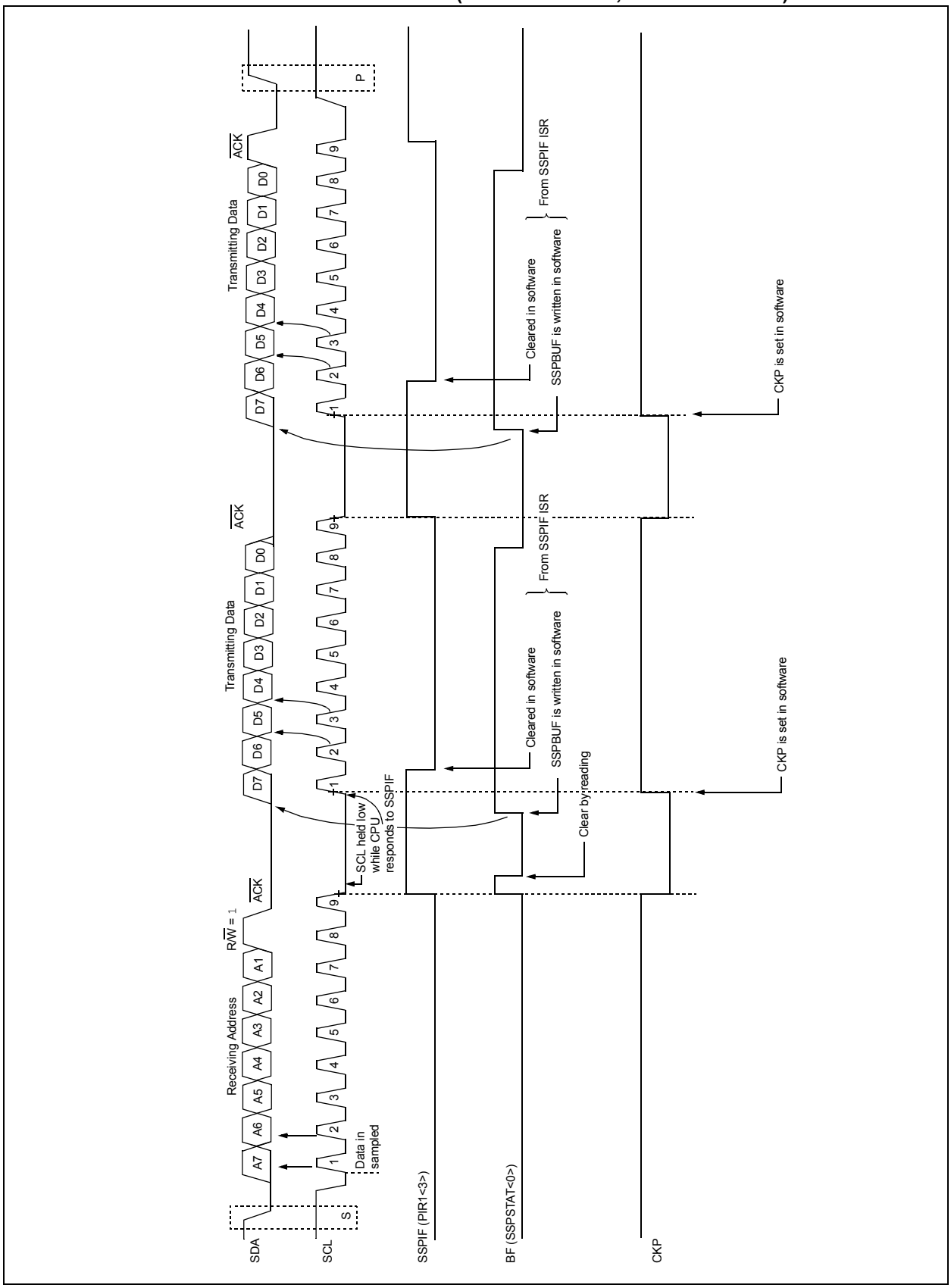
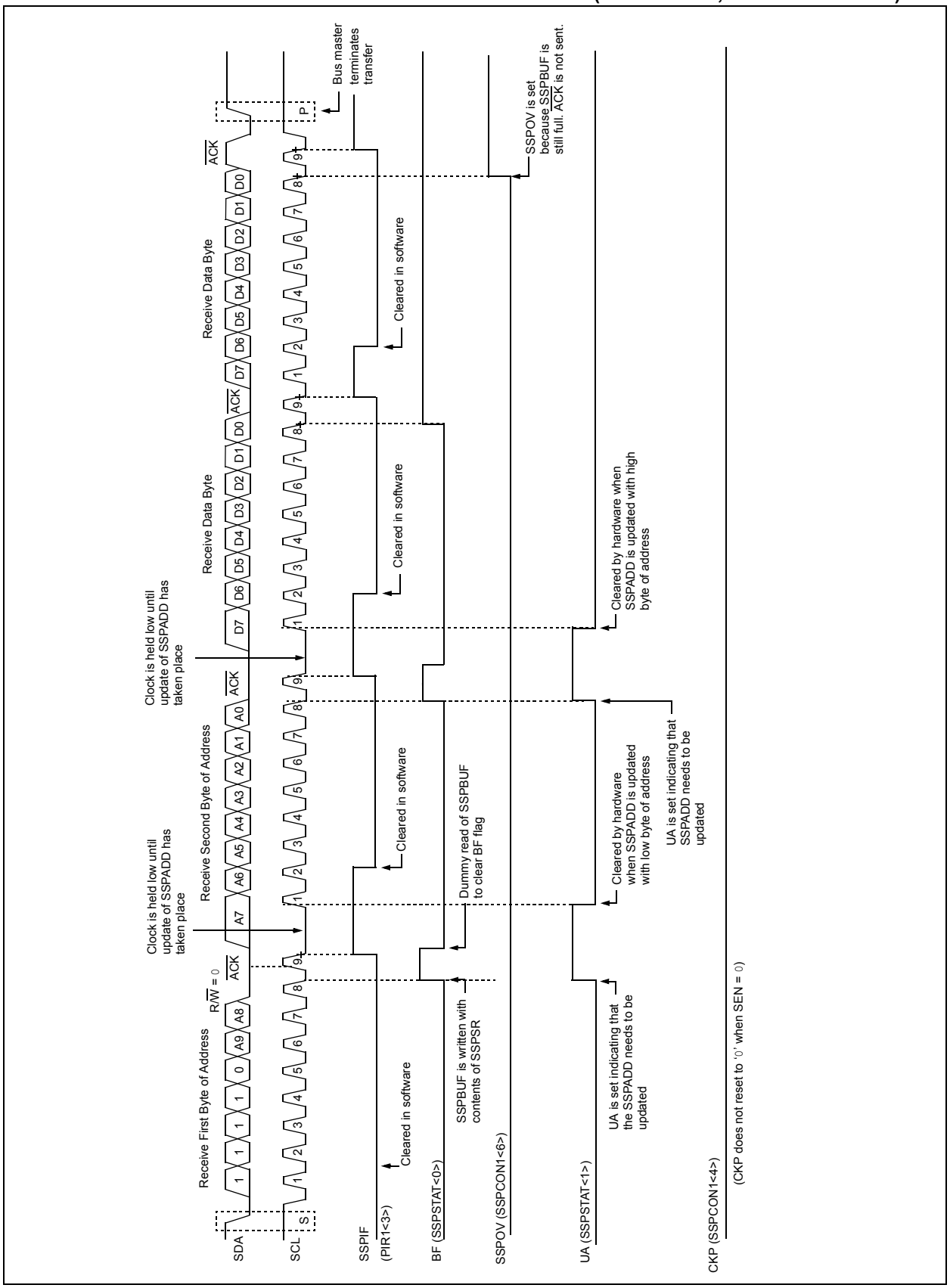


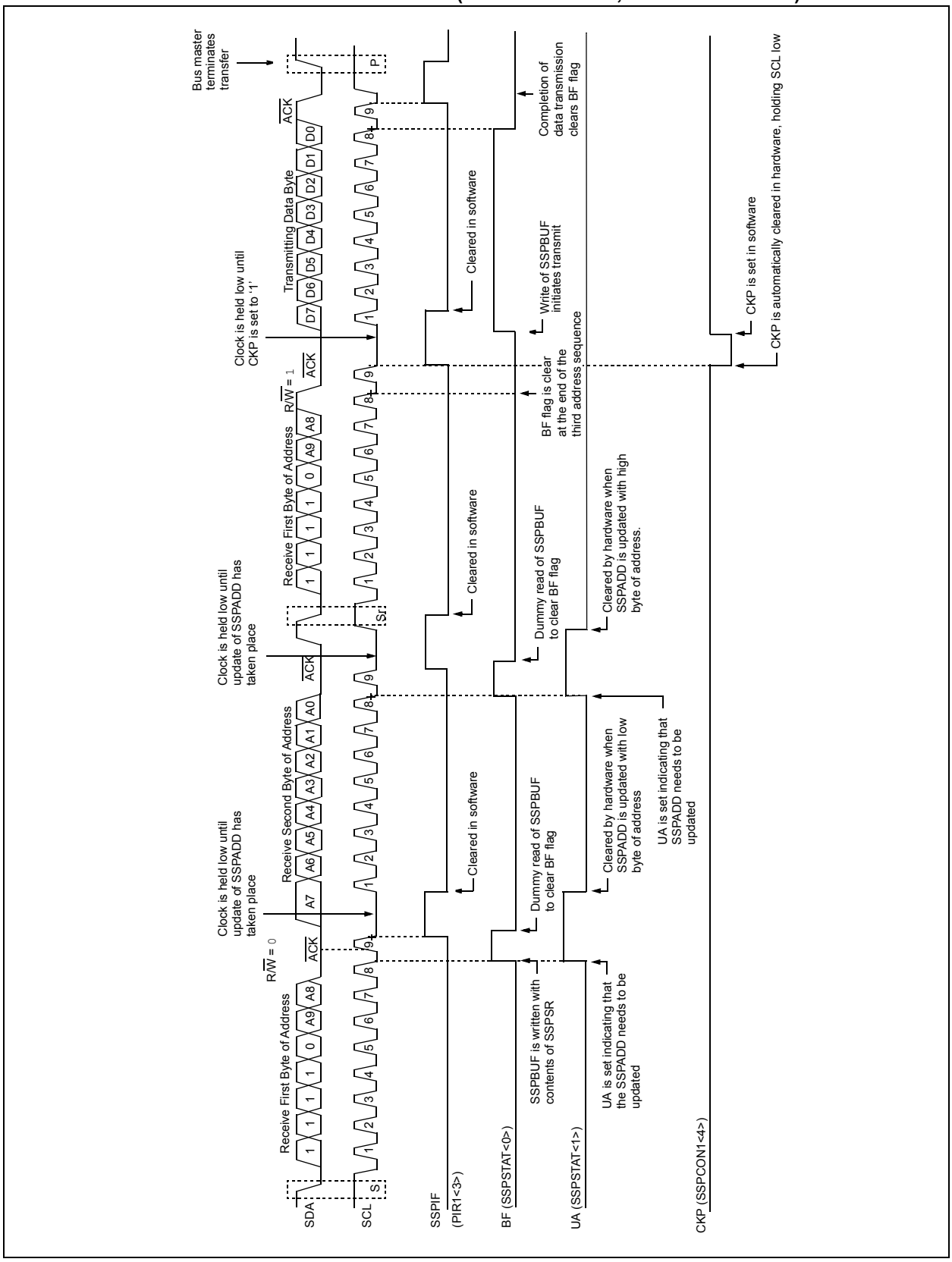
**FIGURE 18-10: I²C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)**





© 2009 Microchip Technology Inc.

**FIGURE 18-11: I²C™ SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)**



### 18.4.4 CLOCK STRETCHING

Both 7 and 10-Bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

#### 18.4.4.1 Clock Stretching for 7-Bit Slave Receive Mode (SEN = 1)

In 7-Bit Slave Receive mode, on the falling edge of the ninth clock at the end of the $\overline{ACK}$ sequence if the BF bit is set, the CKP bit in the SSPCON1 register is automatically cleared, forcing the SCL output to be held low. The CKP bit being cleared to '0' will assert the SCL line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the SSPBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 18-13).

**Note 1:** If the user reads the contents of the SSPBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.

**2:** The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

#### 18.4.4.2 Clock Stretching for 10-Bit Slave Receive Mode (SEN = 1)

In 10-Bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{W}$ bit cleared to '0'. The release of the clock line occurs upon updating SSPADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

**Note:** If the user polls the UA bit and clears it by updating the SSPADD register before the falling edge of the ninth clock occurs, and if the user hasn't cleared the BF bit by reading the SSPBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

#### 18.4.4.3 Clock Stretching for 7-Bit Slave Transmit Mode

7-Bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the SSPBUF before the master device can initiate another transmit sequence (see Figure 18-9).

**Note 1:** If the user loads the contents of SSPBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

**2:** The CKP bit can be set in software regardless of the state of the BF bit.

#### 18.4.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence, which contains the high-order bits of the 10-bit address and the R/$\overline{W}$ bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-Bit Slave Transmit mode (see Figure 18-11).


© 2009 Microchip Technology Inc.

#### 18.4.4.5 Clock Synchronization and the CKP bit

When the CKP bit is cleared, the SCL output is forced to '0'. However, setting the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external $I^2C$ master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the $I^2C$ bus have deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 18-12).

**FIGURE 18-12: CLOCK SYNCHRONIZATION TIMING**

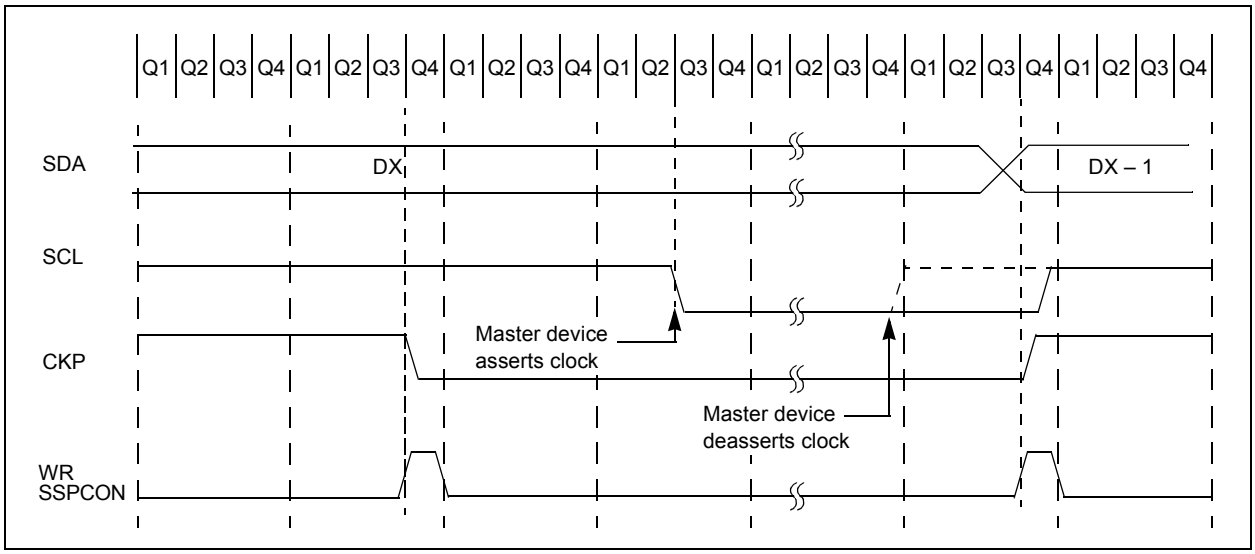
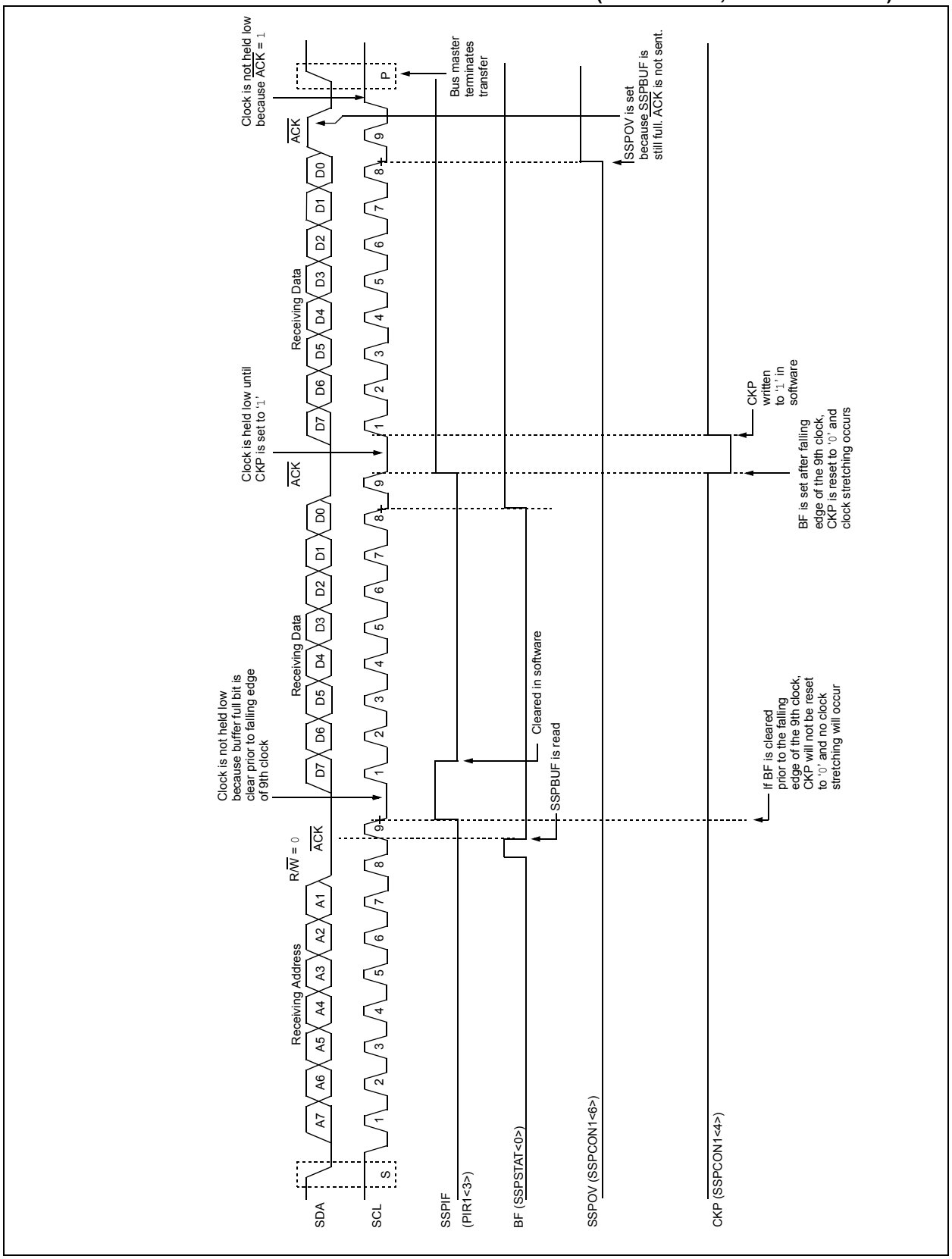


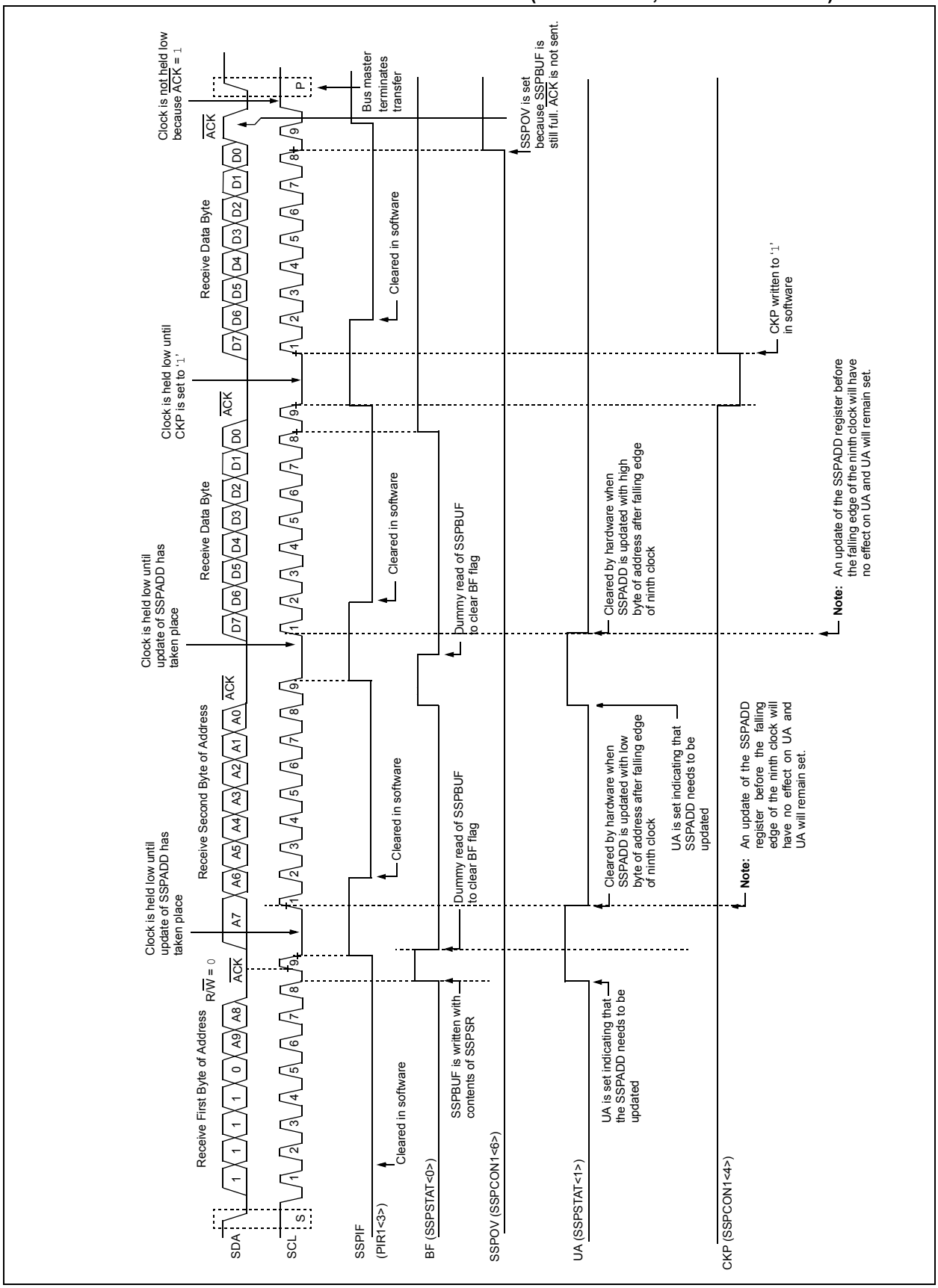
**FIGURE 18-13: I²C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)**





© 2009 Microchip Technology Inc.

**FIGURE 18-14: I²C™ SLAVE MODE TIMING SEN = 1 (RECEPTION, 10-BIT ADDRESS)**



### 18.4.5 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the $I^2C$ bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an Acknowledge.

The general call address is one of eight addresses reserved for specific purposes by the $I^2C$ protocol. It consists of all '0's with R/$\overline{W}$ = 0.

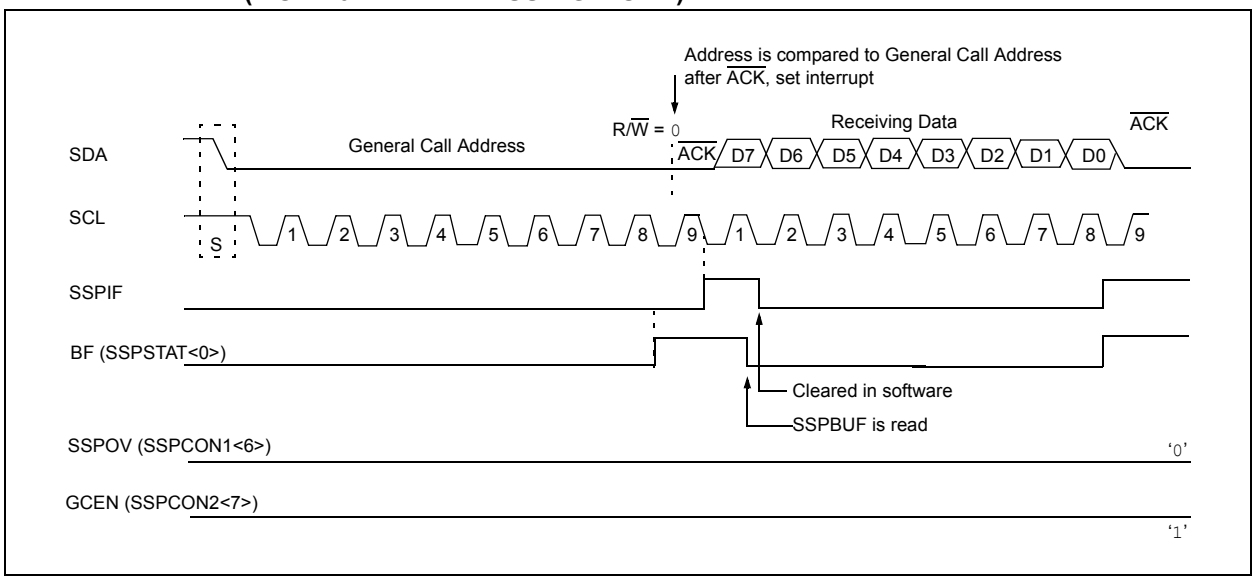
The general call address is recognized when the General Call Enable bit (GCEN) is enabled (SSPCON2<7> set). Following a Start bit detect, 8 bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF flag bit is set (eighth bit), and on the falling edge of the ninth bit ($\overline{ACK}$ bit), the SSPIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPBUF. The value can be used to determine if the address was device-specific or a general call address.

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match and the UA bit is set (SSPSTAT<1>). If the general call address is sampled when the GCEN bit is set, while the slave is configured in 10-Bit Addressing mode, then the second half of the address is not necessary, the UA bit will not be set and the slave will begin receiving data after the Acknowledge (Figure 18-15).

**FIGURE 18-15: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESSING MODE)**





© 2009 Microchip Technology Inc.

### 18.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the $I^2C$ bus may be taken when the P bit is set or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all $I^2C$ bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

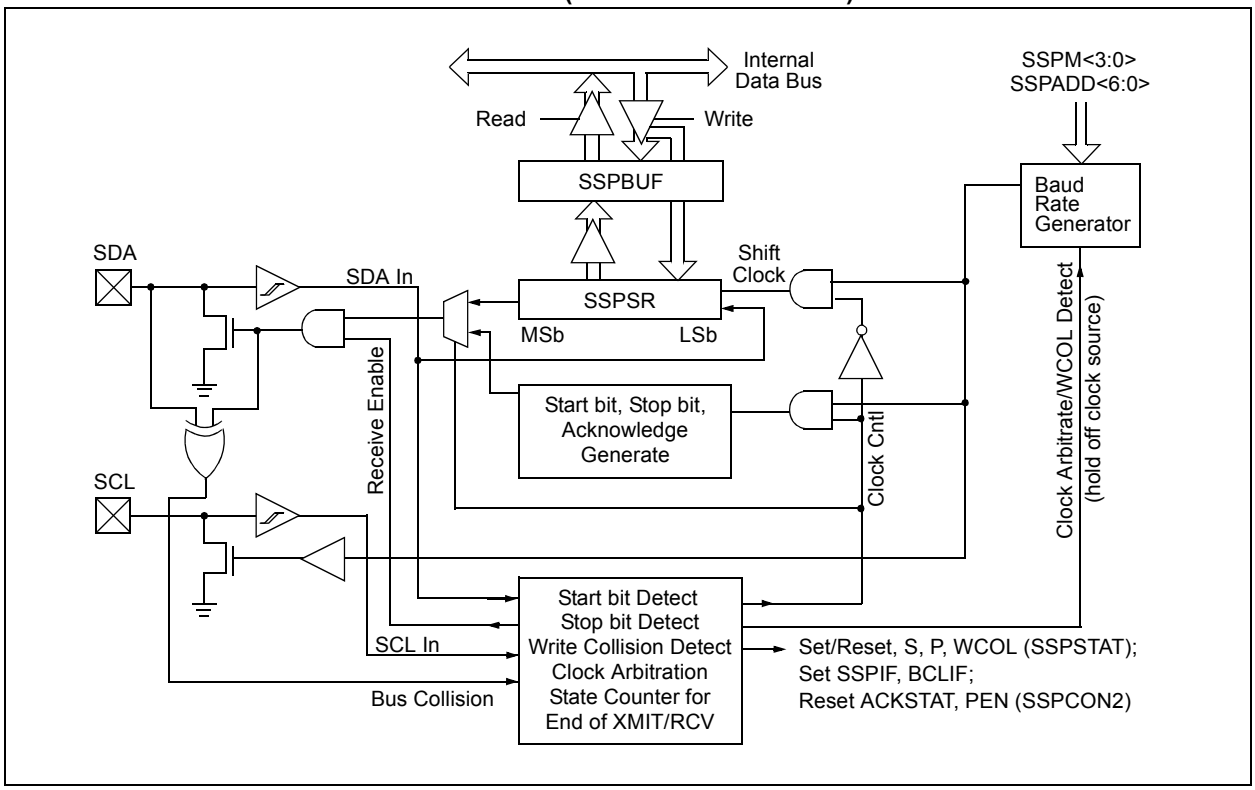
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the $I^2C$ port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDA and SCL.

**Note:** The MSSP module, when configured in $I^2C$ Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

The following events will cause the MSSP Interrupt Flag bit, SSPIF, to be set (MSSP interrupt, if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmitted
- Repeated Start

**FIGURE 18-16: MSSP BLOCK DIAGRAM ($I^2C$™ MASTER MODE)**



#### 18.4.6.1 I²C Master Mode Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/$\overline{W}$) bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1' Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

The Baud Rate Generator used for the SPI mode operation is used to set the SCL clock frequency for either 100 kHz, 400 kHz or 1 MHz I²C operation. See **Section 18.4.7 "Baud Rate"** for more details.

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the Start Enable bit, SEN (SSPCON2<0>).
2. SSPIF is set. The MSSP module will wait the required start time before any other operation takes place.
3. The user loads the SSPBUF with the slave address to transmit.
4. Address is shifted out the SDA pin until all 8 bits are transmitted.
5. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
7. The user loads the SSPBUF with eight bits of data.
8. Data is shifted out the SDA pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
11. The user generates a Stop condition by setting the Stop Enable bit, PEN (SSPCON2<2>).
12. Interrupt is generated once the Stop condition is complete.


© 2009 Microchip Technology Inc.

### 18.4.7 BAUD RATE

In $I^2C$ Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPADD register (Figure 18-17). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (TCY) on the Q2 and Q4 clocks. In $I^2C$ Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

Table 18-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

**FIGURE 18-17: BAUD RATE GENERATOR BLOCK DIAGRAM**

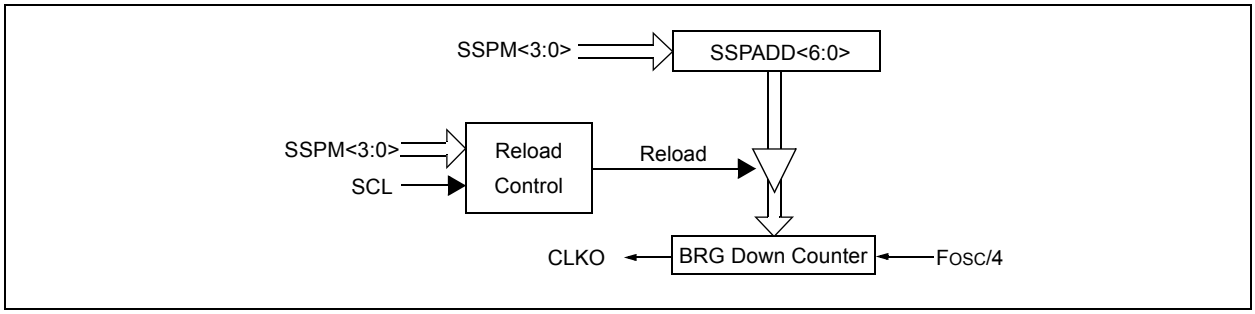


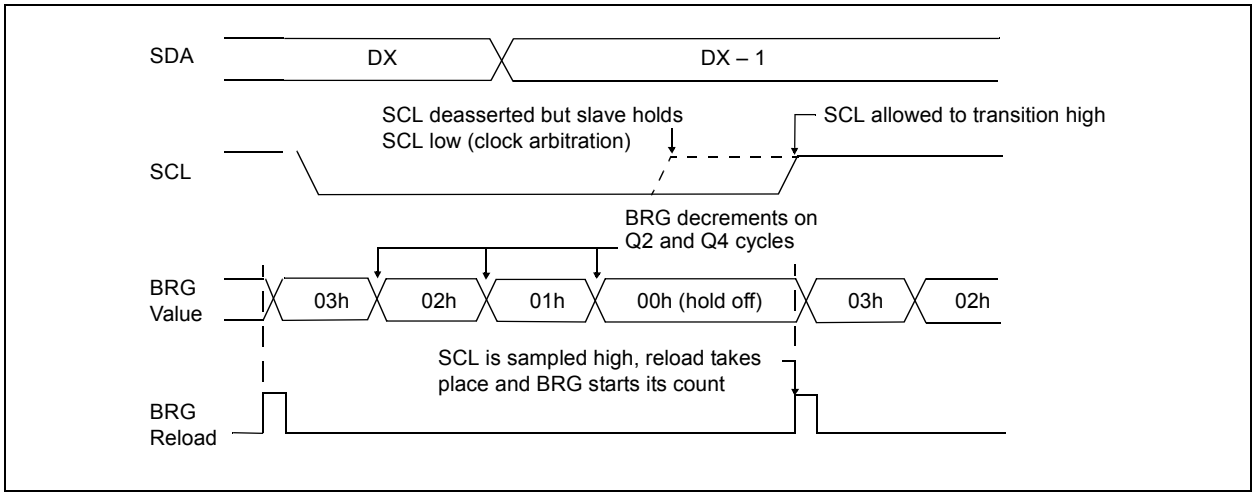

**TABLE 18-3: $I^2C$™ CLOCK RATE W/BRG**

| FCY | FCY*2 | BRG Value | FSCL (2 Rollovers of BRG) |
|---|---|---|---|
| 10 MHz | 20 MHz | 19h | 400 kHz |
| 10 MHz | 20 MHz | 20h | 312.5 kHz |
| 10 MHz | 20 MHz | 64h | 100 kHz |
| 4 MHz | 8 MHz | 0Ah | 400 kHz |
| 4 MHz | 8 MHz | 0Dh | 308 kHz |
| 4 MHz | 8 MHz | 28h | 100 kHz |
| 1 MHz | 2 MHz | 03h | 333 kHz |
| 1 MHz | 2 MHz | 0Ah | 100 kHz |

#### 18.4.7.1 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 18-18).

**FIGURE 18-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**





© 2009 Microchip Technology Inc.

### 18.4.8 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Condition Enable bit, SEN (SSPCON2<0>). If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the S bit (SSPSTAT<3>) to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit (SSPCON2<0>) will be automatically cleared by hardware, the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

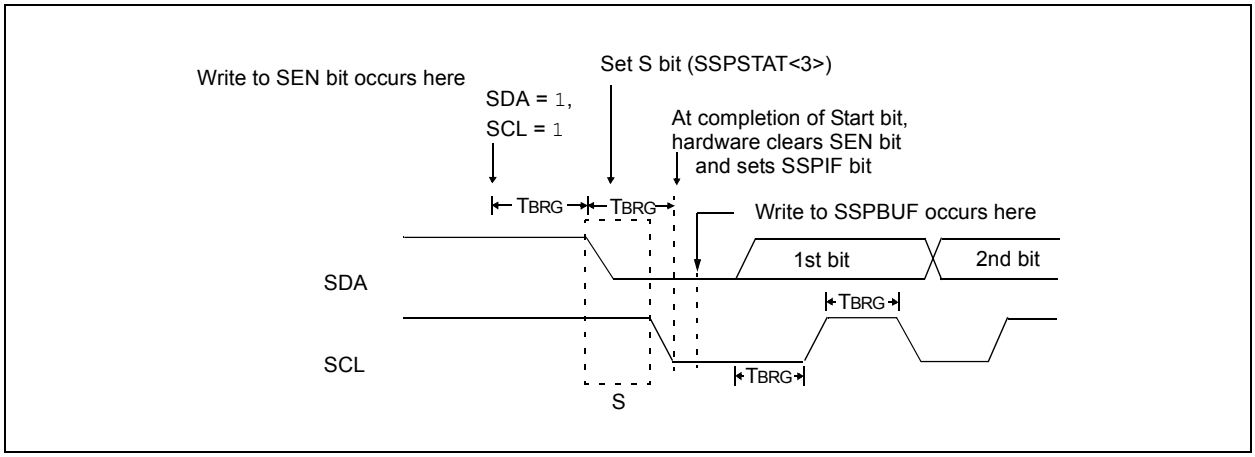
| **Note:** | If, at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state. |
|---|---|

#### 18.4.8.1 WCOL Status Flag

If the user writes the SSPBUF when a Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

| **Note:** | Because queueing of events is not allowed, writing to the lower 5 bits of SSPCON2 is disabled until the Start condition is complete. |
|---|---|

**FIGURE 18-19: FIRST START BIT TIMING**



### 18.4.9 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) is programmed high and the I²C logic module is in the Idle state. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded with the contents of SSPADD<5:0> and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, and if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA = 0) for one TBRG while SCL is high. Following this, the RSEN bit (SSPCON2<1>) will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit (SSPSTAT<3>) will be set. The SSPIF bit will not be set until the Baud Rate Generator has timed out.

**Note 1:** If RSEN is programmed while any other event is in progress, it will not take effect.

**2:** A bus collision during the Repeated Start condition occurs if:

- SDA is sampled low when SCL goes from low-to-high.
- SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data '1'.

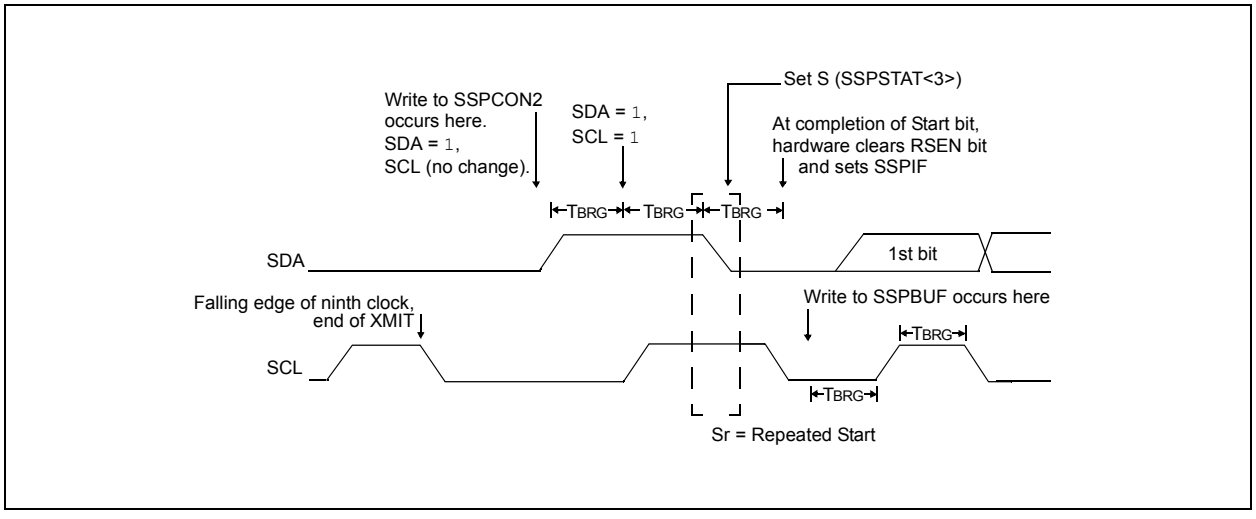
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

#### 18.4.9.1 WCOL Status Flag

If the user writes the SSPBUF when a Repeated Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

**Note:** Because queueing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated Start condition is complete.

**FIGURE 18-20: REPEAT START CONDITION WAVEFORM**





© 2009 Microchip Technology Inc.

### 18.4.10 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPBUF register. This action will set the Buffer Full flag bit, BF, and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted (see data hold time specification parameter 106). SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high (see data setup time specification parameter 107). When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an $\overline{ACK}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{ACK}$ is written into the ACKDT bit on the falling edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge status bit, ACKSTAT, is cleared. If not, the bit is set. After the ninth clock, the SSPIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 18-21).

After the write to the SSPBUF, each bit of address will be shifted out on the falling edge of SCL until all seven address bits and the $\overline{R}/W$ bit are completed. On the falling edge of the eighth clock, the master will deassert the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the $\overline{ACK}$ bit is loaded into the ACKSTAT status bit (SSPCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPIF flag is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

#### 18.4.10.1 BF Status Flag

In Transmit mode, the BF bit (SSPSTAT<0>) is set when the CPU writes to SSPBUF and is cleared when all 8 bits are shifted out.

#### 18.4.10.2 WCOL Status Flag

If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

WCOL must be cleared in software.

#### 18.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPCON2<6>) is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the slave does not Acknowledge ($\overline{ACK}$ = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

### 18.4.11 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPCON2<3>).

| **Note:** | The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded. |
|---|---|

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge sequence enable bit, ACKEN (SSPCON2<4>).

#### 18.4.11.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

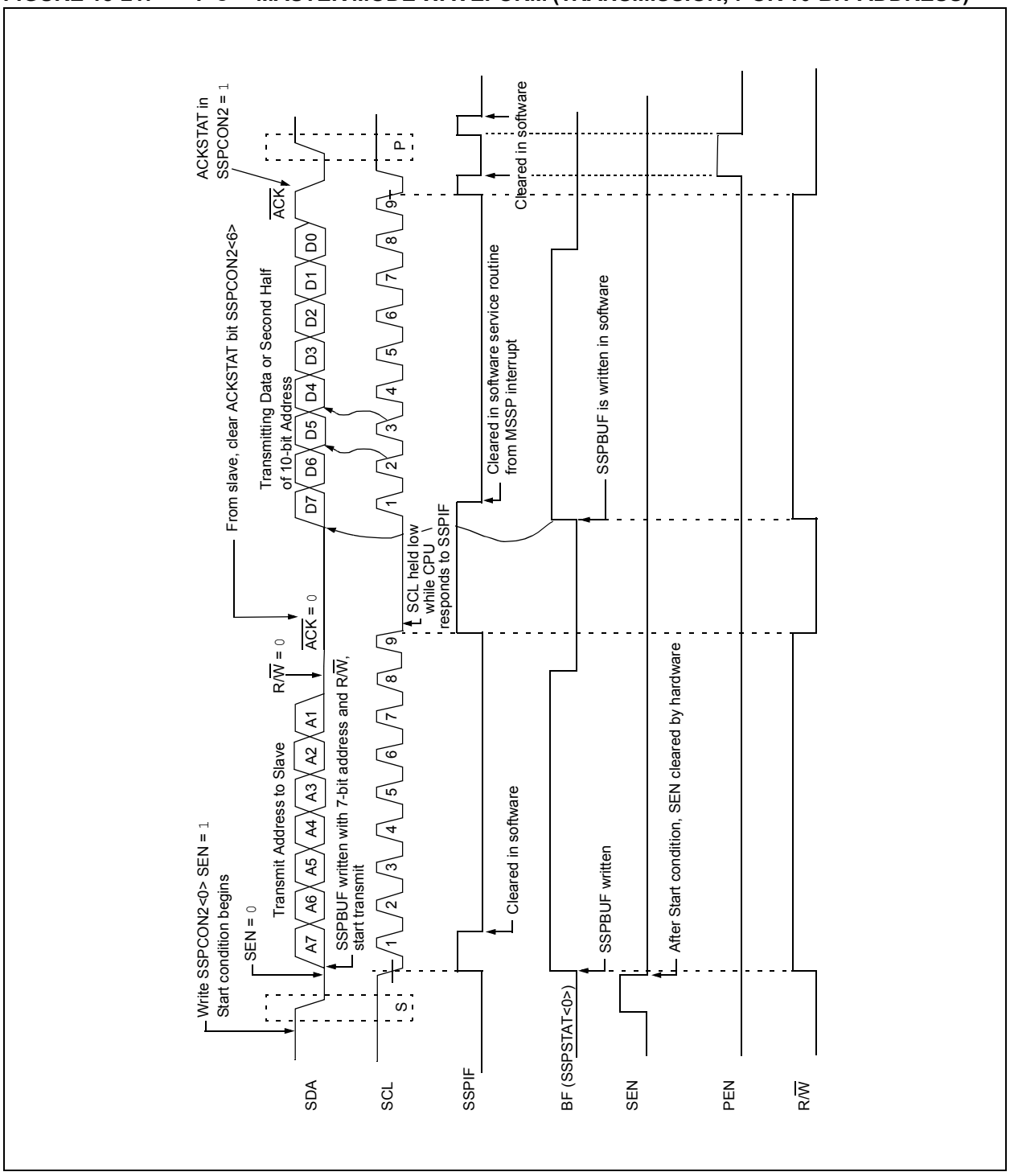
#### 18.4.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

#### 18.4.11.3 WCOL Status Flag

If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

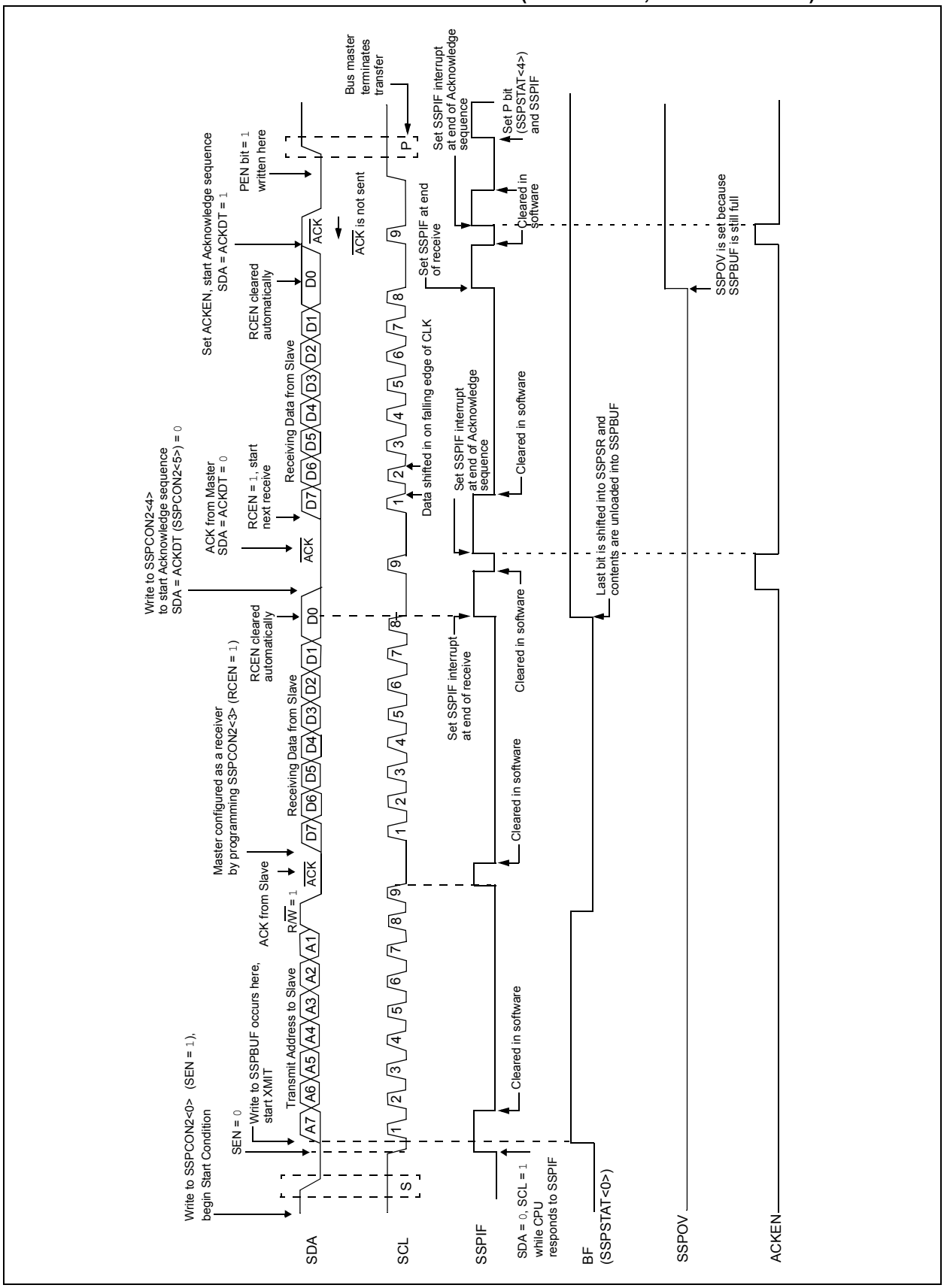
**FIGURE 18-21: I²C™ MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)**





© 2009 Microchip Technology Inc.

**FIGURE 18-22: I²C™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)**



### 18.4.12 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge sequence enable bit, ACKEN (SSPCON2<4>). When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG; the SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into Idle mode (Figure 18-23).

#### 18.4.12.1 WCOL Status Flag

If the user writes the SSPBUF when an Acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

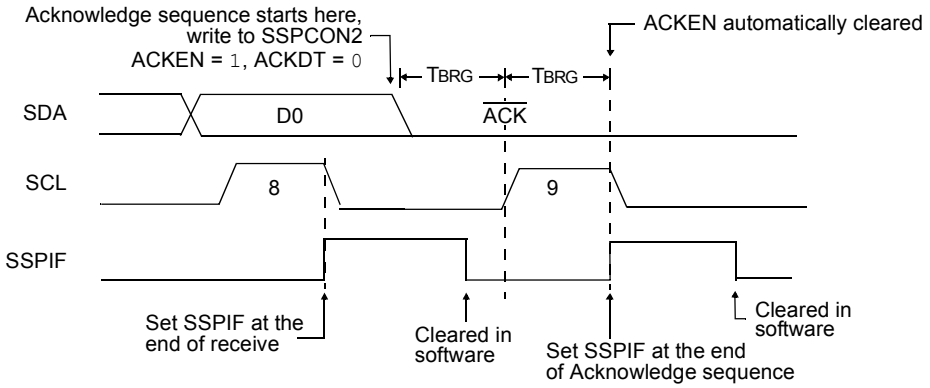
### 18.4.13 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN (SSPCON2<2>). At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to 0. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit (SSPSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 18-24).

#### 18.4.13.1 WCOL Status Flag

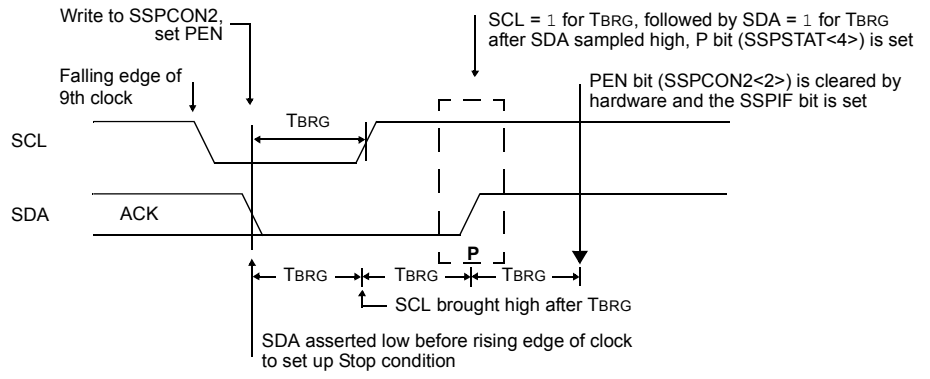
If the user writes the SSPBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 18-23: ACKNOWLEDGE SEQUENCE WAVEFORM**




**Note:** TBRG = one Baud Rate Generator period.

**FIGURE 18-24: STOP CONDITION RECEIVE OR TRANSMIT MODE**




**Note:** TBRG = one Baud Rate Generator period.


© 2009 Microchip Technology Inc.

### 18.4.14 SLEEP OPERATION

While in Sleep mode, the $I^2C$ module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

### 18.4.15 EFFECT OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

### 18.4.16 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the $I^2C$ bus may be taken when the P bit (SSPSTAT<4>) is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the MSSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDA line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed in hardware with the result placed in the BCLIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

### 18.4.17 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high, and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin = 0, then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF, and reset the $I^2C$ port to its Idle state (Figure 18-25).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are deasserted and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the $I^2C$ bus is free, the user can resume communication by asserting a Start condition.

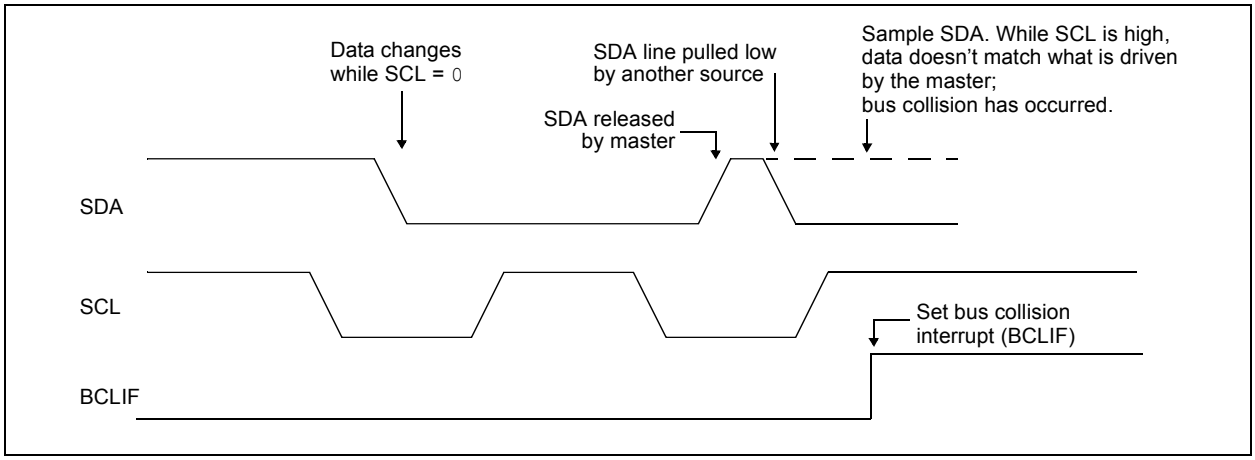
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, and if the $I^2C$ bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the $I^2C$ bus can be taken when the P bit is set in the SSPSTAT register, or the bus is Idle and the S and P bits are cleared.

**FIGURE 18-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**



#### 18.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL is sampled low at the beginning of the Start condition (Figure 18-26).

b) SCL is sampled low before SDA is asserted low (Figure 18-27).

During a Start condition, both the SDA and the SCL pins are monitored.

If the SDA pin is already low, or the SCL pin is already low, then all of the following occur:

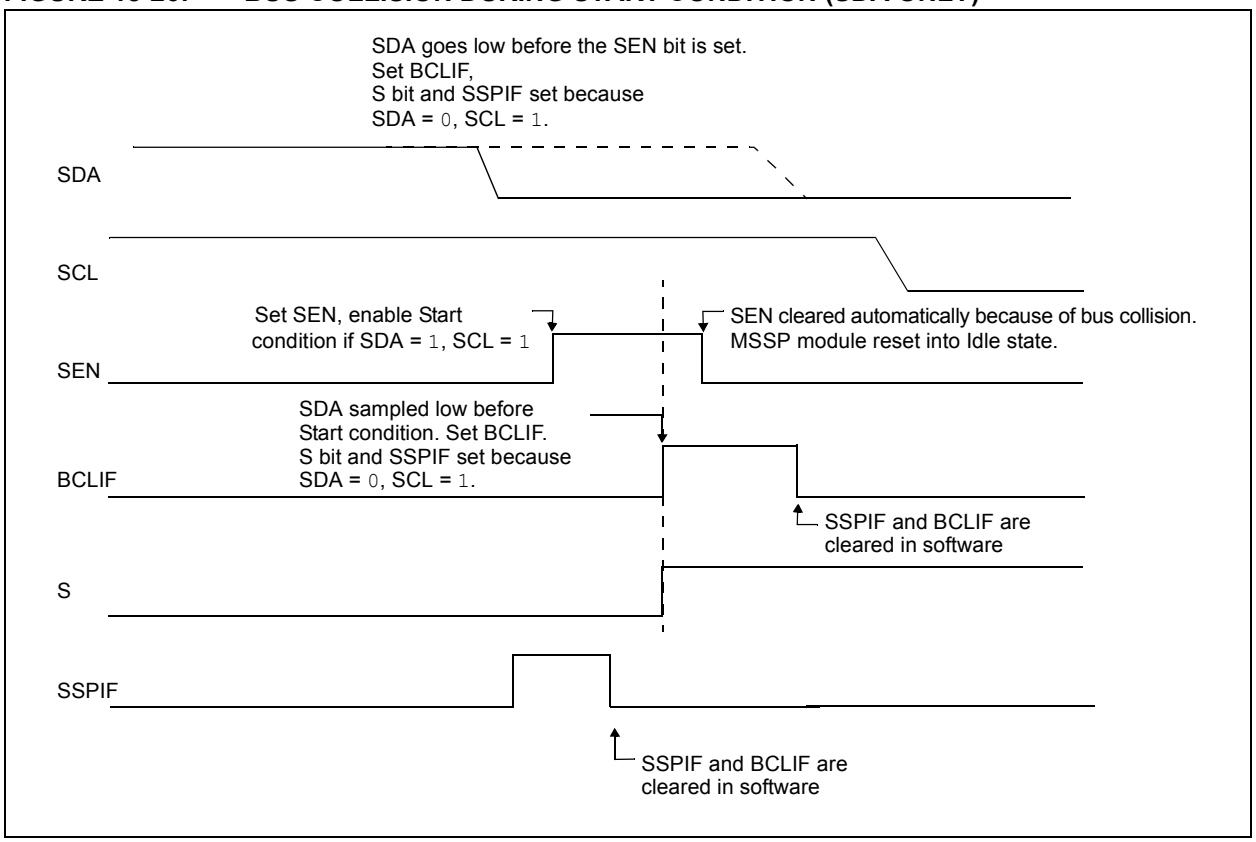
- the Start condition is aborted,
- the BCLIF flag is set; and
- the MSSP module is reset to its Idle state (Figure 18-26)

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded from SSPADD<6:0> and counts down to 0. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 18-28). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to 0 and during this time, if the SCL pins are sampled as '0', a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

| **Note:** | The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions. |
|---|---|

**FIGURE 18-26: BUS COLLISION DURING START CONDITION (SDA ONLY)**





© 2009 Microchip Technology Inc.

**FIGURE 18-27: BUS COLLISION DURING START CONDITION (SCL = 0)**

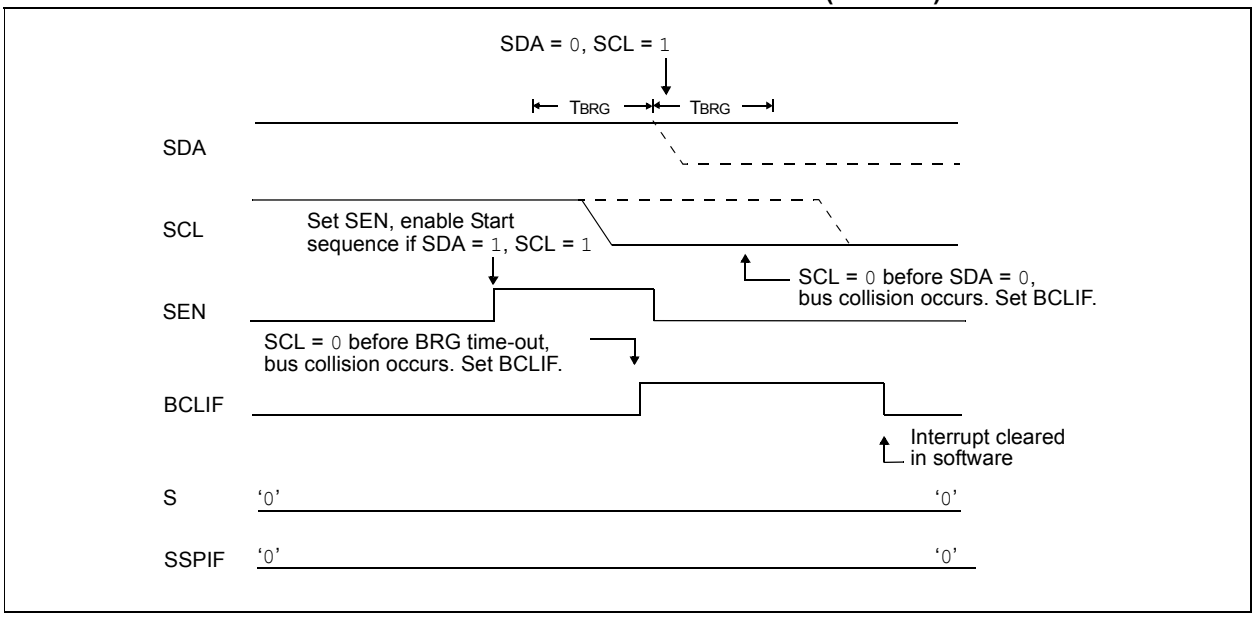
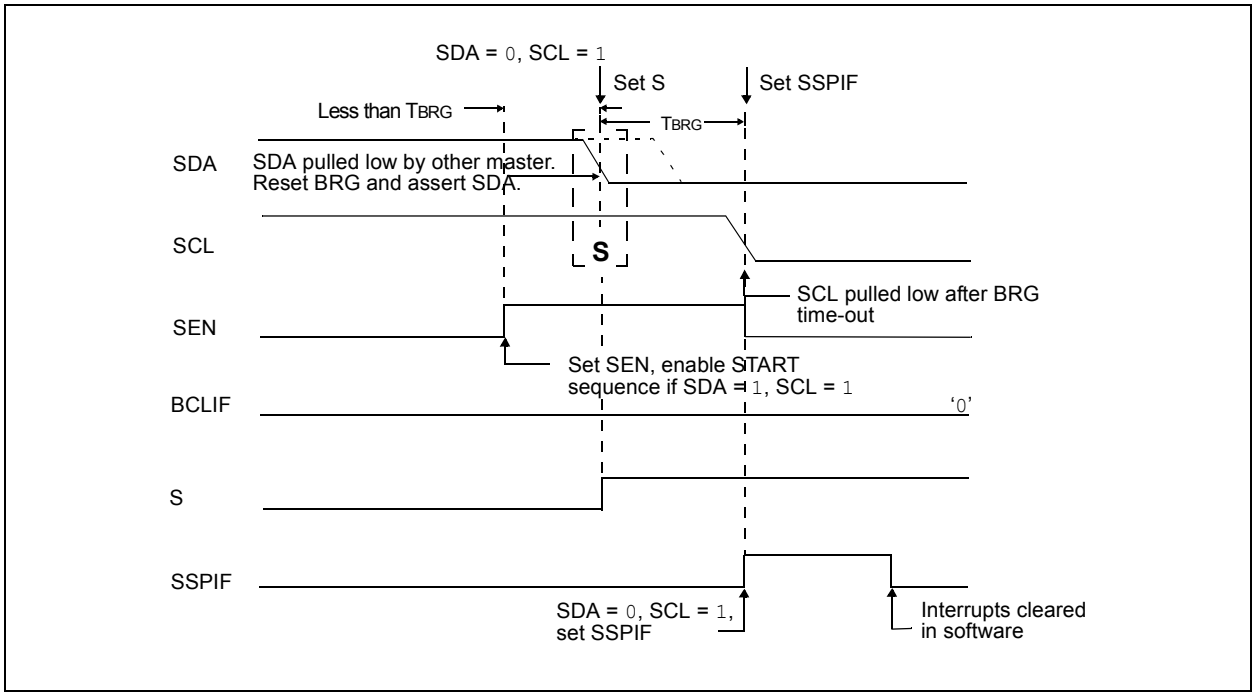



**FIGURE 18-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION**



### 18.4.17.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from a low level to a high level.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to 0. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', see Figure 18-29). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high-to-low before the BRG times out, and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition, see Figure 18-30.

If, at the end of the BRG time-out, both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete.

**FIGURE 18-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**

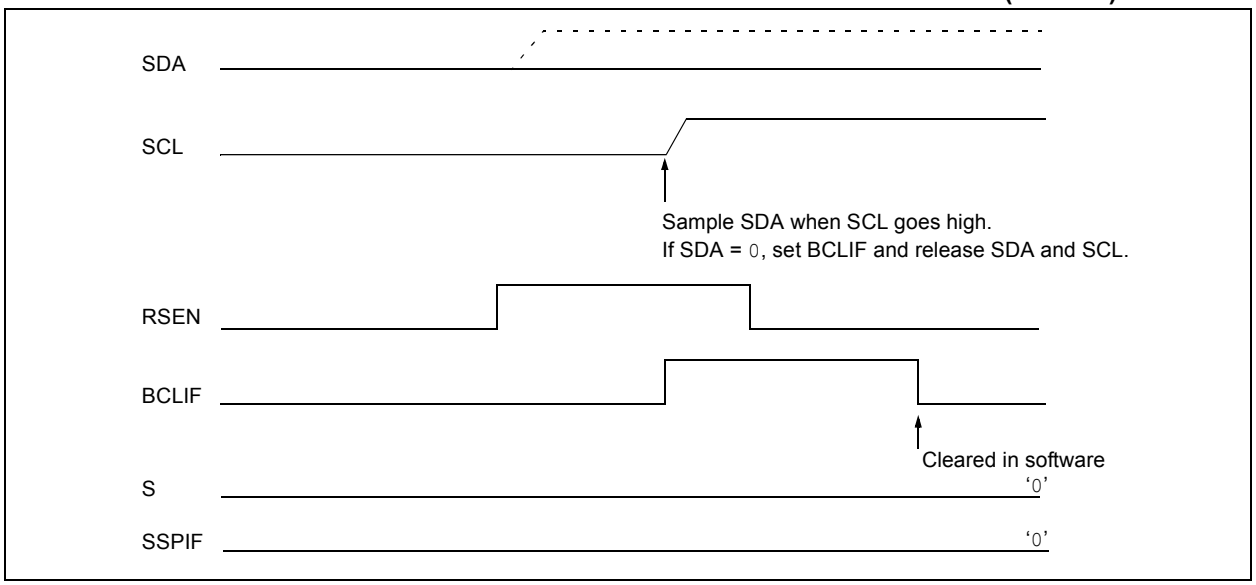
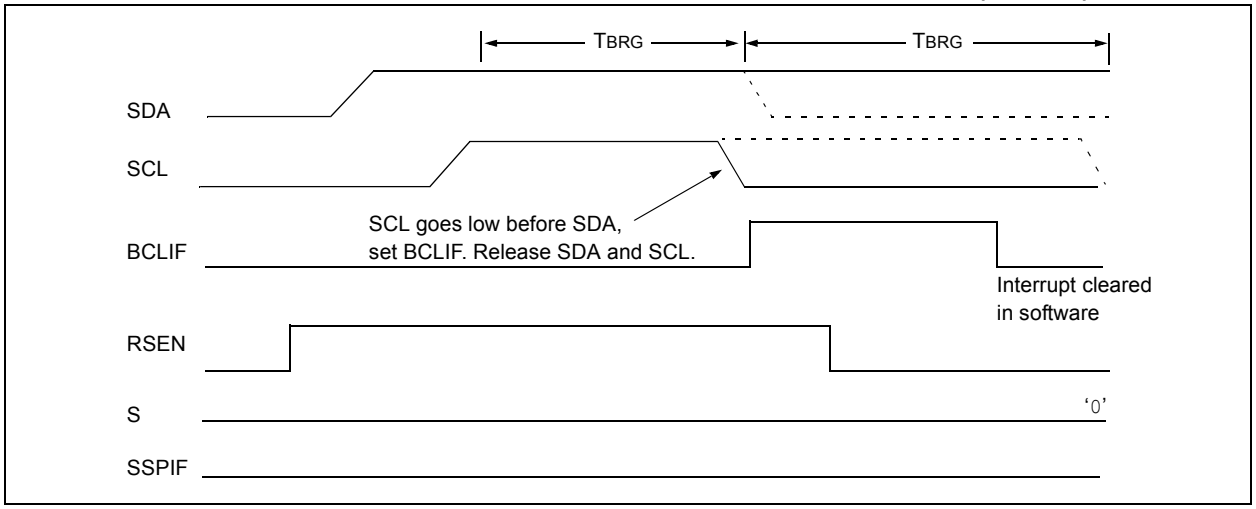



**FIGURE 18-30: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)**





© 2009 Microchip Technology Inc.

### 18.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD<6:0> and counts down to 0. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 18-31). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 18-32).

**FIGURE 18-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

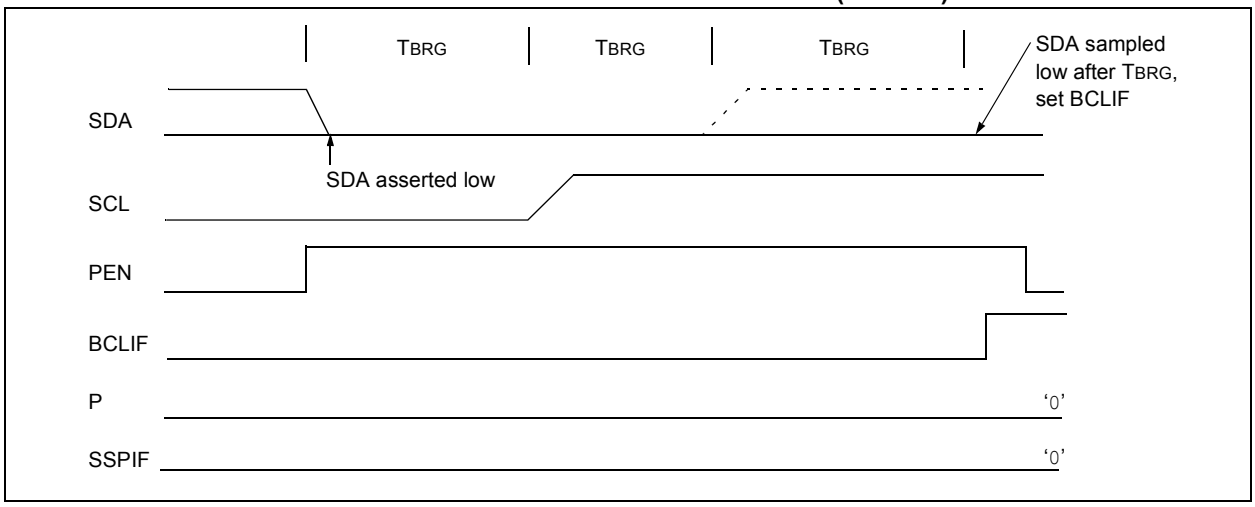
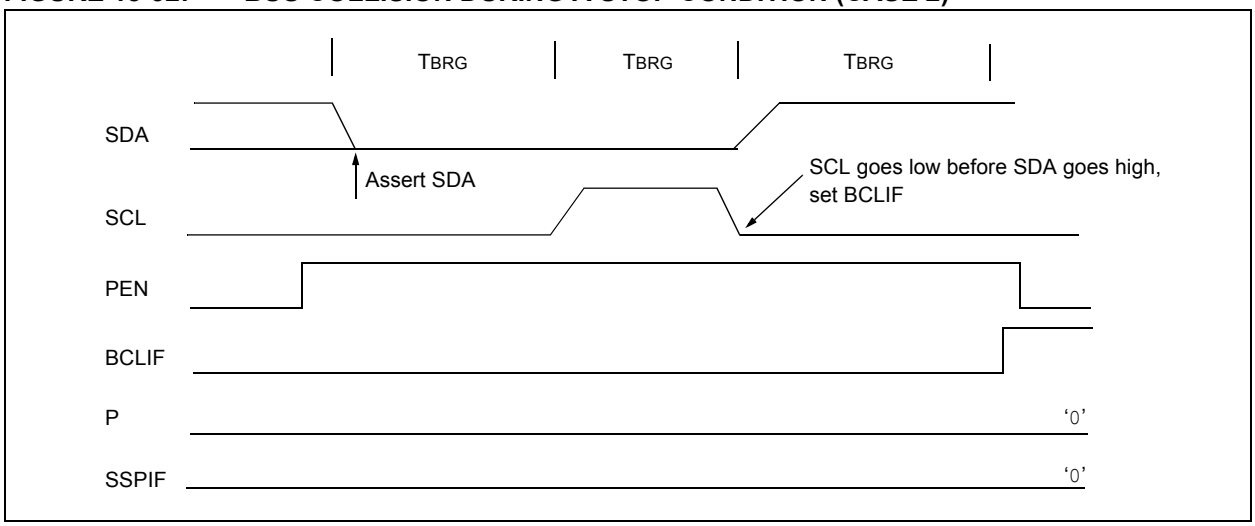



**FIGURE 18-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)**



**NOTES:**


© 2009 Microchip Technology Inc.

## 19.0 ENHANCED UNIVERSAL SYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART 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 and so on.

The EUSART 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 make it ideally suited for use in Local Interconnect Network bus (LIN/J2602 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 pins of the Enhanced USART are multiplexed with PORTC. In order to configure RC6/TX/CK and RC7/RX/DT as a USART:

- bit, SPEN (RCSTA<7>), must be set (= 1)
- bit, TRISC<7>, must be set (= 1)
- bit, TRISC<6>, must be cleared (= 0) for Asynchronous and Synchronous Master modes, or set (= 1) for Synchronous Slave mode

| **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 (BAUDCON)

These are detailed on the following pages in Register 19-1, Register 19-2 and Register 19-3, respectively.

**REGISTER 19-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-1 | R/W-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' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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 empty
0 = TSR full

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.


© 2009 Microchip Technology Inc.

**REGISTER 19-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-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' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-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 an address/data bit or a parity bit and must be calculated by user firmware.

**REGISTER 19-3: BAUDCON: BAUD RATE CONTROL REGISTER**

| R/W-0 | R-1 | U-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **ABDOVF**: Auto-Baud Acquisition Rollover Status bit
1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
0 = No BRG rollover has occurred

bit 6 **RCIDL**: Receive Operation Idle Status bit
1 = Receive operation is Idle
0 = Receive operation is active

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 field (55h); cleared in hardware upon completion.
0 = Baud rate measurement disabled or completed
Synchronous mode:
Unused in this mode.


© 2009 Microchip Technology Inc.

## 19.1 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 (BAUDCON<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 (BAUDCON<3>) also control the baud rate. In Synchronous mode, BRGH is ignored. Table 19-1 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 19-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 19-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 19-2. 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.

| **Note:** | BRG value of '0' is not supported. |
|---|---|

### 19.1.1 OPERATION IN POWER-MANAGED MODES

The device clock is used to generate the desired baud rate. When one of the power-managed modes is entered, the new clock source may be operating at a different frequency. This may require an adjustment to the value in the SPBRG register pair.

### 19.1.2 SAMPLING

The data on the RX 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 when SYNC is clear or when BRG16 and BRGH are both not set. The data on the RX pin is sampled once when SYNC is set or when BRGH16 and BRGH are both set.

**TABLE 19-1: 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 19-1: 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%

**TABLE 19-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 57 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 57 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 57 |
| SPBRGH | EUSART Baud Rate Generator Register High Byte | | | | | | | | 57 |
| SPBRG | EUSART Baud Rate Generator Register Low Byte | | | | | | | | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the BRG.


© 2009 Microchip Technology Inc.

**TABLE 19-3: 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 | 1.201 | -0.16 | 103 |
| 2.4 | 2.441 | 1.73 | 255 | 2.404 | 0.16 | 129 | 2.404 | 0.16 | 64 | 2.403 | -0.16 | 51 |
| 9.6 | 9.615 | 0.16 | 64 | 9.766 | 1.73 | 31 | 9.766 | 1.73 | 15 | 9.615 | -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 | 0.300 | -0.16 | 103 | 0.300 | -0.16 | 51 |
| 1.2 | 1.202 | 0.16 | 51 | 1.201 | -0.16 | 25 | 1.201 | -0.16 | 12 |
| 2.4 | 2.404 | 0.16 | 25 | 2.403 | -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 | — | — | — | — | — | — |

| 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) |
| 0.3 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1.2 | — | — | — | — | — | — | — | — | — | — | — | — |
| 2.4 | — | — | — | — | — | — | 2.441 | 1.73 | 255 | 2.403 | -0.16 | 207 |
| 9.6 | 9.766 | 1.73 | 255 | 9.615 | 0.16 | 129 | 9.615 | 0.16 | 64 | 9.615 | -0.16 | 51 |
| 19.2 | 19.231 | 0.16 | 129 | 19.231 | 0.16 | 64 | 19.531 | 1.73 | 31 | 19.230 | -0.16 | 25 |
| 57.6 | 58.140 | 0.94 | 42 | 56.818 | -1.36 | 21 | 56.818 | -1.36 | 10 | 55.555 | 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 | — | — | — | — | — | — | 0.300 | -0.16 | 207 |
| 1.2 | 1.202 | 0.16 | 207 | 1.201 | -0.16 | 103 | 1.201 | -0.16 | 51 |
| 2.4 | 2.404 | 0.16 | 103 | 2.403 | -0.16 | 51 | 2.403 | -0.16 | 25 |
| 9.6 | 9.615 | 0.16 | 25 | 9.615 | -0.16 | 12 | — | — | — |
| 19.2 | 19.231 | 0.16 | 12 | — | — | — | — | — | — |
| 57.6 | 62.500 | 8.51 | 3 | — | — | — | — | — | — |
| 115.2 | 125.000 | 8.51 | 1 | — | — | — | — | — | — |

**TABLE 19-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)**

| 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 | 0.300 | -0.04 | 1665 |
| 1.2 | 1.200 | 0.02 | 2082 | 1.200 | -0.03 | 1041 | 1.200 | -0.03 | 520 | 1.201 | -0.16 | 415 |
| 2.4 | 2.402 | 0.06 | 1040 | 2.399 | -0.03 | 520 | 2.404 | 0.16 | 259 | 2.403 | -0.16 | 207 |
| 9.6 | 9.615 | 0.16 | 259 | 9.615 | 0.16 | 129 | 9.615 | 0.16 | 64 | 9.615 | -0.16 | 51 |
| 19.2 | 19.231 | 0.16 | 129 | 19.231 | 0.16 | 64 | 19.531 | 1.73 | 31 | 19.230 | -0.16 | 25 |
| 57.6 | 58.140 | 0.94 | 42 | 56.818 | -1.36 | 21 | 56.818 | -1.36 | 10 | 55.555 | 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 | 0.300 | -0.16 | 415 | 0.300 | -0.16 | 207 |
| 1.2 | 1.202 | 0.16 | 207 | 1.201 | -0.16 | 103 | 1.201 | -0.16 | 51 |
| 2.4 | 2.404 | 0.16 | 103 | 2.403 | -0.16 | 51 | 2.403 | -0.16 | 25 |
| 9.6 | 9.615 | 0.16 | 25 | 9.615 | -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 = 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 | 0.300 | -0.01 | 6665 |
| 1.2 | 1.200 | 0.00 | 8332 | 1.200 | 0.02 | 4165 | 1.200 | 0.02 | 2082 | 1.200 | -0.04 | 1665 |
| 2.4 | 2.400 | 0.02 | 4165 | 2.400 | 0.02 | 2082 | 2.402 | 0.06 | 1040 | 2.400 | -0.04 | 832 |
| 9.6 | 9.606 | 0.06 | 1040 | 9.596 | -0.03 | 520 | 9.615 | 0.16 | 259 | 9.615 | -0.16 | 207 |
| 19.2 | 19.193 | -0.03 | 520 | 19.231 | 0.16 | 259 | 19.231 | 0.16 | 129 | 19.230 | -0.16 | 103 |
| 57.6 | 57.803 | 0.35 | 172 | 57.471 | -0.22 | 86 | 58.140 | 0.94 | 42 | 57.142 | 0.79 | 34 |
| 115.2 | 114.943 | -0.22 | 86 | 116.279 | 0.94 | 42 | 113.636 | -1.36 | 21 | 117.647 | -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 | 0.300 | -0.04 | 1665 | 0.300 | -0.04 | 832 |
| 1.2 | 1.200 | 0.04 | 832 | 1.201 | -0.16 | 415 | 1.201 | -0.16 | 207 |
| 2.4 | 2.404 | 0.16 | 415 | 2.403 | -0.16 | 207 | 2.403 | -0.16 | 103 |
| 9.6 | 9.615 | 0.16 | 103 | 9.615 | -0.16 | 51 | 9.615 | -0.16 | 25 |
| 19.2 | 19.231 | 0.16 | 51 | 19.230 | -0.16 | 25 | 19.230 | -0.16 | 12 |
| 57.6 | 58.824 | 2.12 | 16 | 55.555 | 3.55 | 8 | — | — | — |
| 115.2 | 111.111 | -3.55 | 8 | — | — | — | — | — | — |


© 2009 Microchip Technology Inc.

### 19.1.3 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 19-1) 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 Rate Detection must receive a byte with the value 55h (ASCII “U”, which is also the LIN/J2602 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 totalling the proper BRG period is left in the SPBRGH:SPBRG register pair. Once the 5th edge is seen (this should correspond to the Stop bit), the ABDEN bit is automatically cleared.

If a rollover of the BRG occurs (an overflow from FFFFh to 0000h), the event is trapped by the ABDOVF status bit (BAUDCON<7>). It is set in hardware by BRG rollovers and can be set or cleared by the user in software. ABD mode remains active after rollover events and the ABDEN bit remains set (Figure 19-2).

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. Note that the BRG clock can be configured by the BRG16 and BRGH bits. The BRG16 bit must be set to use both SPBRG1 and SPBRGH1 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 19-4 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. The contents of RCREG should be discarded.

**Note 1:** If the WUE bit is set with the ABDEN bit, Auto-Baud Rate Detection will occur on the byte *following* the Break character.

**2:** 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.

**3:** To maximize baud rate range, it is recommended to set the BRG16 bit if the auto-baud feature is used.

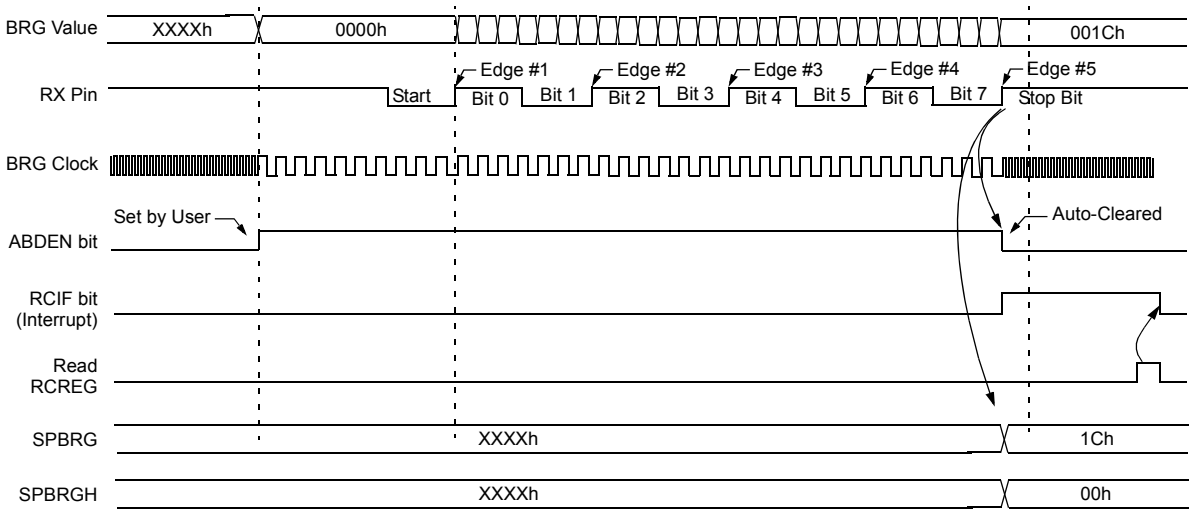
**TABLE 19-4: BRG COUNTER CLOCK RATES**

| BRG16 | BRGH | BRG Counter Clock |
|---|---|---|
| 0 | 0 | FOSC/512 |
| 0 | 1 | FOSC/128 |
| 1 | 0 | FOSC/128 |
| 1 | 1 | FOSC/32 |

#### 19.1.3.1 ABD and EUSART Transmission

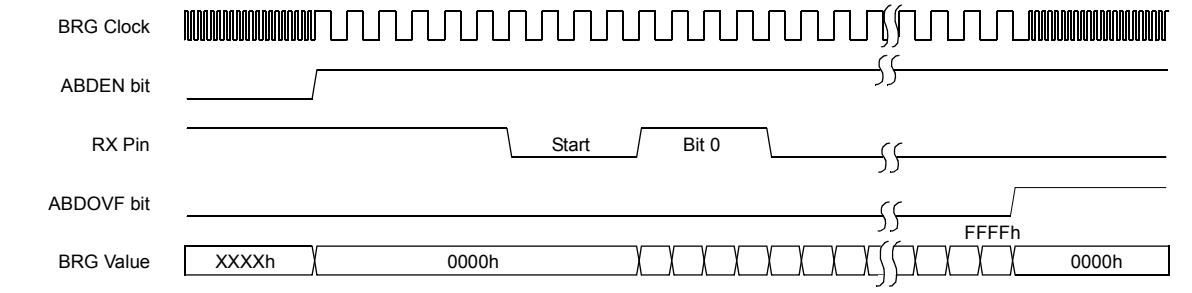
Since the BRG clock is reversed during ABD acquisition, the EUSART transmitter cannot be used during ABD. This means that whenever the ABDEN bit is set, TXREG cannot be written to. Users should also ensure that ABDEN does not become set during a transmit sequence. Failing to do this may result in unpredictable EUSART operation.

**FIGURE 19-1: AUTOMATIC BAUD RATE CALCULATION**




**Note:** The ABD sequence requires the EUSART module to be configured in Asynchronous mode and WUE = 0.

**FIGURE 19-2: BRG OVERFLOW SEQUENCE**





© 2009 Microchip Technology Inc.

## 19.2 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 8 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 BAUDCON<3>). Parity is not supported by the hardware, but can be implemented in software and stored as the 9th data bit.

When operating in Asynchronous mode, the EUSART module consists of the following important elements:

In Asynchronous mode, clock polarity is selected with the TXCKP bit (BAUDCON<4>). Setting TXCKP sets the Idle state on CK as high, while clearing the bit sets the Idle state as low. Data polarity is selected with the RXDTP bit (BAUDCON<5>).

Setting RXDTP inverts data on RX, while clearing the bit has no affect on received data.

- 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

### 19.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 19-3. 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 the TXIF flag bit (PIR1<4>) is set. This interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXIE (PIE1<4>). TXIF will be set regardless of the state of TXIE; it cannot be cleared in software. TXIF is also not cleared immediately upon loading TXREG, but becomes valid in the second instruction cycle following the load instruction. Polling TXIF immediately following a load of TXREG will return invalid results.

While 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 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).
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 19-3: EUSART TRANSMIT BLOCK DIAGRAM**

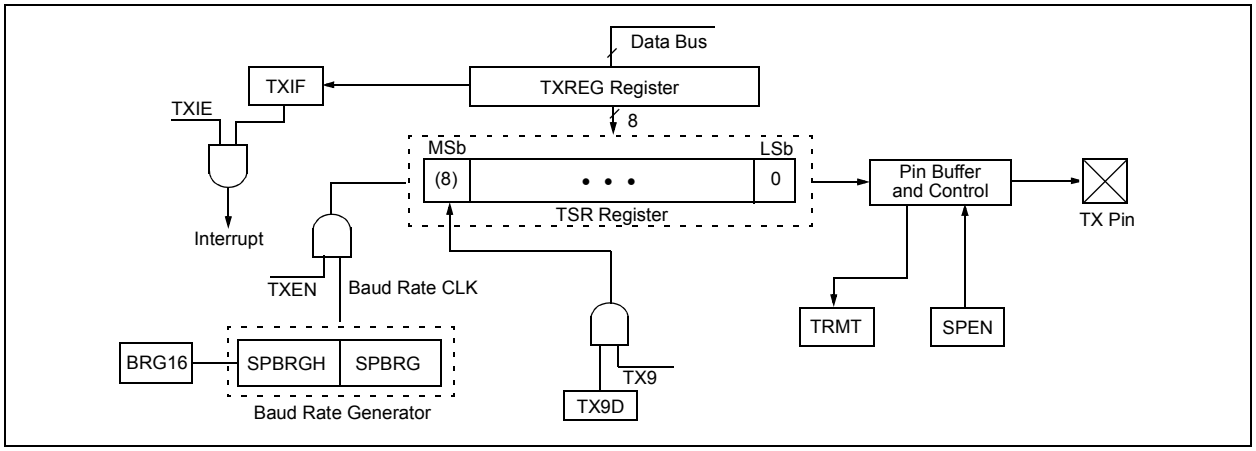



**FIGURE 19-4: ASYNCHRONOUS TRANSMISSION**

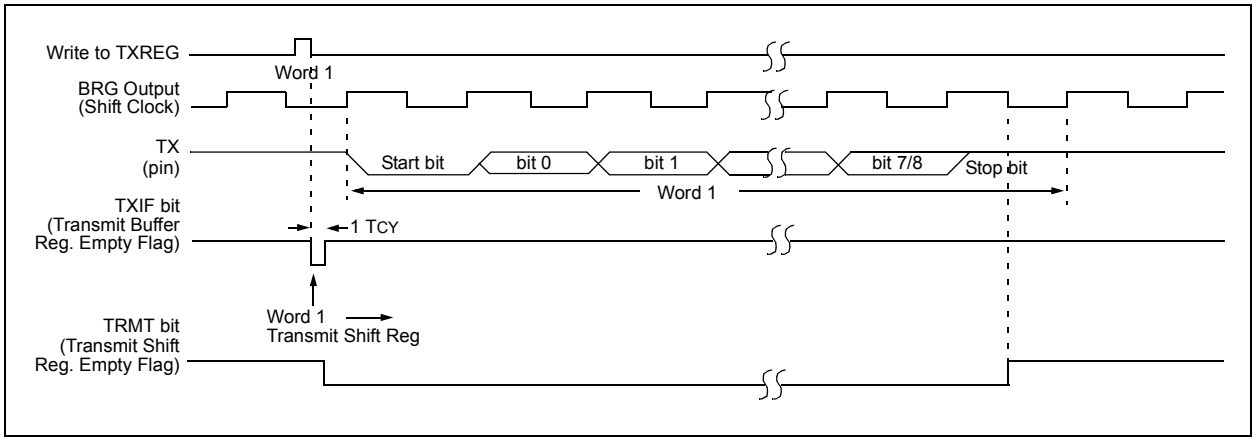
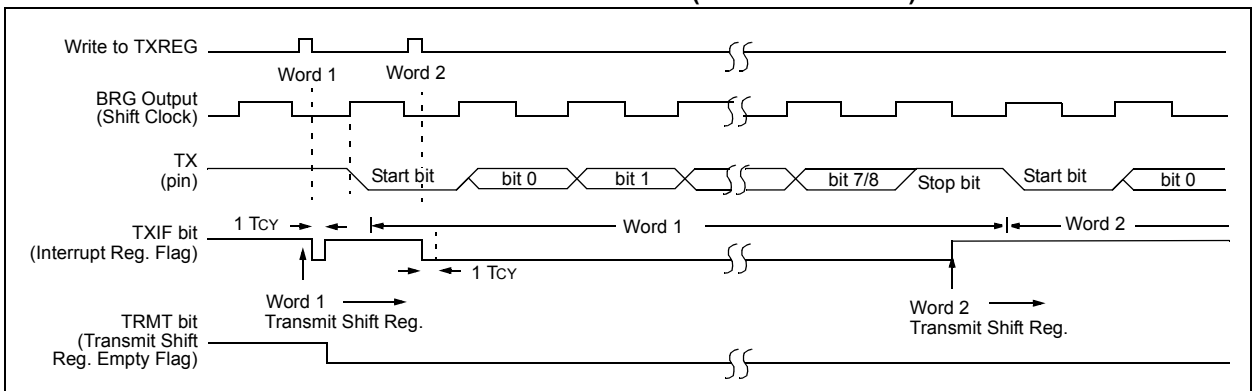



**FIGURE 19-5: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**




**Note:** This timing diagram shows two consecutive transmissions.


© 2009 Microchip Technology Inc.

**TABLE 19-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 57 |
| TXREG | EUSART Transmit Register | | | | | | | | 57 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 57 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 57 |
| SPBRGH | EUSART Baud Rate Generator Register High Byte | | | | | | | | 57 |
| SPBRG | EUSART Baud Rate Generator Register Low Byte | | | | | | | | 57 |

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

**Note 1:** Reserved in PIC18F2X80 devices; always maintain these bits clear.

### 19.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 19-6. The data is received on the RX 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:

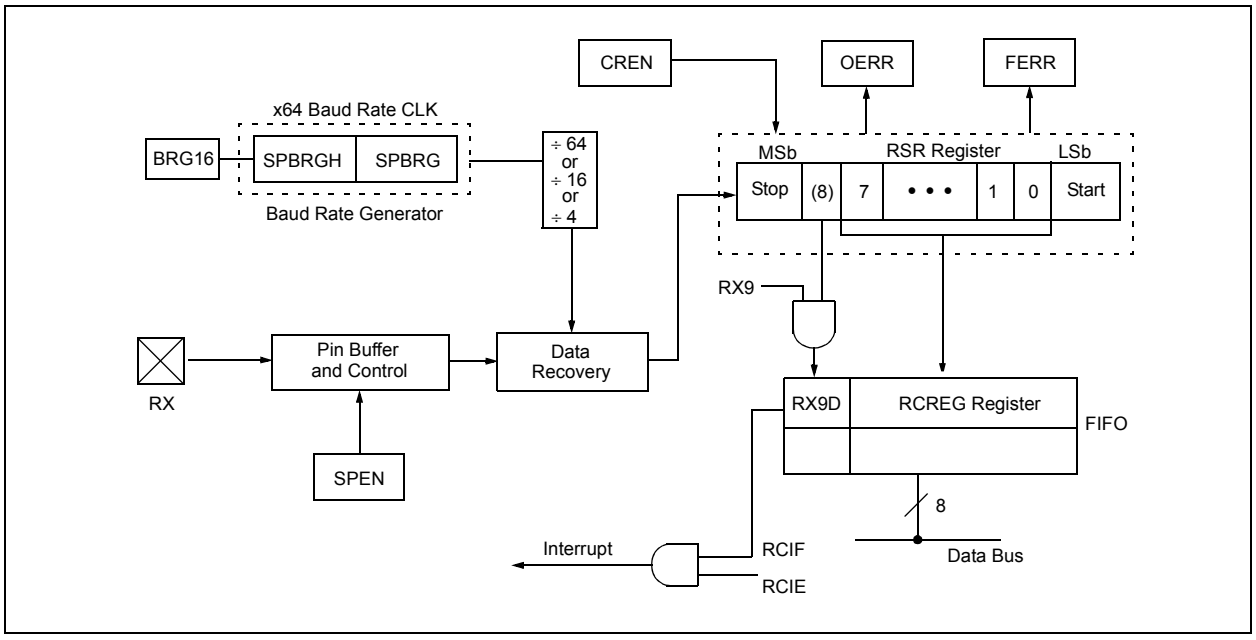
1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RCIE.
4. If 9-bit 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.

### 19.2.3 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.

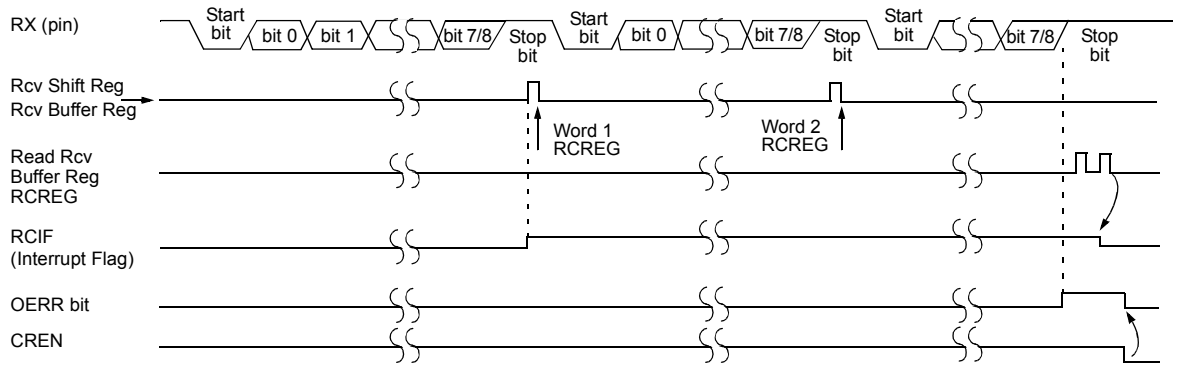
**FIGURE 19-6: EUSART RECEIVE BLOCK DIAGRAM**





© 2009 Microchip Technology Inc.

**FIGURE 19-7: 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 19-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 57 |
| RCREG | EUSART Receive Register | | | | | | | | 57 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 57 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 57 |
| SPBRGH | EUSART Baud Rate Generator Register, High Byte | | | | | | | | 57 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 57 |

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

**Note 1:** Reserved in PIC18F2X80 devices; always maintain these bits clear.

### 19.2.4 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 (BAUDCON<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/J2602 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 19-8) and asynchronously, if the device is in Sleep mode (Figure 19-9). 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.

#### 19.2.4.1 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 (EOC) 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 bits) for standard RS-232 devices or 000h (12 bits) for LIN/J2602 bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., XT or HS mode). The Sync Break (or Wake-up Signal) character must be of sufficient length and be followed by a sufficient interval to allow enough time for the selected oscillator to start and provide proper initialization of the EUSART.

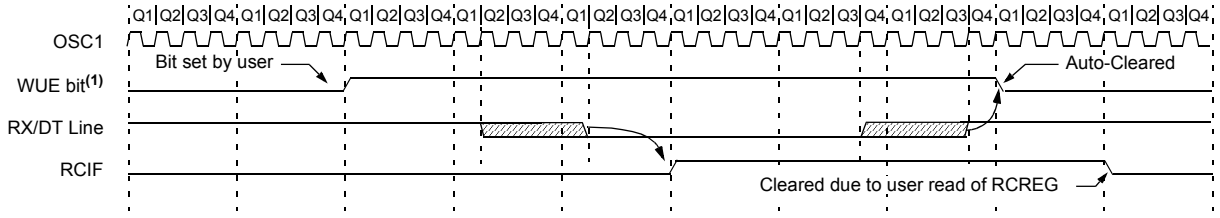
#### 19.2.4.2 Special Considerations Using the WUE Bit

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.

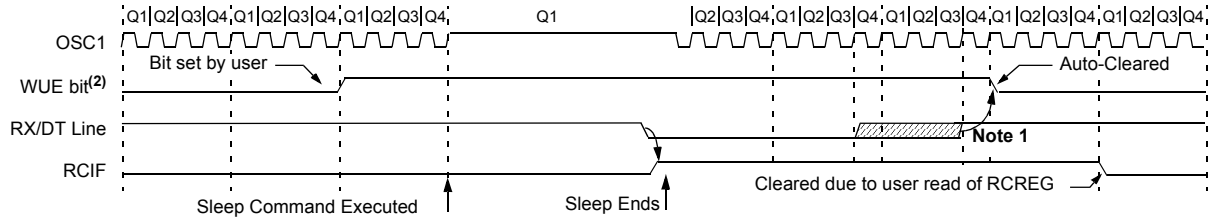
To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**FIGURE 19-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION**




**Note 1:** The EUSART remains in Idle while the WUE bit is set.

**FIGURE 19-9: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**




**Note 1:** If the wake-up event requires long oscillator warm-up time, the auto-clear of the WUE bit can occur while the *stposc* signal is still active. This sequence should not depend on the presence of Q clocks.

**2:** The EUSART remains in Idle while the WUE bit is set.


© 2009 Microchip Technology Inc.

### 19.2.5 BREAK CHARACTER SEQUENCE

The Enhanced EUSART module has the capability of sending the special Break character sequences that are required by the LIN/J2602 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/J2602 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 19-10 for the timing of the Break character sequence.

#### 19.2.5.1 Break and Sync Transmit Sequence

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/J2602 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.

### 19.2.6 RECEIVING A BREAK CHARACTER

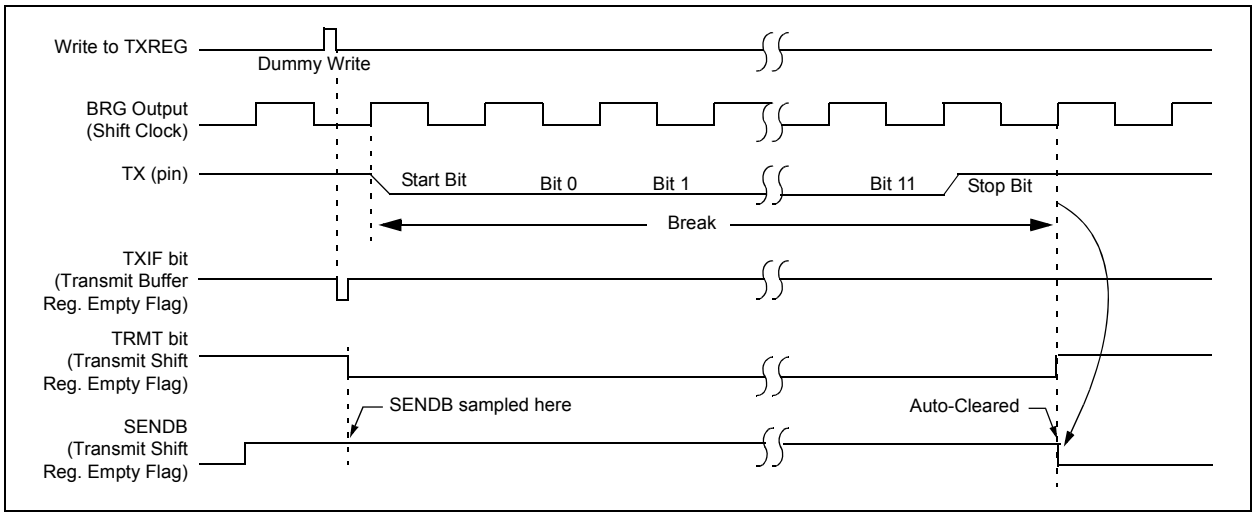
The Enhanced USART module can receive a Break character in two ways.

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 (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 19.2.4 "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 once the TXIF interrupt is observed.

**FIGURE 19-10: SEND BREAK CHARACTER SEQUENCE**



## 19.3 EUSART Synchronous Master Mode

The Master mode indicates that the processor transmits the master clock on the CK line. 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 TX and RX 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 (BAUDCON<4>). 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.

### 19.3.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in Figure 19-3. 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).

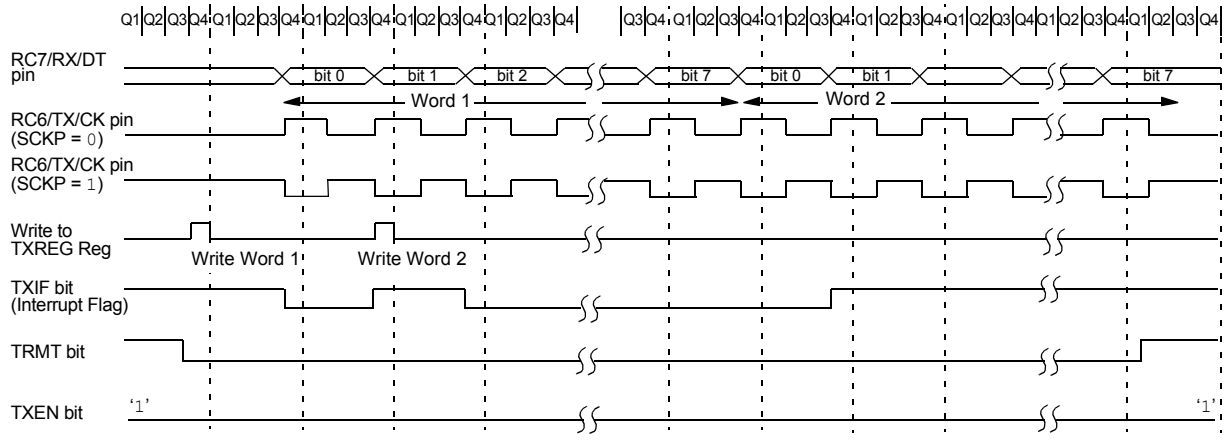
Once the TXREG register transfers the data to the TSR register (occurs in one TCYCLE), the TXREG is empty and the TXIF flag bit (PIR1<4>) is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXIE (PIE1<4>). TXIF is set regardless of the state of enable bit TXIE; it 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 BRG16 bit, 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 19-11: SYNCHRONOUS TRANSMISSION**




**Note:** Sync Master mode, SPBRG = 0, continuous transmission of two 8-bit words.


© 2009 Microchip Technology Inc.

**FIGURE 19-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**

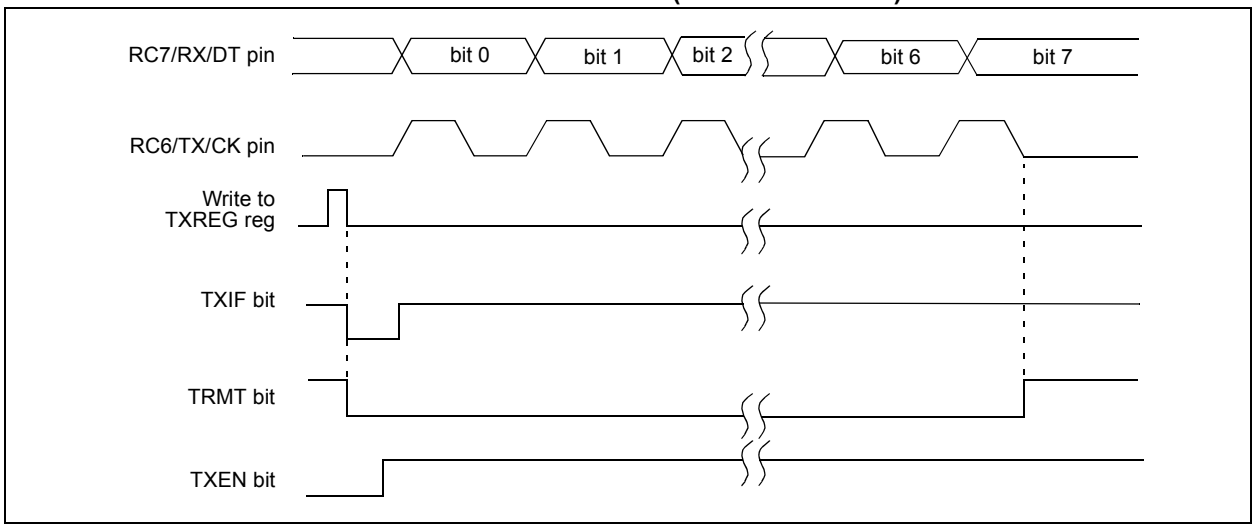



**TABLE 19-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 57 |
| TXREG | EUSART Transmit Register | | | | | | | | 57 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 57 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 57 |
| SPBRGH | EUSART Baud Rate Generator Register, High Byte | | | | | | | | 57 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

**Note 1:** Reserved in PIC18F2X80 devices; always maintain these bits clear.

### 19.3.2 EUSART SYNCHRONOUS MASTER RECEPTION

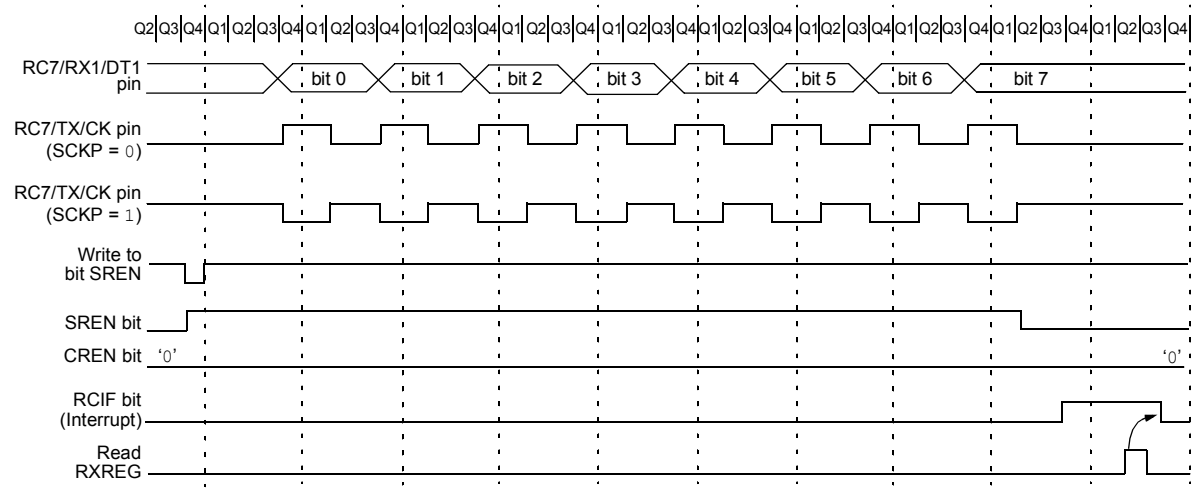
Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTA<5>), or the Continuous Receive Enable bit, CREN (RCSTA<4>). Data is sampled on the RX pin on the falling edge of the clock.

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 BRG16 bit, 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 9th 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 19-13: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**




**Note:** Timing diagram demonstrates Sync Master mode with bit, SREN = 1, and bit, BRGH = 0.

**TABLE 19-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 57 |
| RCREG | EUSART Receive Register | | | | | | | | 57 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 57 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 57 |
| SPBRGH | EUSART Baud Rate Generator Register High Byte | | | | | | | | 57 |
| SPBRG | EUSART Baud Rate Generator Register Low Byte | | | | | | | | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.

**Note 1:** Reserved in PIC18F2X80 devices; always maintain these bits clear.


© 2009 Microchip Technology Inc.

## 19.4 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 CK pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

### 19.4.1 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 TXREGx register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 19-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 57 |
| TXREG | EUSART Transmit Register | | | | | | | | 57 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 57 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 57 |
| SPBRGH | EUSART Baud Rate Generator Register High Byte | | | | | | | | 57 |
| SPBRG | EUSART Baud Rate Generator Register Low Byte | | | | | | | | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

**Note 1:** Reserved in PIC18F2X80 devices; always maintain these bits clear.

### 19.4.2 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 the 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 9th 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 19-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 57 |
| RCREG | EUSART Receive Register | | | | | | | | 57 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 57 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 57 |
| SPBRGH | EUSART Baud Rate Generator Register High Byte | | | | | | | | 57 |
| SPBRG | EUSART Baud Rate Generator Register Low Byte | | | | | | | | 57 |

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.

**Note 1:** Reserved in PIC18F2X80 devices; always maintain these bits clear.


© 2009 Microchip Technology Inc.

## 20.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has 8 inputs for the PIC18F2X80 devices and 11 for the PIC18F4X80 devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

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 20-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 20-2, configures the functions of the port pins. The ADCON2 register, shown in Register 20-3, configures the A/D clock source, programmed acquisition time and justification.

**REGISTER 20-1: ADCON0: A/D CONTROL REGISTER 0**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | CHS3 | 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' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-2 **CHS<3:0>:** Analog Channel Select bits

0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)**(1,2)**
0110 = Channel 6 (AN6)**(1,2)**
0111 = Channel 7 (AN7)**(1,2)**
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Unused
1100 = Unused
1101 = Unused
1110 = Unused
1111 = Unused

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

**Note 1:** These channels are not implemented on PIC18F2X80 devices.
**2:** Performing a conversion on unimplemented channels will return full-scale measurements.

**REGISTER 20-2: ADCON1: A/D CONTROL REGISTER 1**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0(1) | R/W-q(1) | R/W-q(1) | R/W-q(1) |
|---|---|---|---|---|---|---|---|
| — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **VCFG1:** Voltage Reference Configuration bit ($V_{REF-}$ source)
1 = $V_{REF-}$ (AN2)
0 = $AV_{SS}$

bit 4 **VCFG0:** Voltage Reference Configuration bit ($V_{REF+}$ source)
1 = $V_{REF+}$ (AN3)
0 = $AV_{DD}$

bit 3-0 **PCFG<3:0>:** A/D Port Configuration Control bits:

| PCFG<3:0> | AN10 | AN9 | AN8 | AN7(2) | AN6(2) | AN5(2) | AN4 | AN3 | AN2 | AN1 | AN0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000(1) | A | A | A | A | A | A | A | A | A | A | A |
| 0001 | A | A | A | A | A | A | A | A | A | A | A |
| 0010 | A | A | A | A | A | A | A | A | A | A | A |
| 0011 | A | A | A | A | A | A | A | A | A | A | A |
| 0100 | A | A | A | A | A | A | A | A | A | A | A |
| 0101 | D | A | A | A | A | A | A | A | A | A | A |
| 0110 | D | D | A | A | A | A | A | A | A | A | A |
| 0111(1) | D | D | D | A | A | A | A | A | A | A | A |
| 1000 | D | D | D | D | A | A | A | A | A | A | A |
| 1001 | D | D | D | D | D | A | A | A | A | A | A |
| 1010 | D | D | D | D | D | D | A | A | A | A | A |
| 1011 | D | D | D | D | D | D | D | A | A | A | A |
| 1100 | D | D | D | D | D | D | D | D | A | A | A |
| 1101 | D | D | D | D | D | D | D | D | D | A | A |
| 1110 | D | D | D | D | D | D | D | D | D | D | A |
| 1111 | D | D | D | D | D | D | D | D | D | D | D |

A = Analog input D = Digital I/O

**Note 1:** The POR value of the PCFG bits depends on the value of the PBADEN bit in Configuration Register 3H. When PBADEN = 1, PCFG<3:0> = 0000; when PBADEN = 0, PCFG<3:0> = 0111.

**2:** AN5 through AN7 are available only on PIC18F4X80 devices.


© 2009 Microchip Technology Inc.

**REGISTER 20-3: ADCON2: A/D CONTROL REGISTER 2**

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-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' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

bit 2-0 **ADCS<2:0>:** A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

**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.

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/$V_{REF-}$/$CV_{REF}$ pins.

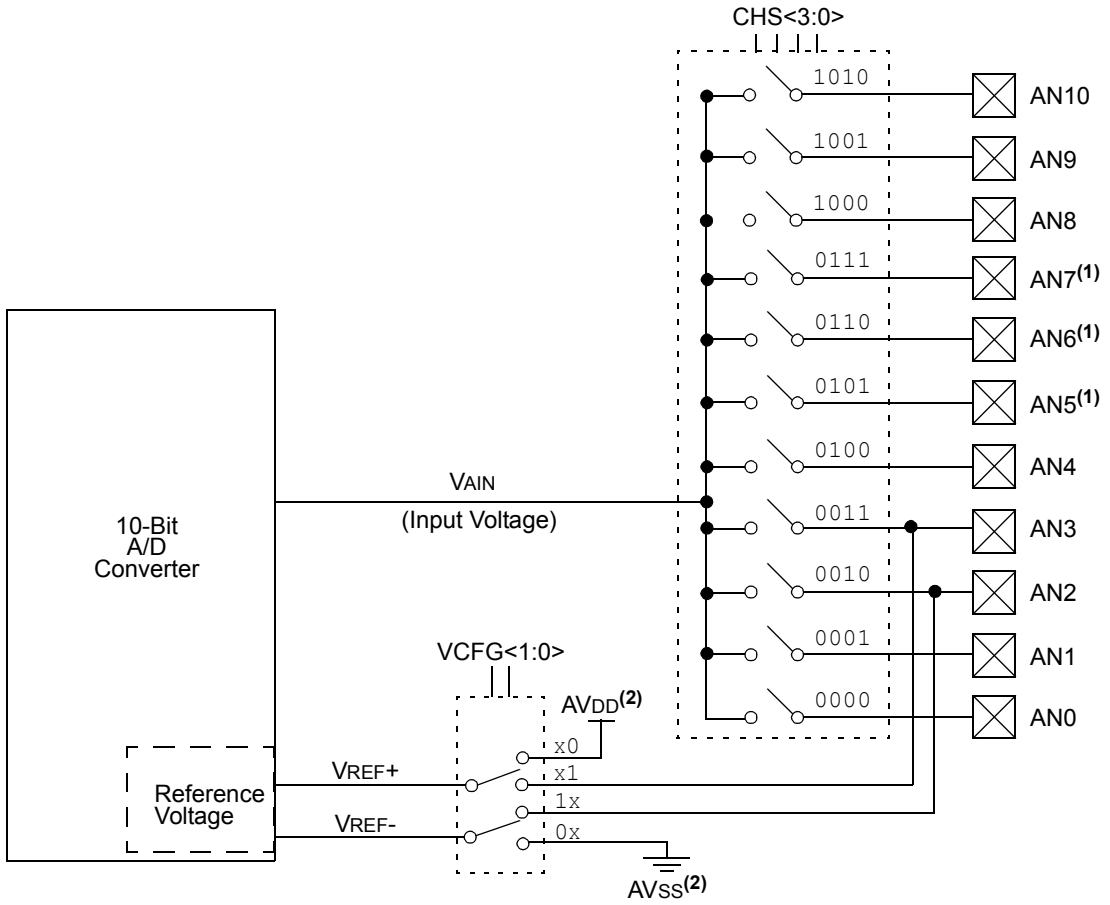
The A/D Converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

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/$\overline{DONE}$ 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 20-1.

**FIGURE 20-1: A/D BLOCK DIAGRAM**




**Note 1:** Channels, AN5 through AN7, are not available on PIC18F2X80 devices.

**2:** I/O pins have diode protection to $V_{DD}$ and $V_{SS}$.


© 2009 Microchip Technology Inc.

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.

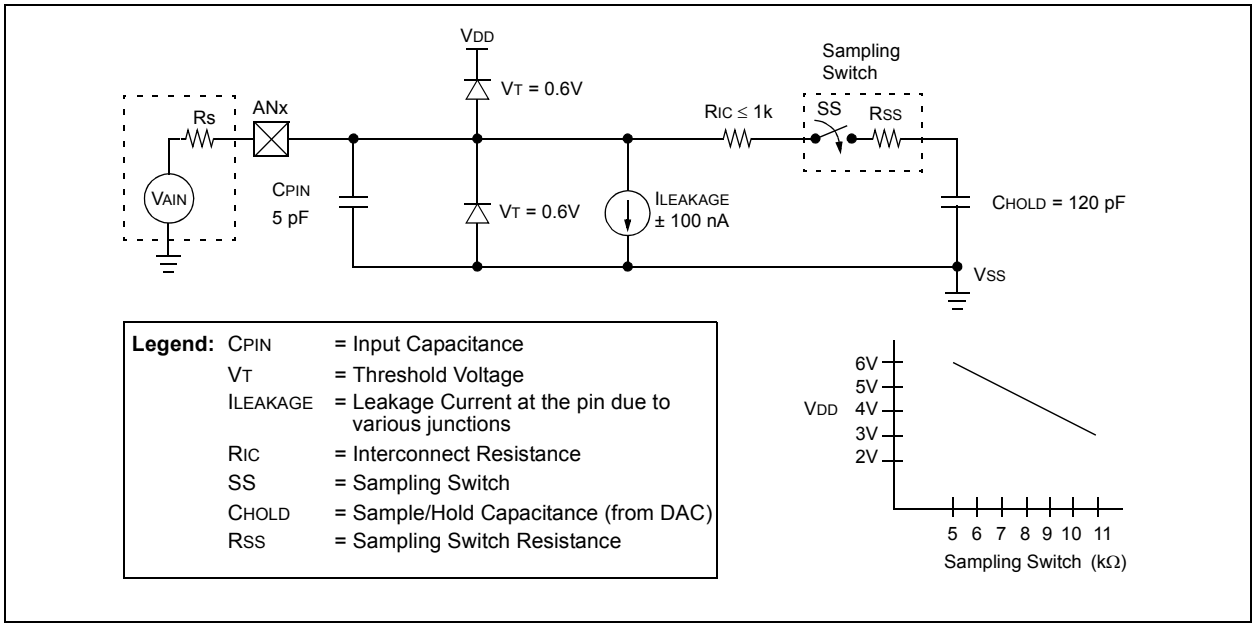
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see **Section 20.1 "A/D Acquisition Requirements"**. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/$\overline{\text{DONE}}$ bit and the actual start of the conversion.

The following steps should be followed to perform 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/$\overline{\text{DONE}}$ bit (ADCON0 register)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ 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 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 next acquisition starts.

**FIGURE 20-2: ANALOG INPUT MODEL**



## 20.1 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 20-2. 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 20-1 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 20-3 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 |

**EQUATION 20-1: ACQUISITION TIME**

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$

**EQUATION 20-2: A/D MINIMUM CHARGING TIME**

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$
or
$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2048)$$

**EQUATION 20-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME**

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$
$$T_{AMP} = 5\ \mu s$$
$$T_{COFF} = (\text{Temp} - 25°C)(0.05\ \mu s/°C)$$
$$(50°C - 25°C)(0.05\ \mu s/°C)$$
$$1.25\ \mu s$$

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, $T_{COFF}$ = 0 ms.

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2047)\ \mu s$$
$$-(120\ pF)(1\ k\Omega + 7\ k\Omega + 2.5\ k\Omega)\ln(0.0004883)\ \mu s$$
$$9.61\ \mu s$$

$$T_{ACQ} = 5\ \mu s + 1.25\ \mu s + 9.61\ \mu s$$
$$12.86\ \mu s$$


© 2009 Microchip Technology Inc.

## 20.2 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 ACQT<2:0> 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.

## 20.3 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 20-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

**TABLE 20-1: TAD vs. DEVICE OPERATING FREQUENCIES**

| AD Clock Source (TAD) | | Maximum Device Frequency | |
|---|---|---|---|
| **Operation** | **ADCS2:ADCS0** | **PIC18F2X80/4X80** | **PIC18LF2X80/4X80(4)** |
| 2 TOSC | 000 | 2.86 MHz | 1.43 kHz |
| 4 TOSC | 100 | 5.71 MHz | 2.86 MHz |
| 8 TOSC | 001 | 11.43 MHz | 5.72 MHz |
| 16 TOSC | 101 | 22.86 MHz | 11.43 MHz |
| 32 TOSC | 010 | 40.0 MHz | 22.86 MHz |
| 64 TOSC | 110 | 40.0 MHz | 22.86 MHz |
| RC(3) | x11 | 1.00 MHz(1) | 1.00 MHz(2) |

**Note 1:** The RC source has a typical TAD time of 1.2 ms.

**2:** The RC source has a typical TAD time of 2.5 ms.

**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 (PIC18LFXXXX) devices only.

## 20.4 Operation in Power-Managed Modes

The selection of the automatic acquisition time and A/D conversion clock is determined in part, by the clock source and frequency while in a power-managed mode.

If the A/D is expected to operate while the device is in a power-managed mode, the ACQT<2:0> and ADCS<2:0> bits in ADCON2 should be updated in accordance with the clock source to be used in that mode. After entering the mode, an A/D acquisition or conversion may be started. Once started, the device should continue to be clocked by the same clock source until the conversion has been completed.

If desired, the device may be placed into the corresponding Idle mode during the conversion. If the device clock frequency is less than 1 MHz, the A/D RC clock source should be selected.

Operation in the Sleep mode requires the A/D FRC clock to be selected. If bits, ACQT<2:0>, 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 Sleep mode. The IDLEN bit (OSCCON<7>) must have already been cleared prior to starting the conversion.

## 20.5 Configuring Analog Port Pins

The ADCON1, TRISA, TRISB and TRISE 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 CHS<3:0> 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.

**3:** The PBADEN bit in Configuration Register 3H configures PORTB pins to reset as analog or digital pins by controlling how the PCFG bits in ADCON1 are reset.


© 2009 Microchip Technology Inc.

## 20.6 A/D Conversions

Figure 20-3 shows the operation of the A/D Converter after the GO/$\overline{\text{DONE}}$ bit has been set and the ACQT<2:0> bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 20-4 shows the operation of the A/D Converter after the GO/$\overline{\text{DONE}}$ bit has been set and the ACQT<2:0> 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 20-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**

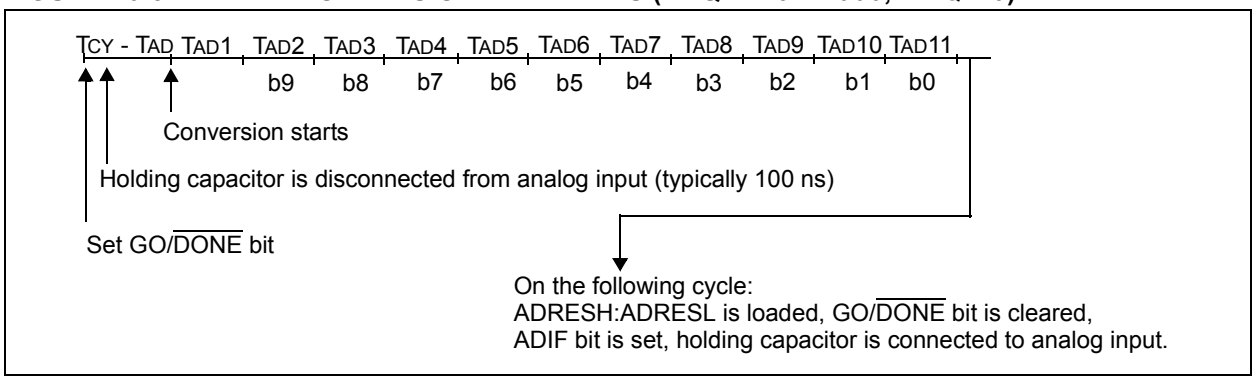
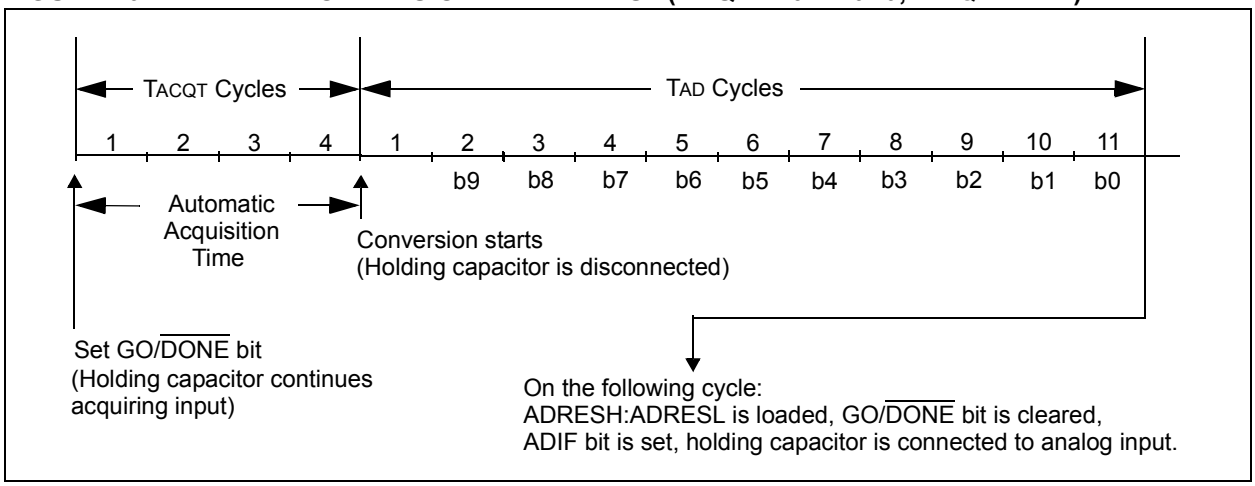



**FIGURE 20-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**



## 20.7 Use of the CCP1 Trigger

An A/D conversion can be started by the "Special Event Trigger" of the ECCP1 module. This requires that the ECCP1M<3:0> bits (ECCP1CON<3:0>) be programmed as '1011' and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/$\overline{\text{DONE}}$ 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/$\overline{\text{DONE}}$ 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 20-2: REGISTERS ASSOCIATED WITH A/D OPERATION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| IPR1 | PSPIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 58 |
| PIR1 | PSPIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 58 |
| PIE1 | PSPIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 58 |
| IPR2 | OSCFIP | CMIP | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP(5) | 57 |
| PIR2 | OSCFIF | CMIF | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF(5) | 58 |
| PIE2 | OSCFIE | CMIE | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE(5) | 58 |
| ADRESH | A/D Result Register High Byte | | | | | | | | 56 |
| ADRESL | A/D Result Register Low Byte | | | | | | | | 56 |
| ADCON0 | — | — | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON | 56 |
| ADCON1 | — | — | VCFG1 | VCFG0 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 56 |
| ADCON2 | ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 57 |
| PORTA | RA7(2) | RA6(2) | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 58 |
| TRISA | TRISA7(2) | TRISA6(2) | PORTA Data Direction Register | | | | | | 58 |
| PORTB | Read PORTB pins, Write LATB Latch | | | | | | | | 58 |
| TRISB | PORTB Data Direction Register | | | | | | | | 58 |
| LATB | PORTB Output Data Latch | | | | | | | | 58 |
| PORTE(4) | — | — | — | — | RE3(3) | Read PORTE pins, Write LATE(1) | | | 58 |
| TRISE(4) | IBF | OBF | IBOV | PSPMODE | — | PORTE Data Direction | | | 58 |
| LATE(4) | — | — | — | — | — | LATE2 | LATE1 | LATE0 | 58 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

**Note 1:** These bits are unimplemented on PIC18F2X80 devices; always maintain these bits clear.

**2:** These pins may be configured as port pins depending on the Oscillator mode selected.

**3:** RE3 port bit is available only as an input pin when the MCLRE Configuration bit is '0'.

**4:** These registers are not implemented on PIC18F2X80 devices.

**5:** These bits are available on PIC18F4X80 and reserved on PIC18F2X80 devices.


© 2009 Microchip Technology Inc.

## 21.0 COMPARATOR MODULE

The analog comparator module contains two comparators that can be configured in a variety of ways. The inputs can be selected from the analog inputs multiplexed with pins, RA0 through RA5, as well as the on-chip voltage reference (see **Section 22.0 "Comparator Voltage Reference Module"**). The digital outputs (normal or inverted) are available at the pin level and can also be read through the control register.

The CMCON register (Register 21-1) selects the comparator input and output configuration. Block diagrams of the various comparator configurations are shown in Figure 21-1.

**REGISTER 21-1: CMCON: COMPARATOR CONTROL REGISTER**

| R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **C2OUT**: Comparator 2 Output bit
When C2INV = 0:
1 = C2 VIN+ > C2 VIN-
0 = C2 VIN+ < C2 VIN-
When C2INV = 1:
1 = C2 VIN+ < C2 VIN-
0 = C2 VIN+ > C2 VIN-

bit 6 **C1OUT**: Comparator 1 Output bit
When C1INV = 0:
1 = C1 VIN+ > C1 VIN-
0 = C1 VIN+ < C1 VIN-
When C1INV = 1:
1 = C1 VIN+ < C1 VIN-
0 = C1 VIN+ > C1 VIN-

bit 5 **C2INV**: Comparator 2 Output Inversion bit
1 = C2 output inverted
0 = C2 output not inverted

bit 4 **C1INV**: Comparator 1 Output Inversion bit
1 = C1 output inverted
0 = C1 output not inverted

bit 3 **CIS**: Comparator Input Switch bit
When CM<2:0> = 110:
1 = C1 VIN- connects to RD0/PSP0/C1IN+
C2 VIN- connects to RD2/PSP2/C2IN+
0 = C1 VIN- connects to RD1/PSP1/C1IN-
C2 VIN- connects to RD3/PSP3/C2IN-

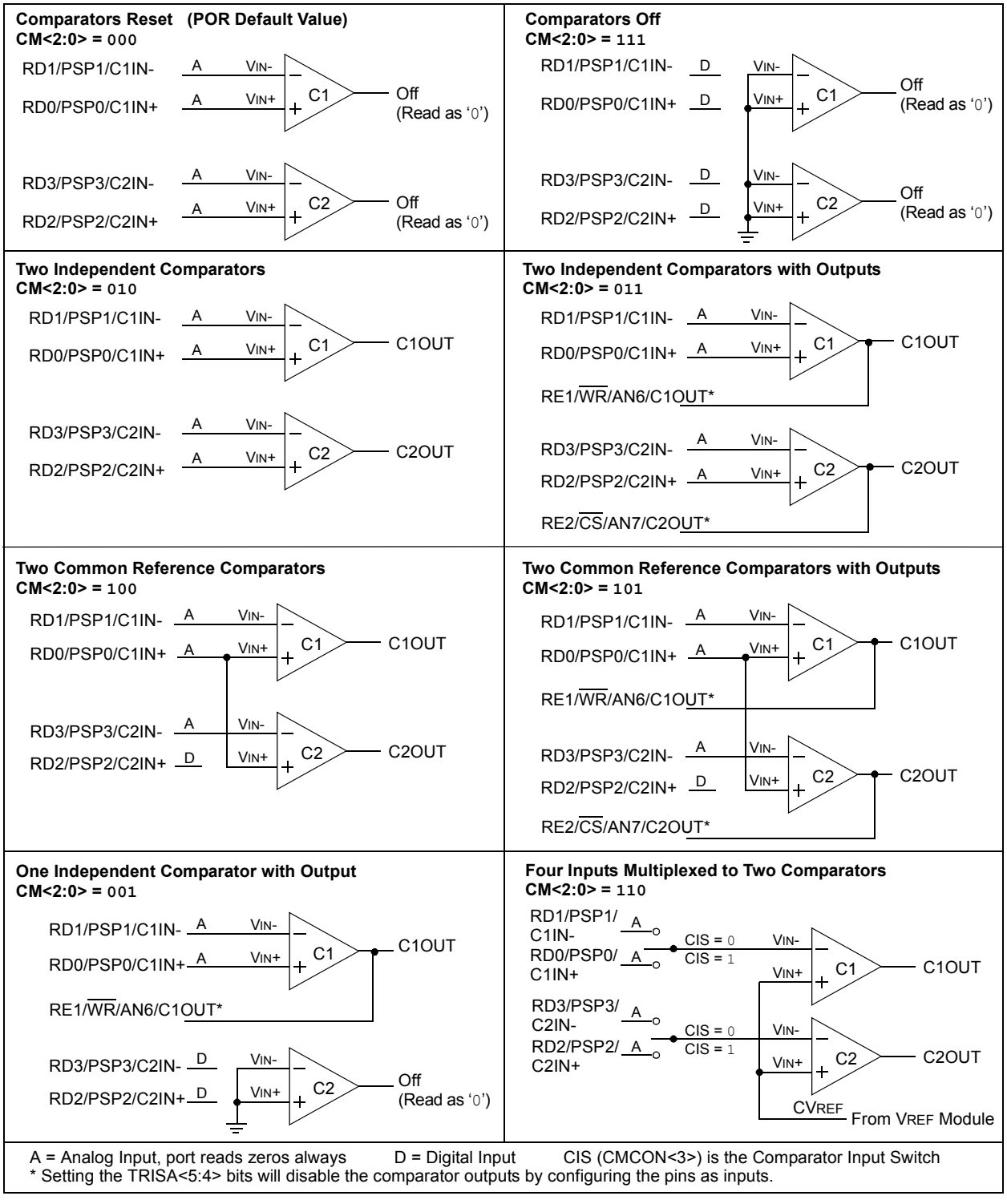
bit 2-0 **CM<2:0>**: Comparator Mode bits
Figure 21-1 shows the Comparator modes and the CM<2:0> bit settings.

## 21.1 Comparator Configuration

There are eight modes of operation for the comparators, shown in Figure 21-1. Bits, CM<2:0> of the CMCON register, are used to select these modes. The TRISA register controls the data direction of the comparator pins for each mode. If the Comparator mode is changed, the comparator output level may not be valid for the specified mode change delay shown in **Section 28.0 "Electrical Characteristics"**.

| **Note:** | Comparator interrupts should be disabled during a Comparator mode change; otherwise, a false interrupt may occur. |
|---|---|

**FIGURE 21-1: COMPARATOR I/O OPERATING MODES**




A = Analog Input, port reads zeros always D = Digital Input CIS (CMCON<3>) is the Comparator Input Switch
* Setting the TRISA<5:4> bits will disable the comparator outputs by configuring the pins as inputs.


© 2009 Microchip Technology Inc.

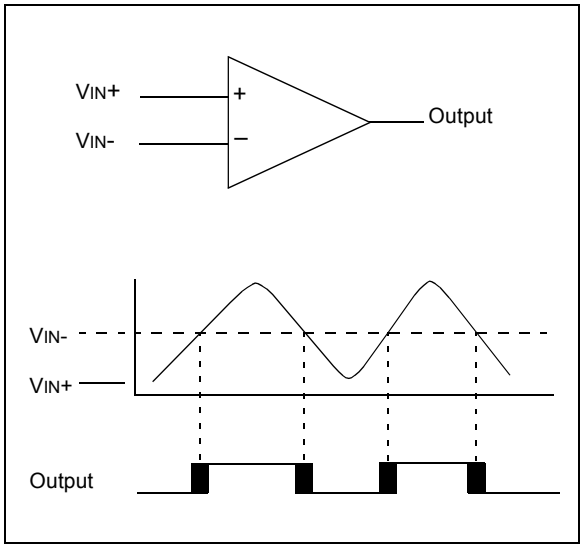
## 21.2 Comparator Operation

A single comparator is shown in Figure 21-2, along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input, VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 21-2 represent the uncertainty, due to input offsets and response time.

## 21.3 Comparator Reference

Depending on the comparator operating mode, either an external or internal voltage reference may be used. The analog signal present at VIN- is compared to the signal at VIN+ and the digital output of the comparator is adjusted accordingly (Figure 21-2).

**FIGURE 21-2: SINGLE COMPARATOR**




### 21.3.1 EXTERNAL REFERENCE SIGNAL

When external voltage references are used, the comparator module can be configured to have the comparators operate from the same or different reference sources. However, threshold detector applications may require the same reference. The reference signal must be between VSS and VDD and can be applied to either pin of the comparator(s).

### 21.3.2 INTERNAL REFERENCE SIGNAL

The comparator module also allows the selection of an internally generated voltage reference from the comparator voltage reference module. This module is described in more detail in **Section 22.0 "Comparator Voltage Reference Module"**.

The internal reference is only available in the mode where four inputs are multiplexed to two comparators (CM<2:0> = 110). In this mode, the internal voltage reference is applied to the VIN+ pin of both comparators.

## 21.4 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level. If the internal reference is changed, the maximum delay of the internal voltage reference must be considered when using the comparator outputs. Otherwise, the maximum delay of the comparators should be used (see **Section 28.0 "Electrical Characteristics"**).

## 21.5 Comparator Outputs

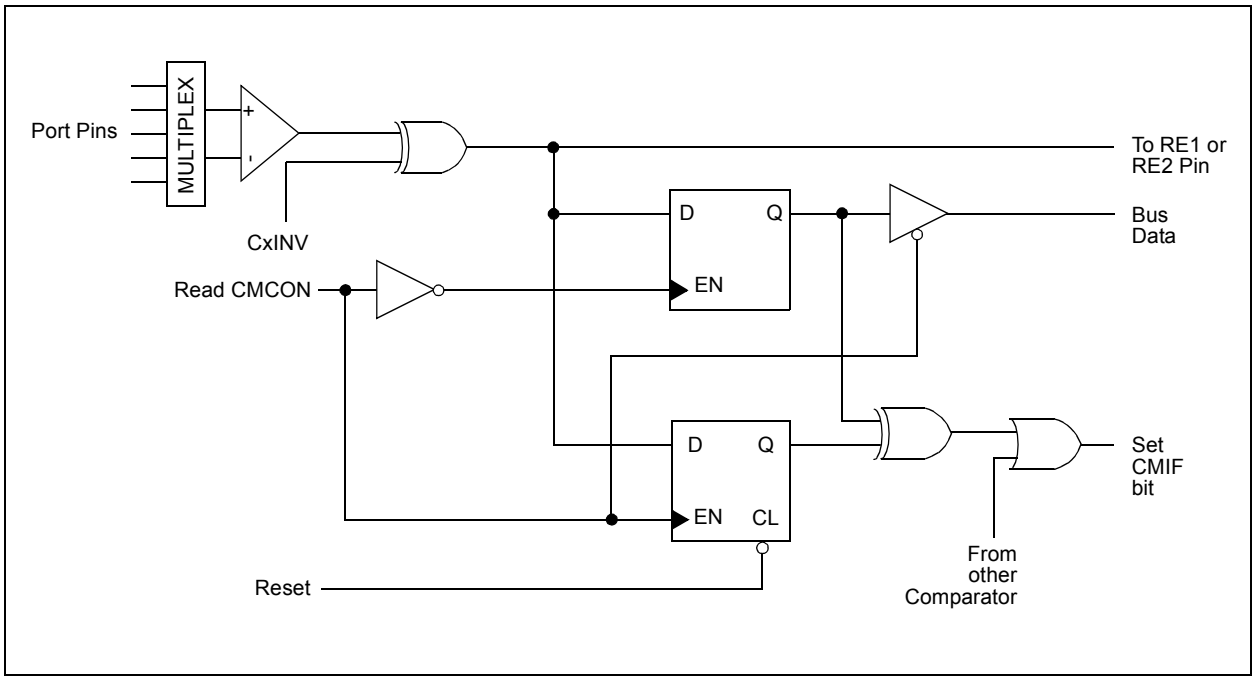
The comparator outputs are read through the CMCON register. These bits are read-only. The comparator outputs may also be directly output to the RE1 and RE2 I/O pins. When enabled, multiplexers in the output path of the RE1 and RE2 pins will switch and the output of each pin will be the unsynchronized output of the comparator. The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications. Figure 21-3 shows the comparator output block diagram.

The TRISE bits will still function as an output enable/ disable for the RE1 and RE2 pins while in this mode.

The polarity of the comparator outputs can be changed using the C2INV and C1INV bits (CMCON<4:5>).

| | |
|---|---|
| **Note 1:** | When reading the PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert an analog input according to the Schmitt Trigger input specification. |
| **2:** | Analog levels on any pin defined as a digital input may cause the input buffer to consume more current than is specified. |

**FIGURE 21-3: COMPARATOR OUTPUT BLOCK DIAGRAM**




## 21.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that occurred. The CMIF bit (PIR2<6>) is the Comparator Interrupt Flag. The CMIF bit must be reset by clearing it. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

Both the CMIE bit (PIE2<6>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit (INTCON<7>) must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.

| **Note:** | If a change in the CMCON register (C1OUT or C2OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMIF (PIR registers) interrupt flag may not get set. |
|---|---|

The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of CMCON will end the mismatch condition.

b) Clear flag bit CMIF.

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

## 21.7 Comparator Operation During Sleep

When a comparator is active and the device is placed in Sleep mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will wake-up the device from Sleep mode when enabled. While the comparator is powered up, higher Sleep currents than shown in the power-down current specification will occur. Each operational comparator will consume additional current, as shown in the comparator specifications. To minimize power consumption while in Sleep mode, turn off the comparators (CM<2:0> = 111) before entering Sleep. If the device wakes up from Sleep, the contents of the CMCON register are not affected.

## 21.8 Effects of a Reset

A device Reset forces the CMCON register to its Reset state, causing the comparator module to be in the Comparator Reset mode (CM<2:0> = 000). This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at Reset time. The comparators are powered down during the Reset interval.


© 2009 Microchip Technology Inc.

## 21.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 21-4. Since the analog pins are connected to a digital output, they have reverse biased diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up condition may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

**FIGURE 21-4: COMPARATOR ANALOG INPUT MODEL**

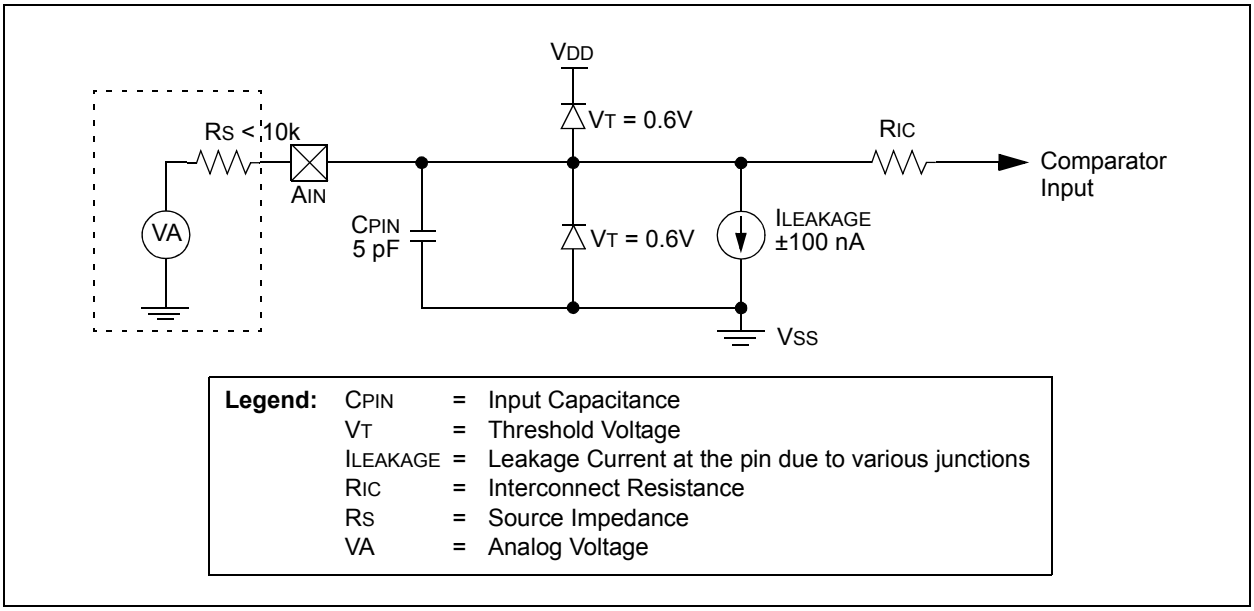



**TABLE 21-1: REGISTERS ASSOCIATED WITH COMPARATOR MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| CMCON(3) | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 57 |
| CVRCON(3) | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 57 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 58 |
| IPR2 | OSCFIP | CMIP(2) | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP | 57 |
| PIR2 | OSCFIF | CMIF(2) | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF | 58 |
| PIE2 | OSCFIE | CMIE(2) | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE | 58 |
| PORTA | RA7(1) | RA6(1) | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 58 |
| LATA | LATA7(1) | LATA6(1) | LATA Data Output Register | | | | | | 58 |
| TRISA | TRISA7(1) | TRISA6(1) | PORTA Data Direction Register | | | | | | 58 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

**Note 1:** PORTA pins are enabled based on oscillator configuration.

**2:** These bits are available in PIC18F4X80 devices and reserved in PIC18F2X80 devices.

**3:** These registers are unimplemented on PIC18F2X80 devices.

**NOTES:**


© 2009 Microchip Technology Inc.

## 22.0 COMPARATOR VOLTAGE REFERENCE MODULE

The comparator voltage reference is a 16-tap resistor ladder network that provides a selectable reference voltage. Although its primary purpose is to provide a reference for the analog comparators, it may also be used independently of them.

A block diagram is of the module shown in Figure 22-1.The resistor ladder is segmented to provide two ranges of CVREF values and has a power-down function to conserve power when the reference is not being used. The module's supply reference can be provided from either device VDD/VSS or an external voltage reference.

### 22.1 Configuring the Comparator Voltage Reference

The voltage reference module is controlled through the CVRCON register (Register 22-1). The comparator voltage reference provides two ranges of output voltage, each with 16 distinct levels. The range to be used is selected by the CVRR bit (CVRCON<5>). The primary difference between the ranges is the size of the steps selected by the CVREF Selection bits (CVR<3:0>), with one range offering finer resolution. The equations used to calculate the output of the comparator voltage reference are as follows:

If CVRR = 1:
CVREF = ((CVR<3:0>)/24) x CVRSRC

If CVRR = 0:
CVREF = (CVDD x 1/4) + (((CVR<3:0>)/32) x CVRSRC)

The comparator reference supply voltage can come from either VDD and VSS, or the external VREF+ and VREF- that are multiplexed with RA2 and RA3. The voltage source is selected by the CVRSS bit (CVRCON<4>).

The settling time of the comparator voltage reference must be considered when changing the CVREF output (see Table 28-3 in **Section 28.0 "Electrical Characteristics"**).

**REGISTER 22-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CVREN | CVROE(1) | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **CVREN**: Comparator Voltage Reference Enable bit
1 = CVREF circuit powered on
0 = CVREF circuit powered down

bit 6 **CVROE**: Comparator VREF Output Enable bit(1)
1 = CVREF voltage level is also output on the RA0/AN0/CVREF pin
0 = CVREF voltage is disconnected from the RA0/AN0/CVREF pin

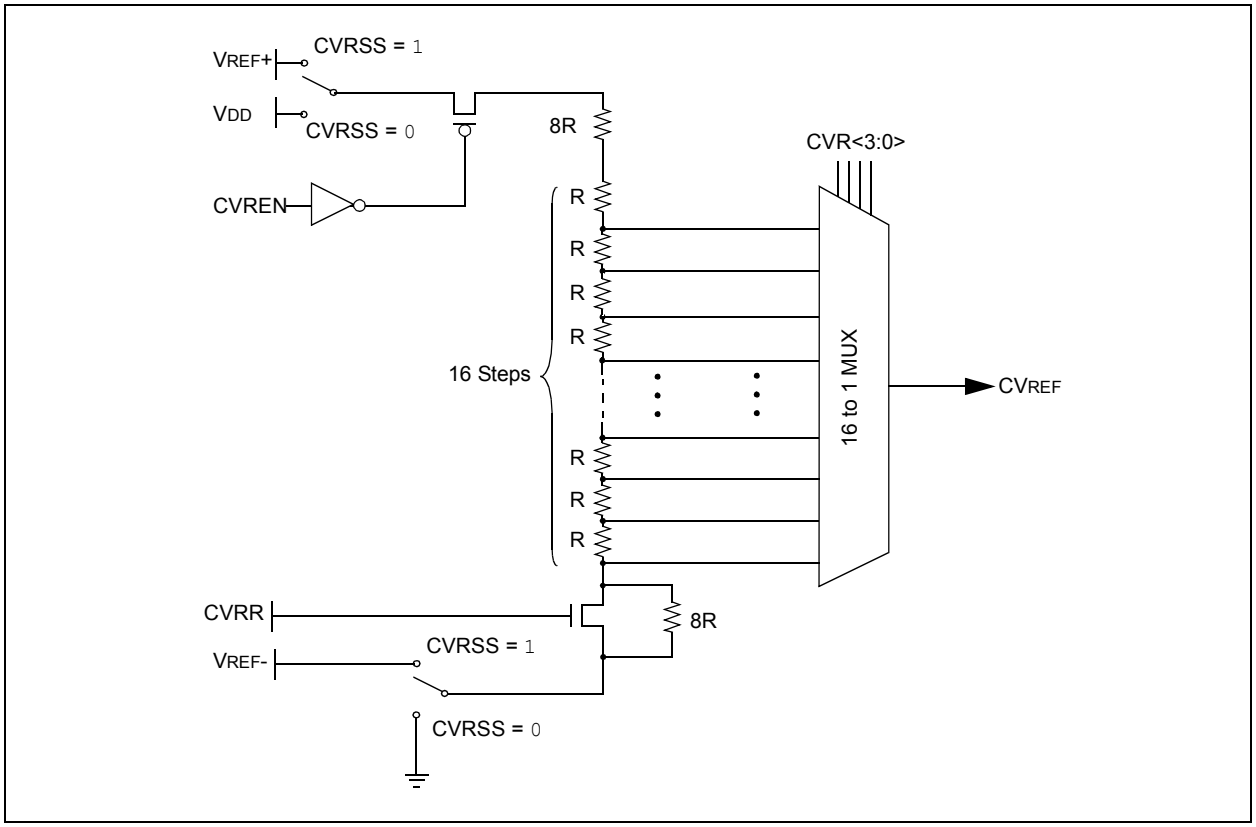
bit 5 **CVRR**: Comparator VREF Range Selection bit
1 = 0.00 CVRSRC to 0.75 CVRSRC, with CVRSRC/24 step size
0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size

bit 4 **CVRSS**: Comparator VREF Source Selection bit
1 = Comparator reference source, CVRSRC = (VREF+) – (VREF-)
0 = Comparator reference source, CVRSRC = VDD – VSS

bit 3-0 **CVR<3:0>:** Comparator VREF Value Selection bits ($0 \le$ (CVR<3:0>) $\le 15$)
When CVRR = 1:
CVREF = ((CVR<3:0>)/24) • (CVRSRC)
When CVRR = 0:
CVREF = (CVRSRC/4) + ((CVR<3:0>)/32) • (CVRSRC)

**Note 1:** CVROE overrides the TRISA<0> bit setting. If enabled for output, RA2 must also be configured as an input by setting TRISA<2> to '1'.

**FIGURE 22-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM**




## 22.2 Voltage Reference Accuracy/Error

The full range of voltage reference cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 22-1) keep CVREF from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the CVREF output changes with fluctuations in that source. The tested absolute accuracy of the voltage reference can be found in **Section 28.0 "Electrical Characteristics"**.

## 22.3 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the CVRCON register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

## 22.4 Effects of a Reset

A device Reset disables the voltage reference by clearing bit, CVREN (CVRCON<7>). This Reset also disconnects the reference from the RA0 pin by clearing bit, CVROE (CVRCON<6>), and selects the high-voltage range by clearing bit, CVRR (CVRCON<5>). The CVR value select bits are also cleared.

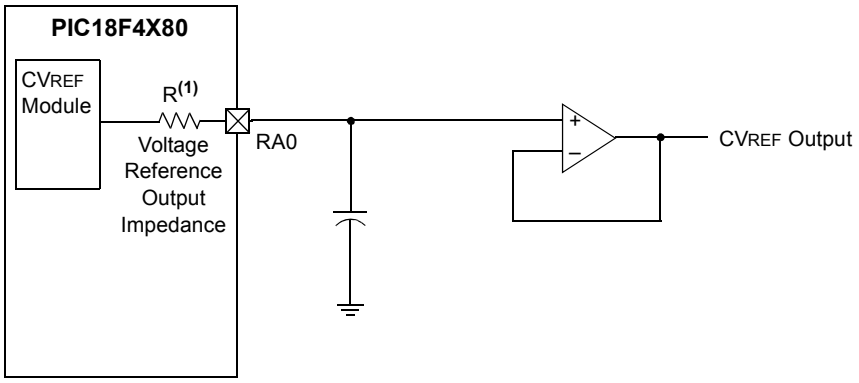
## 22.5 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RA0 pin if the TRISA<0> bit and the CVROE bit are both set. Enabling the voltage reference output onto the RA0 pin, with an input signal present, will increase current consumption. Connecting RA0 as a digital output with CVRSS enabled will also increase current consumption.

The RA0 pin can be used as a simple D/A output with limited drive capability. Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to VREF. Figure 22-2 shows an example buffering technique.


© 2009 Microchip Technology Inc.

**FIGURE 22-2: COMPARATOR VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE**




**Note 1:** R is dependent upon the voltage reference configuration bits, CVRCON<3:0> and CVRCON<5>.

**TABLE 22-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| CVRCON(2) | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 57 |
| CMCON(2) | C2OUT | C1OUT | C2INV | C1INV | CIS | CM2 | CM1 | CM0 | 57 |
| TRISA | TRISA7(1) | TRISA6(1) | PORTA Data Direction Register | | | | | | 58 |

**Legend:** Shaded cells are not used with the comparator voltage reference.

**Note 1:** PORTA pins are enabled based on oscillator configuration.

**2:** These registers are unimplemented on PIC18F2X80 devices.

**NOTES:**


© 2009 Microchip Technology Inc.

## 23.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

PIC18F2480/2580/4480/4580 devices have a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that allows the user to specify both a device voltage trip point and the direction of change from that point. If the device experiences an excursion past the trip point in that direction, 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 the interrupt.

The High/Low-Voltage Detect Control register (Register 23-1) completely controls the operation of the HLVD module. This allows the circuitry to be "turned off" by the user under software control, which minimizes the current consumption for the device.

The block diagram for the HLVD module is shown in Figure 23-1.

**REGISTER 23-1: HLVDCON: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER**

| R/W-0 | U-0 | R-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-1 |
|---|---|---|---|---|---|---|---|
| VDIRMAG | — | IRVST | HLVDEN | HLVDL3(1) | HLVDL2(1) | HLVDL1(1) | HLVDL0(1) |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **VDIRMAG:** Voltage Direction Magnitude Select bit
1 = Event occurs when voltage equals or exceeds trip point (HLVDL<3:0>)
0 = Event occurs when voltage equals or falls below trip point (HLVDL<3:0>)

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

bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit
1 = Indicates that the voltage detect logic will generate the interrupt flag at the specified voltage range
0 = Indicates that the voltage detect logic will not generate the interrupt flag at the specified voltage range and the HLVD interrupt should not be enabled

bit 4 **HLVDEN:** High/Low-Voltage Detect Power Enable bit
1 = HLVD enabled
0 = HLVD disabled

bit 3-0 **HLVDL<3:0>:** High/Low-Voltage Detection Limit bits(1)
1111 = External analog input is used (input comes from the HLVDIN pin)
1110 = 4.48V-4.69V
1101 = 4.23V-4.43V
1100 = 4.01V-4.20V
1011 = 3.81V-3.99V
1010 = 3.63V-3.80V
1001 = 3.46V-3.63V
1000 = 3.31V-3.47V
0111 = 3.05V-3.19V
0110 = 2.82V-2.95V
0101 = 2.72V-2.85V
0100 = 2.54V-2.66V
0011 = 2.38V-2.49V
0010 = 2.31V-2.42V
0001 = 2.18V-2.28V
0000 = 2.12V-2.22V

**Note 1:** HLVDL<3:0> modes that result in a trip point below the valid operating voltage of the device are not tested.

The module is enabled by setting the HLVDEN bit. Each time that the HLVD module is enabled, the circuitry requires some time to stabilize. The IRVST bit is a read-only bit and is used to indicate when the circuit is stable. The module can only generate an interrupt after the circuit is stable and IRVST is set.

The VDIRMAG bit determines the overall operation of the module. When VDIRMAG is cleared, the module monitors for drops in VDD below a predetermined set point. When the bit is set, the module monitors for rises in VDD above the set point.

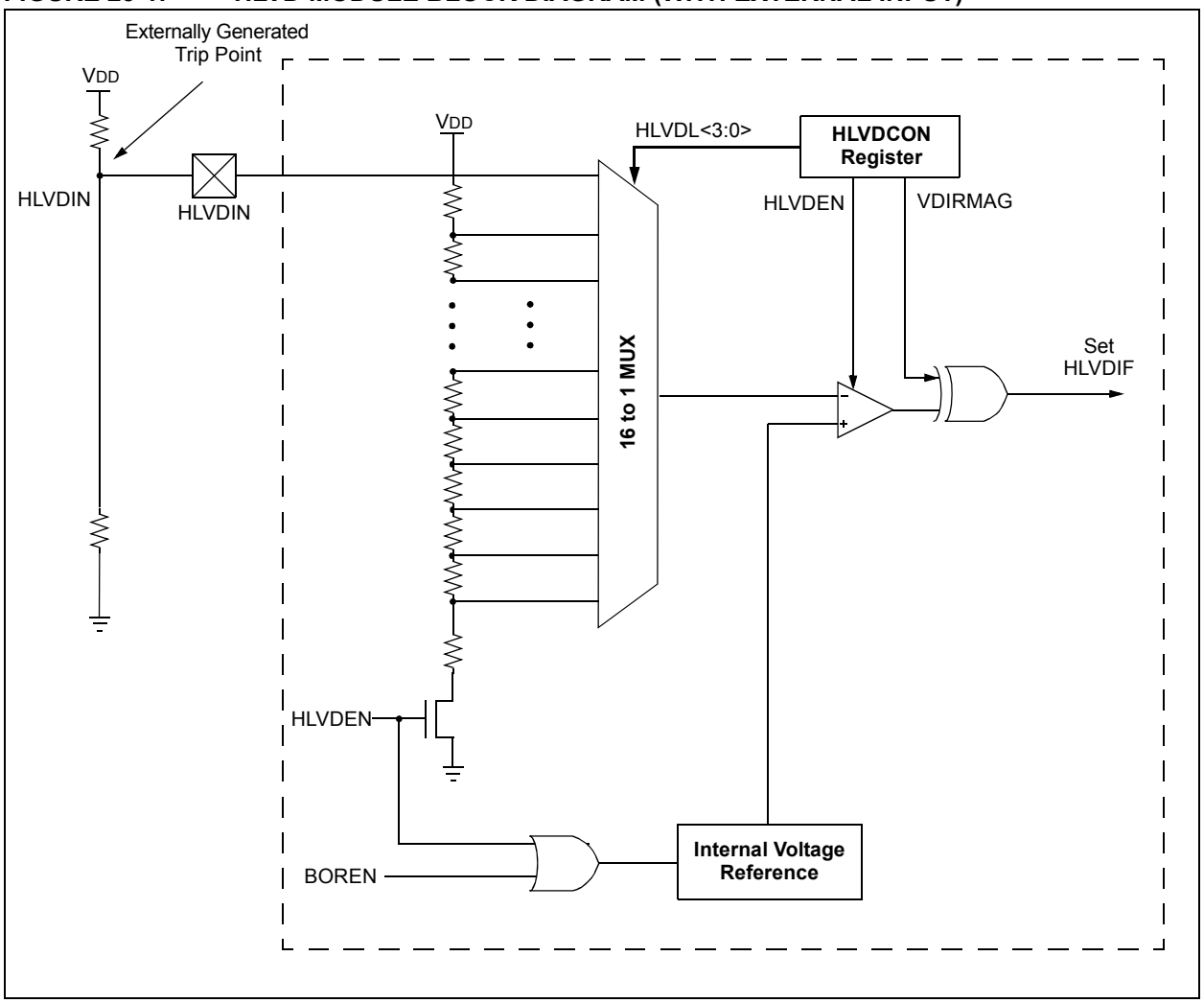
## 23.1 Operation

When the HLVD module is enabled, a comparator uses an internally generated reference voltage as the set point. The set point is compared with the trip point where each node in the resistor divider represents a trip point voltage. The “trip point” voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module. When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable to any one of 16 values. The trip point is selected by programming the HLVDL<3:0> bits (HLVDCON<3:0>).

The HLVD 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, HLVDL<3:0>, are set to ‘1111’. In this state, the comparator input is multiplexed from the external input pin, HLVDIN. This gives users flexibility because it allows them to configure the High/Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

**FIGURE 23-1: HLVD MODULE BLOCK DIAGRAM (WITH EXTERNAL INPUT)**





© 2009 Microchip Technology Inc.

## 23.2 HLVD Setup

The following steps are needed to set up the HLVD module:

1. Disable the module by clearing the HLVDEN bit (HLVDCON<4>).
2. Write the value to the HLVDL<3:0> bits that select the desired HLVD trip point.
3. Set the VDIRMAG bit to detect high voltage (VDIRMAG = 1) or low voltage (VDIRMAG = 0).
4. Enable the HLVD module by setting the HLVDEN bit.
5. Clear the HLVD interrupt flag (PIR2<2>), which may have been set from a previous interrupt.
6. Enable the HLVD interrupt if interrupts are desired by setting the HLVDIE and GIE bits (PIE<2> and INTCON<7>). An interrupt will not be generated until the IRVST bit is set.

## 23.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled and will consume static current. The total current consumption, when enabled, is specified in electrical specification parameter D022B.

Depending on the application, the HLVD module does not need to be operating constantly. To decrease the current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After doing the check, the HLVD module may be disabled.

## 23.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification parameter D420, may be used by other internal circuitry, such as the Programmable Brown-out Reset. If the HLVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time, TIRVST, is an interval that is independent of device clock speed. It is specified in electrical specification parameter 36.

The HLVD interrupt flag is not enabled until TIRVST has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval. Refer to Figure 23-2 or Figure 23-3.

**FIGURE 23-2: LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 0)**

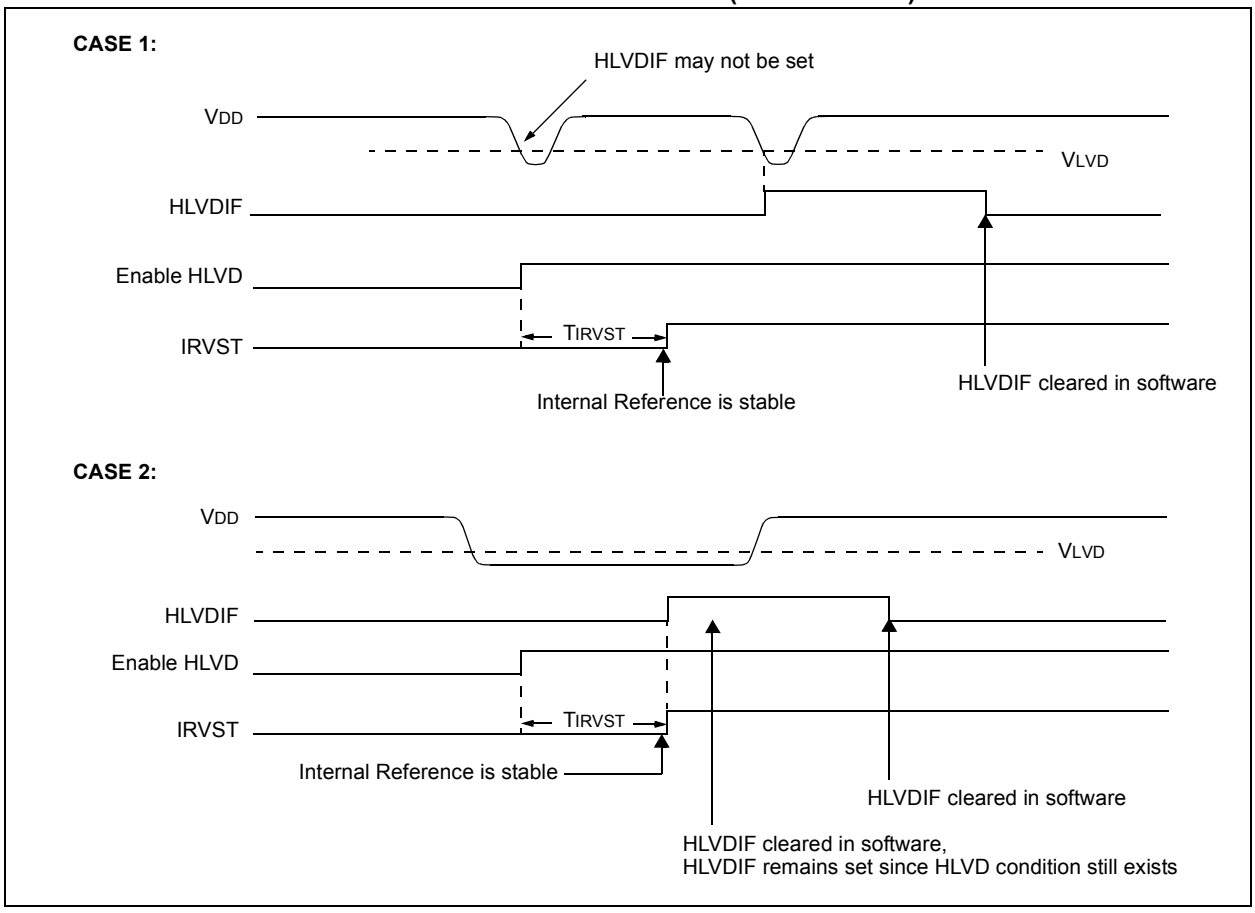
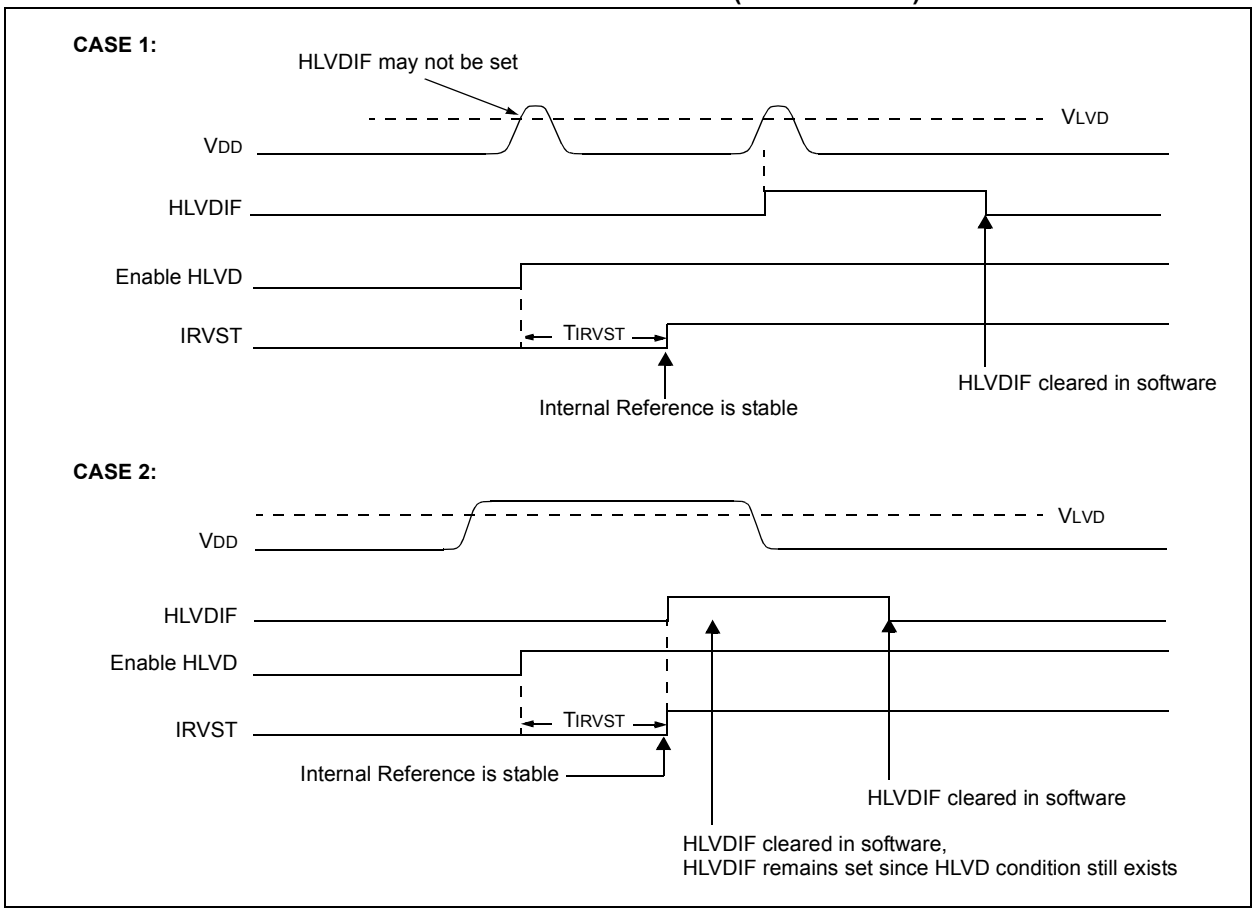


**FIGURE 23-3: HIGH-VOLTAGE DETECT OPERATION (VDIRMAG = 1)**

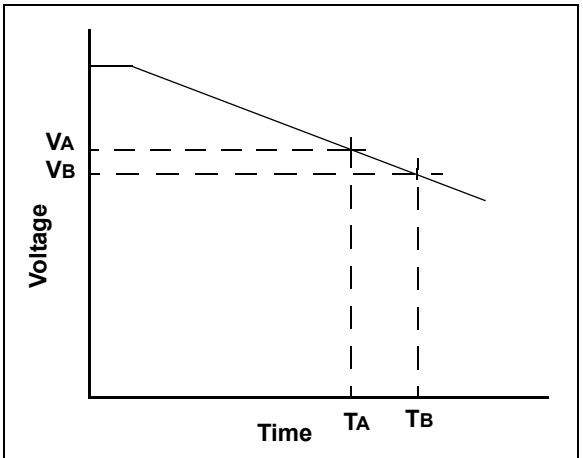



## 23.5 Applications

In many applications, the ability to detect a drop below, or rise above a particular threshold is desirable. For example, the HLVD module could be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a high-voltage detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 23-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, VA, the HLVD logic generates an interrupt at time, TA. The interrupt could cause the execution of an ISR, which would allow the application to perform "housekeeping tasks" and perform a controlled shutdown before the device voltage exits the valid operating range at TB. The HLVD, thus, would give the application a time window, represented by the difference between TA and TB, to safely exit.

**FIGURE 23-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION**




**Legend:** VA = HLVD trip point
VB = Minimum valid device operating voltage


© 2009 Microchip Technology Inc.

## 23.6 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the HLVDIF 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.

## 23.7 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

**TABLE 23-1: REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| HLVDCON | VDIRMAG | — | IRVST | HLVDEN | HLVDL3 | HLVDL2 | HLVDL1 | HLVDL0 | 56 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 55 |
| PIR2 | OSCFIF | CMIF | — | EEIF | BCLIF | HLVDIF | TMR3IF | ECCP1IF | 58 |
| PIE2 | OSCFIE | CMIE | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE | 58 |
| IPR2 | OSCFIP | CMIP | — | EEIP | BCLIP | HLVDIP | TMR3IP | ECCP1IP | 57 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the HLVD module.

**NOTES:**


© 2009 Microchip Technology Inc.

## 24.0 ECAN MODULE

PIC18F2480/2580/4480/4580 devices contain an Enhanced Controller Area Network (ECAN) module. The ECAN module is fully backward compatible with the CAN module available in PIC18CXX8 and PIC18FXX8 devices.

The Controller Area Network (CAN) module is a serial interface which is useful for communicating with other peripherals or microcontroller devices. This interface, or protocol, was designed to allow communications within noisy environments.

The ECAN module is a communication controller, implementing the CAN 2.0A or B protocol as defined in the BOSCH specification. The module will support CAN 1.2, CAN 2.0A, CAN 2.0B Passive and CAN 2.0B Active versions of the protocol. The module implementation is a full CAN system; however, the CAN specification is not covered within this data sheet. Refer to the BOSCH CAN specification for further details.

The module features are as follows:

- Implementation of the CAN protocol, CAN 1.2, CAN 2.0A and CAN 2.0B
- DeviceNet™ data bytes filter support
- Standard and extended data frames
- 0-8 bytes data length
- Programmable bit rate up to 1 Mbit/sec
- Fully backward compatible with the PIC18XXX8 CAN module
- Three modes of operation:
  - Mode 0 – Legacy mode
  - Mode 1 – Enhanced Legacy mode with DeviceNet support
  - Mode 2 – FIFO mode with DeviceNet support
- Support for remote frames with automated handling
- Double-buffered receiver with two prioritized received message storage buffers
- Six buffers programmable as RX and TX message buffers
- 16 full (standard/extended identifier) acceptance filters that can be linked to one of four masks
- Two full acceptance filter masks that can be assigned to any filter
- One full acceptance filter that can be used as either an acceptance filter or acceptance filter mask
- Three dedicated transmit buffers with application specified prioritization and abort capability
- Programmable wake-up functionality with integrated low-pass filter
- Programmable Loopback mode supports self-test operation
- Signaling via interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to timer module for time-stamping and network synchronization
- Low-power Sleep mode

### 24.1 Module Overview

The CAN bus module consists of a protocol engine and message buffering and control. The CAN protocol engine automatically handles all functions for receiving and transmitting messages on the CAN bus. Messages are transmitted by first loading the appropriate data registers. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against filters to see if it should be received and stored in one of the two receive registers.

The CAN module supports the following frame types:

- Standard Data Frame
- Extended Data Frame
- Remote Frame
- Error Frame
- Overload Frame Reception

The CAN module uses the RB2/CANTX and RB3/CANRX pins to interface with the CAN bus. In normal mode, the CAN module automatically overrides TRISB<2>. The user must ensure that TRISB<3> is set.

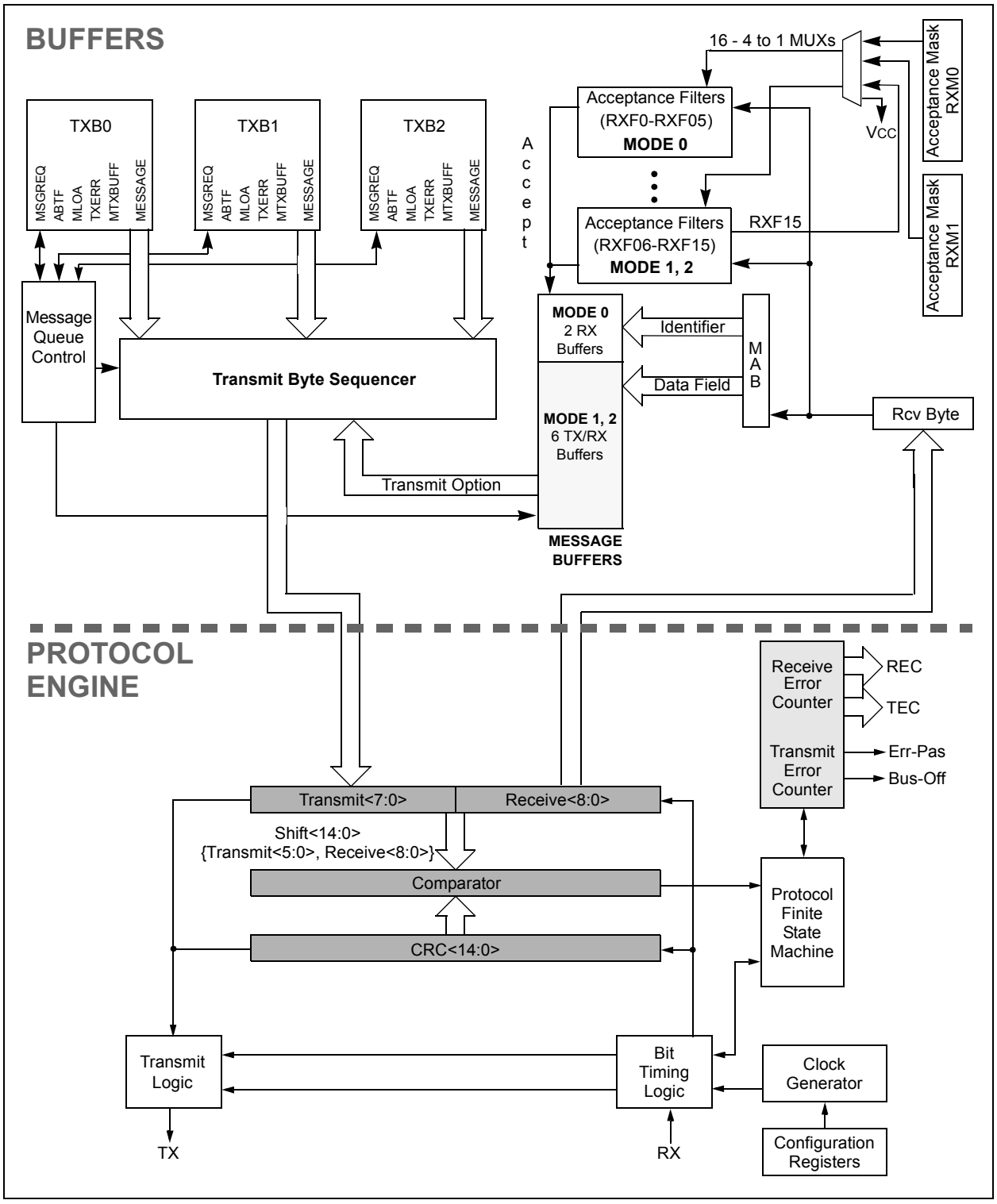
#### 24.1.1 MODULE FUNCTIONALITY

The CAN bus module consists of a protocol engine, message buffering and control (see Figure 24-1). The protocol engine can best be understood by defining the types of data frames to be transmitted and received by the module.

The following sequence illustrates the necessary initialization steps before the ECAN module can be used to transmit or receive a message. Steps can be added or removed depending on the requirements of the application.

1. Initial LAT and TRIS bits for RX and TX CAN.
2. Ensure that the ECAN module is in Configuration mode.
3. Select ECAN Operational mode.
4. Set up the Baud Rate registers.
5. Set up the Filter and Mask registers.
6. Set the ECAN module to normal mode or any other mode required by the application logic.

**FIGURE 24-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM**





© 2009 Microchip Technology Inc.

## 24.2 CAN Module Registers

| **Note:** | Not all CAN registers are available in the Access Bank. |
|---|---|

There are many control and data registers associated with the CAN module. For convenience, their descriptions have been grouped into the following sections:

- Control and Status Registers
- Dedicated Transmit Buffer Registers
- Dedicated Receive Buffer Registers
- Programmable TX/RX and Auto RTR Buffers
- Baud Rate Control Registers
- I/O Control Register
- Interrupt Status and Control Registers

Detailed descriptions of each register and their usage are described in the following sections.

### 24.2.1 CAN CONTROL AND STATUS REGISTERS

The registers described in this section control the overall operation of the CAN module and show its operational status.

**REGISTER 24-1: CANCON: CAN CONTROL REGISTER**

| Mode 0 | R/W-1 | R/W-0 | R/W-0 | R/S-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|---|
| | REQOP2 | REQOP1 | REQOP0 | ABAT | WIN2 | WIN1 | WIN0 | — |

| Mode 1 | R/W-1 | R/W-0 | R/W-0 | R/S-0 | U0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|---|
| | REQOP2 | REQOP1 | REQOP0 | ABAT | — | — | — | — |

| Mode 2 | R/W-1 | R/W-0 | R/W-0 | R/S-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|---|
| | REQOP2 | REQOP1 | REQOP0 | ABAT | FP3 | FP2 | FP1 | FP0 |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | S = Settable bit | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **REQOP<2:0>:** Request CAN Operation Mode bits
1xx = Request Configuration mode
011 = Request Listen Only mode
010 = Request Loopback mode
001 = Disabled/Sleep mode
000 = Request Normal mode

bit 4 **ABAT:** Abort All Pending Transmissions bit
1 = Abort all pending transmissions (in all transmit buffers)**(1)**
0 = Transmissions proceeding as normal

bit 3-1 Mode 0:
**WIN<2:0>:** Window Address bits
These bits select which of the CAN buffers to switch into the Access Bank area. This allows access to the buffer registers from any data memory bank. After a frame has caused an interrupt, the ICODE<3:0> bits can be copied to the WIN<2:0> bits to select the correct buffer. See Example 24-2 for a code example.
111 = Receive Buffer 0
110 = Receive Buffer 0
101 = Receive Buffer 1
100 = Transmit Buffer 0
011 = Transmit Buffer 1
010 = Transmit Buffer 2
001 = Receive Buffer 0
000 = Receive Buffer 0

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

bit 4-0 Mode 1:
**Unimplemented:** Read as '0'
Mode 2:
**FP<3:0>:** FIFO Read Pointer bits
These bits point to the message buffer to be read.
0000 = Receive Message Buffer 0
0001 = Receive Message Buffer 1
0010 = Receive Message Buffer 2
0011 = Receive Message Buffer 3
0100 = Receive Message Buffer 4
0101 = Receive Message Buffer 5
0110 = Receive Message Buffer 6
0111 = Receive Message Buffer 7
1000:1111 Reserved

**Note 1:** This bit will clear when all transmissions are aborted.


© 2009 Microchip Technology Inc.

**REGISTER 24-2: CANSTAT: CAN STATUS REGISTER**

| Mode 0 | R-1 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | U-0 |
|---|---|---|---|---|---|---|---|---|
| | OPMODE2(1) | OPMODE1(1) | OPMODE0(1) | — | ICODE3 | ICODE2 | ICODE1 | — |

| Mode 1,2 | R-1 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|---|
| | OPMODE2(1) | OPMODE1(1) | OPMODE0(1) | EICODE4 | EICODE3 | EICODE2 | EICODE1 | EICODE0 |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **OPMODE<2:0>:** Operation Mode Status bits(1)

111 = Reserved
110 = Reserved
101 = Reserved
100 = Configuration mode
011 = Listen Only mode
010 = Loopback mode
001 = Disable/Sleep mode
000 = Normal mode

bit 4 Mode 0:
**Unimplemented:** Read as '0'

bit 3-1 **ICODE<3:1>:** Interrupt Code bits

When an interrupt occurs, a prioritized coded interrupt value will be present in these bits. This code indicates the source of the interrupt. By copying ICODE<3:1> to WIN<3:0> (Mode 0) or EICODE<4:0> to EWIN<4:0> (Mode 1 and 2), it is possible to select the correct buffer to map into the Access Bank area. See Example 24-2 for a code example. To simplify the description, the following table lists all five bits.

| | Mode 0 | Mode 1 | Mode 2 |
|---|---|---|---|
| No interrupt | 00000 | 00000 | 00000 |
| CAN bus error interrupt | 00010 | 00010 | 00010 |
| TXB2 interrupt | 00100 | 00100 | 00100 |
| TXB1 interrupt | 00110 | 00110 | 00110 |
| TXB0 interrupt | 01000 | 01000 | 01000 |
| RXB1 interrupt | 01010 | 10001 | ----- |
| RXB0 interrupt | 01100 | 10000 | 10000 |
| Wake-up interrupt | 00010 | 01110 | 01110 |
| RXB0 interrupt | ----- | 10000 | 10000 |
| RXB1 interrupt | ----- | 10001 | 10000 |
| RX/TX B0 interrupt | ----- | 10010 | 10010(2) |
| RX/TX B1 interrupt | ----- | 10011 | 10011(2) |
| RX/TX B2 interrupt | ----- | 10100 | 10100(2) |
| RX/TX B3 interrupt | ----- | 10101 | 10101(2) |
| RX/TX B4 interrupt | ----- | 10110 | 10110(2) |
| RX/TX B5 interrupt | ----- | 10111 | 10111(2) |

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

bit 4-0 Mode 1, 2:
**EICODE<4:0>:** Interrupt Code bits
See ICODE<3:1> above.

**Note 1:** To achieve maximum power saving and/or able to wake-up on CAN bus activity, switch CAN module in Disable/Sleep mode before putting device to Sleep.

**2:** If buffer is configured as receiver, EICODE bits will contain '10000' upon interrupt.

**EXAMPLE 24-1: CHANGING TO CONFIGURATION MODE**

```
    ; Request Configuration mode.
    MOVLW   B'10000000'                 ; Set to Configuration Mode.
    MOVWF   CANCON
    ; A request to switch to Configuration mode may not be immediately honored.
    ; Module will wait for CAN bus to be idle before switching to Configuration Mode.
    ; Request for other modes such as Loopback, Disable etc. may be honored immediately.
    ; It is always good practice to wait and verify before continuing.
ConfigWait:
    MOVF    CANSTAT, W                  ; Read current mode state.
    ANDLW   B'10000000'                 ; Interested in OPMODE bits only.
    TSTFSZ  WREG                        ; Is it Configuration mode yet?
    BRA     ConfigWait                  ; No. Continue to wait...
    ; Module is in Configuration mode now.
    ; Modify configuration registers as required.
    ; Switch back to Normal mode to be able to communicate.
```

**EXAMPLE 24-2: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS**

```
    ; Save application required context.
    ; Poll interrupt flags and determine source of interrupt
    ; This was found to be CAN interrupt
    ; TempCANCON and TempCANSTAT are variables defined in Access Bank low
    MOVFF   CANCON, TempCANCON          ; Save CANCON.WIN bits
                                        ; This is required to prevent CANCON
                                        ; from corrupting CAN buffer access
                                        ; in-progress while this interrupt
                                        ; occurred
    MOVFF   CANSTAT, TempCANSTAT        ; Save CANSTAT register
                                        ; This is required to make sure that
                                        ; we use same CANSTAT value rather
                                        ; than one changed by another CAN
                                        ; interrupt.
    MOVF    TempCANSTAT, W              ; Retrieve ICODE bits
    ANDLW   B'00001110'
    ADDWF   PCL, F                      ; Perform computed GOTO
                                        ; to corresponding interrupt cause
    BRA     NoInterrupt                 ; 000 = No interrupt
    BRA     ErrorInterrupt              ; 001 = Error interrupt
    BRA     TXB2Interrupt               ; 010 = TXB2 interrupt
    BRA     TXB1Interrupt               ; 011 = TXB1 interrupt
    BRA     TXB0Interrupt               ; 100 = TXB0 interrupt
    BRA     RXB1Interrupt               ; 101 = RXB1 interrupt
    BRA     RXB0Interrupt               ; 110 = RXB0 interrupt
                                        ; 111 = Wake-up on interrupt
WakeupInterrupt
    BCF     PIR3, WAKIF                 ; Clear the interrupt flag
    ;
    ; User code to handle wake-up procedure
    ;
    ;
    ; Continue checking for other interrupt source or return from here
    …
NoInterrupt
    …                                   ; PC should never vector here. User may
                                        ; place a trap such as infinite loop or pin/port
                                        ; indication to catch this error.
```


© 2009 Microchip Technology Inc.

**EXAMPLE 24-2: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS (CONTINUED)**

```
ErrorInterrupt
    BCF     PIR3, ERRIF                 ; Clear the interrupt flag
    …                                   ; Handle error.
    RETFIE
TXB2Interrupt
    BCF     PIR3, TXB2IF                ; Clear the interrupt flag
    GOTO    AccessBuffer
TXB1Interrupt
    BCF     PIR3, TXB1IF                ; Clear the interrupt flag
    GOTO    AccessBuffer
TXB0Interrupt
    BCF     PIR3, TXB0IF                ; Clear the interrupt flag
    GOTO    AccessBuffer
RXB1Interrupt
    BCF     PIR3, RXB1IF                ; Clear the interrupt flag
    GOTO    Accessbuffer
RXB0Interrupt
    BCF     PIR3, RXB0IF                ; Clear the interrupt flag
    GOTO    AccessBuffer
AccessBuffer                            ; This is either TX or RX interrupt
    ; Copy CANSTAT.ICODE bits to CANCON.WIN bits
    MOVF    TempCANCON, W               ; Clear CANCON.WIN bits before copying
                                        ; new ones.
    ANDLW   B'11110001'                 ; Use previously saved CANCON value to
                                        ; make sure same value.
    MOVWF   TempCANCON                  ; Copy masked value back to TempCANCON
    MOVF    TempCANSTAT, W              ; Retrieve ICODE bits
    ANDLW   B'00001110'                 ; Use previously saved CANSTAT value
                                        ; to make sure same value.
    IORWF   TempCANCON                  ; Copy ICODE bits to WIN bits.
    MOVFF   TempCANCON, CANCON          ; Copy the result to actual CANCON
    ; Access current buffer…
    ; User code
    ; Restore CANCON.WIN bits
    MOVF    CANCON, W                   ; Preserve current non WIN bits
    ANDLW   B'11110001'
    IORWF   TempCANCON                  ; Restore original WIN bits
    ; Do not need to restore CANSTAT - it is read-only register.
    ; Return from interrupt or check for another module interrupt source
```

**REGISTER 24-3: ECANCON: ENHANCED CAN CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| MDSEL1(1) | MDSEL0(1) | FIFOWM(2) | EWIN4 | EWIN3 | EWIN2 | EWIN1 | EWIN0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **MDSEL<1:0>:** Mode Select bits(1)
00 = Legacy mode (Mode 0, default)
01 = Enhanced Legacy mode (Mode 1)
10 = Enhanced FIFO mode (Mode 2)
11 = Reserved

bit 5 **FIFOWM:** FIFO High Water Mark bit(2)
1 = Will cause FIFO interrupt when one receive buffer remains
0 = Will cause FIFO interrupt when four receive buffers remain(3)

bit 4-0 **EWIN<4:0>:** Enhanced Window Address bits
These bits map the group of 16 banked CAN SFRs into Access Bank addresses 0F60-0F6Dh. The exact group of registers to map is determined by the binary value of these bits.
Mode 0:
**Unimplemented:** Read as '0'
Mode 1, 2:
00000 = Acceptance Filters 0, 1, 2 and BRGCON2, 3
00001 = Acceptance Filters 3, 4, 5 and BRGCON1, CIOCON
00010 = Acceptance Filter Masks, Error and Interrupt Control
00011 = Transmit Buffer 0
00100 = Transmit Buffer 1
00101 = Transmit Buffer 2
00110 = Acceptance Filters 6, 7, 8
00111 = Acceptance Filters 9, 10, 11
01000 = Acceptance Filters 12, 13, 14
01001 = Acceptance Filters 15
01010-01110 = Reserved
01111 = RXINT0, RXINT1
10000 = Receive Buffer 0
10001 = Receive Buffer 1
10010 = TX/RX Buffer 0
10011 = TX/RX Buffer 1
10100 = TX/RX Buffer 2
10101 = TX/RX Buffer 3
10110 = TX/RX Buffer 4
10111 = TX/RX Buffer 5
11000-11111 = Reserved

**Note 1:** These bits can only be changed in Configuration mode. See Register 24-1 to change to Configuration mode.
**2:** This bit is used in Mode 2 only.
**3:** If FIFO is configured to contain four or less buffers, then the FIFO interrupt will trigger.


© 2009 Microchip Technology Inc.

**REGISTER 24-4: COMSTAT: COMMUNICATION STATUS REGISTER**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 0** | R/C-0 | R/C-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| | RXB0OVFL | RXB1OVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 1** | R/C-0 | R/C-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| | — | RXBnOVFL | TXB0 | TXBP | RXBP | TXWARN | RXWARN | EWARN |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 2** | R/C-0 | R/C-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| | $\overline{\text{FIFOEMPTY}}$ | RXBnOVFL | TXBO | TXBP | RXBP | TXWARN | RXWARN | EWARN |
| | bit 7 | | | | | | | bit 0 |

| | | | |
|---|---|---|---|
| **Legend:** | C = Clearable bit | | |
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| -n = Value at POR | ‘1’ = Bit is set | ‘0’ = Bit is cleared | x = Bit is unknown |

bit 7 Mode 0:
**RXB0OVFL:** Receive Buffer 0 Overflow bit
1 = Receive Buffer 0 overflowed
0 = Receive Buffer 0 has not overflowed
Mode 1:
**Unimplemented:** Read as ‘0’
Mode 2:
**$\overline{\text{FIFOEMPTY}}$:** FIFO Not Empty bit
1 = Receive FIFO is not empty
0 = Receive FIFO is empty

bit 6 Mode 0:
**RXB1OVFL:** Receive Buffer 1 Overflow bit
1 = Receive Buffer 1 overflowed
0 = Receive Buffer 1 has not overflowed
Mode 1, 2:
**RXBnOVFL:** Receive Buffer n Overflow bit
1 = Receive Buffer n has overflowed
0 = Receive Buffer n has not overflowed

bit 5 **TXBO:** Transmitter Bus-Off bit
1 = Transmit error counter > 255
0 = Transmit error counter ≤ 255

bit 4 **TXBP:** Transmitter Bus Passive bit
1 = Transmit error counter > 127
0 = Transmit error counter ≤ 127

bit 3 **RXBP:** Receiver Bus Passive bit
1 = Receive error counter > 127
0 = Receive error counter ≤ 127

bit 2 **TXWARN:** Transmitter Warning bit
1 = Transmit error counter > 95
0 = Transmit error counter ≤ 95

bit 1 **RXWARN:** Receiver Warning bit
1 = 127 ≥ Receive error counter > 95
0 = Receive error counter ≤ 95

bit 0 **EWARN:** Error Warning bit
This bit is a flag of the RXWARN and TXWARN bits.
1 = The RXWARN or the TXWARN bits are set
0 = Neither the RXWARN or the TXWARN bits are set

### 24.2.2 DEDICATED CAN TRANSMIT BUFFER REGISTERS

This section describes the dedicated CAN Transmit Buffer registers and their associated control registers.

**REGISTER 24-5: TXBnCON: TRANSMIT BUFFER n CONTROL REGISTERS [0 ≤ n ≤ 2]**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 0** | U-0 | R-0 | R-0 | R-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
| | — | TXABT(1) | TXLARB(1) | TXERR(1) | TXREQ(2) | — | TXPRI1(3) | TXPRI0(3) |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 1,2** | R/C-0 | R-0 | R-0 | R-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
| | TXBIF | TXABT(1) | TXLARB(1) | TXERR(1) | TXREQ(2) | — | TXPRI1(3) | TXPRI0(3) |
| | bit 7 | | | | | | | bit 0 |

| | | | |
|---|---|---|---|
| **Legend:** | C = Clearable bit | | |
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 Mode 0:
**Unimplemented:** Read as '0'
Mode 1, 2:
**TXBIF:** Transmit Buffer Interrupt Flag bit
1 = Transmit buffer has completed transmission of message and may be reloaded
0 = Transmit buffer has not completed transmission of a message

bit 6 **TXABT:** Transmission Aborted Status bit(1)
1 = Message was aborted
0 = Message was not aborted

bit 5 **TXLARB:** Transmission Lost Arbitration Status bit(1)
1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent

bit 4 **TXERR:** Transmission Error Detected Status bit(1)
1 = A bus error occurred while the message was being sent
0 = A bus error did not occur while the message was being sent

bit 3 **TXREQ:** Transmit Request Status bit(2)
1 = Requests sending a message. Clears the TXABT, TXLARB and TXERR bits.
0 = Automatically cleared when the message is successfully sent

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

bit 1-0 **TXPRI<1:0>:** Transmit Priority bits(3)
11 = Priority Level 3 (highest priority)
10 = Priority Level 2
01 = Priority Level 1
00 = Priority Level 0 (lowest priority)

**Note 1:** This bit is automatically cleared when TXREQ is set.
**2:** While TXREQ is set, Transmit Buffer registers remain read-only. Clearing this bit in software while the bit is set will request a message abort.
**3:** These bits define the order in which transmit buffers will be transferred. They do not alter the CAN message identifier.


© 2009 Microchip Technology Inc.

**REGISTER 24-6: TXBnSIDH: TRANSMIT BUFFER n STANDARD IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **SID<10:3>:** Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0)
Extended Identifier bits EID<28:21> (if EXIDE = 1).

**REGISTER 24-7: TXBnSIDL: TRANSMIT BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 2]**

| R/W-x | R/W-x | R/W-x | U-0 | R/W-x | U-0 | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **SID<2:0>:** Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0)
Extended Identifier bits EID<20:18> (if EXIDE = 1).

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

bit 3 **EXIDE:** Extended Identifier Enable bit
1 = Message will transmit extended ID, SID<10:0> become EID<28:18>
0 = Message will transmit standard ID, EID<17:0> are ignored

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

bit 1-0 **EID<17:16>:** Extended Identifier bits

**REGISTER 24-8: TXBnEIDH: TRANSMIT BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<15:8>:** Extended Identifier bits (not used when transmitting standard identifier message)

**REGISTER 24-9: TXBnEIDL: TRANSMIT BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 2]**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<7:0>:** Extended Identifier bits (not used when transmitting standard identifier message)

**REGISTER 24-10: TXBnDm: TRANSMIT BUFFER n DATA FIELD BYTE m REGISTERS [0 ≤ n ≤ 2, 0 ≤ m ≤ 7]**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| TXBnDm7 | TXBnDm6 | TXBnDm5 | TXBnDm4 | TXBnDm3 | TXBnDm2 | TXBnDm1 | TXBnDm0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **TXBnDm<7:0>:** Transmit Buffer n Data Field Byte m bits (where 0 ≤ n < 3 and 0 ≤ m < 8)

Each transmit buffer has an array of registers. For example, Transmit Buffer 0 has 7 registers: TXB0D0 to TXB0D7.


© 2009 Microchip Technology Inc.

**REGISTER 24-11: TXBnDLC: TRANSMIT BUFFER n DATA LENGTH CODE REGISTERS [0 ≤ n ≤ 2]**

| U-0 | R/W-x | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **TXRTR:** Transmit Remote Frame Transmission Request bit
1 = Transmitted message will have TXRTR bit set
0 = Transmitted message will have TXRTR bit cleared

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **DLC<3:0>:** Data Length Code bits
1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = Reserved
1011 = Reserved
1010 = Reserved
1001 = Reserved
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

**REGISTER 24-12: TXERRCNT: TRANSMIT ERROR COUNT REGISTER**

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| TEC7 | TEC6 | TEC5 | TEC4 | TEC3 | TEC2 | TEC1 | TEC0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **TEC<7:0>:** Transmit Error Counter bits
This register contains a value which is derived from the rate at which errors occur. When the error count overflows, the bus-off state occurs. When the bus has 128 occurrences of 11 consecutive recessive bits, the counter value is cleared.

**EXAMPLE 24-3: TRANSMITTING A CAN MESSAGE USING BANKED METHOD**

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank. And since we want banked method, we need to make sure
; that correct bank is selected.
BANKSEL  TXB0CON                    ; One BANKSEL in beginning will make sure that we are
                                    ; in correct bank for rest of the buffer access.
; Now load transmit data into TXB0 buffer.
MOVLW    MY_DATA_BYTE1              ; Load first data byte into buffer
MOVWF    TXB0D0                     ; Compiler will automatically set "BANKED" bit
; Load rest of data bytes - up to 8 bytes into TXB0 buffer.
...
; Load message identifier
MOVLW    60H                        ; Load SID2:SID0, EXIDE = 0
MOVWF    TXB0SIDL
MOVLW    24H                        ; Load SID10:SID3
MOVWF    TXB0SIDH
; No need to load TXB0EIDL:TXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW    B'00001000'                ; Normal priority; Request transmission
MOVWF    TXB0CON

; If required, wait for message to get transmitted
BTFSC    TXB0CON, TXREQ             ; Is it transmitted?
BRA      $-2                        ; No. Continue to wait...

; Message is transmitted.
```

**EXAMPLE 24-4: TRANSMITTING A CAN MESSAGE USING WIN BITS**

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank. Use WIN bits to map it to RXB0 area.
MOVF    CANCON, W                   ; WIN bits are in lower 4 bits only. Read CANCON
                                    ; register to preserve all other bits. If operation
                                    ; mode is already known, there is no need to preserve
                                    ; other bits.
ANDLW   B'11110000'                 ; Clear WIN bits.
IORLW   B'00001000'                 ; Select Transmit Buffer 0
MOVWF   CANCON                      ; Apply the changes.
; Now TXB0 is mapped in place of RXB0. All future access to RXB0 registers will actually
; yield TXB0 register values.

; Load transmit data into TXB0 buffer.
MOVLW   MY_DATA_BYTE1               ; Load first data byte into buffer
MOVWF   RXB0D0                      ; Access TXB0D0 via RXB0D0 address.
; Load rest of the data bytes - up to 8 bytes into "TXB0" buffer using RXB0 registers.
...
; Load message identifier
MOVLW   60H                         ; Load SID2:SID0, EXIDE = 0
MOVWF   RXB0SIDL
MOVLW   24H                         ; Load SID10:SID3
MOVWF   RXB0SIDH
; No need to load RXB0EIDL:RXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW   B'00001000'                 ; Normal priority; Request transmission
MOVWF   RXB0CON

; If required, wait for message to get transmitted
BTFSC   RXB0CON, TXREQ              ; Is it transmitted?
BRA     $-2                         ; No. Continue to wait...

; Message is transmitted.
; If required, reset the WIN bits to default state.
```


© 2009 Microchip Technology Inc.

### 24.2.3 DEDICATED CAN RECEIVE BUFFER REGISTERS

This section shows the dedicated CAN Receive Buffer registers with their associated control registers.

**REGISTER 24-13: RXB0CON: RECEIVE BUFFER 0 CONTROL REGISTER**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 0** | R/C-0 | R/W-0 | R/W-0 | U-0 | R-0 | R/W-0 | R-0 | R-0 |
| | RXFUL(1) | RXM1 | RXM0 | — | RXRTRRO | RXB0DBEN | JTOFF(2) | FILHIT0 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Mode 1,2** | R/C-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| | RXFUL(1) | RXM1 | RTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 |
| | bit 7 | | | | | | | bit 0 |

| | | | |
|---|---|---|---|
| **Legend:** | C = Clearable bit | | |
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **RXFUL:** Receive Full Status bit(1)
1 = Receive buffer contains a received message
0 = Receive buffer is open to receive a new message

bit 6 Mode 0:
**RXM1:** Receive Buffer Mode bit 1 (combines with RXM0 to form RXM<1:0> bits, see bit 5)
11 = Receive all messages (including those with errors); filter criteria is ignored
10 = Receive only valid messages with extended identifier; EXIDEN in RXFnSIDL must be '1'
01 = Receive only valid messages with standard identifier; EXIDEN in RXFnSIDL must be '0'
00 = Receive all valid messages as per EXIDEN bit in the RXFnSIDL register
Mode 1, 2:
**RXM1**: Receive Buffer Mode bit 1
1 = Receive all messages (including those with errors); acceptance filters are ignored
0 = Receive all valid messages as per acceptance filters

bit 5 Mode 0:
**RXM0:** Receive Buffer Mode bit 0 (combines with RXM1 to form RXM<1:0>bits, see bit 6)
Mode 1, 2:
**RTRRO:** Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received

bit 4 Mode 0:
**Unimplemented:** Read as '0'
Mode 1, 2:
**FILHIT4:** Filter Hit bit 4
This bit combines with other bits to form filter acceptance bits<4:0>.

bit 3 Mode 0:
**RXRTRRO:** Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received
Mode 1, 2:
**FILHIT3:** Filter Hit bit 3
This bit combines with other bits to form filter acceptance bits<4:0>.

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and buffer will be considered full. After clearing the RXFUL flag, the PIR3 bit, RXB0IF, can be cleared. If RXB0IF is cleared, but RXFUL is not cleared, then RXB0IF is set again.

**2:** This bit allows same filter jump table for both RXB0CON and RXB1CON.

**REGISTER 24-13: RXB0CON: RECEIVE BUFFER 0 CONTROL REGISTER (CONTINUED)**

bit 2 Mode 0:
**RXB0DBEN:** Receive Buffer 0 Double-Buffer Enable bit
1 = Receive Buffer 0 overflow will write to Receive Buffer 1
0 = No Receive Buffer 0 overflow to Receive Buffer 1
Mode 1, 2:
**FILHIT2:** Filter Hit bit 2
This bit combines with other bits to form filter acceptance bits<4:0>.

bit 1 Mode 0:
**JTOFF:** Jump Table Offset bit (read-only copy of RXB0DBEN)**(2)**
1 = Allows jump table offset between 6 and 7
0 = Allows jump table offset between 1 and 0
Mode 1, 2:
**FILHIT1:** Filter Hit bit 1
This bit combines with other bits to form filter acceptance bits<4:0>.

bit 0 Mode 0:
**FILHIT0:** Filter Hit bit 0
This bit indicates which acceptance filter enabled the message reception into Receive Buffer 0.
1 = Acceptance Filter 1 (RXF1)
0 = Acceptance Filter 0 (RXF0)
Mode 1, 2:
**FILHIT0:** Filter Hit bit 0
This bit, in combination with FILHIT<4:1>, indicates which acceptance filter enabled the message reception into this receive buffer.
01111 = Acceptance Filter 15 (RXF15)
01110 = Acceptance Filter 14 (RXF14)
...
00000 = Acceptance Filter 0 (RXF0)

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and buffer will be considered full. After clearing the RXFUL flag, the PIR3 bit, RXB0IF, can be cleared. If RXB0IF is cleared, but RXFUL is not cleared, then RXB0IF is set again.
**2:** This bit allows same filter jump table for both RXB0CON and RXB1CON.


© 2009 Microchip Technology Inc.

**REGISTER 24-14: RXB1CON: RECEIVE BUFFER 1 CONTROL REGISTER**

| Mode 0 | R/C-0 | R/W-0 | R/W-0 | U-0 | R-0 | R/W-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|---|
| | RXFUL(1) | RXM1 | RXM0 | — | RXRTRRO | FILHIT2 | FILHIT1 | FILHIT0 |

| Mode 1,2 | R/C-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|---|
| | RXFUL(1) | RXM1 | RTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | C = Clearable bit | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **RXFUL:** Receive Full Status bit(1)
1 = Receive buffer contains a received message
0 = Receive buffer is open to receive a new message

bit 6 Mode 0:
**RXM1:** Receive Buffer Mode bit 1 (combines with RXM0 to form RXM<1:0> bits, see bit 5)
11 = Receive all messages (including those with errors); filter criteria is ignored
10 = Receive only valid messages with extended identifier; EXIDEN in RXFnSIDL must be '1'
01 = Receive only valid messages with standard identifier, EXIDEN in RXFnSIDL must be '0'
00 = Receive all valid messages as per EXIDEN bit in RXFnSIDL register

Mode 1, 2:
**RXM1:** Receive Buffer Mode bit
1 = Receive all messages (including those with errors); acceptance filters are ignored
0 = Receive all valid messages as per acceptance filters

bit 5 Mode 0:
**RXM0:** Receive Buffer Mode bit 0 (combines with RXM1 to form RXM<1:0> bits, see bit 6)

Mode 1, 2:
**RTRRO:** Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received

bit 4 Mode 0:
**Unimplemented:** Read as '0'

Mode 1, 2:
**FILHIT4:** Filter Hit bit 4
This bit combines with other bits to form the filter acceptance bits<4:0>.

bit 3 Mode 0:
**RXRTRRO:** Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received

Mode 1, 2:
**FILHIT3:** Filter Hit bit 3
This bit combines with other bits to form the filter acceptance bits<4:0>.

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and buffer will be considered full.

**REGISTER 24-14: RXB1CON: RECEIVE BUFFER 1 CONTROL REGISTER (CONTINUED)**

bit 2-0 Mode 0:
**FILHIT<2:0>:** Filter Hit bits
These bits indicate which acceptance filter enabled the last message reception into Receive Buffer 1.
111 = Reserved
110 = Reserved
101 = Acceptance Filter 5 (RXF5)
100 = Acceptance Filter 4 (RXF4)
011 = Acceptance Filter 3 (RXF3)
010 = Acceptance Filter 2 (RXF2)
001 = Acceptance Filter 1 (RXF1), only possible when RXB0DBEN bit is set
000 = Acceptance Filter 0 (RXF0), only possible when RXB0DBEN bit is set

Mode 1, 2:
**FILHIT<2:0>** Filter Hit bits <2:0>
These bits, in combination with FILHIT<4:3>, indicate which acceptance filter enabled the message reception into this receive buffer.
01111 = Acceptance Filter 15 (RXF15)
01110 = Acceptance Filter 14 (RXF14)
...
00000 = Acceptance Filter 0 (RXF0)

**Note 1:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and buffer will be considered full.

**REGISTER 24-15: RXBnSIDH: RECEIVE BUFFER n STANDARD IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]**

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| -n = Value at POR | ‘1’ = Bit is set | ‘0’ = Bit is cleared | x = Bit is unknown |

bit 7-0 **SID<10:3>:** Standard Identifier bits (if EXID (RXBnSIDL<3>) = 0)
Extended Identifier bits, EID<28:21> (if EXID = 1).


© 2009 Microchip Technology Inc.

**REGISTER 24-16: RXBnSIDL: RECEIVE BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]**

| R-x | R-x | R-x | R-x | R-x | U-0 | R-x | R-x |
|---|---|---|---|---|---|---|---|
| SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **SID<2:0>:** Standard Identifier bits (if EXID = 0)
Extended Identifier bits, EID<20:18> (if EXID = 1).

bit 4 **SRR:** Substitute Remote Request bit
This bit is always '1' when EXID = 1 or equal to the value of RXRTRRO (RBXnCON<3>) when EXID = 0.

bit 3 **EXID:** Extended Identifier bit
1 = Received message is an extended data frame, SID<10:0> are EID<28:18 >
0 = Received message is a standard data frame

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

bit 1-0 **EID<17:16>:** Extended Identifier bits

**REGISTER 24-17: RXBnEIDH: RECEIVE BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]**

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<15:8>:** Extended Identifier bits

**REGISTER 24-18: RXBnEIDL: RECEIVE BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]**

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<7:0>:** Extended Identifier bits

**REGISTER 24-19: RXBnDLC: RECEIVE BUFFER n DATA LENGTH CODE REGISTERS [0 ≤ n ≤ 1]**

| U-0 | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **RXRTR:** Receiver Remote Transmission Request bit
1 = Remote transfer request
0 = No remote transfer request

bit 5 **RB1:** Reserved bit 1
Reserved by CAN Spec and read as '0'.

bit 4 **RB0:** Reserved bit 0
Reserved by CAN Spec and read as '0'.

bit 3-0 **DLC<3:0>:** Data Length Code bits
1111 = Invalid
1110 = Invalid
1101 = Invalid
1100 = Invalid
1011 = Invalid
1010 = Invalid
1001 = Invalid
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

**REGISTER 24-20: RXBnDm: RECEIVE BUFFER n DATA FIELD BYTE m REGISTERS [0 ≤ n ≤ 1, 0 ≤ m ≤ 7]**

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| RXBnDm7 | RXBnDm6 | RXBnDm5 | RXBnDm4 | RXBnDm3 | RXBnDm2 | RXBnDm1 | RXBnDm0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **RXBnDm<7:0>:** Receive Buffer n Data Field Byte m bits (where 0 ≤ n < 1 and 0 < m < 7)
Each receive buffer has an array of registers. For example, Receive Buffer 0 has 8 registers: RXB0D0 to RXB0D7.


© 2009 Microchip Technology Inc.

**REGISTER 24-21: RXERRCNT: RECEIVE ERROR COUNT REGISTER**

| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| REC7 | REC6 | REC5 | REC4 | REC3 | REC2 | REC1 | REC0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **REC<7:0>:** Receive Error Counter bits

This register contains the receive error value as defined by the CAN specifications. When RXERRCNT > 127, the module will go into an error-passive state. RXERRCNT does not have the ability to put the module in "bus-off" state.

**EXAMPLE 24-5: READING A CAN MESSAGE**

```
; Need to read a pending message from RXB0 buffer.
; To receive any message, filter, mask and RXM1:RXM0 bits in RXB0CON registers must be
; programmed correctly.
;
; Make sure that there is a message pending in RXB0.
BTFSS   RXB0CON, RXFUL                ; Does RXB0 contain a message?
BRA     NoMessage                     ; No. Handle this situation...
; We have verified that a message is pending in RXB0 buffer.
; If this buffer can receive both Standard or Extended Identifier messages,
; identify type of message received.
BTFSS   RXB0SIDL, EXID                ; Is this Extended Identifier?
BRA     StandardMessage               ; No. This is Standard Identifier message.
                                      ; Yes. This is Extended Identifier message.
; Read all 29-bits of Extended Identifier message.
...
; Now read all data bytes
MOVFF   RXB0DO, MY_DATA_BYTE1
...
; Once entire message is read, mark the RXB0 that it is read and no longer FULL.
BCF     RXB0CON, RXFUL                ; This will allow CAN Module to load new messages
                                      ; into this buffer.
...
```

#### 24.2.3.1 Programmable TX/RX and Auto-RTR Buffers

The ECAN module contains 6 message buffers that can be programmed as transmit or receive buffers. Any of these buffers can also be programmed to automatically handle RTR messages.

| **Note:** | These registers are not used in Mode 0. |
|---|---|

**REGISTER 24-22: BnCON: TX/RX BUFFER n CONTROL REGISTERS IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0](1)**

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| RXFUL(2) | RXM1 | RXRTRRO | FILHIT4 | FILHIT3 | FILHIT2 | FILHIT1 | FILHIT0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **RXFUL:** Receive Full Status bit(2)
1 = Receive buffer contains a received message
0 = Receive buffer is open to receive a new message

bit 6 **RXM1:** Receive Buffer Mode bit
1 = Receive all messages including partial and invalid (acceptance filters are ignored)
0 = Receive all valid messages as per acceptance filters

bit 5 **RXRTRRO:** Read-Only Remote Transmission Request for Received Message bit
1 = Received message is a remote transmission request
0 = Received message is not a remote transmission request

bit 4-0 **FILHIT<4:0>:** Filter Hit bits
These bits indicate which acceptance filter enabled the last message reception into this buffer.
01111 = Acceptance Filter 15 (RXF15)
01110 = Acceptance Filter 14 (RXF14)
...
00001 = Acceptance Filter 1 (RXF1)
00000 = Acceptance Filter 0 (RXF0)

**Note 1:** These registers are available in Mode 1 and 2 only.
**2:** This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and the buffer will be considered full.


© 2009 Microchip Technology Inc.

**REGISTER 24-23: BnCON: TX/RX BUFFER n CONTROL REGISTERS IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1](1)**

| R/W-0 | R-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| TXBIF(3) | TXABT(3) | TXLARB(3) | TXERR(3) | TXREQ(2,4) | RTREN | TXPRI1(5) | TXPRI0(5) |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **TXBIF:** Transmit Buffer Interrupt Flag bit(3)
1 = A message is successfully transmitted
0 = No message was transmitted

bit 6 **TXABT:** Transmission Aborted Status bit(3)
1 = Message was aborted
0 = Message was not aborted

bit 5 **TXLARB:** Transmission Lost Arbitration Status bit(3)
1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent

bit 4 **TXERR:** Transmission Error Detected Status bit(3)
1 = A bus error occurred while the message was being sent
0 = A bus error did not occur while the message was being sent

bit 3 **TXREQ:** Transmit Request Status bit(2,4)
1 = Requests sending a message; clears the TXABT, TXLARB and TXERR bits
0 = Automatically cleared when the message is successfully sent

bit 2 **RTREN:** Automatic Remote Transmission Request Enable bit
1 = When a remote transmission request is received, TXREQ will be automatically set
0 = When a remote transmission request is received, TXREQ will be unaffected

bit 1-0 **TXPRI<1:0>:** Transmit Priority bits(5)
11 = Priority Level 3 (highest priority)
10 = Priority Level 2
01 = Priority Level 1
00 = Priority Level 0 (lowest priority)

**Note 1:** These registers are available in Mode 1 and 2 only.
**2:** Clearing this bit in software while the bit is set will request a message abort.
**3:** This bit is automatically cleared when TXREQ is set.
**4:** While TXREQ is set or a transmission is in progress, Transmit Buffer registers remain read-only.
**5:** These bits set the order in which the Transmit Buffer register will be transferred. They do not alter the CAN message identifier.

**REGISTER 24-24: BnSIDH: TX/RX BUFFER n STANDARD IDENTIFIER REGISTERS, HIGH BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0](1)**

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **SID<10:3>:** Standard Identifier bits (if EXIDE (BnSIDL<3>) = 0)
Extended Identifier bits, EID<28:21> (if EXIDE = 1).

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 24-25: BnSIDH: TX/RX BUFFER n STANDARD IDENTIFIER REGISTERS, HIGH BYTE IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1](1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **SID<10:3>:** Standard Identifier bits (if EXIDE (BnSIDL<3>) = 0)
Extended Identifier bits, EID<28:21> (if EXIDE = 1).

**Note 1:** These registers are available in Mode 1 and 2 only.


© 2009 Microchip Technology Inc.

**REGISTER 24-26: BnSIDL: TX/RX BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0](1)**

| R-x | R-x | R-x | R-x | R-x | U-0 | R-x | R-x |
|---|---|---|---|---|---|---|---|
| SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **SID<2:0>:** Standard Identifier bits (if EXID = 0)
Extended Identifier bits, EID<20:18> (if EXID = 1).

bit 4 **SRR:** Substitute Remote Transmission Request bit
This bit is always '1' when EXID = 1 or equal to the value of RXRTRRO (BnCON<5>) when EXID = 0.

bit 3 **EXID:** Extended Identifier Enable bit
1 = Received message is an extended identifier frame (SID<10:0> are EID<28:18>)
0 = Received message is a standard identifier frame

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

bit 1-0 **EID<17:16>:** Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 24-27: BnSIDL: TX/RX BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1](1)**

| R/W-x | R/W-x | R/W-x | U-0 | R/W-x | U-0 | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **SID<2:0>:** Standard Identifier bits (if EXIDE = 0)
Extended Identifier bits, EID<20:18> (if EXIDE = 1).

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

bit 3 **EXIDE:** Extended Identifier Enable bit
1 = Received message is an extended identifier frame (SID<10:0> are EID<28:18>)
0 = Received message is a standard identifier frame

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

bit 1-0 **EID<17:16>:** Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 24-28: BnEIDH: TX/RX BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0](1)**

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<15:8>:** Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 24-29: BnEIDH: TX/RX BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1](1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<15:8>:** Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 24-30: BnEIDL: TX/RX BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 0](1)**

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<7:0>:** Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.


© 2009 Microchip Technology Inc.

**REGISTER 24-31: BnEIDL: TX/RX BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 1](1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<7:0>:** Extended Identifier bits

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 24-32: BnDm: TX/RX BUFFER n DATA FIELD BYTE m REGISTERS IN RECEIVE MODE [0 ≤ n ≤ 5, 0 ≤ m ≤ 7, TXnEN (BSEL<n>) = 0](1)**

| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| BnDm7 | BnDm6 | BnDm5 | BnDm4 | BnDm3 | BnDm2 | BnDm1 | BnDm0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **BnDm<7:0>:** Receive Buffer n Data Field Byte m bits (where 0 ≤ n < 3 and 0 < m < 8)
Each receive buffer has an array of registers. For example, Receive Buffer 0 has 7 registers: B0D0 to B0D7.

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 24-33: BnDm: TX/RX BUFFER n DATA FIELD BYTE m REGISTERS IN TRANSMIT MODE [0 ≤ n ≤ 5, 0 ≤ m ≤ 7, TXnEN (BSEL<n>) = 1](1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| BnDm7 | BnDm6 | BnDm5 | BnDm4 | BnDm3 | BnDm2 | BnDm1 | BnDm0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **BnDm<7:0>:** Transmit Buffer n Data Field Byte m bits (where 0 ≤ n < 3 and 0 < m < 8)
Each transmit buffer has an array of registers. For example, Transmit Buffer 0 has 7 registers: TXB0D0 to TXB0D7.

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 24-34: BnDLC: TX/RX BUFFER n DATA LENGTH CODE REGISTERS IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 0](1)**

| U-0 | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
|---|---|---|---|---|---|---|---|
| — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **RXRTR:** Receiver Remote Transmission Request bit
1 = This is a remote transmission request
0 = This is not a remote transmission request

bit 5 **RB1:** Reserved bit 1
Reserved by CAN Spec and read as '0'.

bit 4 **RB0:** Reserved bit 0
Reserved by CAN Spec and read as '0'.

bit 3-0 **DLC<3:0>:** Data Length Code bits
1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = Reserved
1011 = Reserved
1010 = Reserved
1001 = Reserved
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

**Note 1:** These registers are available in Mode 1 and 2 only.


© 2009 Microchip Technology Inc.

**REGISTER 24-35: BnDLC: TX/RX BUFFER n DATA LENGTH CODE REGISTERS IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 1](1)**

| U-0 | R/W-x | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 6 **TXRTR:** Transmitter Remote Transmission Request bit
1 = Transmitted message will have RTR bit set
0 = Transmitted message will have RTR bit cleared

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **DLC<3:0>:** Data Length Code bits
1111-1001 = Reserved
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

**Note 1:** These registers are available in Mode 1 and 2 only.

**REGISTER 24-36: BSEL0: BUFFER SELECT REGISTER 0(1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| B5TXEN | B4TXEN | B3TXEN | B2TXEN | B1TXEN | B0TXEN | — | — |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-2 **B5TXEN:B0TXEN:** Buffer 5 to Buffer 0 Transmit Enable bit
1 = Buffer is configured in Transmit mode
0 = Buffer is configured in Receive mode

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

**Note 1:** These registers are available in Mode 1 and 2 only.

#### 24.2.3.2 Message Acceptance Filters and Masks

This section describes the message acceptance filters and masks for the CAN receive buffers.

**REGISTER 24-37: RXFnSIDH: RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15](1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **SID<10:3>:** Standard Identifier Filter bits (if EXIDEN = 0)
Extended Identifier Filter bits, EID<28:21> (if EXIDEN = 1).

**Note 1:** Registers, RXF6SIDH:RXF15SIDH, are available in Mode 1 and 2 only.

**REGISTER 24-38: RXFnSIDL: RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER REGISTERS, LOW BYTE [0 ≤ n ≤ 15](1)**

| R/W-x | R/W-x | R/W-x | U-0 | R/W-x | U-0 | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID2 | SID1 | SID0 | — | EXIDEN(2) | — | EID17 | EID16 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **SID<2:0>:** Standard Identifier Filter bits (if EXIDEN = 0)
Extended Identifier Filter bits, EID<20:18> (if EXIDEN = 1).

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

bit 3 **EXIDEN:** Extended Identifier Filter Enable bit(2)
1 = Filter will only accept extended ID messages
0 = Filter will only accept standard ID messages

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

bit 1-0 **EID<17:16>:** Extended Identifier Filter bits

**Note 1:** Registers, RXF6SIDL:RXF15SIDL, are available in Mode 1 and 2 only.
**2:** In Mode 0, this bit must be set/cleared as required, irrespective of corresponding mask register value.


© 2009 Microchip Technology Inc.

**REGISTER 24-39: RXFnEIDH: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15](1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<15:8>:** Extended Identifier Filter bits

**Note 1:** Registers, RXF6EIDH:RXF15EIDH, are available in Mode 1 and 2 only.

**REGISTER 24-40: RXFnEIDL: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 15](1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<7:0>:** Extended Identifier Filter bits

**Note 1:** Registers, RXF6EIDL:RXF15EIDL, are available in Mode 1 and 2 only.

**REGISTER 24-41: RXMnSIDH: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **SID<10:3>:** Standard Identifier Mask bits or Extended Identifier Mask bits (EID<28:21>)

**REGISTER 24-42: RXMnSIDL: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]**

| R/W-x | R/W-x | R/W-x | U-0 | R/W-0 | U-0 | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| SID2 | SID1 | SID0 | — | EXIDEN(1) | — | EID17 | EID16 |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **SID<2:0>:** Standard Identifier Mask bits or Extended Identifier Mask bits (EID<20:18>)

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

bit 3 Mode 0:
**Unimplemented:** Read as '0'
Mode 1, 2:
**EXIDEN:** Extended Identifier Filter Enable Mask bit(1)
1 = Messages selected by the EXIDEN bit in RXFnSIDL will be accepted
0 = Both standard and extended identifier messages will be accepted

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

bit 1-0 **EID<17:16>:** Extended Identifier Mask bits

**Note 1:** This bit is available in Mode 1 and 2 only.

**REGISTER 24-43: RXMnEIDH: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<15:8>:** Extended Identifier Mask bits

**REGISTER 24-44: RXMnEIDL: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **EID<7:0>:** Extended Identifier Mask bits


© 2009 Microchip Technology Inc.

**REGISTER 24-45: RXFCONn: RECEIVE FILTER CONTROL REGISTER n [0 ≤ n ≤ 1](1)**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **RXFCON0** | R/W-0 | R/W-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| | RXF7EN | RXF6EN | RXF5EN | RXF4EN | RXF3EN | RXF2EN | RXF1EN | RXF0EN |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **RXFCON1** | R/W-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | RXF15EN | RXF14EN | RXF13EN | RXF12EN | RXF11EN | RXF10EN | RXF9EN | RXF8EN |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-0 **RXFnEN:** Receive Filter n Enable bits
0 = Filter is disabled
1 = Filter is enabled

**Note 1:** This register is available in Mode 1 and 2 only.

**Note:** Register 24-46 through Register 24-51 are writable in Configuration mode only.

**REGISTER 24-46: SDFLC: STANDARD DATA BYTES FILTER LENGTH COUNT REGISTER(1)**

| U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | FLC4 | FLC3 | FLC2 | FLC1 | FLC0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 4-0 **FLC<4:0>:** Filter Length Count bits

Mode 0:

Not used; forced to '00000'.

00000-10010 = 0 18 bits are available for standard data byte filter. Actual number of bits used depends on the DLC<3:0> bits (RXBnDLC<3:0> or BnDLC<3:0> if configured as RX buffer) of the message being received.

If DLC<3:0> = 0000 No bits will be compared with incoming data bits.

If DLC<3:0> = 0001 Up to 8 data bits of RXFnEID<7:0>, as determined by FLC<2:0>, will be compared with the corresponding number of data bits of the incoming message.

If DLC<3:0> = 0010 Up to 16 data bits of RXFnEID<15:0>, as determined by FLC<3:0>, will be compared with the corresponding number of data bits of the incoming message.

If DLC<3:0> = 0011 Up to 18 data bits of RXFnEID<17:0>, as determined by FLC<4:0>, will be compared with the corresponding number of data bits of the incoming message.

**Note 1:** This register is available in Mode 1 and 2 only.

**REGISTER 24-47: RXFBCONn: RECEIVE FILTER BUFFER CONTROL REGISTER n(1)**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **RXFBCON0** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | F1BP_3 | F1BP_2 | F1BP_1 | F1BP_0 | F0BP_3 | F0BP_2 | F0BP_1 | F0BP_0 |
| **RXFBCON1** | R/W-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-1 |
| | F3BP_3 | F3BP_2 | F3BP_1 | F3BP_0 | F2BP_3 | F2BP_2 | F2BP_1 | F2BP_0 |
| **RXFBCON2** | R/W-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-1 |
| | F5BP_3 | F5BP_2 | F5BP_1 | F5BP_0 | F4BP_3 | F4BP_2 | F4BP_1 | F4BP_0 |
| **RXFBCON3** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | F7BP_3 | F7BP_2 | F7BP_1 | F7BP_0 | F6BP_3 | F6BP_2 | F6BP_1 | F6BP_0 |
| **RXFBCON4** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | F9BP_3 | F9BP_2 | F9BP_1 | F9BP_0 | F8BP_3 | F8BP_2 | F8BP_1 | F8BP_0 |
| **RXFBCON5** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | F11BP_3 | F11BP_2 | F11BP_1 | F11BP_0 | F10BP_3 | F10BP_2 | F10BP_1 | F10BP_0 |
| **RXFBCON6** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | F13BP_3 | F13BP_2 | F13BP_1 | F13BP_0 | F12BP_3 | F12BP_2 | F12BP_1 | F12BP_0 |
| **RXFBCON7** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| | F15BP_3 | F15BP_2 | F15BP_1 | F15BP_0 | F14BP_3 | F14BP_2 | F14BP_1 | F14BP_0 |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| -n = Value at POR | ‘1’ = Bit is set | ‘0’ = Bit is cleared | x = Bit is unknown |

bit 7-0 **FnBP_<3:0>:** Filter n Buffer Pointer Nibble bits

0000 = Filter n is associated with RXB0
0001 = Filter n is associated with RXB1
0010 = Filter n is associated with B0
0011 = Filter n is associated with B1
...
0111 = Filter n is associated with B5
1111-1000 = Reserved

**Note 1:** This register is available in Mode 1 and 2 only.


© 2009 Microchip Technology Inc.

**REGISTER 24-48: MSEL0: MASK SELECT REGISTER 0(1)**

| R/W-0 | R/W-1 | R/W-0 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FIL3_1 | FIL3_0 | FIL2_1 | FIL2_0 | FIL1_1 | FIL1_0 | FIL0_1 | FIL0_0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **FIL3_<1:0>:** Filter 3 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL2_<1:0>:** Filter 2 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL1_<1:0>:** Filter 1 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL0_<1:0>:** Filter 0 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.

**REGISTER 24-49: MSEL1: MASK SELECT REGISTER 1(1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-1 |
|---|---|---|---|---|---|---|---|
| FIL7_1 | FIL7_0 | FIL6_1 | FIL6_0 | FIL5_1 | FIL5_0 | FIL4_1 | FIL4_0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **FIL7_<1:0>:** Filter 7 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL6_<1:0>:** Filter 6 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL5_<1:0>:** Filter 5 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL4_<1:0>:** Filter 4 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.


© 2009 Microchip Technology Inc.

**REGISTER 24-50: MSEL2: MASK SELECT REGISTER 2(1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FIL11_1 | FIL11_0 | FIL10_1 | FIL10_0 | FIL9_1 | FIL9_0 | FIL8_1 | FIL8_0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **FIL11_<1:0>:** Filter 11 Select bits 1 and 0
- 11 = No mask
- 10 = Filter 15
- 01 = Acceptance Mask 1
- 00 = Acceptance Mask 0

bit 5-4 **FIL10_<1:0>:** Filter 10 Select bits 1 and 0
- 11 = No mask
- 10 = Filter 15
- 01 = Acceptance Mask 1
- 00 = Acceptance Mask 0

bit 3-2 **FIL9_<1:0>:** Filter 9 Select bits 1 and 0
- 11 = No mask
- 10 = Filter 15
- 01 = Acceptance Mask 1
- 00 = Acceptance Mask 0

bit 1-0 **FIL8_<1:0>:** Filter 8 Select bits 1 and 0
- 11 = No mask
- 10 = Filter 15
- 01 = Acceptance Mask 1
- 00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.

**REGISTER 24-51: MSEL3: MASK SELECT REGISTER 3(1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| FIL15_1 | FIL15_0 | FIL14_1 | FIL14_0 | FIL13_1 | FIL13_0 | FIL12_1 | FIL12_0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **FIL15_<1:0>:** Filter 15 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL14_<1:0>:** Filter 14 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL13_<1:0>:** Filter 13 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL12_<1:0>:** Filter 12 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

**Note 1:** This register is available in Mode 1 and 2 only.


© 2009 Microchip Technology Inc.

### 24.2.4 CAN BAUD RATE REGISTERS

This section describes the CAN Baud Rate registers.

| | |
|---|---|
| **Note:** | These registers are writable in Configuration mode only. |

**REGISTER 24-52: BRGCON1: BAUD RATE CONTROL REGISTER 1**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| -n = Value at POR | ‘1’ = Bit is set | ‘0’ = Bit is cleared | x = Bit is unknown |

bit 7-6 **SJW<1:0>:** Synchronized Jump Width bits

11 = Synchronization jump width time = 4 x TQ
10 = Synchronization jump width time = 3 x TQ
01 = Synchronization jump width time = 2 x TQ
00 = Synchronization jump width time = 1 x TQ

bit 5-0 **BRP<5:0>:** Baud Rate Prescaler bits

111111 = TQ = (2 x 64)/FOSC
111110 = TQ = (2 x 63)/FOSC
:
:
000001 = TQ = (2 x 2)/FOSC
000000 = TQ = (2 x 1)/FOSC

**REGISTER 24-53: BRGCON2: BAUD RATE CONTROL REGISTER 2**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SEG2PHTS | SAM | SEG1PH2 | SEG1PH1 | SEG1PH0 | PRSEG2 | PRSEG1 | PRSEG0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **SEG2PHTS:** Phase Segment 2 Time Select bit
1 = Freely programmable
0 = Maximum of PHEG1 or Information Processing Time (IPT), whichever is greater

bit 6 **SAM:** Sample of the CAN bus Line bit
1 = Bus line is sampled three times prior to the sample point
0 = Bus line is sampled once at the sample point

bit 5-3 **SEG1PH<2:0>:** Phase Segment 1 bits
111 = Phase Segment 1 time = 8 x TQ
110 = Phase Segment 1 time = 7 x TQ
101 = Phase Segment 1 time = 6 x TQ
100 = Phase Segment 1 time = 5 x TQ
011 = Phase Segment 1 time = 4 x TQ
010 = Phase Segment 1 time = 3 x TQ
001 = Phase Segment 1 time = 2 x TQ
000 = Phase Segment 1 time = 1 x TQ

bit 2-0 **PRSEG<2:0>:** Propagation Time Select bits
111 = Propagation time = 8 x TQ
110 = Propagation time = 7 x TQ
101 = Propagation time = 6 x TQ
100 = Propagation time = 5 x TQ
011 = Propagation time = 4 x TQ
010 = Propagation time = 3 x TQ
001 = Propagation time = 2 x TQ
000 = Propagation time = 1 x TQ


© 2009 Microchip Technology Inc.

**REGISTER 24-54: BRGCON3: BAUD RATE CONTROL REGISTER 3**

| R/W-0 | R/W-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| WAKDIS | WAKFIL | — | — | — | SEG2PH2(1) | SEG2PH1(1) | SEG2PH0(1) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **WAKDIS:** Wake-up Disable bit
1 = Disable CAN bus activity wake-up feature
0 = Enable CAN bus activity wake-up feature

bit 6 **WAKFIL:** Selects CAN bus Line Filter for Wake-up bit
1 = Use CAN bus line filter for wake-up
0 = CAN bus line filter is not used for wake-up

bit 5-3 **Unimplemented:** Read as '0'

bit 2-0 **SEG2PH<2:0>:** Phase Segment 2 Time Select bits(1)
111 = Phase Segment 2 time = 8 x TQ
110 = Phase Segment 2 time = 7 x TQ
101 = Phase Segment 2 time = 6 x TQ
100 = Phase Segment 2 time = 5 x TQ
011 = Phase Segment 2 time = 4 x TQ
010 = Phase Segment 2 time = 3 x TQ
001 = Phase Segment 2 time = 2 x TQ
000 = Phase Segment 2 time = 1 x TQ

**Note 1:** Ignored if SEG2PHTS bit (BRGCON2<7>) is '0'.

### 24.2.5 CAN MODULE I/O CONTROL REGISTER

This register controls the operation of the CAN module's I/O pins in relation to the rest of the microcontroller.

**REGISTER 24-55: CIOCON: CAN I/O CONTROL REGISTER**

| U-0 | U-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | ENDRHI(1) | CANCAP | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **ENDRHI:** Enable Drive High bit(1)
1 = CANTX pin will drive VDD when recessive
0 = CANTX pin will be tri-state when recessive

bit 4 **CANCAP:** CAN Message Receive Capture Enable bit
1 = Enable CAN capture, CAN message receive signal replaces input on RC2/CCP1
0 = Disable CAN capture, RC2/CCP1 input to CCP1 module

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

**Note 1:** Always set this bit when using a differential bus to avoid signal crosstalk in CANTX from other nearby pins.


© 2009 Microchip Technology Inc.

### 24.2.6 CAN INTERRUPT REGISTERS

The registers in this section are the same as described in **Section 10.0 "Interrupts"**. They are duplicated here for convenience.

**REGISTER 24-56: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3**

| **Mode 0** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|---|
| | IRXIF | WAKIF | ERRIF | TXB2IF | TXB1IF(1) | TXB0IF(1) | RXB1IF | RXB0IF |

| **Mode 1,2** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|---|
| | IRXIF | WAKIF | ERRIF | TXBnIF | TXB1IF(1) | TXB0IF(1) | RXBnIF | FIFOWMIF |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **IRXIF:** CAN Bus Error Message Received Interrupt Flag bit
1 = An invalid message has occurred on the CAN bus
0 = No invalid message on CAN bus

bit 6 **WAKIF:** CAN Bus Activity Wake-up Interrupt Flag bit
1 = Activity on CAN bus has occurred
0 = No activity on CAN bus

bit 5 **ERRIF:** CAN Module Error Interrupt Flag bit
1 = An error has occurred in the CAN module (multiple sources; refer to **Section 24.15.6 "Error Interrupt"**)
0 = No CAN module errors

bit 4 When CAN is in Mode 0:
**TXB2IF:** CAN Transmit Buffer 2 Interrupt Flag bit
1 = Transmit Buffer 2 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 2 has not completed transmission of a message
When CAN is in Mode 1 or 2:
**TXBnIF:** Any Transmit Buffer Interrupt Flag bit
1 = One or more transmit buffers have completed transmission of a message and may be reloaded
0 = No transmit buffer is ready for reload

bit 3 **TXB1IF:** CAN Transmit Buffer 1 Interrupt Flag bit(1)
1 = Transmit Buffer 1 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 1 has not completed transmission of a message

bit 2 **TXB0IF:** CAN Transmit Buffer 0 Interrupt Flag bit(1)
1 = Transmit Buffer 0 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 0 has not completed transmission of a message

bit 1 When CAN is in Mode 0:
**RXB1IF:** CAN Receive Buffer 1 Interrupt Flag bit
1 = Receive Buffer 1 has received a new message
0 = Receive Buffer 1 has not received a new message
When CAN is in Mode 1 or 2:
**RXBnIF:** Any Receive Buffer Interrupt Flag bit
1 = One or more receive buffers has received a new message
0 = No receive buffer has received a new message

bit 0 When CAN is in Mode 0:
**RXB0IF:** CAN Receive Buffer 0 Interrupt Flag bit
1 = Receive Buffer 0 has received a new message
0 = Receive Buffer 0 has not received a new message
When CAN is in Mode 1:
**Unimplemented:** Read as '0'
When CAN is in Mode 2:
**FIFOWMIF:** FIFO Watermark Interrupt Flag bit
1 = FIFO high watermark is reached
0 = FIFO high watermark is not reached

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.

**REGISTER 24-57: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3**

| **Mode 0** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|---|
| | IRXIE | WAKIE | ERRIE | TXB2IE | TXB1IE(1) | TXB0IE(1) | RXB1IE | RXB0IE |

| **Mode 1** | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|---|
| | IRXIE | WAKIE | ERRIE | TXBnIE | TXB1IE(1) | TXB0IE(1) | RXBnIE | FIFOWMIE |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **IRXIE:** CAN Bus Error Message Received Interrupt Enable bit
1 = Enable invalid message received interrupt
0 = Disable invalid message received interrupt

bit 6 **WAKIE:** CAN bus Activity Wake-up Interrupt Enable bit
1 = Enable bus activity wake-up interrupt
0 = Disable bus activity wake-up interrupt

bit 5 **ERRIE:** CAN bus Error Interrupt Enable bit
1 = Enable CAN module error interrupt
0 = Disable CAN module error interrupt

bit 4 When CAN is in Mode 0:
**TXB2IE:** CAN Transmit Buffer 2 Interrupt Enable bit
1 = Enable Transmit Buffer 2 interrupt
0 = Disable Transmit Buffer 2 interrupt
When CAN is in Mode 1 or 2:
**TXBnIE:** CAN Transmit Buffer Interrupts Enable bit
1 = Enable transmit buffer interrupt; individual interrupt is enabled by TXBIE and BIE0
0 = Disable all transmit buffer interrupts

bit 3 **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 1 interrupt
0 = Disable Transmit Buffer 1 interrupt

bit 2 **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 0 interrupt
0 = Disable Transmit Buffer 0 interrupt

bit 1 When CAN is in Mode 0:
**RXB1IE:** CAN Receive Buffer 1 Interrupt Enable bit
1 = Enable Receive Buffer 1 interrupt
0 = Disable Receive Buffer 1 interrupt
When CAN is in Mode 1 or 2:
**RXBnIE:** CAN Receive Buffer Interrupts Enable bit
1 = Enable receive buffer interrupt; individual interrupt is enabled by BIE0
0 = Disable all receive buffer interrupts

bit 0 When CAN is in Mode 0:
**RXB0IE:** CAN Receive Buffer 0 Interrupt Enable bit
1 = Enable Receive Buffer 0 interrupt
0 = Disable Receive Buffer 0 interrupt
When CAN is in Mode 1:
**Unimplemented:** Read as '0'
When CAN is in Mode 2:
**FIFOWMIE:** FIFO Watermark Interrupt Enable bit
1 = Enable FIFO watermark interrupt
0 = Disable FIFO watermark interrupt

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.


© 2009 Microchip Technology Inc.

**REGISTER 24-58: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3**

| Mode 0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|---|
| | IRXIP | WAKIP | ERRIP | TXB2IP | TXB1IP(1) | TXB0IP(1) | RXB1IP | RXB0IP |

| Mode 1,2 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|---|
| | IRXIP | WAKIP | ERRIP | TXBnIP | TXB1IP(1) | TXB0IP(1) | RXBnIP | FIFOWMIP |
| | bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **IRXIP:** CAN Bus Error Message Received Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **WAKIP:** CAN Bus Activity Wake-up Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **ERRIP:** CAN Module Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 When CAN is in Mode 0:
**TXB2IP:** CAN Transmit Buffer 2 Interrupt Priority bit
1 = High priority
0 = Low priority
When CAN is in Mode 1 or 2:
**TXBnIP:** CAN Transmit Buffer Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **TXB1IP:** CAN Transmit Buffer 1 Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 2 **TXB0IP:** CAN Transmit Buffer 0 Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 1 When CAN is in Mode 0:
**RXB1IP:** CAN Receive Buffer 1 Interrupt Priority bit
1 = High priority
0 = Low priority
When CAN is in Mode 1 or 2:
**RXBnIP:** CAN Receive Buffer Interrupts Priority bit
1 = High priority
0 = Low priority

bit 0 When CAN is in Mode 0:
**RXB0IP:** CAN Receive Buffer 0 Interrupt Priority bit
1 = High priority
0 = Low priority
When CAN is in Mode 1:
**Unimplemented:** Read as '0'
When CAN is in Mode 2:
**FIFOWMIP:** FIFO Watermark Interrupt Priority bit
1 = High priority
0 = Low priority

**Note 1:** In CAN Mode 1 and 2, these bits are forced to '0'.

**REGISTER 24-59: TXBIE: TRANSMIT BUFFERS INTERRUPT ENABLE REGISTER(1)**

| U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | TXB2IE(2) | TXB1IE(2) | TXB0IE(2) | — | — |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

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

bit 4-2 **TXB2IE:TXB0IE:** Transmit Buffer 2-0 Interrupt Enable bits(2)
1 = Transmit buffer interrupt is enabled
0 = Transmit buffer interrupt is disabled

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

**Note 1:** This register is available in Mode 1 and 2 only.
**2:** TXBnIE in PIE3 register must be set to get an interrupt.

**REGISTER 24-60: BIE0: BUFFER INTERRUPT ENABLE REGISTER 0(1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| B5IE(2) | B4IE(2) | B3IE(2) | B2IE(2) | B1IE(2) | B0IE(2) | RXB1IE(2) | RXB0IE(2) |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-2 **B5IE:B0IE:** Programmable Transmit/Receive Buffer 5-0 Interrupt Enable bits(2)
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 1-0 **RXB1IE:RXB0IE:** Dedicated Receive Buffer 1-0 Interrupt Enable bits(2)
1 = Interrupt is enabled
0 = Interrupt is disabled

**Note 1:** This register is available in Mode 1 and 2 only.
**2:** Either TXBnIE or RXBnIE in the PIE3 register must be set to get an interrupt.


© 2009 Microchip Technology Inc.

**TABLE 24-1: CAN CONTROLLER REGISTER MAP**

| Address[1] | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| F7Fh | SPBRGH[3] | F5Fh | CANCON_RO0 | F3Fh | CANCON_RO2 | F1Fh | RXM1EIDL |
| F7Eh | BAUDCON[3] | F5Eh | CANSTAT_RO0 | F3Eh | CANSTAT_RO2 | F1Eh | RXM1EIDH |
| F7Dh | —[4] | F5Dh | RXB1D7 | F3Dh | TXB1D7 | F1Dh | RXM1SIDL |
| F7Ch | —[4] | F5Ch | RXB1D6 | F3Ch | TXB1D6 | F1Ch | RXM1SIDH |
| F7Bh | —[4] | F5Bh | RXB1D5 | F3Bh | TXB1D5 | F1Bh | RXM0EIDL |
| F7Ah | —[4] | F5Ah | RXB1D4 | F3Ah | TXB1D4 | F1Ah | RXM0EIDH |
| F79h | ECCP1DEL[3] | F59h | RXB1D3 | F39h | TXB1D3 | F19h | RXM0SIDL |
| F78h | —[4] | F58h | RXB1D2 | F38h | TXB1D2 | F18h | RXM0SIDH |
| F77h | ECANCON | F57h | RXB1D1 | F37h | TXB1D1 | F17h | RXF5EIDL |
| F76h | TXERRCNT | F56h | RXB1D0 | F36h | TXB1D0 | F16h | RXF5EIDH |
| F75h | RXERRCNT | F55h | RXB1DLC | F35h | TXB1DLC | F15h | RXF5SIDL |
| F74h | COMSTAT | F54h | RXB1EIDL | F34h | TXB1EIDL | F14h | RXF5SIDH |
| F73h | CIOCON | F53h | RXB1EIDH | F33h | TXB1EIDH | F13h | RXF4EIDL |
| F72h | BRGCON3 | F52h | RXB1SIDL | F32h | TXB1SIDL | F12h | RXF4EIDH |
| F71h | BRGCON2 | F51h | RXB1SIDH | F31h | TXB1SIDH | F11h | RXF4SIDL |
| F70h | BRGCON1 | F50h | RXB1CON | F30h | TXB1CON | F10h | RXF4SIDH |
| F6Fh | CANCON | F4Fh | CANCON_RO1[2] | F2Fh | CANCON_RO3[2] | F0Fh | RXF3EIDL |
| F6Eh | CANSTAT | F4Eh | CANSTAT_RO1[2] | F2Eh | CANSTAT_RO3[2] | F0Eh | RXF3EIDH |
| F6Dh | RXB0D7 | F4Dh | TXB0D7 | F2Dh | TXB2D7 | F0Dh | RXF3SIDL |
| F6Ch | RXB0D6 | F4Ch | TXB0D6 | F2Ch | TXB2D6 | F0Ch | RXF3SIDH |
| F6Bh | RXB0D5 | F4Bh | TXB0D5 | F2Bh | TXB2D5 | F0Bh | RXF2EIDL |
| F6Ah | RXB0D4 | F4Ah | TXB0D4 | F2Ah | TXB2D4 | F0Ah | RXF2EIDH |
| F69h | RXB0D3 | F49h | TXB0D3 | F29h | TXB2D3 | F09h | RXF2SIDL |
| F68h | RXB0D2 | F48h | TXB0D2 | F28h | TXB2D2 | F08h | RXF2SIDH |
| F67h | RXB0D1 | F47h | TXB0D1 | F27h | TXB2D1 | F07h | RXF1EIDL |
| F66h | RXB0D0 | F46h | TXB0D0 | F26h | TXB2D0 | F06h | RXF1EIDH |
| F65h | RXB0DLC | F45h | TXB0DLC | F25h | TXB2DLC | F05h | RXF1SIDL |
| F64h | RXB0EIDL | F44h | TXB0EIDL | F24h | TXB2EIDL | F04h | RXF1SIDH |
| F63h | RXB0EIDH | F43h | TXB0EIDH | F23h | TXB2EIDH | F03h | RXF0EIDL |
| F62h | RXB0SIDL | F42h | TXB0SIDL | F22h | TXB2SIDL | F02h | RXF0EIDH |
| F61h | RXB0SIDH | F41h | TXB0SIDH | F21h | TXB2SIDH | F01h | RXF0SIDL |
| F60h | RXB0CON | F40h | TXB0CON | F20h | TXB2CON | F00h | RXF0SIDH |

**Note 1:** Shaded registers are available in Access Bank low area, while the rest are available in Bank 15.

**2:** CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

**3:** These registers are not CAN registers.

**4:** Unimplemented registers are read as '0'.

**TABLE 24-1: CAN CONTROLLER REGISTER MAP (CONTINUED)**

| Address[1] | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| EFFh | —[4] | EDFh | —[4] | EBFh | —[4] | E9Fh | —[4] |
| EFEh | —[4] | EDEh | —[4] | EBEh | —[4] | E9Eh | —[4] |
| EFDh | —[4] | EDDh | —[4] | EBDh | —[4] | E9Dh | —[4] |
| EFCh | —[4] | EDCh | —[4] | EBCh | —[4] | E9Ch | —[4] |
| EFBh | —[4] | EDBh | —[4] | EBBh | —[4] | E9Bh | —[4] |
| EFAh | —[4] | EDAh | —[4] | EBAh | —[4] | E9Ah | —[4] |
| EF9h | —[4] | ED9h | —[4] | EB9h | —[4] | E99h | —[4] |
| EF8h | —[4] | ED8h | —[4] | EB8h | —[4] | E98h | —[4] |
| EF7h | —[4] | ED7h | —[4] | EB7h | —[4] | E97h | —[4] |
| EF6h | —[4] | ED6h | —[4] | EB6h | —[4] | E96h | —[4] |
| EF5h | —[4] | ED5h | —[4] | EB5h | —[4] | E95h | —[4] |
| EF4h | —[4] | ED4h | —[4] | EB4h | —[4] | E94h | —[4] |
| EF3h | —[4] | ED3h | —[4] | EB3h | —[4] | E93h | —[4] |
| EF2h | —[4] | ED2h | —[4] | EB2h | —[4] | E92h | —[4] |
| EF1h | —[4] | ED1h | —[4] | EB1h | —[4] | E91h | —[4] |
| EF0h | —[4] | ED0h | —[4] | EB0h | —[4] | E90h | —[4] |
| EEFh | —[4] | ECFh | —[4] | EAFh | —[4] | E8Fh | —[4] |
| EEEh | —[4] | ECEh | —[4] | EAEh | —[4] | E8Eh | —[4] |
| EEDh | —[4] | ECDh | —[4] | EADh | —[4] | E8Dh | —[4] |
| EECh | —[4] | ECCh | —[4] | EACh | —[4] | E8Ch | —[4] |
| EEBh | —[4] | ECBh | —[4] | EABh | —[4] | E8Bh | —[4] |
| EEAh | —[4] | ECAh | —[4] | EAAh | —[4] | E8Ah | —[4] |
| EE9h | —[4] | EC9h | —[4] | EA9h | —[4] | E89h | —[4] |
| EE8h | —[4] | EC8h | —[4] | EA8h | —[4] | E88h | —[4] |
| EE7h | —[4] | EC7h | —[4] | EA7h | —[4] | E87h | —[4] |
| EE6h | —[4] | EC6h | —[4] | EA6h | —[4] | E86h | —[4] |
| EE5h | —[4] | EC5h | —[4] | EA5h | —[4] | E85h | —[4] |
| EE4h | —[4] | EC4h | —[4] | EA4h | —[4] | E84h | —[4] |
| EE3h | —[4] | EC3h | —[4] | EA3h | —[4] | E83h | —[4] |
| EE2h | —[4] | EC2h | —[4] | EA2h | —[4] | E82h | —[4] |
| EE1h | —[4] | EC1h | —[4] | EA1h | —[4] | E81h | —[4] |
| EE0h | —[4] | EC0h | —[4] | EA0h | —[4] | E80h | —[4] |

**Note 1:** Shaded registers are available in Access Bank low area, while the rest are available in Bank 15.

**2:** CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

**3:** These registers are not CAN registers.

**4:** Unimplemented registers are read as '0'.


© 2009 Microchip Technology Inc.

**TABLE 24-1: CAN CONTROLLER REGISTER MAP (CONTINUED)**

| Address[1] | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| E7Fh | CANCON_RO4[2] | E5Fh | CANCON_RO6[2] | E3Fh | CANCON_RO8[2] | E1Fh | —[4] |
| E7Eh | CANSTAT_RO4[2] | E5Eh | CANSTAT_RO6[2] | E3Eh | CANSTAT_RO8[2] | E1Eh | —[4] |
| E7Dh | B5D7 | E5Dh | B3D7 | E3Dh | B1D7 | E1Dh | —[4] |
| E7Ch | B5D6 | E5Ch | B3D6 | E3Ch | B1D6 | E1Ch | —[4] |
| E7Bh | B5D5 | E5Bh | B3D5 | E3Bh | B1D5 | E1Bh | —[4] |
| E7Ah | B5D4 | E5Ah | B3D4 | E3Ah | B1D4 | E1Ah | —[4] |
| E79h | B5D3 | E59h | B3D3 | E39h | B1D3 | E19h | —[4] |
| E78h | B5D2 | E58h | B3D2 | E38h | B1D2 | E18h | —[4] |
| E77h | B5D1 | E57h | B3D1 | E37h | B1D1 | E17h | —[4] |
| E76h | B5D0 | E56h | B3D0 | E36h | B1D0 | E16h | —[4] |
| E75h | B5DLC | E55h | B3DLC | E35h | B1DLC | E15h | —[4] |
| E74h | B5EIDL | E54h | B3EIDL | E34h | B1EIDL | E14h | —[4] |
| E73h | B5EIDH | E53h | B3EIDH | E33h | B1EIDH | E13h | —[4] |
| E72h | B5SIDL | E52h | B3SIDL | E32h | B1SIDL | E12h | —[4] |
| E71h | B5SIDH | E51h | B3SIDH | E31h | B1SIDH | E11h | —[4] |
| E70h | B5CON | E50h | B3CON | E30h | B1CON | E10h | —[4] |
| E6Fh | CANCON_RO5 | E4Fh | CANCON_RO7 | E2Fh | CANCON_RO9 | E0Fh | —[4] |
| E6Eh | CANSTAT_RO5 | E4Eh | CANSTAT_RO7 | E2Eh | CANSTAT_RO9 | E0Eh | —[4] |
| E6Dh | B4D7 | E4Dh | B2D7 | E2Dh | B0D7 | E0Dh | —[4] |
| E6Ch | B4D6 | E4Ch | B2D6 | E2Ch | B0D6 | E0Ch | —[4] |
| E6Bh | B4D5 | E4Bh | B2D5 | E2Bh | B0D5 | E0Bh | —[4] |
| E6Ah | B4D4 | E4Ah | B2D4 | E2Ah | B0D4 | E0Ah | —[4] |
| E69h | B4D3 | E49h | B2D3 | E29h | B0D3 | E09h | —[4] |
| E68h | B4D2 | E48h | B2D2 | E28h | B0D2 | E08h | —[4] |
| E67h | B4D1 | E47h | B2D1 | E27h | B0D1 | E07h | —[4] |
| E66h | B4D0 | E46h | B2D0 | E26h | B0D0 | E06h | —[4] |
| E65h | B4DLC | E45h | B2DLC | E25h | B0DLC | E05h | —[4] |
| E64h | B4EIDL | E44h | B2EIDL | E24h | B0EIDL | E04h | —[4] |
| E63h | B4EIDH | E43h | B2EIDH | E23h | B0EIDH | E03h | —[4] |
| E62h | B4SIDL | E42h | B2SIDL | E22h | B0SIDL | E02h | —[4] |
| E61h | B4SIDH | E41h | B2SIDH | E21h | B0SIDH | E01h | —[4] |
| E60h | B4CON | E40h | B2CON | E20h | B0CON | E00h | —[4] |

**Note 1:** Shaded registers are available in Access Bank low area, while the rest are available in Bank 15.

**2:** CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

**3:** These registers are not CAN registers.

**4:** Unimplemented registers are read as '0'.

**TABLE 24-1: CAN CONTROLLER REGISTER MAP (CONTINUED)**

| Address[1] | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|
| DFFh | —[4] | DDFh | —[4] | DBFh | —[4] | D9Fh | —[4] |
| DFEh | —[4] | DDEh | —[4] | DBEh | —[4] | D9Eh | —[4] |
| DFDh | —[4] | DDDh | —[4] | DBDh | —[4] | D9Dh | —[4] |
| DFCh | TXBIE | DDCh | —[4] | DBCh | —[4] | D9Ch | —[4] |
| DFBh | —[4] | DDBh | —[4] | DBBh | —[4] | D9Bh | —[4] |
| DFAh | BIE0 | DDAh | —[4] | DBAh | —[4] | D9Ah | —[4] |
| DF9h | —[4] | DD9h | —[4] | DB9h | —[4] | D99h | —[4] |
| DF8h | BSEL0 | DD8h | SDFLC | DB8h | —[4] | D98h | —[4] |
| DF7h | —[4] | DD7h | —[4] | DB7h | —[4] | D97h | —[4] |
| DF6h | —[4] | DD6h | —[4] | DB6h | —[4] | D96h | —[4] |
| DF5h | —[4] | DD5h | RXFCON1 | DB5h | —[4] | D95h | —[4] |
| DF4h | —[4] | DD4h | RXFCON0 | DB4h | —[4] | D94h | —[4] |
| DF3h | MSEL3 | DD3h | —[4] | DB3h | —[4] | D93h | RXF15EIDL |
| DF2h | MSEL2 | DD2h | —[4] | DB2h | —[4] | D92h | RXF15EIDH |
| DF1h | MSEL1 | DD1h | —[4] | DB1h | —[4] | D91h | RXF15SIDL |
| DF0h | MSEL0 | DD0h | —[4] | DB0h | —[4] | D90h | RXF15SIDH |
| DEFh | —[4] | DCFh | —[4] | DAFh | —[4] | D8Fh | —[4] |
| DEEh | —[4] | DCEh | —[4] | DAEh | —[4] | D8Eh | —[4] |
| DEDh | —[4] | DCDh | —[4] | DADh | —[4] | D8Dh | —[4] |
| DECh | —[4] | DCCh | —[4] | DACh | —[4] | D8Ch | —[4] |
| DEBh | —[4] | DCBh | —[4] | DABh | —[4] | D8Bh | RXF14EIDL |
| DEAh | —[4] | DCAh | —[4] | DAAh | —[4] | D8Ah | RXF14EIDH |
| DE9h | —[4] | DC9h | —[4] | DA9h | —[4] | D89h | RXF14SIDL |
| DE8h | —[4] | DC8h | —[4] | DA8h | —[4] | D88h | RXF14SIDH |
| DE7h | RXFBCON7 | DC7h | —[4] | DA7h | —[4] | D87h | RXF13EIDL |
| DE6h | RXFBCON6 | DC6h | —[4] | DA6h | —[4] | D86h | RXF13EIDH |
| DE5h | RXFBCON5 | DC5h | —[4] | DA5h | —[4] | D85h | RXF13SIDL |
| DE4h | RXFBCON4 | DC4h | —[4] | DA4h | —[4] | D84h | RXF13SIDH |
| DE3h | RXFBCON3 | DC3h | —[4] | DA3h | —[4] | D83h | RXF12EIDL |
| DE2h | RXFBCON2 | DC2h | —[4] | DA2h | —[4] | D82h | RXF12EIDH |
| DE1h | RXFBCON1 | DC1h | —[4] | DA1h | —[4] | D81h | RXF12SIDL |
| DE0h | RXFBCON0 | DC0h | —[4] | DA0h | —[4] | D80h | RXF12SIDH |

**Note 1:** Shaded registers are available in Access Bank low area, while the rest are available in Bank 15.

**2:** CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

**3:** These registers are not CAN registers.

**4:** Unimplemented registers are read as '0'.


© 2009 Microchip Technology Inc.

**TABLE 24-1: CAN CONTROLLER REGISTER MAP (CONTINUED)**

| Address[1] | Name |
|---|---|
| D7Fh | —[4] |
| D7Eh | —[4] |
| D7Dh | —[4] |
| D7Ch | —[4] |
| D7Bh | RXF11EIDL |
| D7Ah | RXF11EIDH |
| D79h | RXF11SIDL |
| D78h | RXF11SIDH |
| D77h | RXF10EIDL |
| D76h | RXF10EIDH |
| D75h | RXF10SIDL |
| D74h | RXF10SIDH |
| D73h | RXF9EIDL |
| D72h | RXF9EIDH |
| D71h | RXF9SIDL |
| D70h | RXF9SIDH |
| D6Fh | —[4] |
| D6Eh | —[4] |
| D6Dh | —[4] |
| D6Ch | —[4] |
| D6Bh | RXF8EIDL |
| D6Ah | RXF8EIDH |
| D69h | RXF8SIDL |
| D68h | RXF8SIDH |
| D67h | RXF7EIDL |
| D66h | RXF7EIDH |
| D65h | RXF7SIDL |
| D64h | RXF7SIDH |
| D63h | RXF6EIDL |
| D62h | RXF6EIDH |
| D61h | RXF6SIDL |
| D60h | RXF6SIDH |

**Note 1:** Shaded registers are available in Access Bank low area while the rest are available in Bank 15.

**2:** CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

**3:** These registers are not CAN registers.

**4:** Unimplemented registers are read as '0'.

## 24.3 CAN Modes of Operation

The PIC18F2480/2580/4480/4580 has six main modes of operation:

- Configuration mode
- Disable/Sleep mode
- Normal Operation mode
- Listen Only mode
- Loopback mode
- Error Recognition mode

All modes, except Error Recognition, are requested by setting the REQOP bits (CANCON<7:5>). Error Recognition mode is requested through the RXM bits of the Receive Buffer register(s). Entry into a mode is Acknowledged by monitoring the OPMODE bits.

When changing modes, the mode will not actually change until all pending message transmissions are complete. Because of this, the user must verify that the device has actually changed into the requested mode before further operations are executed.

### 24.3.1 CONFIGURATION MODE

The CAN module has to be initialized before the activation. This is only possible if the module is in the Configuration mode. The Configuration mode is requested by setting the REQOP2 bit. Only when the status bit, OPMODE2, has a high level can the initialization be performed. Afterwards, the Configuration registers, the acceptance mask registers and the acceptance filter registers can be written. The module is activated by setting the REQOP control bits to zero.

The module will protect the user from accidentally violating the CAN protocol through programming errors. All registers which control the configuration of the module can not be modified while the module is online. The CAN module will not be allowed to enter the Configuration mode while a transmission or reception is taking place. The Configuration mode serves as a lock to protect the following registers:

- Configuration Registers
- Functional Mode Selection Registers
- Bit Timing Registers
- Identifier Acceptance Filter Registers
- Identifier Acceptance Mask Registers
- Filter and Mask Control Registers
- Mask Selection Registers

In the Configuration mode, the module will not transmit or receive. The error counters are cleared and the interrupt flags remain unchanged. The programmer will have access to Configuration registers that are access restricted in other modes. I/O pins will revert to normal I/O functions.

### 24.3.2 DISABLE/SLEEP MODE

In Disable/Sleep mode, the module will not transmit or receive. The module has the ability to set the WAKIF bit due to bus activity; however, any pending interrupts will remain and the error counters will retain their value.

If the REQOP<2:0> bits are set to '001', the module will enter the module Disable/Sleep mode. This mode is similar to disabling other peripheral modules by turning off the module enables. This causes the module internal clock to stop unless the module is active (i.e., receiving or transmitting a message). If the module is active, the module will wait for 11 recessive bits on the CAN bus, detect that condition as an Idle bus, then accept the module Disable/Sleep command. OPMODE<2:0> = 001 indicates whether the module successfully went into the module Disable/Sleep mode.

The WAKIF interrupt is the only module interrupt that is still active in the Disable/Sleep mode. If the WAKDIS is cleared and WAKIE is set, the processor will receive an interrupt whenever the module detects recessive to dominant transition. On wake-up, the module will automatically be set to the previous mode of operation. For example, if the module was switched from Normal to Disable/Sleep mode on bus activity wake-up, the module will automatically enter into Normal mode and the first message that caused the module to wake-up is lost. The module will not generate any error frame. Firmware logic must detect this condition and make sure that retransmission is requested. If the processor receives a wake-up interrupt while it is sleeping, more than one message may get lost. The actual number of messages lost would depend on the processor oscillator start-up time and incoming message bit rate.

The TXCAN pin will stay in the recessive state while the module is in Disable/Sleep mode.

### 24.3.3 NORMAL MODE

This is the standard operating mode of the PIC18F2480/2580/4480/4580 devices. In this mode, the device actively monitors all bus messages and generates Acknowledge bits, error frames, etc. This is also the only mode in which the PIC18F2480/2580/4480/4580 devices will transmit messages over the CAN bus.


© 2009 Microchip Technology Inc.

### 24.3.4 LISTEN ONLY MODE

Listen Only mode provides a means for the PIC18F2480/2580/4480/4580 devices to receive all messages, including messages with errors. This mode can be used for bus monitor applications or for detecting the baud rate in 'hot plugging' situations. For auto-baud detection, it is necessary that there are at least two other nodes which are communicating with each other. The baud rate can be detected empirically by testing different values until valid messages are received. The Listen Only mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or Acknowledge signals. The filters and masks can be used to allow only particular messages to be loaded into the receive registers or the filter masks can be set to all zeros to allow a message with any identifier to pass. The error counters are reset and deactivated in this state. The Listen Only mode is activated by setting the mode request bits in the CANCON register.

### 24.3.5 LOOPBACK MODE

This mode will allow internal transmission of messages from the transmit buffers to the receive buffers without actually transmitting messages on the CAN bus. This mode can be used in system development and testing. In this mode, the ACK bit is ignored and the device will allow incoming messages from itself, just as if they were coming from another node. The Loopback mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or Acknowledge signals. The TXCAN pin will revert to port I/O while the device is in this mode. The filters and masks can be used to allow only particular messages to be loaded into the receive registers. The masks can be set to all zeros to provide a mode that accepts all messages. The Loopback mode is activated by setting the mode request bits in the CANCON register.

### 24.3.6 ERROR RECOGNITION MODE

The module can be set to ignore all errors and receive any message. In functional Mode 0, the Error Recognition mode is activated by setting the RXM<1:0> bits in the RXBnCON registers to '11'. In this mode, the data which is in the message assembly buffer until the error time, is copied in the receive buffer and can be read via the CPU interface.

## 24.4 CAN Module Functional Modes

In addition to CAN modes of operation, the ECAN module offers a total of 3 functional modes. Each of these modes are identified as Mode 0, Mode 1 and Mode 2.

### 24.4.1 MODE 0 – LEGACY MODE

Mode 0 is designed to be fully compatible with CAN modules used in PIC18CXX8 and PIC18FXX8 devices. This is the default mode of operation on all Reset conditions. As a result, module code written for the PIC18XX8 CAN module may be used on the ECAN module without any code changes.

The following is the list of resources available in Mode 0:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Two acceptance masks, one for each receive buffer: RXM0, RXM1
- Six acceptance filters, 2 for RXB0 and 4 for RXB1: RXF0, RXF1, RXF2, RXF3, RXF4, RXF5

### 24.4.2 MODE 1 – ENHANCED LEGACY MODE

Mode 1 is similar to Mode 0, with the exception that more resources are available in Mode 1. There are 16 acceptance filters and two acceptance mask registers. Acceptance Filter 15 can be used as either an acceptance filter or an acceptance mask register. In addition to three transmit and two receive buffers, there are six more message buffers. One or more of these additional buffers can be programmed as transmit or receive buffers. These additional buffers can also be programmed to automatically handle RTR messages.

Fourteen of sixteen acceptance filter registers can be dynamically associated to any receive buffer and acceptance mask register. One can use this capability to associate more than one filter to any one buffer.

When a receive buffer is programmed to use standard identifier messages, part of the full acceptance filter register can be used as a data byte filter. The length of the data byte filter is programmable from 0 to 18 bits. This functionality simplifies implementation of high-level protocols, such as the DeviceNet™ protocol.

The following is the list of resources available in Mode 1:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen dynamically assigned acceptance filters: RXF0-RXF15
- Two dedicated acceptance mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC

### 24.4.3 MODE 2 – ENHANCED FIFO MODE

In Mode 2, two or more receive buffers are used to form the receive FIFO (first in, first out) buffer. There is no one-to-one relationship between the receive buffer and acceptance filter registers. Any filter that is enabled and linked to any FIFO receive buffer can generate acceptance and cause FIFO to be updated.

FIFO length is user-programmable, from 2-8 buffers deep. FIFO length is determined by the very first programmable buffer that is configured as a transmit buffer. For example, if Buffer 2 (B2) is programmed as a transmit buffer, FIFO consists of RXB0, RXB1, B0 and B1 – creating a FIFO length of 4. If all programmable buffers are configured as receive buffers, FIFO will have the maximum length of 8.

The following is the list of resources available in Mode 2:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX; receive buffers form FIFO: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen acceptance filters: RXF0-RXF15
- Two dedicated acceptance mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC, useful for DeviceNet protocol

## 24.5 CAN Message Buffers

### 24.5.1 DEDICATED TRANSMIT BUFFERS

The PIC18F2480/2580/4480/4580 devices implement three dedicated transmit buffers – TXB0, TXB1 and TXB2. Each of these buffers occupies 14 bytes of SRAM and are mapped into the SFR memory map. These are the only transmit buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each transmit buffer contains one Control register (TXBnCON), four Identifier registers (TXBnSIDL, TXBnSIDH, TXBnEIDL, TXBnEIDH), one Data Length Count register (TXBnDLC) and eight Data Byte registers (TXBnDm).

### 24.5.2 DEDICATED RECEIVE BUFFERS

The PIC18F2480/2580/4480/4580 devices implement two dedicated receive buffers: RXB0 and RXB1. Each of these buffers occupies 14 bytes of SRAM and are mapped into SFR memory map. These are the only receive buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each receive buffer contains one Control register (RXBnCON), four Identifier registers (RXBnSIDL, RXBnSIDH, RXBnEIDL, RXBnEIDH), one Data Length Count register (RXBnDLC) and eight Data Byte registers (RXBnDm).

There is also a separate Message Assembly Buffer (MAB) which acts as an additional receive buffer. MAB is always committed to receiving the next message from the bus and is not directly accessible to user firmware. The MAB assembles all incoming messages one by one. A message is transferred to appropriate receive buffers only if the corresponding acceptance filter criteria is met.

### 24.5.3 PROGRAMMABLE TRANSMIT/ RECEIVE BUFFERS

The ECAN module implements six new buffers: B0-B5. These buffers are individually programmable as either transmit or receive buffers. These buffers are available only in Mode 1 and 2. As with dedicated transmit and receive buffers, each of these programmable buffers occupies 14 bytes of SRAM and are mapped into SFR memory map.

Each buffer contains one Control register (BnCON), four Identifier registers (BnSIDL, BnSIDH, BnEIDL, BnEIDH), one Data Length Count register (BnDLC) and eight Data Byte registers (BnDm). Each of these registers contains two sets of control bits. Depending on whether the buffer is configured as transmit or receive, one would use the corresponding control bit set. By default, all buffers are configured as receive buffers. Each buffer can be individually configured as a transmit or receive buffer by setting the corresponding TXENn bit in the BSEL0 register.

When configured as transmit buffers, user firmware may access transmit buffers in any order similar to accessing dedicated transmit buffers. In receive configuration with Mode 1 enabled, user firmware may also access receive buffers in any order required. But in Mode 2, all receive buffers are combined to form a single FIFO. Actual FIFO length is programmable by user firmware. Access to FIFO must be done through the FIFO Pointer bits (FP<4:0>) in the CANCON register. It must be noted that there is no hardware protection against out of order FIFO reads.


© 2009 Microchip Technology Inc.

### 24.5.4 PROGRAMMABLE AUTO-RTR BUFFERS

In Mode 1 and 2, any of six programmable transmit/receive buffers may be programmed to automatically respond to predefined RTR messages without user firmware intervention. Automatic RTR handling is enabled by setting the TXnEN bit in the BSEL0 register and the RTREN bit in the BnCON register. After this setup, when an RTR request is received, the TXREQ bit is automatically set and the current buffer content is automatically queued for transmission as a RTR response. As with all transmit buffers, once the TXREQ bit is set, buffer registers become read-only and any writes to them will be ignored.

The following outlines the steps required to automatically handle RTR messages:

1. Set buffer to Transmit mode by setting the TXnEN bit to '1' in BSEL0 register.
2. At least one acceptance filter must be associated with this buffer and preloaded with the expected RTR identifier.
3. Bit, RTREN in the BnCON register, must be set to '1'.
4. Buffer must be preloaded with the data to be sent as a RTR response.

Normally, user firmware will keep buffer data registers up to date. If firmware attempts to update the buffer while an automatic RTR response is in the process of transmission, all writes to buffers are ignored.

## 24.6 CAN Message Transmission

### 24.6.1 INITIATING TRANSMISSION

For the MCU to have write access to the message buffer, the TXREQ bit must be clear, indicating that the message buffer is clear of any pending message to be transmitted. At a minimum, the SIDH, SIDL and DLC registers must be loaded. If data bytes are present in the message, the Data registers must also be loaded. If the message is to use extended identifiers, the EIDH:EIDL registers must also be loaded and the EXIDE bit set.

To initiate message transmission, the TXREQ bit must be set for each buffer to be transmitted. When TXREQ is set, the TXABT, TXLARB and TXERR bits will be cleared. To successfully complete the transmission, there must be at least one node with matching baud rate on the network.

Setting the TXREQ bit does not initiate a message transmission; it merely flags a message buffer as ready for transmission. Transmission will start when the device detects that the bus is available. The device will then begin transmission of the highest priority message that is ready.

When the transmission has completed successfully, the TXREQ bit will be cleared, the TXBnIF bit will be set and an interrupt will be generated if the TXBnIE bit is set.

If the message transmission fails, the TXREQ will remain set, indicating that the message is still pending for transmission and one of the following condition flags will be set. If the message started to transmit but encountered an error condition, the TXERR and the IRXIF bits will be set and an interrupt will be generated. If the message lost arbitration, the TXLARB bit will be set.

### 24.6.2 ABORTING TRANSMISSION

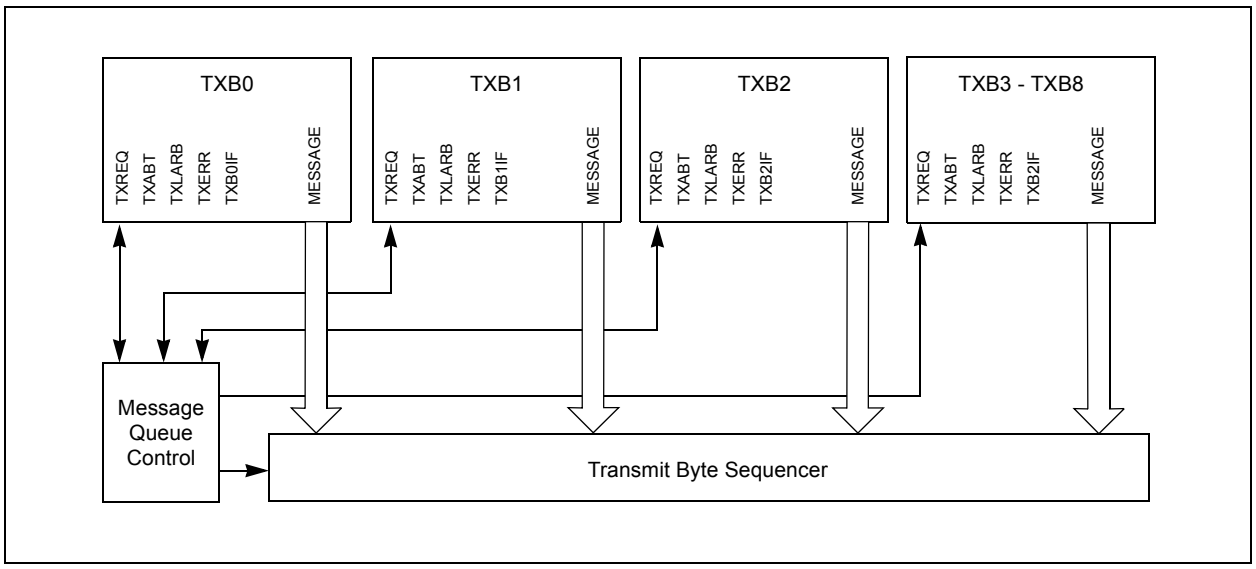
The MCU can request to abort a message by clearing the TXREQ bit associated with the corresponding message buffer (TXBnCON<3> or BnCON<3>). Setting the ABAT bit (CANCON<4>) will request an abort of all pending messages. If the message has not yet started transmission, or if the message started but is interrupted by loss of arbitration or an error, the abort will be processed. The abort is indicated when the module sets the TXABT bit for the corresponding buffer (TXBnCON<6> or BnCON<6>). If the message has started to transmit, it will attempt to transmit the current message fully. If the current message is transmitted fully and is not lost to arbitration or an error, the TXABT bit will not be set because the message was transmitted successfully. Likewise, if a message is being transmitted during an abort request and the message is lost to arbitration or an error, the message will not be retransmitted and the TXABT bit will be set, indicating that the message was successfully aborted.

Once an abort is requested by setting the ABAT or TXABT bits, it cannot be cleared to cancel the abort request. Only CAN module hardware or a POR condition can clear it.

### 24.6.3 TRANSMIT PRIORITY

Transmit priority is a prioritization within the PIC18F2480/2580/4480/4580 devices of the pending transmittable messages. This is independent from and not related to any prioritization implicit in the message arbitration scheme built into the CAN protocol. Prior to sending the Start-Of-Frame (SOF), the priority of all buffers that are queued for transmission is compared. The transmit buffer with the highest priority will be sent first. If two buffers have the same priority setting, the buffer with the highest buffer number will be sent first. There are four levels of transmit priority. If the TXP bits for a particular message buffer are set to '11', that buffer has the highest possible priority. If the TXP bits for a particular message buffer are set to '00', that buffer has the lowest possible priority.

**FIGURE 24-2: TRANSMIT BUFFERS**





© 2009 Microchip Technology Inc.

## 24.7 Message Reception

### 24.7.1 RECEIVING A MESSAGE

Of all receive buffers, the MAB is always committed to receiving the next message from the bus. The MCU can access one buffer while the other buffer is available for message reception or holding a previously received message.

| | |
|---|---|
| **Note:** | The entire contents of the MAB are moved into the receive buffer once a message is accepted. This means that regardless of the type of identifier (standard or extended) and the number of data bytes received, the entire receive buffer is overwritten with the MAB contents. Therefore, the contents of all registers in the buffer must be assumed to have been modified when any message is received. |

When a message is moved into either of the receive buffers, the associated RXFUL bit is set. This bit must be cleared by the MCU when it has completed processing the message in the buffer in order to allow a new message to be received into the buffer. This bit provides a positive lockout to ensure that the firmware has finished with the message before the module attempts to load a new message into the receive buffer. If the receive interrupt is enabled, an interrupt will be generated to indicate that a valid message has been received.

Once a message is loaded into any matching buffer, user firmware may determine exactly what filter caused this reception by checking the filter hit bits in the RXBnCON or BnCON registers. In Mode 0, FILHIT<3:0> of RXBnCON serve as filter hit bits. In Mode 1 and 2, FILHIT<4:0> bits of BnCON serve as filter hit bits. The same registers also indicate whether the current message is an RTR frame or not. A received message is considered a standard identifier message if the EXID bit in the RXBnSIDL or the BnSIDL register is cleared. Conversely, a set EXID bit indicates an extended identifier message. If the received message is a standard identifier message, user firmware needs to read the SIDL and SIDH registers. In the case of an extended identifier message, firmware should read the SIDL, SIDH, EIDL and EIDH registers. If the RXBnDLC or BnDLC register contain non-zero data count, user firmware should also read the corresponding number of data bytes by accessing the RXBnDm or the BnDm registers. When a received message is an RTR and if the current buffer is not configured for automatic RTR handling, user firmware must take appropriate action and respond manually.

Each receive buffer contains RXM bits to set special Receive modes. In Mode 0, RXM<1:0> bits in RXBnCON define a total of four Receive modes. In Mode 1 and 2, RXM1 bit, in combination with the EXID mask and filter bit, define the same four receive modes. Normally, these bits are set to '00' to enable reception of all valid messages as determined by the appropriate acceptance filters. In this case, the determination of whether or not to receive standard or extended messages is determined by the EXIDE bit in the acceptance filter register. In Mode 0, if the RXM bits are set to '01' or '10', the receiver will accept only messages with standard or extended identifiers, respectively. If an acceptance filter has the EXIDE bit set such that it does not correspond with the RXM mode, that acceptance filter is rendered useless. In Mode 1 and 2, setting EXID in the SIDL Mask register will ensure that only standard or extended identifiers are received. These two modes of RXM bits can be used in systems where it is known that only standard or extended messages will be on the bus. If the RXM bits are set to '11' (RXM1 = 1 in Mode 1 and 2), the buffer will receive all messages regardless of the values of the acceptance filters. Also, if a message has an error before the end of frame, that portion of the message assembled in the MAB before the error frame will be loaded into the buffer. This mode may serve as a valuable debugging tool for a given CAN network. It should not be used in an actual system environment as the actual system will always have some bus errors and all nodes on the bus are expected to ignore them.

In Mode 1 and 2, when a programmable buffer is configured as a transmit buffer and one or more acceptance filters are associated with it, all incoming messages matching this acceptance filter criteria will be discarded. To avoid this scenario, user firmware must make sure that there are no acceptance filters associated with a buffer configured as a transmit buffer.

### 24.7.2 RECEIVE PRIORITY

When in Mode 0, RXB0 is the higher priority buffer and has two message acceptance filters associated with it. RXB1 is the lower priority buffer and has four acceptance filters associated with it. The lower number of acceptance filters makes the match on RXB0 more restrictive and implies a higher priority for that buffer. Additionally, the RXB0CON register can be configured such that if RXB0 contains a valid message and another valid message is received, an overflow error will not occur and the new message will be moved into RXB1 regardless of the acceptance criteria of RXB1. There are also two programmable acceptance filter masks available, one for each receive buffer (see **Section 24.5 "CAN Message Buffers"**).

In Mode 1 and 2, there are a total of 16 acceptance filters available and each can be dynamically assigned to any of the receive buffers. A buffer with a lower number has higher priority. Given this, if an incoming message matches with two or more receive buffer acceptance criteria, the buffer with the lower number will be loaded with that message.

### 24.7.3 ENHANCED FIFO MODE

When configured for Mode 2, two of the dedicated receive buffers in combination with one or more programmable transmit/receive buffers, are used to create a maximum of an 8 buffer deep FIFO buffer. In this mode, there is no direct correlation between filters and receive buffer registers. Any filter that has been enabled can generate an acceptance. When a message has been accepted, it is stored in the next available receive buffer register and an internal Write Pointer is incremented. The FIFO can be a maximum of 8 buffers deep. The entire FIFO must consist of contiguous receive buffers. The FIFO head begins at RXB0 buffer and its tail spans toward B5. The maximum length of the FIFO is limited by the presence or absence of the first transmit buffer starting from B0. If a buffer is configured as a transmit buffer, the FIFO length is reduced accordingly. For instance, if B3 is configured as a transmit buffer, the actual FIFO will consist of RXB0, RXB1, B0, B1 and B2, a total of 5 buffers. If B0 is configured as a transmit buffer, the FIFO length will be 2. If none of the programmable buffers are configured as a transmit buffer, the FIFO will be 8 buffers deep. A system that requires more transmit buffers should try to locate transmit buffers at the very end of B0-B5 buffers to maximize available FIFO length.

When a message is received in FIFO mode, the interrupt flag code bits (EICODE<4:0>) in the CANSTAT register will have a value of '10000', indicating the FIFO has received a message. FIFO Pointer bits, FP<3:0> in the CANCON register, point to the buffer that contains data not yet read. The FIFO Pointer bits, in this sense, serve as the FIFO Read Pointer. The user should use FP bits and read corresponding buffer data. When receive data is no longer needed, the RXFUL bit in the current buffer must be cleared, causing FP<3:0> to be updated by the module.

To determine whether FIFO is empty or not, the user may use the FP<3:0> bits to access the RXFUL bit in the current buffer. If RXFUL is cleared, the FIFO is considered to be empty. If it is set, the FIFO may contain one or more messages. In Mode 2, the module also provides a bit called FIFO High Water Mark (FIFOWM) in the ECANCON register. This bit can be used to cause an interrupt whenever the FIFO contains only one or four empty buffers. The FIFO high water mark interrupt can serve as an early warning to a full FIFO condition.

### 24.7.4 TIME-STAMPING

The CAN module can be programmed to generate a time-stamp for every message that is received. When enabled, the module generates a capture signal for CCP1, which in turn captures the value of either Timer1 or Timer3. This value can be used as the message time-stamp.

To use the time-stamp capability, the CANCAP bit (CIOCON<4>) must be set. This replaces the capture input for CCP1 with the signal generated from the CAN module. In addition, CCP1CON<3:0> must be set to '0011' to enable the CCP Special Event Trigger for CAN events.

## 24.8 Message Acceptance Filters and Masks

The message acceptance filters and masks are used to determine if a message in the Message Assembly Buffer should be loaded into any of the receive buffers. Once a valid message has been received into the MAB, the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer. The filter masks are used to determine which bits in the identifier are examined with the filters. A truth table is shown below in Table 24-2 that indicates how each bit in the identifier is compared to the masks and filters to determine if a message should be loaded into a receive buffer. The mask essentially determines which bits to apply the acceptance filters to. If any mask bit is set to a zero, then that bit will automatically be accepted regardless of the filter bit.

**TABLE 24-2: FILTER/MASK TRUTH TABLE**

| Mask bit n | Filter bit n | Message Identifier bit n001 | Accept or Reject bit n |
|---|---|---|---|
| 0 | x | x | Accept |
| 1 | 0 | 0 | Accept |
| 1 | 0 | 1 | Reject |
| 1 | 1 | 0 | Reject |
| 1 | 1 | 1 | Accept |

**Legend:** x = don't care

In Mode 0, acceptance filters, RXF0 and RXF1, and filter mask, RXM0, are associated with RXB0. Filters, RXF2, RXF3, RXF4 and RXF5, and mask, RXM1, are associated with RXB1.


© 2009 Microchip Technology Inc.

In Mode 1 and 2, there are an additional 10 acceptance filters, RXF6-RXF15, creating a total of 16 available filters. RXF15 can be used either as an acceptance filter or acceptance mask register. Each of these acceptance filters can be individually enabled or disabled by setting or clearing the RXFENn bit in the RXFCONn register. Any of these 16 acceptance filters can be dynamically associated with any of the receive buffers. Actual association is made by setting the appropriate bits in the RXFBCONn register. Each RXFBCONn register contains a nibble for each filter. This nibble can be used to associate a specific filter to any of available receive buffers. User firmware may associate more than one filter to any one specific receive buffer.

In addition to dynamic filter to buffer association, in Mode 1 and 2, each filter can also be dynamically associated to available Acceptance Mask registers. The FILn_m bits in the MSELn register can be used to link a specific acceptance filter to an acceptance mask register. As with filter to buffer association, one can also associate more than one mask to a specific acceptance filter.

When a filter matches and a message is loaded into the receive buffer, the filter number that enabled the message reception is loaded into the FILHIT bit(s). In Mode 0 for RXB1, the RXB1CON register contains the FILHIT<2:0> bits. They are coded as follows:

- 101 = Acceptance Filter 5 (RXF5)
- 100 = Acceptance Filter 4 (RXF4)
- 011 = Acceptance Filter 3 (RXF3)
- 010 = Acceptance Filter 2 (RXF2)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0 (RXF0)

| **Note:** | '000' and '001' can only occur if the RXB0DBEN bit is set in the RXB0CON register, allowing RXB0 messages to rollover into RXB1. |
|---|---|

The coding of the RXB0DBEN bit enables these three bits to be used similarly to the FILHIT bits and to distinguish a hit on filter, RXF0 and RXF1, in either RXB0 or after a rollover into RXB1.

- 111 = Acceptance Filter 1 (RXF1)
- 110 = Acceptance Filter 0 (RXF0)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0 (RXF0)

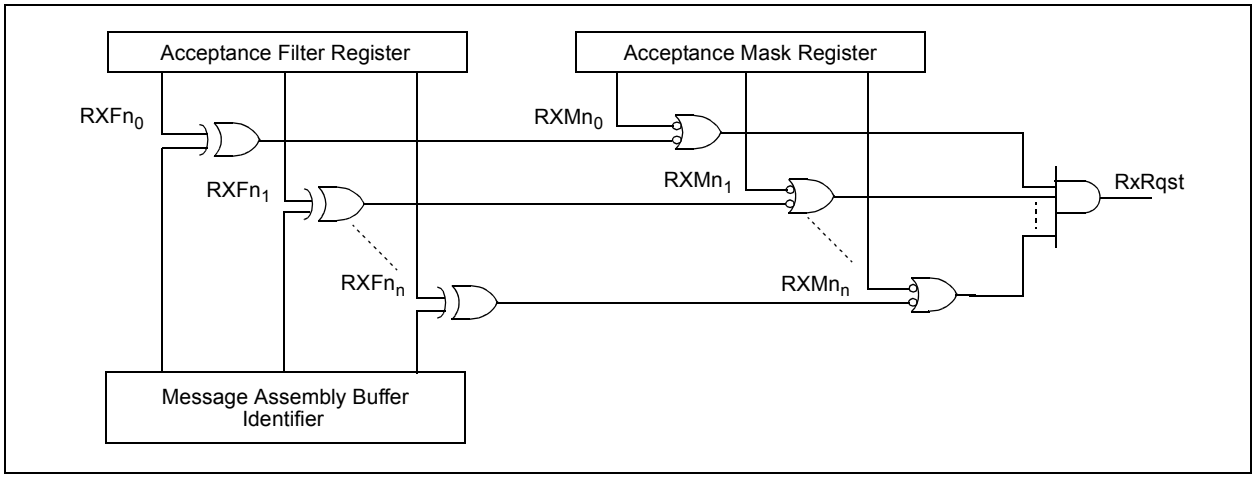
If the RXB0DBEN bit is clear, there are six codes corresponding to the six filters. If the RXB0DBEN bit is set, there are six codes corresponding to the six filters, plus two additional codes corresponding to RXF0 and RXF1 filters, that rollover into RXB1.

In Mode 1 and 2, each buffer control register contains 5 bits of filter hit bits (FILHIT<4:0>). A binary value of '0' indicates a hit from RXF0 and 15 indicates RXF15.

If more than one acceptance filter matches, the FILHIT bits will encode the binary value of the lowest numbered filter that matched. In other words, if filter RXF2 and filter RXF4 match, FILHIT will be loaded with the value for RXF2. This essentially prioritizes the acceptance filters with a lower number filter having higher priority. Messages are compared to filters in ascending order of filter number.

The mask and filter registers can only be modified when the PIC18F2480/2580/4480/4580 devices are in Configuration mode.

**FIGURE 24-3: MESSAGE ACCEPTANCE MASK AND FILTER OPERATION**



## 24.9 Baud Rate Setting

All nodes on a given CAN bus must have the same nominal bit rate. The CAN protocol uses Non-Return-to-Zero (NRZ) coding which does not encode a clock within the data stream. Therefore, the receive clock must be recovered by the receiving nodes and synchronized to the transmitter's clock.

As oscillators and transmission time may vary from node to node, the receiver must have some type of Phase Lock Loop (PLL) synchronized to data transmission edges to synchronize and maintain the receiver clock. Since the data is NRZ coded, it is necessary to include bit stuffing to ensure that an edge occurs at least every six bit times to maintain the Digital Phase Lock Loop (DPLL) synchronization.

The bit timing of the PIC18F2480/2580/4480/4580 is implemented using a DPLL that is configured to synchronize to the incoming data and provides the nominal timing for the transmitted data. The DPLL breaks each bit time into multiple segments made up of minimal periods of time called the *Time Quanta* (TQ).

Bus timing functions executed within the bit time frame, such as synchronization to the local oscillator, network transmission delay compensation and sample point positioning, are defined by the programmable bit timing logic of the DPLL.

All devices on the CAN bus must use the same bit rate. However, all devices are not required to have the same master oscillator clock frequency. For the different clock frequencies of the individual devices, the bit rate has to be adjusted by appropriately setting the baud rate prescaler and number of time quanta in each segment.

The *Nominal Bit Rate* is the number of bits transmitted per second, assuming an ideal transmitter with an ideal oscillator, in the absence of resynchronization. The nominal bit rate is defined to be a maximum of 1 Mb/s.

The *Nominal Bit Time* is defined as:

**EQUATION 24-1:**

$$T_{BIT} = 1/\text{Nominal Bit Rate}$$

The Nominal Bit Time can be thought of as being divided into separate, non-overlapping time segments. These segments (Figure 24-4) include:

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

The time segments (and thus, the Nominal Bit Time) are, in turn, made up of integer units of time called Time Quanta or TQ (see Figure 24-4). By definition, the Nominal Bit Time is programmable from a minimum of 8 TQ to a maximum of 25 TQ. Also by definition, the minimum Nominal Bit Time is 1 μs, corresponding to a maximum 1 Mb/s rate. The actual duration is given by the following relationship.

**EQUATION 24-2:**

$$\text{Nominal Bit Time} = T_Q * (\text{Sync_Seg} + \text{Prop_Seg} + \text{Phase_Seg1} + \text{Phase_Seg2})$$

The Time Quantum is a fixed unit derived from the oscillator period. It is also defined by the programmable baud rate prescaler, with integer values from 1 to 64, in addition to a fixed divide-by-two for clock generation. Mathematically, this is:

**EQUATION 24-3:**

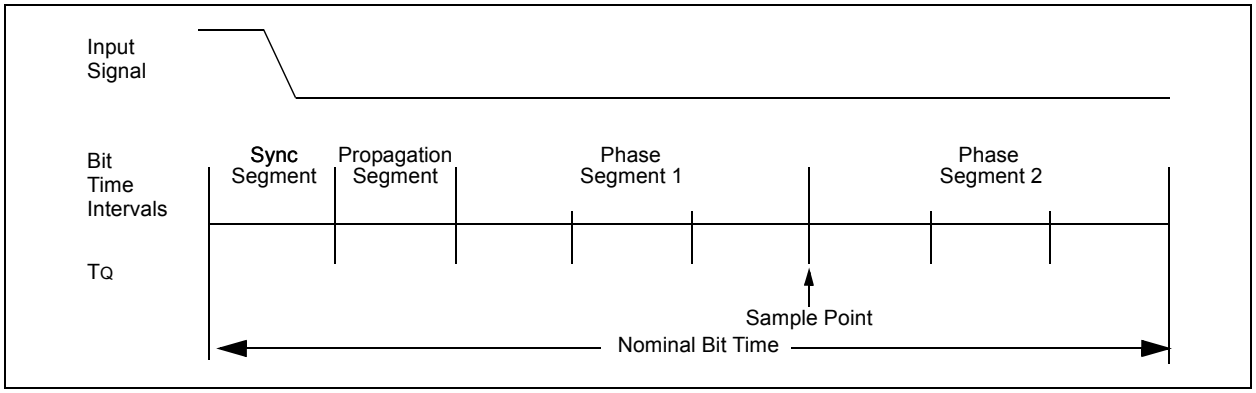
$$T_Q\ (\mu s) = (2 * (BRP+1))/F_{OSC}\ (MHz)$$

or

$$T_Q\ (\mu s) = (2 * (BRP+1)) * T_{OSC}\ (\mu s)$$

where FOSC is the clock frequency, TOSC is the corresponding oscillator period and BRP is an integer (0 through 63) represented by the binary values of BRGCON1<5:0>. The equation above refers to the effective clock frequency used by the microcontroller. If, for example, a 10 MHz crystal in HS mode is used, then FOSC = 10 MHz and TOSC = 100 ns. If the same 10 MHz crystal is used in HS-PLL mode, then the effective frequency is FOSC = 40 MHz and TOSC = 25 ns.

**FIGURE 24-4: BIT TIME PARTITIONING**





© 2009 Microchip Technology Inc.

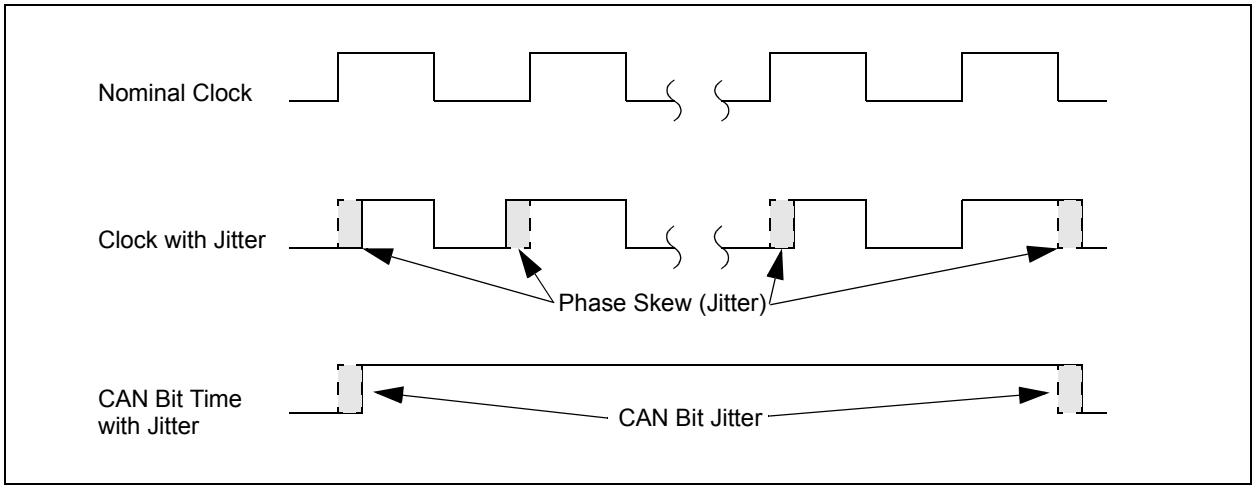
### 24.9.1 EXTERNAL CLOCK, INTERNAL CLOCK AND MEASURABLE JITTER IN HS-PLL BASED OSCILLATORS

The microcontroller clock frequency generated from a PLL circuit is subject to a jitter, also defined as Phase Jitter or Phase Skew. For its PIC18 Enhanced microcontrollers, Microchip specifies phase jitter ($P_{jitter}$) as being 2% (Gaussian distribution, within 3 standard deviations, see parameter F13 in Table 28-7) and Total Jitter ($T_{jitter}$) as being 2*$P_{jitter}$.

The CAN protocol uses a bit-stuffing technique that inserts a bit of a given polarity following five bits with the opposite polarity. This gives a total of 10 bits transmitted without re-synchronization (compensation for jitter or phase error).

Given the random nature of the jitter error added, it can be shown that the total error caused by the jitter tends to cancel itself over time. For a period of 10 bits, it is necessary to add only two jitter intervals to correct for jitter-induced error: one interval in the beginning of the 10-bit period and another at the end. The overall effect is shown in Figure 24-5.

**FIGURE 24-5: EFFECTS OF PHASE JITTER ON THE MICROCONTROLLER CLOCK AND CAN BIT TIME**




Once these considerations are taken into account, it is possible to show that the relation between the jitter and the total frequency error can be defined as:

$$\Delta f = \frac{T_{jitter}}{10 \times NBT} = \frac{2 \times P_{jitter}}{10 \times NBT}$$

where jitter is expressed in terms of time and NBT is the Nominal Bit Time.

For example, assume a CAN bit rate of 125 Kb/s, which gives an NBT of 8 µs. For a 16 MHz clock generated from a 4x PLL, the jitter at this clock frequency is:

$$2\% \times \frac{1}{16\ MHz} = \frac{0.02}{16 \times 10^6} = 1.25ns$$

and resultant frequency error is:

$$\frac{2 \times (1.25 \times 10^{-9})}{10 \times (8 \times 10^{-6})} = 3.125 \times 10^{-5} = 0.0031\%$$

Table 24-3 shows the relation between the clock generated by the PLL and the frequency error from jitter (measured jitter-induced error of 2%, Gaussian distribution, within 3 standard deviations), as a percentage of the nominal clock frequency.

This is clearly smaller than the expected drift of a crystal oscillator, typically specified at 100 ppm or 0.01%. If we add jitter to oscillator drift, we have a total frequency drift of 0.0132%. The total oscillator frequency errors for common clock frequencies and bit rates, including both drift and jitter, are shown in Table 24-4.

**TABLE 24-3: FREQUENCY ERROR FROM JITTER AT VARIOUS PLL-GENERATED CLOCK SPEEDS**

| PLL Output | $P_{jitter}$ | $T_{jitter}$ | Frequency Error at Various Nominal Bit Times (Bit Rates) | | | |
|---|---|---|---|---|---|---|
| | | | 8 μs (125 Kb/s) | 4 μs (250 Kb/s) | 2 μs (500 Kb/s) | 1 μs (1 Mb/s) |
| 40 MHz | 0.5 ns | 1 ns | 0.00125% | 0.00250% | 0.005% | 0.01% |
| 24 MHz | 0.83 ns | 1.67 ns | 0.00209% | 0.00418% | 0.008% | 0.017% |
| 16 MHz | 1.25 ns | 2.5 ns | 0.00313% | 0.00625% | 0.013% | 0.025% |

**TABLE 24-4: TOTAL FREQUENCY ERROR AT VARIOUS PLL-GENERATED CLOCK SPEEDS (100 PPM OSCILLATOR DRIFT, INCLUDING ERROR FROM JITTER)**

| Nominal PLL Output | Frequency Error at Various Nominal Bit Times (Bit Rates) | | | |
|---|---|---|---|---|
| | 8 μs (125 Kb/s) | 4 μs (250 Kb/s) | 2 μs (500 Kb/s) | 1 μs (1 Mb/s) |
| 40 MHz | 0.01125% | 0.01250% | 0.015% | 0.02% |
| 24 MHz | 0.01209% | 0.01418% | 0.018% | 0.027% |
| 16 MHz | 0.01313% | 0.01625% | 0.023% | 0.035% |


© 2009 Microchip Technology Inc.

### 24.9.2 TIME QUANTA

As already mentioned, the Time Quanta is a fixed unit derived from the oscillator period and baud rate prescaler. Its relationship to TBIT and the Nominal Bit Rate is shown in Example 24-6.

**EXAMPLE 24-6: CALCULATING TQ, NOMINAL BIT RATE AND NOMINAL BIT TIME**

TQ (µs) = (2 * (BRP+1))/FOSC (MHz)

TBIT (µs) = TQ (µs) * number of TQ per bit interval

Nominal Bit Rate (bits/s) = 1/TBIT

This frequency (FOSC) refers to the effective frequency used. If, for example, a 10 MHz external signal is used along with a PLL, then the effective frequency will be 4 x 10 MHz which equals 40 MHz.

**CASE 1:**

*For FOSC = 16 MHz, BRP<5:0> = 00h and Nominal Bit Time = 8 TQ:*

TQ = (2 * 1)/16 = 0.125 µs (125 ns)

TBIT = 8 * 0.125 = 1 µs ($10^{-6}$s)

Nominal Bit Rate = $1/10^{-6} = 10^6$ bits/s (1 Mb/s)

**CASE 2:**

*For FOSC = 20 MHz, BRP<5:0> = 01h and Nominal Bit Time = 8 TQ:*

TQ = (2 * 2)/20 = 0.2 µs (200 ns)

TBIT = 8 * 0.2 = 1.6 µs ($1.6 * 10^{-6}$s)

Nominal Bit Rate = $1/1.6 * 10^{-6}$s = 625,000 bits/s (625 Kb/s)

**CASE 3:**

*For FOSC = 25 MHz, BRP<5:0> = 3Fh and Nominal Bit Time = 25 TQ:*

TQ = (2 * 64)/25 = 5.12 µs

TBIT = 25 * 5.12 = 128 µs ($1.28 * 10^{-4}$s)

Nominal Bit Rate = $1/1.28 * 10^{-4}$ = 7813 bits/s (7.8 Kb/s)

The frequencies of the oscillators in the different nodes must be coordinated in order to provide a system wide specified nominal bit time. This means that all oscillators must have a TOSC that is an integral divisor of TQ. It should also be noted that although the number of TQ is programmable from 4 to 25, the usable minimum is 8 TQ. There is no assurance that a bit time of less than 8 TQ in length will operate correctly.

### 24.9.3 SYNCHRONIZATION SEGMENT

This part of the bit time is used to synchronize the various CAN nodes on the bus. The edge of the input signal is expected to occur during the sync segment. The duration is 1 TQ.

### 24.9.4 PROPAGATION SEGMENT

This part of the bit time is used to compensate for physical delay times within the network. These delay times consist of the signal propagation time on the bus line and the internal delay time of the nodes. The length of the propagation segment can be programmed from 1 TQ to 8 TQ by setting the PRSEG<2:0> bits.

### 24.9.5 PHASE BUFFER SEGMENTS

The phase buffer segments are used to optimally locate the sampling point of the received bit within the nominal bit time. The sampling point occurs between Phase Segment 1 and Phase Segment 2. These segments can be lengthened or shortened by the resynchronization process. The end of Phase Segment 1 determines the sampling point within a bit time. Phase Segment 1 is programmable from 1 TQ to 8 TQ in duration. Phase Segment 2 provides a delay before the next transmitted data transition and is also programmable from 1 TQ to 8 TQ in duration. However, due to IPT requirements, the actual minimum length of Phase Segment 2 is 2 TQ, or it may be defined to be equal to the greater of Phase Segment 1 or the Information Processing Time (IPT). The sampling point should be as late as possible or approximately 80% of the bit time.

### 24.9.6 SAMPLE POINT

The sample point is the point of time at which the bus level is read and the value of the received bit is determined. The sampling point occurs at the end of Phase Segment 1. If the bit timing is slow and contains many TQ, it is possible to specify multiple sampling of the bus line at the sample point. The value of the received bit is determined to be the value of the majority decision of three values. The three samples are taken at the sample point and twice before, with a time of TQ/2 between each sample.

### 24.9.7 INFORMATION PROCESSING TIME

The Information Processing Time (IPT) is the time segment starting at the sample point that is reserved for calculation of the subsequent bit level. The CAN specification defines this time to be less than or equal to 2 TQ. The PIC18F2480/2580/4480/4580 devices define this time to be 2 TQ. Thus, Phase Segment 2 must be at least 2 TQ long.

## 24.10 Synchronization

To compensate for phase shifts between the oscillator frequencies of each of the nodes on the bus, each CAN controller must be able to synchronize to the relevant signal edge of the incoming signal. When an edge in the transmitted data is detected, the logic will compare the location of the edge to the expected time (Sync_Seg). The circuit will then adjust the values of Phase Segment 1 and Phase Segment 2 as necessary. There are two mechanisms used for synchronization.

### 24.10.1 HARD SYNCHRONIZATION

Hard synchronization is only done when there is a recessive to dominant edge during a bus Idle condition, indicating the start of a message. After hard synchronization, the bit time counters are restarted with Sync_Seg. Hard synchronization forces the edge, which has occurred to lie within the synchronization segment of the restarted bit time. Due to the rules of synchronization, if a hard synchronization occurs, there will not be a resynchronization within that bit time.

### 24.10.2 RESYNCHRONIZATION

As a result of resynchronization, Phase Segment 1 may be lengthened or Phase Segment 2 may be shortened. The amount of lengthening or shortening of the phase buffer segments has an upper bound given by the Synchronization Jump Width (SJW). The value of the SJW will be added to Phase Segment 1 (see Figure 24-6) or subtracted from Phase Segment 2 (see Figure 24-7). The SJW is programmable between 1 $T_Q$ and 4 $T_Q$.

Clocking information will only be derived from recessive to dominant transitions. The property, that only a fixed maximum number of successive bits have the same value, ensures resynchronization to the bit stream during a frame.

The phase error of an edge is given by the position of the edge relative to Sync_Seg, measured in $T_Q$. The phase error is defined in magnitude of $T_Q$ as follows:

- e = 0 if the edge lies within Sync_Seg.
- e > 0 if the edge lies before the sample point.
- e < 0 if the edge lies after the sample point of the previous bit.

If the magnitude of the phase error is less than, or equal to, the programmed value of the Synchronization Jump Width, the effect of a resynchronization is the same as that of a hard synchronization.

If the magnitude of the phase error is larger than the Synchronization Jump Width and if the phase error is positive, then Phase Segment 1 is lengthened by an amount equal to the Synchronization Jump Width.

If the magnitude of the phase error is larger than the resynchronization jump width and if the phase error is negative, then Phase Segment 2 is shortened by an amount equal to the Synchronization Jump Width.

### 24.10.3 SYNCHRONIZATION RULES

- Only one synchronization within one bit time is allowed.
- An edge will be used for synchronization only if the value detected at the previous sample point (previously read bus value) differs from the bus value immediately after the edge.
- All other recessive to dominant edges fulfilling rules 1 and 2 will be used for resynchronization, with the exception that a node transmitting a dominant bit will not perform a resynchronization as a result of a recessive to dominant edge with a positive phase error.


© 2009 Microchip Technology Inc.

**FIGURE 24-6: LENGTHENING A BIT PERIOD (ADDING SJW TO PHASE SEGMENT 1)**

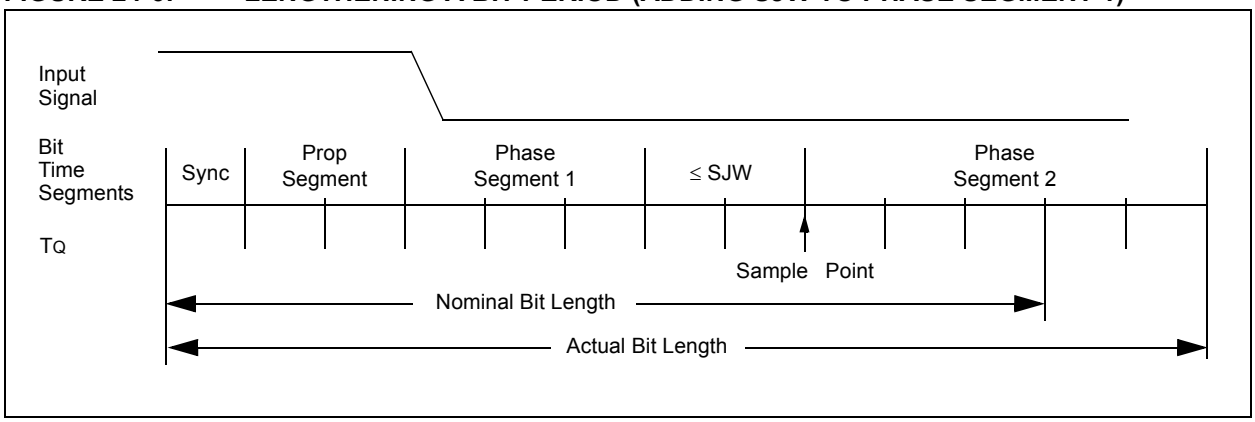
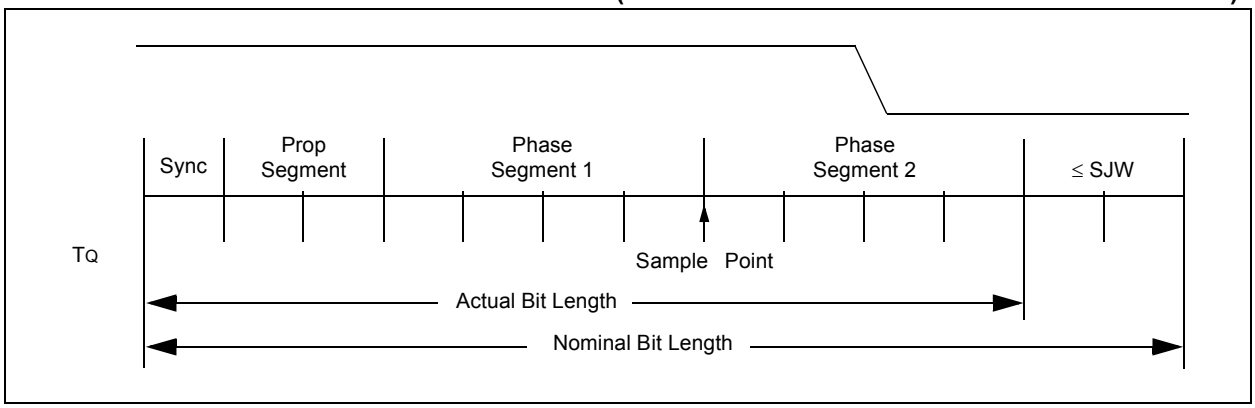



**FIGURE 24-7: SHORTENING A BIT PERIOD (SUBTRACTING SJW FROM PHASE SEGMENT 2)**




## 24.11 Programming Time Segments

Some requirements for programming of the time segments:

- Prop_Seg + Phase_Seg 1 ≥ Phase_Seg 2
- Phase_Seg 2 ≥ Sync Jump Width.

For example, assume that a 125 kHz CAN baud rate is desired, using 20 MHz for FOSC. With a TOSC of 50 ns, a baud rate prescaler value of 04h gives a TQ of 500 ns. To obtain a Nominal Bit Rate of 125 kHz, the Nominal Bit Time must be 8 μs or 16 TQ.

Using 1 TQ for the Sync_Seg, 2 TQ for the Prop_Seg and 7 TQ for Phase Segment 1 would place the sample point at 10 TQ after the transition. This leaves 6 TQ for Phase Segment 2.

By the rules above, the Sync Jump Width could be the maximum of 4 TQ. However, normally a large SJW is only necessary when the clock generation of the different nodes is inaccurate or unstable, such as using ceramic resonators. Typically, an SJW of 1 is enough.

## 24.12 Oscillator Tolerance

As a rule of thumb, the bit timing requirements allow ceramic resonators to be used in applications with transmission rates of up to 125 Kbit/sec. For the full bus speed range of the CAN protocol, a quartz oscillator is required. Refer to ISO11898-1 for oscillator tolerance requirements.

## 24.13 Bit Timing Configuration Registers

The Baud Rate Control registers (BRGCON1, BRGCON2, BRGCON3) control the bit timing for the CAN bus interface. These registers can only be modified when the PIC18F2480/2580/4480/4580 devices are in Configuration mode.

### 24.13.1 BRGCON1

The BRP bits control the baud rate prescaler. The SJW<1:0> bits select the synchronization jump width in terms of multiples of TQ.

### 24.13.2 BRGCON2

The PRSEG bits set the length of the propagation segment in terms of TQ. The SEG1PH bits set the length of Phase Segment 1 in TQ. The SAM bit controls how many times the RXCAN pin is sampled. Setting this bit to a '1' causes the bus to be sampled three times: twice at TQ/2 before the sample point and once at the normal sample point (which is at the end of Phase Segment 1). The value of the bus is determined to be the value read during at least two of the samples. If the SAM bit is set to a '0', then the RXCAN pin is sampled only once at the sample point. The SEG2PHTS bit controls how the length of Phase Segment 2 is determined. If this bit is set to a '1', then the length of Phase Segment 2 is determined by the SEG2PH bits of BRGCON3. If the SEG2PHTS bit is set to a '0', then the length of Phase Segment 2 is the greater of Phase Segment 1 and the information processing time (which is fixed at 2 TQ for the PIC18F2480/2580/4480/4580).

### 24.13.3 BRGCON3

The PHSEG2<2:0> bits set the length (in TQ) of Phase Segment 2 if the SEG2PHTS bit is set to a '1'. If the SEG2PHTS bit is set to a '0', then the PHSEG2<2:0> bits have no effect.

## 24.14 Error Detection

The CAN protocol provides sophisticated error detection mechanisms. The following errors can be detected.

### 24.14.1 CRC ERROR

With the Cyclic Redundancy Check (CRC), the transmitter calculates special check bits for the bit sequence, from the start of a frame until the end of the data field. This CRC sequence is transmitted in the CRC field. The receiving node also calculates the CRC sequence using the same formula and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an error frame is generated. The message is repeated.

### 24.14.2 ACKNOWLEDGE ERROR

In the Acknowledge field of a message, the transmitter checks if the Acknowledge slot (which was sent out as a recessive bit) contains a dominant bit. If not, no other node has received the frame correctly. An Acknowledge error has occurred, an error frame is generated and the message will have to be repeated.

### 24.14.3 FORM ERROR

If a node detects a dominant bit in one of the four segments, including End-Of-Frame (EOF), interframe space, Acknowledge delimiter or CRC delimiter, then a form error has occurred and an error frame is generated. The message is repeated.

### 24.14.4 BIT ERROR

A bit error occurs if a transmitter sends a dominant bit and detects a recessive bit, or if it sends a recessive bit and detects a dominant bit, when monitoring the actual bus level and comparing it to the just transmitted bit. In the case where the transmitter sends a recessive bit and a dominant bit is detected during the arbitration field and the Acknowledge slot, no bit error is generated because normal arbitration is occurring.

### 24.14.5 STUFF BIT ERROR

If, between the Start-Of-Frame (SOF) and the CRC delimiter, six consecutive bits with the same polarity are detected, the bit stuffing rule has been violated. A stuff bit error occurs and an error frame is generated. The message is repeated.

### 24.14.6 ERROR STATES

Detected errors are made public to all other nodes via error frames. The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states; "error-active", "error-passive" or "bus-off", according to the value of the internal error counters. The error-active state is the usual state where the bus node can transmit messages and activate error frames (made of dominant bits) without any restrictions. In the error-passive state, messages and passive error frames (made of recessive bits) may be transmitted. The bus-off state makes it temporarily impossible for the node to participate in the bus communication. During this state, messages can neither be received nor transmitted.

### 24.14.7 ERROR MODES AND ERROR COUNTERS

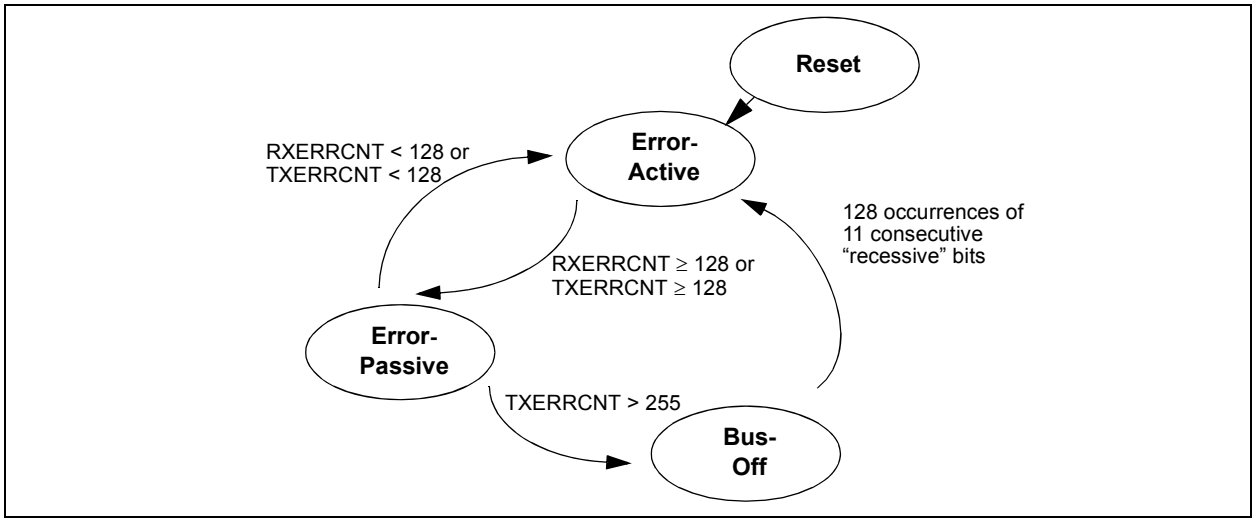
The PIC18F2480/2580/4480/4580 devices contain two error counters: the Receive Error Counter (RXERRCNT) and the Transmit Error Counter (TXERRCNT). The values of both counters can be read by the MCU. These counters are incremented or decremented in accordance with the CAN bus specification.


© 2009 Microchip Technology Inc.

The PIC18F2480/2580/4480/4580 devices are error-active if both error counters are below the error-passive limit of 128. They are error-passive if at least one of the error counters equals or exceeds 128. They go to bus-off if the transmit error counter equals or exceeds the bus-off limit of 256. The devices remain in this state until the bus-off recovery sequence is finished. The bus-off recovery sequence consists of 128 occurrences of 11 consecutive recessive bits (see Figure 24-8). Note that the CAN module, after going bus-off, will recover back to error-active without any intervention by the MCU if the bus remains Idle for 128 x 11 bit times. If this is not desired, the error Interrupt Service Routine should address this. The current Error mode of the CAN module can be read by the MCU via the COMSTAT register.

Additionally, there is an Error State Warning flag bit, EWARN, which is set if at least one of the error counters equals or exceeds the error warning limit of 96. EWARN is reset if both error counters are less than the error warning limit.

**FIGURE 24-8: ERROR MODES STATE DIAGRAM**




## 24.15 CAN Interrupts

The module has several sources of interrupts. Each of these interrupts can be individually enabled or disabled. The PIR3 register contains interrupt flags. The PIE3 register contains the enables for the 8 main interrupts. A special set of read-only bits in the CANSTAT register, the ICODE bits, can be used in combination with a jump table for efficient handling of interrupts.

All interrupts have one source, with the exception of the error interrupt and buffer interrupts in Mode 1 and 2. Any of the error interrupt sources can set the error interrupt flag. The source of the error interrupt can be determined by reading the Communication Status register, COMSTAT. In Mode 1 and 2, there are two interrupt enable/disable and flag bits – one for all transmit buffers and the other for all receive buffers.

The interrupts can be broken up into two categories: receive and transmit interrupts.

The receive related interrupts are:

- Receive Interrupts
- Wake-up Interrupt
- Receiver Overrun Interrupt
- Receiver Warning Interrupt
- Receiver Error-Passive Interrupt

The transmit related interrupts are:

- Transmit Interrupts
- Transmitter Warning Interrupt
- Transmitter Error-Passive Interrupt
- Bus-Off Interrupt

### 24.15.1 INTERRUPT CODE BITS

To simplify the interrupt handling process in user firmware, the ECAN module encodes a special set of bits. In Mode 0, these bits are ICODE<3:1> in the CANSTAT register. In Mode 1 and 2, these bits are EICODE<4:0> in the CANSTAT register. Interrupts are internally prioritized such that the higher priority interrupts are assigned lower values. Once the highest priority interrupt condition has been cleared, the code for the next highest priority interrupt that is pending (if any) will be reflected by the ICODE bits (see Table 24-5). Note that only those interrupt sources that have their associated interrupt enable bit set will be reflected in the ICODE bits.

In Mode 2, when a receive message interrupt occurs, the EICODE bits will always consist of '10000'. User firmware may use FIFO Pointer bits to actually access the next available buffer.

### 24.15.2 TRANSMIT INTERRUPT

When the transmit interrupt is enabled, an interrupt will be generated when the associated transmit buffer becomes empty and is ready to be loaded with a new message. In Mode 0, there are separate interrupt enable/disable and flag bits for each of the three dedicated transmit buffers. The TXBnIF bit will be set to indicate the source of the interrupt. The interrupt is cleared by the MCU, resetting the TXBnIF bit to a '0'. In Mode 1 and 2, all transmit buffers share one interrupt enable/disable bit and one flag bit. In Mode 1 and 2, TXBIE in PIE3 and TXBIF in PIR3 indicate when a transmit buffer has completed transmission of its message. TXBnIF, TXBnIE and TXBnIP in PIR3, PIE3 and IPR3, respectively, are not used in Mode 1 and 2. Individual transmit buffer interrupts can be enabled or disabled by setting or clearing TXBIE and B0IE register bits. When a shared interrupt occurs, user firmware must poll the TXREQ bit of all transmit buffers to detect the source of interrupt.

### 24.15.3 RECEIVE INTERRUPT

When the receive interrupt is enabled, an interrupt will be generated when a message has been successfully received and loaded into the associated receive buffer. This interrupt is activated immediately after receiving the End-Of-Frame (EOF) field.

In Mode 0, the RXBnIF bit is set to indicate the source of the interrupt. The interrupt is cleared by the MCU, resetting the RXBnIF bit to a '0'.

In Mode 1 and 2, all receive buffers share RXBIE, RXBIF and RXBIP in PIE3, PIR3 and IPR3, respectively. Bits, RXBnIE, RXBnIF and RXBnIP, are not used. Individual receive buffer interrupts can be controlled by the TXBIE and BIE0 registers. In Mode 1, when a shared receive interrupt occurs, user firmware must poll the RXFUL bit of each receive buffer to detect the source of interrupt. In Mode 2, a receive interrupt indicates that the new message is loaded into FIFO. FIFO can be read by using FIFO Pointer bits, FP.

**TABLE 24-5: VALUES FOR ICODE<2:0>**

| ICODE <2:0> | Interrupt | Boolean Expression |
|---|---|---|
| 000 | None | $\overline{ERR}$•$\overline{WAK}$•$\overline{TX0}$•$\overline{TX1}$•$\overline{TX2}$•$\overline{RX0}$•$\overline{RX1}$ |
| 001 | Error | ERR |
| 010 | TXB2 | $\overline{ERR}$•$\overline{TX0}$•$\overline{TX1}$•TX2 |
| 011 | TXB1 | $\overline{ERR}$•$\overline{TX0}$•TX1 |
| 100 | TXB0 | $\overline{ERR}$•TX0 |
| 101 | RXB1 | $\overline{ERR}$•$\overline{TX0}$•$\overline{TX1}$•$\overline{TX2}$•$\overline{RX0}$•RX1 |
| 110 | RXB0 | $\overline{ERR}$•$\overline{TX0}$•$\overline{TX1}$•$\overline{TX2}$•RX0 |
| 111 | Wake on Interrupt | $\overline{ERR}$•$\overline{TX0}$•$\overline{TX1}$•$\overline{TX2}$•$\overline{RX0}$•$\overline{RX1}$•WAK |

**Legend:**
ERR = ERRIF * ERRIE RX0 = RXB0IF * RXB0IE
TX0 = TXB0IF * TXB0IE RX1 = RXB1IF * RXB1IE
TX1 = TXB1IF * TXB1IE WAK = WAKIF * WAKIE
TX2 = TXB2IF * TXB2IE

### 24.15.4 MESSAGE ERROR INTERRUPT

When an error occurs during transmission or reception of a message, the message error flag, IRXIF, will be set and if the IRXIE bit is set, an interrupt will be generated. This is intended to be used to facilitate baud rate determination when used in conjunction with Listen Only mode.

### 24.15.5 BUS ACTIVITY WAKE-UP INTERRUPT

When the PIC18F2480/2580/4480/4580 devices are in Sleep mode and the bus activity wake-up interrupt is enabled, an interrupt will be generated and the WAKIF bit will be set when activity is detected on the CAN bus. This interrupt causes the PIC18F2480/2580/4480/4580 devices to exit Sleep mode. The interrupt is reset by the MCU, clearing the WAKIF bit.

### 24.15.6 ERROR INTERRUPT

When the CAN module error interrupt (ERRIE in PIE3) is enabled, an interrupt is generated if an overflow condition occurs, or if the error state of the transmitter or receiver has changed. The error flags in COMSTAT will indicate one of the following conditions.


© 2009 Microchip Technology Inc.

#### 24.15.6.1 Receiver Overflow

An overflow condition occurs when the MAB has assembled a valid received message (the message meets the criteria of the acceptance filters) and the receive buffer associated with the filter is not available for loading of a new message. The associated RXBnOVFL bit in the COMSTAT register will be set to indicate the overflow condition. This bit must be cleared by the MCU.

#### 24.15.6.2 Receiver Warning

The receive error counter has reached the MCU warning limit of 96.

#### 24.15.6.3 Transmitter Warning

The transmit error counter has reached the MCU warning limit of 96.

#### 24.15.6.4 Receiver Bus Passive

This will occur when the device has gone to the error-passive state because the receive error counter is greater or equal to 128.

#### 24.15.6.5 Transmitter Bus Passive

This will occur when the device has gone to the error-passive state because the transmit error counter is greater or equal to 128.

#### 24.15.6.6 Bus-Off

The transmit error counter has exceeded 255 and the device has gone to bus-off state.

© 2009 Microchip Technology Inc.

**NOTES:**


© 2009 Microchip Technology Inc.

## 25.0 SPECIAL FEATURES OF THE CPU

PIC18F2480/2580/4480/4580 devices include several features intended to maximize reliability and minimize cost through elimination of external components. 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

The oscillator can be configured for the application depending on frequency, power, accuracy and cost. All of the options are discussed in detail in **Section 3.0 "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, PIC18F2480/2580/4480/4580 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.

### 25.1 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 WR bit in the EECON1 register 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 7.5 "Writing to Flash Program Memory"**.

**TABLE 25-1: 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 | FCMEN | — | — | FOSC3 | FOSC2 | FOSC1 | FOSC0 | 00-- 0111 |
| 300002h | CONFIG2L | — | — | — | BORV1 | BORV0 | BOREN1 | BOREN0 | $\overline{\text{PWRTEN}}$ | ---1 1111 |
| 300003h | CONFIG2H | — | — | — | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN | ---1 1111 |
| 300005h | CONFIG3H | MCLRE | — | — | — | — | LPT1OSC | PBADEN | — | 1--- -01- |
| 300006h | CONFIG4L | $\overline{\text{DEBUG}}$ | XINST | — | BBSIZ | — | LVP | — | STVREN | 10-0 -1-1 |
| 300008h | CONFIG5L | — | — | — | — | CP3 | CP2 | CP1 | CP0 | ---- 1111 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — | 11-- ---- |
| 30000Ah | CONFIG6L | — | — | — | — | WRT3 | WRT2 | WRT1 | WRT0 | ---- 1111 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC | — | — | — | — | — | 111- ---- |
| 30000Ch | CONFIG7L | — | — | — | — | EBTR3 | EBTR2 | EBTR1 | EBTR0 | ---- 1111 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — | -1-- ---- |
| 3FFFFEh | DEVID1 | DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 | xxxx xxxx(1) |
| 3FFFFFh | DEVID2 | DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 | 0000 1100 |

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

**Note 1:** See Register 25-12 for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

**REGISTER 25-1: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)**

| R/P-0 | R/P-0 | U-0 | U-0 | R/P-0 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| IESO | FCMEN | — | — | FOSC3 | FOSC2 | FOSC1 | FOSC0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

bit 7 **IESO:** Internal/External Oscillator Switchover bit
1 = Oscillator Switchover mode enabled
0 = Oscillator Switchover mode disabled

bit 6 **FCMEN:** 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
101x = External RC oscillator, CLKO function on RA6
1001 = Internal oscillator block, CLKO function on RA6, port function on RA7
1000 = Internal oscillator block, port function on RA6 and 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
0011 = External RC oscillator, CLKO function on RA6
0010 = HS oscillator
0001 = XT oscillator
0000 = LP oscillator


© 2009 Microchip Technology Inc.

**REGISTER 25-2: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)**

| U-0 | U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | BORV1 | BORV0 | BOREN1(1) | BOREN0(1) | $\overline{\text{PWRTEN}}$(1) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

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

bit 4-3 **BORV<1:0>:** Brown-out Reset Voltage bits
11 = $V_{BOR}$ set to 2.1V
10 = $V_{BOR}$ set to 2.8V
01 = $V_{BOR}$ set to 4.3V
00 = $V_{BOR}$ set to 4.6V

bit 2-1 **BOREN<1:0>:** Brown-out Reset Enable bits(1)
11 = Brown-out Reset enabled in hardware only (SBOREN is disabled)
10 = Brown-out Reset enabled in hardware only and disabled in Sleep mode (SBOREN is disabled)
01 = Brown-out Reset enabled and controlled by software (SBOREN is enabled)
00 = Brown-out Reset disabled in hardware and software

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 25-3: 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 | P = Programmable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

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 **WDTEN:** Watchdog Timer Enable bit

1 = WDT enabled
0 = WDT disabled (control is placed on the SWDTEN bit)


© 2009 Microchip Technology Inc.

**REGISTER 25-4: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)**

| R/P-1 | U-0 | U-0 | U-0 | U-0 | R/P-0 | R/P-1 | U-0 |
|---|---|---|---|---|---|---|---|
| MCLRE | — | — | — | — | LPT1OSC | PBADEN | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as ‘0’ |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

bit 7 **MCLRE:** $\overline{\text{MCLR}}$ Pin Enable bit
1 = $\overline{\text{MCLR}}$ pin enabled; RE3 input pin disabled
0 = RE3 input pin enabled; $\overline{\text{MCLR}}$ disabled

bit 6-3 **Unimplemented:** Read as ‘0’

bit 2 **LPT1OSC:** Low-Power Timer1 Oscillator Enable bit
1 = Timer1 configured for low-power operation
0 = Timer1 configured for higher power operation

bit 1 **PBADEN:** PORTB A/D Enable bit
(Affects ADCON1 Reset state. ADCON1 controls PORTB<4:0> pin configuration.)
1 = PORTB<4:0> pins are configured as analog input channels on Reset
0 = PORTB<4:0> pins are configured as digital I/O on Reset

bit 0 **Unimplemented:** Read as ‘0’

**REGISTER 25-5: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)**

| R/P-1 | R/P-0 | U-0 | R/P-0 | U-0 | R/P-1 | U-0 | R/P-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{DEBUG}}$ | XINST | — | BBSIZ | — | LVP | — | STVREN |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as ‘0’ |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

bit 7 **$\overline{\text{DEBUG}}$:** Background Debugger Enable bit
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 **XINST:** Extended Instruction Set Enable bit
1 = Instruction set extension and Indexed Addressing mode enabled
0 = Instruction set extension and Indexed Addressing mode disabled (Legacy mode)

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **BBSIZ:** Boot Block Size Select Bit 0
01 = 2K words (4 Kbytes) boot block
00 = 1K words (2 Kbytes) boot block

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **LVP:** Single-Supply ICSP™ Enable bit
1 = Single-Supply ICSP enabled
0 = Single-Supply ICSP disabled

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **STVREN:** Stack Full/Underflow Reset Enable bit
1 = Stack full/underflow will cause Reset
0 = Stack full/underflow will not cause Reset

**REGISTER 25-6: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)**

| U-0 | U-0 | U-0 | U-0 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | CP3(1) | CP2(1) | CP1 | CP0 |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---|---|---|
| R = Readable bit | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **CP3:** Code Protection bit(1)
1 = Block 3 (006000-007FFFh) not code-protected
0 = Block 3 (006000-007FFFh) code-protected

bit 2 **CP2:** Code Protection bit(1)
1 = Block 2 (004000-005FFFh) not code-protected
0 = Block 2 (004000-005FFFh) code-protected

bit 1 **CP1:** Code Protection bit
1 = Block 1 (002000-003FFFh) not code-protected
0 = Block 1 (002000-003FFFh) code-protected

bit 0 **CP0:** Code Protection bit
1 = Block 0 (000800-001FFFh) not code-protected
0 = Block 0 (000800-001FFFh) code-protected

**Note 1:** Unimplemented in PIC18FX480 devices; maintain this bit set.

**REGISTER 25-7: 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 | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

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 (000000-0007FFh) not code-protected
0 = Boot Block (000000-0007FFh) code-protected

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


© 2009 Microchip Technology Inc.

**REGISTER 25-8: CONFIG6L: CONFIGURATION REGISTER 6 LOW (BYTE ADDRESS 30000Ah)**

| U-0 | U-0 | U-0 | U-0 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | WRT3(1) | WRT2(1) | WRT1 | WRT0 |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---|---|---|
| R = Readable bit | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **WRT3:** Write Protection bit(1)
1 = Block 3 (006000-007FFFh) not write-protected
0 = Block 3 (006000-007FFFh) write-protected

bit 2 **WRT2:** Write Protection bit(1)
1 = Block 2 (004000-005FFFh) not write-protected
0 = Block 2 (004000-005FFFh) write-protected

bit 1 **WRT1:** Write Protection bit
1 = Block 1 (002000-003FFFh) not write-protected
0 = Block 1 (002000-003FFFh) write-protected

bit 0 **WRT0:** Write Protection bit
1 = Block 0 (000800-001FFFh) not write-protected
0 = Block 0 (000800-001FFFh) write-protected

**Note 1:** Unimplemented in PIC18FX480 devices; maintain this bit set.

**REGISTER 25-9: CONFIG6H: CONFIGURATION REGISTER 6 HIGH (BYTE ADDRESS 30000Bh)**

| R/C-1 | R/C-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 | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

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 (000000-0007FFh) not write-protected
0 = Boot Block (000000-0007FFh) 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 25-10: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 30000Ch)**

| U-0 | U-0 | U-0 | U-0 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | EBTR3(1,2) | EBTR2(1,2) | EBTR1(2) | EBTR0(2) |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---|---|---|
| R = Readable bit | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **EBTR3:** Table Read Protection bit(1,2)
1 = Block 3 (006000-007FFFh) not protected from table reads executed in other blocks
0 = Block 3 (006000-007FFFh) protected from table reads executed in other blocks

bit 2 **EBTR2**: Table Read Protection bit(1,2)
1 = Block 2 (004000-005FFFh) not protected from table reads executed in other blocks
0 = Block 2 (004000-005FFFh) protected from table reads executed in other blocks

bit 1 **EBTR1:** Table Read Protection bit(2)
1 = Block 1 (002000-003FFFh) not protected from table reads executed in other blocks
0 = Block 1 (002000-003FFFh) protected from table reads executed in other blocks

bit 0 **EBTR0:** Table Read Protection bit(2)
1 = Block 0 (000800-001FFFh) not protected from table reads executed in other blocks
0 = Block 0 (000800-001FFFh) protected from table reads executed in other blocks

**Note 1:** Unimplemented in PIC18FX480 devices; maintain this bit set.
**2:** It is recommended to enable the corresponding CPx bit to protect the block from external read operations.

**REGISTER 25-11: CONFIG7H: CONFIGURATION REGISTER 7 HIGH (BYTE ADDRESS 30000Dh)**

| U-0 | R/C-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | EBTRB(1) | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---|---|---|
| R = Readable bit | C = Clearable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

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

bit 6 **EBTRB:** Boot Block Table Read Protection bit(1)
1 = Boot Block (000000-0007FFh) not protected from table reads executed in other blocks
0 = Boot Block (000000-0007FFh) protected from table reads executed in other blocks

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

**Note 1:** It is recommended to enable the corresponding CPx bit to protect the block from external read operations.


© 2009 Microchip Technology Inc.

**REGISTER 25-12: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F2480/2580/4480/4580**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Read-only bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

bit 7-5 **DEV<2:0>:** Device ID bits
111 = PIC18F2480
110 = PIC18F2580
101 = PIC18F4480
100 = PIC18F4580

bit 4-0 **REV<3:0>:** Revision ID bits
These bits are used to indicate the device revision.

**REGISTER 25-13: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F2480/2580/4480/4580**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Read-only bit | P = Programmable bit | U = Unimplemented bit, read as '0' |
| -n = Value when device is unprogrammed | | u = Unchanged from programmed state |

bit 7-0 **DEV<10:3>:** Device ID bits
These bits are used with the DEV<2:0> bits in Device ID Register 1 to identify the part number.
0001 1010 = PIC18F2480/2580/4480/4580 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.

## 25.2 Watchdog Timer (WDT)

For PIC18F2480/2580/4480/4580 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: a SLEEP or CLRWDT instruction is executed, the IRCF bits (OSCCON<6:4>) are changed or a clock failure has occurred.

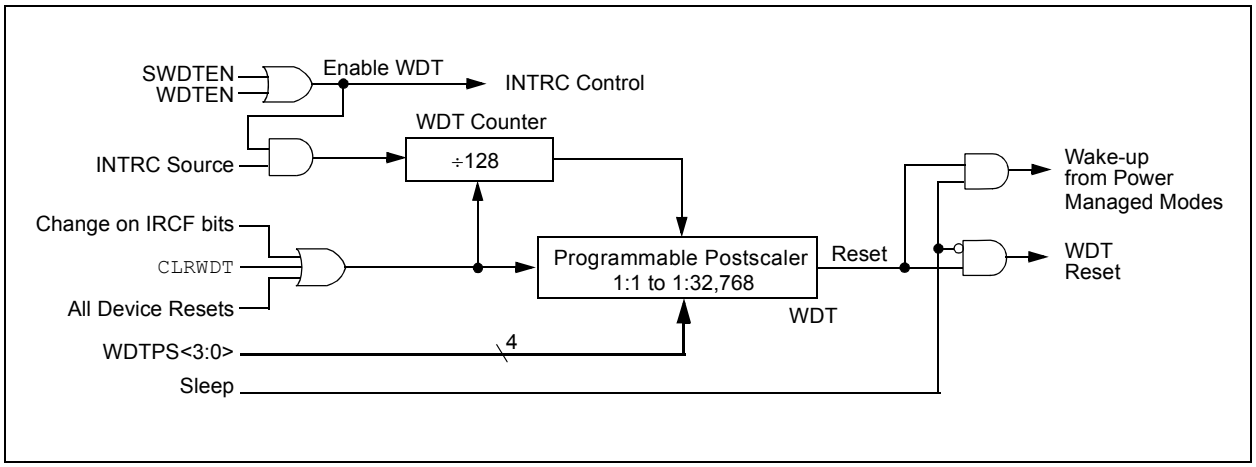
.

| | |
|---|---|
| **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. |

### 25.2.1 CONTROL REGISTER

Register 25-14 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, but only if the Configuration bit has disabled the WDT.

**FIGURE 25-1: WDT BLOCK DIAGRAM**





© 2009 Microchip Technology Inc.

**REGISTER 25-14: WDTCON: WATCHDOG TIMER CONTROL REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | SWDTEN(1) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 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, is enabled.

**TABLE 25-2: SUMMARY OF WATCHDOG TIMER REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| RCON | IPEN | SBOREN | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 54 |
| WDTCON | — | — | — | — | — | — | — | SWDTEN | 56 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Watchdog Timer.

## 25.3 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 Configuration bit.

Two-Speed Start-up should be enabled only if the primary oscillator mode is LP, XT, HS or HSPLL (Crystal-Based modes). Other sources do not require an Oscillator Start-up Timer delay; for these, Two-Speed Start-up should be 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, IRCF<2:0>, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF2:IRCF0 bits prior to entering Sleep mode.

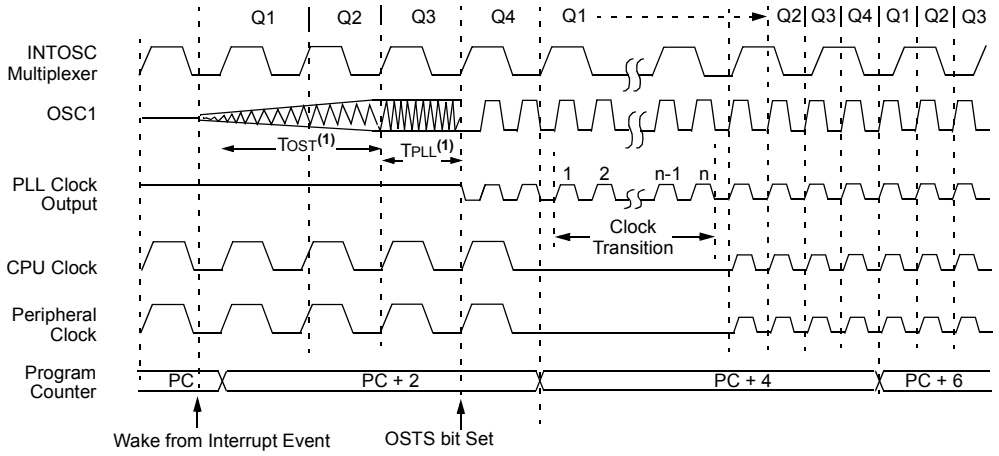
In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

### 25.3.1 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 4.1.4 "Multiple Sleep Commands"**). In practice, this means that user code can change the SCS<1:0> bit settings or issue `SLEEP` instructions 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 device clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

**FIGURE 25-2: 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.

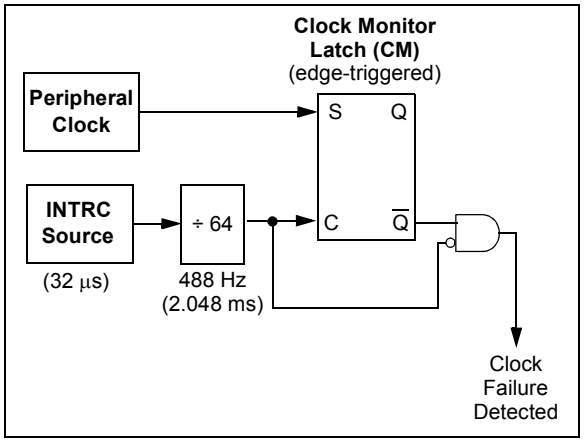

© 2009 Microchip Technology Inc.

## 25.4 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 device clock to the internal oscillator block. The FSCM function is enabled by setting the FCMEN Configuration bit.

When FSCM is enabled, the INTRC oscillator runs at all times to monitor clocks to peripherals and provide a backup clock in the event of a clock failure. Clock monitoring (shown in Figure 25-3) 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 device clock and the sample clock are presented as inputs to the Clock Monitor (CM) latch. The CM is set on the falling edge of the device clock source, but cleared on the rising edge of the sample clock.

**FIGURE 25-3: 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 25-4). This causes the following:

- the FSCM generates an oscillator fail interrupt by setting bit, OSCFIF (PIR2<7>);
- the device 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.

During switchover, the postscaler frequency from the internal oscillator block may not be sufficiently stable for timing-sensitive applications. In these cases, it may be desirable to select another clock configuration and enter an alternate power-managed mode. This can be done to attempt a partial recovery or execute a controlled shutdown. See **Section 4.1.4 "Multiple Sleep Commands"** and **Section 25.3.1 "Special Considerations for Using Two-Speed Start-up"** for more details.

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, IRCF<2:0>, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF<2:0> bits prior to entering Sleep mode.

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.

### 25.4.1 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.

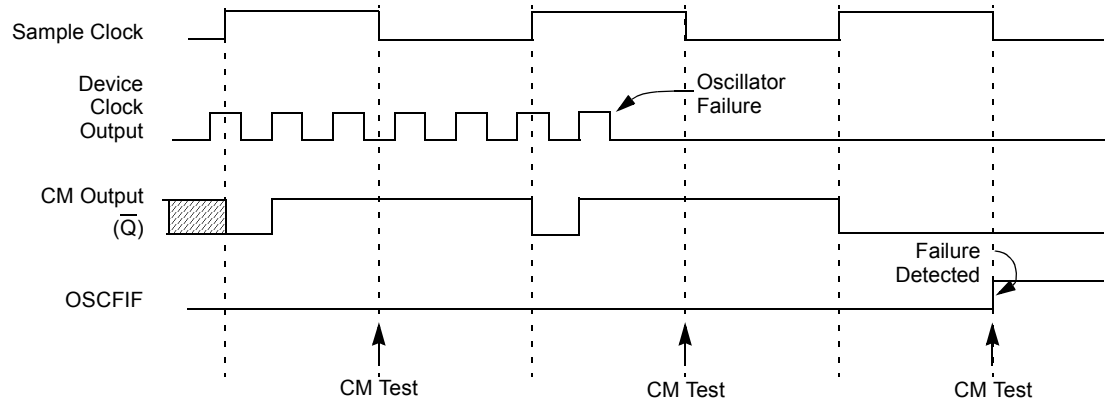
As already noted, the clock source is switched to the INTOSC clock when a clock failure is detected. Depending on the frequency selected by the IRCF<2:0> bits, this may mean a substantial change in the speed of code execution. If the WDT is enabled with a small prescale value, a decrease in clock speed 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.

### 25.4.2 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 device clock until the primary clock source becomes ready (similar to a Two-Speed Start-up). The clock 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.

**FIGURE 25-4: FSCM TIMING DIAGRAM**




**Note:** The device clock is normally at a much higher frequency than the sample clock. The relative frequencies in this example have been chosen for clarity.

### 25.4.3 FSCM INTERRUPTS IN POWER-MANAGED MODES

By entering a power-managed mode, the clock multiplexer selects the clock source selected by the OSCCON register. Fail-Safe Clock 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, subsequent interrupts while in Idle mode will cause the CPU to begin executing instructions while being clocked by the INTOSC source.

### 25.4.4 POR OR WAKE-UP 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 device 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 device 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 25.3.1 "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 clock to become stable. When the new power-managed mode is selected, the primary clock is disabled.


© 2009 Microchip Technology Inc.

## 25.5 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 five blocks. One of these is a boot block of 2 Kbytes. The remainder of the memory is divided into four blocks on binary boundaries.

Each of the five blocks has three code protection bits associated with them. They are:

- Code-Protect bit (CPn)
- Write-Protect bit (WRTn)
- External Block Table Read bit (EBTRn)

Figure 25-5 shows the program memory organization for 16 and 32-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 25-3.

**FIGURE 25-5: CODE-PROTECTED PROGRAM MEMORY FOR PIC18F2480/2580/4480/4580**

| Address Range | MEMORY SIZE/DEVICE | | | | Block Code Protection Controlled by: |
|---|---|---|---|---|---|
| | 32 Kbytes (PIC18F2580/4580) | | 16 Kbytes (PIC18F2480/4480) | | |
| BBSIZ | 0 | 1 | 0 | 1 | |
| 000000h | Boot Block 1 kW | Boot Block 2 kW | Boot Block 1 kW | Boot Block 2 kW | CPB, WRTB, EBRTB (Boot Block) |
| 0007FFh | | | | | |
| 000800h | Block 0 3 kW | | Block 0 3 kW | | |
| 000FFFh | | | | | |
| 001000h | | Block 0 2 kW | | Block 0 2 kW | CP0, WRT0, EBRT0 (Block 0) |
| 001FFFh | | | | | |
| 002000h | Block 1 4 kW | Block 1 4 kW | Block 1 4 kW | Block 1 4 kW | CP!, WRT1, EBRT1 (Block 1) |
| 003FFFh | | | | | |
| 004000h | Block 2 4 kW | Block 2 4 kW | Unimplemented Read '0's | Unimplemented Read '0's | CP2, WRT2, EBRT2 (Block 2) |
| 005FFFh | | | | | |
| 006000h | Block 3 4 kW | Block 3 4 kW | | | CP3, WRT3, EBTR3 (Block 3) |
| 007FFFh | | | | | |
| 008000h | Unimplemented Read '0's | Unimplemented Read '0's | | | (Unimplemented Memory Space) |
| 1FFFFFh | | | | | |

**TABLE 25-3: 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 | — | — | — | — | CP3* | CP2 | CP1 | CP0 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — |
| 30000Ah | CONFIG6L | — | — | — | — | WRT3* | WRT2 | WRT1 | WRT0 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC | — | — | — | — | — |
| 30000Ch | CONFIG7L | — | — | — | — | EBTR3* | EBTR2 | EBTR1 | EBTR0 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — |

**Legend:** Shaded cells are unimplemented.

* Unimplemented in PIC18FX480 devices; maintain this bit set.

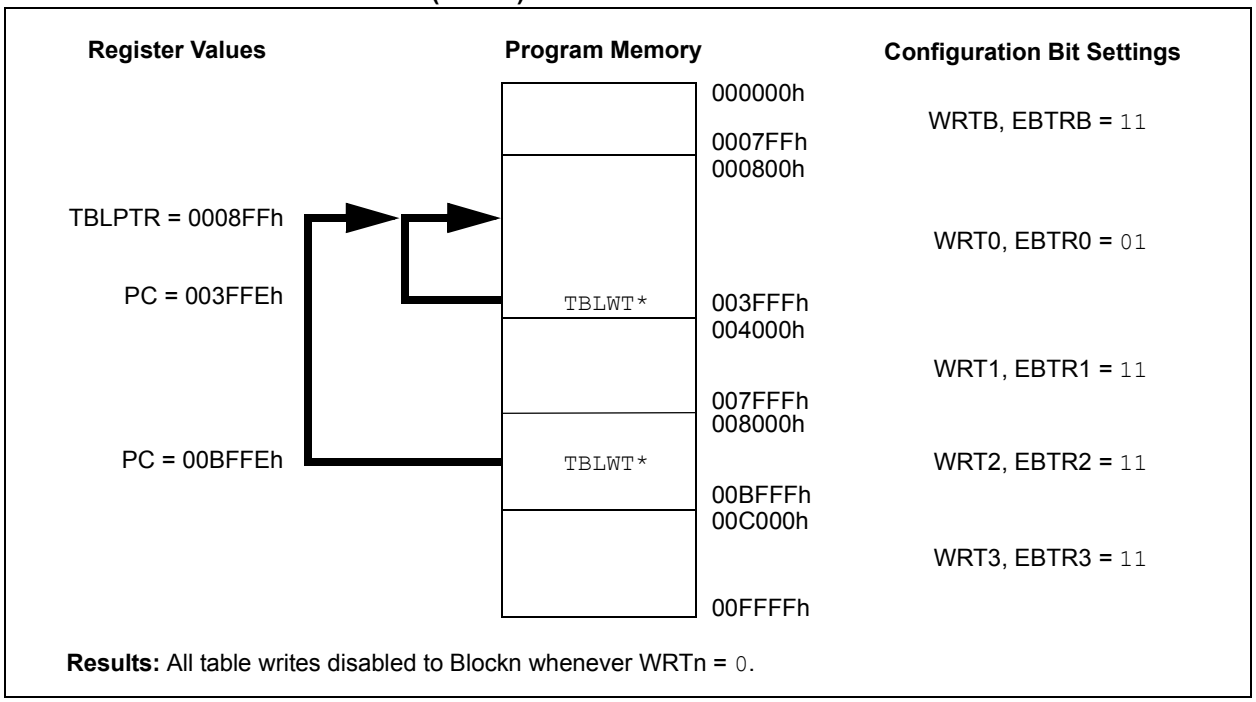
### 25.5.1 PROGRAM MEMORY CODE PROTECTION

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 25-6 through 25-8 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. |

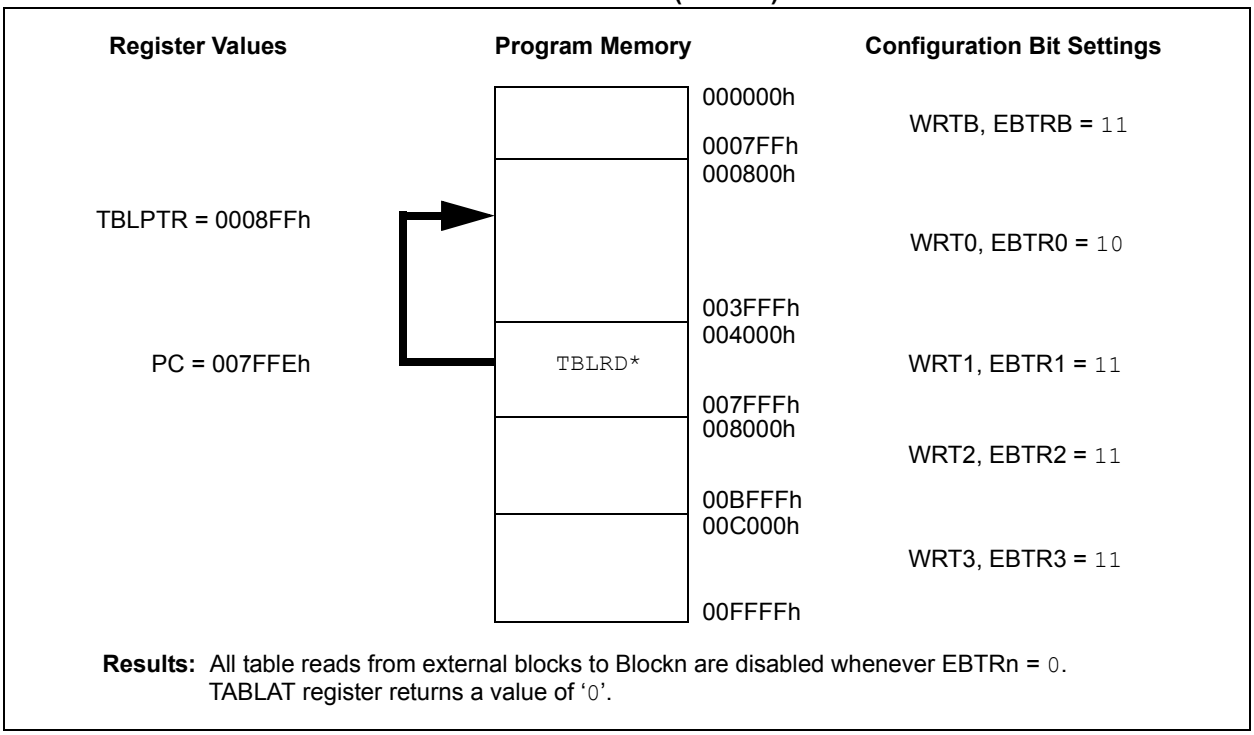
**FIGURE 25-6: TABLE WRITE (WRTn) DISALLOWED**




**Results:** All table writes disabled to Blockn whenever WRTn = 0.

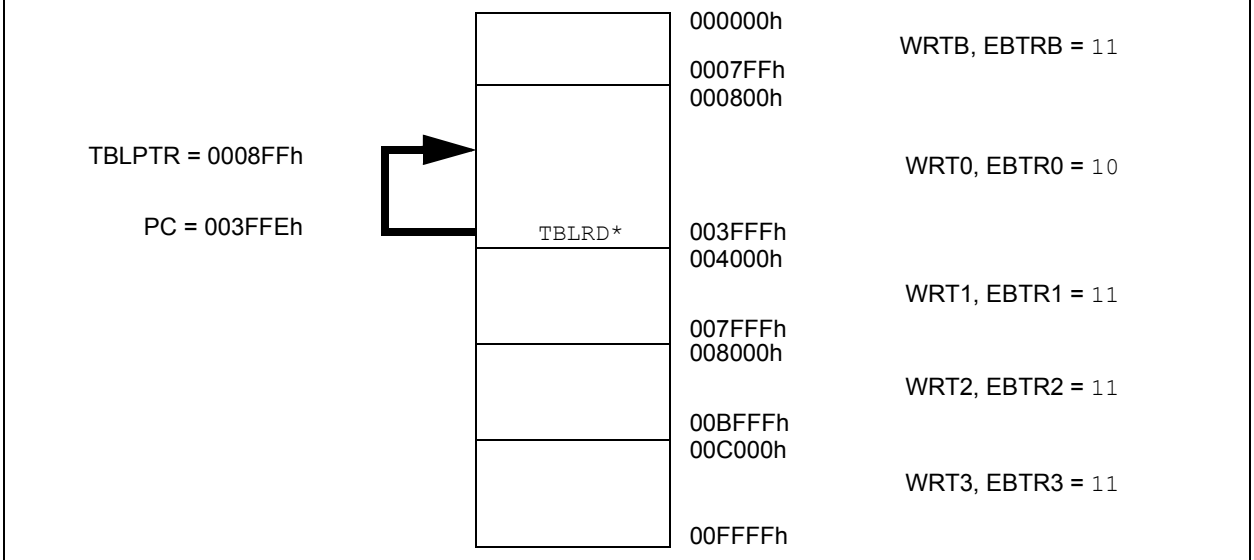

© 2009 Microchip Technology Inc.

**FIGURE 25-7: EXTERNAL BLOCK TABLE READ (EBTRn) DISALLOWED**




**Results:** All table reads from external blocks to Blockn are disabled whenever EBTRn = 0.
TABLAT register returns a value of '0'.

**FIGURE 25-8: EXTERNAL BLOCK TABLE READ (EBTRn) ALLOWED**




**Results:** Table reads permitted within Blockn, even when EBTRBn = 0.
TABLAT register returns the value of the data at the location TBLPTR.

### 25.5.2 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 internal and external writes to data EEPROM. The CPU can continue to read and write data EEPROM regardless of the protection bit settings.

### 25.5.3 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.

## 25.6 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.

## 25.7 In-Circuit Serial Programming

PIC18F2480/2580/4480/4580 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.

## 25.8 In-Circuit Debugger

When the DEBUG Configuration bit 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 25-4 shows which resources are required by the background debugger.

**TABLE 25-4: DEBUGGER RESOURCES**

| | |
|---|---|
| I/O pins: | RB6, RB7 |
| Stack: | 2 levels |

**Note:** Memory resources listed in MPLAB® IDE.

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to MCLR/VPP/RE3, 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.

## 25.9 Single-Supply ICSP Programming

The LVP Configuration bit enables Single-Supply ICSP Programming (formerly known as *Low-Voltage ICSP Programming* or *LVP*). When Single-Supply Programming is enabled, the microcontroller can be programmed without requiring high voltage being applied to the MCLR/VPP/RE3 pin, but the RB5/KBI1/PGM pin is then dedicated to controlling Program mode entry and is not available as a general purpose I/O pin.

While programming using Single-Supply Programming, VDD is applied to the MCLR/VPP/RE3 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, by applying VIHH to the MCLR pin.

**2:** While in Low-Voltage ICSP Programming mode, the RB5 pin can no longer be used as a general purpose I/O pin and should be held low during normal operation.

**3:** When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 5 in the TRISB register must be cleared to disable the pull-up on RB5 and ensure the proper operation of the device.

**4:** If the device Master Clear is disabled, verify that either of the following is done to ensure proper entry into ICSP mode:

a) disable Low-Voltage Programming (CONFIG4l<2> = 0); or

b) make certain that RB5/PGM is held low during entry into ICSP.

If Single-Supply ICSP Programming mode will not be used, the LVP bit can be cleared. RB5/KBI1/PGM then 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/RE3 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.


© 2009 Microchip Technology Inc.

## 26.0 INSTRUCTION SET SUMMARY

PIC18F2480/2580/4480/4580 devices incorporate the standard set of 75 PIC18 core instructions, as well as an extended set of 8 new instructions for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

### 26.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are four 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 26-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 26-1 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 ‘0’, the result is placed in the WREG register. If ‘d’ is ‘1’, 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 four double-word instructions. These instructions were made double-word to contain the required information in 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 26-1 shows the general formats that the instructions can have. All examples use the convention ‘nnh’ to represent a hexadecimal number.

The instruction set summary, shown in Table 26-2, lists the standard instructions recognized by the Microchip MPASM Assembler.

**Section 26.1.1 “Standard Instruction Set”** provides a description of each instruction.

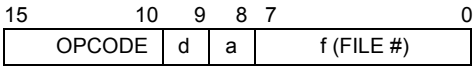
**TABLE 26-1: OPCODE FIELD DESCRIPTIONS**

| 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. |
| C, DC, Z, OV, N | ALU Status bits: **C**arry, **D**igit **C**arry, **Z**ero, **Ov**erflow, **N**egative. |
| 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 (00h to FFh), or 2-bit FSR designator (0h to 3h). |
| $f_s$ | 12-bit Register file address (000h to FFFh). This is the source address. |
| $f_d$ | 12-bit Register file address (000h to FFFh). This is the destination address. |
| GIE | Global Interrupt Enable bit. |
| 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 |
| 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. |
| $\overline{PD}$ | Power-down bit. |
| 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) |
| TBLPTR | 21-bit Table Pointer (points to a program memory location). |
| TABLAT | 8-bit Table Latch. |
| $\overline{TO}$ | Time-out bit. |
| TOS | Top-of-Stack. |
| u | Unused or unchanged. |
| WDT | Watchdog Timer. |
| WREG | Working register (accumulator). |
| x | Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools. |
| $z_s$ | 7-bit offset value for indirect addressing of register files (source). |
| $z_d$ | 7-bit offset value for indirect addressing of register files (destination). |
| { } | Optional argument. |
| [text] | Indicates an indexed address. |
| (text) | The contents of text. |
| [expr]<n> | Specifies bit n of the register indicated by the pointer expr. |
| → | Assigned to. |
| < > | Register bit field. |
| ∈ | In the set of. |
| italics | User-defined term (font is Courier New). |


© 2009 Microchip Technology Inc.

**FIGURE 26-1: GENERAL FORMAT FOR INSTRUCTIONS**

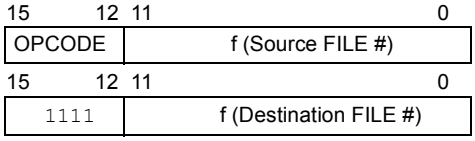
**Byte-oriented** file register operations — **Example Instruction**




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

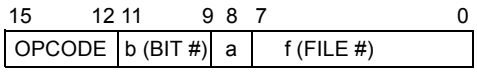
**Byte to Byte** move operations (2-word)




MOVFF MYREG1, MYREG2

f = 12-bit file register address

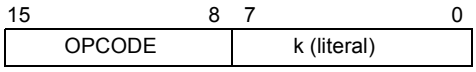
**Bit-oriented** file register operations




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

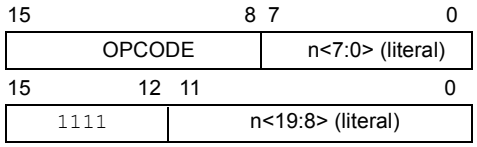



MOVLW 7Fh

k = 8-bit immediate value

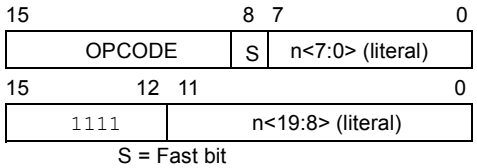
**Control** operations

**CALL, GOTO and Branch** operations



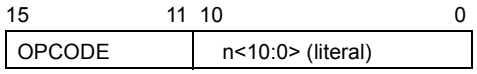

GOTO Label

n = 20-bit immediate value

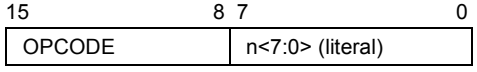



CALL MYFUNC

S = Fast bit




BRA MYFUNC




BC MYFUNC

**TABLE 26-2: PIC18FXXXX INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word MSb LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|
| **BYTE-ORIENTED 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 ffff ffff ffff<br>1111 ffff ffff 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 | 1, 2 |
| NEGF | f, a | Negate f | 1 | 0110 110a ffff ffff | C, DC, Z, OV, N | |
| RLCF | f, d, a | Rotate Left f through Carry | 1 | 0011 01da ffff ffff | C, Z, N | 1, 2 |
| RLNCF | f, d, a | Rotate Left f (No Carry) | 1 | 0100 01da ffff ffff | Z, N | |
| 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 | 1, 2 |
| SUBFWB | f, d, a | Subtract f from WREG with Borrow | 1 | 0101 01da ffff ffff | C, DC, Z, OV, N | |
| SUBWF | f, d, a | Subtract WREG from f | 1 | 0101 11da ffff ffff | C, DC, Z, OV, N | 1, 2 |
| SUBWFB | f, d, a | Subtract WREG from f with Borrow | 1 | 0101 10da ffff ffff | C, DC, Z, OV, N | |
| 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 | |

**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 the 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 two-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.


© 2009 Microchip Technology Inc.

**TABLE 26-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word MSb LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|
| **BIT-ORIENTED 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, b, a | Bit Toggle f | 1 | 0111 bbba ffff ffff | None | 1, 2 |
| **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 the 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 two-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 26-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word MSb LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|
| **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 FSR(f) 1st word | 2 | 1110 1110 00ff kkkk<br>1111 0000 kkkk 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 | 0000 0000 0000 1100 | None | 5 |
| TBLWT*+ | | Table Write with Post-Increment | | 0000 0000 0000 1101 | None | 5 |
| TBLWT*- | | Table Write with Post-Decrement | | 0000 0000 0000 1110 | None | 5 |
| TBLWT+* | | Table Write with Pre-Increment | | 0000 0000 0000 1111 | None | 5 |

**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 the 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 two-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.


© 2009 Microchip Technology Inc.

### 26.1.1 STANDARD INSTRUCTION SET

**ADDLW** **ADD Literal to W**

Syntax: 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 15h`

Before Instruction
W = 10h
After Instruction
W = 25h

**ADDWF** **ADD W to f**

Syntax: 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ADDWF REG, 0, 0`

Before Instruction
W = 17h
REG = 0C2h
After Instruction
W = 0D9h
REG = 0C2h

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

**ADDWFC** **ADD W and Carry bit to f**

Syntax: 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ADDWFC REG, 0, 1`

Before Instruction
Carry bit = 1
REG = 02h
W = 4Dh
After Instruction
Carry bit = 0
REG = 02h
W = 50h

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

Syntax: 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 ANDed 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 05Fh`

Before Instruction
W = A3h
After Instruction
W = 03h


© 2009 Microchip Technology Inc.

**ANDWF** **AND W with f**

Syntax: 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ANDWF REG, 0, 0`

Before Instruction
W = 17h
REG = C2h
After Instruction
W = 02h
REG = C2h

**BC** **Branch if Carry**

Syntax: 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 two-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 5`

Before Instruction
PC = address (HERE)
After Instruction
If Carry = 1;
PC = address (HERE + 12)
If Carry = 0;
PC = address (HERE + 2)

**BCF** **Bit Clear f**

Syntax: 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

**BN** **Branch if Negative**

Syntax: 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 two-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)


© 2009 Microchip Technology Inc.

**BNC** **Branch if Not Carry**

Syntax: 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 two-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: 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 two-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)`

## BNOV Branch if Not Overflow

Syntax: 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 two-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: 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 two-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)


© 2009 Microchip Technology Inc.

**BRA** **Unconditional Branch**

Syntax: 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 two-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: 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1`

Before Instruction
FLAG_REG = 0Ah
After Instruction
FLAG_REG = 8Ah

**BTFSC** **Bit Test File, Skip if Clear**

Syntax: 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 two-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh).
See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 0
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: 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 two-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh).
See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 0
FALSE   :
TRUE    :
```
Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)


© 2009 Microchip Technology Inc.

**BTG** **Bit Toggle f**

Syntax: 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `BTG PORTC, 4, 0`

Before Instruction:
PORTC = 0111 0101 [75h]
After Instruction:
PORTC = 0110 0101 [65h]

**BOV** **Branch if Overflow**

Syntax: 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 two-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)

**BZ** **Branch if Zero**

Syntax: BZ n

Operands: -128 ≤ n ≤ 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 two-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: CALL k {,s}

Operands: 0 ≤ k ≤ 1048575
s ∈ [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. Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-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,1

Before Instruction
PC = address (HERE)
After Instruction
PC = address (THERE)
TOS = address (HERE + 4)
WS = W
BSRS = BSR
STATUSS= STATUS


© 2009 Microchip Technology Inc.

**CLRF** **Clear f**

Syntax: 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: CLRF FLAG_REG,1

Before Instruction
FLAG_REG = 5Ah
After Instruction
FLAG_REG = 00h

**CLRWDT** **Clear Watchdog Timer**

Syntax: 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 post-scaler 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 = 00h
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

**COMF** **Complement f**

Syntax: 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 '1', the result is stored in W. If 'd' is '0', the result is stored back in register 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `COMF REG, 0, 0`

Before Instruction
REG = 13h
After Instruction
REG = 13h
W = ECh

**CPFSEQ** **Compare f with W, Skip if f = W**

Syntax: 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 two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '0', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 0
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)


© 2009 Microchip Technology Inc.

## CPFSGT — Compare f with W, Skip if f > W

Syntax: 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 the 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 two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 0
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: 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 two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

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, 1
NLESS   :
LESS    :
```

Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG < W;
PC = Address (LESS)
If REG ≥ W;
PC = Address (NLESS)

## DAW — Decimal Adjust W Register

Syntax: 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>;
C = 1,
else,
(W<7:4>) → W<7:4>

Status Affected: C

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0111 |

Description: DAW adjusts the eight-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

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 = A5h
C = 0
DC = 0
After Instruction
W = 05h
C = 1
DC = 0

Example 2:

Before Instruction
W = CEh
C = 0
DC = 0
After Instruction
W = 34h
C = 1
DC = 0

## DECF — Decrement f

Syntax: 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `DECF CNT, 1, 0`

Before Instruction
CNT = 01h
Z = 0
After Instruction
CNT = 00h
Z = 1


© 2009 Microchip Technology Inc.

**DECFSZ** **Decrement f, Skip if 0**

Syntax: 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'.

If the result is '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1, 1
        GOTO    LOOP
CONTINUE
```

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT – 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE + 2)

**DCFSNZ** **Decrement f, Skip if not 0**

Syntax: 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'.

If the result is not '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1, 0
ZERO    :
NZERO   :
```

Before Instruction
TEMP = ?
After Instruction
TEMP = TEMP – 1,
If TEMP = 0;
PC = Address (ZERO)
If TEMP ≠ 0;
PC = Address (NZERO)

## GOTO Unconditional Branch

Syntax: 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 entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-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: INCF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(f) + 1 \rightarrow dest$

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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: INCF CNT, 1, 0

Before Instruction
CNT = FFh
Z = 0
C = ?
DC = ?
After Instruction
CNT = 00h
Z = 1
C = 1
DC = 1


© 2009 Microchip Technology Inc.

**INCFSZ** **Increment f, Skip if 0**

Syntax: 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'.

If the result is '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1, 0
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: 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'.

If the result is not '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1, 0
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

**IORLW** **Inclusive OR Literal with W**

Syntax: 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 ORed with the eight-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 35h

Before Instruction
W = 9Ah
After Instruction
W = BFh

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

Syntax: 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: IORWF RESULT, 0, 1

Before Instruction
RESULT = 13h
W = 91h
After Instruction
RESULT = 13h
W = 93h


© 2009 Microchip Technology Inc.

**LFSR** **Load FSR**

Syntax: 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, 3ABh`

After Instruction
FSR2H = 03h
FSR2L = ABh

**MOVF** **Move f**

Syntax: 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 '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. Location 'f' can be anywhere in the 256-byte bank.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `MOVF REG, 0, 0`

Before Instruction
REG = 22h
W = FFh
After Instruction
REG = 22h
W = 22h

**MOVFF** **Move f to f**

Syntax: 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.

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 = 33h
REG2 = 11h
After Instruction
REG1 = 33h
REG2 = 33h

**MOVLB** **Move Literal to Low Nibble in BSR**

Syntax: MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow BSR$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0001 | kkkk | kkkk |

Description: The eight-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '0', regardless of the value of $k_7$:$k_4$.

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 = 02h
After Instruction
BSR Register = 05h


© 2009 Microchip Technology Inc.

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

Syntax: MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1110 | kkkk | kkkk |

Description: The eight-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 5Ah`

After Instruction
W = 5Ah

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

Syntax: 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `MOVWF REG, 0`

Before Instruction
W = 4Fh
REG = FFh
After Instruction
W = 4Fh
REG = 4Fh

**MULLW** **Multiply Literal with W**

Syntax: 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 0C4h

Before Instruction
W = E2h
PRODH = ?
PRODL = ?
After Instruction
W = E2h
PRODH = ADh
PRODL = 08h

**MULWF** **Multiply W with f**

Syntax: 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write registers PRODH: PRODL |

Example: MULWF REG, 1

Before Instruction
W = C4h
REG = B5h
PRODH = ?
PRODL = ?
After Instruction
W = C4h
REG = B5h
PRODH = 8Ah
PRODL = 94h


© 2009 Microchip Technology Inc.

**NEGF** **Negate f**

Syntax: NEGF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(\overline{f}) + 1 \rightarrow f$

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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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 [3Ah]
After Instruction
REG = 1100 0110 [C6h]

**NOP** **No Operation**

Syntax: 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.

**POP** **Pop Top of Return Stack**

Syntax: 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 = 0031A2h
Stack (1 level down) = 014332h

After Instruction
TOS = 014332h
PC = NEW

**PUSH** **Push Top of Return Stack**

Syntax: 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 = 345Ah
PC = 0124h

After Instruction
PC = 0126h
TOS = 0126h
Stack (1 level down) = 345Ah


© 2009 Microchip Technology Inc.

**RCALL** **Relative Call**

Syntax: RCALL n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 → TOS,
(PC) + 2 + 2n → 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 two-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: RESET

Operands: None

Operation: Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.

Status Affected: All

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 1111 | 1111 |

Description: This instruction provides a way to execute a $\overline{\text{MCLR}}$ 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

**RETFIE** **Return from Interrupt**

Syntax: 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.

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: 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 eight-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 = 07h
After Instruction
W = value of kn


© 2009 Microchip Technology Inc.

**RETURN** **Return from Subroutine**

Syntax: 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 (TOS) 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.

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: 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

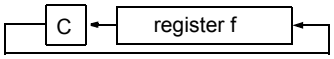
Encoding:

| | | | |
|---|---|---|---|
| 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RLCF REG, 0, 0

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 1100 1100
C = 1

**RLNCF** **Rotate Left f (No Carry)**

Syntax: 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

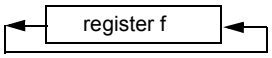
Encoding:

| | | | |
|---|---|---|---|
| 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RLNCF REG, 1, 0

Before Instruction
REG = 1010 1011
After Instruction
REG = 0101 0111

**RRCF** **Rotate Right f through Carry**

Syntax: 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

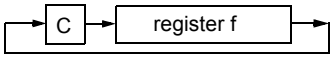
Encoding:

| | | | |
|---|---|---|---|
| 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RRCF REG, 0, 0

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 0111 0011
C = 0


© 2009 Microchip Technology Inc.

**RRNCF** **Rotate Right f (No Carry)**

Syntax: 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

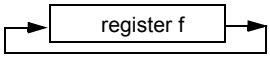
Encoding:

| | | | |
|---|---|---|---|
| 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'.

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.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.




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, 0, 0
Before Instruction
    W    =  ?
    REG  =  1101 0111
After Instruction
    W    =  1110 1011
    REG  =  1101 0111
```

**SETF** **Set f**

Syntax: 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example:
```
SETF    REG,1
Before Instruction
    REG  =  5Ah
After Instruction
    REG  =  FFh
```

**SLEEP** **Enter Sleep mode**

Syntax: 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. 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: 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: SUBFWB REG, 1, 0

Before Instruction
REG = 3
W = 2
C = 1

After Instruction
REG = FF
W = 2
C = 0
Z = 0
N = 1 ; 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


© 2009 Microchip Technology Inc.

**SUBLW** **Subtract W from Literal**

Syntax: 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 eight-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  02h
```
Before Instruction
W = 01h
C = ?
After Instruction
W = 01h
C = 1 ; result is positive
Z = 0
N = 0

Example 2:
```
SUBLW  02h
```
Before Instruction
W = 02h
C = ?
After Instruction
W = 00h
C = 1 ; result is zero
Z = 1
N = 0

Example 3:
```
SUBLW  02h
```
Before Instruction
W = 03h
C = ?
After Instruction
W = FFh; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

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

Syntax: 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1:
```
SUBWF   REG, 1, 0
```
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, 0, 0
```
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, 1, 0
```
Before Instruction
REG = 1
W = 2
C = ?
After Instruction
REG = FFh ;(2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

© 2009 Microchip Technology Inc.

**SUBWFB** **Subtract W from f with Borrow**

Syntax: 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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 = 19h (0001 1001)
W = 0Dh (0000 1101)
C = 1
After Instruction
REG = 0Ch (0000 1011)
W = 0Dh (0000 1101)
C = 1
Z = 0
N = 0 ; result is positive

Example 2: `SUBWFB REG, 0, 0`

Before Instruction
REG = 1Bh (0001 1011)
W = 1Ah (0001 1010)
C = 0
After Instruction
REG = 1Bh (0001 1011)
W = 00h
C = 1
Z = 1 ; result is zero
N = 0

Example 3: `SUBWFB REG, 1, 0`

Before Instruction
REG = 03h (0000 0011)
W = 0Eh (0000 1101)
C = 1
After Instruction
REG = F5h (1111 0100)
; [2's comp]
W = 0Eh (0000 1101)
C = 0
Z = 0
N = 1 ; result is negative

**SWAPF** **Swap f**

Syntax: 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `SWAPF REG, 1, 0`

Before Instruction
REG = 53h
After Instruction
REG = 35h


© 2009 Microchip Technology Inc.

**TBLRD Table Read**

Syntax: 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 = 55h
TBLPTR = 00A356h
MEMORY(00A356h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 00A357h

Example 2:
```
TBLRD  +* ;
```
Before Instruction
TABLAT = 0AAh
TBLPTR = 01A357h
MEMORY(01A357h) = 12h
MEMORY(01A358h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 01A358h

**TBLWT Table Write**

Syntax: 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 the TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 7.0 "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-MBtye 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

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 ) |

**TBLWT Table Write (Continued)**

Example 1: TBLWT *+;

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
HOLDING REGISTER (00A356h) = FFh
After Instructions (table write completion)
TABLAT = 55h
TBLPTR = 00A357h
HOLDING REGISTER (00A356h) = 55h

Example 2: TBLWT +*;

Before Instruction
TABLAT = 34h
TBLPTR = 01389Ah
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = FFh
After Instruction (table write completion)
TABLAT = 34h
TBLPTR = 01389Bh
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = 34h


© 2009 Microchip Technology Inc.

**TSTFSZ** **Test f, Skip if 0**

Syntax: 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 two-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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, 1
NZERO   :
ZERO    :
```
Before Instruction
PC = Address (HERE)
After Instruction
If CNT = 00h,
PC = Address (ZERO)
If CNT ≠ 00h,
PC = Address (NZERO)

**XORLW** **Exclusive OR Literal with W**

Syntax: XORLW k

Operands: $0 \le k \le 255$

Operation: (W) .XOR. $k \rightarrow W$

Status Affected: N, Z

Encoding:

| 0000 | 1010 | kkkk | kkkk |
|---|---|---|---|

Description: The contents of W are XORed 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   0AFh
```
Before Instruction
W = B5h
After Instruction
W = 1Ah

**XORWF** **Exclusive OR W with f**

Syntax: 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'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `XORWF REG, 1, 0`

Before Instruction
REG = AFh
W = B5h
After Instruction
REG = 1Ah
W = B5h


© 2009 Microchip Technology Inc.

## 26.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18F2480/2580/4480/4580 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers or use them for indexed addressing. Two of the instructions, `ADDFSR` and `SUBFSR`, each have an additional special instantiation for using FSR2. These versions (`ADDULNK` and `SUBULNK`) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and de-allocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 26-3. Detailed descriptions are provided in **Section 26.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 26-1 apply to both the standard and extended PIC18 instruction sets.

| **Note:** | The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler. |
|---|---|

### 26.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets ("[ ]"). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 26.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**.

| **Note:** | In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{ }"). |
|---|---|

**TABLE 26-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected |
|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | |
| ADDFSR | f, k | Add Literal to FSR | 1 | 1110 | 1000 | ffkk | kkkk | None |
| ADDULNK | k | Add Literal to FSR2 and Return | 2 | 1110 | 1000 | 11kk | kkkk | None |
| CALLW | | Call Subroutine using WREG | 2 | 0000 | 0000 | 0001 | 0100 | None |
| MOVSF | $z_s$, $f_d$ | Move $z_s$ (source) to 1st word | 2 | 1110 | 1011 | 0zzz | zzzz | None |
| | | $f_d$ (destination) 2nd word | | 1111 | ffff | ffff | ffff | |
| MOVSS | $z_s$, $z_d$ | Move $z_s$ (source) to 1st word | 2 | 1110 | 1011 | 1zzz | zzzz | None |
| | | $z_d$ (destination) 2nd word | | 1111 | xxxx | xzzz | zzzz | |
| PUSHL | k | Store Literal at FSR2, Decrement FSR2 | 1 | 1110 | 1010 | kkkk | kkkk | None |
| SUBFSR | f, k | Subtract Literal from FSR | 1 | 1110 | 1001 | ffkk | kkkk | None |
| SUBULNK | k | Subtract Literal from FSR2 and Return | 2 | 1110 | 1001 | 11kk | kkkk | None |

## 26.2.2 EXTENDED INSTRUCTION SET

**ADDFSR** **Add Literal to FSR**

Syntax: ADDFSR f, k

Operands: $0 \le k \le 63$
$f \in [0, 1, 2]$

Operation: FSR(f) + k → FSR(f)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 1000 | ffkk | kkkk |

Description: The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to FSR |

Example: ADDFSR 2, 23h

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

**ADDULNK** **Add Literal to FSR2 and Return**

Syntax: ADDULNK k

Operands: $0 \le k \le 63$

Operation: FSR2 + k → FSR2,
PC = (TOS)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 1000 | 11kk | kkkk |

Description: The 6-bit literal 'k' is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS.

The instruction takes two cycles to execute; a NOP is performed during the second cycle.

This may be thought of as a special case of the ADDFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to FSR |
| No Operation | No Operation | No Operation | No Operation |

Example: ADDULNK 23h

Before Instruction
FSR2 = 03FFh
PC = 0100h
TOS = 02AFh
After Instruction
FSR2 = 0422h
PC = 02AFh
TOS = TOS – 1

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).


© 2009 Microchip Technology Inc.

**CALLW** **Subroutine Call Using WREG**

Syntax: CALLW

Operands: None

Operation: (PC + 2) → TOS,
(W) → PCL,
(PCLATH) → PCH,
(PCLATU) → PCU

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 0100 |

Description: First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched into PCH and PCU, respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched.

Unlike CALL, there is no option to update W, STATUS or BSR.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read WREG | Push PC to stack | No operation |
| No operation | No operation | No operation | No operation |

Example: HERE CALLW

Before Instruction
PC = address (HERE)
PCLATH = 10h
PCLATU = 00h
W = 06h

After Instruction
PC = 001006h
TOS = address (HERE + 2)
PCLATH = 10h
PCLATU = 00h
W = 06h

**MOVSF** **Move Indexed to f**

Syntax: MOVSF $[z_s]$, $f_d$

Operands: $0 \le z_s \le 127$
$0 \le f_d \le 4095$

Operation: ((FSR2) + $z_s$) → $f_d$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (source) | 1110 | 1011 | 0zzz | $zzzz_s$ |
| 2nd word (destin.) | 1111 | ffff | ffff | $ffff_d$ |

Description: The contents of the source register are moved to destination register '$f_d$'. The actual address of the source register is determined by adding the 7-bit literal offset '$z_s$' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal '$f_d$' in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh).

The MOVSF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an indirect addressing register, the value returned will be 00h.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | No operation<br>No dummy read | No operation | Write register 'f' (dest) |

Example: MOVSF [05h], REG2

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
REG2 = 11h

After Instruction
FSR2 = 80h
Contents of 85h = 33h
REG2 = 33h

**MOVSS** **Move Indexed to Indexed**

Syntax: MOVSS [$z_s$], [$z_d$]

Operands: $0 \le z_s \le 127$
$0 \le z_d \le 127$

Operation: ((FSR2) + $z_s$) → ((FSR2) + $z_d$)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (source) | 1110 | 1011 | 1zzz | $zzzz_s$ |
| 2nd word (dest.) | 1111 | xxxx | xzzz | $zzzz_d$ |

Description: The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets '$z_s$' or '$z_d$', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).

The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | Determine dest addr | Determine dest addr | Write to dest reg |

Example: MOVSS [05h], [06h]

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 11h
After Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 33h

**PUSHL** **Store Literal at FSR2, Decrement FSR2**

Syntax: PUSHL k

Operands: $0 \le k \le 255$

Operation: k → (FSR2),
FSR2 – 1→ FSR2

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1111 | 1010 | kkkk | kkkk |

Description: The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation.

This instruction allows users to push values onto a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: PUSHL 08h

Before Instruction
FSR2H:FSR2L = 01ECh
Memory (01ECh) = 00h

After Instruction
FSR2H:FSR2L = 01EBh
Memory (01ECh) = 08h


© 2009 Microchip Technology Inc.

**SUBFSR** **Subtract Literal from FSR**

Syntax: SUBFSR f, k

Operands: $0 \le k \le 63$
$f \in [0, 1, 2]$

Operation: FSRf – k → FSRf

Status Affected: None

Encoding:

| 1110 | 1001 | ffkk | kkkk |
|---|---|---|---|

Description: The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: SUBFSR 2, 23h

Before Instruction
FSR2 = 03FFh

After Instruction
FSR2 = 03DCh

**SUBULNK** **Subtract Literal from FSR2 and Return**

Syntax: SUBULNK k

Operands: $0 \le k \le 63$

Operation: FSR2 – k → FSR2
(TOS) → PC

Status Affected: None

Encoding:

| 1110 | 1001 | 11kk | kkkk |
|---|---|---|---|

Description: The 6-bit literal 'k' is subtracted from the contents of the FSR2. A RETURN is then executed by loading the PC with the TOS.

The instruction takes two cycles to execute; a NOP is performed during the second cycle.

This may be thought of as a special case of the SUBFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |
| No Operation | No Operation | No Operation | No Operation |

Example: SUBULNK 23h

Before Instruction
FSR2 = 03FFh
PC = 0100h

After Instruction
FSR2 = 03DCh
PC = (TOS)

### 26.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

| **Note:** | Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely. |
|---|---|

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing mode (**Section 6.6.1 "Indexed Addressing with Literal Offset"**). This has a significant impact on the way that many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: either as a location in the Access Bank (a = 0), or in a GPR bank designated by the BSR (a = 1). When the extended instruction set is enabled and a = 0, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions that use the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between 'C' and assembly routines in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see **Section 26.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**).

Although the Indexed Literal Offset Addressing mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset Addressing mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

#### 26.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument, 'f', in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value, 'k'. As already noted, this occurs only when 'f' is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets ("[ ]"). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within brackets, will generate an error in the MPASM™ Assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be '0'. This is in contrast to standard operation (extended instruction set disabled) when 'a' is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM Assembler.

The destination argument, 'd', functions as before.

In the latest versions of the MPASM assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, /y, or the PE directive in the source listing.

### 26.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18F2480/2580/4480/4580, it is very important to consider the type of code. A large, re-entrant application that is written in 'C' and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.


© 2009 Microchip Technology Inc.

### ADDWF — ADD W to Indexed (Indexed Literal Offset mode)

Syntax: ADDWF [k] {,d}

Operands: $0 \le k \le 95$
$d \in [0,1]$
a = 0

Operation: (W) + ((FSR2) + k) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 01d0 | kkkk | kkkk |

Description: The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'.
If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process Data | Write to destination |

Example: ADDWF [OFST],0

Before Instruction
W = 17h
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 20h
After Instruction
W = 37h
Contents of 0A2Ch = 20h

### BSF — Bit Set Indexed (Indexed Literal Offset mode)

Syntax: BSF [k], b

Operands: $0 \le f \le 95$
$0 \le b \le 7$
a = 0

Operation: 1 → ((FSR2 + k))<b>

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1000 | bbb0 | kkkk | kkkk |

Description: Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: BSF [FLAG_OFST], 7

Before Instruction
FLAG_OFST = 0Ah
FSR2 = 0A00h
Contents of 0A0Ah = 55h
After Instruction
Contents of 0A0Ah = D5h

### SETF — Set Indexed (Indexed Literal Offset mode)

Syntax: SETF [k]

Operands: $0 \le k \le 95$

Operation: FFh → ((FSR2) + k)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 1000 | kkkk | kkkk |

Description: The contents of the register indicated by FSR2, offset by 'k', are set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process Data | Write register |

Example: SETF [OFST]

Before Instruction
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 00h
After Instruction
Contents of 0A2Ch = FFh

### 26.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18F2480/2580/4480/4580 family of devices. This includes the MPLAB C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.


© 2009 Microchip Technology Inc.

## 27.0 DEVELOPMENT SUPPORT

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

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

### 27.1 MPLAB Integrated Development Environment Software

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

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

The MPLAB IDE allows you to:

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

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

## 27.2 MPLAB C Compilers for Various Device Families

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

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

## 27.3 HI-TECH C for Various Device Families

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

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

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

## 27.4 MPASM Assembler

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

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

The MPASM Assembler features include:

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

## 27.5 MPLINK Object Linker/ MPLIB Object Librarian

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

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

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

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

 © 2009 Microchip Technology Inc.

## 27.7 MPLAB SIM Software Simulator

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

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

## 27.8 MPLAB REAL ICE In-Circuit Emulator System

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

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

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

## 27.9 MPLAB ICD 3 In-Circuit Debugger System

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

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

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

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 Integrated Development Environment (IDE). The MPLAB PICkit 3 is connected to the design engineer's PC using a full speed USB interface and can be connected to the target via an Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the reset line to implement in-circuit debugging and In-Circuit Serial Programming™.

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

## 27.11 PICkit 2 Development Programmer/Debugger and PICkit 2 Debug Express

The PICkit™ 2 Development Programmer/Debugger is a low-cost development tool with an easy to use interface for programming and debugging Microchip's Flash families of microcontrollers. The full featured Windows® programming interface supports baseline (PIC10F, PIC12F5xx, PIC16F5xx), midrange (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30, dsPIC33, and PIC32 families of 8-bit, 16-bit, and 32-bit microcontrollers, and many Microchip Serial EEPROM products. With Microchip's powerful MPLAB Integrated Development Environment (IDE) the PICkit™ 2 enables in-circuit debugging on most PIC® microcontrollers. In-Circuit-Debugging runs, halts and single steps the program while the PIC microcontroller is embedded in the application. When halted at a breakpoint, the file registers can be examined and modified.

The PICkit 2 Debug Express include the PICkit 2, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

## 27.12 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.

## 27.13 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.

 © 2009 Microchip Technology Inc.

## 28.0 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 VSS (except VDD and $\overline{MCLR}$) ..... -0.3V to (VDD + 0.3V)
Voltage on VDD with respect to VSS ..... -0.3V to +7.5V
Voltage on $\overline{MCLR}$ with respect to VSS **(Note 2)** ..... 0V to +13.25V
Total power dissipation **(Note 1)** ..... 1.0W
Maximum current out of VSS pin ..... 300 mA
Maximum current into VDD pin ..... 250 mA
Input clamp current, IIK (VI < 0 or VI > VDD) ..... ±20 mA
Output clamp current, IOK (VO < 0 or VO > VDD) ..... ±20 mA
Maximum output current sunk by any I/O pin ..... 25 mA
Maximum output current sourced by any I/O pin ..... 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 = VDD x {IDD – ∑ IOH} + ∑ {(VDD – VOH) x IOH} + ∑(VOL x IOL)

**2:** Voltage spikes below VSS at the $\overline{MCLR}$/VPP 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 $\overline{MCLR}$/VPP/RE3 pin, rather than pulling this pin directly to VSS.

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

**FIGURE 28-1: PIC18F2480/2580/4480/4580 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL, EXTENDED)**

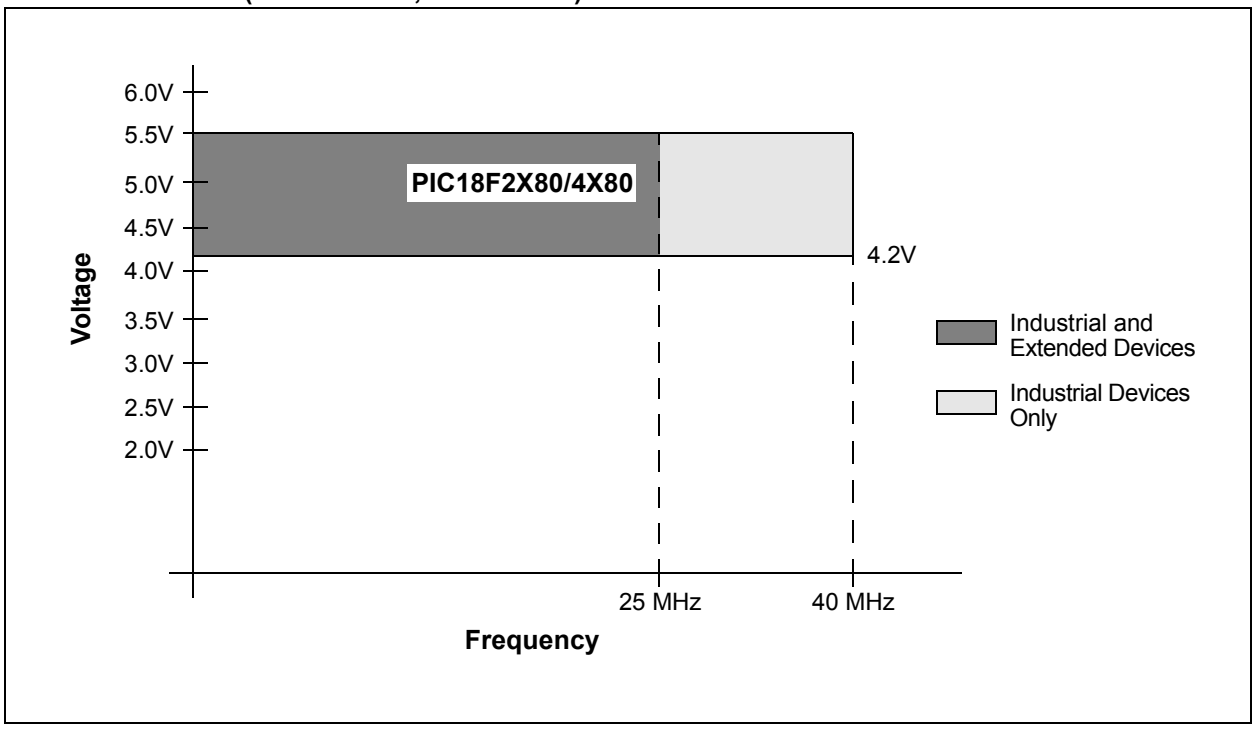
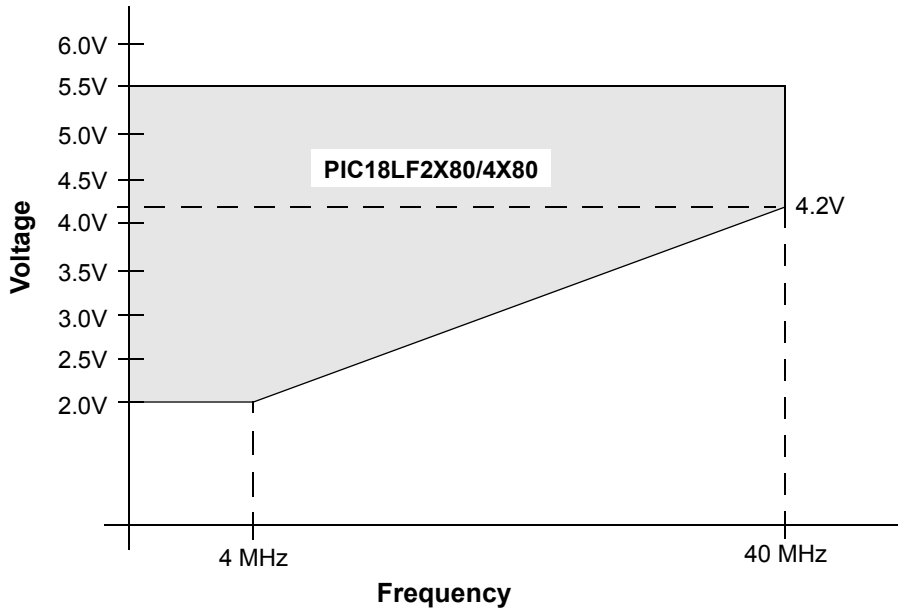



**FIGURE 28-2: PIC18LF2480/2580/4480/4580 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)**




$F_{MAX}$ = (16.36 MHz/V) ($V_{DDAPPMIN}$ – 2.0V) + 4 MHz

**Note:** $V_{DDAPPMIN}$ is the minimum voltage of the PIC® device in the application.


© 2009 Microchip Technology Inc.

## 28.1 DC Characteristics: Supply Voltage PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF2480/2580/4480/4580** (Industrial) | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
| **PIC18F2480/2580/4480/4580** (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** |
| | VDD | **Supply Voltage** | | | | | |
| D001 | | PIC18LF2X80/4X80 | 2.0 | — | 5.5 | V | |
| | | PIC18F2X80/4X80 | 4.2 | — | 5.5 | V | |
| D001C | AVDD | **Analog Supply Voltage** | VDD – 0.3 | — | VDD + 0.3 | V | |
| D001D | AVSS | **Analog Ground Voltage** | VSS – 0.3 | — | VSS + 0.3 | 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 on Power-on Reset for details |
| D004 | SVDD | **VDD Rise Rate** to ensure Internal Power-on Reset Signal | 0.05 | — | — | V/ms | See section on Power-on Reset for details |
| | VBOR | **Brown-out Reset Voltage** | | | | | |
| D005 | | PIC18LF2X80/4X80 | | | | | |
| | | BORV<1:0> = 11 | 2.00 | 2.1 | 2.16 | V | |
| | | BORV<1:0> = 10 | 2.65 | 2.79 | 2.93 | V | |
| D005 | | All Devices | | | | | |
| | | BORV<1:0> = 01(2) | 4.11 | 4.33 | 4.55 | V | |
| | | BORV<1:0> = 00 | 4.36 | 4.59 | 4.82 | V | |

**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:** With BOR enabled, full-speed operation (FOSC = 40 MHz) is supported until a BOR occurs. This is valid although VDD may be below the minimum voltage for this frequency.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial)

| | | | | | | |
|---|---|---|---|---|---|---|
| **PIC18LF2480/2580/4480/4580** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
| **PIC18F2480/2580/4480/4580** (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)** | | | | | |
| | PIC18LF2X80/4X80 | 0.2 | 1.0 | µA | -40°C | VDD = 2.0V (**Sleep** mode) |
| | | 0.2 | 1.0 | µA | +25°C | |
| | | 0.3 | 4.0 | µA | +60°C | |
| | | 0.4 | 6.0 | µA | +85°C | |
| | PIC18LF2X80/4X80 | 0.2 | 1.5 | µA | -40°C | VDD = 3.0V (**Sleep** mode) |
| | | 0.2 | 2.0 | µA | +25°C | |
| | | 0.4 | 5.0 | µA | +60°C | |
| | | 0.5 | 8.0 | µA | +85°C | |
| | All devices | 0.2 | 2.0 | µA | -40°C | VDD = 5.0V (**Sleep** mode) |
| | | 0.2 | 2.0 | µA | +25°C | |
| | | 0.6 | 9.0 | µA | +60°C | |
| | | 1.0 | 15 | µA | +85°C | |
| | Extended devices only | 52.00 | 132.00 | 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.


© 2009 Microchip Technology Inc.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF2480/2580/4480/4580** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F2480/2580/4480/4580** (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.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF2X80/4X80 | 19 | 31 | µA | -40°C | VDD = 2.0V | FOSC = 31 kHz (**RC_RUN** mode, Internal oscillator source) |
| | | 21 | 31 | µA | +25°C | | |
| | | 22 | 31 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 57 | 60 | µA | -40°C | VDD = 3.0V | |
| | | 47 | 60 | µA | +25°C | | |
| | | 42 | 60 | µA | +85°C | | |
| | All devices | 150 | 170 | µA | -40°C | VDD = 5.0V | |
| | | 113 | 170 | µA | +25°C | | |
| | | 98 | 170 | µA | +85°C | | |
| | Extended devices only | 170 | 280 | µA | +125°C | | |
| | PIC18LF2X80/4X80 | 530 | 1030 | µA | -40°C | VDD = 2.0V | FOSC = 1 MHz (**RC_RUN** mode, Internal oscillator source) |
| | | 550 | 1030 | µA | +25°C | | |
| | | 560 | 1030 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 940 | 1150 | µA | -40°C | VDD = 3.0V | |
| | | 900 | 1150 | µA | +25°C | | |
| | | 880.0 | 1150 | µA | +85°C | | |
| | All devices | 1.8 | 2.3 | mA | -40°C | VDD = 5.0V | |
| | | 1.7 | 2.3 | mA | +25°C | | |
| | | 1.7 | 2.3 | mA | +85°C | | |
| | Extended devices only | 2.6 | 3.6 | 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;
$\overline{\text{MCLR}}$ = 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.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial) (Continued)

| PIC18LF2480/2580/4480/4580 (Industrial) | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial |
|---|---|
| PIC18F2480/2580/4480/4580 (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)** | | | | | | |
| | PIC18LF2X80/4X80 | 1.5 | 2.1 | mA | -40°C | VDD = 2.0V | FOSC = 4 MHz (**RC_RUN** mode, Internal oscillator source) |
| | | 1.5 | 2.1 | mA | +25°C | | |
| | | 1.5 | 2.1 | mA | +85°C | | |
| | PIC18LF2X80/4X80 | 2.4 | 3.3 | mA | -40°C | VDD = 3.0V | |
| | | 2.4 | 3.3 | mA | +25°C | | |
| | | 2.4 | 3.3 | mA | +85°C | | |
| | All devices | 4.4 | 5.3 | mA | -40°C | VDD = 5.0V | |
| | | 4.4 | 5.3 | mA | +25°C | | |
| | | 4.4 | 5.3 | mA | +85°C | | |
| | Extended devices only | 9.2 | 11 | mA | +125°C | | |
| | PIC18LF2X80/4X80 | 6.1 | 8.4 | µA | -40°C | VDD = 2.0V | FOSC = 31 kHz (**RC_IDLE** mode, Internal oscillator source) |
| | | 6.7 | 8.4 | µA | +25°C | | |
| | | 7.4 | 21 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 9.6 | 12 | µA | -40°C | VDD = 3.0V | |
| | | 11 | 12 | µA | +25°C | | |
| | | 12 | 33 | µA | +85°C | | |
| | All devices | 20 | 28 | µA | -40°C | VDD = 5.0V | |
| | | 22 | 28 | µA | +25°C | | |
| | | 24 | 55 | µA | +85°C | | |
| | Extended devices only | 84 | 200 | 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;
$\overline{\text{MCLR}}$ = 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.


© 2009 Microchip Technology Inc.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF2480/2580/4480/4580** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F2480/2580/4480/4580** (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.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF2X80/4X80 | 300 | 390 | µA | -40°C | VDD = 2.0V | FOSC = 1 MHz (**RC_IDLE** mode, Internal oscillator source) |
| | | 320 | 390 | µA | +25°C | | |
| | | 330 | 390 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 450 | 550 | µA | -40°C | VDD = 3.0V | |
| | | 470 | 550 | µA | +25°C | | |
| | | 490 | 550 | µA | +85°C | | |
| | All devices | 840 | 1030 | µA | -40°C | VDD = 5.0V | |
| | | 880 | 1030 | µA | +25°C | | |
| | | 900 | 1030 | µA | +85°C | | |
| | Extended devices only | 2.8 | 3.2 | mA | +125°C | | |
| | PIC18LF2X80/4X80 | 760 | 1050 | µA | -40°C | VDD = 2.0V | FOSC = 4 MHz (**RC_IDLE** mode, Internal oscillator source) |
| | | 790 | 1050 | µA | +25°C | | |
| | | 810 | 1050 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 1.2 | 1.5 | mA | -40°C | VDD = 3.0V | |
| | | 1.2 | 1.5 | mA | +25°C | | |
| | | 1.3 | 1.5 | mA | +85°C | | |
| | All devices | 2.2 | 2.7 | mA | -40°C | VDD = 5.0V | |
| | | 2.3 | 2.7 | mA | +25°C | | |
| | | 2.3 | 2.7 | mA | +85°C | | |
| | Extended devices only | 4.7 | 5.5 | mA | +125°C | | |
| | PIC18LF2X80/4X80 | 410 | 550 | µA | -40°C | VDD = 2.0V | FOSC = 1 MHz (**PRI_RUN**, EC oscillator) |
| | | 420 | 550 | µA | +25°C | | |
| | | 420 | 550 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 870 | 830 | µA | -40°C | VDD = 3.0V | |
| | | 770 | 830 | µA | +25°C | | |
| | | 720 | 830 | µA | +85°C | | |
| | All devices | 1.8 | 3.3 | mA | -40°C | VDD = 5.0V | |
| | | 1.6 | 3.3 | mA | +25°C | | |
| | | 1.5 | 3.3 | mA | +85°C | | |
| | Extended devices only | 1.5 | 3.3 | 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;
$\overline{MCLR}$ = 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.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF2480/2580/4480/4580** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F2480/2580/4480/4580** (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)** | | | | | | |
| | PIC18LF2X80/4X80 | 1.4 | 2.2 | mA | -40°C | VDD = 2.0V | FOSC = 4 MHz (**PRI_RUN**, EC oscillator) |
| | | 1.4 | 2.2 | mA | +25°C | | |
| | | 1.4 | 2.2 | mA | +85°C | | |
| | PIC18LF2X80/4X80 | 2.3 | 3.3 | mA | -40°C | VDD = 3.0V | |
| | | 2.3 | 3.3 | mA | +25°C | | |
| | | 2.3 | 3.3 | mA | +85°C | | |
| | All devices | 4.5 | 6.6 | mA | -40°C | VDD = 5.0V | |
| | | 4.3 | 6.6 | mA | +25°C | | |
| | | 4.3 | 6.6 | mA | +85°C | | |
| | Extended devices only | 5 | 7.7 | mA | +125°C | | |
| | | 15 | 23 | mA | +125°C | VDD = 4.2V | FOSC = 25 MHz (**PRI_RUN**, EC oscillator) |
| | | 20 | 31 | mA | +125°C | VDD = 5.0V | |
| | All devices | 30 | 38 | mA | -40°C | VDD = 4.2V | FOSC = 40 MHz (**PRI_RUN**, EC oscillator) |
| | | 31 | 38 | mA | +25°C | | |
| | | 31 | 38 | mA | +85°C | | |
| | All devices | 37 | 44 | mA | -40°C | VDD = 5.0V | |
| | | 38 | 44 | mA | +25°C | | |
| | | 39 | 44 | 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.


© 2009 Microchip Technology Inc.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF2480/2580/4480/4580** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F2480/2580/4480/4580** (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.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF2X80/4X80 | 160 | 220 | µA | -40°C | VDD = 2.0V | FOSC = 1 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 170 | 220 | µA | +25°C | | |
| | | 170 | 220 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 250 | 330 | µA | -40°C | VDD = 3.0V | |
| | | 260 | 330 | µA | +25°C | | |
| | | 260 | 330 | µA | +85°C | | |
| | All devices | 460 | 550 | µA | -40°C | VDD = 5.0V | |
| | | 470 | 550 | µA | +25°C | | |
| | | 480 | 550 | µA | +85°C | | |
| | Extended devices only | 790 | 920 | µA | +125°C | | |
| | PIC18LF2X80/4X80 | 640 | 715 | µA | -40°C | VDD = 2.0V | FOSC = 4 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 650 | 715 | µA | +25°C | | |
| | | 660 | 715 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 0.98 | 1.4 | mA | -40°C | VDD = 3.0V | |
| | | 1 | 1.4 | mA | +25°C | | |
| | | 1.1 | 1.4 | mA | +85°C | | |
| | All devices | 1.9 | 2.2 | mA | -40°C | VDD = 5.0V | |
| | | 1.9 | 2.2 | mA | +25°C | | |
| | | 1.9 | 2.2 | mA | +85°C | | |
| | Extended devices only | 2.1 | 2.4 | mA | +125°C | | |
| | | 9.5 | 11 | mA | +125°C | VDD = 4.2V | FOSC = 25 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 14 | 16 | mA | +125°C | VDD = 5.0V | |
| | All devices | 15 | 18 | mA | -40°C | VDD = 4.2 V | FOSC = 40 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 16 | 18 | mA | +25°C | | |
| | | 16 | 18 | mA | +85°C | | |
| | All devices | 19 | 22 | mA | -40°C | VDD = 5.0V | |
| | | 19 | 22 | mA | +25°C | | |
| | | 20 | 22 | 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;
$\overline{\text{MCLR}}$ = 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.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF2480/2580/4480/4580** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F2480/2580/4480/4580** (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)** | | | | | | |
| | PIC18LF2X80/4X80 | 19 | 44 | µA | -40°C | VDD = 2.0V | FOSC = 32 kHz (**SEC_RUN** mode, Timer1 as clock)(4) |
| | | 20 | 44 | µA | +25°C | | |
| | | 22 | 44 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 56 | 71 | µA | -40°C | VDD = 3.0V | |
| | | 45 | 71 | µA | +25°C | | |
| | | 41 | 71 | µA | +85°C | | |
| | All devices | 140 | 165 | µA | -40°C | VDD = 5.0V | |
| | | 106 | 165 | µA | +25°C | | |
| | | 95 | 165 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 6.1 | 13 | µA | -40°C | VDD = 2.0V | FOSC = 32 kHz (**SEC_IDLE** mode, Timer1 as clock)(4) |
| | | 6.6 | 13 | µA | +25°C | | |
| | | 7.7 | 13 | µA | +85°C | | |
| | PIC18LF2X80/4X80 | 9.3 | 33 | µA | -40°C | VDD = 3.0V | |
| | | 9.4 | 33 | µA | +25°C | | |
| | | 11 | 33 | µA | +85°C | | |
| | All devices | 17 | 50 | µA | -40°C | VDD = 5.0V | |
| | | 17 | 50 | µA | +25°C | | |
| | | 20 | 50 | µA | +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;
$\overline{\text{MCLR}}$ = 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.


© 2009 Microchip Technology Inc.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial) (Continued)

| PIC18LF2480/2580/4480/4580 (Industrial) | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial |
|---|---|
| **PIC18F2480/2580/4480/4580 (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 | | |
|---|---|---|---|---|---|---|---|
| | **Module Differential Currents (ΔIWDT, ΔIBOR, ΔILVD, ΔIOSCB, ΔIAD)** | | | | | | |
| D022 (ΔIWDT) | **Watchdog Timer** | 1.7 | 7.6 | µA | -40°C | VDD = 2.0V | |
| | | 2.1 | 8 | µA | +25°C | | |
| | | 2.6 | 8.4 | µA | +85°C | | |
| | | 2.2 | 11.4 | µA | -40°C | VDD = 3.0V | |
| | | 2.4 | 12 | µA | +25°C | | |
| | | 2.8 | 12.6 | µA | +85°C | | |
| | | 2.9 | 14.3 | µA | -40°C | VDD = 5.0V | |
| | | 3.1 | 15 | µA | +25°C | | |
| | | 3.3 | 15.8 | µA | +85°C | | |
| | | 7.80 | 19 | µA | +125°C | | |
| D022A (ΔIBOR) | **Brown-out Reset** | 17 | 75 | µA | -40°C to +85°C | VDD = 3.0V | |
| | | 47 | 92 | µA | -40°C to +85°C | VDD = 5.0V | |
| | | 30 | 58 | µA | +125°C | | |
| | | 0 | 2 | µA | -40°C to +85°C | | Sleep mode |
| | | 0 | 5 | µA | -40°C to +125°C | | BOREN<1:0> |
| D022B (ΔILVD) | **High/Low-Voltage Detect** | 14 | 47 | µA | -40°C to +85°C | VDD = 2.0V | |
| | | 18 | 58 | µA | -40°C to +85°C | VDD = 3.0V | |
| | | 21 | 69 | µA | -40°C to +85°C | VDD = 5.0V | |
| | | 19 | 50 | µA | +125°C | | |
| D025 (ΔIOSCB) | **Timer1 Oscillator** | 1.0 | 8 | µA | -40°C | VDD = 2.0V | 32 kHz on Timer1(4) |
| | | 1.1 | 8 | µA | +25°C | | |
| | | 1.1 | 8 | µA | +85°C | | |
| | | 1.2 | 8.2 | µA | -40°C | VDD = 3.0V | 32 kHz on Timer1(4) |
| | | 1.3 | 8.2 | µA | +25°C | | |
| | | 1.2 | 8.2 | µA | +85°C | | |
| | | 1.8 | 10 | µA | -40°C | VDD = 5.0V | 32 kHz on Timer1(4) |
| | | 1.9 | 10 | µA | +25°C | | |
| | | 1.9 | 10 | µA | +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.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F2480/2580/4480/4580 (Industrial, Extended) PIC18LF2480/2580/4480/4580 (Industrial) (Continued)

| PIC18LF2480/2580/4480/4580 (Industrial) | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial |
|---|---|
| **PIC18F2480/2580/4480/4580** (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 | | |
|---|---|---|---|---|---|---|---|
| D026 (ΔIAD) | **A/D Converter** | 1.0 | 2.0 | μA | -40°C to +85°C | VDD = 2.0V | A/D on, not converting |
| | | 1.0 | 2.0 | μA | -40°C to +85°C | VDD = 3.0V | |
| | | 1.0 | 2.0 | μA | -40°C to +85°C | VDD = 5.0V | |
| | | 2.0 | 8.0 | μA | -40°C to +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.


© 2009 Microchip Technology Inc.

## 28.3 DC Characteristics: PIC18F2480/2580/4480/4580 (Industrial) PIC18LF2480/2580/4480/4580 (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 | |
| D031A | | RC3 and RC4 | VSS | 0.3 Vdd | V | I²C™ enabled |
| D031B | | | VSS | 0.8 | V | SMBus enabled |
| D032 | | $\overline{MCLR}$ | VSS | 0.2 VDD | V | |
| D033 | | OSC1 | VSS | 0.3 VDD | V | HS, HSPLL modes |
| D033A | | OSC1 | VSS | 0.2 VDD | V | RC, EC modes(1) |
| D033B | | OSC1 | VSS | 0.3 | V | XT, LP modes |
| D034 | | T13CKI | VSS | 0.3 | 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 | |
| D041A | | RC3 and RC4 | 0.7 VDD | VDD | V | I²C™ enabled |
| D041B | | | 2.1 | VDD | V | SMBus enabled, VDD ≥ 3V |
| D042 | | $\overline{MCLR}$ | 0.8 VDD | VDD | V | |
| D043 | | OSC1 | 0.7 VDD | VDD | V | HS, HSPLL modes |
| D043A | | OSC1 | 0.8 VDD | VDD | V | EC mode |
| D043B | | OSC1 | 0.9 VDD | VDD | V | RC mode(1) |
| D043C | | OSC1 | 1.6 | VDD | V | XT, LP modes |
| D044 | | T13CKI | 1.6 | VDD | V | |
| | IIL | **Input Leakage Current(2,3)** | | | | |
| D060 | | I/O Ports | — | ±200 | nA | VDD < 5.5V,<br>VSS ≤ VPIN ≤ VDD,<br>Pin at high-impedance |
| | | | — | ±50 | nA | VDD < 3V,<br>VSS ≤ VPIN ≤ VDD,<br>Pin at high-impedance |
| D061 | | $\overline{MCLR}$ | — | ±1 | µA | Vss ≤ VPIN ≤ VDD |
| D063 | | OSC1 | — | ±1 | 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.

## 28.3 DC Characteristics: PIC18F2480/2580/4480/4580 (Industrial) PIC18LF2480/2580/4480/4580 (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, RCIO, EC, ECIO modes) | — | 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, RCIO, EC, ECIO modes) | VDD – 0.7 | — | V | IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C |
| | | **Capacitive Loading Specs on Output Pins** | | | | |
| D100 | 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 | $I^2C$™ Specification |

**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.


© 2009 Microchip Technology Inc.

**TABLE 28-1: MEMORY PROGRAMMING REQUIREMENTS**

| 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.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| | | **Internal Program Memory Programming Specifications(1)** | | | | | |
| D110 | VPP | Voltage on $\overline{\text{MCLR}}$/VPP/RE3 Pin | 9.00 | — | 13.25 | V | **(Note 3)** |
| 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(2) | 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 | 100 | — | 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:** Refer to **Section 8.8 “Using the Data EEPROM”** for a more detailed discussion on data EEPROM endurance.

**3:** Required only if Single-Supply Programming is disabled.

**TABLE 28-2: COMPARATOR SPECIFICATIONS**

| **Operating Conditions:** 3.0V < VDD < 5.5V, -40°C < TA < +85°C (unless otherwise stated)<br>-40°C < TA < +125°C for extended | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| D300 | VIOFF | Input Offset Voltage | — | ±5.0 | ±10 | mV | |
| D301 | VICM | Input Common Mode Voltage | 0 | — | VDD – 1.5 | V | |
| D302 | CMRR | Common Mode Rejection Ratio | 55 | — | — | dB | |
| D303 | TRESP | Response Time(1)* | — | 150 | 400 | ns | PIC18**F**XXXX |
| D303A | | | — | 150 | 600 | ns | PIC18L**F**XXXX, VDD = 2.0V |
| D304 | TMC2OV | Comparator Mode Change to Output Valid* | — | — | 10 | μs | |

**Note 1:** Response time measured with one comparator input at (VDD – 1.5)/2 while the other input transitions from VSS to VDD.

**TABLE 28-3: VOLTAGE REFERENCE SPECIFICATIONS**

| **Operating Conditions:** 3.0V < VDD < 5.5V, -40°C < TA < +85°C (unless otherwise stated)<br>-40°C < TA < +125°C for extended | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| D310 | VRES | Resolution | VDD/24 | — | VDD/32 | LSb | |
| D311 | VRAA | Absolute Accuracy | —<br>— | —<br>— | 1/4<br>1/2 | LSb<br>LSb | Low Range (CVRR = 1)<br>High Range (CVRR = 0) |
| D312 | VRUR | Unit Resistor Value (R) | — | 2k | — | Ω | |
| D313 | TSET | Settling Time(1) | — | — | 10 | μs | |

**Note 1:** Settling time measured while CVRR = 1 and CVR<3:0> bits transition from '0000' to '1111'.


© 2009 Microchip Technology Inc.

**FIGURE 28-3: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS**

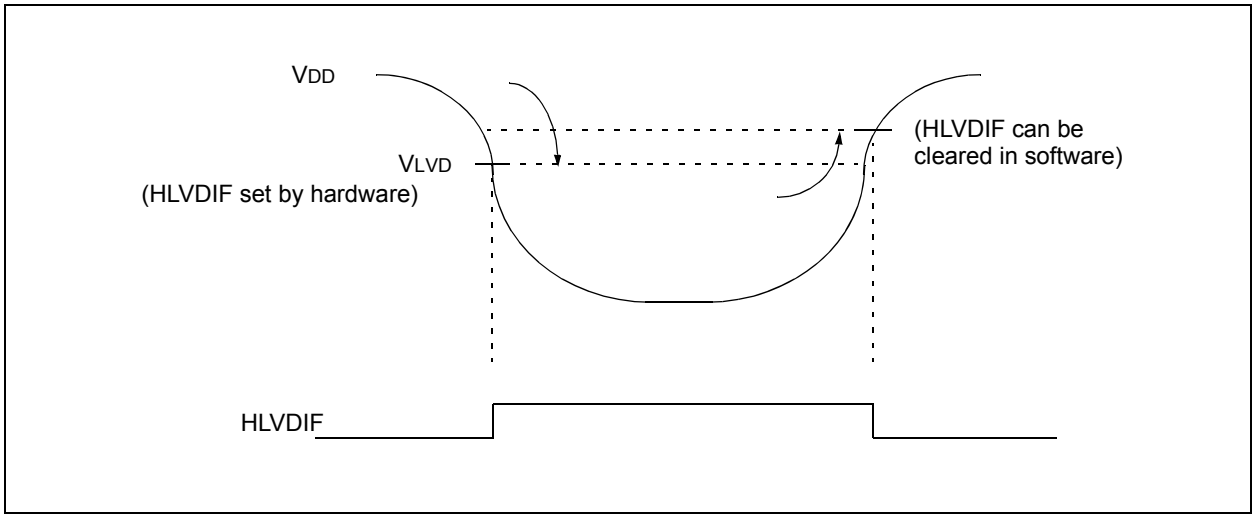



**TABLE 28-4: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS**

| | | | | **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** | **Typ** | **Max** | **Units** | **Conditions** |
| D420 | | HLVD Voltage on VDD Transition High-to-Low | LVV = 0000 | 2.12 | 2.17 | 2.22 | V | |
| | | | LVV = 0001 | 2.18 | 2.23 | 2.28 | V | |
| | | | LVV = 0010 | 2.31 | 2.36 | 2.42 | V | |
| | | | LVV = 0011 | 2.38 | 2.44 | 2.49 | V | |
| | | | LVV = 0100 | 2.54 | 2.60 | 2.66 | V | |
| | | | LVV = 0101 | 2.72 | 2.79 | 2.85 | V | |
| | | | LVV = 0110 | 2.82 | 2.89 | 2.95 | V | |
| | | | LVV = 0111 | 3.05 | 3.12 | 3.19 | V | |
| | | | LVV = 1000 | 3.31 | 3.39 | 3.47 | V | |
| | | | LVV = 1001 | 3.46 | 3.55 | 3.63 | V | |
| | | | LVV = 1010 | 3.63 | 3.71 | 3.80 | V | |
| | | | LVV = 1011 | 3.81 | 3.90 | 3.99 | V | |
| | | | LVV = 1100 | 4.01 | 4.11 | 4.20 | V | |
| | | | LVV = 1101 | 4.23 | 4.33 | 4.43 | V | |
| | | | LVV = 1110 | 4.48 | 4.59 | 4.69 | V | |
| | | | LVV = 1111 | 1.14 | 1.2 | 1.26 | V | |

## 28.4 AC (Timing) Characteristics

### 28.4.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created using 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 | | |


© 2009 Microchip Technology Inc.

### 28.4.2 TIMING CONDITIONS

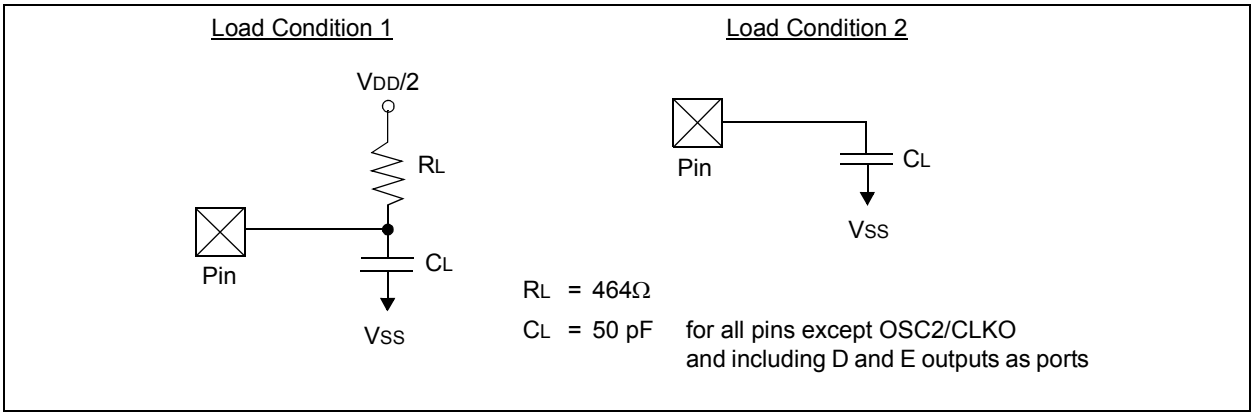
The temperature and voltages specified in Table 28-5 apply to all timing specifications unless otherwise noted. Figure 28-4 specifies the load conditions for the timing specifications.

**Note:** Because of space limitations, the generic terms "PIC18FXXXX" and "PIC18LFXXXX" are used throughout this section to refer to the PIC18F2480/2580/4480/4580 and PIC18LF2480/2580/4480/4580 families of devices specifically and only those devices.

**TABLE 28-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

| | |
|---|---|
| **AC CHARACTERISTICS** | **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<br>Operating voltage $V_{DD}$ range as described in DC spec **Section 28.1** and **Section 28.3**. LF parts operate for industrial temperatures only. |

**FIGURE 28-4: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**



28.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 28-5: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)**

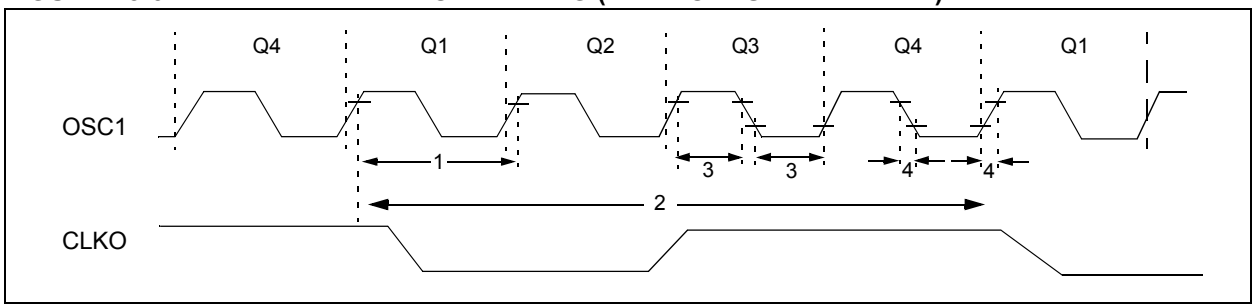



**TABLE 28-6: EXTERNAL CLOCK TIMING REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 1A | FOSC | External CLKI Frequency(1) | DC | 1 | MHz | XT, RC Oscillator mode |
| | | | DC | 25 | MHz | HS Oscillator mode |
| | | | DC | 31.25 | kHz | LP Oscillator mode |
| | | | DC | 40 | MHz | EC Oscillator mode |
| | | Oscillator Frequency(1) | DC | 4 | MHz | RC Oscillator mode |
| | | | 0.1 | 4 | MHz | XT Oscillator mode |
| | | | 4 | 25 | MHz | HS Oscillator mode |
| | | | 4 | 10 | MHz | HSPLL Oscillator mode |
| | | | 5 | 200 | kHz | LP Oscillator mode |
| 1 | TOSC | External CLKI Period(1) | 1000 | — | ns | XT, RC Oscillator mode |
| | | | 40 | — | ns | HS Oscillator mode |
| | | | 32 | — | μs | LP Oscillator mode |
| | | | 25 | — | ns | EC Oscillator mode |
| | | Oscillator Period(1) | 250 | — | ns | RC Oscillator mode |
| | | | 250 | 1 | μs | XT Oscillator mode |
| | | | 40 | 250 | ns | HS Oscillator mode |
| | | | 100 | 250 | ns | HSPLL Oscillator mode |
| | | | 5 | 200 | μs | LP Oscillator mode |
| 2 | TCY | Instruction Cycle Time(1) | 100 | — | ns | TCY = 4/FOSC, Industrial |
| | | | 160 | — | ns | TCY = 4/FOSC, Extended |
| 3 | TOSL, TOSH | External Clock in (OSC1) High or Low Time | 30 | — | ns | XT Oscillator mode |
| | | | 2.5 | — | μs | LP Oscillator mode |
| | | | 10 | — | ns | HS Oscillator mode |
| 4 | TOSR, TOSF | External Clock in (OSC1) Rise or Fall Time | — | 20 | ns | XT Oscillator mode |
| | | | — | 50 | ns | LP Oscillator mode |
| | | | — | 7.5 | ns | HS Oscillator mode |

**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.


© 2009 Microchip Technology Inc.

**TABLE 28-7: PLL CLOCK TIMING SPECIFICATIONS ($V_{DD}$ = 4.2V TO 5.5V)**

| Param No. | Sym | Characteristic | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| F10 | FOSC | Oscillator Frequency Range | 4 | — | 10 | MHz | HS mode only |
| F11 | FSYS | On-Chip VCO System Frequency | 16 | — | 40 | MHz | HS mode only |
| F12 | $t_{rc}$ | PLL Start-up Time (lock time) | — | — | 2 | ms | |
| F13 | ΔCLK | CLKO Stability (jitter) | -2 | — | +2 | % | |

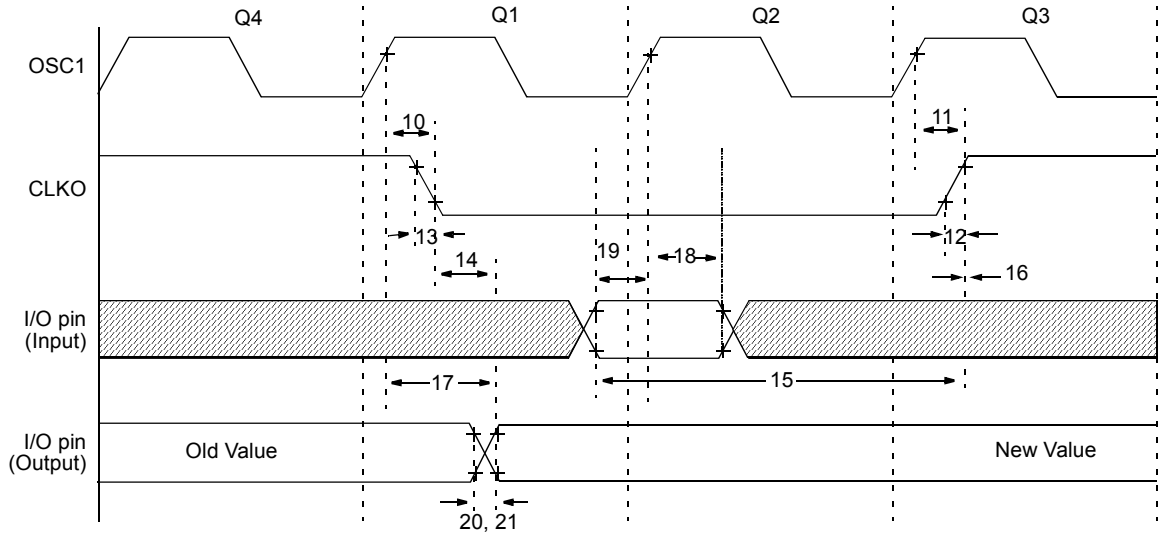
† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**TABLE 28-8: AC CHARACTERISTICS: INTERNAL RC ACCURACY**
**PIC18F2480/2580/4480/4580 (INDUSTRIAL)**
**PIC18LF2480/2580/4480/4580 (INDUSTRIAL)**

| PIC18F2480/2580/4480/4580 (Industrial) | | 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** | **Min** | **Typ** | **Max** | **Units** | **Conditions** | |
| | **INTOSC Accuracy @ Freq = 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz, 125 kHz(1)** | | | | | | |
| | PIC18LF2X80/4X80 | -2 | +/-1 | 2 | % | +25°C | VDD = 2.7-3.3V |
| | | -5 | — | 5 | % | -10°C to +85°C | VDD = 2.7-3.3V |
| | | -10 | +/-1 | 10 | % | -40°C to +85°C | VDD = 2.7-3.3V |
| | PIC18F2X80/4X80 | -2 | +/-1 | 2 | % | +25°C | VDD = 4.5-5.5V |
| | | -5 | — | 5 | % | -10°C to +85°C | VDD = 4.5-5.5V |
| | | -10 | +/-1 | 10 | % | -40°C to +85°C | VDD = 4.5-5.5V |
| | **INTRC Accuracy @ Freq = 31 kHz** | | | | | | |
| | PIC18LF2X80/4X80 | 26.562 | — | 35.938 | kHz | -40°C to +85°C | VDD = 2.7-3.3V |
| | PIC18F2X80/4X80 | 26.562 | — | 35.938 | kHz | -40°C to +85°C | VDD = 4.5-5.5V |

**Note 1:** Frequency calibrated at 25°C. OSCTUNE register can be used to compensate for temperature drift.

**FIGURE 28-6: CLKO AND I/O TIMING**




**Note:** Refer to Figure 28-4 for load conditions.

**TABLE 28-9: 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) | PIC18**F**XXXX | 100 | — | — | ns | |
| 18A | | | PIC18**LF**XXXX | 200 | — | — | ns | VDD = 2.0V |
| 19 | TioV2osH | Port Input Valid to OSC1 ↑ (I/O in setup time) | | 0 | — | — | ns | |
| 20 | TioR | Port Output Rise Time | PIC18**F**XXXX | — | 10 | 25 | ns | |
| 20A | | | PIC18**LF**XXXX | — | — | 60 | ns | VDD = 2.0V |
| 21 | TioF | Port Output Fall Time | PIC18**F**XXXX | — | 10 | 25 | ns | |
| 21A | | | PIC18**LF**XXXX | — | — | 60 | ns | VDD = 2.0V |
| 22† | Tinp | INTx Pin High or Low Time | | TCY | — | — | ns | |
| 23† | Trbp | RB<7:4> Change INTx High or Low Time | | TCY | — | — | ns | |
| 24† | Trcp | RC<7:4> Change INTx High or Low Time | | 20 | | | ns | |

† These parameters are asynchronous events not related to any internal clock edges.

**Note 1:** Measurements are taken in RC mode, where CLKO output is 4 x TOSC.


© 2009 Microchip Technology Inc.

**FIGURE 28-7: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**

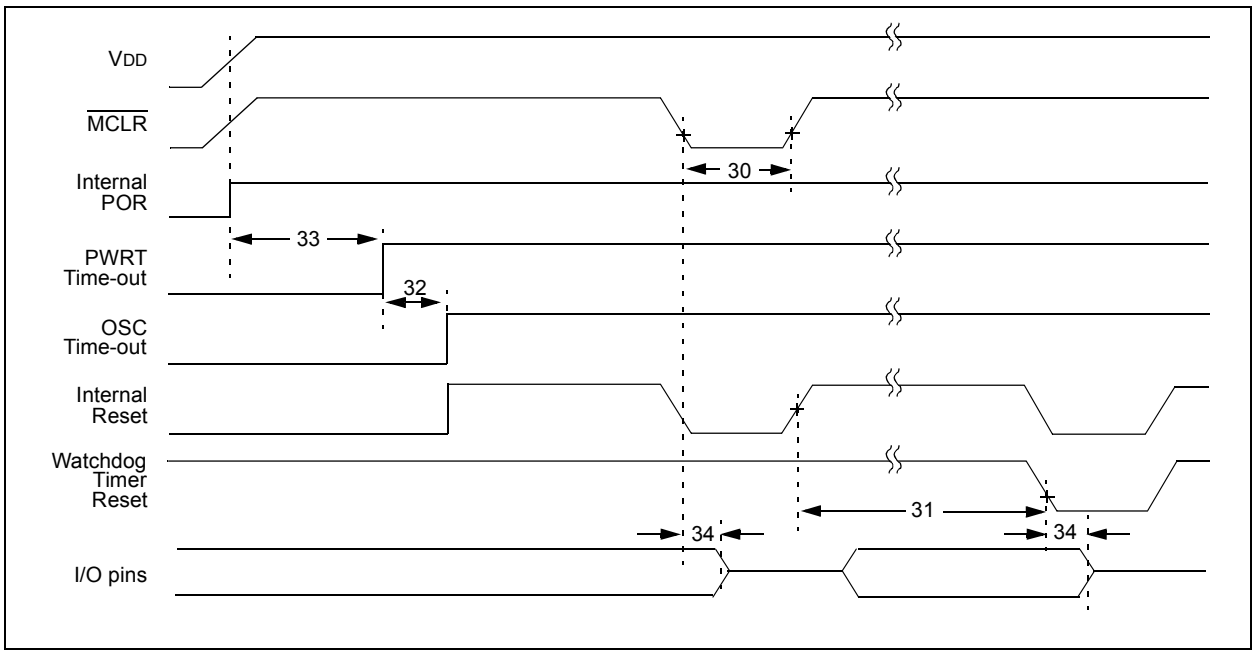



**FIGURE 28-8: BROWN-OUT RESET TIMING**

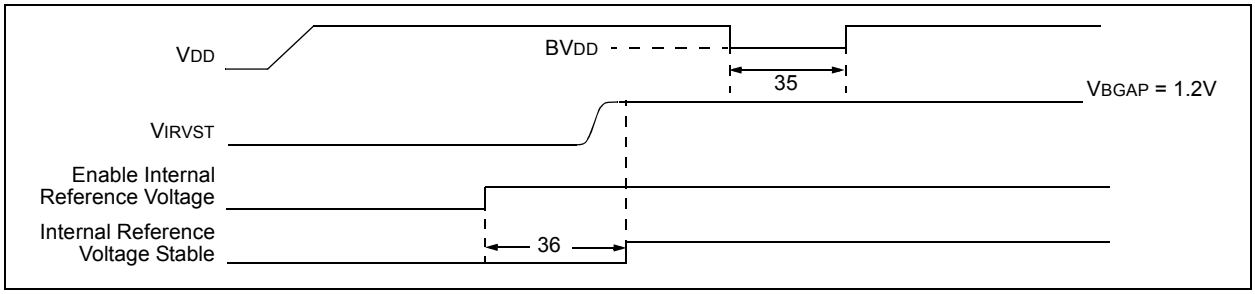



**TABLE 28-10: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS**

| Param. No. | Sym | Characteristic | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 30 | TMCL | $\overline{\text{MCLR}}$ Pulse Width (low) | 2 | — | — | µs | |
| 31 | TWDT | Watchdog Timer Time-out Period (no postscaler) | 3.4 | 4.0 | 4.6 | ms | |
| 32 | TOST | Oscillator Start-up Timer Period | 1024 TOSC | — | 1024 TOSC | — | TOSC = OSC1 period |
| 33 | TPWRT | Power-up Timer Period | 55.6 | 65.5 | 75 | ms | |
| 34 | TIOZ | I/O High-Impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset | — | 2 | — | µs | |
| 35 | TBOR | Brown-out Reset Pulse Width | 200 | — | — | µs | VDD ≤ BVDD (see D005) |
| 36 | TIRVST | Time for Internal Reference Voltage to become Stable | — | 20 | 50 | µs | |
| 37 | TLVD | High/Low-Voltage Detect Pulse Width | 200 | — | — | µs | VDD ≤ VLVD |
| 38 | TCSD | CPU Start-up Time | — | 10 | — | µs | |
| 39 | TIOBST | Time for INTOSC to Stabilize | — | 1 | — | µs | |

**FIGURE 28-9: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**

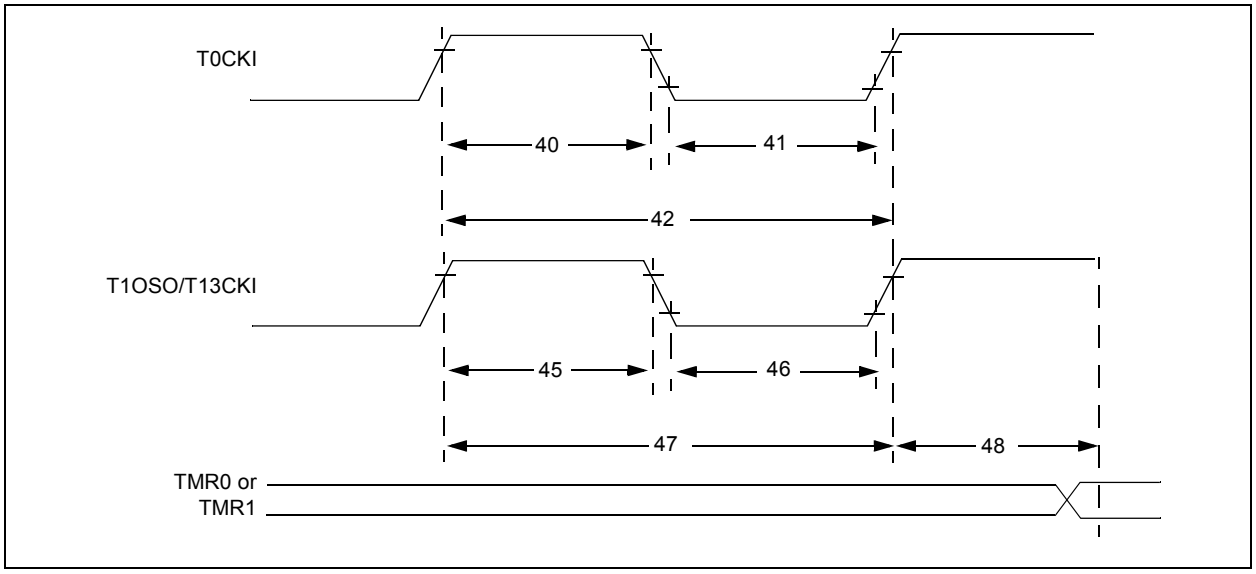



**TABLE 28-11: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

| Param No. | Sym | 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 (TCY + 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 | PIC18**F**XXXX | 10 | — | ns | |
| | | | | PIC18**LF**XXXX | 25 | — | ns | VDD = 2.0V |
| | | | Asynchronous | PIC18**F**XXXX | 30 | — | ns | |
| | | | | PIC18**LF**XXXX | 50 | — | ns | VDD = 2.0V |
| 46 | TT1L | T13CKI Low Time | Synchronous, no prescaler | | 0.5 TCY + 5 | — | ns | |
| | | | Synchronous, with prescaler | PIC18**F**XXXX | 10 | — | ns | |
| | | | | PIC18**LF**XXXX | 25 | — | ns | VDD = 2.0V |
| | | | Asynchronous | PIC18**F**XXXX | 30 | — | ns | |
| | | | | PIC18**LF**XXXX | 50 | — | ns | VDD = 2.0V |
| 47 | TT1P | T13CKI Input Period | Synchronous | | Greater of: 20 ns or (TCY + 40)/N | — | ns | N = prescale value (1, 2, 4, 8) |
| | | | Asynchronous | | 60 | — | ns | |
| | FT1 | T13CKI Oscillator Input Frequency Range | | | DC | 50 | kHz | |
| 48 | TCKE2TMRI | Delay from External T13CKI Clock Edge to Timer Increment | | | 2 TOSC | 7 TOSC | — | |


© 2009 Microchip Technology Inc.

**FIGURE 28-10: CAPTURE/COMPARE/PWM TIMINGS (ALL CCP MODULES)**

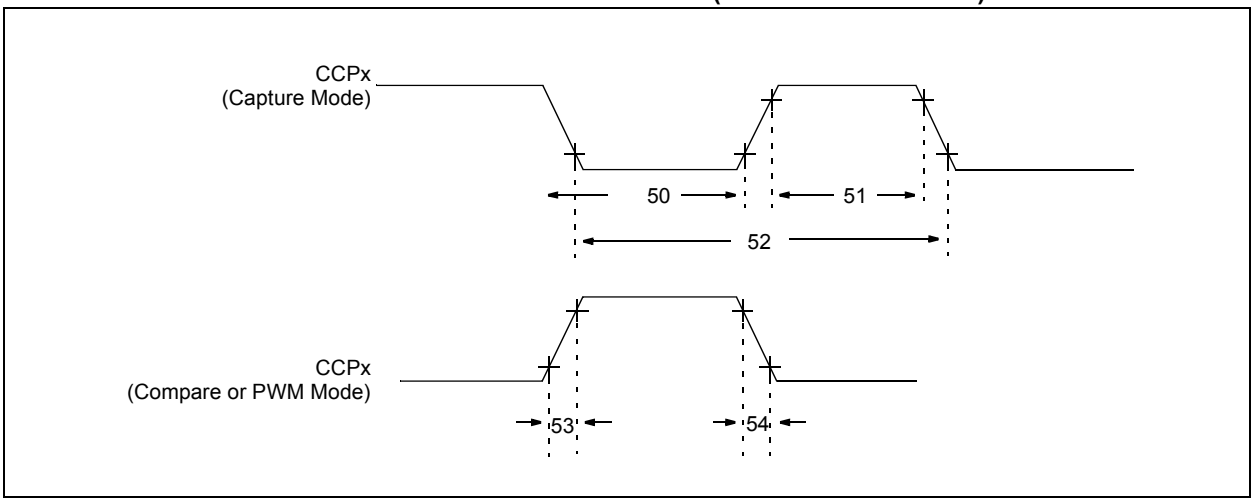



**TABLE 28-12: CAPTURE/COMPARE/PWM REQUIREMENTS (ALL CCP MODULES)**

| Param No. | Sym | Characteristic | | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 50 | TccL | CCPx Input Low Time | No prescaler | | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | PIC18**F**XXXX | 10 | — | ns | |
| | | | | PIC18**LF**XXXX | 20 | — | ns | VDD = 2.0V |
| 51 | TccH | CCPx Input High Time | No prescaler | | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | PIC18**F**XXXX | 10 | — | ns | |
| | | | | PIC18**LF**XXXX | 20 | — | ns | VDD = 2.0V |
| 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 | | PIC18**F**XXXX | — | 25 | ns | |
| | | | | PIC18**LF**XXXX | — | 45 | ns | VDD = 2.0V |
| 54 | TccF | CCPx Output Fall Time | | PIC18**F**XXXX | — | 25 | ns | |
| | | | | PIC18**LF**XXXX | — | 45 | ns | VDD = 2.0V |

**FIGURE 28-11: PARALLEL SLAVE PORT TIMING (PIC18F4480/4580)**

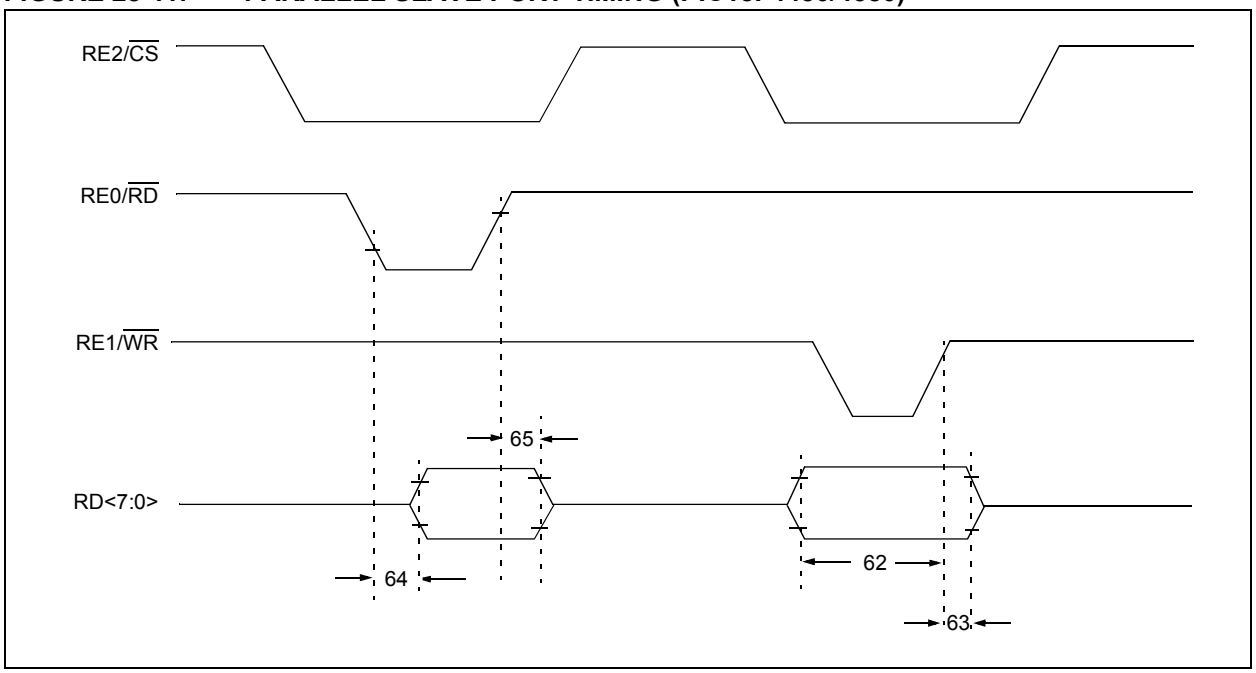



**TABLE 28-13: PARALLEL SLAVE PORT REQUIREMENTS (PIC18F4480/4580)**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 62 | TDTV2WRH | Data In Valid before $\overline{WR}$ ↑ or $\overline{CS}$ ↑ (setup time) | | 20 | — | ns | |
| 63 | TWRH2DTI | $\overline{WR}$ ↑ or $\overline{CS}$ ↑ to Data–In Invalid (hold time) | PIC18**F**XXXX | 20 | — | ns | |
| | | | PIC18**LF**XXXX | 35 | — | ns | VDD = 2.0V |
| 64 | TRDL2DTV | $\overline{RD}$ ↓ and $\overline{CS}$ ↓ to Data–Out Valid | | — | 80 | ns | |
| 65 | TRDH2DTI | $\overline{RD}$ ↑ or $\overline{CS}$ ↓ to Data–Out Invalid | | 10 | 30 | ns | |
| 66 | TIBFINH | Inhibit of the IBF Flag bit being Cleared from $\overline{WR}$ ↑ or $\overline{CS}$ ↑ | | — | 3 TCY | | |


© 2009 Microchip Technology Inc.

**FIGURE 28-12: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)**

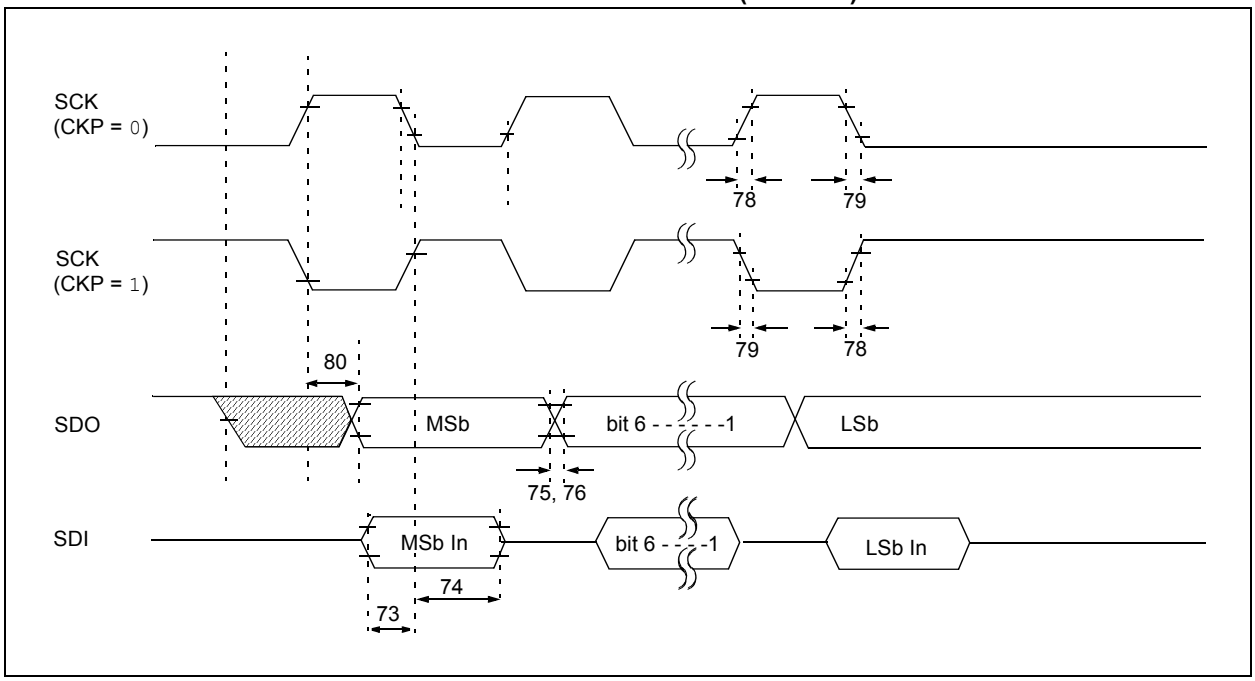



**TABLE 28-14: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 73 | TDIV2SCH, TDIV2SCL | Setup Time of SDI Data Input to SCK Edge | | 100 | — | ns | |
| 74 | TSCH2DIL, TSCL2DIL | Hold Time of SDI Data Input to SCK Edge | | 100 | — | ns | |
| 75 | TDOR | SDO Data Output Rise Time | PIC18**F**XXXX | — | 25 | ns | |
| | | | PIC18**LF**XXXX | — | 45 | ns | VDD = 2.0V |
| 76 | TDOF | SDO Data Output Fall Time | | — | 25 | ns | |
| 78 | TSCR | SCK Output Rise Time | PIC18**F**XXXX | — | 25 | ns | |
| | | | PIC18**LF**XXXX | — | 45 | ns | VDD = 2.0V |
| 79 | TSCF | SCK Output Fall Time | | — | 25 | ns | |
| 80 | TSCH2DOV, TSCL2DOV | SDO Data Output Valid after SCK Edge | PIC18**F**XXXX | — | 50 | ns | |
| | | | PIC18**LF**XXXX | — | 100 | ns | VDD = 2.0V |

**FIGURE 28-13: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)**

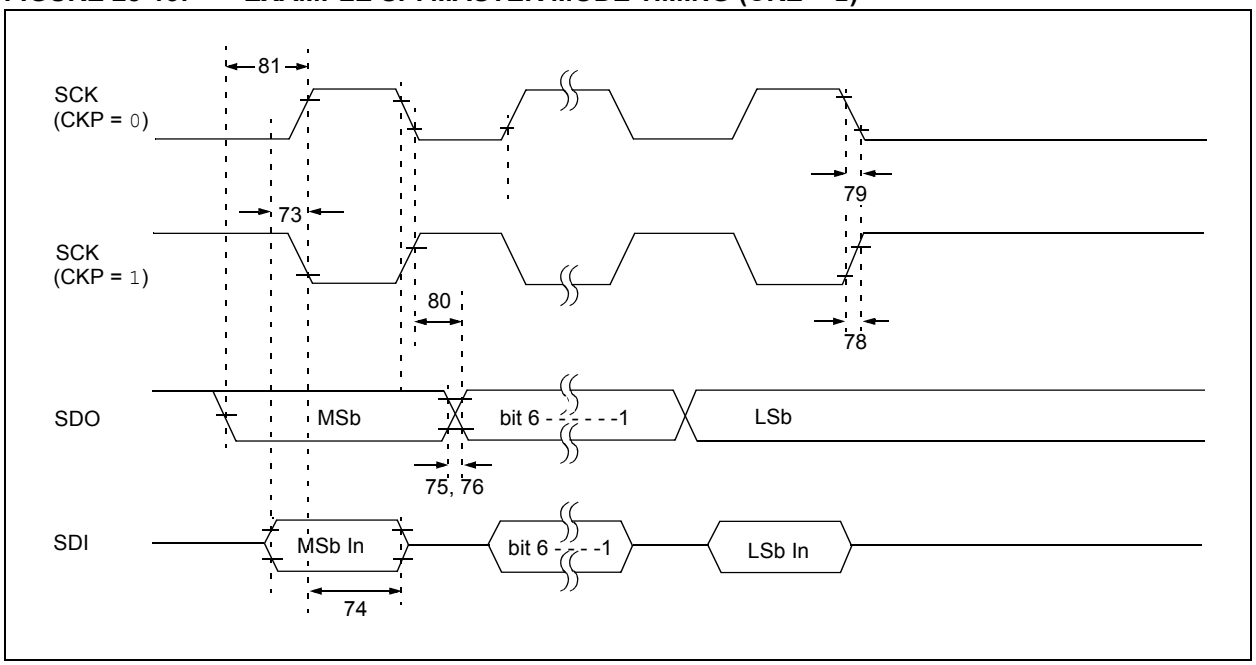



**TABLE 28-15: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 73 | TDIV2SCH, TDIV2SCL | Setup Time of SDI Data Input to SCK Edge | | 100 | — | ns | |
| 74 | TSCH2DIL, TSCL2DIL | Hold Time of SDI Data Input to SCK Edge | | 100 | — | ns | |
| 75 | TDOR | SDO Data Output Rise Time | PIC18**F**XXXX | — | 25 | ns | |
| | | | PIC18**LF**XXXX | | 45 | ns | VDD = 2.0V |
| 76 | TDOF | SDO Data Output Fall Time | | — | 25 | ns | |
| 78 | TSCR | SCK Output Rise Time | PIC18**F**XXXX | — | 25 | ns | |
| | | | PIC18**LF**XXXX | | 45 | ns | VDD = 2.0V |
| 79 | TSCF | SCK Output Fall Time | | — | 25 | ns | |
| 80 | TSCH2DOV, TSCL2DOV | SDO Data Output Valid after SCK Edge | PIC18**F**XXXX | — | 50 | ns | |
| | | | PIC18**LF**XXXX | | 100 | ns | VDD = 2.0V |
| 81 | TDOV2SCH, TDOV2SCL | SDO Data Output Setup to SCK Edge | | TCY | — | ns | |


© 2009 Microchip Technology Inc.

**FIGURE 28-14: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)**

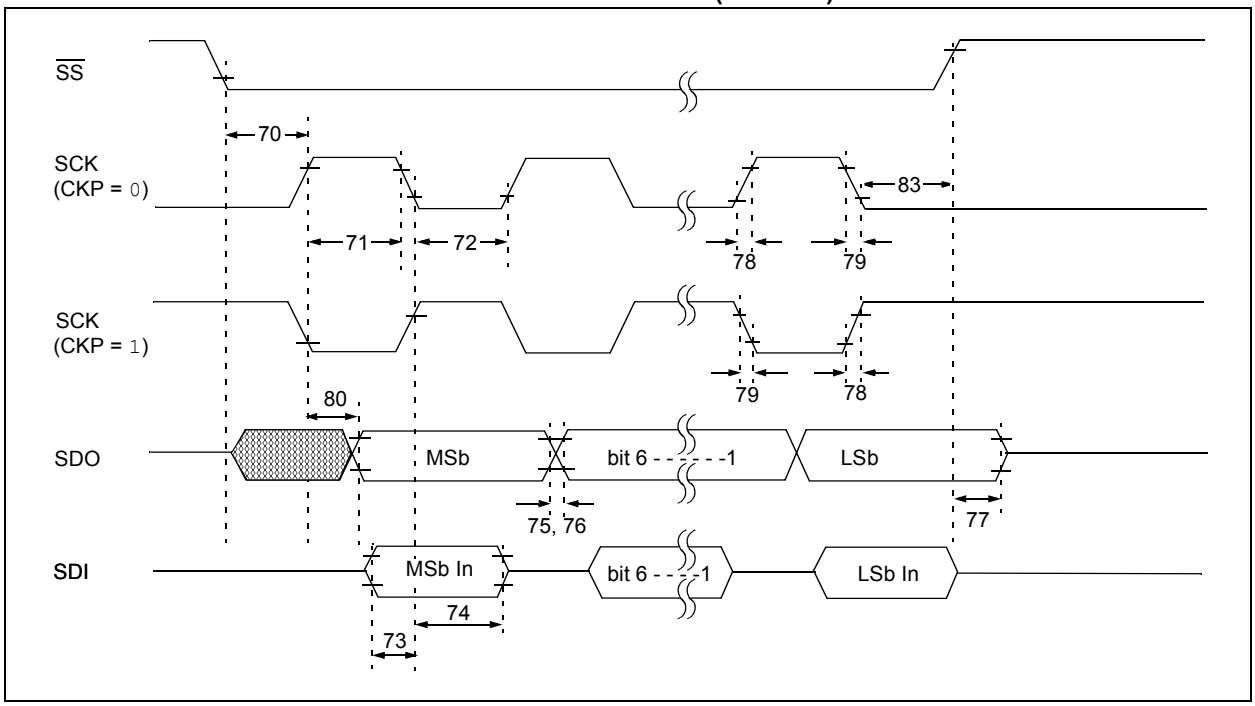



**TABLE 28-16: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input | | 3 Tcy | — | ns | |
| 71 | TscH | SCK Input High Time | Continuous | 1.25 Tcy + 30 | — | ns | |
| 71A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 72 | TscL | SCK Input Low Time | Continuous | 1.25 Tcy + 30 | — | ns | |
| 72A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 73 | TdiV2scH, TdiV2scL | Setup Time of SDI Data Input to SCK Edge | | 20 | — | ns | |
| 73A | Tb2b | Last Clock Edge of Byte1 to the First Clock Edge of Byte 2 | | 1.5 Tcy + 40 | — | ns | **(Note 2)** |
| 74 | TscH2diL, TscL2diL | Hold Time of SDI Data Input to SCK Edge | | 40 | — | ns | |
| 75 | TdoR | SDO Data Output Rise Time | PIC18**F**XXXX | — | 25 | ns | |
| | | | PIC18**LF**XXXX | | 45 | ns | VDD = 2.0V |
| 76 | TdoF | SDO Data Output Fall Time | | — | 25 | ns | |
| 77 | TssH2doZ | $\overline{SS}$ ↑ to SDO Output High-Impedance | | 10 | 50 | ns | |
| 80 | TscH2doV, TscL2doV | SDO Data Output Valid after SCK Edge | PIC18**F**XXXX | — | 50 | ns | |
| | | | PIC18**LF**XXXX | | 100 | ns | VDD = 2.0V |
| 83 | TscH2ssH, TscL2ssH | $\overline{SS}$ ↑ after SCK Edge | | 1.5 Tcy + 40 | — | ns | |

**Note 1:** Requires the use of Parameter #73A.

**2:** Only if Parameter #71A and #72A are used.

**FIGURE 28-15: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)**

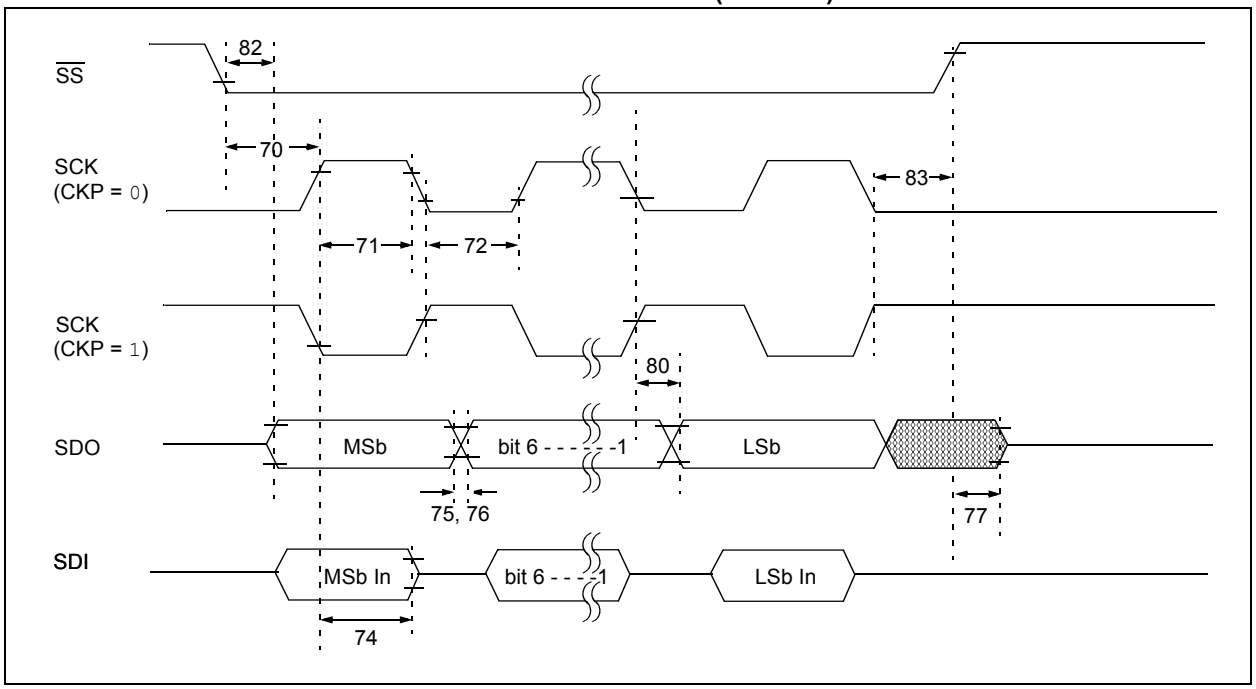



**TABLE 28-17: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input | | 3 Tcy | — | ns | |
| 71 | TscH | SCK Input High Time | Continuous | 1.25 Tcy + 30 | — | ns | |
| 71A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 72 | TscL | SCK Input Low Time | Continuous | 1.25 Tcy + 30 | — | ns | |
| 72A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 73A | Tb2b | Last Clock Edge of Byte 1 to the flrst Clock Edge of Byte 2 | | 1.5 Tcy + 40 | — | ns | **(Note 2)** |
| 74 | TscH2diL, TscL2diL | Hold Time of SDI Data Input to SCK Edge | | 40 | — | ns | |
| 75 | TdoR | SDO Data Output Rise Time | PIC18**F**XXXX | — | 25 | ns | |
| | | | PIC18**LF**XXXX | | 45 | ns | VDD = 2.0V |
| 76 | TdoF | SDO Data Output Fall Time | | — | 25 | ns | |
| 77 | TssH2doZ | $\overline{SS}$↑ to SDO Output High-Impedance | | 10 | 50 | ns | |
| 80 | TscH2doV, TscL2doV | SDO Data Output Valid after SCK Edge | PIC18**F**XXXX | — | 50 | ns | |
| | | | PIC18**LF**XXXX | — | 100 | ns | VDD = 2.0V |
| 82 | TssL2doV | SDO Data Output Valid after $\overline{SS}$ ↓ Edge | PIC18**F**XXXX | — | 50 | ns | |
| | | | PIC18**LF**XXXX | — | 100 | ns | VDD = 2.0V |
| 83 | TscH2ssH, TscL2ssH | $\overline{SS}$ ↑ after SCK Edge | | 1.5 Tcy + 40 | — | ns | |

**Note 1:** Requires the use of Parameter #73A.

**2:** Only if Parameter #71A and #72A are used.


© 2009 Microchip Technology Inc.

**FIGURE 28-16: I²C™ BUS START/STOP BITS TIMING**

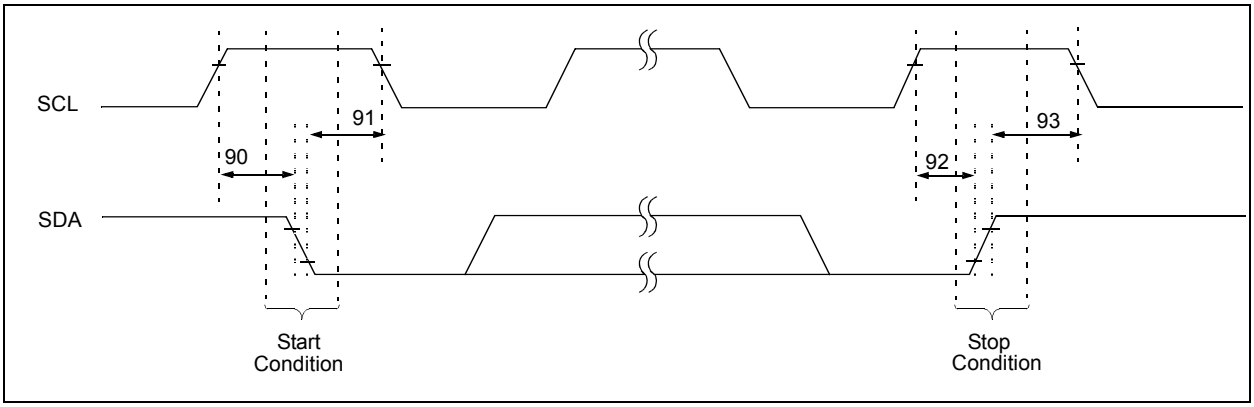



**TABLE 28-18: I²C™ BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 90 | TSU:STA | Start Condition | 100 kHz mode | 4700 | — | ns | Only relevant for Repeated Start condition |
| | | Setup Time | 400 kHz mode | 600 | — | | |
| 91 | THD:STA | Start Condition | 100 kHz mode | 4000 | — | ns | After this period, the first clock pulse is generated |
| | | Hold Time | 400 kHz mode | 600 | — | | |
| 92 | TSU:STO | Stop Condition | 100 kHz mode | 4700 | — | ns | |
| | | Setup Time | 400 kHz mode | 600 | — | | |
| 93 | THD:STO | Stop Condition | 100 kHz mode | 4000 | — | ns | |
| | | Hold Time | 400 kHz mode | 600 | — | | |

**FIGURE 28-17: I²C™ BUS DATA TIMING**

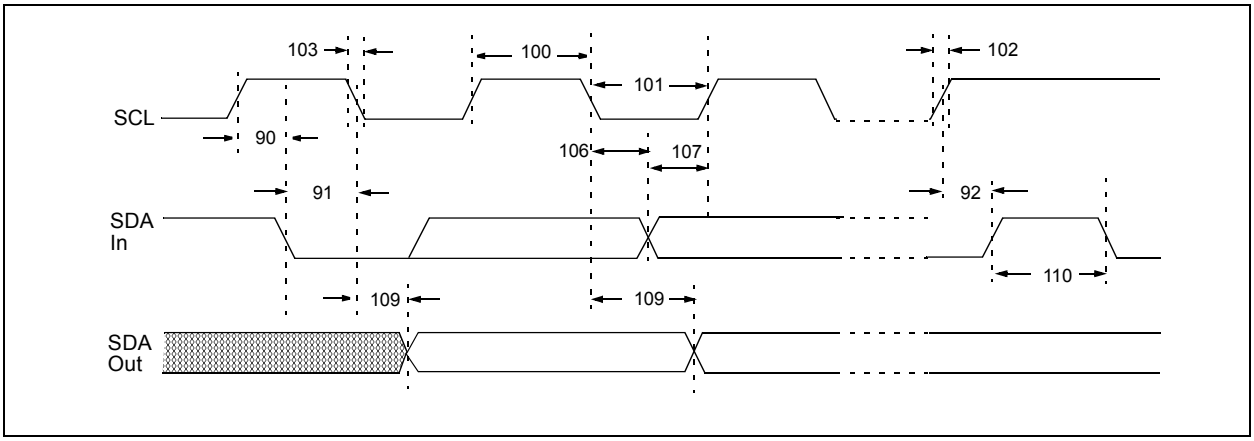


**TABLE 28-19: I²C™ BUS DATA REQUIREMENTS (SLAVE MODE)**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 4.0 | — | μs | PIC18FXXXX must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 0.6 | — | μs | PIC18FXXXX must operate at a minimum of 10 MHz |
| | | | MSSP module | 1.5 TCY | — | | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 4.7 | — | μs | PIC18FXXXX must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 1.3 | — | μs | PIC18FXXXX must operate at a minimum of 10 MHz |
| | | | MSSP module | 1.5 TCY | — | | |
| 102 | TR | SDA and SCL Rise Time | 100 kHz mode | — | 1000 | ns | |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | CB is specified to be from 10 to 400 pF |
| 103 | TF | SDA and SCL Fall Time | 100 kHz mode | — | 300 | ns | |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | CB is specified to be from 10 to 400 pF |
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 4.7 | — | μs | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 0.6 | — | μs | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 4.0 | — | μs | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 0.6 | — | μs | |
| 106 | THD:DAT | Data Input Hold Time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | μs | |
| 107 | TSU:DAT | Data Input Setup Time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 4.7 | — | μs | |
| | | | 400 kHz mode | 0.6 | — | μs | |
| 109 | TAA | Output Valid from Clock | 100 kHz mode | — | 3500 | ns | **(Note 1)** |
| | | | 400 kHz mode | — | — | ns | |
| 110 | TBUF | Bus Free Time | 100 kHz mode | 4.7 | — | μs | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | μs | |
| D102 | CB | Bus Capacitive Loading | | — | 400 | pF | |

**Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

**2:** A Fast mode I²C™ bus device can be used in a Standard mode I²C bus system, but the requirement TSU:DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line, TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I²C bus specification), before the SCL line is released.


© 2009 Microchip Technology Inc.

**FIGURE 28-18: MASTER SSP I²C™ BUS START/STOP BITS TIMING WAVEFORMS**

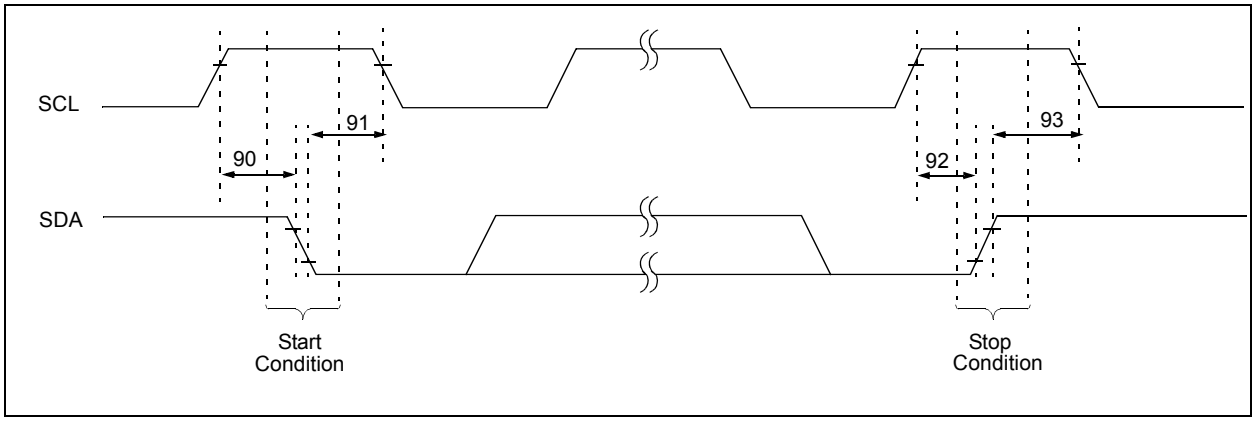



**TABLE 28-20: MASTER SSP I²C™ BUS START/STOP BITS REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 90 | TSU:STA | Start condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |
| 93 | THD:STO | Stop Condition Hold Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |

**Note 1:** Maximum pin capacitance = 10 pF for all I²C pins.

**FIGURE 28-19: MASTER SSP I²C™ BUS DATA TIMING**

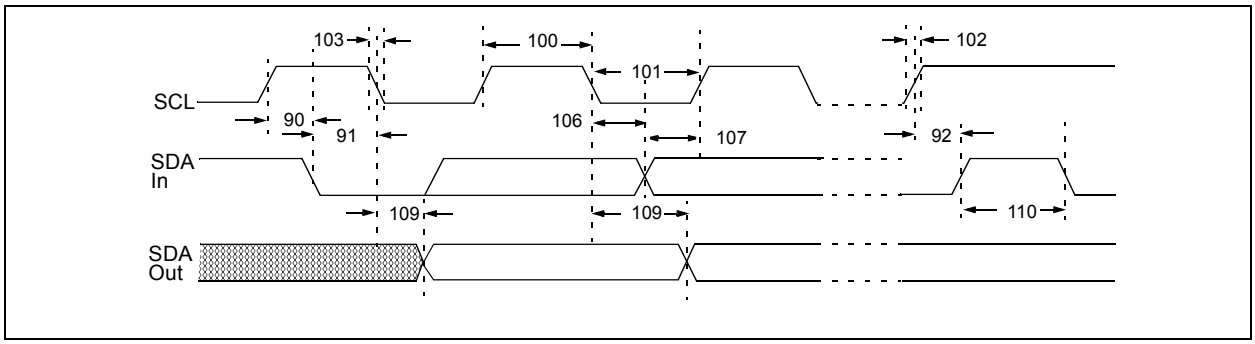


**TABLE 28-21: MASTER SSP I²C™ BUS DATA REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 102 | TR | SDA and SCL Rise Time | 100 kHz mode | — | 1000 | ns | CB is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | |
| | | | 1 MHz mode(1) | — | 300 | ns | |
| 103 | TF | SDA and SCL Fall Time | 100 kHz mode | — | 300 | ns | CB is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | |
| | | | 1 MHz mode(1) | — | 100 | ns | |
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 106 | THD:DAT | Data Input Hold Time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | ms | |
| 107 | TSU:DAT | Data Input Setup Time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 109 | TAA | Output Valid from Clock | 100 kHz mode | — | 3500 | ns | |
| | | | 400 kHz mode | — | 1000 | ns | |
| | | | 1 MHz mode(1) | — | — | ns | |
| 110 | TBUF | Bus Free Time | 100 kHz mode | 4.7 | — | ms | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | ms | |
| D102 | CB | Bus Capacitive Loading | | — | 400 | pF | |

**Note 1:** Maximum pin capacitance = 10 pF for all I²C™ pins.

**2:** A Fast mode I²C bus device can be used in a Standard mode I²C bus system, but parameter #107 ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line, parameter #102 + parameter #107 = 1000 + 250 = 1250 ns (for 100 kHz mode), before the SCL line is released.


© 2009 Microchip Technology Inc.

**FIGURE 28-20: EUSART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

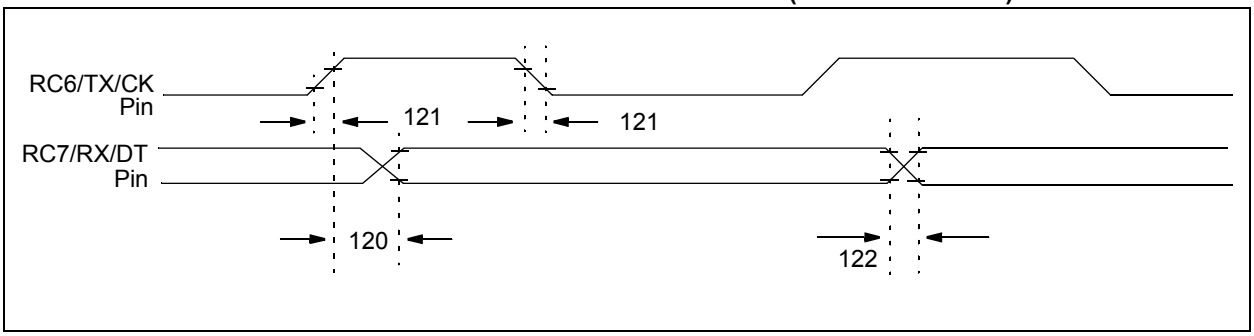
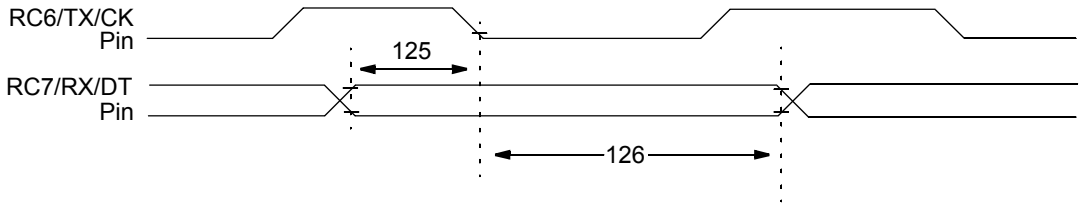



**TABLE 28-22: EUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 120 | TCKH2DTV | SYNC XMIT (MASTER & SLAVE) Clock High to Data Out Valid | PIC18**F**XXXX | — | 40 | ns | |
| | | | PIC18**LF**XXXX | — | 100 | ns | VDD = 2.0V |
| 121 | TCKRF | Clock Out Rise Time and Fall Time (Master mode) | PIC18**F**XXXX | — | 20 | ns | |
| | | | PIC18**LF**XXXX | — | 50 | ns | VDD = 2.0V |
| 122 | TDTRF | Data Out Rise Time and Fall Time | PIC18**F**XXXX | — | 20 | ns | |
| | | | PIC18**LF**XXXX | — | 50 | ns | VDD = 2.0V |

**FIGURE 28-21: EUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**




**Note:** Refer to Figure 28-4 for load conditions.

**TABLE 28-23: EUSART SYNCHRONOUS RECEIVE REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 125 | TDTV2CKL | SYNC RCV (MASTER & SLAVE) Data Hold before CK ↓ (DT hold time) | 10 | — | ns | |
| 126 | TCKL2DTL | Data Hold after CK ↓ (DT hold time) | 15 | — | ns | |

**TABLE 28-24: A/D CONVERTER CHARACTERISTICS: PIC18F2480/2580/4480/4580 (INDUSTRIAL) PIC18LF2480/2580/4480/4580 (INDUSTRIAL)**

| Param No. | Sym | 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 | | — | — | <±2 | LSb | ΔVREF ≥ 3.0V |
| A07 | EGN | Gain Error | | — | — | <±1 | LSb | ΔVREF ≥ 3.0V |
| A10 | — | Monotonicity | | Guaranteed(1) | | | — | |
| 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) | PIC18**F**XXXX | — | 180 | — | μA | Average current consumption when A/D is on **(Note 2)** |
| | | | PIC18**LF**XXXX | — | 90 | — | μA | VDD = 2.0V; average current consumption when A/D is on **(Note 2)** |
| A50 | IREF | VREF Input Current **(Note 3)** | | —<br>— | —<br>— | ±5<br>±150 | μA<br>μA | During VAIN acquisition.<br>During A/D conversion cycle. |

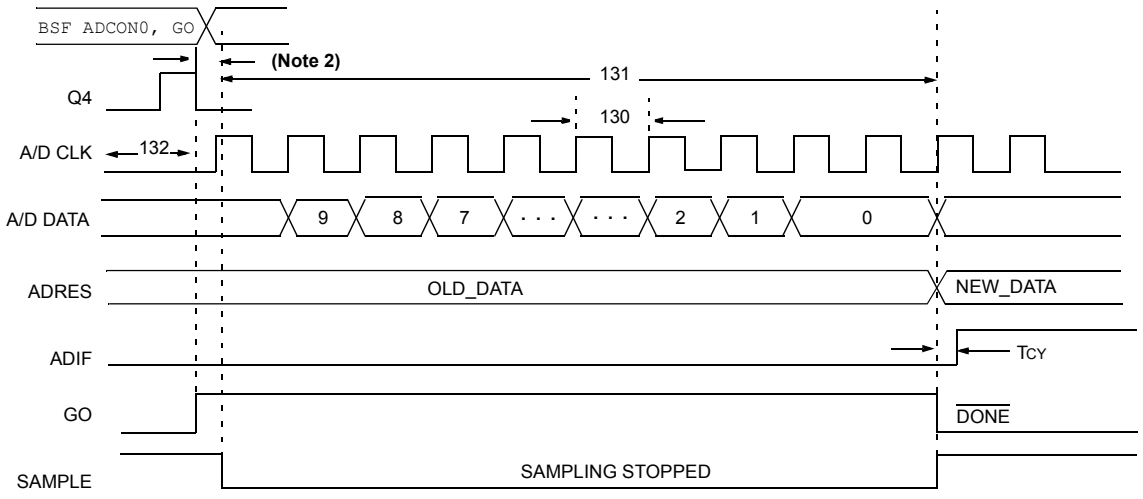
**Note 1:** The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

**2:** When A/D is off, it will not consume any current other than minor leakage current. The power-down current spec includes any such leakage from the A/D module.

**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.


© 2009 Microchip Technology Inc.

**FIGURE 28-22: 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 28-25: A/D CONVERSION REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 130 | TAD | A/D Clock Period | PIC18**F**XXXX | 0.7 | 25.0(1) | µs | TOSC based, VREF ≥ 3.0V |
| | | | PIC18**LF**XXXX | 1.4 | 25.0(1) | µs | VDD = 2.0V; TOSC based, VREF full range |
| | | | PIC18**F**XXXX | — | 1 | µs | A/D RC mode |
| | | | PIC18**LF**XXXX | — | 3 | µs | VDD = 2.0V; A/D RC mode |
| 131 | TCNV | Conversion Time (not including acquisition time) **(Note 2)** | | 11 | 12 | TAD | |
| 132 | TACQ | Acquisition Time **(Note 3)** | | 1.4 | — | µs | -40°C to +85°C |
| 135 | TSWC | Switching Time from Convert → Sample | | — | **(Note 4)** | — | |
| 136 | TAMP | Amplifier Settling Time **(Note 5)** | | 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:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

**2:** ADRES register may be read on the following TCY cycle.

**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 following cycle of the device clock.

**5:** See **Section 20.0 "10-Bit Analog-to-Digital Converter (A/D) Module"** for minimum conditions when input voltage has changed more than 1 LSb.

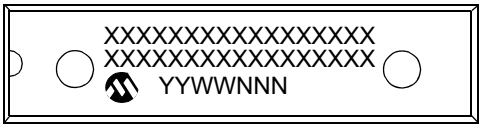
**NOTES:**


© 2009 Microchip Technology Inc.

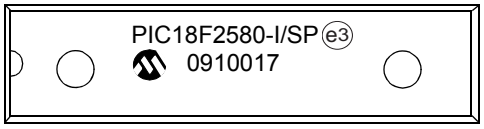
## 29.0 PACKAGING INFORMATION

### 29.1 Package Marking Information

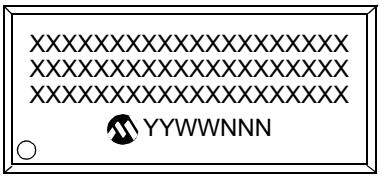
28-Lead SPDIP



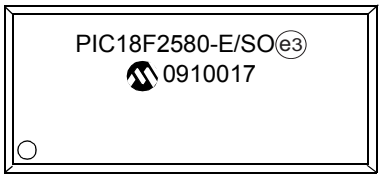

Example



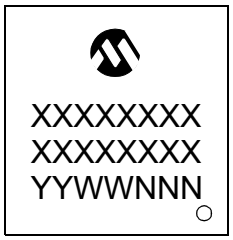

28-Lead SOIC




Example




28-Lead QFN



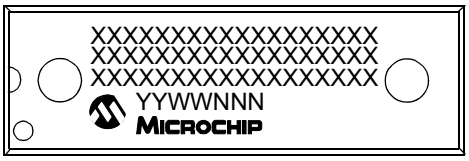

Example

| | | |
|---|---|---|
| **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. |

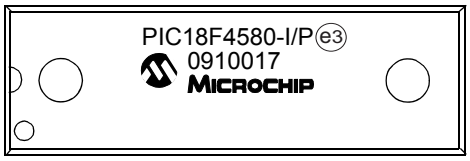
**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

### 29.1 Package Marking Information (Continued)

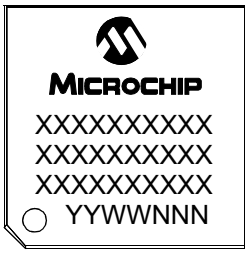
40-Lead PDIP



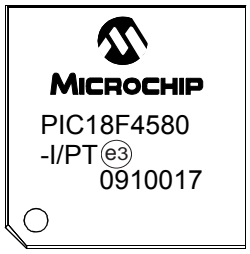

Example



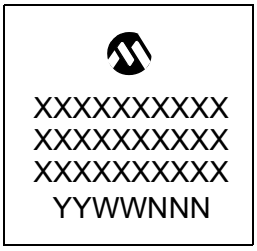

44-Lead TQFP




Example




44-Lead QFN




Example

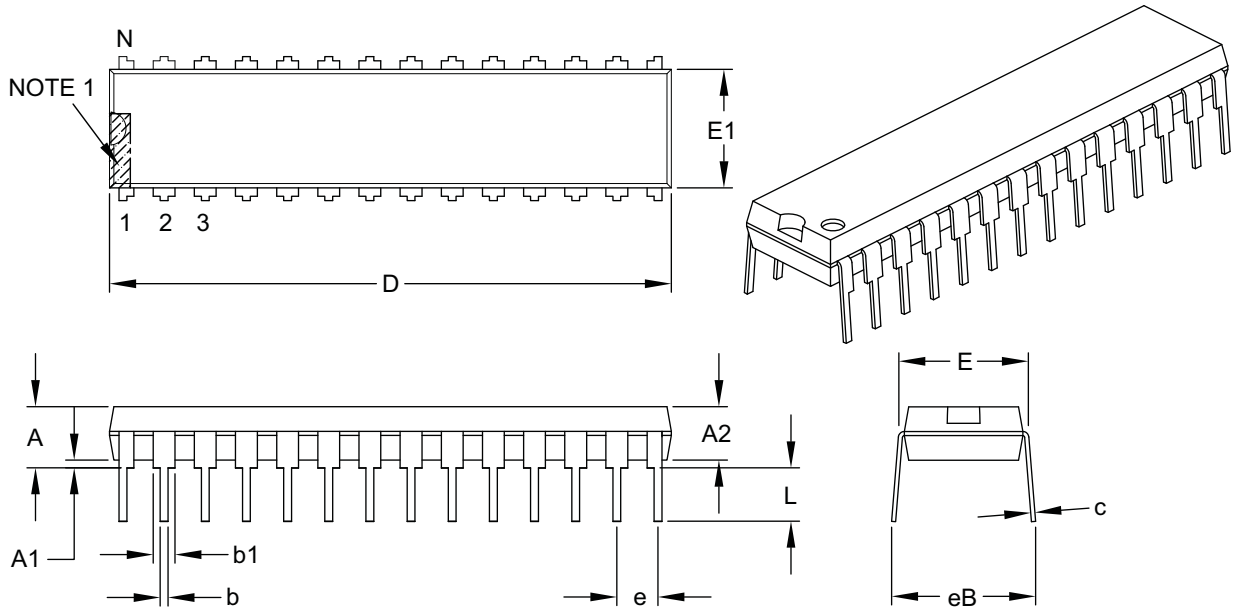
© 2009 Microchip Technology Inc.

## 29.2 Package Details

The following sections give the technical details of the packages.

### 28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

**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 | | 28 | |
| Pitch | e | | .100 BSC | |
| Top to Seating Plane | A | – | – | .200 |
| Molded Package Thickness | A2 | .120 | .135 | .150 |
| Base to Seating Plane | A1 | .015 | – | – |
| Shoulder to Shoulder Width | E | .290 | .310 | .335 |
| Molded Package Width | E1 | .240 | .285 | .295 |
| Overall Length | D | 1.345 | 1.365 | 1.400 |
| Tip to Seating Plane | L | .110 | .130 | .150 |
| Lead Thickness | c | .008 | .010 | .015 |
| Upper Lead Width | b1 | .040 | .050 | .070 |
| Lower Lead Width | b | .014 | .018 | .022 |
| Overall Row Spacing § | eB | – | – | .430 |

**Notes:**

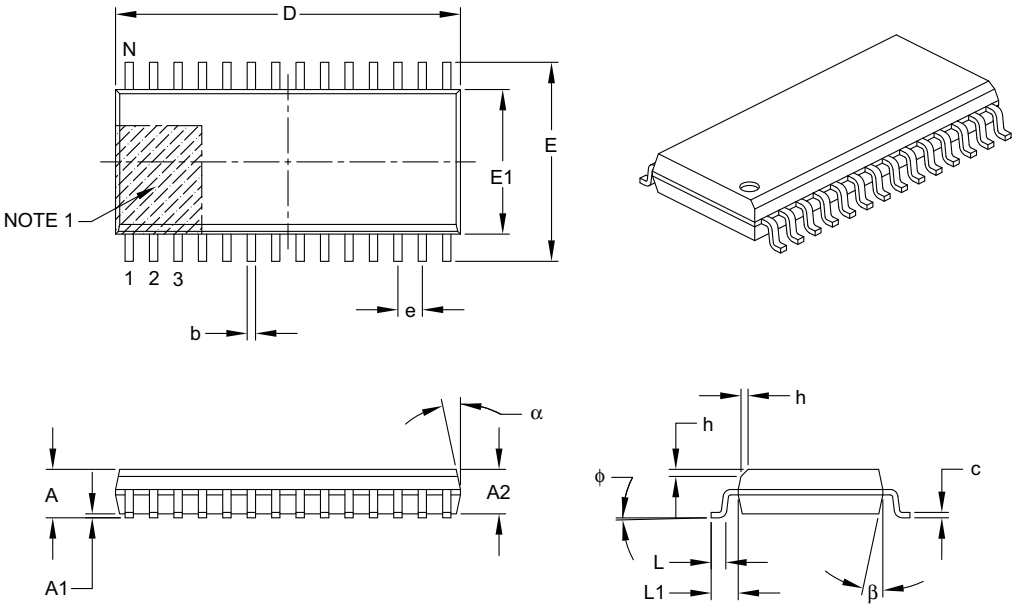
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

### 28-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




| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| 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 | | 17.90 BSC | |
| Chamfer (optional) | h | 0.25 | – | 0.75 |
| Foot Length | L | 0.40 | – | 1.27 |
| Footprint | L1 | | 1.40 REF | |
| Foot Angle Top | ϕ | 0° | – | 8° |
| Lead Thickness | c | 0.18 | – | 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. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

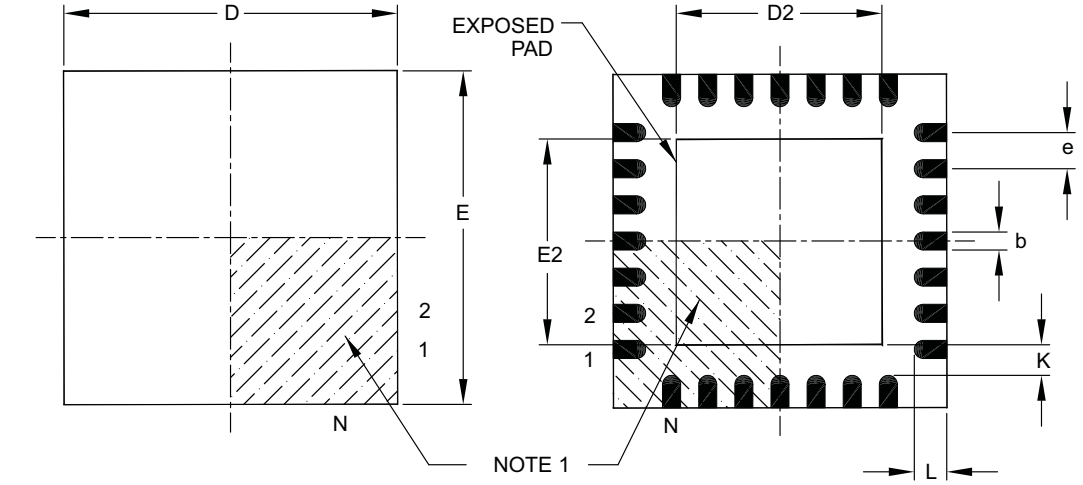
   REF: Reference Dimension, usually without tolerance, for information purposes only.

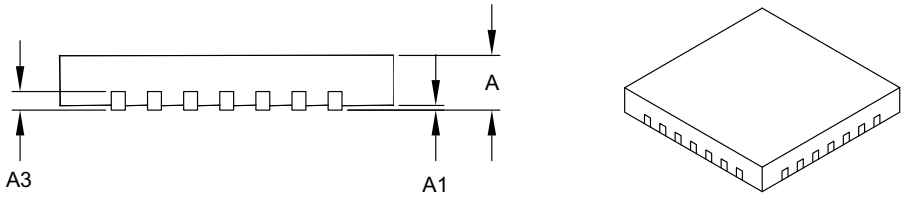
Microchip Technology Drawing C04-052B


© 2009 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




| 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 |
| Contact 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 |
| Contact Width | b | 0.23 | 0.30 | 0.35 |
| Contact Length | L | 0.50 | 0.55 | 0.70 |
| Contact-to-Exposed Pad | K | 0.20 | – | – |

**Notes:**

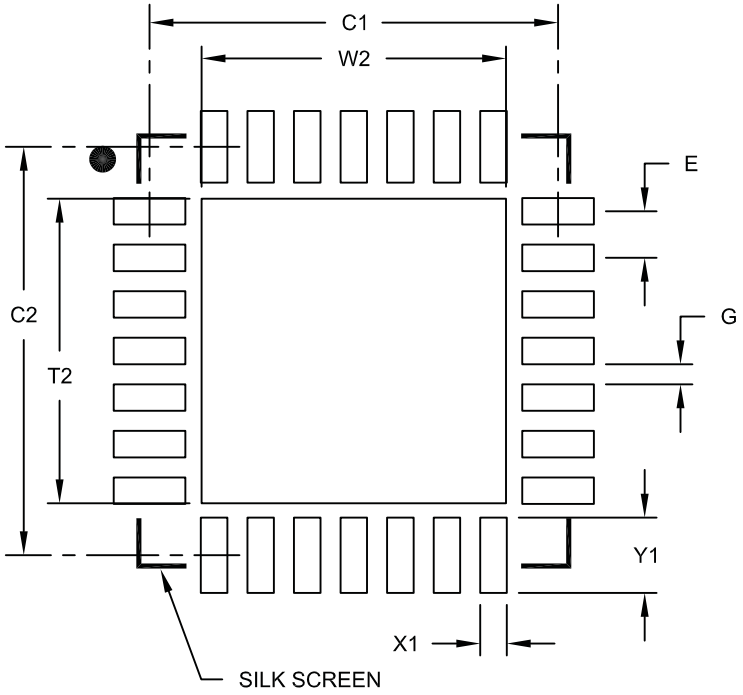
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-105B

### 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

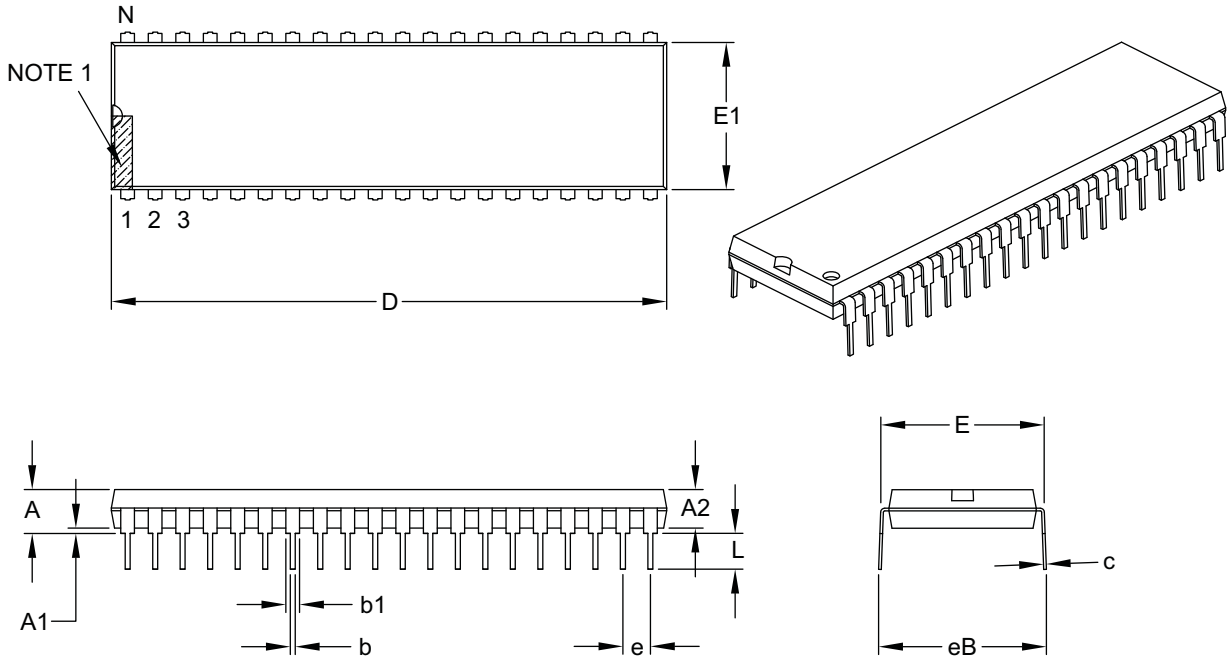
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2105A


© 2009 Microchip Technology Inc.

## 40-Lead Plastic Dual In-Line (P) – 600 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 | | 40 | |
| Pitch | e | | .100 BSC | |
| Top to Seating Plane | A | – | – | .250 |
| Molded Package Thickness | A2 | .125 | – | .195 |
| Base to Seating Plane | A1 | .015 | – | – |
| Shoulder to Shoulder Width | E | .590 | – | .625 |
| Molded Package Width | E1 | .485 | – | .580 |
| Overall Length | D | 1.980 | – | 2.095 |
| Tip to Seating Plane | L | .115 | – | .200 |
| Lead Thickness | c | .008 | – | .015 |
| Upper Lead Width | b1 | .030 | – | .070 |
| Lower Lead Width | b | .014 | – | .023 |
| Overall Row Spacing § | eB | – | – | .700 |

**Notes:**

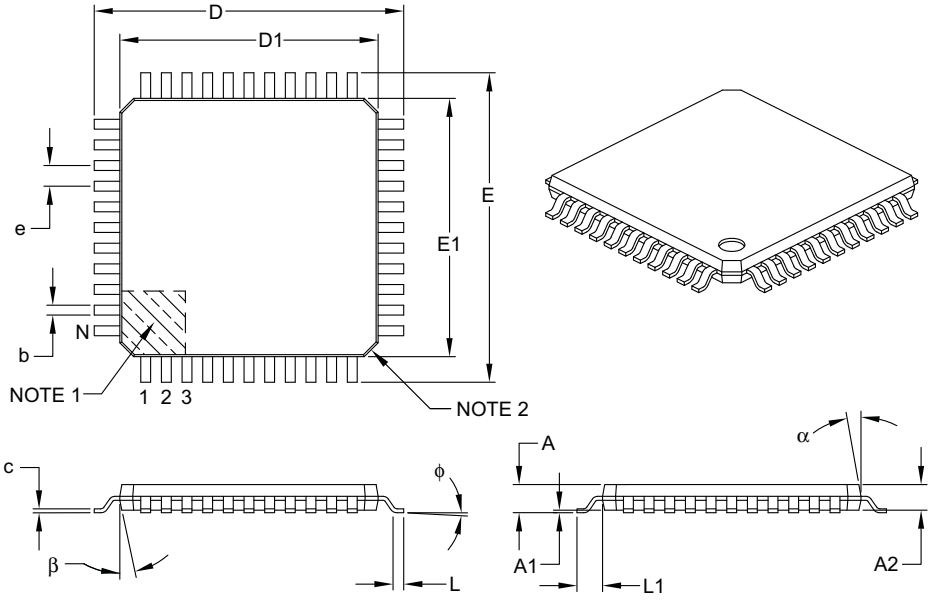
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-016B

### 44-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

**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 Leads | N | | 44 | |
| Lead Pitch | e | | 0.80 BSC | |
| Overall Height | A | – | – | 1.20 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Standoff | A1 | 0.05 | – | 0.15 |
| Foot Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | | 1.00 REF | |
| Foot Angle | φ | 0° | 3.5° | 7° |
| Overall Width | E | | 12.00 BSC | |
| Overall Length | D | | 12.00 BSC | |
| Molded Package Width | E1 | | 10.00 BSC | |
| Molded Package Length | D1 | | 10.00 BSC | |
| Lead Thickness | c | 0.09 | – | 0.20 |
| Lead Width | b | 0.30 | 0.37 | 0.45 |
| Mold Draft Angle Top | α | 11° | 12° | 13° |
| Mold Draft Angle Bottom | β | 11° | 12° | 13° |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions 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.

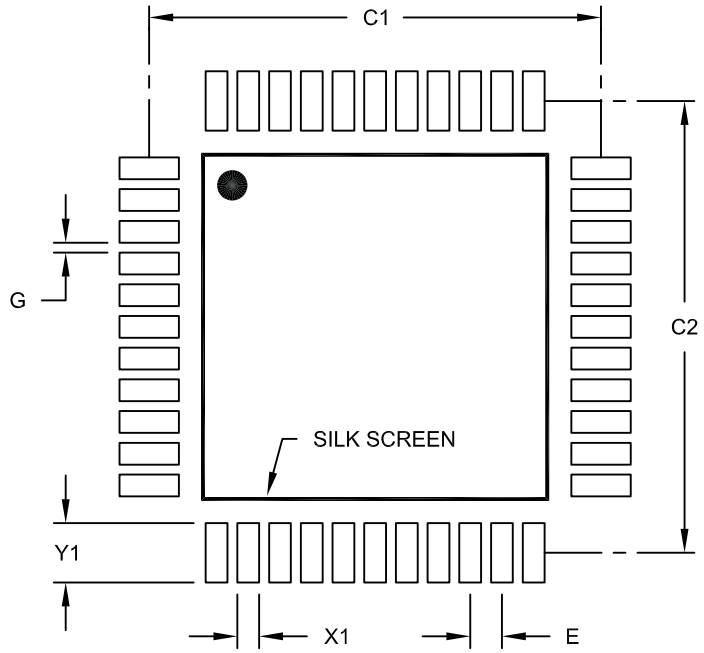
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-076B


© 2009 Microchip Technology Inc.

### 44-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

**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 | | 0.80 BSC | |
| Contact Pad Spacing | C1 | | 11.40 | |
| Contact Pad Spacing | C2 | | 11.40 | |
| Contact Pad Width (X44) | X1 | | | 0.55 |
| Contact Pad Length (X44) | Y1 | | | 1.50 |
| Distance Between Pads | G | 0.25 | | |

Notes:

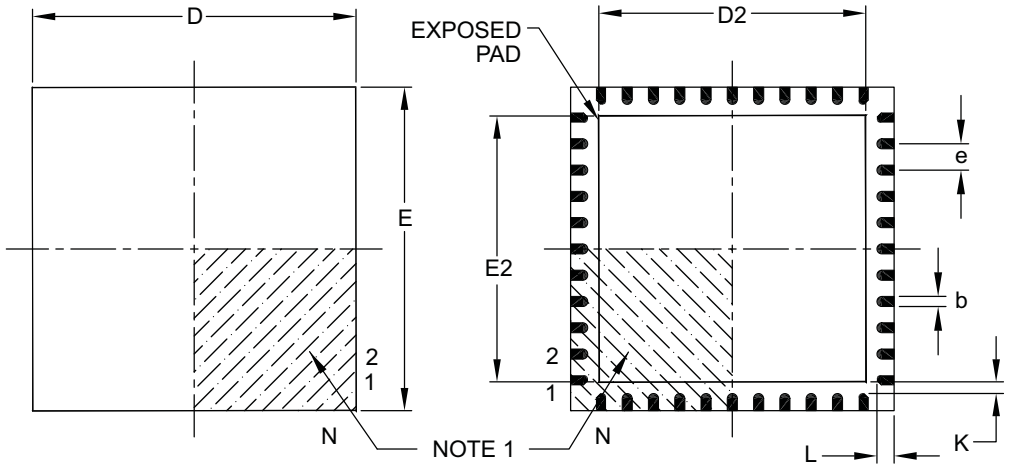
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

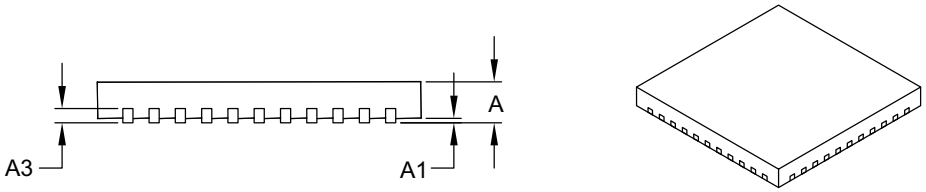
Microchip Technology Drawing No. C04-2076A

### 44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




**TOP VIEW** **BOTTOM VIEW**




| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 44 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Contact Thickness | A3 | | 0.20 REF | |
| Overall Width | E | | 8.00 BSC | |
| Exposed Pad Width | E2 | 6.30 | 6.45 | 6.80 |
| Overall Length | D | | 8.00 BSC | |
| Exposed Pad Length | D2 | 6.30 | 6.45 | 6.80 |
| Contact Width | b | 0.25 | 0.30 | 0.38 |
| Contact Length | L | 0.30 | 0.40 | 0.50 |
| Contact-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.

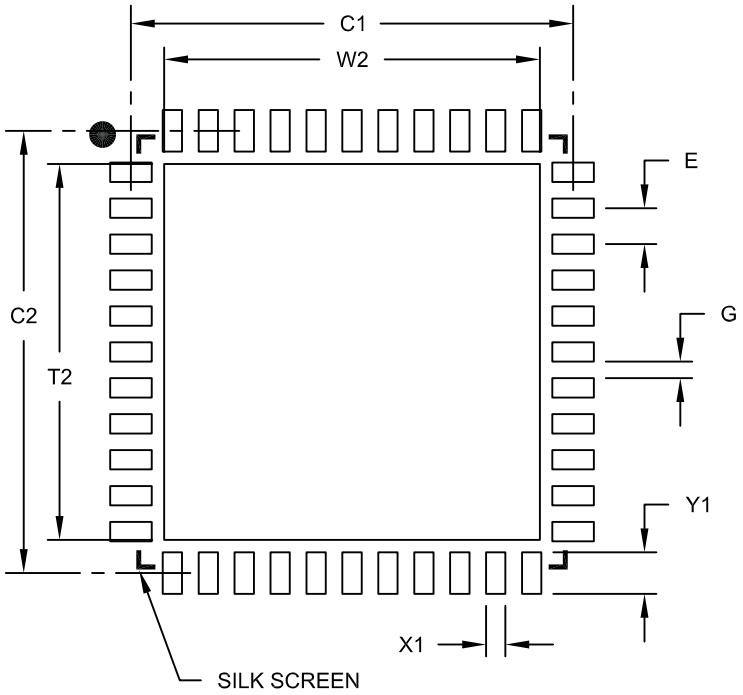
BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-103B


© 2009 Microchip Technology Inc.

### 44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

**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 | | | 6.80 |
| Optional Center Pad Length | T2 | | | 6.80 |
| Contact Pad Spacing | C1 | | 8.00 | |
| Contact Pad Spacing | C2 | | 8.00 | |
| Contact Pad Width (X44) | X1 | | | 0.35 |
| Contact Pad Length (X44) | Y1 | | | 0.80 |
| Distance Between Pads | G | 0.25 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2103A

**NOTES:**


© 2009 Microchip Technology Inc.

## APPENDIX A: REVISION HISTORY

### Revision A (July 2004)

Original data sheet for PIC18F2480/2580/4480/4580 devices.

### Revision B (August 2006)

Edits to Table 6-1 in **Section 6.0 "Memory Organization"** and trademarking updated.

### Revision C (March 2007)

Edits to Table 6-1 in **Section 6.0 "Memory Organization"**, pin name change in **Section 22.5 "Connection Considerations"**, updates to Section 27.3 "DC Characteristics", changes to SPI Mode Requirements in Figure 28-12 and Figure 28-13, and Table 28-14 through Table 28-17, and there have been minor updates to the data sheet text, including trademarking updates.

### Revision D (November 2009)

Removed Preliminary from the Condition tags. Various edits throughout the data sheet text.

## 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 | PIC18F2480 | PIC18F2580 | PIC18F4480 | PIC18F4580 |
|---|---|---|---|---|
| Program Memory (Bytes) | 16384 | 32768 | 16384 | 32768 |
| Program Memory (Instructions) | 8192 | 16384 | 8291 | 16384 |
| Interrupt Sources | 19 | 19 | 20 | 20 |
| I/O Ports | Ports A, B, C, (E) | Ports A, B, C, (E) | Ports A, B, C, D, E | Ports A, B, C, D, E |
| Capture/Compare/PWM Modules | 1 | 1 | 1 | 1 |
| Enhanced Capture/Compare/ PWM Modules | 0 | 0 | 1 | 1 |
| Parallel Communications (PSP) | No | No | Yes | Yes |
| 10-bit Analog-to-Digital Module | 8 input channels | 8 input channels | 11 input channels | 11 input channels |
| Packages | 28-pin SPDIP<br>28-pin SOIC<br>28-pin QFN | 28-pin SPDIP<br>28-pin SOIC<br>28-pin QFN | 40-pin PDIP<br>44-pin TQFP<br>44-pin QFN | 40-pin PDIP<br>44-pin TQFP<br>44-pin QFN |

## 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**


© 2009 Microchip Technology Inc.

## 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.

**NOTES:**


© 2009 Microchip Technology Inc.

# INDEX

## A


A/D ..... 253
A/D Converter Interrupt, Configuring ..... 257
Acquisition Requirements ..... 258
ADCON0 Register ..... 253
ADCON1 Register ..... 253
ADCON2 Register ..... 253
ADRESH Register ..... 253, 256
ADRESL Register ..... 253
Analog Port Pins, Configuring ..... 260
Associated Registers ..... 262
Automatic Acquisition Time ..... 259
Calculating the Minimum Required Acquisition Time ..... 258
Configuring the Module ..... 257
Conversion Clock (TAD) Selection ..... 259
Conversion Status (GO/DONE Bit) ..... 256
Conversions ..... 261
Converter Characteristics ..... 456
Operation in Power-Managed Modes ..... 260
Special Event Trigger (CCP) ..... 262
Special Event Trigger (ECCP) ..... 178
Use of the CCP1 Trigger ..... 262
Absolute Maximum Ratings ..... 421
AC (Timing) Characteristics ..... 438
Load Conditions for Device Timing Specifications ..... 439
Parameter Symbology ..... 438
Temperature and Voltage Specifications ..... 439
Timing Conditions ..... 439
Access Bank ..... 76
Mapping with Indexed Literal Offset Mode ..... 100
ACKSTAT ..... 221
ACKSTAT Status Flag ..... 221
ADCON0 Register ..... 253
GO/DONE Bit ..... 256
ADCON1 Register ..... 253
ADCON2 Register ..... 253
ADDFSR ..... 410
ADDLW ..... 373
ADDULNK ..... 410
ADDWF ..... 373
ADDWFC ..... 374
ADRESH Register ..... 253
ADRESL Register ..... 253, 256
Analog-to-Digital Converter. *See* A/D.
and BSR ..... 100
ANDLW ..... 374
ANDWF ..... 375
Assembler
MPASM Assembler ..... 418
Auto-Wake-up on Sync Break Character ..... 246


## B


Bank Select Register (BSR) ..... 73
Baud Rate Generator ..... 217
Baud Rate Generator (BRG) ..... 235
BC ..... 375
BCF ..... 376
BF ..... 221
BF Status Flag ..... 221
Bit Timing Configuration Registers
BRGCON1 ..... 344
BRGCON2 ..... 344
BRGCON3 ..... 344
Block Diagrams
A/D ..... 256
Analog Input Model ..... 257
Baud Rate Generator ..... 217
CAN Buffers and Protocol Engine ..... 280
Capture Mode Operation ..... 170
Comparator I/O Operating Modes ..... 264
Comparator Output ..... 266
Comparator Voltage Reference ..... 270
Comparator Voltage Reference Output Buffer Example ..... 271
Compare Mode Operation ..... 171
Device Clock ..... 34
Enhanced PWM ..... 179
EUSART Receive ..... 244
EUSART Transmit ..... 242
External Power-on Reset Circuit (Slow VDD Power-up) ..... 49
Fail-Safe Clock Monitor ..... 361
Generic I/O Port ..... 135
High/Low-Voltage Detect with External Input ..... 274
MSSP (I2C Master Mode) ..... 215
MSSP (I2C Mode) ..... 200
MSSP (SPI Mode) ..... 191
On-Chip Reset Circuit ..... 47
PIC18F2480/2580 ..... 12
PIC18F4480/4580 ..... 13
PLL (HS Mode) ..... 31
PORTD and PORTE (Parallel Slave Port) ..... 149
PWM Operation (Simplified) ..... 173
Reads From Flash Program Memory ..... 105
Single Comparator ..... 265
Table Read Operation ..... 101
Table Write Operation ..... 102
Table Writes to Flash Program Memory ..... 107
Timer0 in 16-Bit Mode ..... 152
Timer0 in 8-Bit Mode ..... 152
Timer1 ..... 156
Timer1 (16-Bit Read/Write Mode) ..... 156
Timer2 ..... 162
Timer3 ..... 164
Timer3 (16-Bit Read/Write Mode) ..... 164
Transmit Buffers ..... 334
Watchdog Timer ..... 358
Block Diagrams Comparator Analog Input Model ..... 267
BN ..... 376
BNC ..... 377
BNN ..... 377
BNOV ..... 378
BNZ ..... 378
BOR. *See* Brown-out Reset.
BOV ..... 381
BRA ..... 379
BRG. *See* Baud Rate Generator.
Brown-out Reset (BOR) ..... 50
Detecting ..... 50
Disabling in Sleep Mode ..... 50
Software Enabled ..... 50

BSF ..... 379
BTFSC ..... 380
BTFSS ..... 380
BTG ..... 381
BZ ..... 382

## C

C Compilers
MPLAB C18 ..... 418
CALL ..... 382
CALLW ..... 411
CAN Module
External-Internal Clock in HS-PLL Based Oscillators ..... 339
Capture (CCP Module) ..... 169
Associated Registers ..... 172
CCP1/ECCP1 Pin Configuration ..... 169
CCPR1H:CCPR1L Registers ..... 169
Software Interrupt ..... 169
Timer1/Timer3 Mode Selection ..... 169
Capture (ECCP Module) ..... 178
Capture/Compare/PWM (CCP) ..... 167
Capture Mode. *See* Capture.
CCP Mode and Timer Resources ..... 168
CCPRxH Register ..... 168
CCPRxL Register ..... 168
Compare Mode. *See* Compare.
Interaction Between CCP1 and ECCP1 for Timer Resources ..... 168
Module Configuration ..... 168
PWM Mode. *See* PWM.
Clock Sources ..... 34
Selecting the 31 kHz Source ..... 35
Selection Using OSCCON Register ..... 35
Clocking Scheme/Instruction Cycle ..... 71
CLRF ..... 383
CLRWDT ..... 383
Code Examples
16 x 16 Signed Multiply Routine ..... 118
16 x 16 Unsigned Multiply Routine ..... 118
8 x 8 Signed Multiply Routine ..... 117
8 x 8 Unsigned Multiply Routine ..... 117
Changing Between Capture Prescalers ..... 169
Changing to Configuration Mode ..... 284
Computed GOTO Using an Offset Value ..... 70
Data EEPROM Read ..... 113
Data EEPROM Refresh Routine ..... 114
Data EEPROM Write ..... 113
Erasing a Flash Program Memory Row ..... 106
Fast Register Stack ..... 70
How to Clear RAM (Bank 1) Using Indirect Addressing ..... 95
Implementing a Real-Time Clock Using a Timer1 Interrupt Service ..... 159
Initializing PORTA ..... 135
Initializing PORTB ..... 138
Initializing PORTC ..... 141
Initializing PORTD ..... 143
Initializing PORTE ..... 146
Loading the SSPBUF (SSPSR) Register ..... 194
Reading a CAN Message ..... 299
Reading a Flash Program Memory Word ..... 105
Saving STATUS, WREG and BSR Registers in RAM ..... 134
Transmitting a CAN Message Using Banked Method ..... 292
Transmitting a CAN Message Using WIN Bits ..... 292
WIN and ICODE Bits Usage in Interrupt Service Routine to Access TX/RX Buffers ..... 284
Writing to Flash Program Memory ..... 108–109
Code Protection ..... 349
COMF ..... 384
Comparator ..... 263
Analog Input Connection Considerations ..... 267
Associated Registers ..... 267
Configuration ..... 264
Effects of a Reset ..... 266
Interrupts ..... 266
Operation ..... 265
Operation During Sleep ..... 266
Outputs ..... 265
Reference ..... 265
External Signal ..... 265
Internal Signal ..... 265
Response Time ..... 265
Comparator Specifications ..... 436
Comparator Voltage Reference ..... 269
Accuracy and Error ..... 270
Associated Registers ..... 271
Configuring ..... 269
Connection Considerations ..... 270
Effects of a Reset ..... 270
Operation During Sleep ..... 270
Compare (CCP Module) ..... 171
Associated Registers ..... 172
CCP Pin Configuration ..... 171
CCPR1 Register ..... 171
Software Interrupt ..... 171
Special Event Trigger ..... 165, 171, 262
Timer1/Timer3 Mode Selection ..... 171
Compare (ECCP Module) ..... 178
Special Event Trigger ..... 178
Configuration Bits ..... 349
Configuration Mode ..... 330
Configuration Register Protection ..... 366
Context Saving During Interrupts ..... 134
Conversion Considerations ..... 472
CPFSEQ ..... 384
CPFSGT ..... 385
CPFSLT ..... 385
Crystal Oscillator/Ceramic Resonator ..... 29
Customer Change Notification Service ..... 486
Customer Notification Service ..... 486
Customer Support ..... 486

## D

Data Addressing Modes ..... 95
Comparing Addressing Modes with the Extended Instruction Set Enabled ..... 99
Direct ..... 95
Indexed Literal Offset ..... 98
Indirect ..... 95
Inherent and Literal ..... 95
Data EEPROM Code Protection ..... 366
Data EEPROM Memory ..... 111
Associated Registers ..... 115
EEADR Register ..... 111
EECON1 and EECON2 Registers ..... 111
Operation During Code-Protect ..... 114
Protection Against Spurious Write ..... 114
Reading ..... 113
Using ..... 114


© 2009 Microchip Technology Inc.

Write Verify ..... 113
Writing ..... 113
Data Memory ..... 73
Access Bank ..... 76
and the Extended Instruction Set ..... 98
Bank Select Register (BSR) ..... 73
General Purpose Registers ..... 76
Map for PIC18F2480/4480 ..... 74
Map for PIC18F2580/4580 ..... 75
Special Function Registers ..... 77
DAW ..... 386
DC Characteristics ..... 433
Power-Down and Supply Current ..... 424
Supply Voltage ..... 423
DCFSNZ ..... 387
DECF ..... 386
DECFSZ ..... 387
Development Support ..... 417
Device Differences ..... 471
Device Overview ..... 9
Features (table) ..... 11
Device Reset Timers ..... 51
Oscillator Start-up Timer (OST) ..... 51
PLL Lock Time-out ..... 51
Power-up Timer (PWRT) ..... 51
Direct Addressing ..... 96


## E


ECAN Module ..... 279
Baud Rate Setting ..... 338
Bit Time Partitioning ..... 338
Bit Timing Configuration Registers ..... 344
Calculating TQ, Nominal Bit Rate and Nominal Bit Time ..... 341
CAN Baud Rate Registers ..... 317
CAN Control and Status Registers ..... 281
CAN Controller Register Map ..... 325
CAN I/O Control Register ..... 320
CAN Interrupt Registers ..... 321
CAN Interrupts ..... 345
Acknowledge ..... 347
Bus Activity Wake-up ..... 346
Bus-Off ..... 347
Code Bits ..... 346
Error ..... 346
Message Error ..... 346
Receive ..... 346
Receiver Bus Passive ..... 347
Receiver Overflow ..... 347
Receiver Warning ..... 347
Transmit ..... 346
Transmitter Bus Passive ..... 347
Transmitter Warning ..... 347
CAN Message Buffers ..... 332
Dedicated Receive ..... 332
Dedicated Transmit ..... 332
Programmable Auto-RTR ..... 333
Programmable Transmit/Receive ..... 332
CAN Message Transmission ..... 333
Aborting ..... 333
Initiating ..... 333
Priority ..... 334
CAN Modes of Operation ..... 330
CAN Registers ..... 281
Configuration Mode ..... 330
Dedicated CAN Receive Buffer Registers ..... 293
Dedicated CAN Transmit Buffer Registers ..... 288
Disable Mode ..... 330
Error Detection ..... 344
Acknowledge ..... 344
Bit ..... 344
CRC ..... 344
Error Modes and Counters ..... 344
Error States ..... 344
Form ..... 344
Stuff Bit ..... 344
Error Modes State (diagram) ..... 345
Error Recognition Mode ..... 331
Filter-Mask Truth (table) ..... 336
Functional Modes ..... 331
Mode 0 (Legacy Mode) ..... 331
Mode 1 (Enhanced Legacy Mode) ..... 331
Mode 2 (Enhanced FIFO Mode) ..... 332
Information Processing Time (IPT) ..... 341
Lengthening a Bit Period ..... 343
Listen Only Mode ..... 331
Loopback Mode ..... 331
Message Acceptance Filters and Masks ..... 308, 336
Message Acceptance Mask and Filter Operation ..... 337
Message Reception ..... 335
Enhanced FIFO Mode ..... 336
Priority ..... 335
Time-Stamping ..... 336
Normal Mode ..... 330
Oscillator Tolerance ..... 343
Overview ..... 279
Phase Buffer Segments ..... 341
Programmable TX/RX and Auto-RTR Buffers ..... 300
Programming Time Segments ..... 343
Propagation Segment ..... 341
Sample Point ..... 341
Shortening a Bit Period ..... 343
Synchronization ..... 342
Hard ..... 342
Resynchronization ..... 342
Rules ..... 342
Synchronization Segment ..... 341
Time Quanta ..... 341
Values for ICODE (table) ..... 346
ECCP
Capture and Compare Modes ..... 178
Standard PWM Mode ..... 178
Effect on Standard Instructions ..... 98
Effect on Standard PIC Instructions ..... 414
Effects of Power-Managed Modes on Various Clock Sources ..... 37
Electrical Characteristics ..... 421
Enhanced Capture/Compare/PWM (ECCP) ..... 177
Capture Mode. *See* Capture (ECCP Module).
Outputs and Configuration ..... 178
Pin Configurations for ECCP Modes ..... 178
PWM Mode. *See* PWM (ECCP Module).
Timer Resources ..... 178
Enhanced PWM Mode. *See* PWM (ECCP Module). ..... 179
Enhanced Universal Synchronous Receiver Transmitter (EUSART). *See* EUSART.
Equations
A/D Acquisition Time ..... 258
A/D Minimum Charging Time ..... 258
Errata ..... 7

Error Recognition Mode ..... 330
EUSART
Asynchronous Mode ..... 241
Associated Registers, Receive ..... 245
Associated Registers, Transmit ..... 243
Auto-Wake-up on Sync Break ..... 246
Break Character Sequence ..... 247
Receiver ..... 244
Setting up 9-Bit Mode with Address Detect ..... 244
Transmitter ..... 241
Baud Rate Generator (BRG)
Associated Registers ..... 236
Auto-Baud Rate Detect ..... 239
Baud Rate Error, Calculating ..... 236
Baud Rates, Asynchronous Modes ..... 237
High Baud Rate Select (BRGH Bit) ..... 235
Operation in Power-Managed Mode ..... 235
Sampling ..... 235
Synchronous Master Mode ..... 248
Associated Registers, Receive ..... 250
Associated Registers, Transmit ..... 249
Reception ..... 250
Transmission ..... 248
Synchronous Slave Mode ..... 251
Associated Registers, Receive ..... 252
Associated Registers, Transmit ..... 251
Reception ..... 252
Transmission ..... 251
Extended Instruction Set
ADDFSR ..... 410
ADDULNK ..... 410
CALLW ..... 411
MOVSF ..... 411
MOVSS ..... 412
PUSHL ..... 412
SUBFSR ..... 413
SUBULNK ..... 413
External Clock Input ..... 30

## F

Fail-Safe Clock Monitor ..... 349, 361
Interrupts in Power-Managed Modes ..... 362
POR or Wake-up from Sleep ..... 362
WDT During Oscillator Failure ..... 361
Fast Register Stack ..... 70
Firmware Instructions ..... 367
Flash Program Memory ..... 101
Associated Registers ..... 109
Control Registers ..... 102
EECON1 and EECON2 ..... 102
TABLAT (Table Latch) Register ..... 104
TBLPTR (Table Pointer) Register ..... 104
Erase Sequence ..... 106
Erasing ..... 106
Operation During Code-Protect ..... 109
Reading ..... 105
Table Pointer
Boundaries Based on Operation ..... 104
Table Pointer Boundaries ..... 104
Table Reads and Table Writes ..... 101
Write Sequence ..... 107
Writing To ..... 107
Protection Against Spurious Writes ..... 109
Unexpected Termination ..... 109
Write Verify ..... 109
FSCM. *See* Fail-Safe Clock Monitor.

## G

GOTO ..... 388

## H

Hardware Multiplier
Introduction ..... 117
Operation ..... 117
Performance Comparison ..... 117
High/Low-Voltage Detect ..... 273
Applications ..... 276
Associated Registers ..... 277
Characteristics ..... 437
Current Consumption ..... 275
Effects of a Reset ..... 277
Operation ..... 274
During Sleep ..... 277
Setup ..... 275
Start-up Time ..... 275
Typical Application ..... 276
HLVD. *See* High/Low-Voltage Detect. ..... 273

## I

I/O Ports ..... 135
$I^2C$ Mode (MSSP)
Acknowledge Sequence Timing ..... 224
Baud Rate Generator ..... 217
Bus Collision
During a Repeated Start Condition ..... 228
During a Stop Condition ..... 229
Clock Arbitration ..... 218
Clock Stretching ..... 210
10-Bit Slave Receive Mode (SEN = 1) ..... 210
10-Bit Slave Transmit Mode ..... 210
7-Bit Slave Receive Mode (SEN = 1) ..... 210
7-Bit Slave Transmit Mode ..... 210
Clock Synchronization and the CKP Bit
(SEN = 1) ..... 211
Effect of a Reset ..... 225
General Call Address Support ..... 214
$I^2C$ Clock Rate w/BRG ..... 217
Master Mode ..... 215
Operation ..... 216
Reception ..... 221
Repeated Start Condition Timing ..... 220
Start Condition ..... 219
Transmission ..... 221
Transmit Sequence ..... 216
Multi-Master Communication, Bus Collision
and Arbitration ..... 225
Multi-Master Mode ..... 225
Operation ..... 204
Read/Write Bit Information (R/$\overline{W}$ Bit) ..... 204, 205
Registers ..... 200
Serial Clock (RC3/SCK/SCL) ..... 205
Slave Mode ..... 204
Addressing ..... 204
Reception ..... 205
Transmission ..... 205
Sleep Operation ..... 225
Stop Condition Timing ..... 224
ID Locations ..... 349, 366
INCF ..... 388
INCFSZ ..... 389
In-Circuit Debugger ..... 366
In-Circuit Serial Programming (ICSP) ..... 349, 366


© 2009 Microchip Technology Inc.

Indexed Literal Offset Addressing Mode ..... 414
and Standard PIC18 Instructions ..... 414
Indexed Literal Offset Mode ..... 98, 100
Indirect Addressing ..... 96
INFSNZ ..... 389
Initialization Conditions for all Registers ..... 55–66
Instruction Cycle ..... 71
Instruction Flow/Pipelining ..... 71
Instruction Set ..... 367
ADDLW ..... 373
ADDWF ..... 373
ADDWF (Indexed Literal Offset mode) ..... 415
ADDWFC ..... 374
ANDLW ..... 374
ANDWF ..... 375
BC ..... 375
BCF ..... 376
BN ..... 376
BNC ..... 377
BNN ..... 377
BNOV ..... 378
BNZ ..... 378
BOV ..... 381
BRA ..... 379
BSF ..... 379
BSF (Indexed Literal Offset mode) ..... 415
BTFSC ..... 380
BTFSS ..... 380
BTG ..... 381
BZ ..... 382
CALL ..... 382
CLRF ..... 383
CLRWDT ..... 383
COMF ..... 384
CPFSEQ ..... 384
CPFSGT ..... 385
CPFSLT ..... 385
DAW ..... 386
DCFSNZ ..... 387
DECF ..... 386
DECFSZ ..... 387
Extended Instructions ..... 409
and Using MPLAB Tools ..... 416
Considerations when Enabling ..... 414
Syntax ..... 409
General Format ..... 369
GOTO ..... 388
INCF ..... 388
INCFSZ ..... 389
INFSNZ ..... 389
IORLW ..... 390
IORWF ..... 390
LFSR ..... 391
MOVF ..... 391
MOVFF ..... 392
MOVLB ..... 392
MOVLW ..... 393
MOVWF ..... 393
MULLW ..... 394
MULWF ..... 394
NEGF ..... 395
NOP ..... 395
POP ..... 396
PUSH ..... 396
RCALL ..... 397
RESET ..... 397
RETFIE ..... 398
RETLW ..... 398
RETURN ..... 399
RLCF ..... 399
RLNCF ..... 400
RRCF ..... 400
RRNCF ..... 401
SETF ..... 401
SETF (Indexed Literal Offset mode) ..... 415
SLEEP ..... 402
Standard Instructions ..... 367
SUBFWB ..... 402
SUBLW ..... 403
SUBWF ..... 403
SUBWFB ..... 404
Summary Table ..... 370
SWAPF ..... 404
TBLRD ..... 405
TBLWT ..... 406
TSTFSZ ..... 407
XORLW ..... 407
XORWF ..... 408
INTCON Registers ..... 121
Inter-Integrated Circuit. *See* $I^2C$.
Internal Oscillator Block ..... 32
Adjustment ..... 32
INTIO Modes ..... 32
INTOSC Output Frequency ..... 32
OSCTUNE Register ..... 32
Internal RC Oscillator
Use with WDT ..... 358
Internet Address ..... 486
Interrupt Sources ..... 349
A/D Conversion Complete ..... 257
Capture Complete (CCP) ..... 169
Compare Complete (CCP) ..... 171
ECAN Module ..... 345
Interrupt-on-Change (RB7:RB4) ..... 138
INTx Pin ..... 134
PORTB, Interrupt-on-Change ..... 134
TMR0 ..... 134
TMR0 Overflow ..... 153
TMR1 Overflow ..... 155
TMR2 to PR2 Match (PWM) ..... 173, 179
TMR3 Overflow ..... 163, 165
Interrupts ..... 119
Logic (diagram) ..... 120
INTOSC Frequency Drift ..... 32
INTOSC, INTRC. *See* Internal Oscillator Block.
IORLW ..... 390
IORWF ..... 390
IPR Registers ..... 130

## L

LFSR ..... 391
Listen Only Mode ..... 330
Loopback Mode ..... 330
Low-Voltage ICSP Programming *See*
Single-Supply ICSP Programming.

## M

Master Clear Reset ($\overline{MCLR}$) ..... 49
Master Synchronous Serial Port (MSSP). *See* MSSP.

Memory Organization ..... 67
Data Memory ..... 73
Program Memory ..... 67
Memory Programming Requirements ..... 435
Microchip Internet Web Site ..... 486
Migration from Baseline to Enhanced Devices ..... 472
Migration from High-End to Enhanced Devices ..... 473
Migration from Mid-Range to Enhanced Devices ..... 473
MOVF ..... 391
MOVFF ..... 392
MOVLB ..... 392
MOVLW ..... 393
MOVSF ..... 411
MOVSS ..... 412
MOVWF ..... 393
MPLAB ASM30 Assembler, Linker, Librarian ..... 418
MPLAB Integrated Development Environment Software ..... 417
MPLAB PM3 Device Programmer ..... 420
MPLAB REAL ICE In-Circuit Emulator System ..... 419
MPLINK Object Linker/MPLIB Object Librarian ..... 418
MSSP
ACK Pulse ..... 204, 205
Control Registers (general) ..... 191
I2C Mode. *See* I2C Mode.
Module Overview ..... 191
SPI Master/Slave Connection ..... 195
SPI Mode. *See* SPI Mode.
SSPBUF ..... 196
SSPSR ..... 196
MULLW ..... 394
MULWF ..... 394

**N**

NEGF ..... 395
NOP ..... 395
Normal Operation Mode ..... 330

**O**

Opcode Field Descriptions ..... 368
Oscillator Configuration ..... 29
EC ..... 29
ECIO ..... 29
HS ..... 29
HSPLL ..... 29
Internal Oscillator Block ..... 32
INTIO1 ..... 29
INTIO2 ..... 29
LP ..... 29
RC ..... 29
RCIO ..... 29
XT ..... 29
Oscillator Selection ..... 349
Oscillator Start-up Timer (OST) ..... 37, 51
Oscillator Switching ..... 34
Oscillator Transitions ..... 35
Oscillator, Timer1 ..... 155, 165
Oscillator, Timer3 ..... 163

**P**

Packaging Information ..... 459
Details ..... 461
Marking ..... 459
Parallel Slave Port (PSP) ..... 143, 149
Associated Registers ..... 150
CS (Chip Select) ..... 149
PORTD ..... 149
RD (Read Input) ..... 149
Select (PSPMODE Bit) ..... 143, 149
WR (Write Input) ..... 149
PIE Registers ..... 127
Pin Functions
MCLR/VPP/RE3 ..... 14, 18
OSC1/CLKI/RA7 ..... 14, 18
OSC2/CLKO/RA6 ..... 14, 18
RA0/AN0 ..... 15
RA0/AN0/CVREF ..... 19
RA1/AN1 ..... 15, 19
RA2/AN2/VREF- ..... 15, 19
RA3/AN3/VREF+ ..... 15, 19
RA4/T0CKI ..... 15, 19
RA5/AN4/SS/HLVDIN ..... 15, 19
RB0/INT0/AN10 ..... 16
RB0/INT0/FLT0/AN10 ..... 20
RB1/INT1/AN8 ..... 16, 20
RB2/INT2/CANTX ..... 16, 20
RB3/CANRX ..... 16, 20
RB4/KBI0/AN9 ..... 16, 20
RB5/KBI1/PGM ..... 16, 20
RB6/KBI2/PGC ..... 16, 20
RB7/KBI3/PGD ..... 16, 20
RC0/T1OSO/T13CKI ..... 17, 21
RC1/T1OSI ..... 17, 21
RC2/CCP1 ..... 17, 21
RC3/SCK/SCL ..... 17, 21
RC4/SDI/SDA ..... 17, 21
RC5/SDO ..... 17, 21
RC6/TX/CK ..... 17, 21
RC7/RX/DT ..... 17, 21
RD0/PSP0/C1IN+ ..... 22
RD1/PSP1/C1IN- ..... 22
RD2/PSP2/C2IN+ ..... 22
RD3/PSP3/C2IN- ..... 22
RD4/PSP4/ECCP1/P1A ..... 22
RD5/PSP5/P1B ..... 22
RD6/PSP6/P1C ..... 22
RD7/PSP7/P1D ..... 22
RE0/RD/AN5 ..... 23
RE1/WR/AN6/C1OUT ..... 23
RE2/CS/AN7/C2OUT ..... 23
VDD ..... 17, 23
VSS ..... 17, 23
Pinout I/O Descriptions
PIC18F2480/2580 ..... 14
PIC18F4480/4580 ..... 18
PIR Registers ..... 124
PLL
Use with INTOSC ..... 31
PLL Frequency Multiplier ..... 31
HSPLL Oscillator Mode ..... 31
In INTOSC Modes ..... 32
PLL Lock Time-out ..... 51
POP ..... 396
POR. *See* Power-on Reset.
PORTA
Associated Registers ..... 137
I/O Summary ..... 136
LATA Register ..... 135
PORTA Register ..... 135
TRISA Register ..... 135



© 2009 Microchip Technology Inc.

PORTB
Associated Registers ... 140
I/O Summary ... 139
LATB Register ... 138
PORTB Register ... 138
TRISB Register ... 138
PORTC
Associated Registers ... 142
I/O Summary ... 142
LATC Register ... 141
PORTC Register ... 141
RC3/SCK/SCL Pin ... 205
TRISC Register ... 141
PORTD
Associated Registers ... 145
I/O Summary ... 144
LATD Register ... 143
Parallel Slave Port (PSP) Function ... 143
PORTD Register ... 143
TRISD Register ... 143
PORTE
Associated Registers ... 148
I/O Summary ... 148
LATE Register ... 146
PORTE Register ... 146
PSP Mode Select (PSPMODE Bit) ... 143
TRISE Register ... 146
Postscaler, WDT
Assignment (PSA Bit) ... 153
Rate Select (T0PS2:T0PS0 Bits) ... 153
Switching Between Timer0 and WDT ... 153
Power-Managed Modes ... 39
and A/D Operation ... 260
and EUSART Operation ... 235
and Multiple Sleep Commands ... 40
Clock Transitions and Status Indicators ... 40
Effects on Clock Sources ... 37
Entering ... 39
Exiting Idle and Sleep Modes ... 45
by Interrupt ... 45
by Reset ... 45
by WDT Time-out ... 45
Without an Oscillator Start-up Delay ... 46
Idle Modes ... 43
PRI_IDLE ... 44
RC_IDLE ... 45
SEC_IDLE ... 44
Run Modes ... 40
PRI_RUN ... 40
RC_RUN ... 41
SEC_RUN ... 40
Selecting ... 39
Sleep Mode ... 43
Summary (table) ... 39
Power-on Reset (POR) ... 49
Oscillator Start-up Timer (OST) ... 51
Power-up Timer (PWRT) ... 51
Time-out Sequence ... 51
Power-up Delays ... 37
Power-up Timer (PWRT) ... 37, 51
Prescaler
Timer2 ... 180
Prescaler, Capture ... 169
Prescaler, Timer0 ... 153
Assignment (PSA Bit) ... 153
Rate Select (T0PS2:T0PS0 Bits) ... 153
Switching Between Timer0 and WDT ... 153
Prescaler, Timer2 ... 174
PRI_IDLE Mode ... 44
PRI_RUN Mode ... 40
Program Counter ... 68
PCL, PCH and PCU Registers ... 68
PCLATH and PCLATU Registers ... 68
Program Memory
and the Extended Instruction Set ... 98
Code Protection ... 364
Instructions ... 72
Two-Word ... 72
Interrupt Vector ... 67
Look-up Tables ... 70
Map and Stack (diagram) ... 67
Reset Vector ... 67
Program Verification and Code Protection ... 363
Associated Registers ... 364
Programming, Device Instructions ... 367
PSP. *See* Parallel Slave Port.
Pulse-Width Modulation. *See* PWM (CCP Module) and PWM (ECCP Module).
PUSH ... 396
PUSH and POP Instructions ... 69
PUSHL ... 412
PWM (CCP Module) ... 173
Associated Registers ... 175
Auto-Shutdown ... 174
CCPR1H:CCPR1L Registers ... 173
Duty Cycle ... 173
Example Frequencies/Resolutions ... 174
Period ... 173
Setup for PWM Operation ... 174
TMR2 to PR2 Match ... 173
PWM (ECCP Module) ... 179
Associated Registers ... 190
Direction Change in Full-Bridge Output Mode ... 184
Duty Cycle ... 180
ECCPR1H:ECCPR1L Registers ... 179
Effects of a Reset ... 189
Enhanced PWM Auto-Shutdown ... 186
Example Frequencies/Resolutions ... 180
Full-Bridge Application Example ... 184
Full-Bridge Mode ... 183
Half-Bridge Mode ... 182
Half-Bridge Output Mode Applications Example ... 182
Output Configurations ... 180
Output Relationships (Active-High) ... 181
Output Relationships (Active-Low) ... 181
Period ... 179
Programmable Dead-Band Delay ... 186
Start-up Considerations ... 188
TMR2 to PR2 Match ... 179

## Q

Q Clock ... 174, 180

## R

RAM. *See* Data Memory.
RC Oscillator ... 31
RCIO Oscillator Mode ... 31

RC_IDLE Mode ..... 45
RC_RUN Mode ..... 41
RCALL ..... 397
RCON Register
Bit Status During Initialization ..... 54
Reader Response ..... 487
Receiver Warning ..... 347
Register File ..... 76
Register File Summary ..... 83–93
Registers
ADCON0 (A/D Control 0) ..... 253
ADCON1 (A/D Control 1) ..... 254
ADCON2 (A/D Control 2) ..... 255
BAUDCON (Baud Rate Control) ..... 234
BIE0 (Buffer Interrupt Enable 0) ..... 324
BnCON (TX/RX Buffer n Control, Receive Mode) ..... 300
BnCON (TX/RX Buffer n Control, Transmit Mode) ..... 301
BnDLC (TX/RX Buffer n Data Length Code in Receive Mode) ..... 306
BnDLC (TX/RX Buffer n Data Length Code in Transmit Mode) ..... 307
BnDm (TX/RX Buffer n Data Field Byte m in Receive Mode) ..... 305
BnDm (TX/RX Buffer n Data Field Byte m in Transmit Mode) ..... 305
BnEIDH (TX/RX Buffer n Extended Identifier, High Byte in Receive Mode) ..... 304
BnEIDH (TX/RX Buffer n Extended Identifier, High Byte in Transmit Mode) ..... 304
BnEIDL (TX/RX Buffer n Extended Identifier, Low Byte in Receive Mode) ..... 304, 305
BnSIDH (TX/RX Buffer n Standard Identifier, High Byte in Receive Mode) ..... 302
BnSIDH (TX/RX Buffer n Standard Identifier, High Byte in Transmit Mode) ..... 302
BnSIDL (TX/RX Buffer n Standard Identifier, Low Byte in Receive Mode) ..... 303
BRGCON1 (Baud Rate Control 1) ..... 317
BRGCON2 (Baud Rate Control 2) ..... 318
BRGCON3 (Baud Rate Control 3) ..... 319
BSEL0 (Buffer Select 0) ..... 307
CANCON (CAN Control) ..... 282
CANSTAT (CAN Status) ..... 283
CCP1CON (Capture/Compare/PWM Control) ..... 167
CIOCON (CAN I/O Control) ..... 320
CMCON (Comparator Control) ..... 263
COMSTAT (CAN Communication Status) ..... 287
CONFIG1H (Configuration 1 High) ..... 350
CONFIG2H (Configuration 2 High) ..... 352
CONFIG2L (Configuration 2 Low) ..... 351
CONFIG3H (Configuration 3 High) ..... 353
CONFIG4L (Configuration 4 Low) ..... 353
CONFIG5H (Configuration 5 High) ..... 354
CONFIG5L (Configuration 5 Low) ..... 354
CONFIG6H (Configuration 6 High) ..... 355
CONFIG6L (Configuration 6 Low) ..... 355
CONFIG7H (Configuration 7 High) ..... 356
CONFIG7L (Configuration 7 Low) ..... 356
CVRCON (Comparator Voltage Reference Control) ..... 269
DEVID1 (Device ID 1) ..... 357
DEVID2 (Device ID 2) ..... 357
ECANCON (Enhanced CAN Control) ..... 286
ECCP1AS (ECCP Auto-Shutdown Control) ..... 187
ECCP1CON (Enhanced Capture/Compare/PWM Control) ..... 177
ECCP1DEL (ECCP PWM Dead-Band Delay) ..... 187
EECON1 (Data EEPROM Control 1) ..... 103, 112
HLVDCON (HLVD Control) ..... 273
INTCON (Interrupt Control) ..... 121
INTCON2 (Interrupt Control 2) ..... 122
INTCON3 (Interrupt Control 3) ..... 123
IPR1 (Peripheral Interrupt Priority 1) ..... 130
IPR2 (Peripheral Interrupt Priority 2) ..... 131
IPR3 (Peripheral Interrupt Priority 3) ..... 132, 323
MSEL0 (Mask Select 0) ..... 313
MSEL1 (Mask Select 1) ..... 314
MSEL2 (Mask Select 2) ..... 315
MSEL3 (Mask Select 3) ..... 316
OSCCON (Oscillator Control) ..... 36
OSCTUNE (Oscillator Tuning) ..... 33
PIE1 (Peripheral Interrupt Enable 1) ..... 127
PIE2 (Peripheral Interrupt Enable 2) ..... 128
PIE3 (Peripheral Interrupt Enable 3) ..... 129, 322
PIR1 (Peripheral Interrupt Request (Flag) 1) ..... 124
PIR2 (Peripheral Interrupt Request (Flag) 2) ..... 125
PIR3 (Peripheral Interrupt Request (Flag) 3) ... 126, 321
RCON (Reset Control) ..... 48, 133
RCSTA (Receive Status and Control) ..... 233
RXB0CON (Receive Buffer 0 Control) ..... 293
RXB1CON (Receive Buffer 1 Control) ..... 295
RXBnDLC (Receive Buffer n Data Length Code) ..... 298
RXBnDm (Receive Buffer n Data Field Byte m) ..... 298
RXBnEIDH (Receive Buffer n Extended Identifier, High Byte) ..... 297
RXBnEIDL (Receive Buffer n Extended Identifier, Low Byte) ..... 297
RXBnSIDH (Receive Buffer n Standard Identifier, High Byte) ..... 296
RXBnSIDL (Receive Buffer n Standard Identifier, Low Byte) ..... 297
RXERRCNT (Receive Error Count) ..... 299
RXFBCONn (Receive Filter Buffer Control n) ..... 312
RXFCONn (Receive Filter Control n) ..... 311
RXFnEIDH (Receive Acceptance Filter n Extended Identifier, High Byte) ..... 309
RXFnEIDL (Receive Acceptance Filter n Extended Identifier, Low Byte) ..... 309
RXFnSIDH (Receive Acceptance Filter n Standard Identifier Filter, High Byte) ..... 308
RXFnSIDL (Receive Acceptance Filter n Standard Identifier Filter, Low Byte) ..... 308
RXMnEIDH (Receive Acceptance Mask n Extended Identifier Mask, High Byte) ..... 310
RXMnEIDL (Receive Acceptance Mask n Extended Identifier Mask, Low Byte) ..... 310
RXMnSIDH (Receive Acceptance Mask n Standard Identifier Mask, High Byte) ..... 309
RXMnSIDL (Receive Acceptance Mask n Standard Identifier Mask, Low Byte) ..... 310
SDFLC (Standard Data Bytes Filter Length Count) ..... 311
SSPCON1 (MSSP Control 1, $I^2C$ Mode) ..... 202
SSPCON1 (MSSP Control 1, SPI Mode) ..... 193
SSPCON2 (MSSP Control 2, $I^2C$ Mode) ..... 203
SSPSTAT (MSSP Status, $I^2C$ Mode) ..... 201
SSPSTAT (MSSP Status, SPI Mode) ..... 192


© 2009 Microchip Technology Inc.

STATUS ..... 94
STKPTR (Stack Pointer) ..... 69
T0CON (Timer0 Control) ..... 151
T1CON (Timer 1 Control) ..... 155
T2CON (Timer2 Control) ..... 161
T3CON (Timer3 Control) ..... 163
TRISE (PORTE/PSP Control) ..... 147
TXBIE (Transmit Buffers Interrupt Enable) ..... 324
TXBnCON (Transmit Buffer n Control) ..... 288
TXBnDLC (Transmit Buffer n Data Length Code) ..... 291
TXBnDm (Transmit Buffer n Data Field Byte m) ..... 290
TXBnEIDH (Transmit Buffer n Extended Identifier, High Byte) ..... 289
TXBnEIDL (Transmit Buffer n Extended Identifier, Low Byte) ..... 290
TXBnSIDH (Transmit Buffer n Standard Identifier, High Byte) ..... 289
TXBnSIDL (Transmit Buffer n Standard Identifier, Low Byte) ..... 289
TXERRCNT (Transmit Error Count) ..... 291
TXSTA (Transmit Status and Control) ..... 232
WDTCON (Watchdog Timer Control) ..... 359
RESET ..... 397
Resets ..... 47, 349
Brown-out Reset (BOR) ..... 349
Oscillator Start-up Timer (OST) ..... 349
Power-on Reset (POR) ..... 349
Power-up Timer (PWRT) ..... 349
RETFIE ..... 398
RETLW ..... 398
RETURN ..... 399
Return Address Stack ..... 68
and Associated Registers ..... 68
Return Stack Pointer (STKPTR) ..... 69
Revision History ..... 471
RLCF ..... 399
RLNCF ..... 400
RRCF ..... 400
RRNCF ..... 401


## S


SCK ..... 191
SDI ..... 191
SDO ..... 191
SEC_IDLE Mode ..... 44
SEC_RUN Mode ..... 40
Serial Clock, SCK ..... 191
Serial Data In (SDI) ..... 191
Serial Data Out (SDO) ..... 191
Serial Peripheral Interface. *See* SPI Mode.
SETF ..... 401
Slave Select ($\overline{SS}$) ..... 191
SLEEP ..... 402
Sleep
OSC1 and OSC2 Pin States ..... 37
Software Simulator (MPLAB SIM) ..... 419
Special Event Trigger. *See* Compare (ECCP Mode).
Special Event Trigger. *See* Compare (ECCP Module).
Special Features of the CPU ..... 349
Special Function Registers ..... 77
Map ..... 77–82
SPI Mode (MSSP)
Associated Registers ..... 199
Bus Mode Compatibility ..... 199
Effects of a Reset ..... 199
Enabling SPI I/O ..... 195
Master Mode ..... 196
Master/Slave Connection ..... 195
Operation ..... 194
Operation in Power-Managed Modes ..... 199
Serial Clock ..... 191
Serial Data In ..... 191
Serial Data Out ..... 191
Slave Mode ..... 197
Slave Select ..... 191
Slave Select Synchronization ..... 197
SPI Clock ..... 196
Typical Connection ..... 195
$\overline{SS}$ ..... 191
SSPOV ..... 221
SSPOV Status Flag ..... 221
SSPSTAT Register
R/$\overline{W}$ Bit ..... 204, 205
Stack Full/Underflow Resets ..... 70
STATUS Register ..... 94
SUBFSR ..... 413
SUBFWB ..... 402
SUBLW ..... 403
SUBULNK ..... 413
SUBWF ..... 403
SUBWFB ..... 404
SWAPF ..... 404


## T


T0CON Register
PSA Bit ..... 153
T0CS Bit ..... 152
T0PS2:T0PS0 Bits ..... 153
T0SE Bit ..... 152
Table Pointer Operations (table) ..... 104
Table Reads/Table Writes ..... 70
TBLRD ..... 405
TBLWT ..... 406
Time-out in Various Situations (table) ..... 51
Timer0 ..... 151
16-Bit Mode Reads and Writes ..... 152
Associated Registers ..... 153
Clock Source Edge Select (T0SE Bit) ..... 152
Clock Source Select (T0CS Bit) ..... 152
Operation ..... 152
Overflow Interrupt ..... 153
Prescaler. *See* Prescaler, Timer0.
Timer1 ..... 155
16-Bit Read/Write Mode ..... 157
Associated Registers ..... 159
Interrupt ..... 158
Operation ..... 156
Oscillator ..... 155, 157
Oscillator Layout Considerations ..... 158
Overflow Interrupt ..... 155
Resetting, Using a Special Event Trigger Output (CCP) ..... 158
Special Event Trigger (ECCP) ..... 178
TMR1H Register ..... 155
TMR1L Register ..... 155
Use as a Real-Time Clock ..... 158
Timer2 ..... 161
Associated Registers ..... 162
Interrupt ..... 162
Operation ..... 161
Output ..... 162

PR2 Register ..... 173, 179
TMR2 to PR2 Match Interrupt ..... 173, 179
Timer3 ..... 163
16-Bit Read/Write Mode ..... 165
Associated Registers ..... 165, 172
Operation ..... 164
Oscillator ..... 163, 165
Overflow Interrupt ..... 163, 165
Special Event Trigger (CCP) ..... 165
TMR3H Register ..... 163
TMR3L Register ..... 163
Timing Diagrams
A/D Conversion ..... 457
Acknowledge Sequence ..... 224
Asynchronous Reception ..... 245
Asynchronous Transmission ..... 242
Asynchronous Transmission (Back-to-Back) ..... 242
Automatic Baud Rate Calculation ..... 240
Auto-Wake-up Bit (WUE) During Normal Operation ..... 246
Auto-Wake-up Bit (WUE) During Sleep ..... 246
Baud Rate Generator with Clock Arbitration ..... 218
BRG Overflow Sequence ..... 240
BRG Reset Due to SDA Arbitration During Start Condition ..... 227
Brown-out Reset (BOR) ..... 443
Bus Collision During a Repeated Start Condition (Case 1) ..... 228
Bus Collision During a Repeated Start Condition (Case 2) ..... 228
Bus Collision During a Start Condition (SCL = 0) ..... 227
Bus Collision During a Start Condition (SDA only) ..... 226
Bus Collision During a Stop Condition (Case 1) ..... 229
Bus Collision During a Stop Condition (Case 2) ..... 229
Bus Collision for Transmit and Acknowledge ..... 225
Capture/Compare/PWM (CCP) ..... 445
CLKO and I/O ..... 442
Clock Synchronization ..... 211
EUSART Synchronous Receive (Master/Slave) ..... 455
EUSART Synchronous Transmission (Master/Slave) ..... 455
Example SPI Master Mode (CKE = 0) ..... 447
Example SPI Master Mode (CKE = 1) ..... 448
Example SPI Slave Mode (CKE = 0) ..... 449
Example SPI Slave Mode (CKE = 1) ..... 450
External Clock (All Modes except PLL) ..... 440
Fail-Safe Clock Monitor ..... 362
First Start Bit Timing ..... 219
Full-Bridge PWM Output ..... 183
Half-Bridge PWM Output ..... 182
High-Voltage Detect (VDIRMAG = 1) ..... 276
I2C Bus Data ..... 451
I2C Bus Start/Stop Bits ..... 451
I2C Master Mode (7 or 10-Bit Transmission) ..... 222
I2C Master Mode (7-Bit Reception) ..... 223
I2C Slave Mode (10-Bit Reception, SEN = 0) ..... 208
I2C Slave Mode (10-Bit Reception, SEN = 1) ..... 213
I2C Slave Mode (10-Bit Transmission) ..... 209
I2C Slave Mode (7-Bit Reception, SEN = 0) ..... 206
I2C Slave Mode (7-Bit Reception, SEN = 1) ..... 212
I2C Slave Mode (7-Bit Transmission) ..... 207
I2C Slave Mode General Call Address Sequence (7 or 10-Bit Address Mode) ..... 214
Low-Voltage Detect (VDIRMAG = 0) ..... 275
Master SSP I2C Bus Data ..... 453
Master SSP I2C Bus Start/Stop Bits ..... 453
Parallel Slave Port (PIC18F4480/4580) ..... 446
Parallel Slave Port (PSP) Read ..... 150
Parallel Slave Port (PSP) Write ..... 150
PWM Auto-Shutdown (PRSEN = 0, Auto-Restart Disabled) ..... 188
PWM Auto-Shutdown (PRSEN = 1, Auto-Restart Enabled) ..... 188
PWM Direction Change ..... 185
PWM Direction Change at Near 100% Duty Cycle ..... 185
PWM Output ..... 173
Repeat Start Condition ..... 220
Reset, Watchdog Timer (WDT), Oscillator Start-up Timer (OST) and Power-up Timer (PWRT) ..... 443
Send Break Character Sequence ..... 247
Slave Synchronization ..... 197
Slow Rise Time (MCLR Tied to VDD, VDD Rise > TPWRT) ..... 53
SPI Mode (Master Mode) ..... 196
SPI Mode (Slave Mode with CKE = 0) ..... 198
SPI Mode (Slave Mode with CKE = 1) ..... 198
Stop Condition Receive or Transmit Mode ..... 224
Synchronous Reception (Master Mode, SREN) ..... 250
Synchronous Transmission ..... 248
Synchronous Transmission (Through TXEN) ..... 249
Time-out Sequence on POR w/ PLL Enabled (MCLR Tied to VDD) ..... 53
Time-out Sequence on Power-up (MCLR Not Tied to VDD), Case 1 ..... 52
Time-out Sequence on Power-up (MCLR Not Tied to VDD), Case 2 ..... 52
Time-out Sequence on Power-up (MCLR Tied to VDD, VDD Rise Tpwrt) ..... 52
Timer0 and Timer1 External Clock ..... 444
Transition for Entry to Idle Mode ..... 44
Transition for Entry to SEC_RUN Mode ..... 41
Transition for Entry to Sleep Mode ..... 43
Transition for Two-Speed Start-up (INTOSC to HSPLL) ..... 360
Transition for Wake From Idle to Run Mode ..... 44
Transition for Wake From Sleep (HSPLL) ..... 43
Transition From RC_RUN Mode to PRI_RUN Mode ..... 42
Transition From SEC_RUN Mode to PRI_RUN Mode (HSPLL) ..... 41
Transition to RC_RUN Mode ..... 42
Timing Diagrams and Specifications ..... 440
A/D Conversion Requirements ..... 457
AC Characteristics Internal RC Accuracy ..... 441
Capture/Compare/PWM Requirements ..... 445
CLKO and I/O Requirements ..... 442
EUSART Synchronous Receive Requirements ..... 455
EUSART Synchronous Transmission Requirements ..... 455



© 2009 Microchip Technology Inc.

Example SPI Mode Requirements
Master Mode, CKE = 0 ..... 447
Master Mode, CKE = 1 ..... 448
Slave Mode, CKE = 0 ..... 449
Slave Mode, CKE = 1 ..... 450
External Clock Requirements ..... 440
High/Low-Voltage Detect Characteristics ..... 437
$I^2C$ Bus Data Requirements (Slave Mode) ..... 452
Master SSP $I^2C$ Bus Data Requirements ..... 454
Master SSP $I^2C$ Bus Start/Stop Bits Requirements ..... 453
Parallel Slave Port Requirements (PIC18F4480/4580) ..... 446
PLL Clock ..... 441
Reset, Watchdog Timer, Oscillator Start-up Timer, Power-up Timer and Brown-out Reset Requirements ..... 443
Timer0 and Timer1 External Clock Requirements ..... 444
Top-of-Stack Access ..... 68
TRISE Register
PSPMODE Bit ..... 143
TSTFSZ ..... 407
Two-Speed Start-up ..... 349, 360
Two-Word Instructions
Example Cases ..... 72
TXSTA Register
BRGH Bit ..... 235

**V**

Voltage Reference Specifications ..... 436

**W**

Watchdog Timer (WDT) ..... 349, 358
Associated Registers ..... 359
Control Register ..... 358
During Oscillator Failure ..... 361
Programming Considerations ..... 358
WCOL ..... 219, 220, 221, 224
WCOL Status Flag ..... 219, 220, 221, 224
WWW Address ..... 486
WWW, On-Line Support ..... 7

**X**

XORLW ..... 407
XORWF ..... 408

**NOTES:**


© 2009 Microchip Technology Inc.

## THE MICROCHIP WEB SITE

Microchip provides online support via our WWW 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, 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
- Development Systems Information Line

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://support.microchip.com**

## READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

To: Technical Publications Manager Total Pages Sent _____

RE: Reader Response

From: Name _____

Company _____

Address _____

City / State / ZIP / Country _____

Telephone: (_____) _____ - _____ FAX: (_____) _____ - _____

Application (optional):

Would you like a reply? ___Y ___N

Device: PIC18F2480/2580/4480/4580 Literature Number: DS39637D

Questions:

1. What are the best features of this document?

2. How does this document meet your hardware and software development needs?

3. Do you find the organization of this document easy to follow? If not, why?

4. What additions to the document do you think would enhance the structure and subject?

5. What deletions from the document could be made without affecting the overall usefulness?

6. Is there any incorrect or misleading information (what and where)?

7. How would you improve this document?


© 2009 Microchip Technology Inc.

## PIC18F2480/2580/4480/4580 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 | PIC18F2480/2580[1], PIC18F4480/4580 [1], PIC18F2480/2580T [2], PIC18F4480/4580T [2]; VDD range 4.2V to 5.5V<br>PIC18LF2480/2580[1], PIC18LF4480/4580[1], PIC18LF2480/25800T[2], PIC18LF4480/4580T[2]; VDD range 2.0V to 5.5V |
| Temperature Range | I = -40°C to +85°C (Industrial)<br>E = -40°C to +125°C (Extended) |
| Package | PT = TQFP (Thin Quad Flatpack)<br>SO = SOIC<br>SP = SPDIP (Skinny Plastic DIP)<br>P = PDIP<br>ML = QFN |
| Pattern | QTP, SQTP, Code or Special Requirements (blank otherwise) |

**Examples:**

a) PIC18F4580-I/P 301 = Industrial temp., PDIP package, Extended VDD limits, QTP pattern #301.

b) PIC18F2580-I/SO = Industrial temp., SOIC package, Extended VDD limits.

c) PIC18F45805-I/P = Industrial temp., PDIP package, normal VDD limits.

**Note 1:** F = Standard Voltage Range
LF = Wide Voltage Range
**2:** T = in tape and reel

# 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://support.microchip.com
Web Address:
www.microchip.com

**Atlanta**
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

**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**
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

**Kokomo**
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

**Santa Clara**
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

**Toronto**
Mississauga, Ontario,
Canada
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: 852-2401-1200
Fax: 852-2401-3431

**Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**China - Beijing**
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

**China - Chengdu**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**China - Hong Kong SAR**
Tel: 852-2401-1200
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: 86-755-8203-2660
Fax: 86-755-8203-1760

**China - Wuhan**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**China - Xiamen**
Tel: 86-592-2388138
Fax: 86-592-2388130

**China - Xian**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

**China - Zhuhai**
Tel: 86-756-3210040
Fax: 86-756-3210049

## ASIA/PACIFIC

**India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4080

**India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**India - Pune**
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

**Japan - Yokohama**
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

**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: 886-3-6578-300
Fax: 886-3-6578-370

**Taiwan - Kaohsiung**
Tel: 886-7-536-4818
Fax: 886-7-536-4803

**Taiwan - Taipei**
Tel: 886-2-2500-6610
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 - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**UK - Wokingham**
Tel: 44-118-921-5869
Fax: 44-118-921-5820

03/26/09


© 2009 Microchip Technology Inc.

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View PIC18F2580-E/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