



**MICROCHIP**

---

**PIC18F6310/6410/8310/8410**  
**Data Sheet**

64/80-Pin Flash Microcontrollers  
with nanoWatt XLP Technology

---

**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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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, PIC<sup>32</sup> logo, rPIC 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, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICKit, PICtail, REAL ICE, rLAB, 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.

© 2010, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-60932-582-4

*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<sup>®</sup> MCUs and dsPIC<sup>®</sup> DSCs, KEELOQ<sup>®</sup> 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.*

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



# PIC18F6310/6410/8310/8410

## 64/80-Pin Flash Microcontrollers with nanoWatt Technology

### Power-Managed Modes:

- Run: CPU on, Peripherals on
- Idle: CPU off, Peripherals on
- Sleep: CPU off, Peripherals off
- Ultra Low 50 nA Input Leakage
- Idle mode Currents Down to 2.3  $\mu$ A Typical
- Ultra Low 50 nA Input Leakage
- Sleep mode Currents Down to 0.1  $\mu$ A Typical
- Timer1 Oscillator: 1.0  $\mu$ A, 32 kHz, 2V Typical
- Watchdog Timer: 1.7  $\mu$ A Typical
- Two-Speed Oscillator Start-up

### Flexible Oscillator Structure:

- Four Crystal modes up to 40 MHz
- 4x Phase Lock Loop (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  $\mu$ s 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

### External Memory Interface (PIC18F8310/8410 Devices only):

- Address Capability of up to 2 Mbytes
- 16-Bit/8-Bit Interface

### Peripheral Highlights:

- High-Current Sink/Source 25 mA/25 mA
- Four External Interrupts
- Four Input Change Interrupts
- Four 8-Bit/16-Bit Timer/Counter modules
- Up to 3 Capture/Compare/PWM (CCP) modules

### Peripheral Highlights (Continued):

- Master Synchronous Serial Port (MSSP) module Supporting 3-Wire SPI (all 4 modes) and I<sup>2</sup>C™ Master and Slave modes
- Addressable USART module:
  - Supports RS-485 and RS-232
- Enhanced Addressable USART module:
  - Supports RS-485, RS-232 and LIN/J2602
  - Auto-Wake-up on Start bit
  - Auto-Baud Detect
- 10-Bit, up to 12-Channel Analog-to-Digital (A/D) Converter module:
  - Auto-acquisition capability
  - Conversion available during Sleep
- Dual Analog Comparators with Input Multiplexing
- Programmable 16-Level High/Low-Voltage Detection (HLVD) module:
  - Supports interrupt on High/Low-Voltage Detection

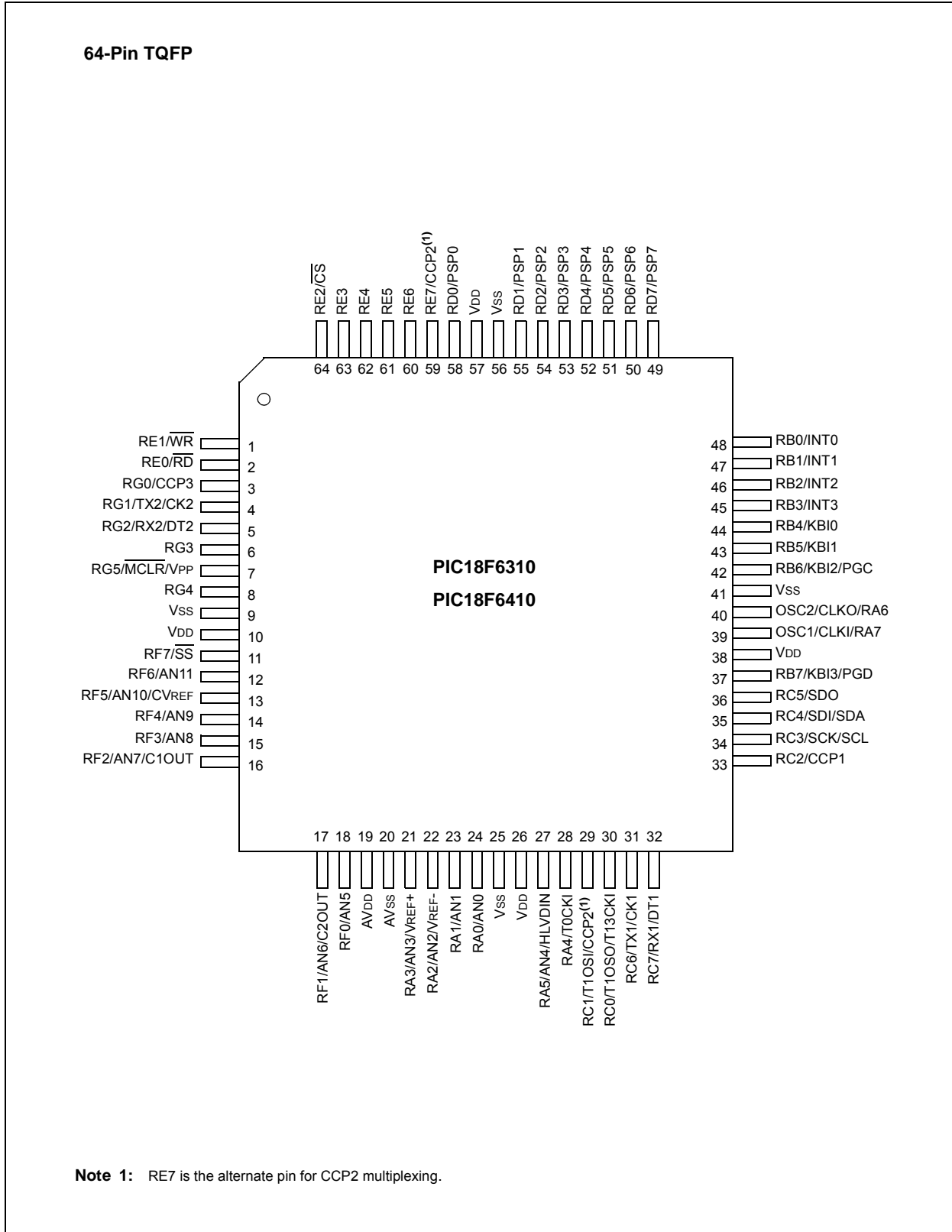
### Special Microcontroller Features:

- C Compiler Optimized Architecture:
  - Optional extended instruction set designed to optimize re-entrant code
- 1000 Erase/Write Cycle Flash Program Memory Typical
- Flash Retention: 100 Years Typical
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
  - 2% stability over V<sub>DD</sub> and temperature
- In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) via Two Pins
- Wide Operating Voltage Range: 2.0V to 5.5V
- Programmable Brown-out Reset (BOR) with Software Enable Option

Device	Program Memory (On-Board/External)		Data Memory	I/O	10-Bit A/D (ch)	CCP (PWM)	MSSP		EUSART/AUSART	Comparators	Timers 8/16-Bit	Ext. Bus
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)				SPI	Master I <sup>2</sup> C™				
PIC18F6310	8K/0	4096/0	768	54	12	3	Y	Y	1/1	2	1/3	N
PIC18F6410	16K/0	8192/0	768	54	12	3	Y	Y	1/1	2	1/3	N
PIC18F8310	8K/2M	4096/1M	768	70	12	3	Y	Y	1/1	2	1/3	Y
PIC18F8410	16K/2M	8192/1M	768	70	12	3	Y	Y	1/1	2	1/3	Y

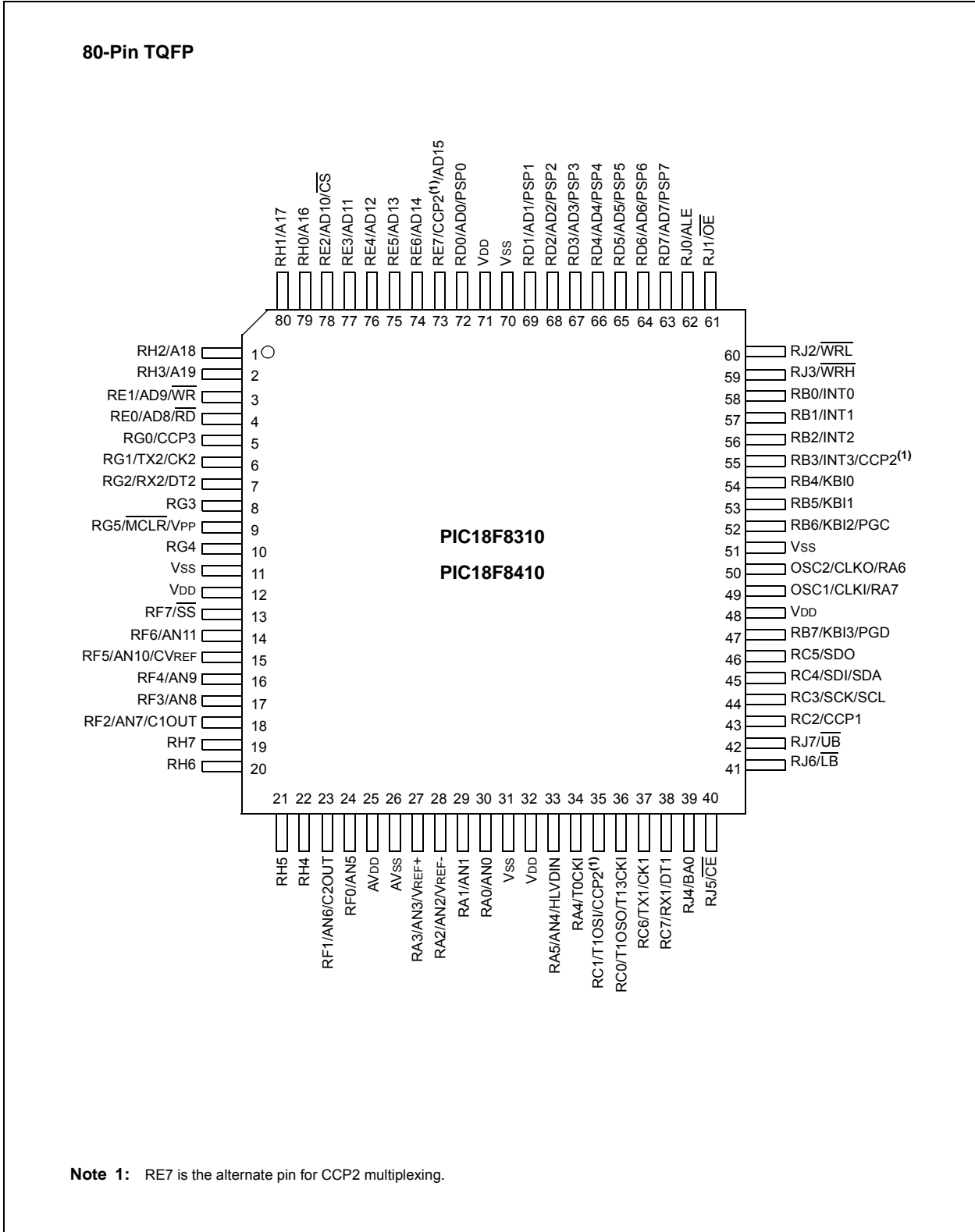
# PIC18F6310/6410/8310/8410

## Pin Diagrams



# PIC18F6310/6410/8310/8410

## Pin Diagrams (Continued)



# PIC18F6310/6410/8310/8410

---

## Table of Contents

1.0	Device Overview .....	9
2.0	Guidelines for Getting Started with PIC18F Microcontrollers .....	31
3.0	Oscillator Configurations .....	35
4.0	Power-Managed Modes .....	45
5.0	Reset .....	55
6.0	Memory Organization .....	67
7.0	Program Memory .....	89
8.0	External Memory Interface .....	95
9.0	8 x 8 Hardware Multiplier .....	107
10.0	Interrupts .....	109
11.0	I/O Ports .....	125
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	Master Synchronous Serial Port (MSSP) Module .....	177
18.0	Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) .....	217
19.0	Addressable Universal Synchronous Asynchronous Receiver Transmitter (AUSART) .....	241
20.0	10-Bit Analog-to-Digital Converter (A/D) Module .....	255
21.0	Comparator Module .....	265
22.0	Comparator Voltage Reference Module .....	271
23.0	High/Low-Voltage Detect (HLVD) .....	275
24.0	Special Features of the CPU .....	281
25.0	Instruction Set Summary .....	297
26.0	Development Support .....	347
27.0	Electrical Characteristics .....	351
28.0	Packaging Information .....	389
	Appendix A: Revision History .....	395
	Appendix B: Device Differences .....	395
	Appendix C: Conversion Considerations .....	396
	Appendix D: Migration from Baseline to Enhanced Devices .....	396
	Appendix E: Migration from Mid-Range to Enhanced Devices .....	397
	Appendix F: Migration from High-End to Enhanced Devices .....	397
	Index .....	399
	The Microchip Web Site .....	409
	Customer Change Notification Service .....	409
	Customer Support .....	409
	Reader Response .....	410
	PIC18F6310/6410/8310/8410 Product Identification System .....	411

## 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](mailto: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](http://www.microchip.com/cn) to receive the most current information on all of our products.

# PIC18F6310/6410/8310/8410

---

NOTES:



# PIC18F6310/6410/8310/8410

## 1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F6310
- PIC18F6410
- PIC18F8310
- PIC18F8410
- PIC18LF6310
- PIC18LF6410
- PIC18LF8310
- PIC18LF8410

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price. In addition to these features, the PIC18F6310/6410/8310/8410 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 PIC18F6310/6410/8310/8410 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  $\mu\text{A}$  and 2.1  $\mu\text{A}$ , respectively.

#### 1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F6310/6410/8310/8410 family offer nine different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

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

# PIC18F6310/6410/8310/8410

---

## 1.2 Other Special Features

- **Memory Endurance:** The Flash cells for program memory are rated to last for approximately a thousand erase/write cycles. Data retention without refresh is conservatively estimated to be greater than 100 years.
- **External Memory Interface:** For those applications where more program or data storage is needed, the PIC18F8310/8410 devices provide the ability to access external memory devices. The memory interface is configurable for both 8-bit and 16-bit data widths and uses a standard range of control signals to enable communication with a wide range of memory devices. With their 21-bit program counters, the 80-pin devices can access a linear memory space of up to 2 Mbytes.
- **Extended Instruction Set:** The PIC18F6310/6410/8310/8410 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 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 (ABD) 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, reduces code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit prescaler, allowing a time-out range from 4 ms to over 2 minutes that is stable across operating voltage and temperature.

## 1.3 Details on Individual Family Members

Devices in the PIC18F6310/6410/8310/8410 family are available in 64-pin (PIC18F6310/8310) and 80-pin (PIC18F6410/8410) packages. Block diagrams for the two groups are shown in [Figure 1-1](#) and [Figure 1-2](#), respectively.

The devices are differentiated from each other in three ways:

1. Flash Program Memory: 8 Kbytes in PIC18FX310 devices, 16 Kbytes in PIC18FX410 devices.
2. I/O Ports: 7 bidirectional ports on 64-pin devices, 9 bidirectional ports on 80-pin devices.
3. External Memory Interface: present on 80-pin devices only.

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 PIC18F6310/6410/8310/8410 family are available as both standard and low-voltage devices. Standard devices with Flash memory, designated with an "F" in the part number (such as PIC18F6310), accommodate an operating V<sub>DD</sub> range of 4.2V to 5.5V. Low-voltage parts, designated by "LF" (such as PIC18LF6410), function over an extended V<sub>DD</sub> range of 2.0V to 5.5V.

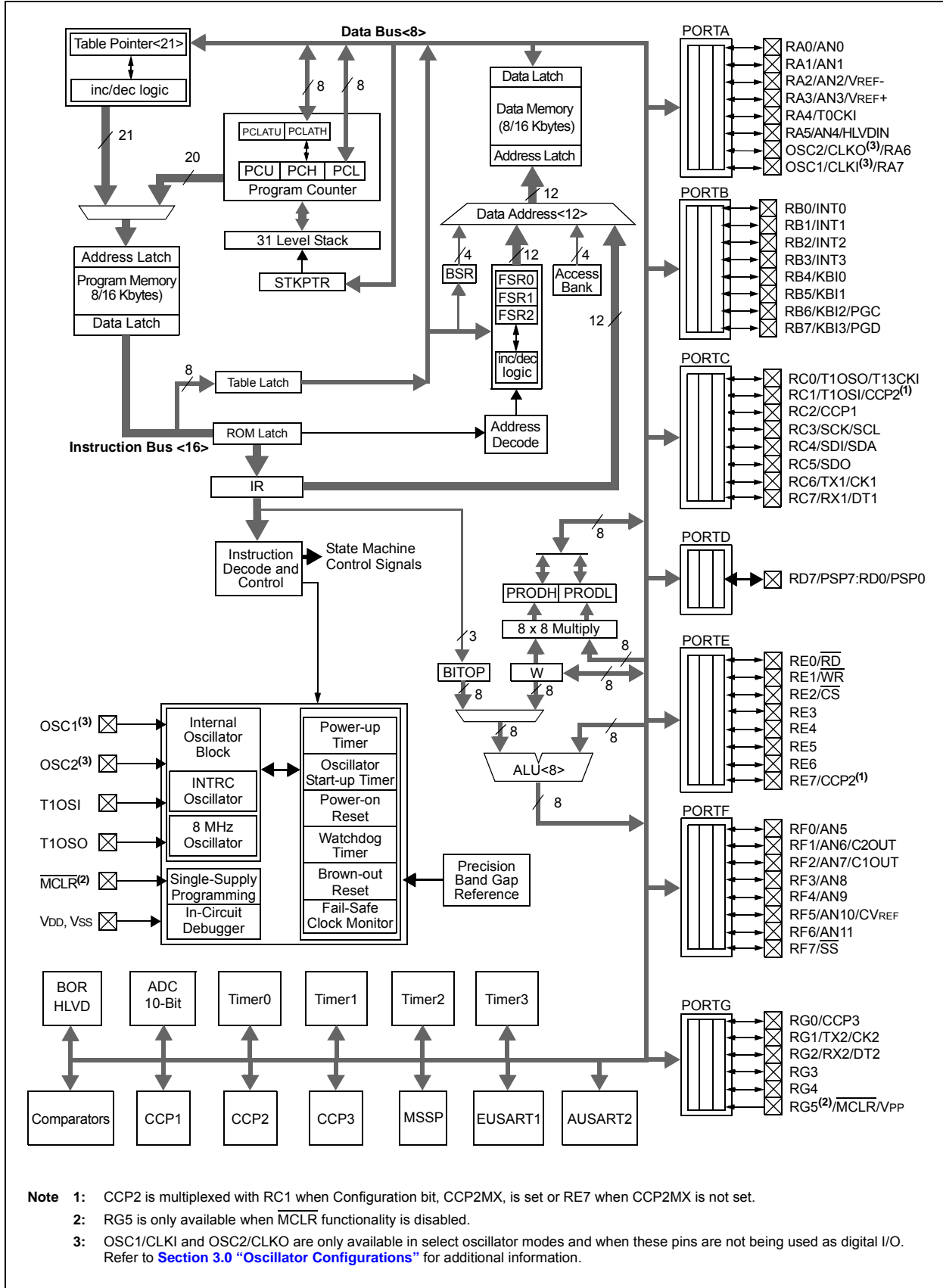
# PIC18F6310/6410/8310/8410

**TABLE 1-1: DEVICE FEATURES**

Features	PIC18F6310	PIC18F6410	PIC18F8310	PIC18F8410
Operating Frequency	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz
Program Memory (Bytes)	8K	16K	8K	16K
Program Memory (Instructions)	4096	8192	4096	8192
Data Memory (Bytes)	768	768	768	768
External Memory Interface	No	No	Yes	Yes
Interrupt Sources	22	22	22	22
I/O Ports	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G, H, J	Ports A, B, C, D, E, F, G, H, J
Timers	4	4	4	4
Capture/Compare/PWM Modules	3	3	3	3
Serial Communications	MSSP, AUSART Enhanced USART	MSSP, AUSART Enhanced USART	MSSP, AUSART Enhanced USART	MSSP, AUSART Enhanced USART
Parallel Communications	PSP	PSP	PSP	PSP
10-Bit Analog-to-Digital Module	12 Input Channels	12 Input Channels	12 Input Channels	12 Input Channels
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	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	64-Pin TQFP	64-Pin TQFP	80-Pin TQFP	80-Pin TQFP

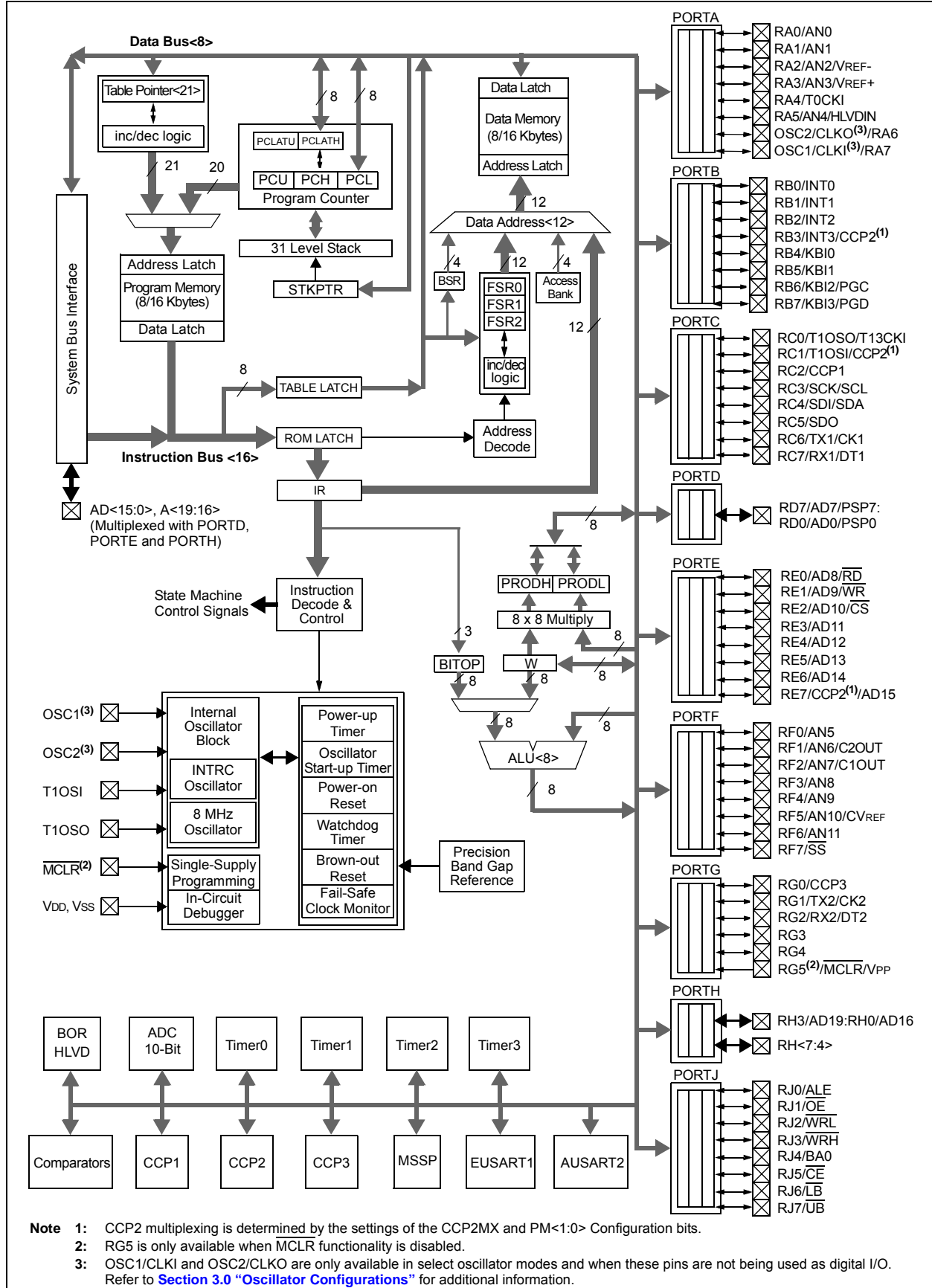
# PIC18F6310/6410/8310/8410

FIGURE 1-1: PIC18F6310/6410 (64-PIN) BLOCK DIAGRAM



# PIC18F6310/6410/8310/8410

FIGURE 1-2: PIC18F8310/8410 (80-PIN) BLOCK DIAGRAM































# PIC18F6310/6410/8310/8410

**TABLE 1-3: PIC18F8310/8410 PINOUT I/O DESCRIPTIONS (CONTINUED)**

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
RD0/AD0/PSP0	72			PORTD is a bidirectional I/O port.
RD0		I/O	ST	Digital I/O.
AD0		I/O	TTL	External Memory Address/Data 0.
PSP0		I/O	TTL	Parallel Slave Port data.
RD1/AD1/PSP1	69			
RD1		I/O	ST	Digital I/O.
AD1		I/O	TTL	External Memory Address/Data 1.
PSP1		I/O	TTL	Parallel Slave Port data.
RD2/AD2/PSP2	68			
RD2		I/O	ST	Digital I/O.
AD2		I/O	TTL	External Memory Address/Data 2.
PSP2		I/O	TTL	Parallel Slave Port data.
RD3/AD3/PSP3	67			
RD3		I/O	ST	Digital I/O.
AD3		I/O	TTL	External Memory Address/Data 3.
PSP3		I/O	TTL	Parallel Slave Port data.
RD4/AD4/PSP4	66			
RD4		I/O	ST	Digital I/O.
AD4		I/O	TTL	External Memory Address/Data 4.
PSP4		I/O	TTL	Parallel Slave Port data.
RD5/AD5/PSP5	65			
RD5		I/O	ST	Digital I/O.
AD5		I/O	TTL	External Memory Address/Data 5.
PSP5		I/O	TTL	Parallel Slave Port data.
RD6/AD6/PSP6	64			
RD6		I/O	ST	Digital I/O.
AD6		I/O	TTL	External Memory Address/Data 6.
PSP6		I/O	TTL	Parallel Slave Port data.
RD7/AD7/PSP7	63			
RD7		I/O	ST	Digital I/O.
AD7		I/O	TTL	External Memory Address/Data 7.
PSP7		I/O	TTL	Parallel Slave Port data.

**Legend:** TTL = TTL compatible input      CMOS = CMOS compatible input or output  
 ST = Schmitt Trigger input with CMOS levels      Analog = Analog input  
 I = Input      O = Output  
 P = Power      I<sup>2</sup>C = ST with I<sup>2</sup>C™ or SMB levels

- Note 1:** Alternate assignment for CCP2 when Configuration bit, CCP2MX, is cleared (all operating modes except Microcontroller mode).
- 2:** Default assignment for CCP2 in all operating modes (CCP2MX is set).
- 3:** Alternate assignment for CCP2 when CCP2MX is cleared (Microcontroller mode only).









## 2.0 GUIDELINES FOR GETTING STARTED WITH PIC18F MICROCONTROLLERS

### 2.1 Basic Connection Requirements

Getting started with the PIC18F6310/6410/8310/8410 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”](#))
- 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”](#))

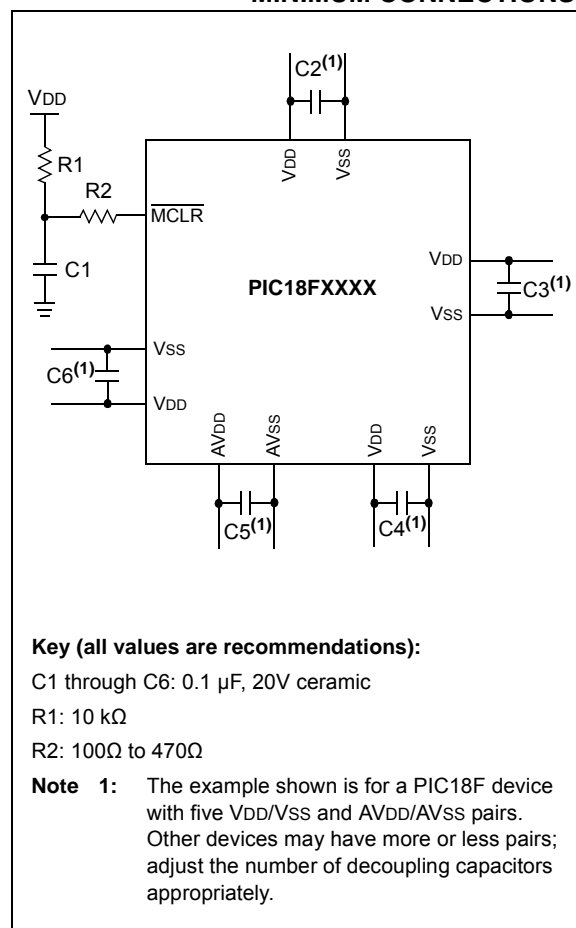
Additionally, the following pins may be required:

- 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**



# PIC18F6310/6410/8310/8410

## 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  $\mu\text{F}$  (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  $\mu\text{F}$  to 0.001  $\mu\text{F}$ . 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  $\mu\text{F}$  in parallel with 0.001  $\mu\text{F}$ ).
- **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  $\mu\text{F}$  to 47  $\mu\text{F}$ .

### 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/ $\mu\text{s}$ .

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.

<b>Note:</b> Not all devices incorporate software BOR control. See <a href="#">Section 5.0 "Reset"</a> for device-specific information.
---



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

The  $\overline{\text{MCLR}}$  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  $V_{DD}$  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  $\overline{\text{MCLR}}$  pin. Consequently, specific voltage levels ( $V_{IH}$  and  $V_{IL}$ ) 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  $\overline{\text{MCLR}}$  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  $\overline{\text{MCLR}}$  pin should be placed within 0.25 inch (6 mm) of the pin.

## 2.4 ICSP Pins

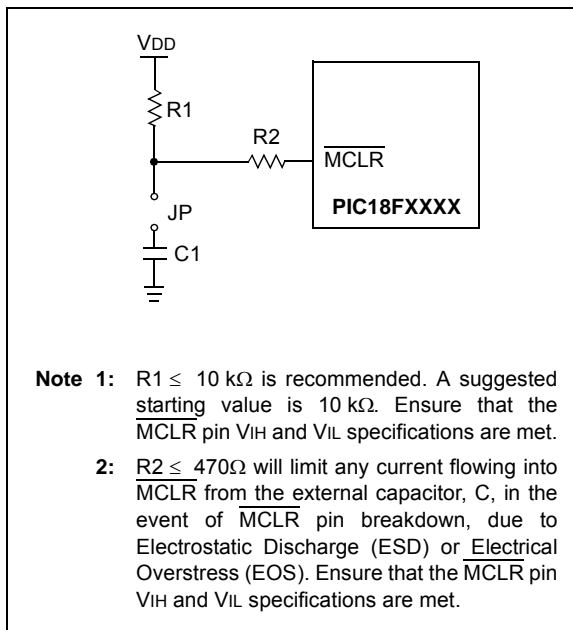
The PGC and PGD pins are used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor 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 ( $V_{IH}$ ) and input low ( $V_{IL}$ ) 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 26.0 “Development Support”**.

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



# PIC18F6310/6410/8310/8410

## 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).

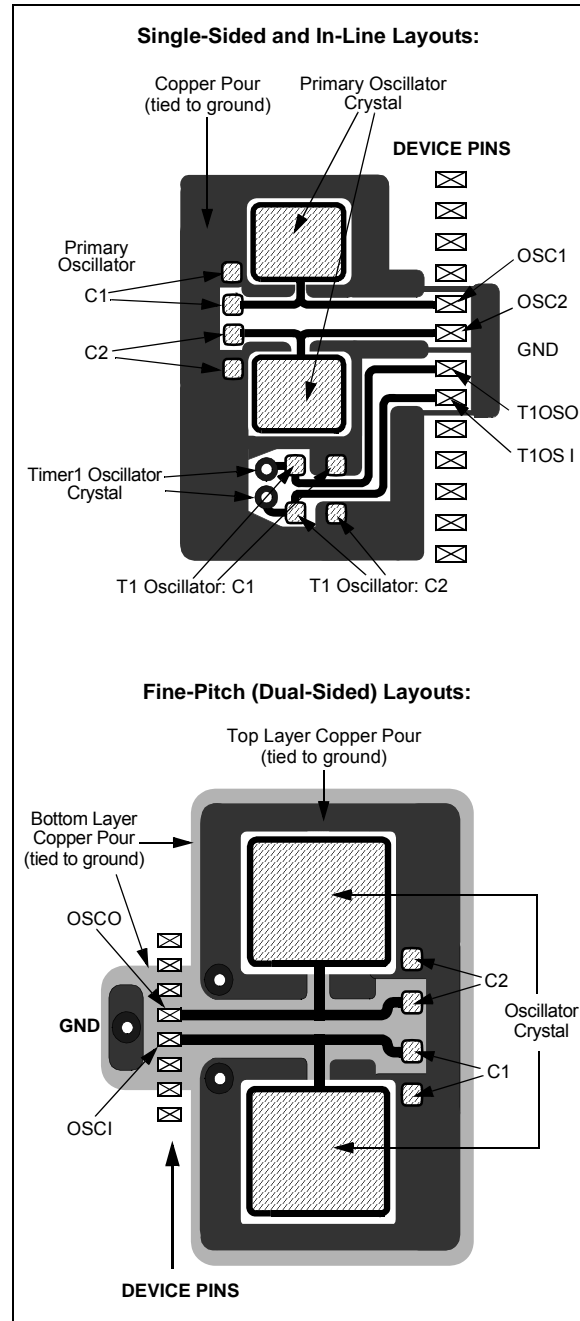
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate web site ([www.microchip.com](http://www.microchip.com)):

- AN826, “Crystal Oscillator Basics and Crystal Selection for rPIC™ 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**



# PIC18F6310/6410/8310/8410

## 3.0 OSCILLATOR CONFIGURATIONS

### 3.1 Oscillator Types

PIC18F6310/6410/8310/8410 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

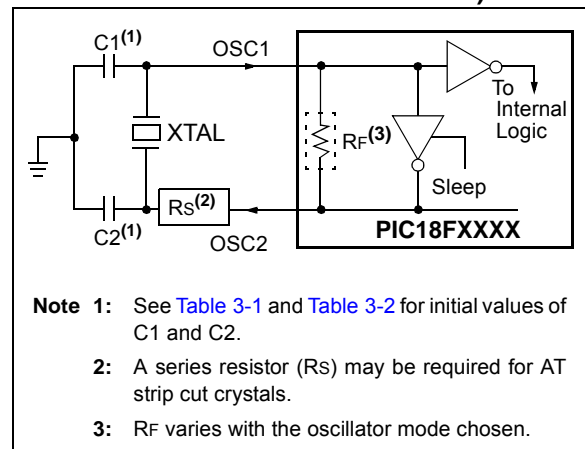
### 3.2 Crystal Oscillator/Ceramic Resonators

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)**



**TABLE 3-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS**

Typical Capacitor Values Used:			
Mode	Freq	OSC1	OSC2
XT	455 kHz	56 pF	56 pF
	2.0 MHz	47 pF	47 pF
	4.0 MHz	33 pF	33 pF
HS	8.0 MHz	27 pF	27 pF
	16.0 MHz	22 pF	22 pF

**Capacitor values are for design guidance only.**  
 These capacitors were tested with the resonators listed below for basic start-up and operation. **These values are not optimized.**  
 Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.  
 See the notes following Table 3-2 for additional information.

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

# PIC18F6310/6410/8310/8410

**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:**

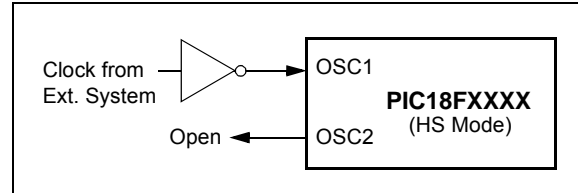
32 kHz	4 MHz
200 kHz	8 MHz
1 MHz	20 MHz

**Note 1:** Higher capacitance increases the stability of 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)**

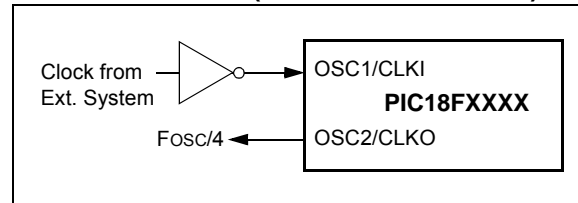


### 3.3 External Clock Input

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.

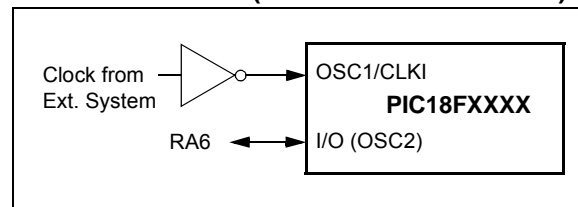
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. [Figure 3-3](#) shows the pin connections for the EC Oscillator mode.

**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)**



## 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:

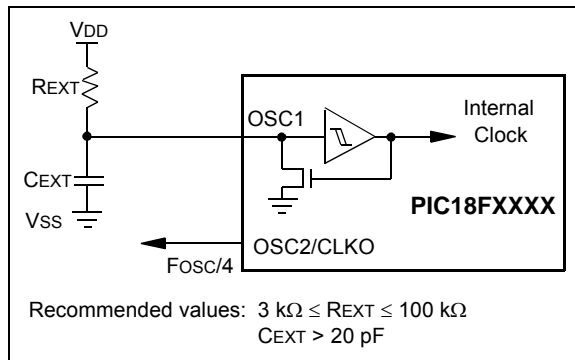
- Supply voltage
- Values of the external resistor (R<sub>EXT</sub>) and capacitor (C<sub>EXT</sub>)
- Operating temperature

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 C<sub>EXT</sub> values)
- Variations within the tolerance of limits of R<sub>EXT</sub> and C<sub>EXT</sub>

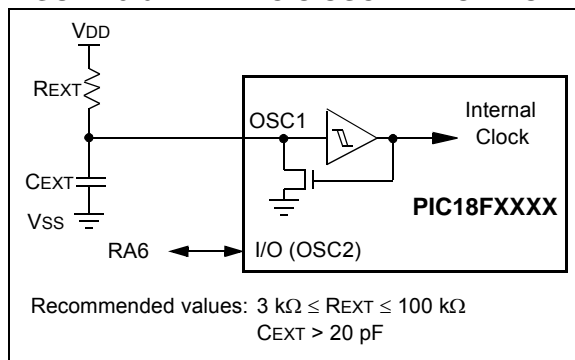
In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. [Figure 3-5](#) shows how the R/C combination is connected.

**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

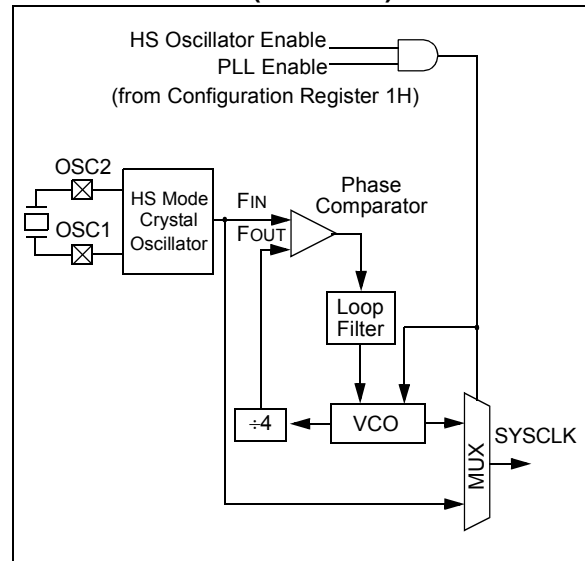
A Phase Locked Loop (PLL) circuit is provided as an option for users who want to use a lower frequency oscillator circuit, or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals, or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

### 3.5.1 HSPLL OSCILLATOR MODE

The HSPLL mode makes use of the HS Oscillator mode 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”](#).

# PIC18F6310/6410/8310/8410

## 3.6 Internal Oscillator Block

The PIC18F6310/6410/8310/8410 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 24.0 "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  $F_{osc}/4$ , 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 frequency will begin shifting to the new frequency. The INTOSC clock will stabilize within 1 ms. Code execution continues during this shift. There is no indication that the shift has occurred.

The OSCTUNE register also implements the INTSRC and PLEN 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 PLEN 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, PLEN (OSCTUNE<6>), is used to enable or disable its operation.

The PLL is available when the device is configured to use the internal oscillator block as its primary clock source ( $F_{osc}<3:0> = 1001$  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 PLEN 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  $V_{DD}$  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 OSTUNE 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 examples follow, but other techniques may be used. Three compensation techniques are discussed in **Section 3.6.5.1 "Compensating with the AUSART"**, **Section 3.6.5.2 "Compensating with the Timers"** and **Section 3.6.5.3 "Compensating with the Timers"**, but other techniques may be used.

# PIC18F6310/6410/8310/8410

## 3.6.5.1 Compensating with the AUSART

An adjustment may be required when the AUSART 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 OSTUNE to reduce the clock frequency. On the other hand, errors in data may suggest that the clock speed is too low; to compensate, increment OSTUNE 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 Timers

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. 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, then the internal oscillator block is running too fast; to compensate, decrement the OSTUNE register. If the measured time is much less than the calculated time, then the internal oscillator block is running too slow; to compensate, increment the OSTUNE register.

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

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INTSRC	PLLEN <sup>(1)</sup>	—	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<sup>(1)</sup>  
 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 [Section 3.6.4 "PLL in INTOSC Modes"](#) for details.

# PIC18F6310/6410/8310/8410

## 3.7 Clock Sources and Oscillator Switching

Like previous PIC18 devices, the PIC18F6310/6410/8310/8410 family includes a feature that allows the device clock source to be switched from the main oscillator to an alternate low-frequency clock source. PIC18F6310/6410/8310/8410 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.

PIC18F6310/6410/8310/8410 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).

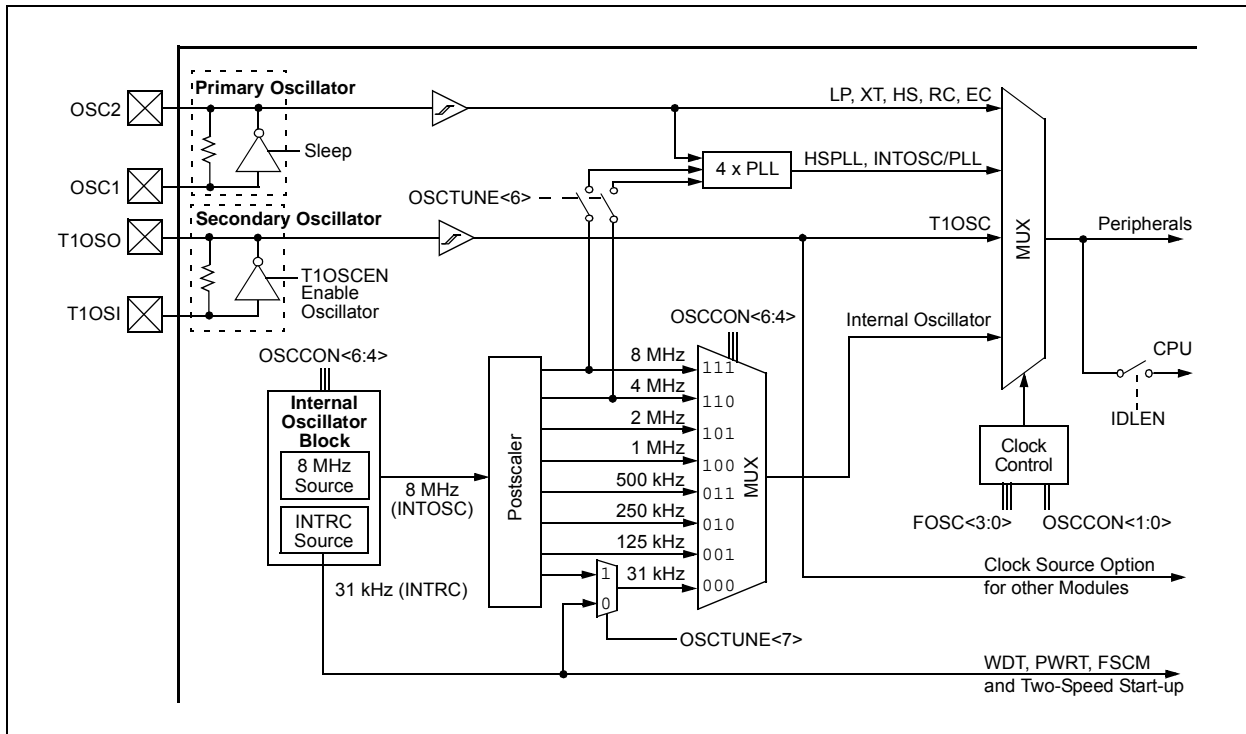
Most often, a 32.768 kHz watch crystal is connected between the RC0/T1OSO/T13CK1 and RC1/T1OSI/CCP2 pins. Like the LP mode oscillator circuit, loading capacitors are also connected from each pin to ground.

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 PIC18F6310/6410/8310/8410 devices are shown in [Figure 3-8](#). See [Section 24.0 “Special Features of the CPU”](#) for Configuration register details.

**FIGURE 3-8: PIC18F6310/6410/8310/8410 CLOCK DIAGRAM**





## 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 post-scaler (31.25 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. Resets, the default output frequency of the internal oscillator block, are 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 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

PIC18F6310/6410/8310/8410 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"](#).

# PIC18F6310/6410/8310/8410

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

R/W-0	R/W-1	R/W-0	R/W-0	R <sup>(1)</sup>	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<sup>(3)</sup>  
011 = 500 kHz  
010 = 250 kHz  
001 = 125 kHz  
000 = 31 kHz (from either INTOSC/256 or INTRC directly)<sup>(2)</sup>
- bit 3            **OSTS:** Oscillator Start-up Time-out Status bit<sup>(1)</sup>  
1 = Oscillator Start-up Timer time-out has expired; primary oscillator is running  
0 = Oscillator Start-up Timer time-out is running; primary oscillator is not ready
- bit 2            **IOFS:** INTOSC Frequency Stable bit  
1 = INTOSC frequency is stable  
0 = INTOSC frequency is not stable
- bit 1-0        **SCS<1:0>:** System Clock Select bits  
1x = Internal oscillator block  
01 = Secondary (Timer1) oscillator  
00 = Primary oscillator

- Note 1:** Depends on the state of the IESO Configuration bit.  
**2:** Source selected by the INTSRC bit (OSCTUNE<7>), see [Section 3.6.3 “OSCTUNE Register”](#).  
**3:** Default output frequency of INTOSC on Reset.

## 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 24.2 “Watchdog Timer \(WDT\)”](#) through [Section 24.4 “Fail-Safe Clock Monitor”](#) for more information on WDT, Fail-Safe Clock Monitor and Two-Speed Start-up). 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.

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

Enabling any on-chip feature that will operate during Sleep will increase the current consumed during Sleep. The INTRC is required to support WDT operation. The Timer1 oscillator may be operating to support a real-time clock. Other features may be operating that do not require a device clock source (i.e., MSSP slave, PSP, INTx pins and others). Peripherals that may add significant current consumption are listed in [Section 27.2 “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 27-12](#)). 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 27-12](#)), 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<sup>(1)</sup>**

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

**Note 1:** See [Table 5-2](#) in [Section 5.0 “Reset”](#) for time-outs due to Sleep and MCLR Reset.

# PIC18F6310/6410/8310/8410

---

NOTES:

# PIC18F6310/6410/8310/8410

## 4.0 POWER-MANAGED MODES

PIC18F6310/6410/8310/8410 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:

- Sleep mode
- Idle modes
- Run modes

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

The power-managed modes include several power-saving features. One of these 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 deciding if the CPU is to be clocked or not and selecting a clock source. The IDLEN bit controls CPU clocking, while the SCS<1:0> bits select a 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

Entering power-managed Run mode, or 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 being 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> Bits		Module Clocking		Available Clock and Oscillator Source
	IDLEN <sup>(1)</sup>	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 <sup>(2)</sup> 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 <sup>(2)</sup>
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 <sup>(2)</sup>

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

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

# PIC18F6310/6410/8310/8410

## 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 power-managed RC 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 from 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 will 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 24.3 “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.

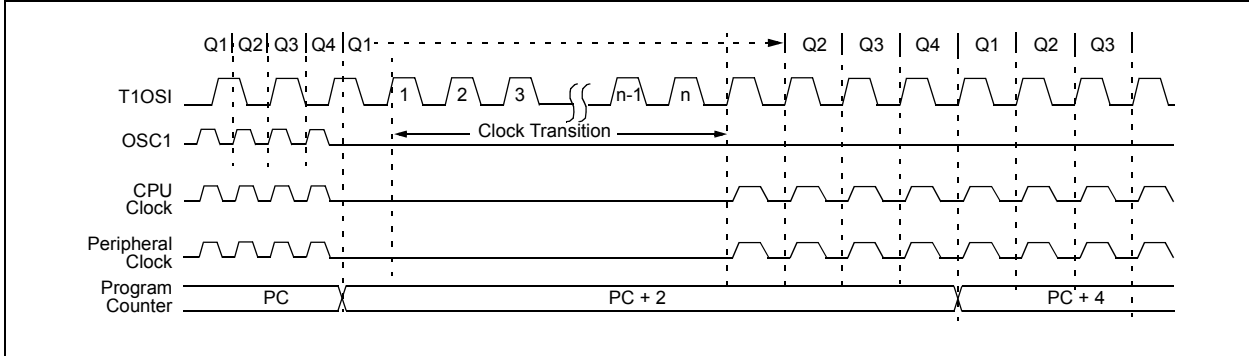
**Note:** The Timer1 oscillator should already be running prior to entering SEC\_RUN mode. If the T1OSCEN bit is not set when the SCS<1:0> bits are set to ‘01’, entry to SEC\_RUN mode will not occur. If the Timer1 oscillator is enabled, but not yet running, peripheral clocks will be delayed until the oscillator has started; in such situations, initial oscillator operation is far from stable and unpredictable operation may result.

# PIC18F6310/6410/8310/8410

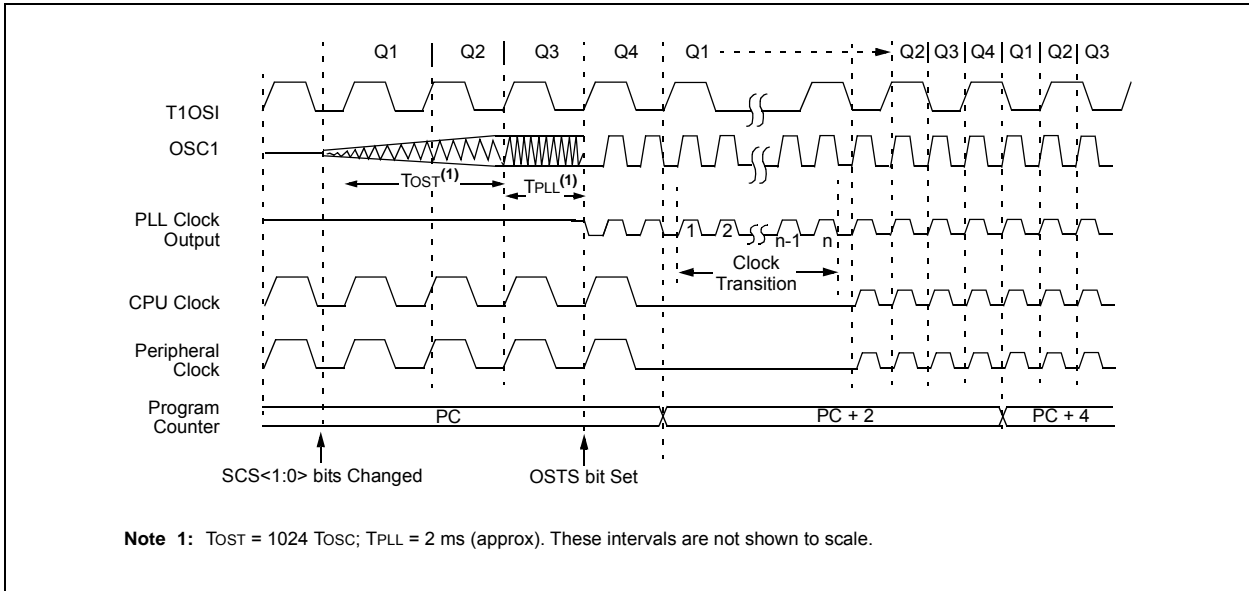
On transitions from SEC\_RUN mode to PRI\_RUN, the peripherals and CPU continue to be clocked from the Timer1 oscillator while the primary clock is started. When the primary clock becomes ready, a clock switch back to the primary clock occurs (see [Figure 4-2](#)).

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.

# PIC18F6310/6410/8310/8410

---

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

If the primary clock source is the internal oscillator block (either 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 the SCS1 bit to '1'. Although it is ignored, it is recommended that the SCS0 bit 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.

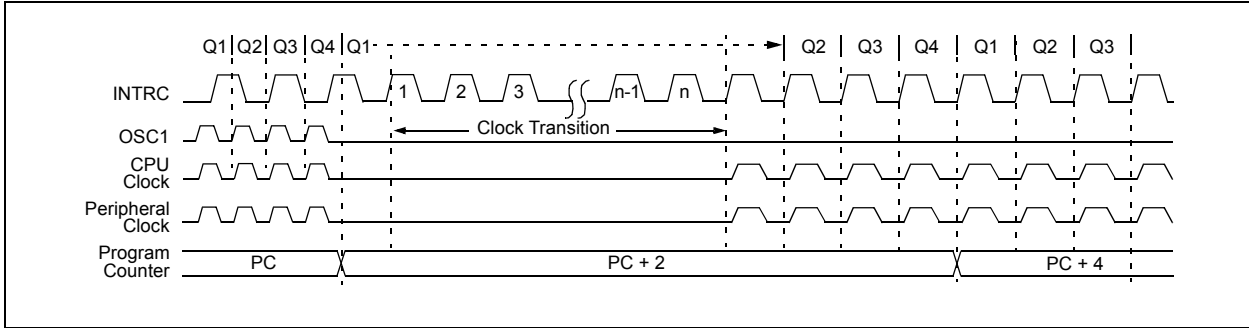
If the IRCF bits were previously at a non-zero value, or if INTSRC was set before setting SCS1 and the INTOSC source was already stable, the IOFS bit will remain set.

On transitions from RC\_RUN mode to PRI\_RUN, 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.

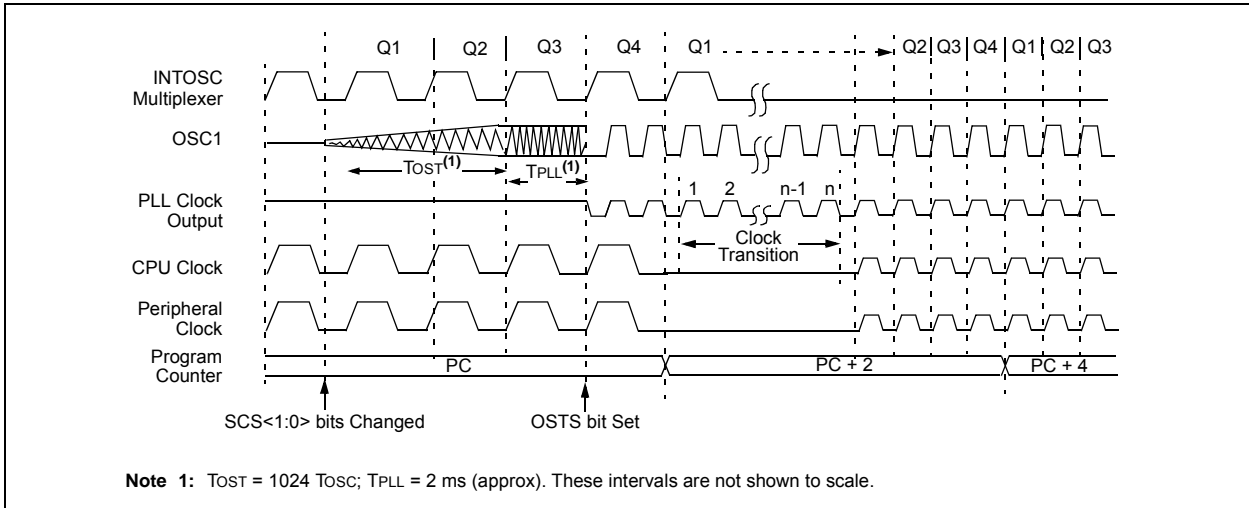


# PIC18F6310/6410/8310/8410

**FIGURE 4-3: TRANSITION TIMING TO RC\_RUN MODE**



**FIGURE 4-4: TRANSITION TIMING FROM RC\_RUN MODE TO PRI\_RUN MODE**



# PIC18F6310/6410/8310/8410

## 4.3 Sleep Mode

The power-managed Sleep mode in the PIC18F6310/6410/8310/8410 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 (see 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 primary clock source 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 24.0 “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.

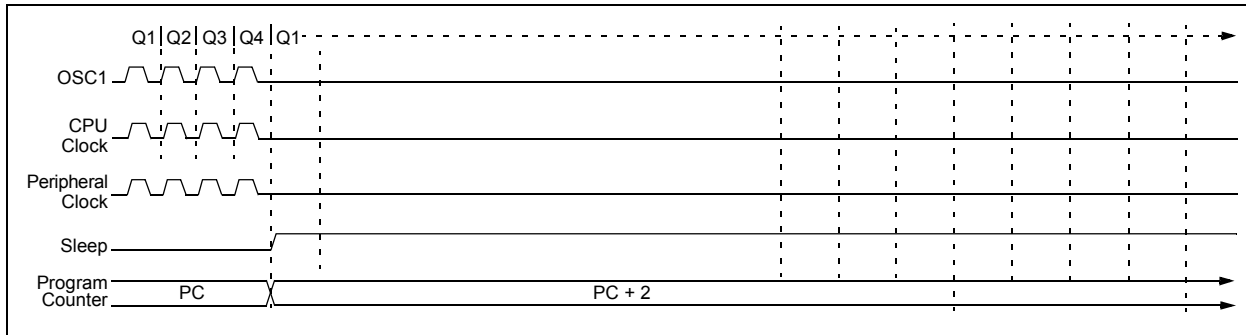
If the IDLEN bit is set to a ‘1’ when a SLEEP instruction is executed, the peripherals will be clocked from the clock source selected using the SCS<1:0> bits; however, the CPU will not be clocked. The clock source status bits are not affected. Setting IDLEN and executing SLEEP provides a quick method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

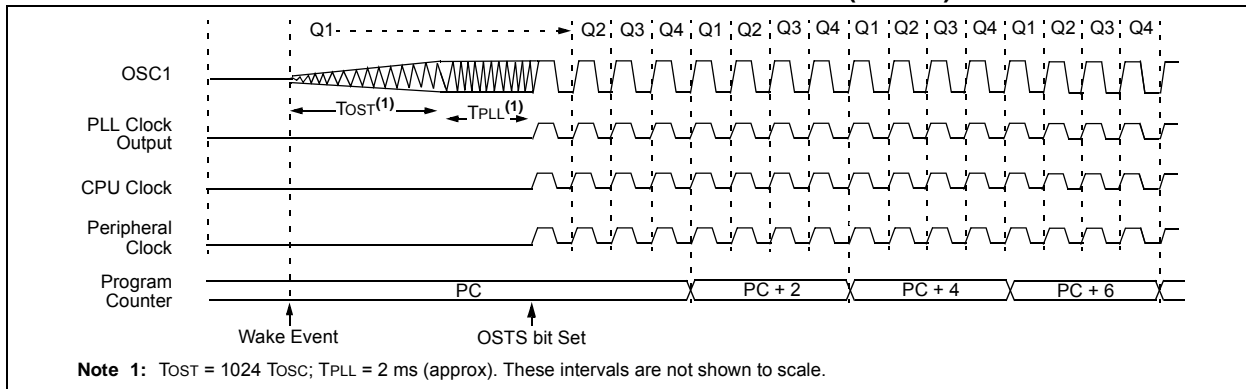
Since the CPU is not executing instructions, the only exits from any of the Idle modes are by interrupt, WDT time-out or a Reset. When a wake event occurs, CPU execution is delayed by an interval of T<sub>CSD</sub> (Parameter 38, Table 27-12), while it becomes ready to execute code. When the CPU begins 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 the 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)**



# PIC18F6310/6410/8310/8410

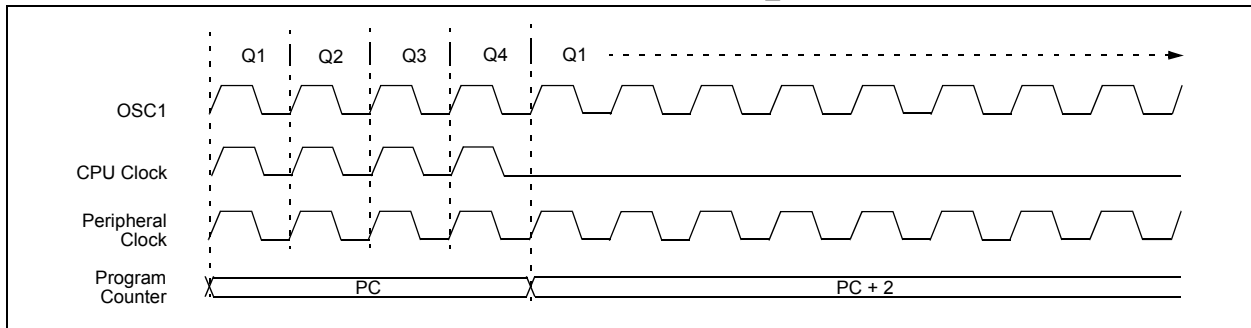
## 4.4.1 PRI\_IDLE MODE

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.

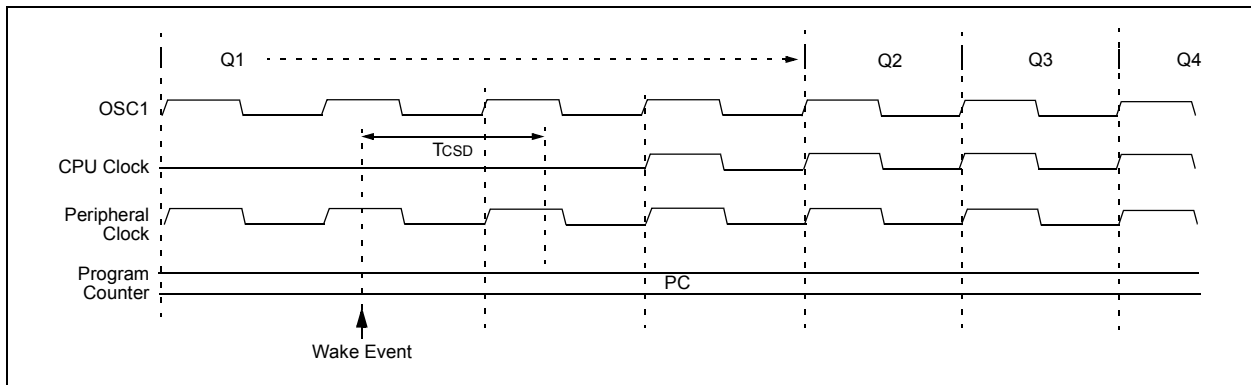
PRI\_IDLE mode is entered from PRI\_RUN mode by setting the IDLEN bit and executing a `SLEEP` instruction. If the device is in another Run mode, set IDLEN first, then clear the SCS bits and execute `SLEEP`. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source 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,  $T_{CSD}$ , 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](#)).

**FIGURE 4-7: TRANSITION TIMING FOR ENTRY TO PRI\_IDLE MODE**



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



# PIC18F6310/6410/8310/8410

---

## 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 IDLEN first, then set SCS<1:0> 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.

## 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 27-12](#)). 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”](#) through [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 or 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, T<sub>CSD</sub>, 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 24.2 “Watchdog Timer \(WDT\)”](#)).

The WDT timer and postscaler are cleared by executing a SLEEP or CLRWDT instruction, losing 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 24.3 “Two-Speed Start-up”](#)) or Fail-Safe Clock Monitor (see [Section 24.4 “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, T<sub>CSD</sub>, 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.

# PIC18F6310/6410/8310/8410

**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 <sub>CSD</sub> <sup>(2)</sup>	OSTS
	HSPLL		—
	EC, RC, INTRC <sup>(1)</sup>		IOFS
	INTOSC <sup>(3)</sup>		
T1OSC or INTRC <sup>(1)</sup>	LP, XT, HS	T <sub>OSt</sub> <sup>(4)</sup>	OSTS
	HSPLL	T <sub>OSt</sub> + t <sub>rc</sub> <sup>(4)</sup>	
	EC, RC, INTRC <sup>(1)</sup>	T <sub>CSD</sub> <sup>(2)</sup>	—
	INTOSC <sup>(2)</sup>	T <sub>IOBST</sub> <sup>(5)</sup>	IOFS
INTOSC <sup>(3)</sup>	LP, XT, HS	T <sub>OSt</sub> <sup>(5)</sup>	OSTS
	HSPLL	T <sub>OSt</sub> + t <sub>rc</sub> <sup>(4)</sup>	
	EC, RC, INTRC <sup>(1)</sup>	T <sub>CSD</sub> <sup>(2)</sup>	—
	INTOSC <sup>(2)</sup>	None	IOFS
None (Sleep mode)	LP, XT, HS	T <sub>OSt</sub> <sup>(4)</sup>	OSTS
	HSPLL	T <sub>OSt</sub> + t <sub>rc</sub> <sup>(4)</sup>	
	EC, RC, INTRC <sup>(1)</sup>	T <sub>CSD</sub> <sup>(2)</sup>	—
	INTOSC <sup>(2)</sup>	T <sub>IOBST</sub> <sup>(5)</sup>	IOFS

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

- 2:** T<sub>CSD</sub> (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:** T<sub>OSt</sub> is the Oscillator Start-up Timer (Parameter 32). t<sub>rc</sub> is the PLL Lock-out Timer (Parameter F12); it is also designated as T<sub>PLL</sub>.
- 5:** Execution continues during T<sub>IOBST</sub> (Parameter 39), the INTOSC stabilization period.

# PIC18F6310/6410/8310/8410

## 5.0 RESET

The PIC18F6310/6410/8310/8410 devices differentiate between various kinds of Reset:

- Power-on Reset (POR)
- $\overline{\text{MCLR}}$  Reset during normal operation
- $\overline{\text{MCLR}}$  Reset during power-managed modes
- Watchdog Timer (WDT) Reset (during execution)
- Programmable Brown-out Reset (BOR)
- RESET Instruction
- Stack Full Reset
- 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 6.1.3.4 “Stack Full and Underflow Resets”](#). WDT Resets are covered in [Section 24.2 “Watchdog Timer \(WDT\)”](#).

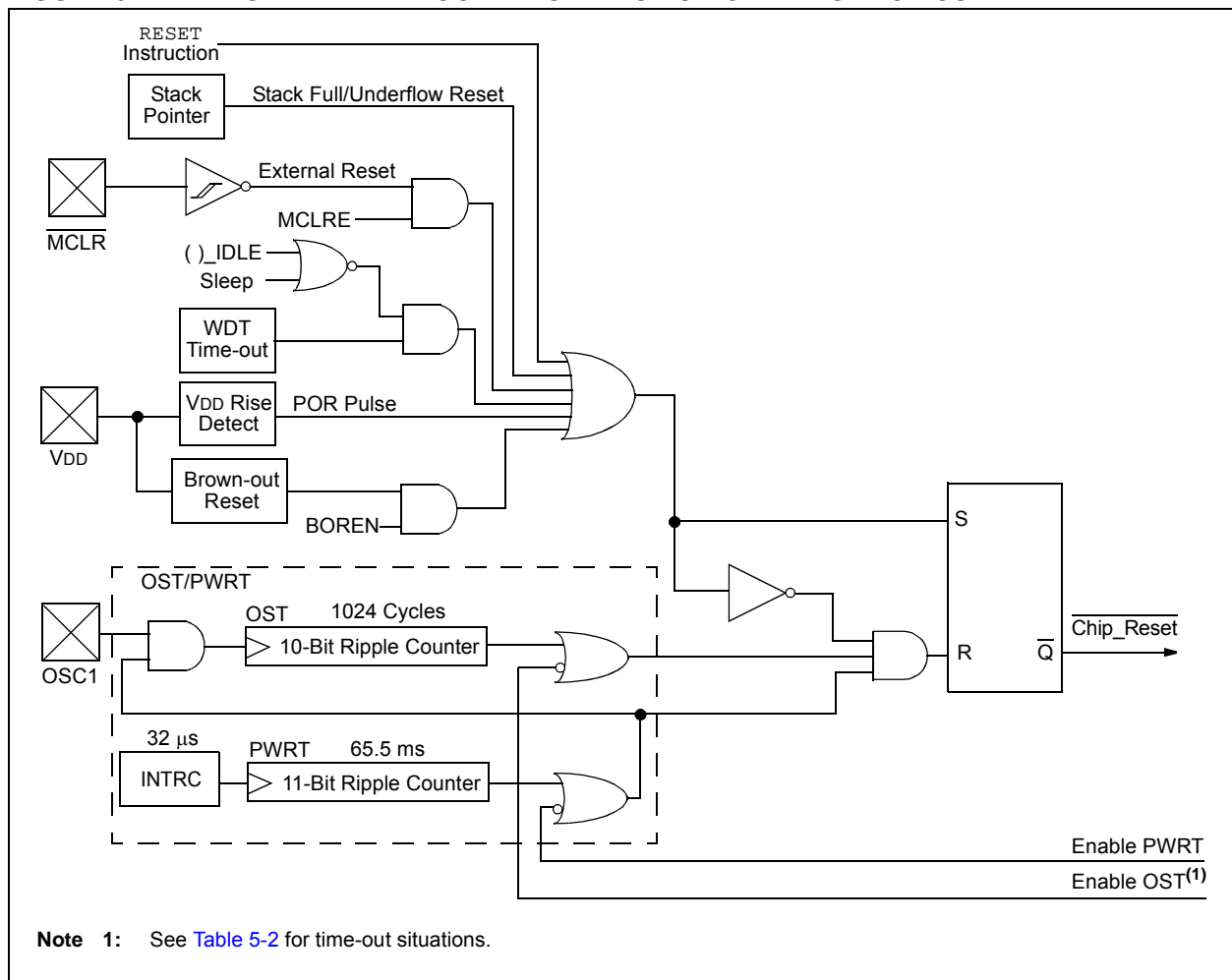
A simplified block diagram of the On-Chip Reset Circuit is shown in [Figure 5-1](#).

## 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 set by the event and must be cleared 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**



# PIC18F6310/6410/8310/8410

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

R/W-0	R/W-0 <sup>(1)</sup>	U-0	R/W-1	R-1	R-1	R/W-0	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<sup>(1)</sup>  
                   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 read 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 Timer 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  
                   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'.

**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).



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

The  $\overline{\text{MCLR}}$  pin provides a method for triggering a hard external Reset of the device. A Reset is generated by holding the pin low. PIC18 Extended MCU 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 PIC18F6310/6410/8310/8410 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 11.7 “PORTG, TRISG and LATG Registers”](#) for more information.

## 5.3 Power-on Reset (POR)

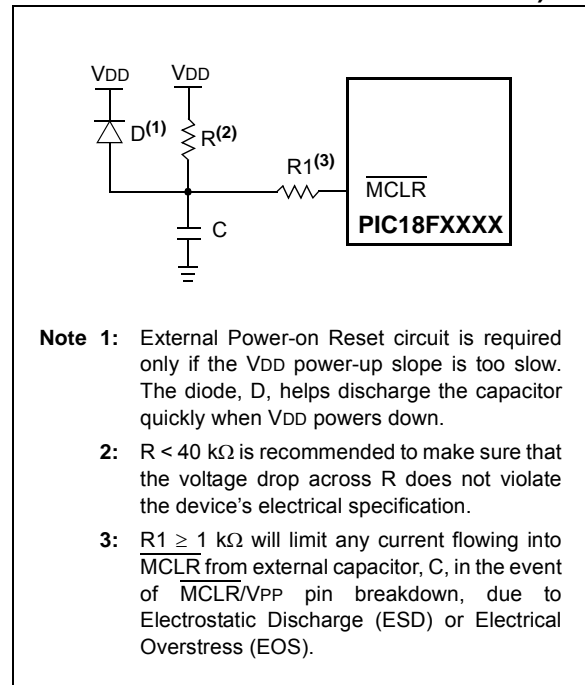
A Power-on Reset pulse is generated on-chip whenever  $V_{DD}$  rises above a certain threshold. This allows the device to start in the initialized state when  $V_{DD}$  is adequate for operation.

To take advantage of the POR circuitry, tie the  $\overline{\text{MCLR}}$  pin through a resistor ( $1\text{ k}\Omega$  to  $10\text{ k}\Omega$ ) to  $V_{DD}$ . This will eliminate external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for  $V_{DD}$  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 POR 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 POR.

**FIGURE 5-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW  $V_{DD}$  POWER-UP)**



# PIC18F6310/6410/8310/8410

## 5.4 Brown-out Reset (BOR)

PIC18F6310/6410/8310/8410 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-up Timer (PWRT) are independently configured. Enabling the 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 the 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 BOR or POR 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 POR and BOR. This assumes that the POR bit is reset to '1' in software immediately after any POR event. If  $\overline{\text{BOR}}$  is '0' while POR is '1', it can be reliably assumed that a BOR 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 is disabled; must be enabled by reprogramming the Configuration bits.
0	1	Available	BOR is enabled in software; operation controlled by SBOREN.
1	0	Unavailable	BOR is enabled in hardware and active during the Run and Idle modes; disabled during Sleep mode.
1	1	Unavailable	BOR is enabled in hardware; must be disabled by reprogramming the Configuration bits.

## 5.5 Device Reset Timers

PIC18F6310/6410/8310/8410 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 PIC18F6310/6410/8310/8410 devices is an 11-bit counter which uses the INTRC source as the clock input. This yields an approximate time interval of  $2048 \times 32 \mu\text{s} = 65.6 \text{ ms}$ . 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  $\overline{\text{PWRTE}}\text{N}$  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 is 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 (TPLL) 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**

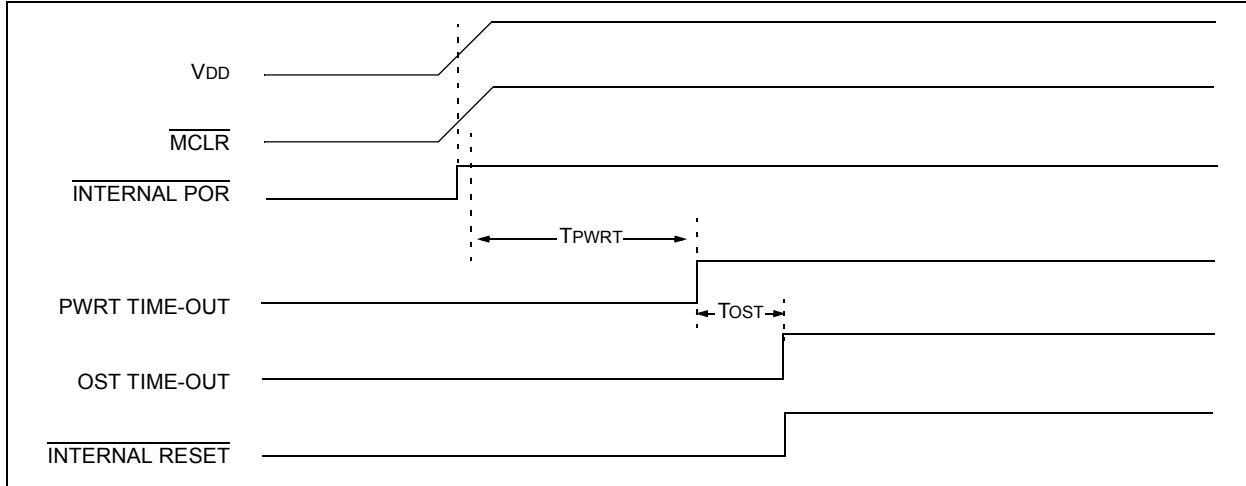
Oscillator Configuration	Power-up <sup>(2)</sup> and Brown-out		Exit from Power-Managed Mode
	$\overline{\text{PWRTE}}\text{N} = 0$	$\overline{\text{PWRTE}}\text{N} = 1$	
HSPLL	$66 \text{ ms}^{(1)} + 1024 \text{ T}_{\text{osc}} + 2 \text{ ms}^{(2)}$	$1024 \text{ T}_{\text{osc}} + 2 \text{ ms}^{(2)}$	$1024 \text{ T}_{\text{osc}} + 2 \text{ ms}^{(2)}$
HS, XT, LP	$66 \text{ ms}^{(1)} + 1024 \text{ T}_{\text{osc}}$	$1024 \text{ T}_{\text{osc}}$	$1024 \text{ T}_{\text{osc}}$
EC, ECIO	$66 \text{ ms}^{(1)}$	—	—
RC, RCIO	$66 \text{ ms}^{(1)}$	—	—
INTIO1, INTIO2	$66 \text{ ms}^{(1)}$	—	—

**Note 1:** 66 ms (65.5 ms) is the nominal Power-up Timer (PWRT) delay.

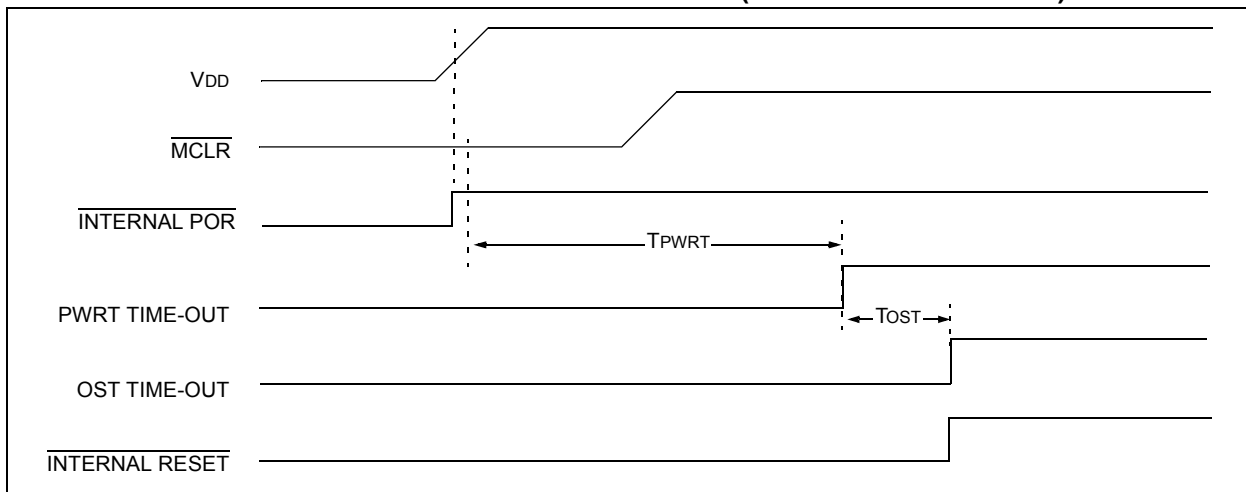
**Note 2:** 2 ms is the nominal time required for the PLL to lock.

# PIC18F6310/6410/8310/8410

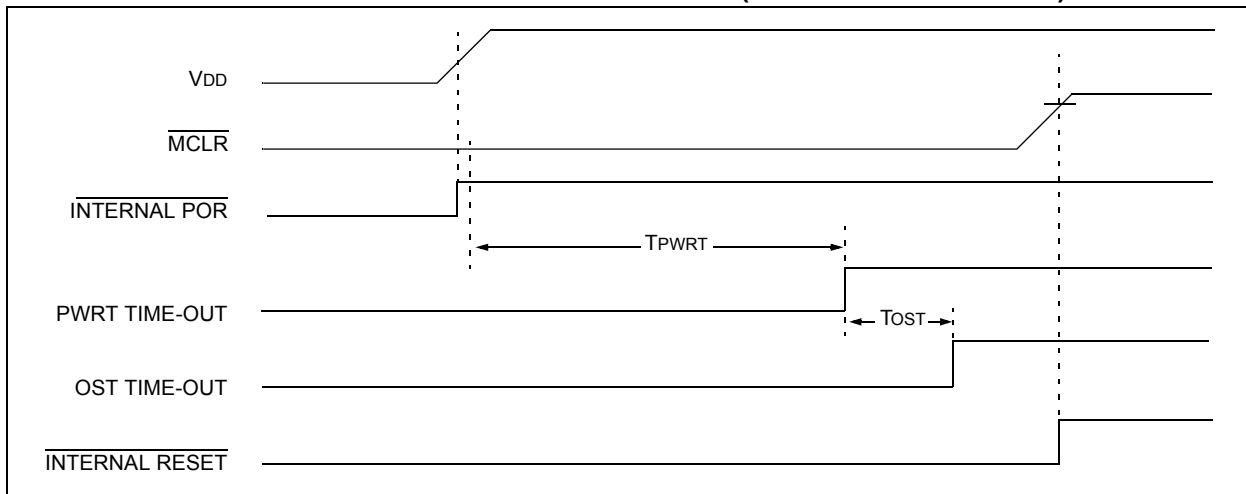
**FIGURE 5-3: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  TIED TO  $V_{\text{DD}}$ ,  $V_{\text{DD}}$  RISE <  $T_{\text{PWRT}}$ )**



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

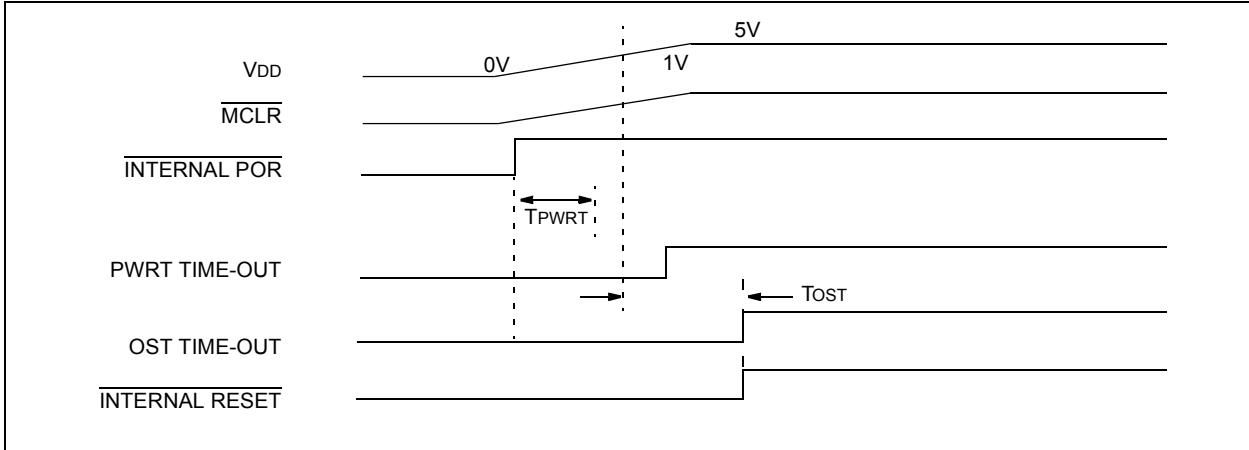


**FIGURE 5-5: TIME-OUT SEQUENCE ON POWER-UP ( $\overline{\text{MCLR}}$  NOT TIED TO  $V_{\text{DD}}$ ): CASE 2**

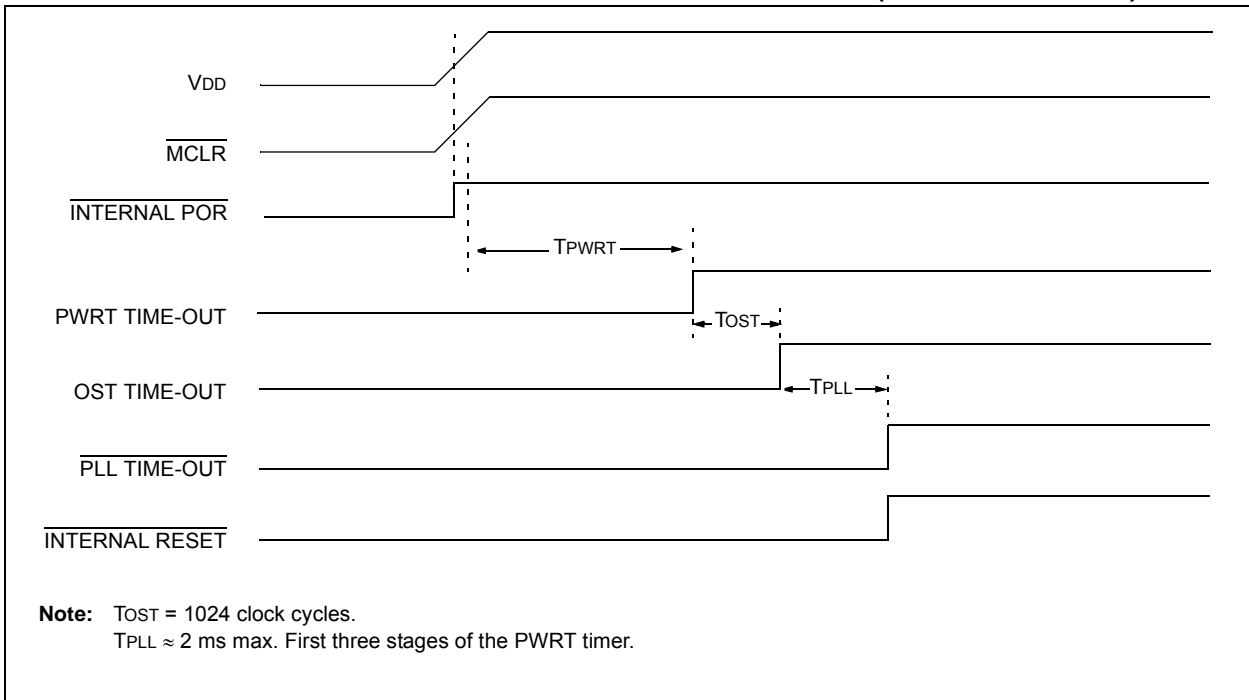


# PIC18F6310/6410/8310/8410

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



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



# PIC18F6310/6410/8310/8410

## 5.6 Reset State of Registers

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

Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register,  $\overline{RI}$ ,  $\overline{TO}$ ,  $\overline{PD}$ ,  $\overline{POR}$  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	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 <sup>(2)</sup>	0	u	u	u	u	u	u
Brown-out Reset	0000h	u <sup>(2)</sup>	1	1	1	u	0	u	u
$\overline{MCLR}$ Reset during Power-Managed Run Modes	0000h	u <sup>(2)</sup>	u	1	u	u	u	u	u
$\overline{MCLR}$ Reset during Power-Managed Idle Modes and Sleep Mode	0000h	u <sup>(2)</sup>	u	1	0	u	u	u	u
WDT Time-out during Full-Power or Power-Managed Run Modes	0000h	u <sup>(2)</sup>	u	0	u	u	u	u	u
$\overline{MCLR}$ Reset during Full-Power Execution	0000h	u <sup>(2)</sup>	u	u	u	u	u	u	u
Stack Full Reset (STVREN = 1)	0000h	u <sup>(2)</sup>	u	u	u	u	u	1	u
Stack Underflow Reset (STVREN = 1)	0000h	u <sup>(2)</sup>	u	u	u	u	u	u	1
Stack Underflow Error (not an actual Reset, STVREN = 0)	0000h	u <sup>(2)</sup>	u	u	u	u	u	u	1
WDT Time-out during Power-Managed Idle or Sleep Modes	PC + 2	u <sup>(2)</sup>	u	0	0	u	u	u	u
Interrupt Exit from Power-Managed Modes	PC + 2 <sup>(1)</sup>	u <sup>(2)</sup>	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'.

# PIC18F6310/6410/8310/8410

**TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS**

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets	Wake-up via WDT or Interrupt
	6X10	8X10		WDT Reset RESET Instruction Stack Resets	
TOSU	6X10	8X10	---0 0000	---0 0000	---0 uuuu <sup>(3)</sup>
TOSH	6X10	8X10	0000 0000	0000 0000	uuuu uuuu <sup>(3)</sup>
TOSL	6X10	8X10	0000 0000	0000 0000	uuuu uuuu <sup>(3)</sup>
STKPTR	6X10	8X10	uu-0 0000	00-0 0000	uu-u uuuu <sup>(3)</sup>
PCLATU	6X10	8X10	---0 0000	---0 0000	---u uuuu
PCLATH	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
PCL	6X10	8X10	0000 0000	0000 0000	PC + 2 <sup>(2)</sup>
TBLPTRU	6X10	8X10	--00 0000	--00 0000	--uu uuuu
TBLPTRH	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
TABLAT	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
PRODH	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	6X10	8X10	0000 000x	0000 000u	uuuu uuuu <sup>(1)</sup>
INTCON2	6X10	8X10	1111 1111	1111 1111	uuuu uuuu <sup>(1)</sup>
INTCON3	6X10	8X10	1100 0000	1100 0000	uuuu uuuu <sup>(1)</sup>
INDF0	6X10	8X10	N/A	N/A	N/A
POSTINC0	6X10	8X10	N/A	N/A	N/A
POSTDEC0	6X10	8X10	N/A	N/A	N/A
PREINC0	6X10	8X10	N/A	N/A	N/A
PLUSW0	6X10	8X10	N/A	N/A	N/A
FSR0H	6X10	8X10	---- xxxx	---- uuuu	---- uuuu
FSR0L	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	6X10	8X10	N/A	N/A	N/A
POSTINC1	6X10	8X10	N/A	N/A	N/A
POSTDEC1	6X10	8X10	N/A	N/A	N/A
PREINC1	6X10	8X10	N/A	N/A	N/A
PLUSW1	6X10	8X10	N/A	N/A	N/A
FSR1H	6X10	8X10	---- xxxx	---- uuuu	---- uuuu
FSR1L	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	6X10	8X10	---- 0000	---- 0000	---- 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'.

# PIC18F6310/6410/8310/8410

**TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
INDF2	6X10	8X10	N/A	N/A	N/A
POSTINC2	6X10	8X10	N/A	N/A	N/A
POSTDEC2	6X10	8X10	N/A	N/A	N/A
PREINC2	6X10	8X10	N/A	N/A	N/A
PLUSW2	6X10	8X10	N/A	N/A	N/A
FSR2H	6X10	8X10	---- xxxx	---- uuuu	---- uuuu
FSR2L	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	6X10	8X10	---x xxxx	---u uuuu	---u uuuu
TMR0H	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
TMR0L	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	6X10	8X10	1111 1111	1111 1111	uuuu uuuu
OSCCON	6X10	8X10	0100 q000	0100 00q0	uuuu uuqu
HLVDCON	6X10	8X10	0-00 0101	0-00 0101	u-uu uuuu
WDTCON	6X10	8X10	---- ---0	---- ---0	---- ---u
RCON <sup>(4)</sup>	6X10	8X10	0q-1 11q0	0q-q qquu	uq-u qquu
TMR1H	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	6X10	8X10	0000 0000	uuuu uuuu	uuuu uuuu
T1CON	6X10	8X10	0000 0000	u0uu uuuu	uuuu uuuu
TMR2	6X10	8X10	1111 1111	0000 0000	uuuu uuuu
PR2	6X10	8X10	-000 0000	-111 1111	-111 1111
T2CON	6X10	8X10	-000 0000	-000 0000	-uuu uuuu
SSPBUF	6X10	8X10	0000 0000	uuuu uuuu	uuuu uuuu
SSPADD	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
SSPCON1	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
SSPCON2	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
ADRESH	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	6X10	8X10	0000 0000	uuuu uuuu	uuuu uuuu
ADCON0	6X10	8X10	--00 0000	--00 0000	--uu uuuu
ADCON1	6X10	8X10	--00 qqqq	--00 0000	--uu uuuu
ADCON2	6X10	8X10	0-00 0000	0-00 0000	u-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).
- Note 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).
- Note 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.
- Note 4:** See [Table 5-3](#) for Reset value for specific condition.
- Note 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'.



# PIC18F6310/6410/8310/8410

**TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
CCPR1H	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	6X10	8X10	--00 0000	--00 0000	--uu uuuu
CCPR2H	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	6X10	8X10	0000 0000	uuuu uuuu	uuuu uuuu
CCP2CON	6X10	8X10	--00 0000	--00 0000	--uu uuuu
CCPR3H	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR3L	6X10	8X10	0000 0000	uuuu uuuu	uuuu uuuu
CCP3CON	6X10	8X10	--00 0000	--00 0000	--uu uuuu
CVRCON	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
CMCON	6X10	8X10	0000 0111	0000 0111	uuuu uuuu
TMR3H	6X10	8X10	0000 0000	uuuu uuuu	uuuu uuuu
TMR3L	6X10	8X10	0000 0000	uuuu uuuu	uuuu uuuu
T3CON	6X10	8X10	0000 0000	uuuu uuuu	uuuu uuuu
PSPCON	6X10	8X10	0000 ----	0000 ----	uuuu ----
SPBRG1	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
RCREG1	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
TXREG1	6X10	8X10	xxxx xxxx	0000 0000	uuuu uuuu
TXSTA1	6X10	8X10	0000 0010	0000 0010	uuuu uuuu
RCSTA1	6X10	8X10	0000 000x	0000 000x	uuuu uuuu
IPR3	6X10	8X10	--11 ---1	--11 ---1	--uu ---u
PIR3	6X10	8X10	--00 ---0	--00 ---0	--uu ---u <sup>(1)</sup>
PIE3	6X10	8X10	--00 ---0	--00 ---0	--uu ---u
IPR2	6X10	8X10	11-- 1111	11-- 1111	uu-- uuuu
PIR2	6X10	8X10	00-- 0000	00-- 0000	uu-- uuuu <sup>(1)</sup>
PIE2	6X10	8X10	00-- 0000	00-- 0000	uu-- uuuu
IPR1	6X10	8X10	1111 1111	1111 1111	uuuu uuuu
PIR1	6X10	8X10	0000 0000	0000 0000	uuuu uuuu <sup>(1)</sup>
PIE1	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
MEMCON	6X10	8X10	0-00 --00	0-00 --00	u-uu --uu
OSCTUNE	6X10	8X10	00-0 0000	00-0 0000	uu-u uuuu
TRISJ	6X10	8X10	1111 1111	1111 1111	uuuu uuuu
TRISH	6X10	8X10	1111 1111	1111 1111	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'.

# PIC18F6310/6410/8310/8410

**TABLE 5-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
TRISG	6X10	8X10	---1 1111	---1 1111	---u uuuu
TRISF	6X10	8X10	1111 1111	1111 1111	uuuu uuuu
TRISE	6X10	8X10	1111 1111	1111 1111	uuuu uuuu
TRISD	6X10	8X10	1111 1111	1111 1111	uuuu uuuu
TRISC	6X10	8X10	1111 1111	1111 1111	uuuu uuuu
TRISB	6X10	8X10	1111 1111	1111 1111	uuuu uuuu
TRISA <sup>(5)</sup>	6X10	8X10	1111 1111 <sup>(5)</sup>	1111 1111 <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>
LATJ	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATH	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATG	6X10	8X10	---x xxxx	---u uuuu	---u uuuu
LATF	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATE	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATD	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA <sup>(5)</sup>	6X10	8X10	xxxx xxxx <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>
PORTJ	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTH	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTG	6X10	8X10	--xx xxxx	--uu uuuu	--uu uuuu
PORTF	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTD	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	6X10	8X10	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA <sup>(5)</sup>	6X10	8X10	xx0x 0000 <sup>(5)</sup>	uu0u 0000 <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>
SPBRGH1	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
BAUDCON1	6X10	8X10	0100 0-00	0100 0-00	uuuu u-uu
SPBRG2	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
RCREG2	6X10	8X10	0000 0000	0000 0000	uuuu uuuu
TXREG2	6X10	8X10	xxxx xxxx	0000 0000	uuuu uuuu
TXSTA2	6X10	8X10	0000 -010	0000 -010	uuuu -uuu
RCSTA2	6X10	8X10	0000 000x	0000 000x	uuuu uuuu

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', α = 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'.

# PIC18F6310/6410/8310/8410

## 6.0 MEMORY ORGANIZATION

There are two types of memory in PIC18 Flash microcontroller devices:

- Program Memory
- Data RAM

As Harvard architecture devices, the data and program memories use separate busses; this allows for concurrent access of the two memory spaces.

Additional detailed information on the operation of the Flash program memory is provided in [Section 7.0 “Program Memory”](#).

## 6.1 Program Memory Organization

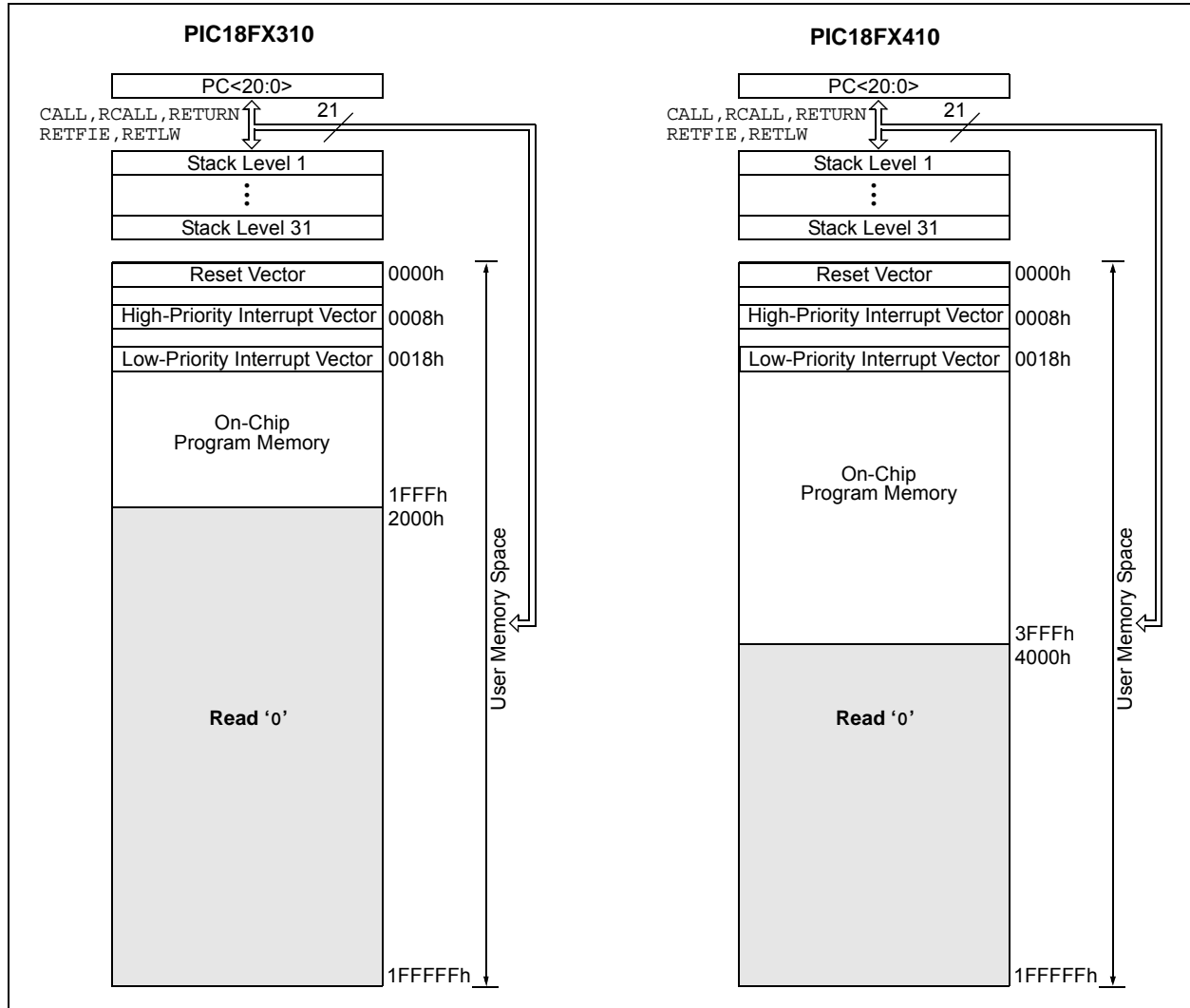
PIC18 microcontrollers implement a 21-bit program counter, which is capable of addressing a 2-Mbyte program memory space. Accessing a location between the upper boundary of the physically implemented memory and the 2-Mbyte address will return all ‘0’s (a NOP instruction).

The PIC18F6310 and PIC18F8310 each have 8 Kbytes of Flash memory and can store up to 4,096 single-word instructions. The PIC18F6410 and PIC18F8410 each have 16 Kbytes of Flash memory and can store up to 8,192 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 the PIC18F6310/6410/8310/8410 devices are shown in [Figure 6-1](#).

**FIGURE 6-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F6310/6410/8310/8410 DEVICES**



# PIC18F6310/6410/8310/8410

## 6.1.1 PIC18F8310/8410 PROGRAM MEMORY MODES

In addition to available on-chip Flash program memory, 80-pin devices in this family can also address up to 2 Mbytes of external program memory through an external memory interface. There are four distinct operating modes available to the controllers:

- Microprocessor (MP)
- Microprocessor with Boot Block (MPBB)
- Extended Microcontroller (EMC)
- Microcontroller (MC)

The program memory mode is determined by setting the two Least Significant bits of the CONFIG3L Configuration byte, as shown in [Register 6-1](#). (See also [Section 24.1 “Configuration Bits”](#) for additional details on the device Configuration bits.)

The program memory modes operate as follows:

- The **Microcontroller Mode** accesses only on-chip Flash memory. Attempts to read above the physical limit of the on-chip Flash (3FFFh) causes a read of all '0's (a NOP instruction). The Microcontroller mode is also the only operating mode available to PIC18F6310 and PIC18F6410 devices.

- The **Extended Microcontroller Mode** allows access to both internal and external program memories as a single block. The device can access its entire on-chip Flash memory; above this, the device accesses external program memory up to the 2-Mbyte program space limit. As with Boot Block mode, execution automatically switches between the two memories as required.
- The **Microprocessor Mode** permits access only to external program memory; the contents of the on-chip Flash memory is ignored. The 21-bit program counter permits access to the entire 2-Mbyte linear program memory space.
- The **Microprocessor with Boot Block Mode** accesses on-chip Flash memory from addresses 000000h to 0007FFh. Above this, external program memory is accessed all the way up to the 2-Mbyte limit. Program execution automatically switches between the two memories as required.

In all modes, the microcontroller has complete access to data RAM.

[Figure 6-2](#) compares the memory maps of the different program memory modes. The differences between on-chip and external memory access limitations are more fully explained in [Table 6-1](#).

**REGISTER 6-1: CONFIG3L: CONFIGURATION BYTE REGISTER LOW**

R/P-1	R/P-1	U-0	U-0	U-0	U-0	R/P-1	R/P-1
WAIT	BW	—	—	—	—	PM1	PM0
bit 7						bit 0	

**Legend:**

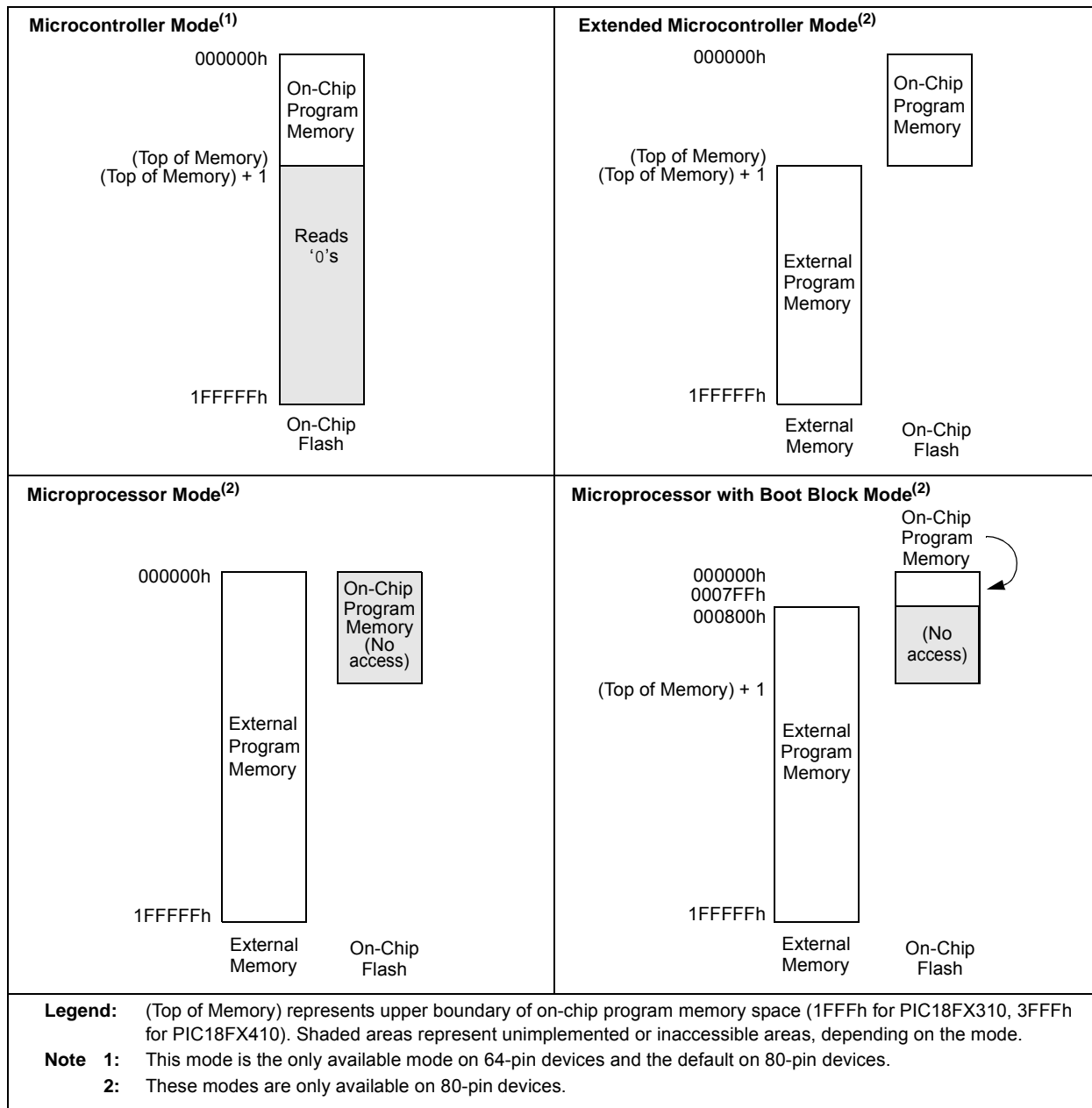
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
-n = Value after erase bit	'1' = Bit is set	'0' = Bit is cleared
		x = Bit is unknown

- bit 7 **WAIT:** External Bus Data Wait Enable bit
  - 1 = Wait selections unavailable, device will not wait
  - 0 = Wait programmed by WAIT1 and WAIT0 bits of MEMCOM register (MEMCOM<5:4>)
- bit 6 **BW:** External Bus Data Width Select bit
  - 1 = 16-bit external bus data width
  - 0 = 8-bit external bus data width
- bit 5-2 **Unimplemented:** Read as '0'
- bit 1-0 **PM<1:0>:** Processor Data Memory Mode Select bits
  - 11 = Microcontroller mode
  - 10 = Microprocessor mode<sup>(1)</sup>
  - 01 = Microcontroller with Boot Block mode<sup>(1)</sup>
  - 00 = Extended Microcontroller mode<sup>(1)</sup>

**Note 1:** This mode is available only on PIC18F8410 devices.

# PIC18F6310/6410/8310/8410

**FIGURE 6-2: MEMORY MAPS FOR PIC18FX310/X410 PROGRAM MEMORY MODES**



**TABLE 6-1: MEMORY ACCESS FOR PIC18F8310/8410 PROGRAM MEMORY MODES**

Operating Mode	Internal Program Memory			External Program Memory		
	Execution From	Table Read From	Table Write To	Execution From	Table Read From	Table Write To
Microcontroller	Yes	Yes	Yes	No Access	No Access	No Access
Extended Microcontroller	Yes	Yes	Yes	Yes	Yes	Yes
Microprocessor	No Access	No Access	No Access	Yes	Yes	Yes
Microprocessor w/Boot Block	Yes	Yes	Yes	Yes	Yes	Yes

# PIC18F6310/6410/8310/8410

## 6.1.2 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 6.1.5.1 “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, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

## 6.1.3 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 register, 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 (TOS) 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.

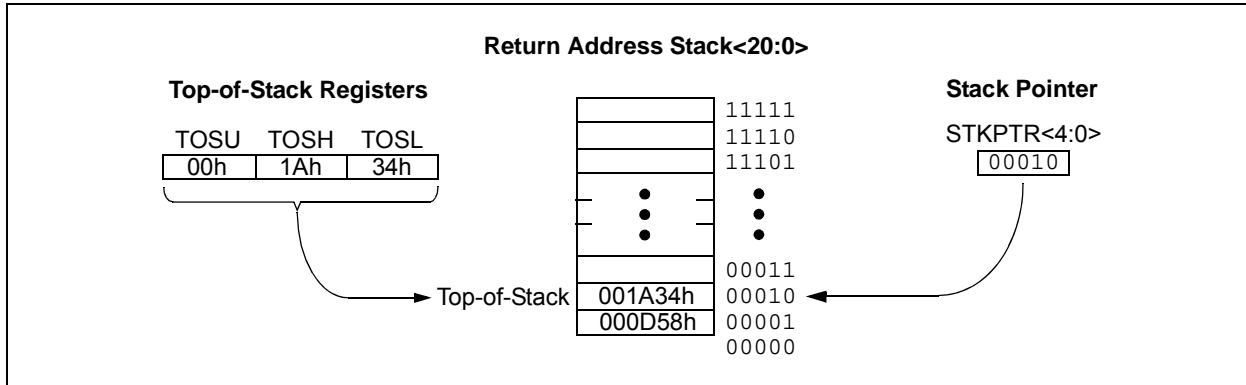
The Stack Pointer is initialized to ‘00000’ after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of ‘00000’; this is only a Reset value. Status bits indicate if the stack is full, has overflowed or has underflowed.

### 6.1.3.1 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 6-3](#)). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH: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 6-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**



# PIC18F6310/6410/8310/8410

## 6.1.3.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 6-2) contains the Stack Pointer value, the STKFUL (Stack Full) status bit and the STKUNF (Stack Underflow) status bit. 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 24.1 “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.

## 6.1.3.3 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 6-2: 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 <sup>(1)</sup>	STKUNF <sup>(1)</sup>	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

### Legend:

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

- bit 7      **STKFUL:** Stack Full Flag bit<sup>(1)</sup>  
1 = Stack became full or overflowed  
0 = Stack has not become full or overflowed
- bit 6      **STKUNF:** Stack Underflow Flag bit<sup>(1)</sup>  
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.

# PIC18F6310/6410/8310/8410

## 6.1.3.4 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 condition 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.

## 6.1.4 FAST REGISTER STACK

A Fast Register Stack is provided for the STATUS, WREG and BSR registers to provide a “fast return” option for interrupts. This 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 the working 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
```

## 6.1.5 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

### 6.1.5.1 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

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

### 6.1.5.2 Table Reads

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 while programming. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from the program memory. Data is transferred from program memory one byte at a time.

Table read operation is discussed further in Section 7.1 “Table Reads and Table Writes”.



# PIC18F6310/6410/8310/8410

## 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 is incremented on every Q1; the instruction is fetched from the program memory and latched into the instruction register during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 6-4.

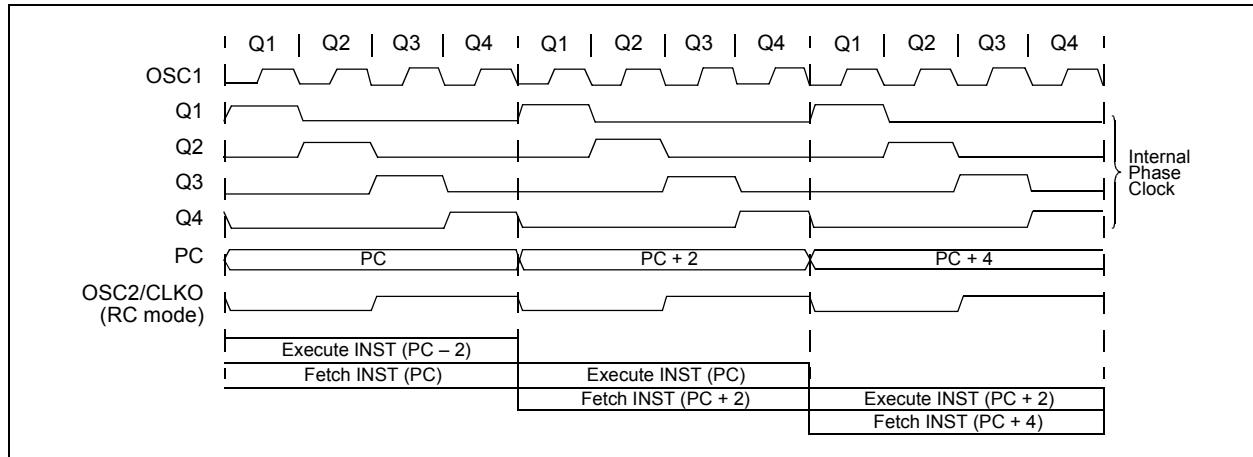
### 6.2.2 INSTRUCTION FLOW/PIPELINING

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).

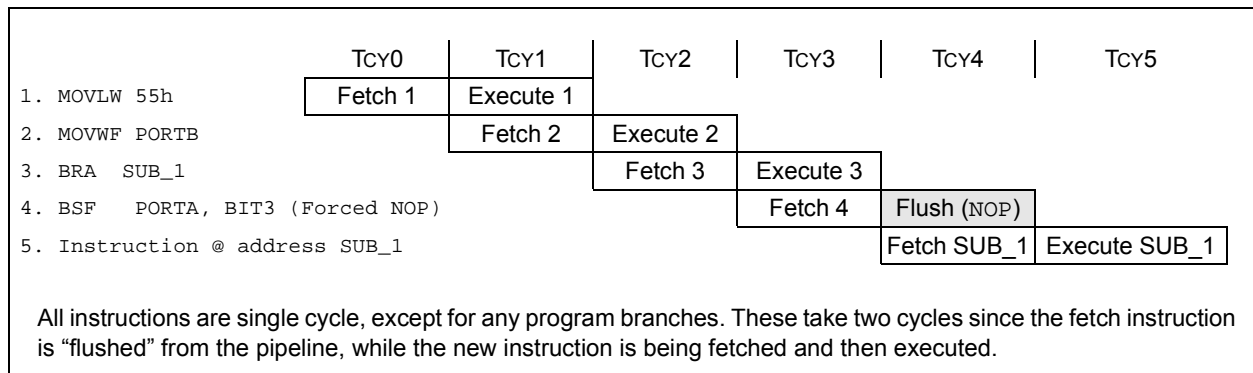
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

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

**FIGURE 6-4: CLOCK/INSTRUCTION CYCLE**



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



# PIC18F6310/6410/8310/8410

## 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 6.1.2 "Program Counter"](#)).

[Figure 6-5](#) 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 6-5](#) 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 25.0 "Instruction Set Summary"](#) provides further details of the instruction set.

**FIGURE 6-5: INSTRUCTIONS IN PROGRAM MEMORY**

Program Memory Byte Locations →			Word Address		
			LSB = 1	LSB = 0	
				000000h	
				000002h	
				000004h	
				000006h	
Instruction 1:	MOVLW	055h	0Fh	55h	000008h
Instruction 2:	GOTO	0006h	EFh	03h	00000Ah
			F0h	00h	00000Ch
			C1h	23h	00000Eh
			F4h	56h	000010h
					000012h
					000014h

## 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 "Data 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

## 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 6.5 “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. PIC18F6310/6410/8310/8410 devices implement only 3 complete banks, for a total of 768 bytes. [Figure 6-6](#) shows the data memory organization for the 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 section.

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

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](#).

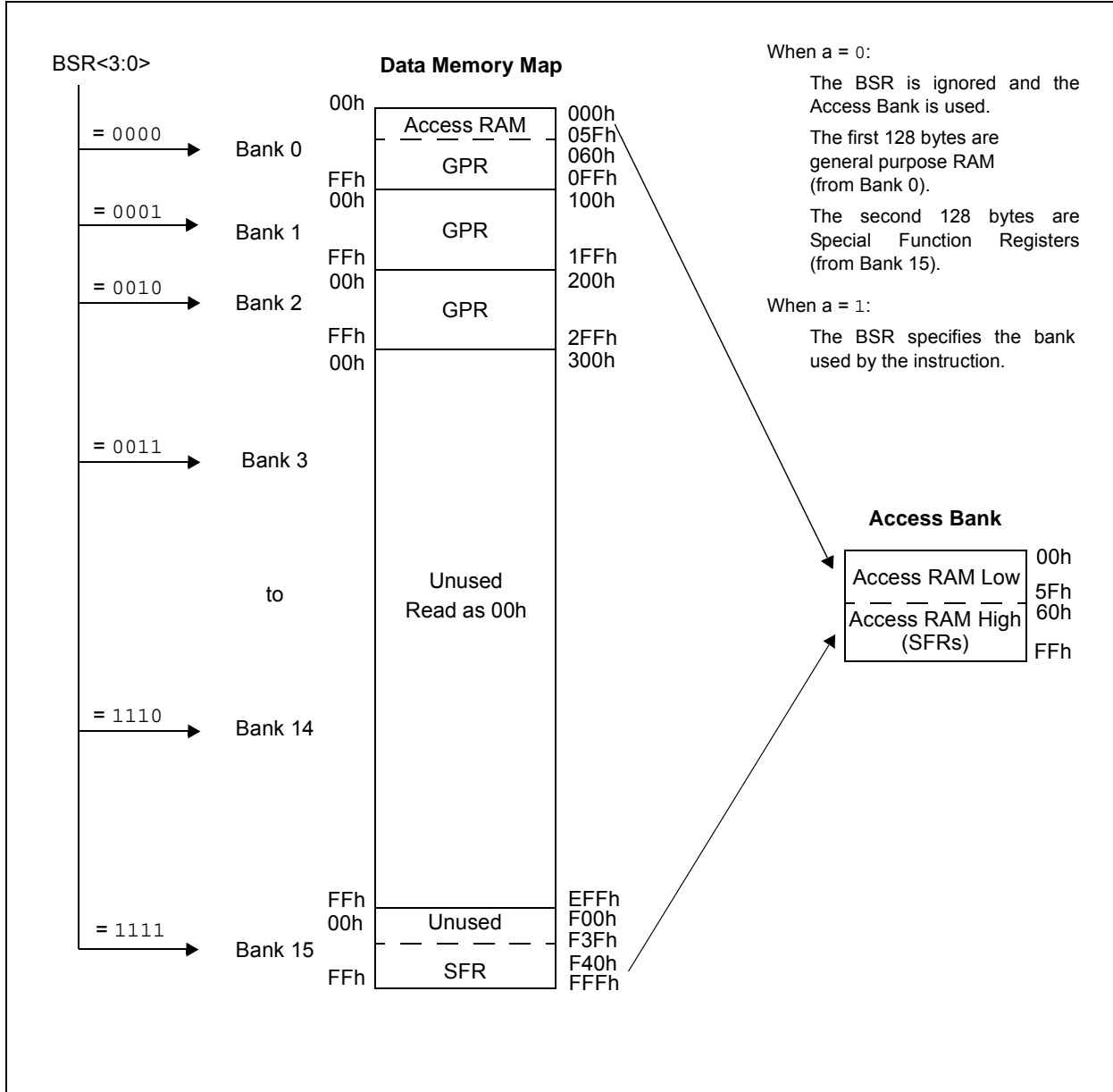
Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 0Fh will end up resetting the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

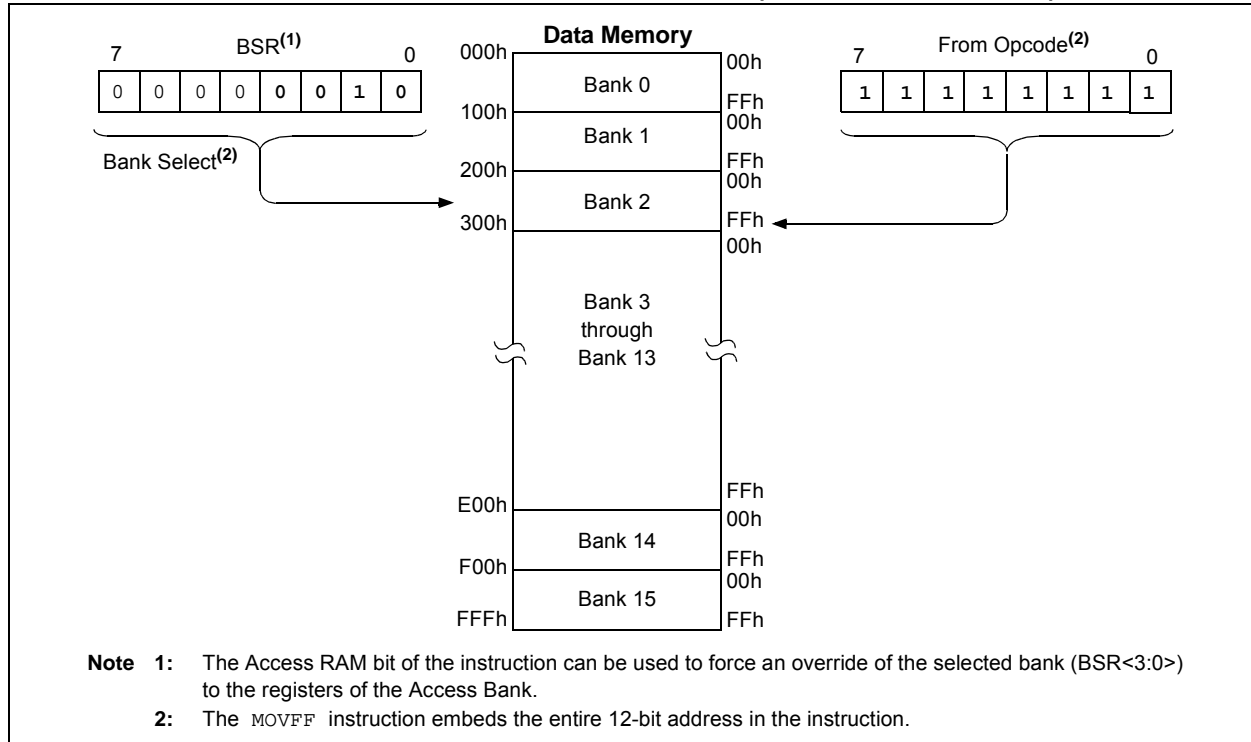
In the core PIC18 instruction set, only the `MOVFF` instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the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

# PIC18F6310/6410/8310/8410

**FIGURE 6-6: DATA MEMORY MAP FOR PIC18F6310/6410/8310/8410 DEVICES**



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



## 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 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Block 15. The lower half is known as the “Access RAM” and is composed of GPRs. This 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 60h 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 6.5.3 “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.

# PIC18F6310/6410/8310/8410

## 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 more than the top half of Bank 15 (F60h to FFFh). A list of these registers is given in [Table 6-2](#) and [Table 6-3](#).

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 the peripheral features 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 6-2: SPECIAL FUNCTION REGISTER MAP FOR PIC18F6310/6410/8310/8410 DEVICES**

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDh	INDF2 <sup>(1)</sup>	FBFh	CCPR1H	F9Fh	IPR1	F7Fh	SPBRGH1
FFEh	TOSH	FDEh	POSTINC2 <sup>(1)</sup>	FBEh	CCPR1L	F9Eh	PIR1	F7Eh	BAUDCON1
FFDh	TOSL	FDDh	POSTDEC2 <sup>(1)</sup>	FBDh	CCP1CON	F9Dh	PIE1	F7Dh	__ <sup>(2)</sup>
FFCh	STKPTR	FDCh	PREINC2 <sup>(1)</sup>	FBCh	CCPR2H	F9Ch	MEMCON <sup>(3)</sup>	F7Ch	__ <sup>(2)</sup>
FFBh	PCLATU	FDBh	PLUSW2 <sup>(1)</sup>	FBh	CCPR2L	F9Bh	OSCTUNE	F7Bh	__ <sup>(2)</sup>
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	TRISJ <sup>(3)</sup>	F7Ah	__ <sup>(2)</sup>
FF9h	PCL	FD9h	FSR2L	FB9h	CCPR3H	F99h	TRISH <sup>(3)</sup>	F79h	__ <sup>(2)</sup>
FF8h	TBLPTRU	FD8h	STATUS	FB8h	CCPR3L	F98h	TRISG	F78h	__ <sup>(2)</sup>
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	CCP3CON	F97h	TRISF	F77h	__ <sup>(2)</sup>
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	__ <sup>(2)</sup>	F96h	TRISE	F76h	__ <sup>(2)</sup>
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD	F75h	__ <sup>(2)</sup>
FF4h	PRODH	FD4h	__ <sup>(2)</sup>	FB4h	CMCON	F94h	TRISC	F74h	__ <sup>(2)</sup>
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB	F73h	__ <sup>(2)</sup>
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA	F72h	__ <sup>(2)</sup>
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ <sup>(3)</sup>	F71h	__ <sup>(2)</sup>
FF0h	INTCON3	FD0h	RCON	FB0h	PSPCON	F90h	LATH <sup>(3)</sup>	F70h	__ <sup>(2)</sup>
FEFh	INDF0 <sup>(1)</sup>	FCFh	TMR1H	FAFh	SPBRG1	F8Fh	LATG	F6Fh	SPBRG2
FEh	POSTINC0 <sup>(1)</sup>	FCEh	TMR1L	FAEh	RCREG1	F8Eh	LATF	F6Eh	RCREG2
FEDh	POSTDEC0 <sup>(1)</sup>	FCDh	T1CON	FADh	TXREG1	F8Dh	LATE	F6Dh	TXREG2
FECh	PREINC0 <sup>(1)</sup>	FCCh	TMR2	FACH	TXSTA1	F8Ch	LATD	F6Ch	TXSTA2
FEb	PLUSW0 <sup>(1)</sup>	FCBh	PR2	FABh	RCSTA1	F8Bh	LATC	F6Bh	RCSTA2
FEAh	FSR0H	FCAh	T2CON	FAAh	__ <sup>(2)</sup>	F8Ah	LATB	F6Ah	__ <sup>(2)</sup>
FE9h	FSR0L	FC9h	SSPBUF	FA9h	__ <sup>(2)</sup>	F89h	LATA	F69h	__ <sup>(2)</sup>
FE8h	WREG	FC8h	SSPADD	FA8h	__ <sup>(2)</sup>	F88h	PORTJ <sup>(3)</sup>	F68h	__ <sup>(2)</sup>
FE7h	INDF1 <sup>(1)</sup>	FC7h	SSPSTAT	FA7h	__ <sup>(2)</sup>	F87h	PORTH <sup>(3)</sup>	F67h	__ <sup>(2)</sup>
FE6h	POSTINC1 <sup>(1)</sup>	FC6h	SSPCON1	FA6h	__ <sup>(2)</sup>	F86h	PORTG	F66h	__ <sup>(2)</sup>
FE5h	POSTDEC1 <sup>(1)</sup>	FC5h	SSPCON2	FA5h	IPR3	F85h	PORTF	F65h	__ <sup>(2)</sup>
FE4h	PREINC1 <sup>(1)</sup>	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE	F64h	__ <sup>(2)</sup>
FE3h	PLUSW1 <sup>(1)</sup>	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD	F63h	__ <sup>(2)</sup>
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC	F62h	__ <sup>(2)</sup>
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB	F61h	__ <sup>(2)</sup>
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA	F60h	__ <sup>(2)</sup>

- Note** 1: This is not a physical register.  
 2: Unimplemented registers are read as ‘0’.  
 3: This register is not available on 64-pin devices.

# PIC18F6310/6410/8310/8410

**TABLE 6-3: REGISTER FILE SUMMARY (PIC18F6310/6410/8310/8410)**

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	63, 70
TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	63, 70
TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	63, 70
STKPTR	STKFUL <sup>(6)</sup>	STKUNF <sup>(6)</sup>	—	Return Stack Pointer					00-0 0000	63, 71
PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000	63, 70
PCLATH	Holding Register for PC<15:8>								0000 0000	63, 70
PCL	PC Low Byte (PC<7:0>)								0000 0000	63, 70
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--00 0000	63, 93
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	63, 93
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	63, 93
TABLAT	Program Memory Table Latch								0000 0000	63, 93
PRODH	Product Register High Byte								xxxx xxxx	63, 107
PRODL	Product Register Low Byte								xxxx xxxx	63, 107
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	63, 111
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	1111 1111	63, 112
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	1100 0000	63, 113
INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								N/A	63, 85
POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								N/A	63, 85
POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								N/A	63, 85
PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								N/A	63, 85
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	63, 85
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte				---- xxxxx	63, 85
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxxx	63, 85
WREG	Working Register								xxxx xxxxx	63
INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								N/A	63, 85
POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								N/A	63, 85
POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								N/A	63, 85
PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								N/A	63, 85
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	63, 85
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte				---- xxxxx	63, 85
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxxx	63, 85
BSR	—	—	—	—	Bank Select Register				---- 0000	63, 75
INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								N/A	64, 85
POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								N/A	64, 85
POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								N/A	64, 85
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								N/A	64, 85
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	64, 85
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte				---- xxxxx	64, 85
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxxx	64, 85
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxxx	64, 83

**Legend:** x = unknown, u = unchanged, - = unimplemented, c = value depends on condition. Shaded locations are unimplemented, read as '0'.

- Note 1:** The SBOREN bit is only available when the BOREN<1:0> Configuration bits = 01; otherwise, it is disabled and reads as '0'. See [Section 5.4 “Brown-out Reset \(BOR\)”](#).
- 2:** These registers and/or bits are not implemented on 64-pin devices, read as '0'.
- 3:** The PLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See [Section 3.6.4 “PLL in INTOSC Modes”](#).
- 4:** The RG5 bit is only available when Master Clear is disabled (MCLRE Configuration bit = 0); otherwise, RG5 reads as '0'. This bit is read-only.
- 5:** 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'.
- 6:** STKFUL and STKUNF bits are cleared by user software or by a POR.

# PIC18F6310/6410/8310/8410

**TABLE 6-3: REGISTER FILE SUMMARY (PIC18F6310/6410/8310/8410) (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:
TMR0H	Timer0 Register High Byte								0000 0000	64, 153
TMR0L	Timer0 Register Low Byte								xxxx xxxx	64, 153
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	64, 151
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0	0100 q000	42, 64
HLVDCON	VDIRMAG	—	IRVST	HLVDEN	HLVDL3	HLVDL2	HLVDL1	HLVDL0	0-00 0101	64, 275
WDTCON	—	—	—	—	—	—	—	SWDTEN	---- --0	64, 291
RCON	IPEN	SBOREN <sup>(1)</sup>	—	RI	TO	PD	POR	BOR	0q-1 11q0	56, 64, 123
TMR1H	Timer1 Register High Byte								xxxx xxxx	64, 159
TMR1L	Timer1 Register Low Byte								0000 0000	64, 159
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0000 0000	64, 155
TMR2	Timer2 Register								1111 1111	64, 162
PR2	Timer2 Period Register								-000 0000	64, 162
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	64, 161
SSPBUF	MSSP Receive Buffer/Transmit Register								0000 0000	64, 178, 186
SSPADD	MSSP Address Register in I <sup>2</sup> C™ Slave Mode. MSSP Baud Rate Reload Register in I <sup>2</sup> C Master Mode.								0000 0000	64, 186
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000	64, 178, 188
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	64, 179, 179
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	64, 189
ADRESH	A/D Result Register High Byte								xxxx xxxx	64, 264
ADRESL	A/D Result Register Low Byte								0000 0000	64, 264
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	--00 0000	64, 255
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 qqqq	64, 256
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	64, 257
CCPR1H	Capture/Compare/PWM Register 1 High Byte								xxxx xxxx	65, 168
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx	65, 168
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	65, 167
CCPR2H	Capture/Compare/PWM Register 2 High Byte								xxxx xxxx	65, 168
CCPR2L	Capture/Compare/PWM Register 2 Low Byte								0000 0000	65, 168
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	65, 167
CCPR3H	Capture/Compare/PWM Register 3 High Byte								xxxx xxxx	65, 168
CCPR3L	Capture/Compare/PWM Register 3 Low Byte								0000 0000	65, 168
CCP3CON	—	—	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	--00 0000	65, 167
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	65, 271
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	65, 265
TMR3H	Timer3 Register High Byte								0000 0000	65, 163
TMR3L	Timer3 Register Low Byte								0000 0000	65, 165
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	0000 0000	65, 163
PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—	0000 ----	65, 149

**Legend:** x = unknown, u = unchanged, - = unimplemented, q = value depends on condition. Shaded locations are unimplemented, read as '0'.

- Note**
- 1: The SBOREN bit is only available when the BOREN<1:0> Configuration bits = 01; otherwise, it is disabled and reads as '0'. See [Section 5.4 "Brown-out Reset \(BOR\)"](#).
  - 2: These registers and/or bits are not implemented on 64-pin devices, read as '0'.
  - 3: The PLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See [Section 3.6.4 "PLL in INTOSC Modes"](#).
  - 4: The RG5 bit is only available when Master Clear is disabled (MCLRE Configuration bit = 0); otherwise, RG5 reads as '0'. This bit is read-only.
  - 5: 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'.
  - 6: STKFUL and STKUNF bits are cleared by user software or by a POR.



# PIC18F6310/6410/8310/8410

**TABLE 6-3: REGISTER FILE SUMMARY (PIC18F6310/6410/8310/8410) (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:	
SPBRG1	EUSART1 Baud Rate Generator Low Byte								0000 0000	65, 221	
RCREG1	EUSART1 Receive Register								0000 0000	65, 229	
TXREG1	EUSART1 Transmit Register								xxxx xxxx	65, 226	
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	65, 218	
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	65, 219	
IPR3	—	—	RC2IP	TX2IP	—	—	—	CCP3IP	--11 ---1	65, 122	
PIR3	—	—	RC2IF	TX2IF	—	—	—	CCP3IF	--00 ---0	65, 116	
PIE3	—	—	RC2IE	TX2IE	—	—	—	CCP3IE	--00 ---0	65, 119	
IPR2	OSCFIP	CMIP	—	—	BCLIP	HLVDIP	TMR3IP	CCP2IP	11-- 1111	65, 121	
PIR2	OSCFIF	CMIF	—	—	BCLIF	HLVDIF	TMR3IF	CCP2IF	00-- 0000	65, 115	
PIE2	OSCFIE	CMIE	—	—	BCLIE	HLVDIE	TMR3IE	CCP2IE	00-- 0000	65, 118	
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	65, 120	
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	65, 114	
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	65, 117	
MEMCON <sup>(2)</sup>	EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0	0-00 --00	65, 95	
OSCTUNE	INTSRC	PLLEN <sup>(3)</sup>	—	TUN4	TUN3	TUN2	TUN1	TUN0	00-0 0000	39, 65	
TRISJ <sup>(2)</sup>	PORTJ Data Direction Register								1111 1111	65, 147	
TRISH <sup>(2)</sup>	PORTH Data Direction Register								1111 1111	65, 145	
TRISG	—	—	—	PORTG Data Direction Register				---	1111	66, 143	
TRISF	PORTF Data Direction Register								1111 1111	66, 141	
TRISE	PORTE Data Direction Register								1111 1111	66, 139	
TRISD	PORTD Data Direction Register								1111 1111	66, 136	
TRISC	PORTC Data Direction Register								1111 1111	66, 133	
TRISB	PORTB Data Direction Register								1111 1111	66, 130	
TRISA	TRISA7 <sup>(5)</sup>	TRISA6 <sup>(5)</sup>	PORTA Data Direction Register					1111	1111	66, 127	
LATJ <sup>(2)</sup>	LATJ Output Latch Register								xxxx xxxx	66, 147	
LATH <sup>(2)</sup>	LATH Output Latch Register								xxxx xxxx	66, 145	
LATG	—	—	—	LATG Output Latch Register				---	xxxx	66, 143	
LATF	LATF Output Latch Register								xxxx xxxx	66, 141	
LATE	LATE Output Latch Register								xxxx xxxx	66, 139	
LATD	LATD Output Latch Register								xxxx xxxx	66, 136	
LATC	LATC Output Latch Register								xxxx xxxx	66, 133	
LATB	LATB Output Latch Register								xxxx xxxx	66, 130	
LATA	LATA7 <sup>(5)</sup>	LATA6 <sup>(5)</sup>	LATA Output Latch Register					xxxx	xxxx	66, 127	
PORTJ <sup>(2)</sup>	Read PORTJ pins, Write PORTJ Data Latch								xxxx xxxx	66, 147	
PORTH <sup>(2)</sup>	Read PORTH pins, Write PORTH Data Latch								xxxx xxxx	66, 145	
PORTG	—	—	RG5 <sup>(4)</sup>	Read PORTG pins<4:0>, Write PORTG Data Latch<4:0>					--x	xxxx	66, 143
PORTF	Read PORTF pins, Write PORTF Data Latch								xxxx xxxx	66, 141	
PORTE	Read PORTE pins, Write PORTE Data Latch								xxxx xxxx	66, 139	
PORTD	Read PORTD pins, Write PORTD Data Latch								xxxx xxxx	66, 136	
PORTC	Read PORTC pins, Write PORTC Data Latch								xxxx xxxx	66, 133	
PORTB	Read PORTB pins, Write PORTB Data Latch								xxxx xxxx	66, 130	
PORTA	RA7 <sup>(5)</sup>	RA6 <sup>(5)</sup>	Read PORTA pins, Write PORTA Data Latch					xx0x	0000	66, 127	

- Legend:** x = unknown, u = unchanged, - = unimplemented, c = value depends on condition. Shaded locations are unimplemented, read as '0'.
- Note 1:** The SBOREN bit is only available when the BOREN<1:0> Configuration bits = 01; otherwise, it is disabled and reads as '0'. See [Section 5.4 "Brown-out Reset \(BOR\)"](#).
- 2:** These registers and/or bits are not implemented on 64-pin devices, read as '0'.
- 3:** The PLLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See [Section 3.6.4 "PLL in INTOSC Modes"](#).
- 4:** The RG5 bit is only available when Master Clear is disabled (MCLRE Configuration bit = 0); otherwise, RG5 reads as '0'. This bit is read-only.
- 5:** 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'.
- 6:** STKFUL and STKUNF bits are cleared by user software or by a POR.

# PIC18F6310/6410/8310/8410

**TABLE 6-3: REGISTER FILE SUMMARY (PIC18F6310/6410/8310/8410) (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:
SPBRGH1	EUSART1 Baud Rate Generator High Byte								0000 0000	<a href="#">66, 221</a>
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	0100 0–00	<a href="#">66, 220</a>
SPBRG2	AUSART2 Baud Rate Generator								0000 0000	<a href="#">66, 234</a>
RCREG2	AUSART2 Receive Register								0000 0000	<a href="#">66, 248</a>
TXREG2	AUSART2 Transmit Register								xxxxx xxxxx	<a href="#">66, 246</a>
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 –010	<a href="#">66, 242</a>
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	<a href="#">66, 243</a>

**Legend:** x = unknown, u = unchanged, – = unimplemented, c = value depends on condition. Shaded locations are unimplemented, read as '0'.

- Note 1:** The SBOREN bit is only available when the BOREN<1:0> Configuration bits = 01; otherwise, it is disabled and reads as '0'. See [Section 5.4 “Brown-out Reset \(BOR\)”](#).
- 2:** These registers and/or bits are not implemented on 64-pin devices, read as '0'.
- 3:** The PLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See [Section 3.6.4 “PLL in INTOSC Modes”](#).
- 4:** The RG5 bit is only available when Master Clear is disabled (MCLRE Configuration bit = 0); otherwise, RG5 reads as '0'. This bit is read-only.
- 5:** 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'.
- 6:** STKFUL and STKUNF bits are cleared by user software or by a POR.

# PIC18F6310/6410/8310/8410

## 6.3.5 STATUS REGISTER

The STATUS register, shown in [Register 6-3](#), 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 25-2](#) and [Table 25-3](#).

**Note:** The C and DC bits operate as a Borrow and Digit Borrow bit, respectively, in subtraction.

### REGISTER 6-3: 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 <sup>(1)</sup>	C <sup>(2)</sup>
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<sup>(1)</sup>  
 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<sup>(2)</sup>  
 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 2's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either bit 4 or bit 3 of the source register.

**2:** For Borrow, the polarity is reversed. A subtraction is executed by adding the 2'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.

# PIC18F6310/6410/8310/8410

## 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 6.5 “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 6.5.1 “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”](#)) 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](#). It also enables users to perform Indexed Addressing and other Stack Pointer operations for program memory in data memory.

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

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

## 6.4.3.1 FSR Registers and the INDF Operand

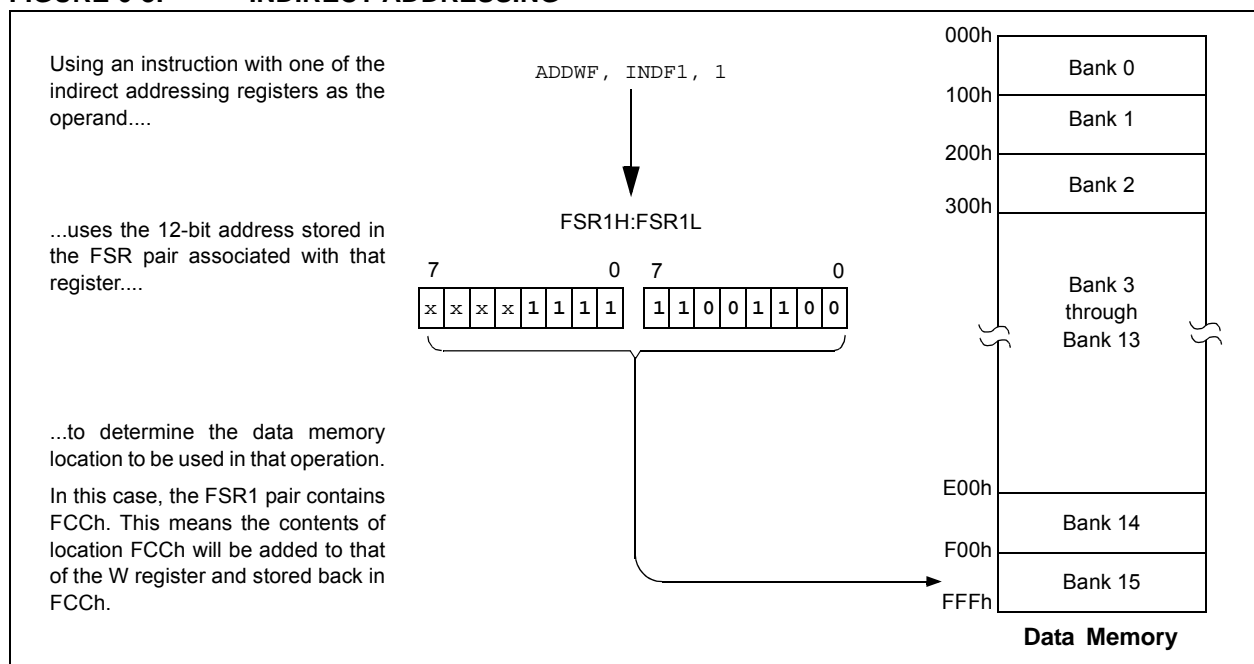
At the core of Indirect Addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers, FSRnH and FSRnL. The four upper bits of the FSRnH register are not used, so each FSR pair holds a 12-bit value. This represents a value that can address the entire range of 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**



## 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 the value 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, rollovers 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.).

# PIC18F6310/6410/8310/8410

---

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 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.

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 remain unchanged.

### 6.5.1 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.

### 6.5.2 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 do not 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 is 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 25.2.1 “Extended Instruction Syntax”](#).

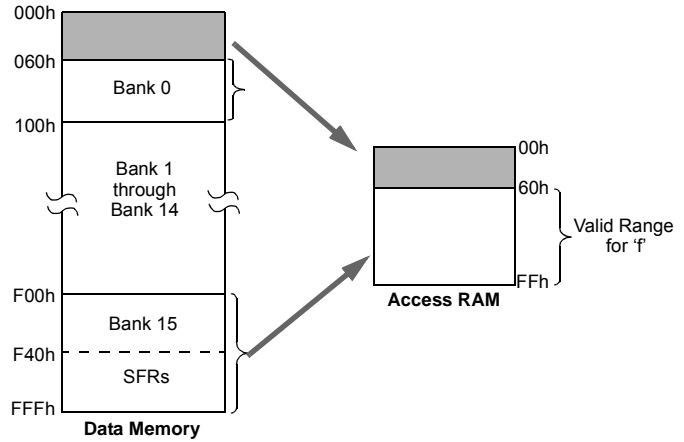
**FIGURE 6-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

**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 FFFh. This is the same as locations F60h to FFFh (Bank 15) of data memory.

Locations below 060h are not available in this addressing mode.

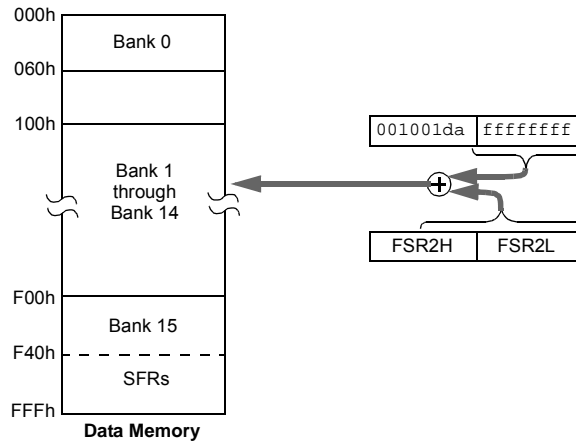


**When a = 0 and f ≤ 5Fh:**

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.

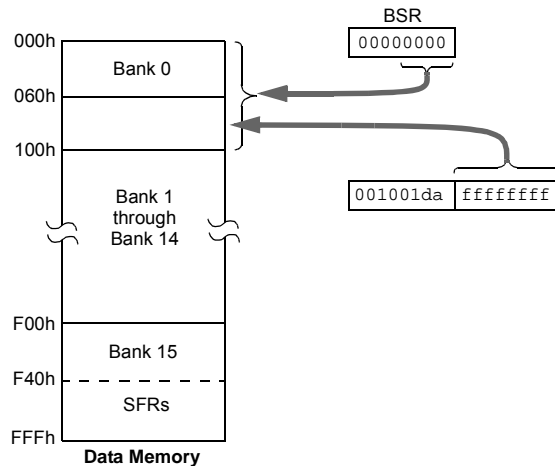
Note that in this mode, the correct syntax is now:

`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.



# PIC18F6310/6410/8310/8410

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

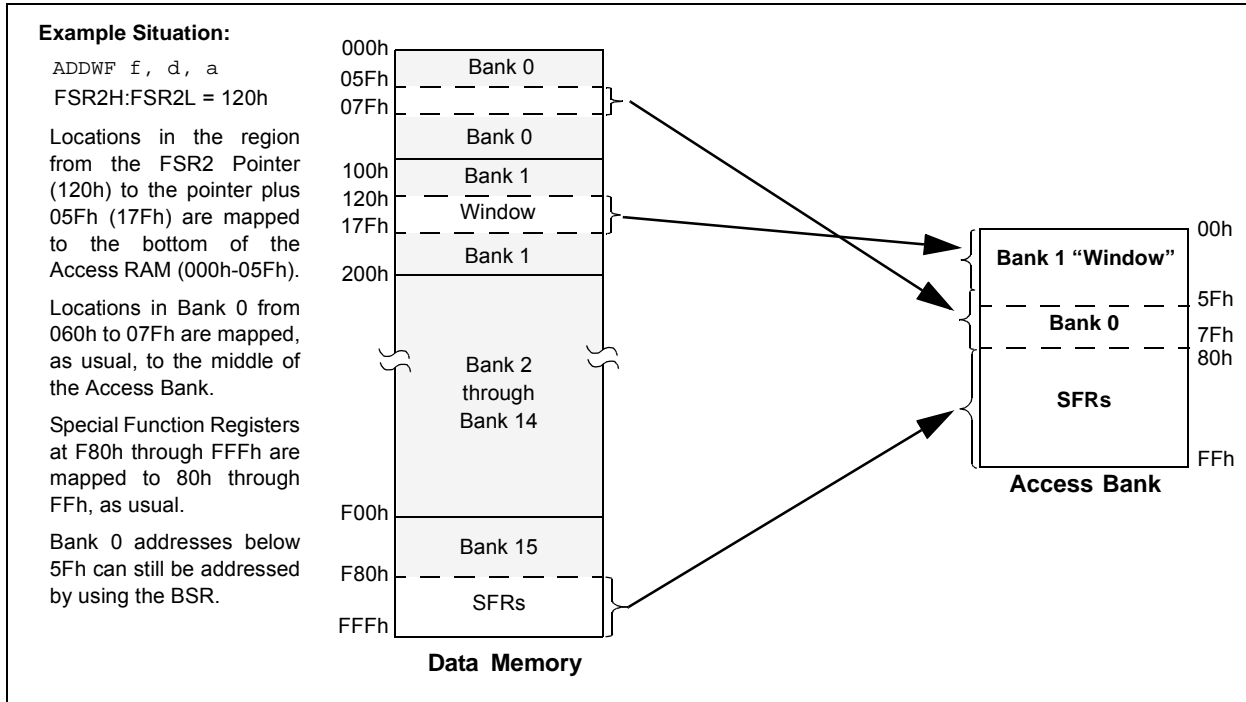
The use of Indexed Literal Offset Addressing mode effectively changes how the lower part of Access RAM (00h to 5Fh) is mapped. Rather than containing just the contents of the bottom part of Bank 0, this mode maps the contents from Bank 0 and a user-defined “window” that can be located anywhere in the 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.

## 6.6 PIC18 Instruction Execution and the Extended Instruction Set

Enabling the extended instruction set adds eight additional commands to the existing PIC18 instruction set. These instructions are executed as described in [Section 25.2 “Extended Instruction Set”](#).

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





## 7.0 PROGRAM MEMORY

For PIC18FX310/X410 devices, the on-chip program memory is implemented as read-only memory. It is readable over the entire VDD range during normal operation; it cannot be written to or erased. Reads from program memory are executed one byte at a time.

PIC18F8410 devices also implement the ability to read, write to and execute code from external memory devices using the external memory interface. In this implementation, external memory is used as all or part of the program memory space. The operation of the physical interface is discussed in [Section 8.0 “External Memory Interface”](#).

In all devices, a value written to the program memory space does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

### 7.1 Table Reads and Table Writes

To read and write to the program memory space, there are two operations that allow the processor to move bytes between the program memory space and the data RAM: table read (TBLRD) and table write (TBLWT).

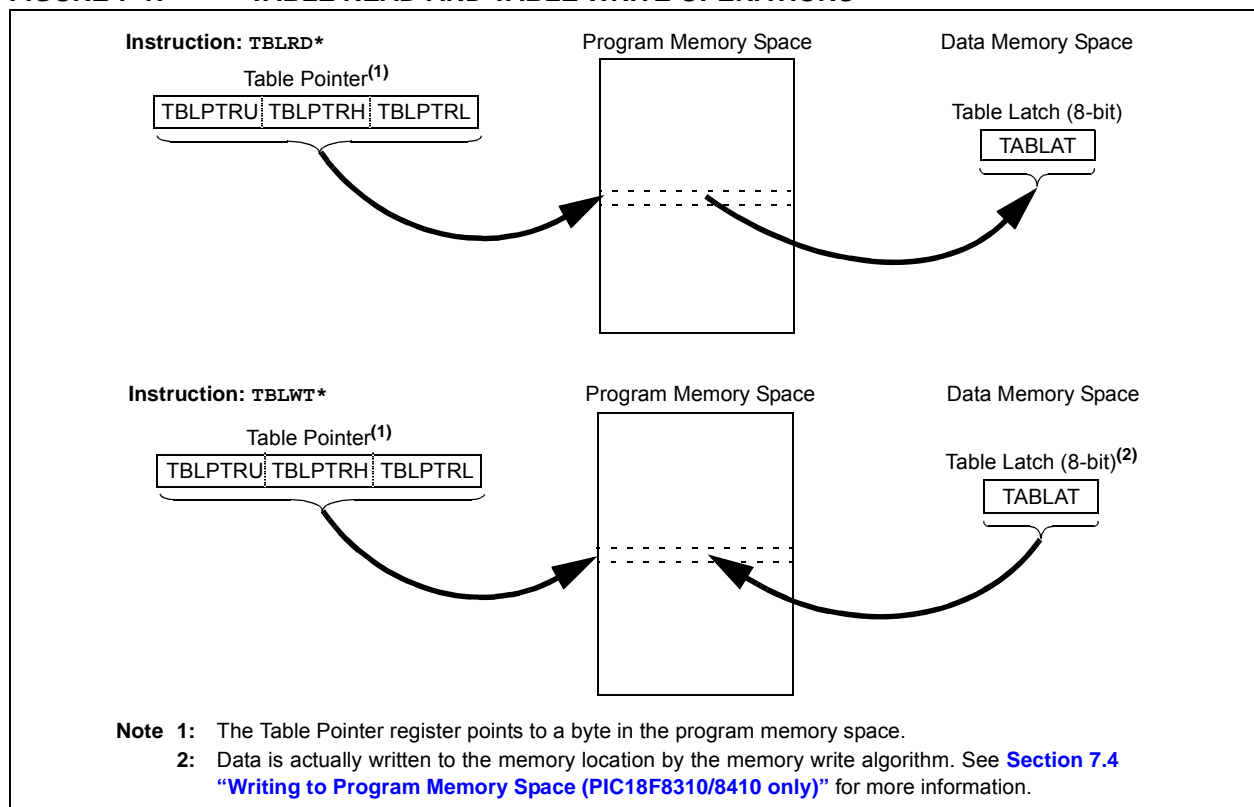
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and places it into the data RAM space. Table write operations place data from the data memory space on the external data bus. The actual process of writing the data to the particular memory device is determined by the requirements of the device itself. [Figure 7-1](#) shows the table operations as they relate to 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 an external program memory, program instructions will need to be word-aligned.

**Note:** Although it cannot be used in PIC18F6310 devices in normal operation, the TBLWT instruction is still implemented in the instruction set. Executing the instruction takes two instruction cycles, but effectively results in a NOP. The TBLWT instruction is available in programming modes and is used during In-Circuit Serial Programming (ICSP).

**FIGURE 7-1: TABLE READ AND TABLE WRITE OPERATIONS**



# PIC18F6310/6410/8310/8410

## 7.2 Control Registers

Two control registers are used in conjunction with the `TBLRD` and `TBLWT` instructions: the `TABLAT` register and the `TBLPTR` register set.

### 7.2.1 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 the program memory space and data RAM.

### 7.2.2 TBLPTR – TABLE POINTER REGISTER

The Table Pointer register (`TBLPTR`) addresses a byte within the program memory. It is comprised of three SFR registers: Table Pointer Upper Byte, Table Pointer High Byte and Table Pointer Low Byte (`TBLPTRU:TBLPTRH:TBLPTRL`). Only the lower six bits of `TBLPTRU` are used with `TBLPTRH` and `TBLPTRL` to form a 22-bit wide pointer.

The contents of `TBLPTR` indicate a location in program memory space. The low-order 21 bits allow the device to address the full 2 Mbytes of program memory space. The 22nd bit allows access to the configuration space, including the device ID, user ID locations and the Configuration bits.

The `TBLPTR` register set is updated when executing a `TBLRD` or `TBLWT` operation in one of four ways, based on the instruction's arguments. These are detailed in [Table 7-1](#). These operations on the `TBLPTR` only affect the low-order 21 bits.

When a `TBLRD` or `TBLWT` is executed, all 22 bits of the `TBLPTR` determine which address in the program memory space is to be read or written to.

**TABLE 7-1: TABLE POINTER OPERATIONS WITH `TBLRD` AND `TBLWT` INSTRUCTIONS**

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

## 7.3 Reading the Flash Program Memory

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

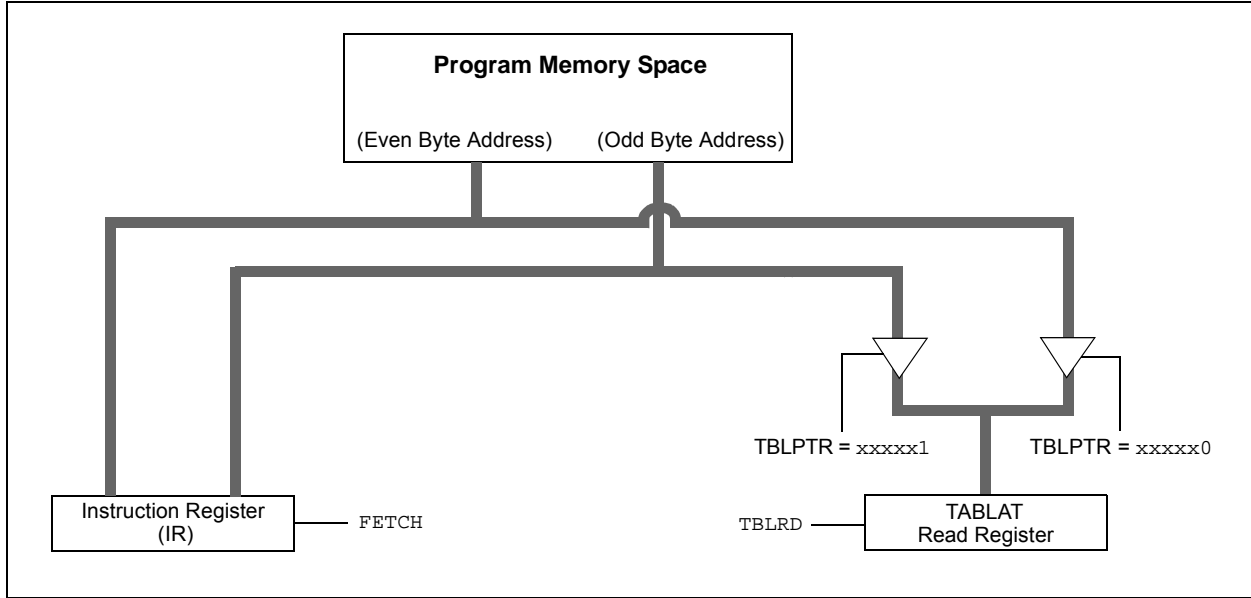
`TBLPTR` points to a byte address in program space. Executing `TBLRD` places the byte pointed to into `TABLAT`.

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 7-2](#) shows the interface between the internal program memory and the `TABLAT`.

A typical method for reading data from program memory is shown in [Example 7-1](#).

# PIC18F6310/6410/8310/8410

**FIGURE 7-2: READS FROM 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
```

# PIC18F6310/6410/8310/8410

---

## 7.4 Writing to Program Memory Space (PIC18F8310/8410 only)

The table write operation outputs the contents of the TBLPTR and TABLAT registers to the external address and data busses of the external memory interface. Depending on the program memory mode selected, the operation may target any byte address in the device's memory space. What happens to this data depends largely on the external memory device being used.

For PIC18 devices with Enhanced Flash memory, a single algorithm is used for writing to the on-chip program array. In the case of external devices, however, the algorithm is determined by the type of memory device and its requirements. In some cases, a specific instruction sequence must be sent before data can be written or erased. Address and data demultiplexing, chip select operation and write time requirements must all be considered in creating the appropriate code.

The connection of the data and address busses to the memory device are dictated by the interface being used, the data bus width and the target device. When using a 16-bit data path, the algorithm must take into account the width of the target memory.

Another important consideration is the write time requirement of the target device. If this is longer than the time that a TBLWT operation makes data available on the interface, the algorithm must be adjusted to lengthen this time. It may be possible, for example, to buy enough time by increasing the length of the wait state on table operations.

In all cases, it is important to remember that instructions in the program memory space are word-aligned, with the Least Significant bit always being written to an even-numbered address (LSb = 0). If data is being stored in the program memory space, word alignment of the data is not required.

A complete overview of interface algorithms is beyond the scope of this data sheet. The best place for timing and instruction sequence requirements is the data sheet of the memory device in question. For additional information on algorithm design for the external memory interface, refer to Microchip application note AN869, "External Memory Interfacing Techniques for the PIC18F8XXX" (DS00869).

### 7.4.1 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.

### 7.4.2 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 application writes to external memory on a frequent basis, it may be necessary to implement an error trapping routine to handle these unplanned events.

## 7.5 Erasing External Memory (PIC18F8310/8410 only)

Erasure is implemented in different ways on different devices. In many cases, it is possible to erase all or part of the memory by issuing a specific command. In some devices, it may be necessary to write '0's to the locations to be erased. For specific information, consult the external memory device's data sheet for clarification.

## 7.6 Writing and Erasing On-Chip Program Memory (ICSP Mode)

While the on-chip program memory is read-only in normal operating mode, it can be written to and erased as a function of In-Circuit Serial Programming (ICSP). In this mode, the TBLWT operation is used in all devices to write to blocks of 64 bytes (32 words) at one time. Write blocks are boundary-aligned with the code protection blocks. Special commands are used to erase one or more code blocks of the program memory, or the entire device.

The TBLWT operation on write blocks is somewhat different than the word write operations for PIC18F8310/8410 devices described here. A more complete description of block write operations is provided in the Microchip document "Programming Specifications for PIC18FX410/X490 Flash MCUs" (DS39624).

## 7.7 Flash Program Operation During Code Protection

See **Section 24.5 "Program Verification and Code Protection"** for details on code protection of Flash program memory.

# PIC18F6310/6410/8310/8410

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

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					63
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								63
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								63
TABLAT	Program Memory Table Latch								63

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

# PIC18F6310/6410/8310/8410

---

NOTES:

# PIC18F6310/6410/8310/8410

## 8.0 EXTERNAL MEMORY INTERFACE

**Note:** The external memory interface is not implemented on PIC18F6310 and PIC18F6410 (64-pin) devices.

The external memory interface allows the device to access external memory devices (such as Flash, EPROM, SRAM, etc.) as program or data memory. It is implemented with 28 pins, multiplexed across four I/O ports. Three ports (PORTD, PORTE and PORTH) are multiplexed with the address/data bus for a total of 20 available lines, while PORTJ is multiplexed with the bus control signals. A list of the pins and their functions is provided in [Table 8-1](#).

As implemented here, the interface is similar to that introduced on PIC18F8X20 microcontrollers. The most notable difference is that the interface on PIC18F8310/8410 devices supports both 16-Bit and Multiplexed 8-Bit Data Width modes; it does not support the 8-Bit Demultiplexed mode. The Bus Width mode is set by the BW Configuration bit when the device is programmed and cannot be changed in software.

The operation of the interface is controlled by the MEMCON register ([Register 8-1](#)). Clearing the EBDIS bit (MEMCON<7>) enables the interface and disables the I/O functions of the ports, as well as any other multiplexed functions. Setting the bit disables the interface and enables the ports.

For a more complete discussion of the operating modes that use the external memory interface, refer to [Section 8.1 “Program Memory Modes and the External Memory Interface”](#).

**REGISTER 8-1: MEMCON: MEMORY CONTROL REGISTER**

R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EBDIS	—	WAIT1	$\overline{\text{WAIT0}}$	—	—	WM1	WM0
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      **EBDIS:** External Bus Disable bit  
 1 = External system bus disabled, all external bus drivers are mapped as I/O ports  
 0 = External system bus enabled, I/O ports are disabled
- bit 6      **Unimplemented:** Read as '0'
- bit 5-4    **WAIT<1:0>:** Table Reads and Writes Bus Cycle Wait Count bits  
 11 = Table reads and writes will wait 0 Tcy  
 10 = Table reads and writes will wait 1 Tcy  
 01 = Table reads and writes will wait 2 Tcy  
 00 = Table reads and writes will wait 3 Tcy
- bit 3-2    **Unimplemented:** Read as '0'
- bit 1-0    **WM<1:0>:** TBLWRT Operation with 16-Bit Bus Width bits  
 1x = Word Write mode: TABLAT0 and TABLAT1 word output;  $\overline{\text{WRH}}$  active when TABLAT1 is written  
 01 = Byte Select mode: TABLAT data copied on both MSB and LSB;  $\overline{\text{WRH}}$  and ( $\overline{\text{UB}}$  or  $\overline{\text{LB}}$ ) will activate  
 00 = Byte Write mode: TABLAT data copied on both MSB and LSB;  $\overline{\text{WRH}}$  or  $\overline{\text{WRL}}$  will activate

**Note 1:** If SBOREN is enabled, its Reset state is '1'; otherwise, it is '0'.

# PIC18F6310/6410/8310/8410

**TABLE 8-1: PIC18F8310/8410 EXTERNAL BUS – I/O PORT FUNCTIONS**

Name	Port	Bit	Function
RD0/AD0/PSP0	PORTD	0	Input/Output or System Bus Address bit 0 or Data bit 0 or Parallel Slave Port bit 0
RD1/AD1/PSP1	PORTD	1	Input/Output or System Bus Address bit 1 or Data bit 1 or Parallel Slave Port bit 1
RD2/AD2/PSP2	PORTD	2	Input/Output or System Bus Address bit 2 or Data bit 2 or Parallel Slave Port bit 2
RD3/AD3/PSP3	PORTD	3	Input/Output or System Bus Address bit 3 or Data bit 3 or Parallel Slave Port bit 3
RD4/AD4/PSP4	PORTD	4	Input/Output or System Bus Address bit 4 or Data bit 4 or Parallel Slave Port bit 4
RD5/AD5/PSP5	PORTD	5	Input/Output or System Bus Address bit 5 or Data bit 5 or Parallel Slave Port bit 5
RD6/AD6/PSP6	PORTD	6	Input/Output or System Bus Address bit 6 or Data bit 6 or Parallel Slave Port bit 6
RD7/AD7/PSP7	PORTD	7	Input/Output or System Bus Address bit 7 or Data bit 7 or Parallel Slave Port bit 7
RE0/AD8/RD	PORTE	0	Input/Output or System Bus Address bit 8 or Data bit 8 or Parallel Slave Port Read Control pin
RE1/AD9/WR	PORTE	1	Input/Output or System Bus Address bit 9 or Data bit 9 or Parallel Slave Port Write Control pin
RE2/AD10/CS	PORTE	2	Input/Output or System Bus Address bit 10 or Data bit 10 or Parallel Slave Port Chip Select pin
RE3/AD11	PORTE	3	Input/Output or System Bus Address bit 11 or Data bit 11
RE4/AD12	PORTE	4	Input/Output or System Bus Address bit 12 or Data bit 12
RE5/AD13	PORTE	5	Input/Output or System Bus Address bit 13 or Data bit 13
RE6/AD14	PORTE	6	Input/Output or System Bus Address bit 14 or Data bit 14
RE7/CCP2 <sup>(1)</sup> /AD15	PORTE	7	Input/Output or Capture 2 Input/Compare 2 Output/PWM 2 Output pin or System Bus Address bit 15 or Data bit 15
RH0/AD16	PORTH	0	Input/Output or System Bus Address bit 16
RH1/AD17	PORTH	1	Input/Output or System Bus Address bit 17
RH2/AD18	PORTH	2	Input/Output or System Bus Address bit 18
RH3/AD19	PORTH	3	Input/Output or System Bus Address bit 19
RJ0/ALE	PORTJ	0	Input/Output or System Bus Address Latch Enable (ALE) Control pin
RJ1/OE	PORTJ	1	Input/Output or System Bus Output Enable (OE) Control pin
RJ2/WRL	PORTJ	2	Input/Output or System Bus Write Low (WRL) Control pin
RJ3/WRH	PORTJ	3	Input/Output or System Bus Write High (WRH) Control pin
RJ4/BA0	PORTJ	4	Input/Output or System Bus Byte Address bit 0
RJ5/CE	PORTJ	5	Input/Output or System Bus Chip Enable (CE) Control pin
RJ6/LB	PORTJ	6	Input/Output or System Bus Lower Byte Enable (LB) Control pin
RJ7/UB	PORTJ	7	Input/Output or System Bus Upper Byte Enable (UB) Control pin

**Note 1:** Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared (all devices in Microcontroller mode).

## 8.1 Program Memory Modes and the External Memory Interface

As previously noted, PIC18F8310/8410 devices are capable of operating in any one of four program memory modes, using combinations of on-chip and external program memory. The functions of the multiplexed port pins depends on the program memory mode selected, as well as the setting of the EBDIS bit.

In **Microcontroller mode**, the bus is not active and the pins have their port functions only. Writes to the MEMCOM register are not permitted.

In **Microprocessor mode**, the external bus is always active and the port pins have only the external bus function.

In **Microprocessor with Boot Block** or **Extended Microcontroller mode**, the external program memory bus shares I/O port functions on the pins. When the device is fetching or doing table read/table write operations on the external program memory space, the pins will have the external bus function. If the device is fetching and accessing internal program memory locations only, the EBDIS control bit will change the pins from external memory to I/O port functions. When EBDIS = 0, the pins function as the external bus. When EBDIS = 1, the pins function as I/O ports.

If the device fetches or accesses external memory while EBDIS = 1, the pins will switch to external bus. If the EBDIS bit is set by a program executing from external memory, the action of setting the bit will be delayed until the program branches into the internal memory. At that time, the pins will change from external bus to I/O ports.



# PIC18F6310/6410/8310/8410

When the device is executing out of internal memory (EBDIS = 0) in Microprocessor with Boot Block mode or Extended Microcontroller mode, the control signals will NOT be active. They will go to a state where the  $\overline{AD}<15:0>$  and  $\overline{A}<19:16>$  are tri-state; the  $\overline{CE}$ ,  $\overline{OE}$ ,  $\overline{WRH}$ ,  $\overline{WRL}$ ,  $\overline{UB}$  and  $\overline{LB}$  signals are '1'; ALE and BA0 are '0'. Note that only those pins associated with the current address width are forced to tri-state; the other pins continue to function as I/O. In the case of a 16-bit address width, for example, only  $\overline{AD}<15:0>$  (PORTD and PORTE) are affected;  $\overline{A}<19:16>$  (PORTH<3:0>) continue to function as I/O. In all external memory modes, the bus takes priority over any other peripherals that may share pins with it. This includes the Parallel Slave Port and serial communications modules which would otherwise take priority over the I/O port.

## 8.2 16-Bit Mode

In 16-bit mode, the external memory interface can be connected to external memories in three different configurations:

- 16-Bit Byte Write
- 16-Bit Word Write
- 16-Bit Byte Select

The configuration to be used is determined by the  $\overline{WM}<1:0>$  bits in the MEMCON register ( $\overline{MEMCON}<1:0>$ ). These three different configurations allow the designer maximum flexibility in using both 8-bit and 16-bit devices with 16-bit data.

For all 16-bit modes, the Address Latch Enable (ALE) pin indicates that the address bits,  $\overline{A}<15:0>$ , are available on the external memory interface bus. Following the address latch, the Output Enable signal ( $\overline{OE}$ ) will enable both bytes of program memory at once to form a 16-bit instruction word. The Chip Enable signal ( $\overline{CE}$ ) is active at any time that the microcontroller accesses external memory, whether reading or writing; it is inactive (asserted high) whenever the device is in Sleep mode.

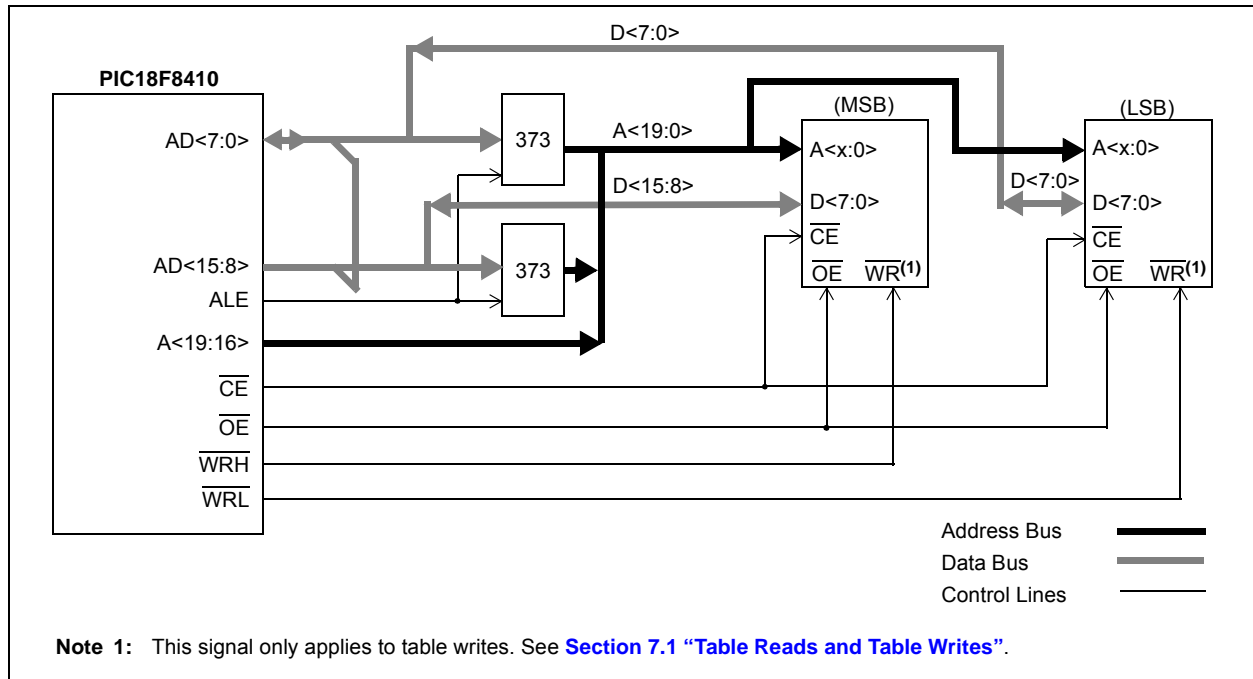
In Byte Select mode, JEDEC standard Flash memories will require BA0 for the byte address line and one I/O line to select between Byte and Word mode. The other 16-bit modes do not need BA0. JEDEC standard static RAM memories will use the  $\overline{UB}$  or  $\overline{LB}$  signals for byte selection.

### 8.2.1 16-BIT BYTE WRITE MODE

Figure 8-1 shows an example of 16-Bit Byte Write mode for PIC18F8310/8410 devices. This mode is used for two separate 8-bit memories connected for 16-bit operation. This generally includes basic EPROM and Flash devices. It allows table writes to byte-wide external memories.

During a  $\overline{TBLWT}$  instruction cycle, the  $\overline{TABLAT}$  data is presented on the upper and lower bytes of the  $\overline{AD}<15:0>$  bus. The appropriate  $\overline{WRH}$  or  $\overline{WRL}$  control line is strobed on the LSb of the  $\overline{TBLPTR}$ .

FIGURE 8-1: 16-BIT BYTE WRITE MODE EXAMPLE



# PIC18F6310/6410/8310/8410

## 8.2.2 16-BIT WORD WRITE MODE

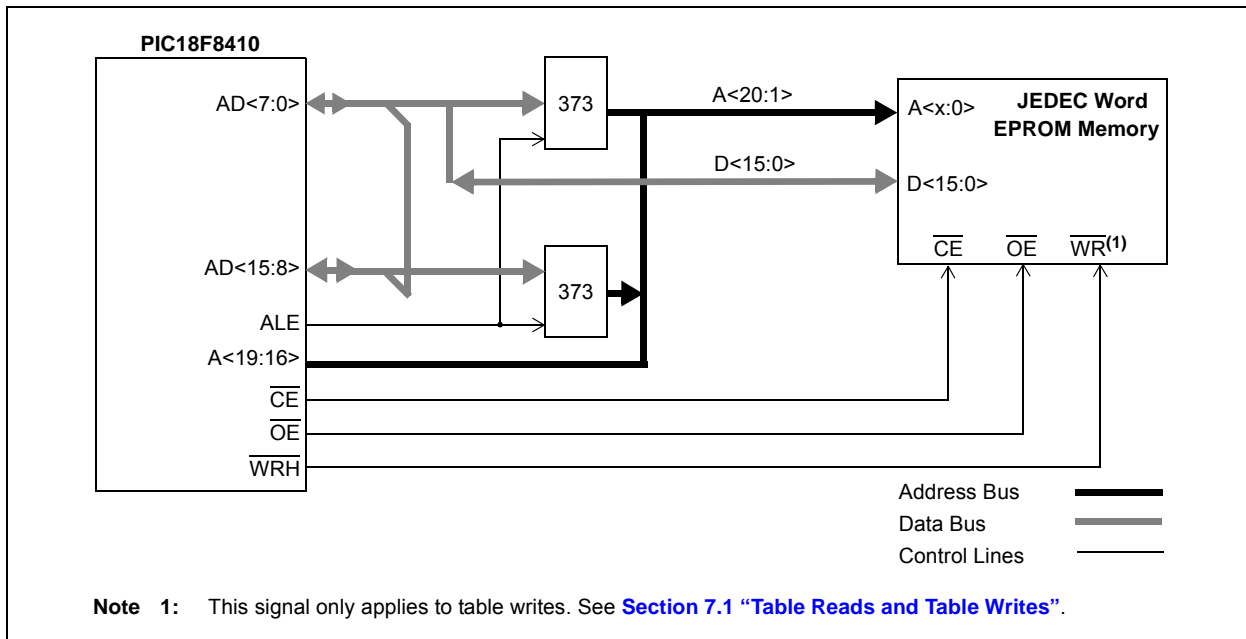
Figure 8-2 shows an example of 16-Bit Word Write mode for PIC18F8410 devices. This mode is used for word-wide memories, which includes some of the EPROM and Flash type memories. This mode allows opcode fetches and table reads from all forms of 16-bit memory and table writes to any type of word-wide external memories. This method makes a distinction between TBLWT cycles to even or odd addresses.

During a TBLWT cycle to an even address (TBLPTR<0> = 0), the TABLAT data is transferred to a holding latch and the external address data bus is tri-stated for the data portion of the bus cycle. No write signals are activated.

During a TBLWT cycle to an odd address (TBLPTR<0> = 1), the TABLAT data is presented on the upper byte of the AD<15:0> bus. The contents of the holding latch are presented on the lower byte of the AD<15:0> bus.

The WRH signal is strobed for each write cycle; the WRL pin is unused. The signal on the BA0 pin indicates the LSB of TBLPTR, but it is left unconnected. Instead, the UB and LB signals are active to select both bytes. The obvious limitation to this method is that the table write must be done in pairs on a specific word boundary to correctly write a word location.

**FIGURE 8-2: 16-BIT WORD WRITE MODE EXAMPLE**



# PIC18F6310/6410/8310/8410

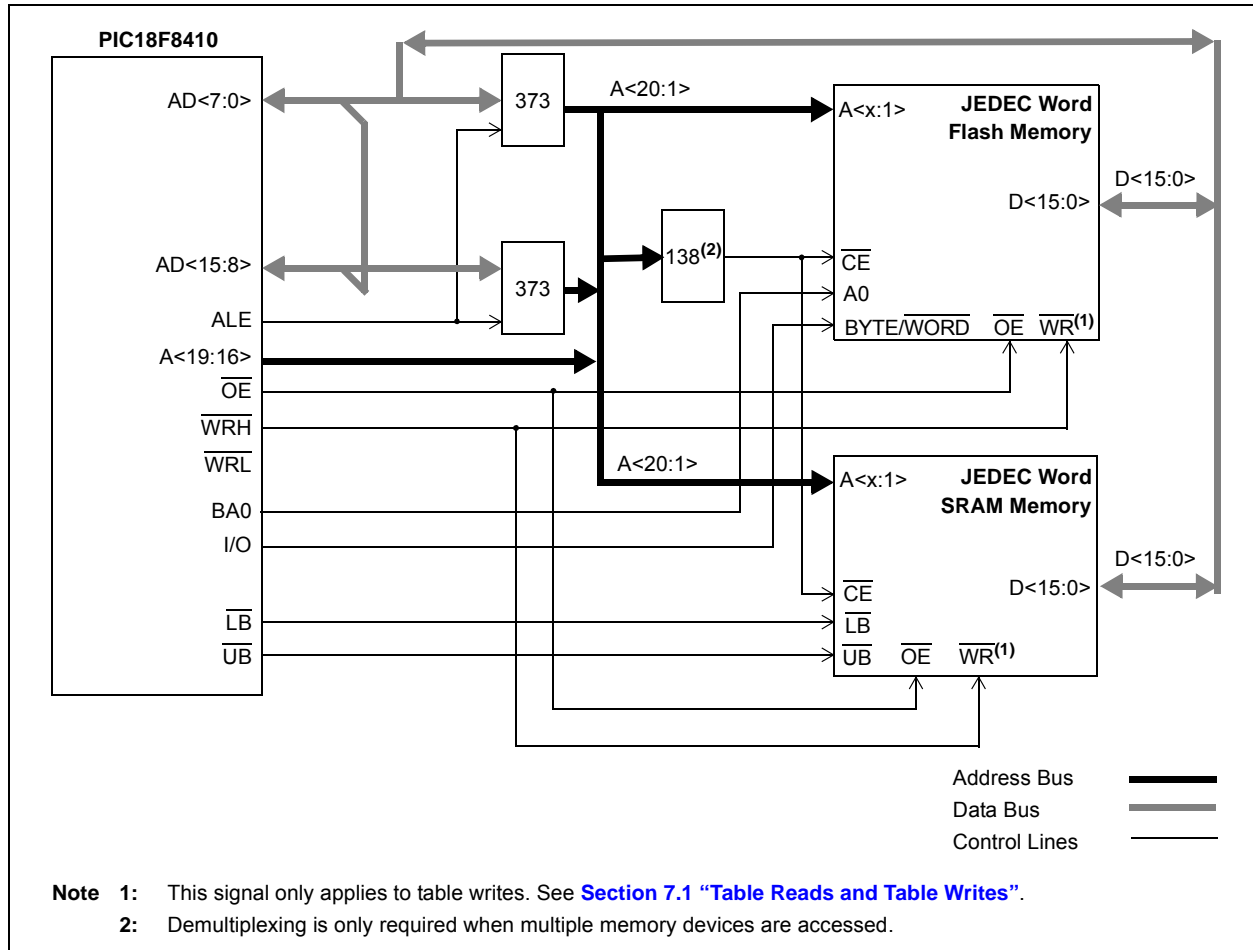
## 8.2.3 16-BIT BYTE SELECT MODE

Figure 8-3 shows an example of 16-Bit Byte Select mode. This mode allows table write operations to word-wide external memories with byte selection capability. This generally includes both word-wide Flash and SRAM devices.

During a TBLWT cycle, the TABLAT data is presented on the upper and lower byte of the AD<15:0> bus. The WRH signal is strobed for each write cycle; the WRL pin is not used. The BA0 or  $\overline{UB}/\overline{LB}$  signals are used to select the byte to be written, based on the Least Significant bit of the TBLPTR register.

Flash and SRAM devices use different control signal combinations to implement Byte Select mode. JEDEC standard Flash memories require that a controller I/O port pin be connected to the memory's  $\overline{BYTE}/\overline{WORD}$  pin to provide the select signal. They also use the BA0 signal from the controller as a byte address. JEDEC standard static RAM memories, on the other hand, use the  $\overline{UB}$  or  $\overline{LB}$  signals to select the byte.

**FIGURE 8-3: 16-BIT BYTE SELECT MODE EXAMPLE**

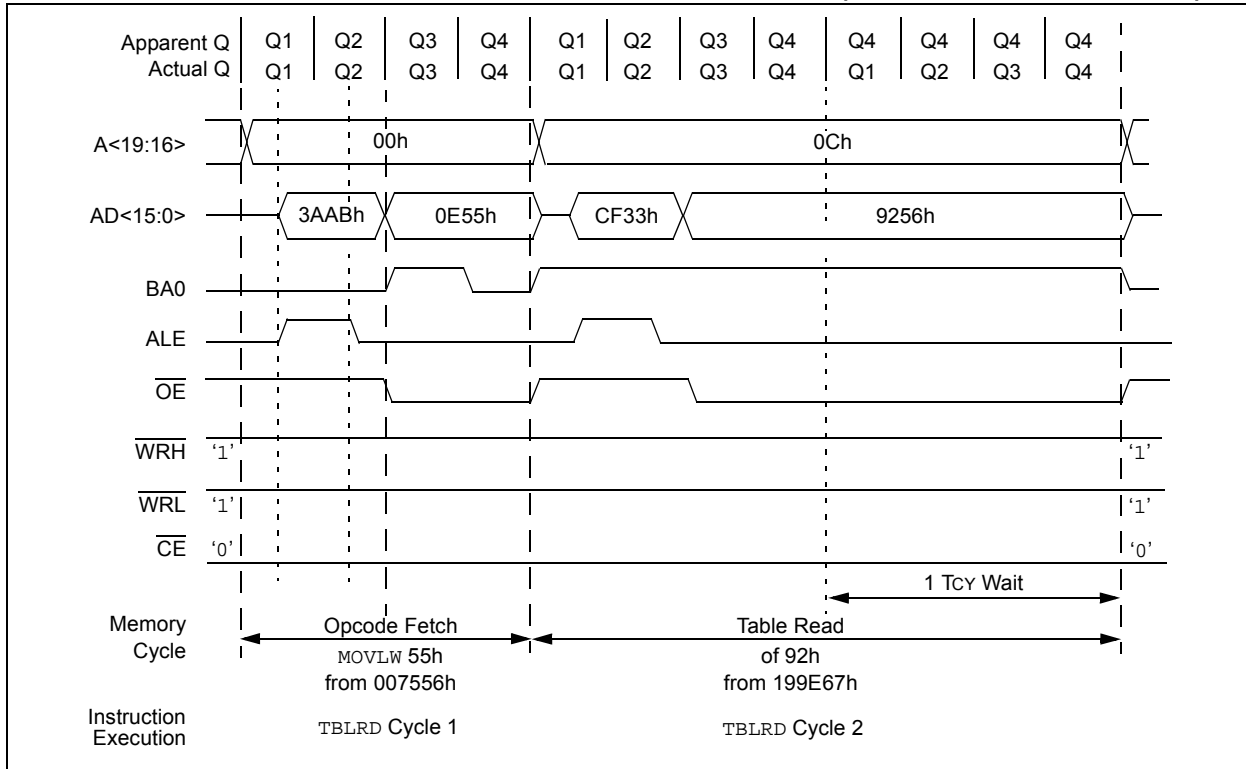


# PIC18F6310/6410/8310/8410

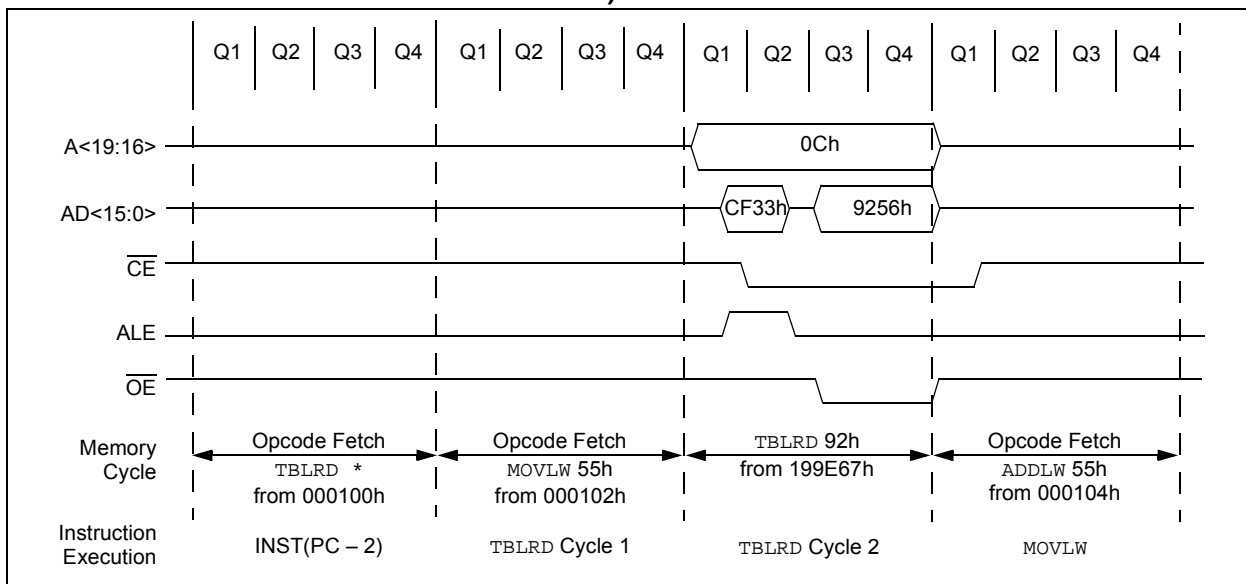
## 8.2.4 16-BIT MODE TIMING

The presentation of control signals on the external memory bus is different for the various operating modes. Typical signal timing diagrams are shown in Figure 8-4 through Figure 8-6.

**FIGURE 8-4: EXTERNAL MEMORY BUS TIMING FOR TBLRD (MICROPROCESSOR MODE)**

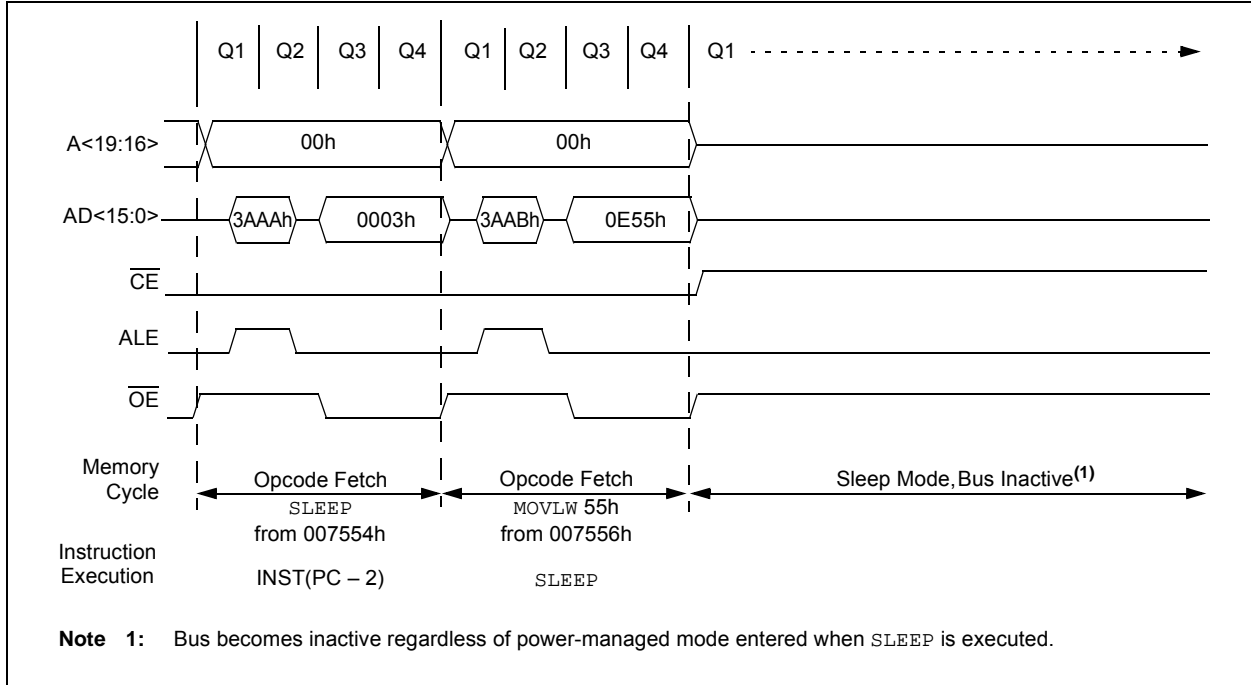


**FIGURE 8-5: EXTERNAL MEMORY BUS TIMING FOR TBLRD (EXTENDED MICROCONTROLLER MODE)**



# PIC18F6310/6410/8310/8410

**FIGURE 8-6: EXTERNAL MEMORY BUS TIMING FOR SLEEP (MICROPROCESSOR MODE)**



# PIC18F6310/6410/8310/8410

## 8.3 8-Bit Mode

The external memory interface implemented in PIC18F8410 devices operates only in 8-Bit Multiplexed mode; data shares the 8 Least Significant bits of the address bus.

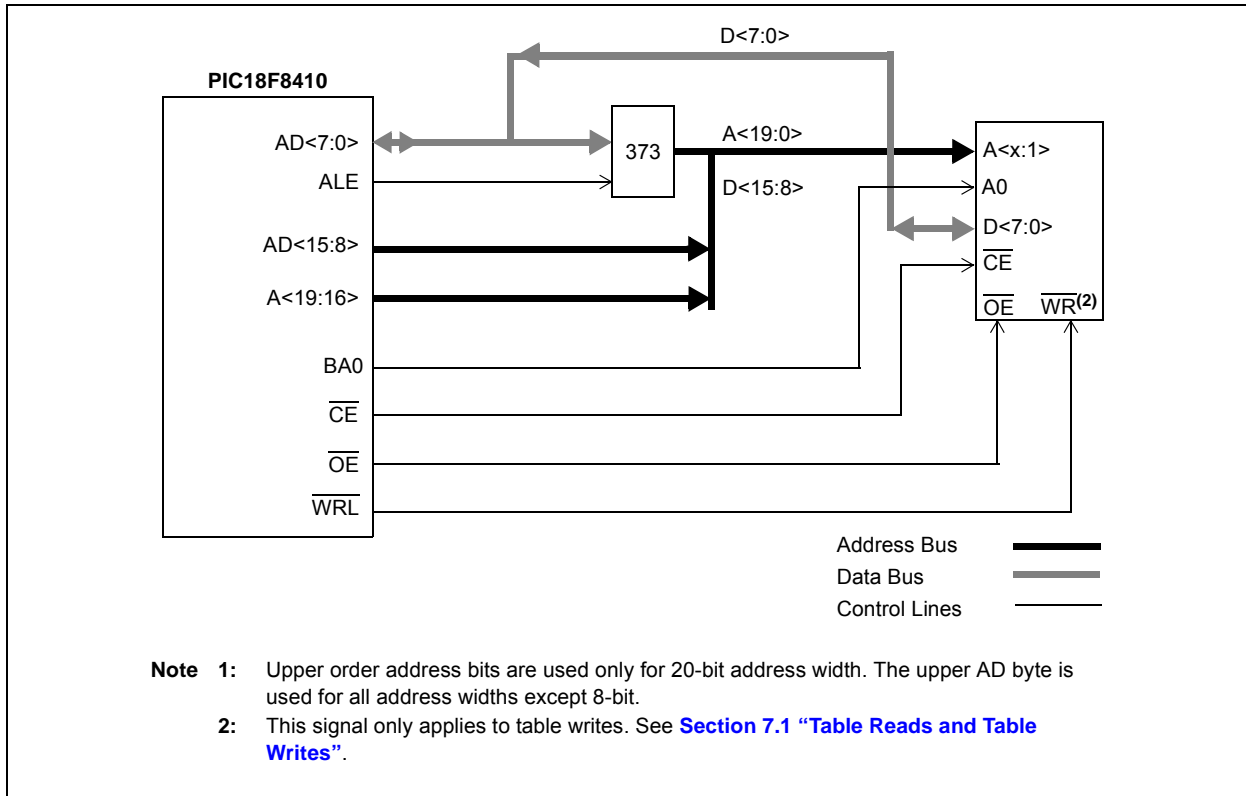
Figure 8-1 shows an example of 8-Bit Multiplexed mode for PIC18F8410 devices. This mode is used for a single 8-bit memory connected for 16-bit operation. The instructions will be fetched as two 8-bit bytes on a shared data/address bus. The two bytes are sequentially fetched within one instruction cycle ( $T_{CY}$ ). Therefore, the designer must choose external memory devices according to timing calculations based on  $1/2 T_{CY}$  (2 times the instruction rate). For proper memory speed selection, glue logic propagation delay times must be considered along with setup and hold times.

The Address Latch Enable (ALE) pin indicates that the address bits,  $A<15:0>$ , are available on the external memory interface bus. The Output Enable signal ( $\overline{OE}$ ) will enable one byte of program memory for a portion of the instruction cycle, then  $BA0$  will change and the second byte will be enabled to form the 16-bit instruction word. The Least Significant bit of the address,  $BA0$ , must be connected to the memory devices in this mode. The Chip Enable signal ( $\overline{CE}$ ) is active at any time that the microcontroller accesses external memory, whether reading or writing; it is inactive (asserted high) whenever the device is in Sleep mode.

This generally includes basic EPROM and Flash devices. It allows table writes to byte-wide external memories.

During a  $TBLWT$  instruction cycle, the  $TBLAT$  data is presented on the upper and lower bytes of the  $AD<15:0>$  bus. The appropriate level of the  $BA0$  control line is strobed on the LSb of the  $TBLPTR$ .

FIGURE 8-7: 8-BIT MULTIPLEXED MODE EXAMPLE

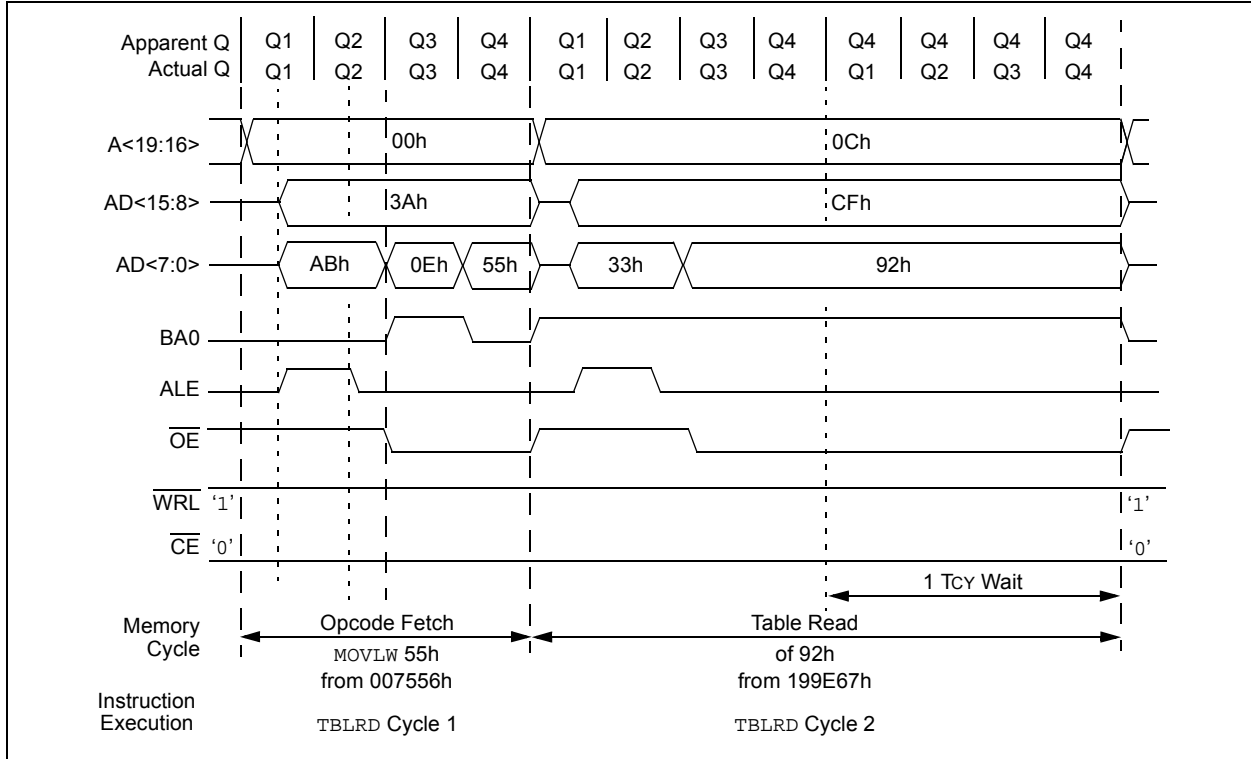


# PIC18F6310/6410/8310/8410

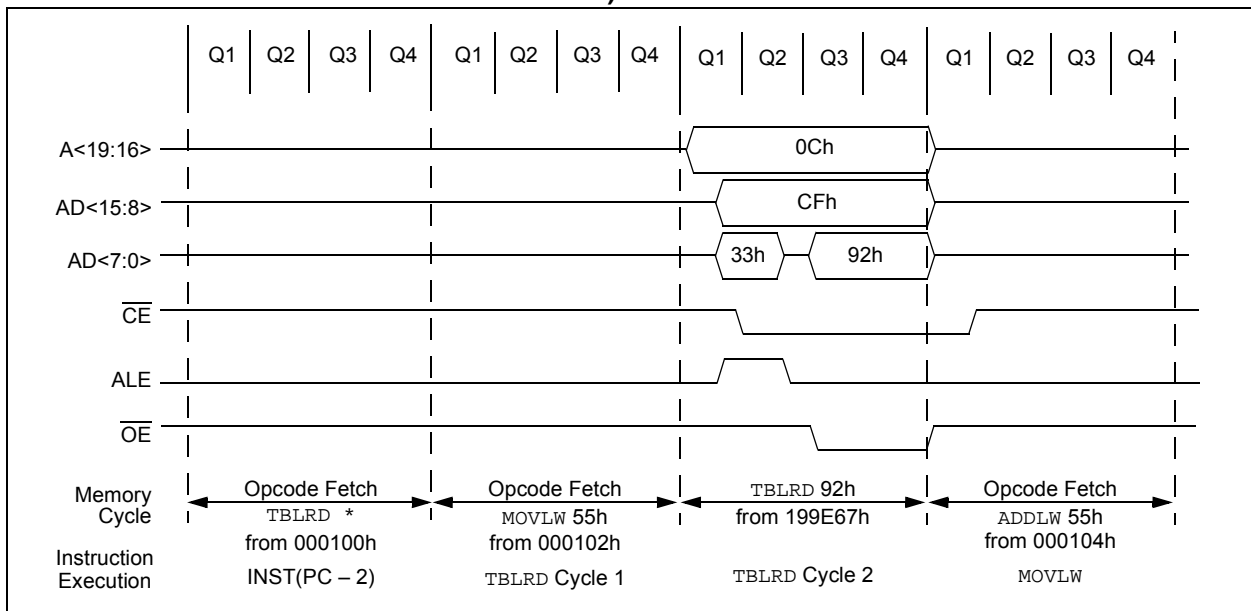
## 8.3.1 8-BIT MODE TIMING

The presentation of control signals on the external memory bus is different for the various operating modes. Typical signal timing diagrams are shown in Figure 8-4 through Figure 8-6.

**FIGURE 8-8: EXTERNAL MEMORY BUS TIMING FOR TBLRD (MICROPROCESSOR MODE)**

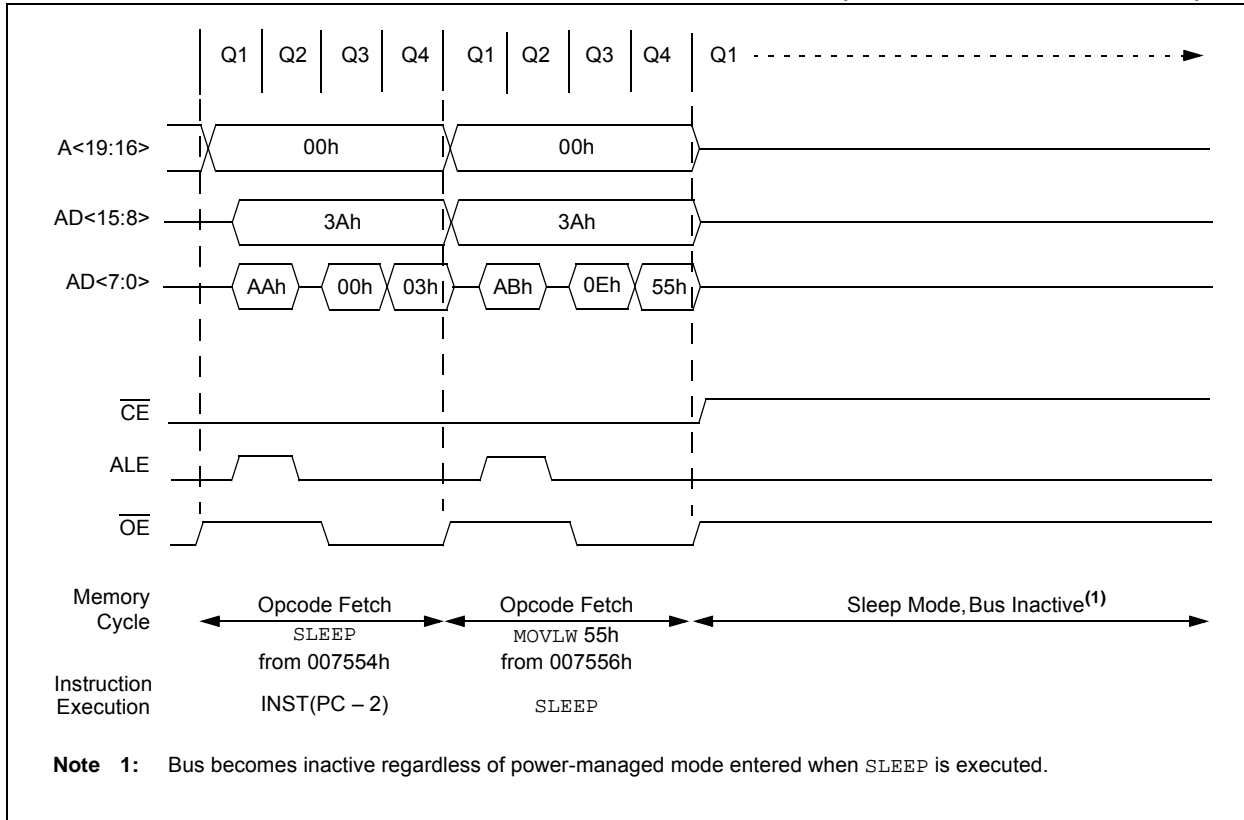


**FIGURE 8-9: EXTERNAL MEMORY BUS TIMING FOR TBLRD (EXTENDED MICROCONTROLLER MODE)**



# PIC18F6310/6410/8310/8410

**FIGURE 8-10: EXTERNAL MEMORY BUS TIMING FOR SLEEP (MICROPROCESSOR MODE)**





# PIC18F6310/6410/8310/8410

## 8.4 Operation in Power-Managed Modes

In alternate, power-managed Run modes, the external bus continues to operate normally. If a clock source with a lower speed is selected, bus operations will run at that speed. In these cases, excessive access times for the external memory may result if wait states have been enabled and added to external memory operations.

If operations in a lower power Run mode are anticipated, users should provide in their applications for adjusting memory access times at the lower clock speeds.

In Sleep and Idle modes, the microcontroller core does not need to access data; bus operations are suspended. The state of the external bus is frozen with the address/data pins and most of the control pins holding at the same state they were in when the mode was invoked. The only potential changes are the CE, LB and UB pins which are held at logic high.

**TABLE 8-2: REGISTERS ASSOCIATED WITH THE EXTERNAL MEMORY INTERFACE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
MEMCON	EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0	65
CONFIG3L	WAIT	BW	—	—	—	—	PM1	PM0	285
CONFIG3H	MCLRE	—	—	—	—	LPT1OSC	—	CCP2MX	286

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for the external memory interface.

# PIC18F6310/6410/8310/8410

---

NOTES:

# PIC18F6310/6410/8310/8410

## 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 sign 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

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

#### EXAMPLE 9-2: 8 x 8 SIGNED MULTIPLY ROUTINE

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

BTFSC ARG2, SB ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH
; - ARG1

MOVWF 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

# PIC18F6310/6410/8310/8410

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

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

## 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     ;
;

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 sign 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

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

## 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

```

## 10.0 INTERRUPTS

The PIC18F6310/6410/8310/8410 devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high-priority level or a low-priority level. The high-priority interrupt vector is at 0008h and the low-priority interrupt vector is at 0018h. 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.

In general, interrupt sources have three bits to control their operation. They 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 0008h or 0018h, 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 0008h 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 either be 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 (0008h or 0018h). 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.

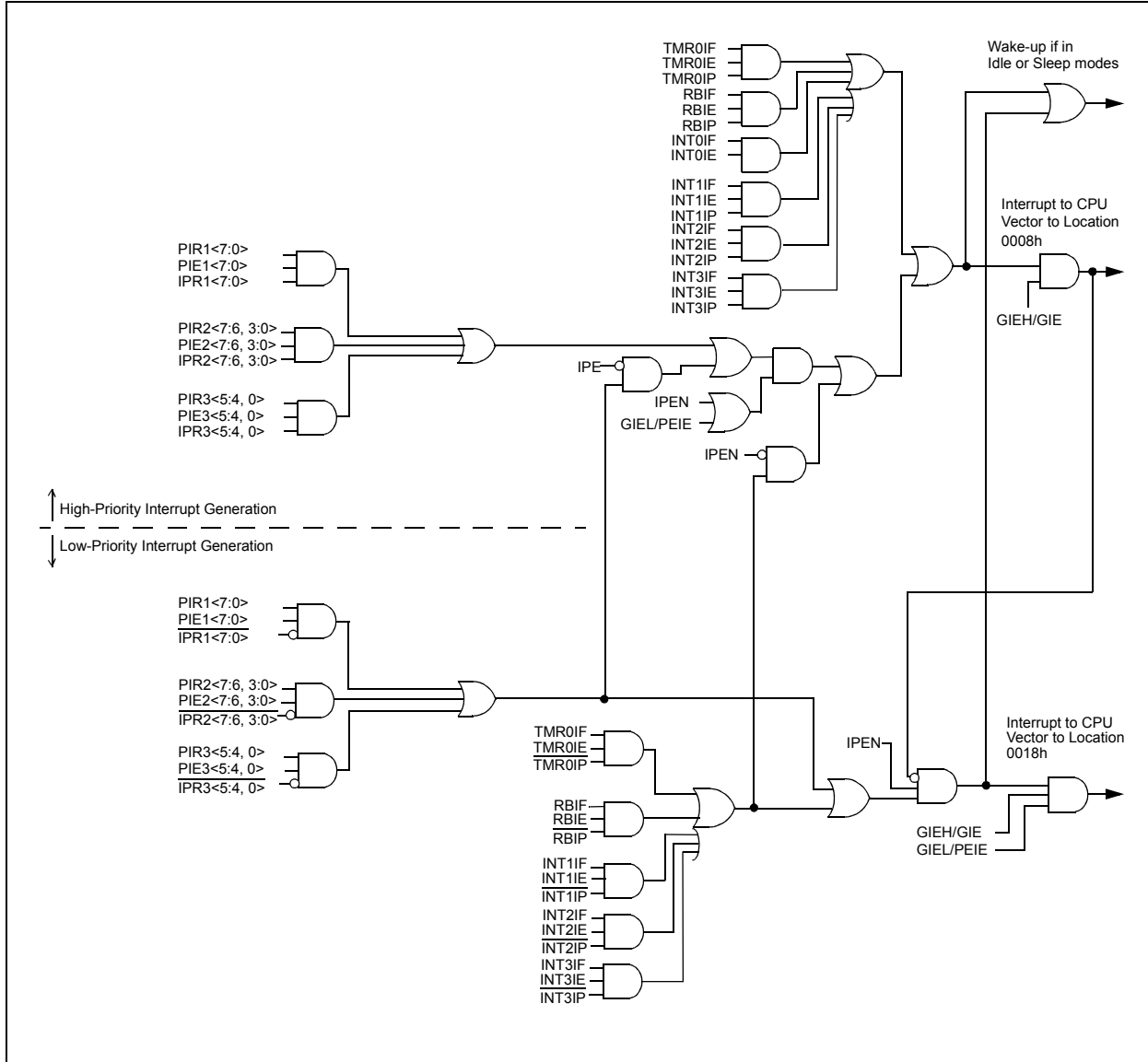
The “return from interrupt” instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

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.

# PIC18F6310/6410/8310/8410

FIGURE 10-1: PIC18F6310/6410/8310/8410 INTERRUPT LOGIC



# PIC18F6310/6410/8310/8410

## 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 that 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 <sup>(1)</sup>
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 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<sup>(1)</sup>  
 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.

# PIC18F6310/6410/8310/8410

## REGISTER 10-2: INTCON2: INTERRUPT CONTROL REGISTER 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
$\overline{\text{RBP}}\text{U}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	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	<b><math>\overline{\text{RBP}}\text{U}</math></b> : 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	<b>INTEDG0</b> : External Interrupt 0 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 5	<b>INTEDG1</b> : External Interrupt 1 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 4	<b>INTEDG2</b> : External Interrupt 2 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 3	<b>INTEDG3</b> : External Interrupt 3 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 2	<b>TMR0IP</b> : TMR0 Overflow Interrupt Priority bit 1 = High priority 0 = Low priority
bit 1	<b>INT3IP</b> : INT3 External Interrupt Priority bit 1 = High priority 0 = Low priority
bit 0	<b>RBIP</b> : 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.



# PIC18F6310/6410/8310/8410

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

R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	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            **INT3IE:** INT3 External Interrupt Enable bit  
                   1 = Enables the INT3 external interrupt  
                   0 = Disables the INT3 external interrupt
- 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            **INT3IF:** INT3 External Interrupt Flag bit  
                   1 = The INT3 external interrupt occurred (must be cleared in software)  
                   0 = The INT3 external interrupt did not occur
- 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.

# PIC18F6310/6410/8310/8410

## 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 three Peripheral Interrupt Request (Flag) registers (PIR1, PIR2, PIR3).

**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	ADIF	RC1IF	TX1IF	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 = 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 **RC1IF:** EUSART Receive Interrupt Flag bit  
 1 = The EUSART receive buffer, RCREG1, is full (cleared when RCREG1 is read)  
 0 = The EUSART receive buffer is empty
- bit 4 **TX1IF:** EUSART Transmit Interrupt Flag bit  
 1 = The EUSART transmit buffer, TXREG1, is empty (cleared when TXREG1 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/TMR3 register capture occurred (must be cleared in software)  
 0 = No TMR1/TMR3 register capture occurred  
Compare mode:  
 1 = A TMR1/TMR3 register compare match occurred (must be cleared in software)  
 0 = No TMR1/TMR3 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

# PIC18F6310/6410/8310/8410

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

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIF	CMIF	—	—	BCLIF	HLVDIF	TMR3IF	CCP2IF
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 = Device oscillator failed, clock input has changed to INTOSC (must be cleared in software)  
 0 = Device clock operating
- bit 6      **CMIF:** Comparator Interrupt Flag bit  
 1 = Comparator input has changed (must be cleared in software)  
 0 = Comparator input has not changed
- bit 5-4    **Unimplemented:** Read as '0'
- 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 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      **CCP2IF:** CCP2 Interrupt Flag bit  
Capture mode:  
 1 = A TMR1/TMR3 register capture occurred (must be cleared in software)  
 0 = No TMR1/TMR3 register capture occurred  
Compare mode:  
 1 = A TMR1/TMR3 register compare match occurred (must be cleared in software)  
 0 = No TMR1/TMR3 register compare match occurred  
PWM mode:  
 Unused in this mode.

# PIC18F6310/6410/8310/8410

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

U-0	U-0	R-0	R-0	U-0	U-0	U-0	U-0
—	—	RC2IF	TX2IF	—	—	—	CCP3IF
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 **RC2IF:** AUSART Receive Interrupt Flag bit

1 = The AUSART receive buffer, RCREG2, is full (cleared when RCREG2 is read)

0 = The AUSART receive buffer is empty

bit 4 **TX2IF:** AUSART Transmit Interrupt Flag bit

1 = The AUSART transmit buffer, TXREG2, is empty (cleared when TXREG2 is written)

0 = The AUSART transmit buffer is full

bit 3-1 **Unimplemented:** Read as '0'

bit 0 **CCP3IF:** CCP3 Interrupt Flag bit

#### Capture mode:

1 = A TMR1/TMR3 register capture occurred (must be cleared in software)

0 = No TMR1/TMR3 register capture occurred

#### Compare mode:

1 = A TMR1/TMR3 register compare match occurred (must be cleared in software)

0 = No TMR1/TMR3 register compare match occurred

#### PWM mode:

Unused in this mode.

# PIC18F6310/6410/8310/8410

## 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 three Peripheral Interrupt Enable registers (PIE1, PIE2, PIE3). 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	ADIE	RC1IE	TX1IE	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	<b>PSPIE:</b> Parallel Slave Port Read/Write Interrupt Enable bit 1 = Enables the PSP read/write interrupt 0 = Disables the PSP read/write interrupt
bit 6	<b>ADIE:</b> A/D Converter Interrupt Enable bit 1 = Enables the A/D interrupt 0 = Disables the A/D interrupt
bit 5	<b>RC1IE:</b> EUSART Receive Interrupt Enable bit 1 = Enables the EUSART receive interrupt 0 = Disables the EUSART receive interrupt
bit 4	<b>TX1IE:</b> EUSART Transmit Interrupt Enable bit 1 = Enables the EUSART transmit interrupt 0 = Disables the EUSART transmit interrupt
bit 3	<b>SSPIE:</b> Master Synchronous Serial Port Interrupt Enable bit 1 = Enables the MSSP interrupt 0 = Disables the MSSP interrupt
bit 2	<b>CCP1IE:</b> CCP1 Interrupt Enable bit 1 = Enables the CCP1 interrupt 0 = Disables the CCP1 interrupt
bit 1	<b>TMR2IE:</b> 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	<b>TMR1IE:</b> TMR1 Overflow Interrupt Enable bit 1 = Enables the TMR1 overflow interrupt 0 = Disables the TMR1 overflow interrupt

# PIC18F6310/6410/8310/8410

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

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIE	CMIE	—	—	BCLIE	HLVDIE	TMR3IE	CCP2IE
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 = Enabled  
0 = Disabled
- bit 5-4    **Unimplemented:** Read as '0'
- 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      **CCP2IE:** CCP2 Interrupt Enable bit  
1 = Enabled  
0 = Disabled

# PIC18F6310/6410/8310/8410

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

U-0	U-0	R-0	R-0	U-0	U-0	U-0	R/W-0
—	—	RC2IE	TX2IE	—	—	—	CCP3IE
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      **RC2IE:** AUSART Receive Interrupt Enable bit  
            1 = Enabled  
            0 = Disabled
- bit 4      **TX2IE:** AUSART Transmit Interrupt Enable bit  
            1 = Enabled  
            0 = Disabled
- bit 3-1      **Unimplemented:** Read as '0'
- bit 0      **CCP3IE:** CCP3 Interrupt Enable bit  
            1 = Enabled  
            0 = Disabled

# PIC18F6310/6410/8310/8410

## 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 three Peripheral Interrupt Priority registers (IPR1, IPR2, IPR3). 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	ADIP	RC1IP	TX1IP	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 = High priority

0 = Low priority

bit 6      **ADIP:** A/D Converter Interrupt Priority bit

1 = High priority

0 = Low priority

bit 5      **RC1IP:** EUSART Receive Interrupt Priority bit

1 = High priority

0 = Low priority

bit 4      **TX1IP:** 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



# PIC18F6310/6410/8310/8410

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

R/W-1	R/W-1	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
OSCFIP	CMIP	—	—	BCLIP	HLVDIP	TMR3IP	CCP2IP
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	<b>OSCFIP:</b> Oscillator Fail Interrupt Priority bit 1 = High priority 0 = Low priority
bit 6	<b>CMIP:</b> Comparator Interrupt Priority bit 1 = High priority 0 = Low priority
bit 5-4	<b>Unimplemented:</b> Read as '0'
bit 3	<b>BCLIP:</b> Bus Collision Interrupt Priority bit 1 = High priority 0 = Low priority
bit 2	<b>HLVDIP:</b> High/Low-Voltage Detect Interrupt Priority bit 1 = High priority 0 = Low priority
bit 1	<b>TMR3IP:</b> TMR3 Overflow Interrupt Priority bit 1 = High priority 0 = Low priority
bit 0	<b>CCP2IP:</b> CCP2 Interrupt Priority bit 1 = High priority 0 = Low priority

# PIC18F6310/6410/8310/8410

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

U-0	U-0	R-1	R-1	U-0	U-0	U-0	R/W-1
—	—	RC2IP	TX2IP	—	—	—	CCP3IP
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      **RC2IP:** AUSART Receive Priority Flag bit

1 = High priority

0 = Low priority

bit 4      **TX2IP:** AUSART Transmit Interrupt Priority bit

1 = High priority

0 = Low priority

bit 3-1      **Unimplemented:** Read as '0'

bit 0      **CCP3IP:** CCP3 Interrupt Priority bit

1 = High priority

0 = Low priority

# PIC18F6310/6410/8310/8410

## 10.5 RCON Register

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

### REGISTER 10-13: RCON REGISTER

R/W-0	R/W-1	U-0	R/W-1	R-1	R-1	R/W-0	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:** Software BOR Enable bit  
For details of bit operation and Reset state, 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 Timer 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  
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](#).

# PIC18F6310/6410/8310/8410

## 10.6 INTx Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1, RB2/INT2 and RB3/INT3 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, INTxIF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxIE. Flag bit, INTxIF, must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1, INT2 and INT3) can wake-up the processor from the power-managed modes if bit, INTxIE, 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, INT2 and INT3 is determined by the value contained in the interrupt priority bits, INT1IP (INTCON3<6>), INT2IP (INTCON3<7>) and INT3IP (INTCON2<1>). 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 12.0 “Timer0 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_TEMP located anywhere
;
; USER ISR CODE
;
MOVFF    BSR_TEMP, BSR          ; Restore BSR
MOVF     W_TEMP, W              ; Restore WREG
MOVFF    STATUS_TEMP, STATUS    ; Restore STATUS
```

## 11.0 I/O PORTS

Depending on the device selected and features enabled, there are up to nine 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.

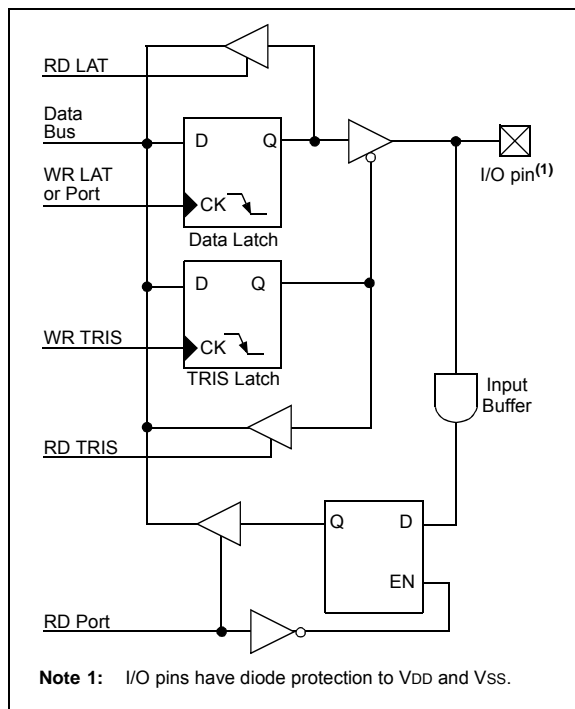
Each port has three registers for its operation. These registers are:

- TRIS register (Data Direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (Output Latch register)

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

A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 11-1.

**FIGURE 11-1: GENERIC I/O PORT OPERATION**



## 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 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 the Configuration register (see Section 24.1 “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 the analog VREF+ and VREF- inputs. The operation of pins, RA<5:0>, as A/D Converter inputs is selected by clearing or setting the PCFG<3:0> control bits in the ADCON1 register.

**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.

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

The TRISA register controls the direction of the PORTA 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  07h     ; 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
    
```

# PIC18F6310/6410/8310/8410

**TABLE 11-1: PORTA FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RA0/AN0	RA0	0	O	DIG	LATA<0> data output; not affected by analog input.
		1	I	TTL	PORTA<0> data input; disabled when analog input enabled.
	AN0	1	I	ANA	A/D Input Channel 0. Default input configuration on POR; does not affect digital output.
RA1/AN1	RA1	0	O	DIG	LATA<1> data output; not affected by analog input.
		1	I	TTL	PORTA<1> data input; disabled when analog input enabled.
	AN1	1	I	ANA	A/D Input Channel 1. Default input configuration on POR; does not affect digital output.
RA2/AN2/VREF-	RA2	0	O	DIG	LATA<2> data output; not affected by analog input. Disabled when CVREF output enabled.
		1	I	TTL	PORTA<2> data input. Disabled when analog functions enabled; disabled when CVREF output enabled.
	AN2	1	I	ANA	A/D Input Channel 2. Default input configuration on POR; not affected by analog output.
	VREF-	1	I	ANA	Comparator voltage reference low input and A/D voltage reference low input.
RA3/AN3/VREF+	RA3	0	O	DIG	LATA<3> data output; not affected by analog input.
		1	I	TTL	PORTA<3> data input; disabled when analog input enabled.
	AN3	1	I	ANA	A/D Input Channel 3. Default input configuration on POR.
	VREF+	1	I	ANA	Comparator voltage reference high input and A/D voltage reference high input.
RA4/T0CKI	RA4	0	O	DIG	LATA<4> data output
		1	I	ST	PORTA<4> data input; default configuration on POR.
	T0CKI	x	I	ST	Timer0 clock input.
RA5/AN4/HLVDIN	RA5	0	O	DIG	LATA<5> data output; not affected by analog input.
		1	I	TTL	PORTA<5> data input; disabled when analog input enabled.
	AN4	1	I	ANA	A/D Input Channel 4. Default configuration on POR.
	HLVDIN	1	I	ANA	High/Low-Voltage Detect external trip point input.
OSC2/CLKO/RA6	OSC2	x	O	ANA	Main oscillator feedback output connection (XT, HS and LP modes).
	CLKO	x	O	DIG	System cycle clock output (FOSC/4) in all oscillator modes except RCIO, INTIO2 and ECIO.
	RA6	0	O	DIG	LATA<6> data output. Enabled in RCIO, INTIO2 and ECIO modes only.
		1	I	TTL	PORTA<6> data input. Enabled in RCIO, INTIO2 and ECIO modes only.
OSC1/CLKI/RA7	OSC1	x	I	ANA	Main oscillator input connection.
	CLKI	x	I	ANA	Main clock input connection.
	RA7	0	O	DIG	LATA<7> data output. Disabled in External Oscillator modes.
		1	I	TTL	PORTA<7> data input. Disabled in External Oscillator modes.

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST= Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

# PIC18F6310/6410/8310/8410

**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 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5	RA4	RA3	RA2	RA1	RA0	66
LATA	LATA7 <sup>(1)</sup>	LATA6 <sup>(1)</sup>	LATA Output Latch Register						66
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	PORTA Data Direction Register						66
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	64

**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'.

# PIC18F6310/6410/8310/8410

## 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.

### EXAMPLE 11-2: INITIALIZING PORTB

```
CLRF   PORTB   ; Initialize PORTB by
              ; clearing output
              ; data latches
CLRF   LATB    ; Alternate method
              ; to clear output
              ; data latches
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 a Power-on Reset.

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 power-managed modes. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

- Any read or write of PORTB (except with the MOVFF (ANY), PORTB instruction). This will end the mismatch condition.
- Wait one T<sub>cy</sub> delay (for example, execute one NOP instruction).
- Clear flag bit, RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB will end the mismatch condition and allow flag bit, RBIF, to be cleared after a one T<sub>cy</sub> delay.

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.

For 80-pin devices, RB3 can be configured as the alternate peripheral pin for the CCP2 module by clearing the CCP2MX Configuration bit. This applies only when the device is in one of the operating modes other than the default Microcontroller mode. If the device is in Microcontroller mode, the alternate assignment for CCP2 is RE7. As with other CCP2 configurations, the user must ensure that the TRISB<3> bit is set appropriately for the intended operation.



# PIC18F6310/6410/8310/8410

**TABLE 11-3: PORTB FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RB0/INT0	RB0	0	O	DIG	LATB<0> data output.
		1	I	TTL	PORTB<0> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	INT0	1	I	ST	External Interrupt 0 input.
RB1/INT1	RB1	0	O	DIG	LATB<1> data output.
		1	I	TTL	PORTB<1> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	INT1	1	I	ST	External Interrupt 1 input.
RB2/INT2	RB2	0	O	DIG	LATB<2> data output.
		1	I	TTL	PORTB<2> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	INT2	1	I	ST	External Interrupt 2 input.
RB3/INT3/ CCP2	RB3	0	O	DIG	LATB<3> data output.
		1	I	TTL	PORTB<3> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	INT3	1	I	ST	External Interrupt 3 input.
	CCP2 <sup>(1)</sup>	0	O	DIG	CCP2 compare output and CCP2 PWM output; takes priority over port data.
1		I	ST	CCP2 capture input.	
RB4/KBI0	RB4	0	O	DIG	LATB<4> data output.
		1	I	TTL	PORTB<4> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	KBI0	1	I	TTL	Interrupt-on-change pin.
RB5/KBI1	RB5	0	O	DIG	LATB<5> data output
		1	I	TTL	PORTB<5> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	KBI1	1	I	TTL	Interrupt-on-change pin.
RB6/KBI2/PGC	RB6	0	O	DIG	LATB<6> data output
		1	I	TTL	PORTB<6> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	KBI2	1	I	TTL	Interrupt-on-change pin.
	PGC	x	I	ST	Serial execution (ICSP™) clock input for ICSP and ICD operation. <sup>(2)</sup>
RB7/KBI3/PGD	RB7	0	O	DIG	LATB<7> data output.
		1	I	TTL	PORTB<7> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared.
	KBI3	1	I	TTL	Interrupt-on-change pin.
	PGD	x	O	DIG	Serial execution data output for ICSP and ICD operation. <sup>(2)</sup>
		x	I	ST	Serial execution data input for ICSP and ICD operation. <sup>(2)</sup>

**Legend:** O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared (Microprocessor, Extended Microcontroller and Microcontroller with Boot Block modes, 80-pin devices only); default assignment is RC1.

**2:** All other pin functions are disabled when ICSP or ICD operations are enabled.

# PIC18F6310/6410/8310/8410

**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	66
LATB	LATB Output Latch Register								66
TRISB	PORTB Data Direction Register								66
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	63
INTCON2	RBP $\bar{U}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	63
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	63

**Legend:** Shaded cells are not used by PORTB.

## 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. RC1 is normally configured by Configuration bit, CCP2MX, as the default peripheral pin of the CCP2 module (default/erased state, CCP2MX = 1).

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
```

# PIC18F6310/6410/8310/8410

**TABLE 11-5: PORTC FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RC0/T1OSO/T13CKI	RC0	0	O	DIG	LATC<0> data output.
		1	I	ST	PORTC<0> data input.
	T1OSO	x	O	ANA	Timer1 oscillator output; enabled when Timer1 oscillator is enabled. Disables digital I/O.
	T13CKI	1	I	ST	Timer1/Timer3 counter input.
RC1/T1OSI/CCP2	RC1	0	O	DIG	LATC<1> data output.
		1	I	ST	PORTC<1> data input.
	T1OSI	x	I	ANA	Timer1 oscillator input; enabled when Timer1 oscillator is enabled. Disables digital I/O.
	CCP2 <sup>(1)</sup>	0	O	DIG	CCP2 compare output and CCP2 PWM output; takes priority over port data.
		1	I	ST	CCP2 capture input
RC2/CCP1	RC2	0	O	DIG	LATC<2> data output.
		1	I	ST	PORTC<2> data input.
	CCP1	0	O	DIG	CCP1 compare output and CCP1 PWM output; takes priority over port data.
		1	I	ST	CCP1 capture input.
RC3/SCK/SCL	RC3	0	O	DIG	LATC<3> data output.
		1	I	ST	PORTC<3> data input.
	SCK	0	O	DIG	SPI clock output (MSSP module); takes priority over port data.
		1	I	ST	SPI clock input (MSSP module).
	SCL	0	O	DIG	I <sup>2</sup> C™ clock output (MSSP module); takes priority over port data.
		1	I	ST	I <sup>2</sup> C clock input (MSSP module); input type depends on module setting.
RC4/SDI/SDA	RC4	0	O	DIG	LATC<4> data output.
		1	I	ST	PORTC<4> data input.
	SDI	1	I	ST	SPI data input (MSSP module).
	SDA	1	O	DIG	I <sup>2</sup> C data output (MSSP module); takes priority over port data.
		1	I	ST	I <sup>2</sup> C data input (MSSP module); input type depends on module setting.
RC5/SDO	RC5	0	O	DIG	LATC<5> data output.
		1	I	ST	PORTC<5> data input.
	SDO	0	O	DIG	SPI data output (MSSP module); takes priority over port data.
RC6/TX1/CK1	RC6	0	O	DIG	LATC<6> data output.
		1	I	ST	PORTC<6> data input.
	TX1	1	O	DIG	Synchronous serial data output (EUSART module); takes priority over port data.
	CK1	1	O	DIG	Synchronous serial data input (EUSART module). User must configure as an input.
		1	I	ST	Synchronous serial clock input (EUSART module).
RC7/RX1/DT1	RC7	0	O	DIG	LATC<7> data output.
		1	I	ST	PORTC<7> data input.
	RX1	1	I	ST	Asynchronous serial receive data input (EUSART module)
	DT1	1	O	DIG	Synchronous serial data output (EUSART module); takes priority over port data.
		1	I	ST	Synchronous serial data input (EUSART module). User must configure as an input.

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input,  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Default assignment for CCP2 when CCP2MX Configuration bit is set.

# PIC18F6310/6410/8310/8410

**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	66
LATC	LATC Output Latch Register								66
TRISC	PORTC Data Direction Register								66

# PIC18F6310/6410/8310/8410

## 11.4 PORTD, TRISD and LATD Registers

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.

**Note:** On a Power-on Reset, these pins are configured as digital inputs.

In 80-pin devices, PORTD is multiplexed with the system bus as part of the external memory interface. I/O port and other functions are only available when the interface is disabled by setting the EBDIS bit (MEMCON<7>). When the interface is enabled, PORTD is the low-order byte of the multiplexed address/data bus (AD<7:0>). The TRISD bits are also overridden.

PORTD can also be configured to function as an 8-bit wide parallel microprocessor port by setting the PSPMODE Control bit (PSPCON<4>). In this mode, parallel port data takes priority over other digital I/O (but not the external memory interface). When the parallel port is active, the input buffers are TTL. For more information, refer to [Section 11.10 “Parallel Slave Port”](#).

### 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
```

# PIC18F6310/6410/8310/8410

**TABLE 11-7: PORTD FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RD0/AD0/PSP0	RD0	0	O	DIG	LATD<0> data output.
		1	I	ST	PORTD<0> data input.
	AD0 <sup>(2)</sup>	x	O	DIG	External memory interface, Address/Data Bit 0 output. <sup>(1)</sup>
		x	I	TTL	External memory interface, Data Bit 0 input. <sup>(1)</sup>
	PSP0	x	O	DIG	PSP read data output (LATD<0>); takes priority over port data.
x		I	TTL	PSP write data input.	
RD1/AD1/PSP1	RD1	0	O	DIG	LATD<1> data output.
		1	I	ST	PORTD<1> data input.
	AD1 <sup>(2)</sup>	x	O	DIG	External memory interface, Address/Data Bit 1 output. <sup>(1)</sup>
		x	I	TTL	External memory interface, Data Bit 1 input. <sup>(1)</sup>
	PSP1	x	O	DIG	PSP read data output (LATD<1>); takes priority over port data.
x		I	TTL	PSP write data input.	
RD2/AD2/PSP2	RD2	0	O	DIG	LATD<2> data output.
		1	I	ST	PORTD<2> data input.
	AD2 <sup>(2)</sup>	x	O	DIG	External memory interface, Address/Data Bit 2 output. <sup>(1)</sup>
		x	I	TTL	External memory interface, Data Bit 2 input. <sup>(1)</sup>
	PSP2	x	O	DIG	PSP read data output (LATD<2>); takes priority over port data.
x		I	TTL	PSP write data input.	
RD3/AD3/PSP3	RD3	0	O	DIG	LATD<3> data output.
		1	I	ST	PORTD<3> data input.
	AD3 <sup>(2)</sup>	x	O	DIG	External memory interface, Address/Data Bit 3 output. <sup>(1)</sup>
		x	I	TTL	External memory interface, Data Bit 3 input. <sup>(1)</sup>
	PSP3	x	O	DIG	PSP read data output (LATD<3>); takes priority over port data.
x		I	TTL	PSP write data input.	
RD4/AD4/PSP4	RD4	0	O	DIG	LATD<4> data output.
		1	I	ST	PORTD<4> data input.
	AD4 <sup>(2)</sup>	x	O	DIG	External memory interface, Address/Data Bit 4 output. <sup>(1)</sup>
		x	I	TTL	External memory interface, Data Bit 4 input. <sup>(1)</sup>
	PSP4	x	O	DIG	PSP read data output (LATD<4>); takes priority over port data.
x		I	TTL	PSP write data input.	
RD5/AD5/PSP5	RD5	0	O	DIG	LATD<5> data output.
		1	I	ST	PORTD<5> data input.
	AD5 <sup>(2)</sup>	x	O	DIG	External memory interface, Address/Data Bit 5 output. <sup>(1)</sup>
		x	I	TTL	External memory interface, Data Bit 5 input. <sup>(1)</sup>
	PSP5	x	O	DIG	PSP read data output (LATD<5>); takes priority over port data.
x		I	TTL	PSP write data input.	
RD6/AD6/PSP6	RD6	0	O	DIG	LATD<6> data output.
		1	I	ST	PORTD<6> data input.
	AD6 <sup>(2)</sup>	x	O	DIG-3	External memory interface, Address/Data Bit 6 output. <sup>(1)</sup>
		x	I	TTL	External memory interface, Data Bit 6 input. <sup>(1)</sup>
	PSP6	x	O	DIG	PSP read data output (LATD<6>); takes priority over port data.
x		I	TTL	PSP write data input.	

**Legend:** O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** External memory interface I/O takes priority over all other digital and PSP I/O.

**2:** Implemented on 80-pin devices only.

# PIC18F6310/6410/8310/8410

**TABLE 11-7: PORTD FUNCTIONS (CONTINUED)**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RD7/AD7/PSP7	RD7	0	O	DIG	LATD<7> data output.
		1	I	ST	PORTD<7> data input.
	AD7 <sup>(2)</sup>	x	O	DIG	External memory interface, Address/Data Bit 7 output <sup>(1)</sup> .
		x	I	TTL	External memory interface, Data Bit 7 input <sup>(1)</sup> .
	PSP7	x	O	DIG	PSP read data output (LATD<7>); takes priority over port data.
		x	I	TTL	PSP write data input.

**Legend:** O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** External memory interface I/O takes priority over all other digital and PSP I/O.

**2:** Implemented on 80-pin devices only.

**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	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	66
LATD	LATD Output Latch Register								66
TRISD	PORTD Data Direction Register								66



## 11.5 PORTE, TRISE and LATE Registers

PORTE is an 8-bit wide, bidirectional port. 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).

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.

All pins on PORTE are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

**Note:** On a Power-on Reset, these pins are configured as digital inputs.

When the device is operating in Microcontroller mode, pin RE7 can be configured as the alternate peripheral pin for the CCP2 module. This is done by clearing the CCP2MX Configuration bit.

In 80-pin devices, PORTE is multiplexed with the system bus as part of the external memory interface. I/O port and other functions are only available when the interface is disabled by setting the EBDIS bit (MEMCON<7>). When the interface is enabled (80-pin devices only), PORTE is the high-order byte of the multiplexed address/data bus (AD<15:8>). The TRISE bits are also overridden.

When the Parallel Slave Port is active on PORTD, three of the PORTE pins (RE0/AD8/RD, RE1/AD9/WR and RE2/AD10/CS) are configured as digital control inputs for the port. The control functions are summarized in [Table 11-9](#). The reconfiguration occurs automatically when the PSPMODE Control bit (PSPCON<4>) is set. Users must still make certain the corresponding TRISE bits are set to configure these pins as digital inputs.

### EXAMPLE 11-5: INITIALIZING PORTE

```
CLRF   PORTE   ; Initialize PORTE by
           ; clearing output
           ; data latches
CLRF   LATE    ; Alternate method
           ; to clear output
           ; data latches
MOVLW  03h    ; Value used to
           ; initialize data
           ; direction
MOVWF  TRISE   ; Set RE<1:0> as inputs
           ; RE<7:2> as outputs
```

# PIC18F6310/6410/8310/8410

**TABLE 11-9: PORTE FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RE0/AD8/RD	RE0	0	O	DIG	LATE<0> data output.
		1	I	ST	PORTE<0> data input.
	AD8 <sup>(3)</sup>	x	O	DIG	External memory interface, Address/Data Bit 8 output. <sup>(2)</sup>
		x	I	TTL	External memory interface, Data Bit 8 input. <sup>(2)</sup>
	R $\overline{D}$	1	I	TTL	Parallel Slave Port read enable control input.
RE1/AD9/WR	RE1	0	O	DIG	LATE<1> data output.
		1	I	ST	PORTE<1> data input.
	AD9 <sup>(3)</sup>	x	O	DIG	External memory interface, Address/Data Bit 9 output. <sup>(2)</sup>
		x	I	TTL	External memory interface, Data Bit 9 input. <sup>(2)</sup>
	W $\overline{R}$	1	I	TTL	Parallel Slave Port write enable control input.
RE2/AD10/CS	RE2	0	O	DIG	LATE<2> data output.
		1	I	ST	PORTE<2> data input.
	AD10 <sup>(3)</sup>	x	O	DIG	External memory interface, Address/Data Bit 10 output. <sup>(2)</sup>
		x	I	TTL	External memory interface, Data Bit 10 input. <sup>(2)</sup>
	C $\overline{S}$	1	I	TTL	Parallel Slave Port chip select control input.
RE3/AD11	RE3	0	O	DIG	LATE<3> data output.
		1	I	ST	PORTE<3> data input.
	AD11 <sup>(3)</sup>	x	O	DIG	External memory interface, Address/Data Bit 11 output. <sup>(2)</sup>
		x	I	TTL	External memory interface, Data Bit 11 input. <sup>(2)</sup>
RE4/AD12	RE4	0	O	DIG	LATE<4> data output.
		1	I	ST	PORTE<4> data input.
	AD12 <sup>(3)</sup>	x	O	DIG	External memory interface, Address/Data Bit 12 output. <sup>(2)</sup>
		x	I	TTL	External memory interface, Data Bit 12 input. <sup>(2)</sup>
RE5/AD13	RE5	0	O	DIG	LATE<5> data output.
		1	I	ST	PORTE<5> data input.
	AD13 <sup>(3)</sup>	x	O	DIG	External memory interface, Address/Data Bit 13 output. <sup>(2)</sup>
		x	I	TTL	External memory interface, Data Bit 13 input. <sup>(2)</sup>
RE6/AD14	RE6	0	O	DIG	LATE<6> data output.
		1	I	ST	PORTE<6> data input.
	AD14 <sup>(3)</sup>	x	O	DIG	External memory interface, Address/Data Bit 14 output. <sup>(2)</sup>
		x	I	TTL	External memory interface, Data Bit 14 input. <sup>(2)</sup>
RE7/CCP2/AD15	RE7	0	O	DIG	LATE<7> data output.
		1	I	ST	PORTE<7> data input.
	CCP2 <sup>(1)</sup>	0	O	DIG	CCP2 compare output and CCP2 PWM output; takes priority over port data.
		1	I	ST	CCP2 capture input.
	AD15 <sup>(3)</sup>	x	O	DIG	External memory interface, Address/Data Bit 15 output. <sup>(2)</sup>
		x	I	TTL	External memory interface, Data Bit 15 input. <sup>(2)</sup>

**Legend:** O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

- Note 1:** Alternate assignment for CCP2 when CCP2MX Configuration bit is cleared (all devices in Microcontroller mode).  
**2:** External memory interface I/O takes priority over all other digital and PSP I/O.  
**3:** Implemented on 80-pin devices only.

# PIC18F6310/6410/8310/8410

**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	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0	66
LATE	LATE Output Latch Register								66
TRISE	PORTE Data Direction Register								66

# PIC18F6310/6410/8310/8410

## 11.6 PORTF, LATF and TRISF Registers

PORTF is an 8-bit wide, bidirectional port. The corresponding Data Direction register is TRISF. Setting a TRISF bit (= 1) will make the corresponding PORTF pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISF bit (= 0) will make the corresponding PORTF pin an output (i.e., put the contents of the output latch on the selected pin).

The Output Latch register (LATF) is also memory mapped. Read-modify-write operations on the LATF register read and write the latched output value for PORTF.

All pins on PORTF are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

PORTF is multiplexed with several analog peripheral functions, including the A/D Converter and comparator inputs, as well as the comparator outputs. Pins, RF2 through RF6, may be used as comparator inputs or outputs by setting the appropriate bits in the CMCON register. To use RF<6:3> as digital inputs, it is also necessary to turn off the comparators.

**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.

**Note 1:** On a Power-on Reset, the RF<6:0> pins are configured as inputs and read as '0'.

**2:** To configure PORTF as a digital I/O, turn off the comparators and set the ADCON1 value.

### EXAMPLE 11-6: INITIALIZING PORTF

```
CLRF    PORTF    ; Initialize PORTF by
                ; clearing output
                ; data latches
CLRF    LATF     ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0x07     ;
MOVWF   CMCON    ; Turn off comparators
MOVLW   0x0F     ;
MOVWF   ADCON1   ; Set PORTF as digital I/O
MOVLW   0xCF     ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISF    ; Set RF3:RF0 as inputs
                ; RF5:RF4 as outputs
                ; RF7:RF6 as inputs
```

# PIC18F6310/6410/8310/8410

**TABLE 11-11: PORTF FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RF0/AN5	RF0	0	O	DIG	LATF<0> data output; not affected by analog input.
		1	I	ST	PORTF<0> data input; disabled when analog input is enabled.
	AN5	1	I	ANA	A/D Input Channel 5. Default configuration on POR.
RF1/AN6/C2OUT	RF1	0	O	DIG	LATF<1> data output; not affected by analog input.
		1	I	ST	PORTF<1> data input; disabled when analog input is enabled.
	AN6	1	I	ANA	A/D Input Channel 6. Default configuration on POR.
	C2OUT	0	O	DIG	Comparator 2 output; takes priority over port data.
RF2/AN7/C1OUT	RF2	0	O	DIG	LATF<2> data output; not affected by analog input.
		1	I	ST	PORTF<2> data input; disabled when analog input is enabled.
	AN7	1	I	ANA	A/D Input Channel 7. Default configuration on POR.
	C1OUT	0	O	TTL	Comparator 1 output; takes priority over port data.
RF3/AN8	RF3	0	O	DIG	LATF<3> data output; not affected by analog input.
		1	I	ST	PORTF<3> data input; disabled when analog input is enabled.
	AN8	1	I	ANA	A/D Input Channel 8 and Comparator C2+ input. Default input configuration on POR; not affected by analog output.
RF4/AN9	RF4	0	O	DIG	LATF<4> data output; not affected by analog input.
		1	I	ST	PORTF<4> data input; disabled when analog input is enabled.
	AN9	1	I	ANA	A/D Input Channel 9 and Comparator C2- input. Default input configuration on POR; does not affect digital output.
RF5/AN10/CVREF	RF5	0	O	DIG	LATF<5> data output; not affected by analog input. Disabled when CVREF output is enabled.
		1	I	ST	PORTF<5> data input; disabled when analog input is enabled. Disabled when CVREF output is enabled
	AN10	1	I	ANA	A/D Input Channel 10 and Comparator C1+ input. Default input configuration on POR.
	CVREF	x	O	ANA	Comparator voltage reference output. Enabling this feature disables digital I/O.
RF6/AN11	RF6	0	O	DIG	LATF<6> data output; not affected by analog input.
		1	I	ST	PORTF<6> data input; disabled when analog input is enabled.
	AN11	1	I	ANA	A/D Input Channel 11 and Comparator C1- input. Default input configuration on POR; does not affect digital output.
RF7/ $\overline{SS}$	RF7	0	O	DIG	LATF<7> data output.
		1	I	ST	PORTF<7> data input.
	$\overline{SS}$	1	I	TTL	Slave select input for MSSP (MSSP module).

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**TABLE 11-12: SUMMARY OF REGISTERS ASSOCIATED WITH PORTF**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
TRISF	PORTF Data Direction Register								66
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	66
LATF	LATF Output Latch Register								66
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	64
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	65
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	65

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PORTF.

# PIC18F6310/6410/8310/8410

## 11.7 PORTG, TRISG and LATG Registers

PORTG is a 6-bit wide, bidirectional port. The corresponding Data Direction register is TRISG. Setting a TRISG bit (= 1) will make the corresponding PORTG pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISG bit (= 0) will make the corresponding PORTG pin an output (i.e., put the contents of the output latch on the selected pin).

The Output Latch register (LATG) is also memory mapped. Read-modify-write operations on the LATG register, read and write the latched output value for PORTG.

PORTG is multiplexed with USART functions (Table 11-13). PORTG pins have Schmitt Trigger input buffers.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTG 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. The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register without concern due to peripheral overrides.

The sixth pin of PORTG (RG5/ $\overline{\text{MCLR}}$ /VPP) 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, RG5 also functions as the programming voltage input during programming.

**Note:** On a Power-on Reset, RG5 is enabled as a digital input only if Master Clear functionality is disabled. All other 5 pins are configured as digital inputs.

### EXAMPLE 11-7: INITIALIZING PORTG

```
CLRF   PORTG   ; Initialize PORTG by
           ; clearing output
           ; data latches
CLRF   LATG    ; Alternate method
           ; to clear output
           ; data latches
MOVLW  0x04    ; Value used to
           ; initialize data
           ; direction
MOVWF  TRISG   ; Set RG1:RG0 as outputs
           ; RG2 as input
           ; RG4:RG3 as inputs
```

# PIC18F6310/6410/8310/8410

**TABLE 11-13: PORTG FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RG0/CCP3	RG0	0	O	DIG	LATG<0> data output.
		1	I	ST	PORTG<0> data input.
	CCP3	0	O	DIG	CCP3 compare and PWM output; takes priority over port data.
		1	I	ST	CCP3 capture input.
RG1/TX2/CK2	R21	0	O	DIG	LATG<1> data output.
		1	I	ST	PORTG<1> data input.
	TX2	1	O	DIG	Synchronous serial data output (AUSART module); takes priority over port data.
		1	O	DIG	Synchronous serial data input (AUSART module). User must configure as an input.
RG2/RX2/DT2	RG2	0	O	DIG	LATG<2> data output.
		1	I	ST	PORTG<2> data input.
	RX2	1	I	ST	Asynchronous serial receive data input (AUSART module).
	DT2	1	O	DIG	Synchronous serial data output (AUSART module); takes priority over port data.
1		I	ST	Synchronous serial data input (AUSART module). User must configure as an input.	
RG3	RG3	0	O	DIG	LATG<3> data output.
		1	I	ST	PORTG<3> data input.
RG4	RG4	0	O	DIG	LATG<4> data output.
		1	I	ST	PORTG<4> data input.
RG5/MCLR/VPP	RG5	— <sup>(1)</sup>	I	ST	PORTG<5> data input; enabled when MCLRE Configuration bit is clear.
	MCLR	—	I	ST	External Master Clear input; enabled when MCLRE Configuration bit is set.
	VPP	—	I	ANA	High-Voltage Detection; used for ICSP™ mode entry detection. Always available, regardless of pin mode.

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input,  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** RG5 does not have a corresponding TRISG bit.

**TABLE 11-14: SUMMARY OF REGISTERS ASSOCIATED WITH PORTG**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
PORTG	—	—	RG5 <sup>(1)</sup>	RG4	RG3	RG2	RG1	RG0	66
LATG	—	—	—	LATG Output Latch Register					66
TRISG	—	—	—	PORTG Data Direction Register					66

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PORTG.

**Note 1:** RG5 is available as an input only when MCLR is disabled.

# PIC18F6310/6410/8310/8410

## 11.8 PORTH, LATH and TRISH Registers

**Note:** PORTH is only available on PIC18F8310/8410 devices.

PORTH is an 8-bit wide, bidirectional I/O port. The corresponding Data Direction register is TRISH. Setting a TRISH bit (= 1) will make the corresponding PORTH pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISH bit (= 0) will make the corresponding PORTH pin an output (i.e., put the contents of the output latch on the selected pin).

The Output Latch register (LATH) is also memory mapped. Read-modify-write operations on the LATH register, read and write the latched output value for PORTH.

All pins on PORTH are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

**Note:** On a Power-on Reset, these pins are configured as digital inputs.

When the external memory interface is enabled, four of the PORTH pins function as the high-order address lines for the interface. The address output from the interface takes priority over other digital I/O. The corresponding TRISH bits are also overridden.

### EXAMPLE 11-8: INITIALIZING PORTH

```
CLRF    PORTH    ; Initialize PORTH by
                ; clearing output
                ; data latches
CLRF    LATH     ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISH   ; Set RH3:RH0 as inputs
                ; RH5:RH4 as outputs
                ; RH7:RH6 as inputs
```



# PIC18F6310/6410/8310/8410

**TABLE 11-15: PORTH FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RH0/AD16	RH0	0	O	DIG	LATH<0> data output.
		1	I	ST	PORTH<0> data input.
	AD16	x	O	DIG	External memory interface, Address Line 16. Takes priority over port data.
RH1/AD17	RH1	0	O	DIG	LATH<1> data output.
		1	I	ST	PORTH<1> data input.
	AD17	x	O	DIG	External memory interface, Address Line 17. Takes priority over port data.
RH2/AD18	RH2	0	O	DIG	LATH<2> data output.
		1	I	ST	PORTH<2> data input.
	AD18	x	O	DIG	External memory interface, Address Line 18. Takes priority over port data.
RH3/AD19	RH3	0	O	DIG	LATH<3> data output.
		1	I	ST	PORTH<3> data input.
	AD19	x	O	DIG	External memory interface, Address Line 19. Takes priority over port data.
RH4	RH4	0	O	DIG	LATH<4> data output.
		1	I	ST	PORTH<4> data input.
RH5	RH5	0	O	DIG	LATH<5> data output.
		1	I	ST	PORTH<5> data input.
RH6	RH6	0	O	DIG	LATH<6> data output.
		1	I	ST	PORTH<6> data input.
RH7	RH7	0	O	DIG	LATH<7> data output.
		1	I	ST	PORTH<7> data input.

**Legend:** O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input,  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**TABLE 11-16: SUMMARY OF REGISTERS ASSOCIATED WITH PORTH**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
TRISH	PORTH Data Direction Register								65
PORTH	RH7	RH6	RH5	RH4	RH3	RH2	RH1	RH0	66
LATH	PORTH Output Latch Register								66

# PIC18F6310/6410/8310/8410

## 11.9 PORTJ, TRISJ and LATJ Registers

**Note:** PORTJ is available only on PIC18F8310/8410 devices.

PORTJ is an 8-bit wide, bidirectional port. The corresponding Data Direction register is TRISJ. Setting a TRISJ bit (= 1) will make the corresponding PORTJ pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISJ bit (= 0) will make the corresponding PORTJ pin an output (i.e., put the contents of the output latch on the selected pin).

The Output Latch register (LATJ) is also memory mapped. Read-modify-write operations on the LATJ register, read and write the latched output value for PORTJ.

All pins on PORTJ are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

**Note:** On a Power-on Reset, these pins are configured as digital inputs.

When the external memory interface is enabled, all of the PORTJ pins function as control outputs for the interface. This occurs automatically when the interface is enabled by clearing the EBDIS control bit (MEMCON<7>). The TRISJ bits are also overridden.

### EXAMPLE 11-9: INITIALIZING PORTJ

```
CLRF   PORTJ    ; Initialize PORTG by
                ; clearing output
                ; data latches
CLRF   LATJ     ; Alternate method
                ; to clear output
                ; data latches
MOVLW 0xCF     ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISJ   ; Set RJ3:RJ0 as inputs
                ; RJ5:RJ4 as output
                ; RJ7:RJ6 as inputs
```

# PIC18F6310/6410/8310/8410

**TABLE 11-17: PORTJ FUNCTIONS**

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RJ0/ALE	RJ0	0	O	DIG	LATJ<0> data output.
		1	I	ST	PORTJ<0> data input.
	ALE	x	O	DIG	External memory interface address latch enable control output; takes priority over digital I/O.
RJ1/OE	RJ1	0	O	DIG	LATJ<1> data output.
		1	I	ST	PORTJ<1> data input.
	OE	x	O	DIG	External memory interface output enable control output; takes priority over digital I/O.
RJ2/WRL	RJ2	0	O	DIG	LATJ<2> data output.
		1	I	ST	PORTJ<2> data input.
	WRL	x	O	DIG	External memory bus write low byte control; takes priority over digital I/O.
RJ3/WRH	RJ3	0	O	DIG	LATJ<3> data output.
		1	I	ST	PORTJ<3> data input.
	WRH	x	O	DIG	External memory interface write high byte control output; takes priority over digital I/O.
RJ4/BA0	RJ4	0	O	DIG	LATJ<4> data output.
		1	I	ST	PORTJ<4> data input.
	BA0	x	O	DIG	External Memory Interface Byte Address 0 control output; takes priority over digital I/O.
RJ5/CE	RJ5	0	O	DIG	LATJ<5> data output.
		1	I	ST	PORTJ<5> data input.
	CE	x	O	DIG	External memory interface chip enable control output; takes priority over digital I/O.
RJ6/LB	RJ6	0	O	DIG	LATJ<6> data output.
		1	I	ST	PORTJ<6> data input.
	LB	x	O	DIG	External memory interface lower byte enable control output; takes priority over digital I/O.
RJ7/UB	RJ7	0	O	DIG	LATJ<7> data output.
		1	I	ST	PORTJ<7> data input.
	UB	x	O	DIG	External memory interface upper byte enable control output; takes priority over digital I/O.

**Legend:** O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input,  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**TABLE 11-18: SUMMARY OF REGISTERS ASSOCIATED WITH PORTJ**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
PORTJ	RJ7	RJ6	RJ5	RJ4	RJ3	RJ2	RJ1	RJ0	66
LATJ	LATJ Output Latch Register								66
TRISJ	PORTJ Data Direction Register								65

# PIC18F6310/6410/8310/8410

## 11.10 Parallel Slave Port

PORTD can also function as an 8-bit wide Parallel Slave Port (PSP), or microprocessor port, when control bit, PSPMODE (PSPCON<4>), is set. It is asynchronously readable and writable by the external world through  $\overline{RD}$  control input pin, RE0/ $\overline{RD}$  and  $\overline{WR}$  control input pin, RE1/ $\overline{WR}$ .

**Note:** For PIC18F8310/8410 devices, the Parallel Slave Port is available only in Microcontroller mode.

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 bit, PSPMODE, enables port pin, RE0/ $\overline{RD}$ , to be the  $\overline{RD}$  input, RE1/ $\overline{WR}$  to be the  $\overline{WR}$  input and RE2/ $\overline{CS}$  to be 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).

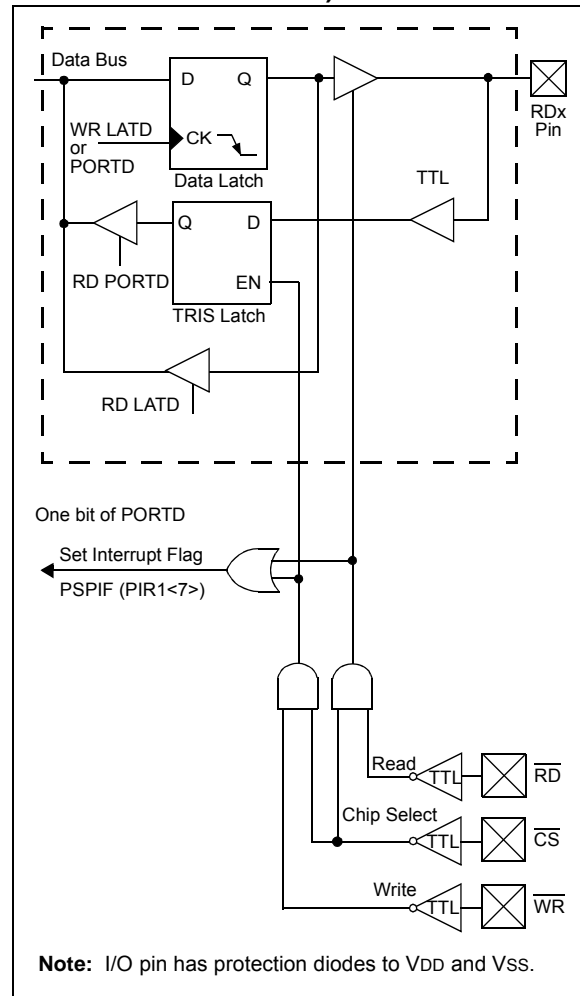
A write to the PSP occurs when both the  $\overline{CS}$  and  $\overline{WR}$  lines are first detected low and ends when either are detected high. The PSPIF and IBF flag bits are both set when the write ends.

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)**



# PIC18F6310/6410/8310/8410

**REGISTER 11-1: PSPCON: PARALLEL SLAVE PORT CONTROL REGISTER**

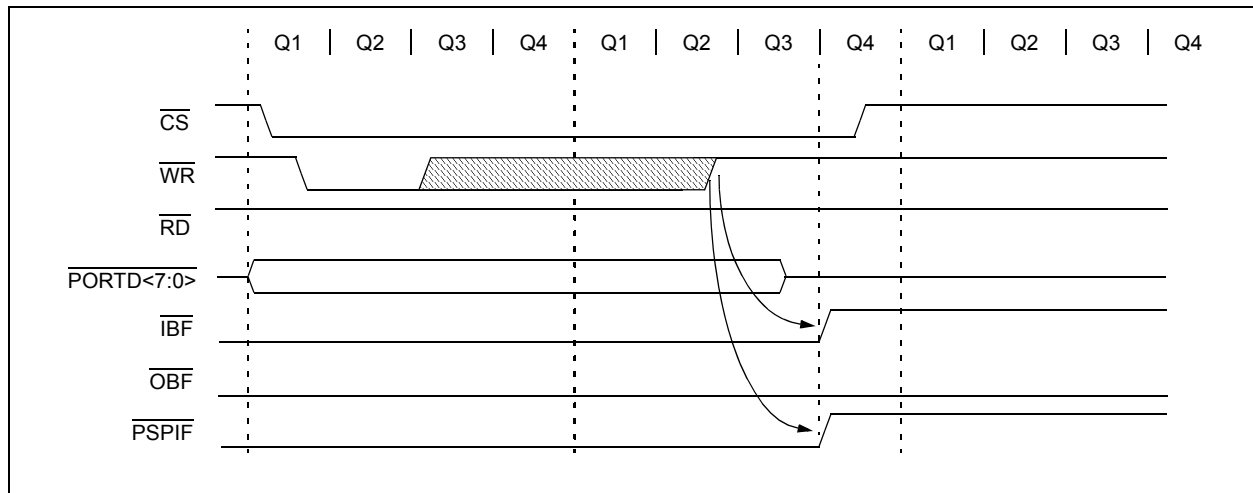
R-0	R-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
IBF	OBF	IBOV	PSPMODE	—	—	—	—
bit 7							bit 0

**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

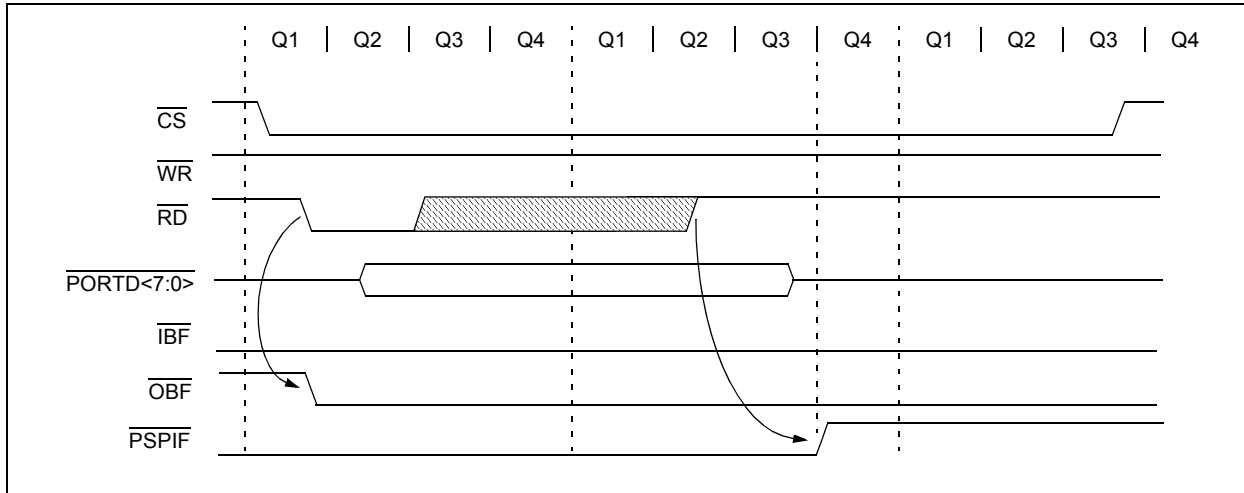
- bit 7            **IBF:** Input Buffer Full Status bit  
                  1 = A word has been received and is waiting to be read by the CPU  
                  0 = No word has been received
- bit 6            **OBF:** Output Buffer Full Status bit  
                  1 = The output buffer still holds a previously written word  
                  0 = The output buffer has been read
- bit 5            **IBOV:** Input Buffer Overflow Detect bit  
                  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 3-0        **Unimplemented:** Read as '0'

**FIGURE 11-3: PARALLEL SLAVE PORT WRITE WAVEFORMS**



# PIC18F6310/6410/8310/8410

**FIGURE 11-4: PARALLEL SLAVE PORT READ WAVEFORMS**



**TABLE 11-19: 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	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	66
LATD	LATD Output Latch Register								66
TRISD	PORTD Data Direction Register								66
PORTE	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0	66
LATE	LATE Output Latch Register								66
TRISE	PORTE Data Direction Register								66
PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—	65
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Parallel Slave Port.

# PIC18F6310/6410/8310/8410

## 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 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	TOSE	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 ( $F_{osc}/4$ )
- bit 4      **TOSE:** 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

# PIC18F6310/6410/8310/8410

## 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 (T0CS = 0), 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

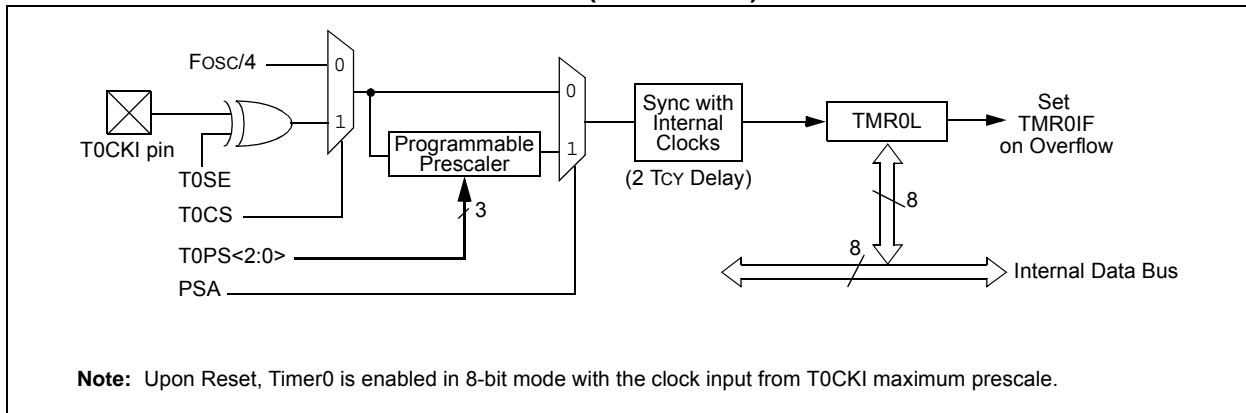
internal phase clock (Tosc). There is a delay between synchronization and the onset of incrementing the timer/counter.

## 12.2 Timer0 Reads and Writes in 16-Bit Mode

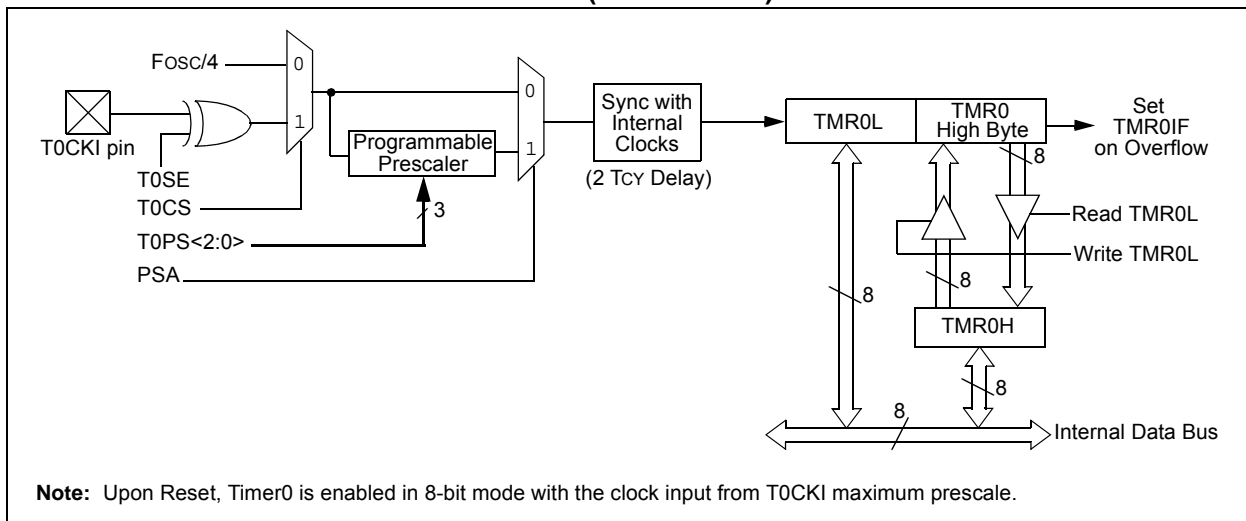
TMR0H is not the actual high byte of Timer0 in 16-bit mode; it is actually a buffered version of the real high byte of Timer0, which is not directly readable nor writable (refer to [Figure 12-2](#)). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the 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)**



**FIGURE 12-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)**





# PIC18F6310/6410/8310/8410

## 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 Module Low Byte Register								64
TMR0H	Timer0 Module High Byte Register								64
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	63
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	64
TRISA	PORTA Data Direction Register								66

**Legend:** Shaded cells are not used by Timer0.

# PIC18F6310/6410/8310/8410

---

NOTES:

# PIC18F6310/6410/8310/8410

## 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
- 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	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      **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

# PIC18F6310/6410/8310/8410

## 13.1 Timer1 Operation

Timer1 can operate in one of these modes:

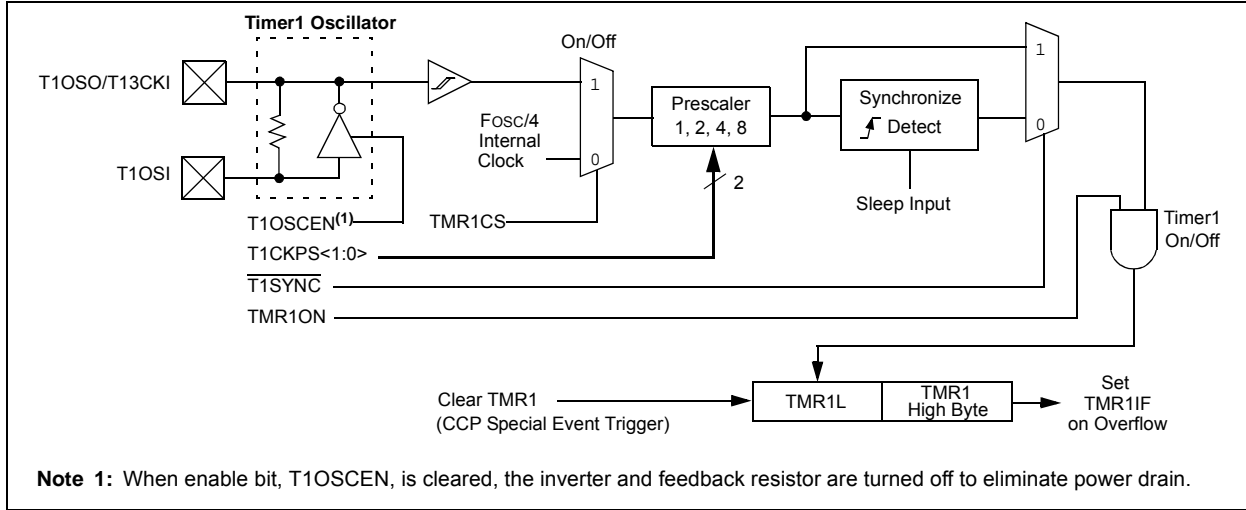
- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>). When TMR1CS is cleared (= 0), Timer1 increments on every internal instruction

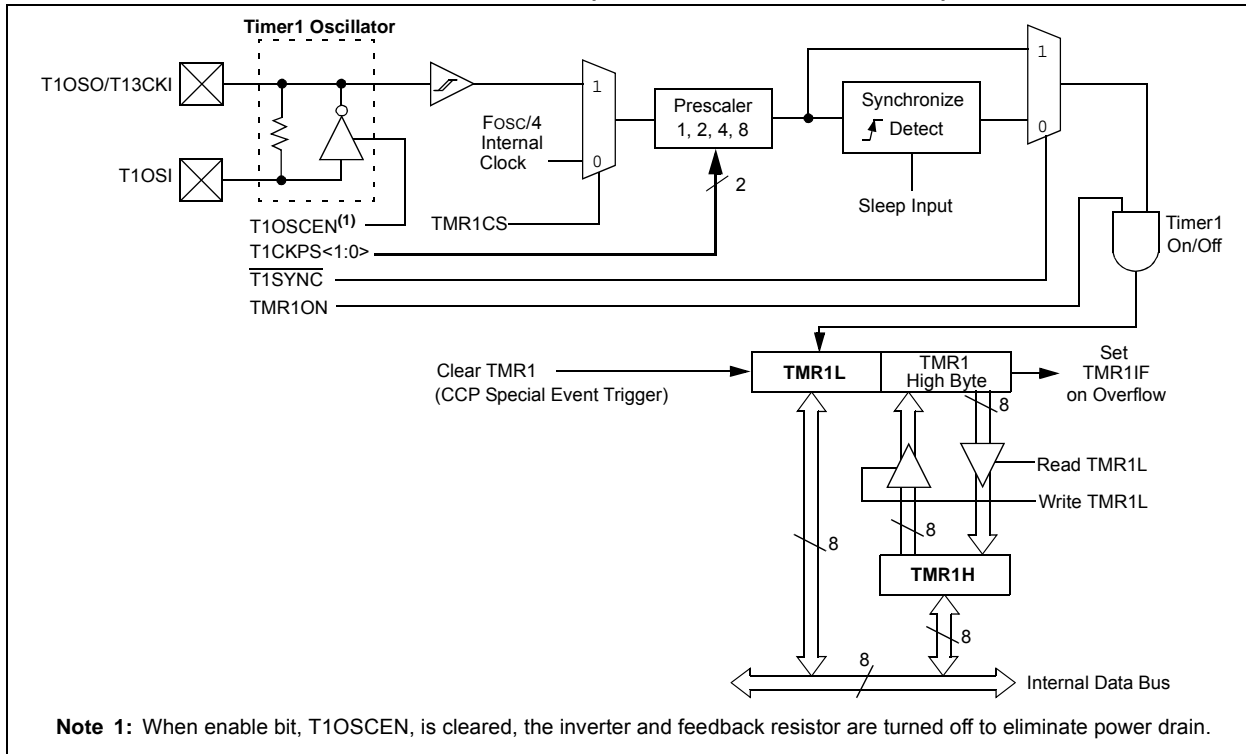
cycle ( $F_{osc}/4$ ). When the bit is set, Timer1 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

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**



**FIGURE 13-2: TIMER1 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**



## 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. 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.

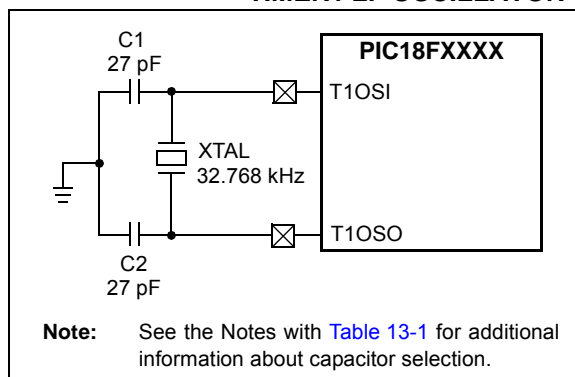
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

## 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**



**TABLE 13-1: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR**

Osc Type	Freq	C1	C2
LP	32 kHz	27 pF <sup>(1)</sup>	27 pF <sup>(1)</sup>

**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.

# PIC18F6310/6410/8310/8410

## 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 may be helpful when used on a single sided PCB, or in addition to a ground plane.

## 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 CCP1 or CCP2 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 CCP2 will also start an A/D conversion if the A/D module is enabled (see [Section 16.3.4 “Special Event Triggers”](#) 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 CCP2 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”](#), above), gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or supercapacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, `RTCISR`, shown in [Example 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 overflows.

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 Most Significant bit 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.

# PIC18F6310/6410/8310/8410

## 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   T1CON              ; 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
TMR1L	Holding Register for the Least Significant Byte of the 16-Bit TMR1 Register								64
TMR1H	Holding Register for the Most Significant Byte of the 16-Bit TMR1 Register								64
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYN $\bar{C}$	TMR1CS	TMR1ON	64

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

# PIC18F6310/6410/8310/8410

---

NOTES:



# PIC18F6310/6410/8310/8410

## 14.0 TIMER2 MODULE

The Timer2 timer module 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 ( $F_{osc}/4$ ). 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,  $\overline{MCLR}$  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

# PIC18F6310/6410/8310/8410

## 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/postscaler. 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>).

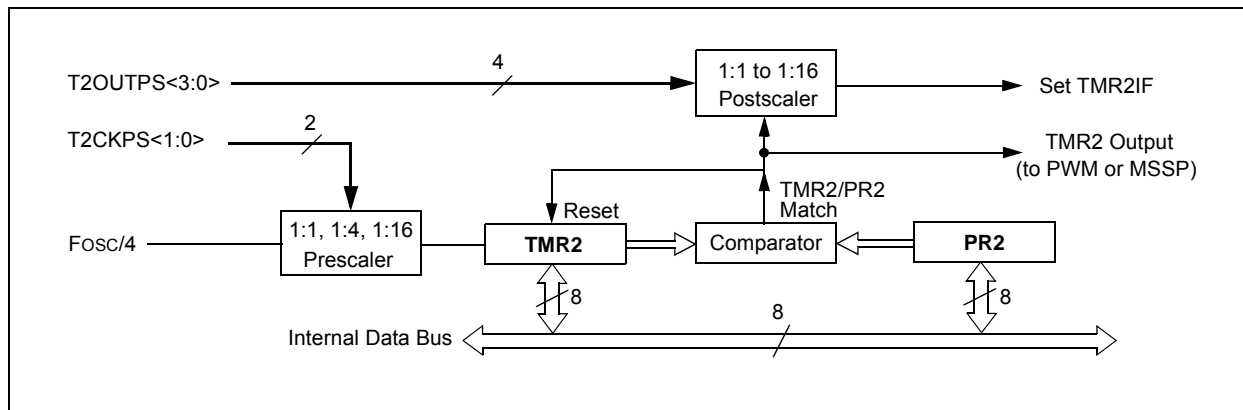
A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS<3:0> (T2CON<6:3>).

## 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 17.0 “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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
TMR2	Timer2 Register								64
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	64
PR2	Timer2 Period Register								64

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

# PIC18F6310/6410/8310/8410

## 15.0 TIMER3 MODULE

The Timer3 timer/counter module 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	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	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      **T3CCP<2:1>:** Timer3 and Timer1 to CCPx Enable bits  
 11 = Timer3 is the clock source for compare/capture of all CCP modules  
 10 = Timer3 is the clock source for compare/capture of CCP3,  
 Timer1 is the clock source for compare/capture of CCP1 and CCP2  
 01 = Timer3 is the clock source for compare/capture of CCP2 and CCP3,  
 Timer1 is the clock source for compare/capture of CCP1  
 00 = Timer1 is the clock source for compare/capture of all CCP 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      **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

# PIC18F6310/6410/8310/8410

## 15.1 Timer3 Operation

Timer3 can operate in one of three modes:

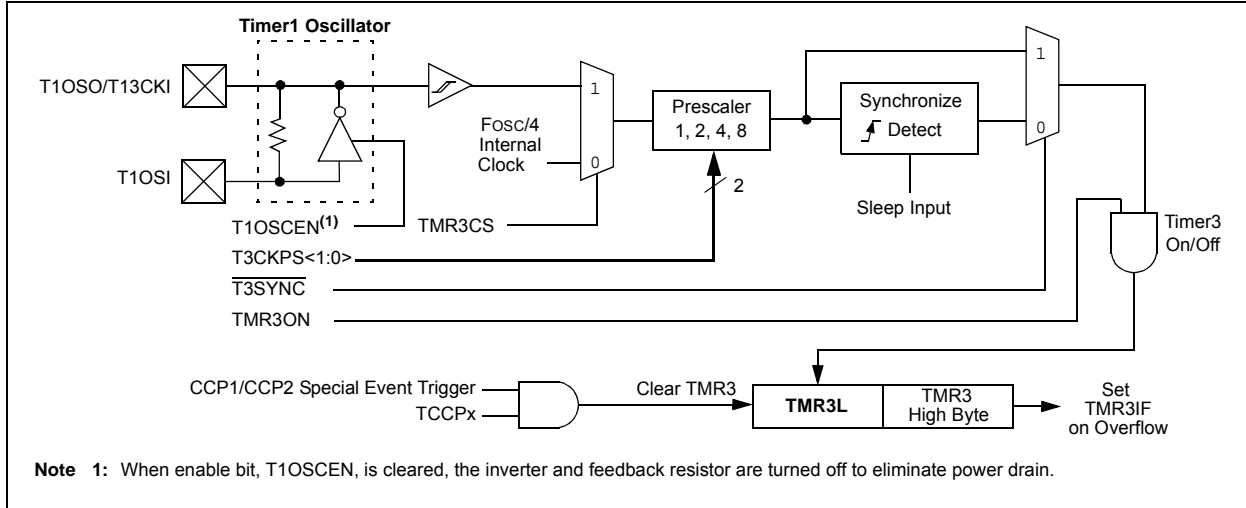
- Timer
- Synchronous counter
- Asynchronous counter

The operating mode is determined by the clock select bit, TMR3CS (T3CON<1>). When TMR3CS is cleared (= 0), Timer3 increments on every internal instruction

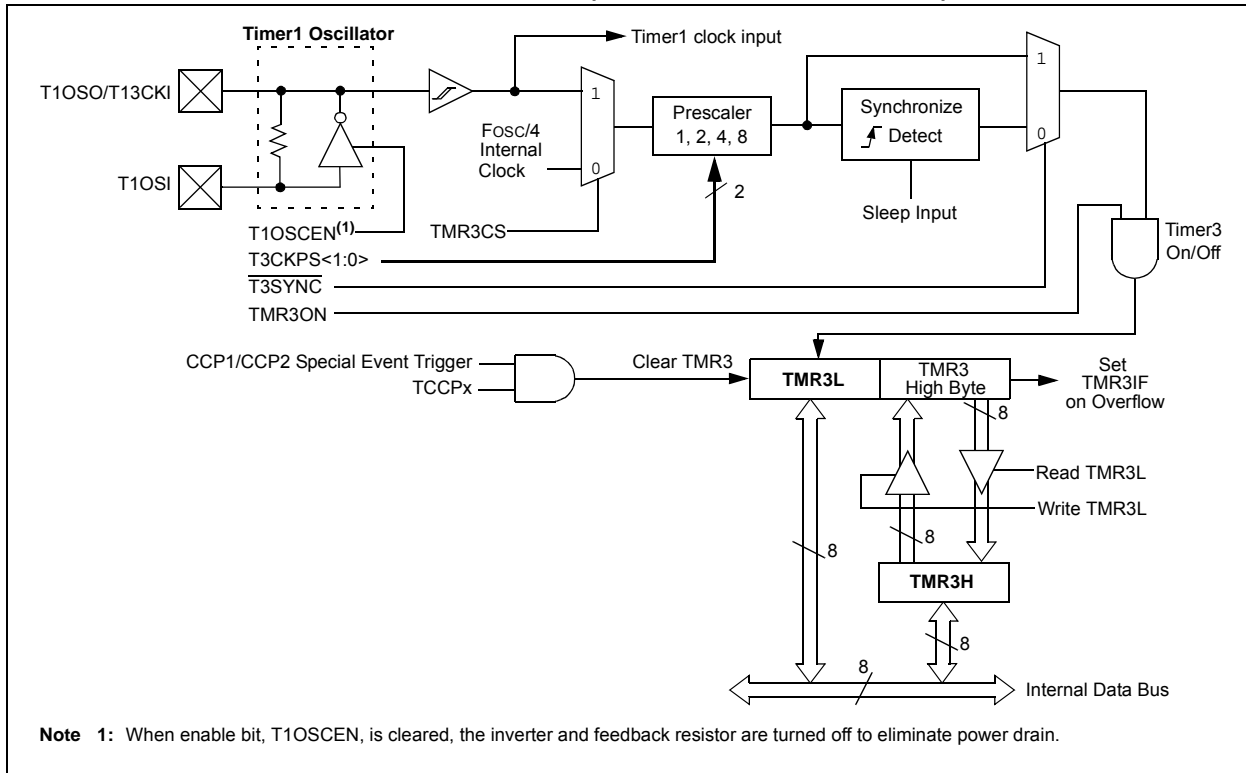
cycle ( $F_{osc}/4$ ). When the bit is set, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

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**



**FIGURE 15-2: TIMER3 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**



# PIC18F6310/6410/8310/8410

## 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 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 either the CCP1 or CCP2 modules is configured to generate a Special Event Trigger in Compare mode (CCP1M<3:0> or CCP2M<3:0> = 1011), this signal will reset Timer3. The trigger of CCP2 will also start an A/D conversion if the A/D module is enabled (see [Section 16.3.4 “Special Event Triggers”](#) 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 CCPR2H:CCPR2L 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 CCP2 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	63
PIR2	OSCFIF	CMIF	—	—	BCLIF	HLVDIF	TMR3IF	CCP2IF	65
PIE2	OSCFIE	CMIE	—	—	BCLIE	HLVDIE	TMR3IE	CCP2IE	65
IPR2	OSCFIP	CMIP	—	—	BCLIP	HLVDIP	TMR3IP	CCP2IP	65
TMR3L	Holding Register for the Least Significant Byte of the 16-Bit TMR3 Register								65
TMR3H	Holding Register for the Most Significant Byte of the 16-Bit TMR3 Register								65
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYN $\bar{C}$	TMR1CS	TMR1ON	64
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYN $\bar{C}$	TMR3CS	TMR3ON	65

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used by the Timer3 module.

# PIC18F6310/6410/8310/8410

---

NOTES:

# PIC18F6310/6410/8310/8410

## 16.0 CAPTURE/COMPARE/PWM (CCP) MODULES

PIC18F6310/6410/8310/8410 devices have three CCP (Capture/Compare/PWM) modules, labelled CCP1, CCP2 and CCP3. All modules implement standard Capture, Compare and Pulse-Width Modulation (PWM) modes.

Each CCP 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 CCP2, but are equally applicable to CCP1 and CCP3.

### REGISTER 16-1: CCPxCON: CCP1/CCP2/CCP3 CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

#### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared  
x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **DCxB<1:0>:** PWM Duty Cycle bit 1 and bit 0 for CCP Module x

Capture mode:  
Unused.

Compare mode:  
Unused.

PWM mode:

These bits are the two Least Significant bits (bit 1 and bit 0) of the 10-bit PWM Duty Cycle register. The eight Most Significant bits (DCx<9:2>) of the PWM Duty Cycle are found in CCPRL.

bit 3-0 **CCPxM<3:0>:** CCP Module x Mode Select bits

0000 = Capture/Compare/PWM disabled (resets CCPx module)

0001 = Reserved

0010 = Compare mode, toggle output on match (CCPxIF bit is set)

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 CCPx pin low; on compare match, force CCPx pin high (CCPxIF bit is set)

1001 = Compare mode: initialize CCPx pin high; on compare match, force CCPx pin low (CCPxIF bit is set)

1010 = Compare mode: generate software interrupt on compare match (CCPxIF bit is set, CCPx pin reflects I/O state)

1011 = Compare mode: trigger special event, reset timer, start A/D conversion on CCPx match (CCPxIF bit is set)<sup>(1,2)</sup>

11xxx = PWM mode

**Note 1:** The Special Event Trigger on CCP1 will reset the timer but not start an A/D conversion on a CCP1 match.

**2:** For CCP3, the Special Event Trigger is not available. This mode functions the same as Compare Generate Interrupt mode (CCP3M<3:0> = 1010).

# PIC18F6310/6410/8310/8410

## 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 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). All three 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.

Depending on the configuration selected, up to three timers may be active at once, with modules in the same configuration (Capture/Compare or PWM) sharing timer resources. The possible configurations are shown in Figure 16-1.

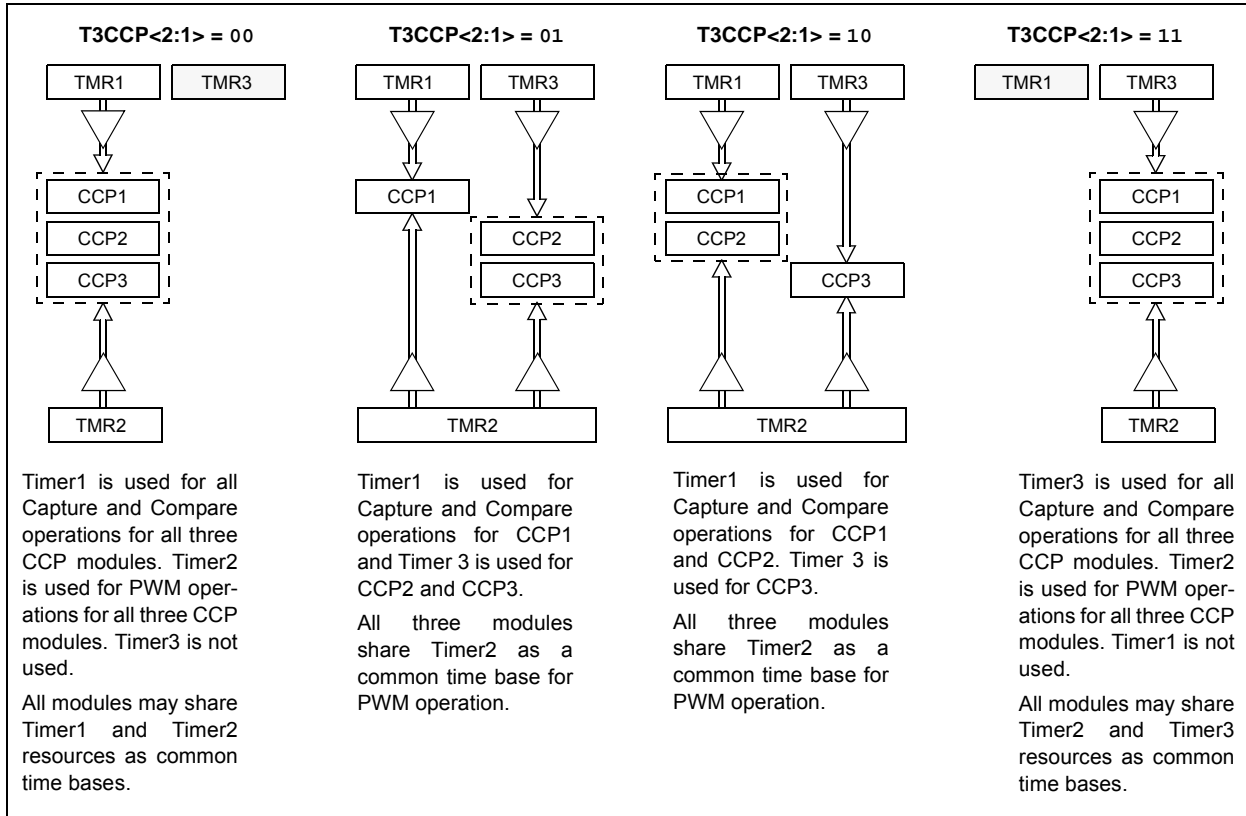
### 16.1.2 CCP2 PIN ASSIGNMENT

The CCP2MX Configuration bit determines if CCP2 is multiplexed to its default or alternate assignment. By default, CCP2 is assigned to RC1 (CCP2MX = 1). If CCP2MX is cleared, CCP2 is multiplexed with either RE7 or RB3 (RE7 is the only alternative assignment for 64-pin devices).

For any device in Microcontroller mode, the alternate CCP2 assignment is RE7. For 80-pin devices in Microprocessor, Extended Microcontroller or Microcontroller with Boot Block mode, the alternate assignment is RB3. Note that RE7 is the only alternative assignment for 64-pin devices.

Changing the pin assignment of CCP2 does not automatically change any requirements for configuring the port pin. Users must always verify that the appropriate TRIS register is configured correctly for CCP2 operation, regardless of where it is located.

**FIGURE 16-1: CCP AND TIMER INTERCONNECT CONFIGURATIONS**





# PIC18F6310/6410/8310/8410

## 16.2 Capture Mode

In Capture mode, the CCPR2H:CCPR2L register pair captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on the CCP2 pin (RC1 or RE7, 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, CCP2M<3:0> (CCP2CON<3:0>). When a capture is made, the interrupt request flag bit, CCP2IF (PIR2<1>), is set; it must be cleared in software. If another capture occurs before the value in register CCPR2 is read, the old captured value is overwritten by the new captured value.

### 16.2.1 CCP PIN CONFIGURATION

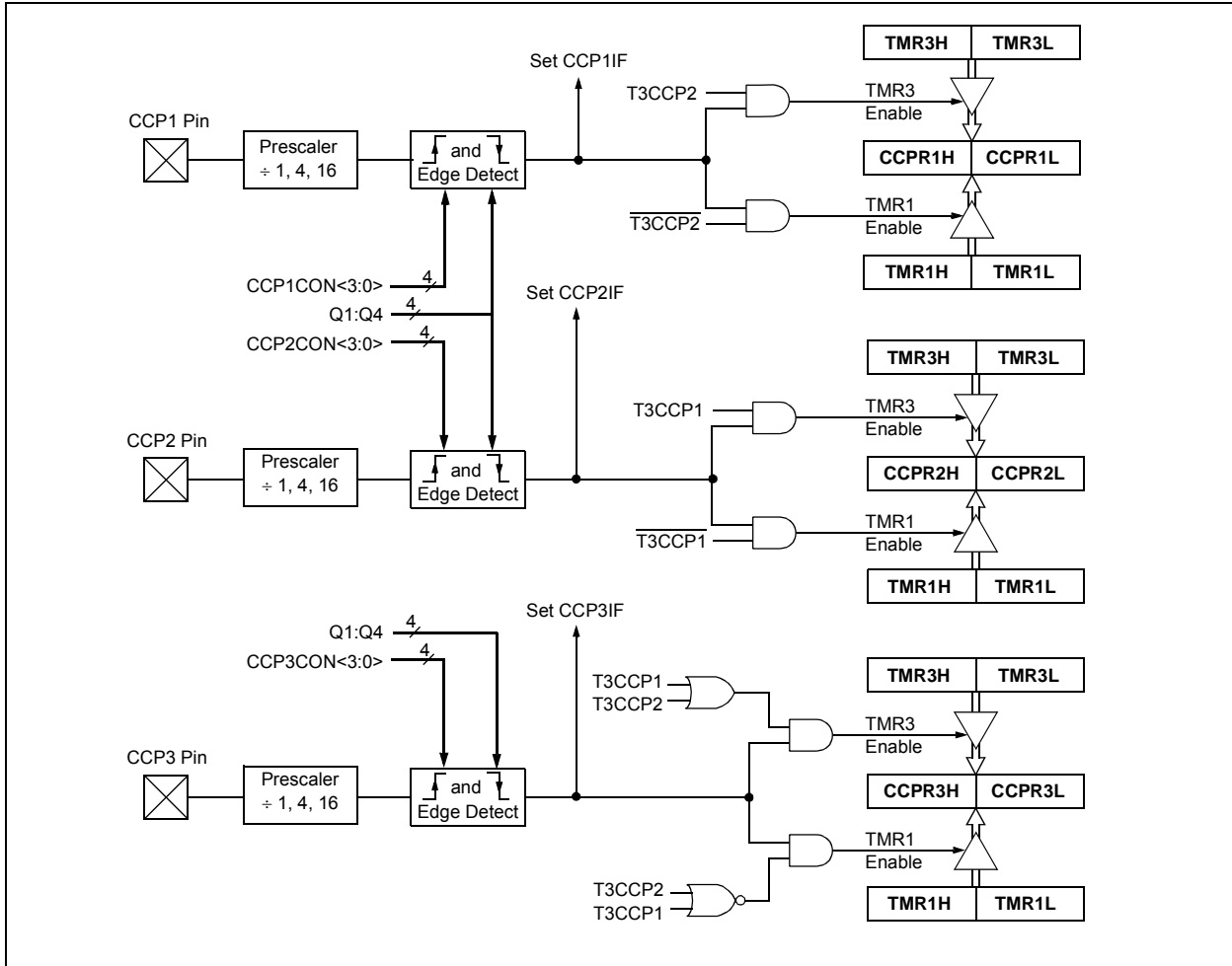
In Capture mode, the appropriate CCPx pin should be configured as an input by setting the corresponding TRIS direction bit.

**Note:** If RC1/CCP2 or RE7/CCP2 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”](#)).

**FIGURE 16-2: CAPTURE MODE OPERATION BLOCK DIAGRAM**



# PIC18F6310/6410/8310/8410

## 16.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit, CCP2IE (PIE2<1>), clear to avoid false interrupts and should clear the flag bit, CCP2IF, 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 (CCP2M<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.

### EXAMPLE 16-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF   CCP2CON      ; Turn CCP module off
MOVLW  NEW_CAPT_PS  ; Load WREG with the
                    ; new prescaler mode
                    ; value and CCP ON
MOVWF  CCP2CON      ; Load CCP2CON with
                    ; this value
```

## 16.3 Compare Mode

In Compare mode, the 16-bit CCPR2 register value is constantly compared against either the TMR1 or TMR3 register pair value. When a match occurs, the CCP2 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 (CCP2M<3:0>). At the same time, the interrupt flag bit, CCP2IF, is set.

## 16.3.1 CCP PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the appropriate TRIS bit.

**Note:** Clearing the CCPxCON register will force the RC1 or RE7 compare output latch (depending on device configuration) to the default low level. This is not the PORTC or PORTE 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 (CCP2M<3:0> = 1010), the CCP2 pin is not affected. Only a CCP interrupt is generated if enabled and the CCP2IE bit is set.

## 16.3.4 SPECIAL EVENT TRIGGERS

CCP1 and CCP2 are both 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 (CCP2M<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 CCPRx registers to serve as a programmable period register for either timer.

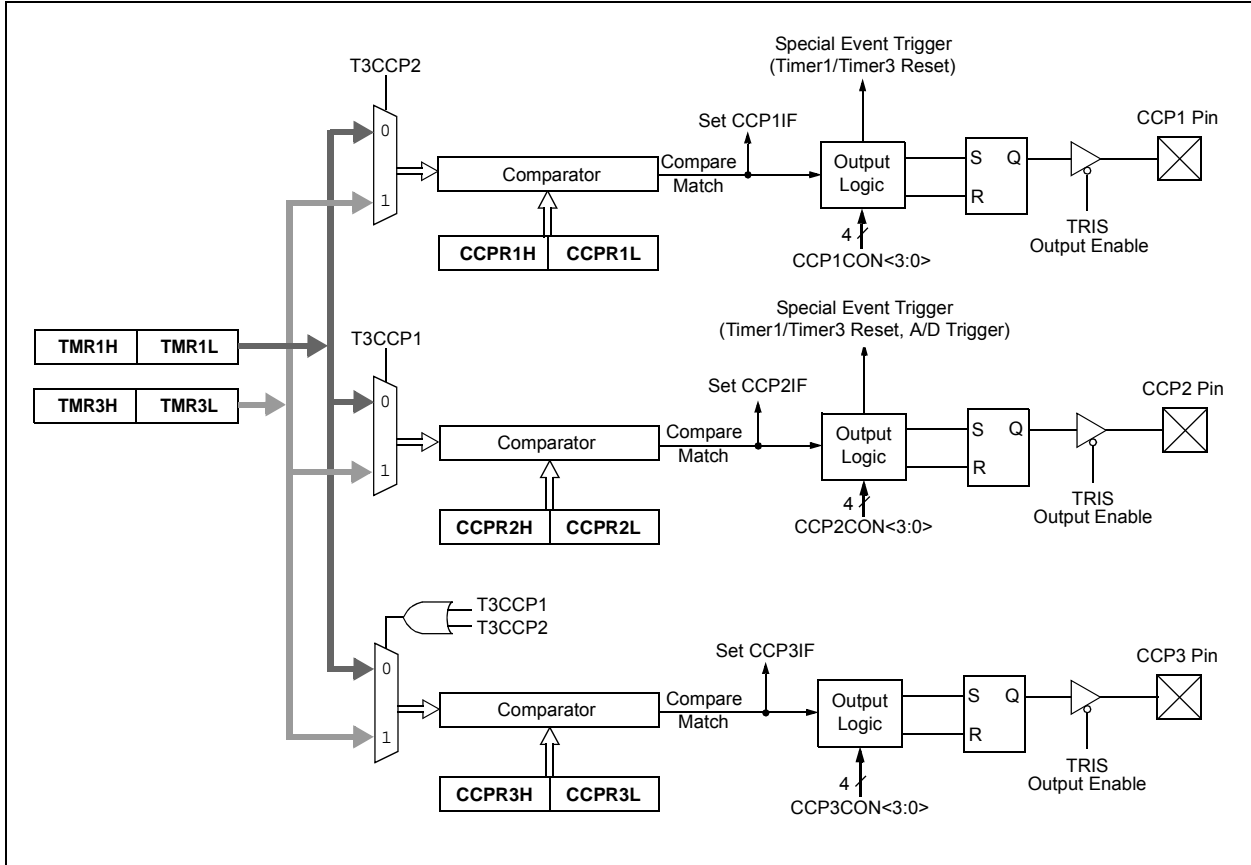
The Special Event Trigger for CCP2 can also start an A/D conversion. In order to do this, the A/D Converter must already be enabled.

**Note:** The Special Event Trigger of CCP1 only resets Timer1/Timer3 and cannot start an A/D conversion even when the A/D Converter is enabled.

CCP3 is not equipped with a Special Event Trigger. Selecting the Compare Special Event Trigger mode for this device (CCP3M<3:0> = 1011) is functionally the same as selecting the Generate Software Interrupt mode (CCP3M<3:0> = 1010).

# PIC18F6310/6410/8310/8410

FIGURE 16-3: COMPARE MODE OPERATION BLOCK DIAGRAM



# PIC18F6310/6410/8310/8410

**TABLE 16-2: 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	RBIF	TMR0IF	INT0IF	RBIF	63
RCON	IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	64
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
PIR2	OSCFIF	CMIF	—	—	BCLIF	HLVDIF	TMR3IF	CCP2IF	65
PIE2	OSCFIE	CMIE	—	—	BCLIE	HLVDIE	TMR3IE	CCP2IE	65
IPR2	OSCFIP	CMIP	—	—	BCLIP	HLVDIP	TMR3IP	CCP2IP	65
PIR3	—	—	RC2IF	TX2IF	—	—	—	CCP3IF	65
PIE3	—	—	RC2IE	TX2IE	—	—	—	CCP3IE	65
IPR3	—	—	RC2IP	TX2IP	—	—	—	CCP3IP	65
TRISB	PORTB Data Direction Register								66
TRISC	PORTC Data Direction Register								66
TRISE	PORTE Data Direction Register								66
TMR1L	Holding Register for the Least Significant Byte of the 16-Bit TMR1 Register								64
TMR1H	Holding Register for the Most Significant Byte of the 16-Bit TMR1 Register								64
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	64
TMR3H	Timer3 Register High Byte								65
TMR3L	Timer3 Register Low Byte								65
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	65
CCPR1L	Capture/Compare/PWM Register 1 (LSB)								65
CCPR1H	Capture/Compare/PWM Register 1 (MSB)								65
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	65
CCPR2L	Capture/Compare/PWM Register 2 (LSB)								65
CCPR2H	Capture/Compare/PWM Register 2 (MSB)								65
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	65
CCPR3L	Capture/Compare/PWM Register 3 (LSB)								65
CCPR3H	Capture/Compare/PWM Register 3 (MSB)								65
CCP3CON	—	—	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	65

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by Capture/Compare, Timer1 or Timer3.

## 16.4 PWM Mode

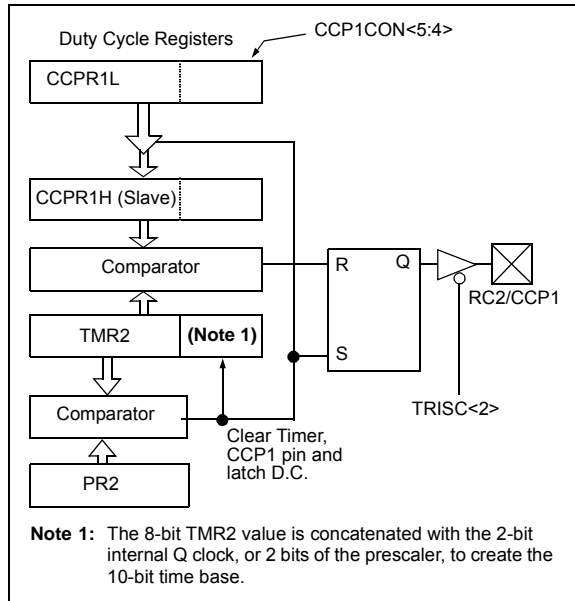
In Pulse-Width Modulation (PWM) mode, the CCP2 pin produces up to a 10-bit resolution PWM output. Since the CCP2 pin is multiplexed with a PORTC or PORTE data latch, the appropriate TRIS bit must be cleared to make the CCP2 pin an output.

**Note:** Clearing the CCP2CON register will force the RC1 or RE7 output latch (depending on device configuration) to the default low level. This is not the PORTC or PORTE I/O data latch.

Figure 16-4 shows a simplified block diagram of the CCP module in PWM mode.

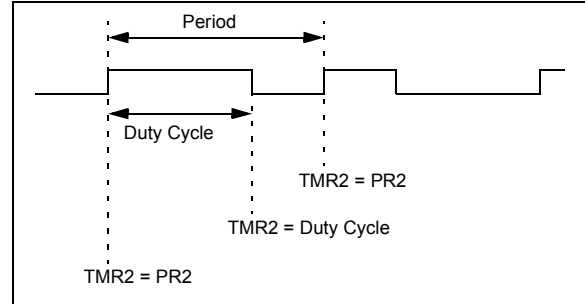
For a step-by-step procedure on how to set up the CCP module for PWM operation, see Section 16.4.3 “Setup for Pwm Operation”.

**FIGURE 16-4: SIMPLIFIED PWM BLOCK DIAGRAM**



A PWM output (Figure 16-5) 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 16-5: PWM OUTPUT**



### 16.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

**EQUATION 16-1:**

$$\text{PWM Period} = (\text{PR2} + 1) \cdot 4 \cdot \text{Tosc} \cdot (\text{TMR2 Prescale Value})$$

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 CCP2 pin is set (exception: if PWM duty cycle = 0%, the CCP2 pin will not be set)
- The PWM duty cycle is latched from CCPR2L into CCPR2H

**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.

# PIC18F6310/6410/8310/8410

## 16.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR2L register and to the CCP2CON<5:4> bits. Up to 10-bit resolution is available. The CCPR2L contains the eight MSBs and the CCP2CON<5:4> contains the two LSBs. This 10-bit value is represented by CCPR2L:CCP2CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

### EQUATION 16-2:

$$\text{PWM Duty Cycle} = (\text{CCPR2L:CCP2CON<5:4>}) \cdot \text{Tosc} \cdot (\text{TMR2 Prescale Value})$$

CCPR2L and CCP2CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR2H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR2H is a read-only register.

The CCPR2H 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 CCPR2H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP2 pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by the equation:

### EQUATION 16-3:

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{\text{OSC}}}{F_{\text{PWM}}}\right)}{\log(2)} \text{ bits}$$

**Note:** If the PWM duty cycle value is longer than the PWM period, the CCP2 pin will not be cleared.

## 16.4.3 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 CCPR2L register and CCP2CON<5:4> bits.
3. Make the CCP2 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 CCP2 module for PWM operation.

**TABLE 16-3: 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

# PIC18F6310/6410/8310/8410

**TABLE 16-4: 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	63
RCON	IPEN	SBOREN	—	RI	TO	PD	POR	BOR	64
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
TRISB	PORTB Data Direction Register								66
TRISC	PORTC Data Direction Register								66
TRISE	PORTE Data Direction Register								66
TMR2	Timer2 Register								64
PR2	Timer2 Period Register								64
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	64
CCPR1L	Capture/Compare/PWM Register 1 (LSB)								65
CCPR1H	Capture/Compare/PWM Register 1 (MSB)								65
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	65
CCPR2L	Capture/Compare/PWM Register 2 (LSB)								65
CCPR2H	Capture/Compare/PWM Register 2 (MSB)								65
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	65
CCPR3L	Capture/Compare/PWM Register 3 (LSB)								65
CCPR3H	Capture/Compare/PWM Register 3 (MSB)								65
CCP3CON	—	—	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	65

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PWM or Timer2.

# PIC18F6310/6410/8310/8410

---

NOTES:



## 17.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

### 17.1 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<sup>2</sup>C)
  - Full Master mode
  - Slave mode (with general address call)

The I<sup>2</sup>C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

### 17.2 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<sup>2</sup>C mode.

Additional details are provided under the individual sections.

### 17.3 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:

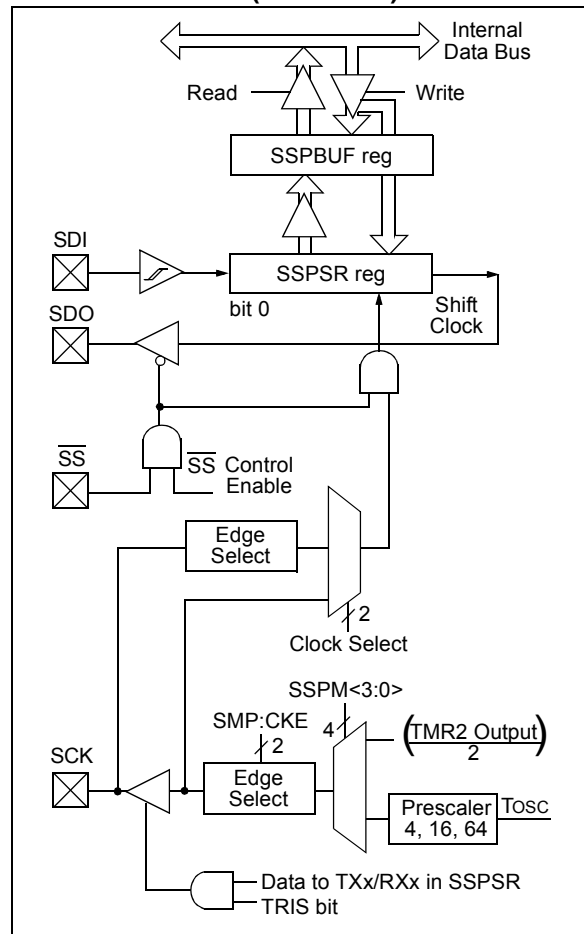
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Serial Clock (SCK)

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ( $\overline{SS}$ )

Figure 17-1 shows the block diagram of the MSSP module when operating in SPI mode.

**FIGURE 17-1: MSSP BLOCK DIAGRAM (SPI MODE)**



# PIC18F6310/6410/8310/8410

## 17.3.1 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 2 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 17-1: 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/ $\bar{A}$	P	S	R/ $\bar{W}$	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 Edge Select bit  
When CKP = 0:  
 1 = Data transmitted on rising edge of SCK  
 0 = Data transmitted on falling edge of SCK  
When CKP = 1:  
 1 = Data transmitted on falling edge of SCK  
 0 = Data transmitted on rising edge of SCK
- bit 5      **D/ $\bar{A}$ :** Data/Address bit  
 Used in I<sup>2</sup>C mode only.
- bit 4      **P:** Stop bit  
 Used in I<sup>2</sup>C™ mode only. This bit is cleared when the MSSP module is disabled; SSPEN is cleared.
- bit 3      **S:** Start bit  
 Used in I<sup>2</sup>C mode only.
- bit 2      **R/ $\bar{W}$ :** Read/Write bit Information  
 Used in I<sup>2</sup>C mode only.
- bit 1      **UA:** Update Address bit  
 Used in I<sup>2</sup>C 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

# PIC18F6310/6410/8310/8410

**REGISTER 17-2: 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 <sup>(1)</sup>	SSPEN <sup>(2)</sup>	CKP	SSPM3 <sup>(3)</sup>	SSPM2 <sup>(3)</sup>	SSPM1 <sup>(3)</sup>	SSPM0 <sup>(3)</sup>
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<sup>(1)</sup>  
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<sup>(2)</sup>  
 1 = Enables serial port and configures SCK, SDO, SDI and  $\overline{SS}$  as serial port pins  
 0 = Disables serial port and configures these pins as I/O port pins
- 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    **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits<sup>(3)</sup>  
 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 inputs or outputs.
- 3:** Bit combinations not specifically listed here are either reserved or implemented in I<sup>2</sup>C™ mode only.

# PIC18F6310/6410/8310/8410

## 17.3.2 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 17-1](#) 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.

### EXAMPLE 17-1: 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

**Note 1:** The SSPBUF register cannot be used with read-modify-write instructions, such as BCF, BTFSC and COMF, etc.

**2:** 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.

# PIC18F6310/6410/8310/8410

## 17.3.3 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 must have TRISC<4> bit cleared
- 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

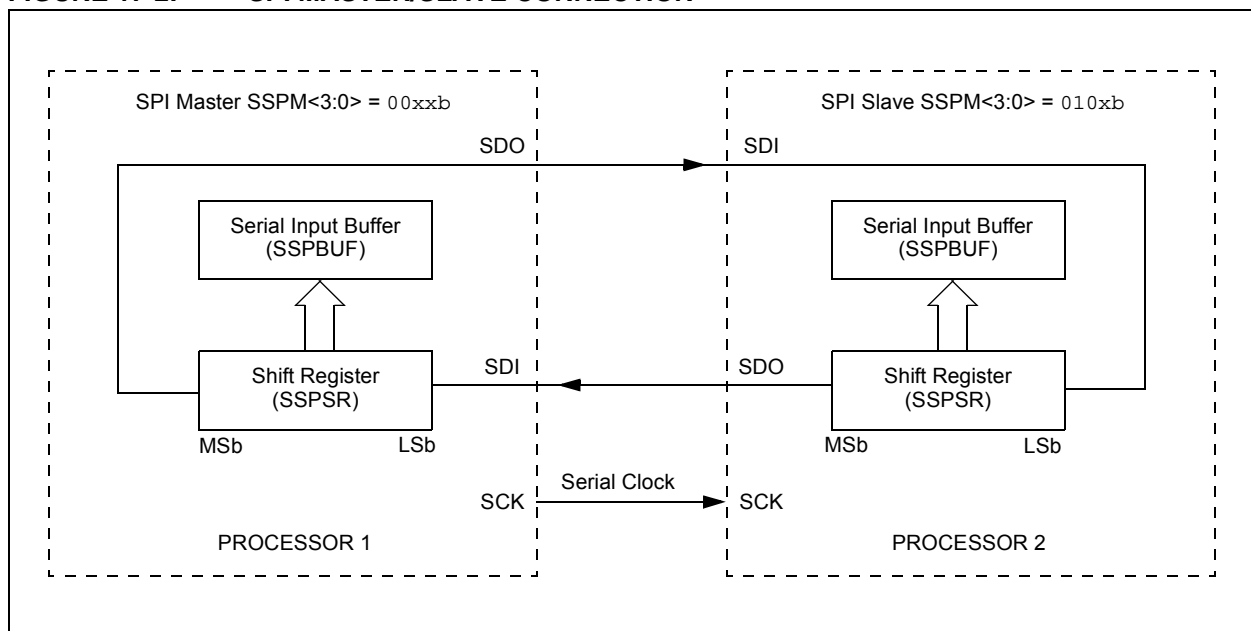
Any serial port function that is not desired may be overridden by programming the corresponding Data Direction (TRIS) register to the opposite value.

## 17.3.4 TYPICAL CONNECTION

Figure 17-2 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

FIGURE 17-2: SPI MASTER/SLAVE CONNECTION



# PIC18F6310/6410/8310/8410

## 17.3.5 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 17-2](#)) is to broadcast data by the software protocol.

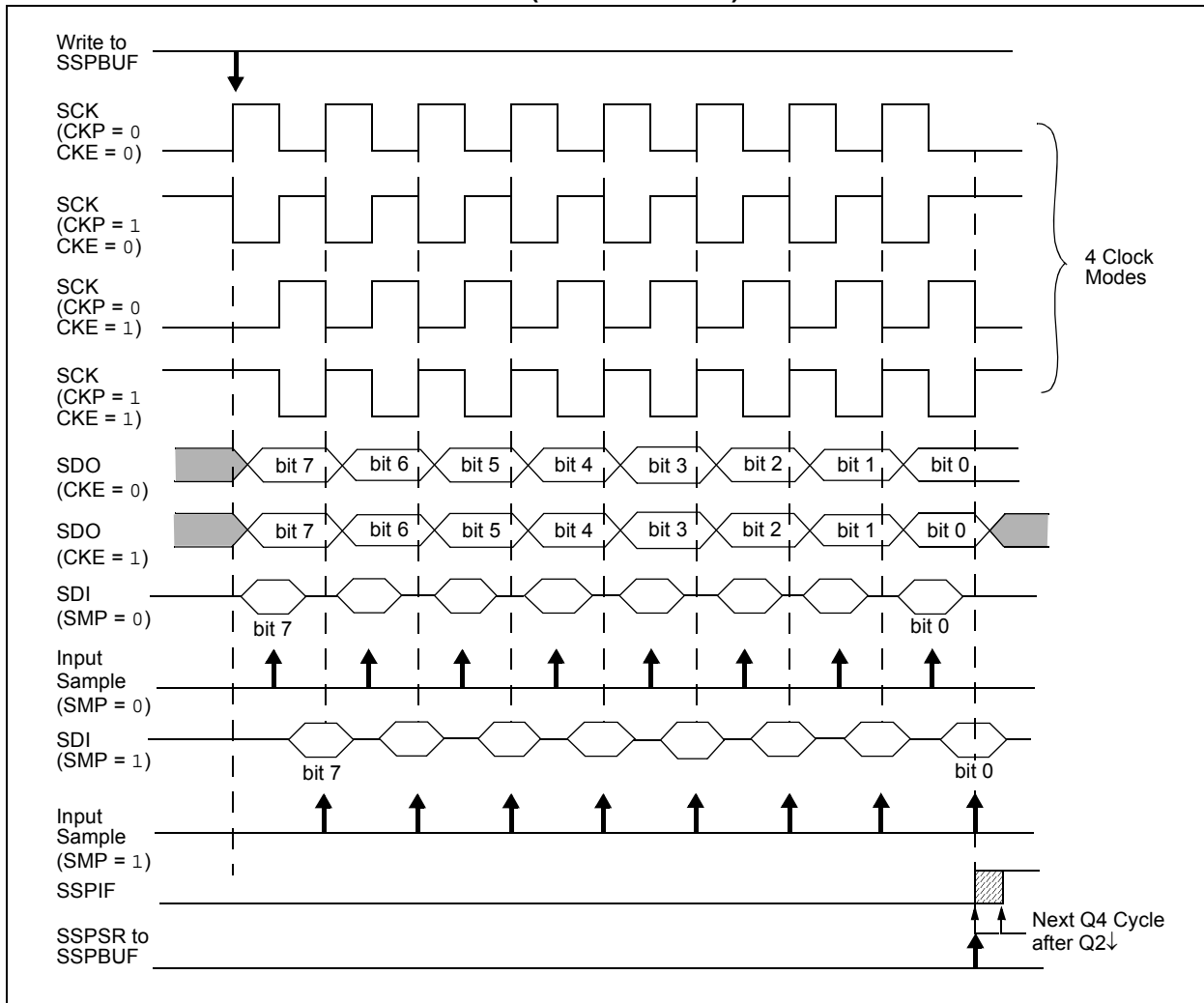
In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As 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 17-3](#), [Figure 17-5](#) and [Figure 17-6](#), where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- $F_{osc}/4$  (or  $T_{CY}$ )
- $F_{osc}/16$  (or  $4 \cdot T_{CY}$ )
- $F_{osc}/64$  (or  $16 \cdot T_{CY}$ )
- $\text{Timer2 output}/2$

[Figure 17-3](#) 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 17-3: SPI MODE WAVEFORM (MASTER MODE)**



## 17.3.6 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.

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.

## 17.3.7 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,

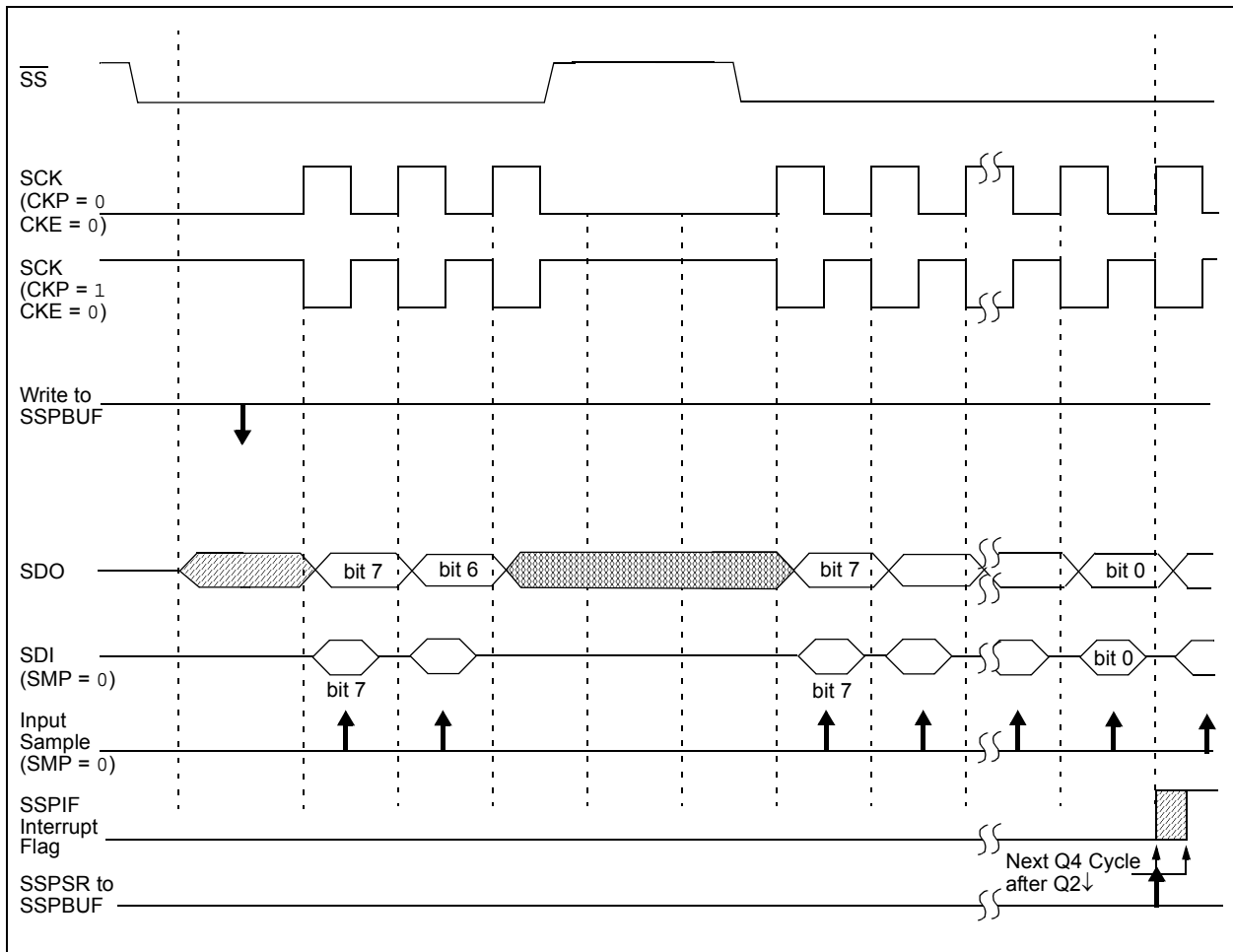
even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable, depending on the application.

- 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.

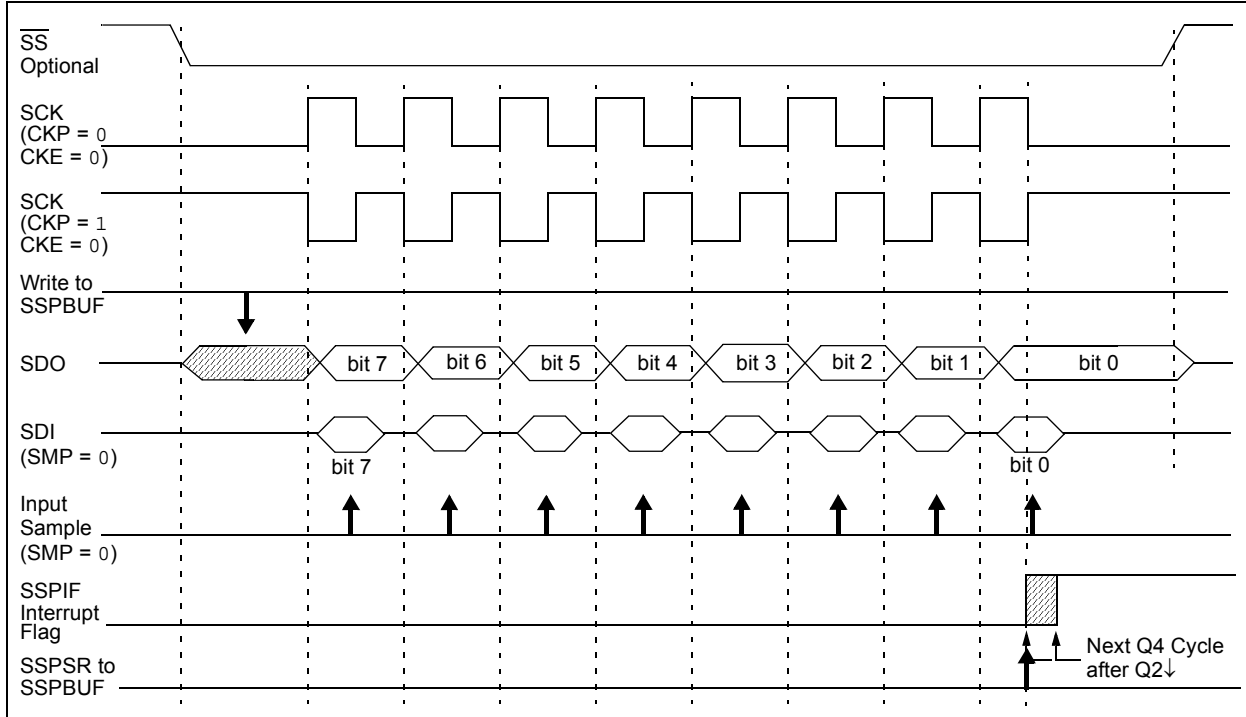
To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function) since it cannot create a bus conflict.

**FIGURE 17-4: SLAVE SYNCHRONIZATION WAVEFORM**

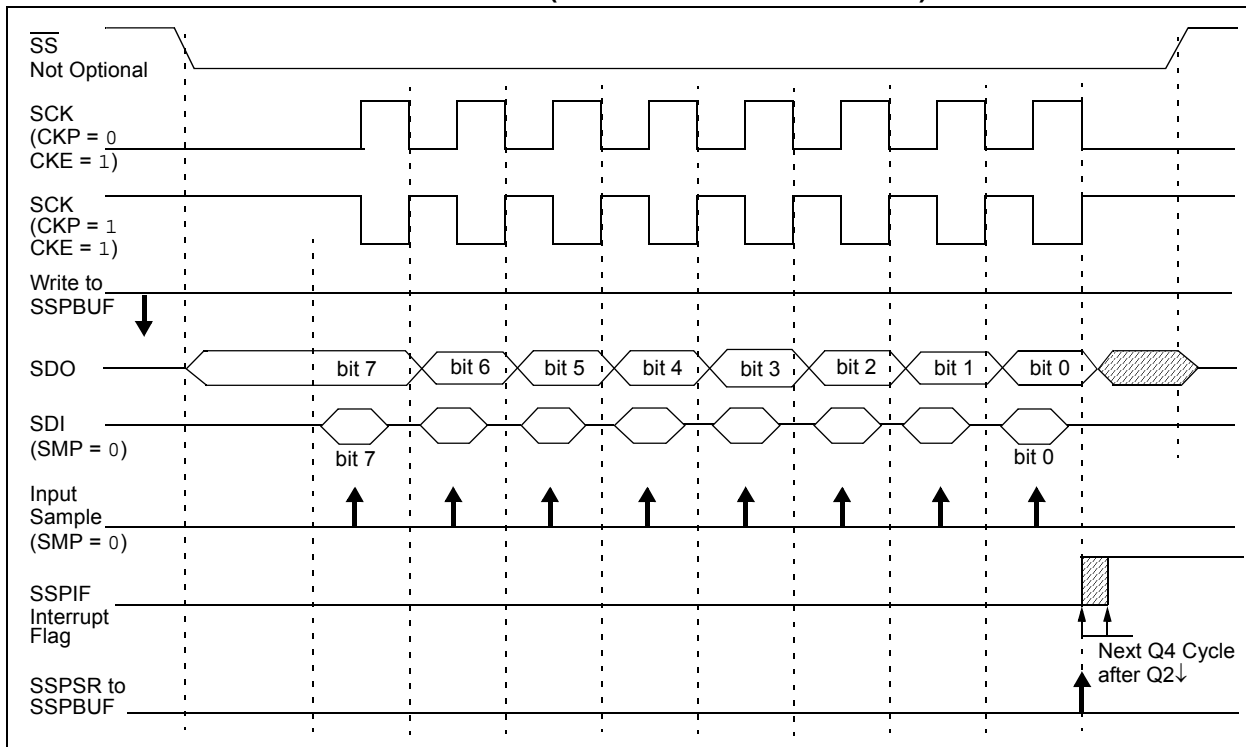


# PIC18F6310/6410/8310/8410

**FIGURE 17-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**



**FIGURE 17-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**





# PIC18F6310/6410/8310/8410

## 17.3.8 SLEEP OPERATION

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.

## 17.3.9 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

## 17.3.10 BUS MODE COMPATIBILITY

[Table 17-1](#) shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

**TABLE 17-1: 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 an SMP bit which controls when the data is sampled.

**TABLE 17-2: 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
TRISC	PORTC Data Direction Register								66
TRISF	PORTF Data Direction Register								66
SSPBUF	Master Synchronous Serial Port Receive Buffer/Transmit Register								64
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	64
SSPSTAT	SMP	CKE	D/Ā	P	S	R/W	UA	BF	64

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the MSSP in SPI mode.

# PIC18F6310/6410/8310/8410

## 17.4 I<sup>2</sup>C Mode

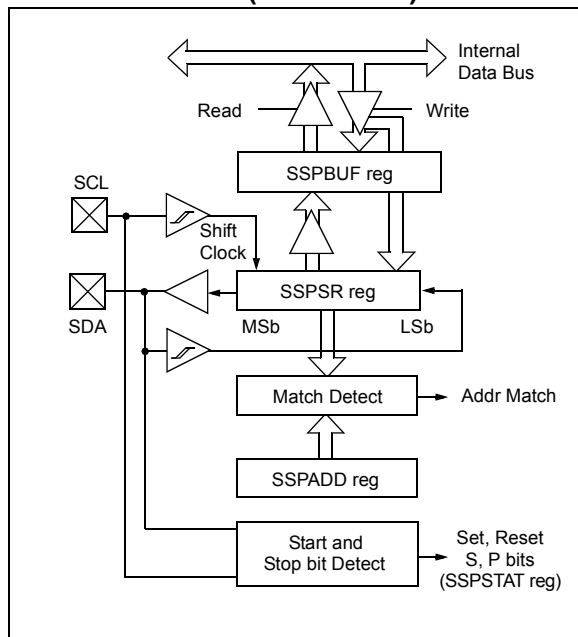
The MSSP module in I<sup>2</sup>C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications as well as 7-bit 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 through the TRISC<4:3> bits.

**FIGURE 17-7: MSSP BLOCK DIAGRAM (I<sup>2</sup>C™ MODE)**



### 17.4.1 REGISTERS

The MSSP module has six registers for I<sup>2</sup>C 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 I<sup>2</sup>C mode operation. The SSPCON1 and SSPCON2 registers are readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper 2 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 I<sup>2</sup>C Slave mode. When the MSSP is configured in Master mode, the lower 7 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.

# PIC18F6310/6410/8310/8410

**REGISTER 17-3: SSPSTAT: MSSP STATUS REGISTER (I<sup>2</sup>C™ MODE)**

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/ $\bar{A}$	P <sup>(1)</sup>	S <sup>(1)</sup>	R/ $\bar{W}$ <sup>(2,3)</sup>	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/ $\bar{A}$ :** Data/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<sup>(1)</sup>  
 1 = Indicates that a Stop bit has been detected last  
 0 = Stop bit was not detected last
- bit 3      **S:** Start bit<sup>(1)</sup>  
 1 = Indicates that a Start bit has been detected last  
 0 = Start bit was not detected last
- bit 2      **R/ $\bar{W}$ :** Read/Write bit Information (I<sup>2</sup>C mode only)  
In Slave mode:<sup>(2)</sup>  
 1 = Read  
 0 = Write  
In Master mode:<sup>(3)</sup>  
 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 Transmit mode:  
 1 = Receive complete, SSPBUF is full  
 0 = Receive not complete, SSPBUF is empty  
In Receive mode:  
 1 = Data transmit in progress (does not include the  $\bar{ACK}$  and Stop bits), SSPBUF is full  
 0 = Data transmit complete (does not include the  $\bar{ACK}$  and Stop bits), SSPBUF is empty

- 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<sup>2</sup>C mode only.

# PIC18F6310/6410/8310/8410

## REGISTER 17-4: SSPCON1: MSSP CONTROL REGISTER 1 (I<sup>2</sup>C™ 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
SMP	CKE	D/A	P <sup>(1)</sup>	S <sup>(1)</sup>	R/W <sup>(2,3)</sup>	UA	BF

bit 7

bit 0

- 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/A:** Data/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<sup>(1)</sup>  
 1 = Indicates that a Stop bit has been detected last  
 0 = Stop bit was not detected last
- bit 3 **S:** Start bit<sup>(1)</sup>  
 1 = Indicates that a Start bit has been detected last  
 0 = Start bit was not detected last
- bit 2 **R/W:** Read/Write bit Information (I<sup>2</sup>C mode only)  
In Slave mode:<sup>(2)</sup>  
 1 = Read  
 0 = Write  
In Master mode:<sup>(3)</sup>  
 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 Transmit mode:  
 1 = Receive complete, SSPBUF is full  
 0 = Receive not complete, SSPBUF is empty  
In Receive mode:  
 1 = Data transmit in progress (does not include the  $\overline{ACK}$  and Stop bits), SSPBUF is full  
 0 = Data transmit complete (does not include the  $\overline{ACK}$  and Stop bits), SSPBUF is empty

- Note 1:** This bit is cleared on Reset and when SSPEN is cleared.
- Note 2:** This bit holds the R/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  $\overline{ACK}$  bit.
- Note 3:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Idle mode.

# PIC18F6310/6410/8310/8410

## REGISTER 17-5: SSPCON2: MSSP CONTROL REGISTER 2 (I<sup>2</sup>C™ 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 <sup>(1)</sup>	ACKEN <sup>(2)</sup>	RCEN <sup>(2)</sup>	PEN <sup>(2)</sup>	RSEN <sup>(2)</sup>	SEN <sup>(2)</sup>

bit 7

bit 0

- bit 7 **GCEN:** General Call Enable bit (Slave mode only)  
 1 = Enable 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)<sup>(1)</sup>  
 1 = Not Acknowledge  
 0 = Acknowledge
- bit 4 **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)<sup>(2)</sup>  
 1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit; automatically cleared by hardware  
 0 = Acknowledge sequence Idle
- bit 3 **RCEN:** Receive Enable bit (Master mode only)<sup>(2)</sup>  
 1 = Enables Receive mode for I<sup>2</sup>C  
 0 = Receive Idle
- bit 2 **PEN:** Stop Condition Enable bit (Master mode only)<sup>(2)</sup>  
 1 = Initiate 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)<sup>(2)</sup>  
 1 = Initiate 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<sup>(2)</sup>  
In Master mode:  
 1 = Initiate 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:** If the I<sup>2</sup>C module is not in Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

# PIC18F6310/6410/8310/8410

## 17.4.2 OPERATION

The MSSP module functions are enabled by setting MSSP Enable bit, SSPEN (SSPCON<5>).

The SSPCON1 register allows control of the I<sup>2</sup>C operation. Four mode selection bits (SSPCON<3:0>) allow one of the following I<sup>2</sup>C modes to be selected:

- I<sup>2</sup>C Master mode, Clock = (Fosc/4) x (SSPADD + 1)
- I<sup>2</sup>C Slave mode (7-bit address)
- I<sup>2</sup>C Slave mode (10-bit address)
- I<sup>2</sup>C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I<sup>2</sup>C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I<sup>2</sup>C Firmware Controlled Master mode, slave is Idle

Selection of any I<sup>2</sup>C 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.

## 17.4.3 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<sup>2</sup>C 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<sup>2</sup>C specification, as well as the requirement of the MSSP module, are shown in timing Parameter #100 and Parameter #101.

## 17.4.3.1 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/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 (SSPIF, BF and UA bits 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.

## 17.4.3.2 Reception

When the  $\overline{R/W}$  bit of the address byte is clear and an address match occurs, the  $\overline{R/W}$  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 17.4.4 “Clock Stretching”](#) for more details.

## 17.4.3.3 Transmission

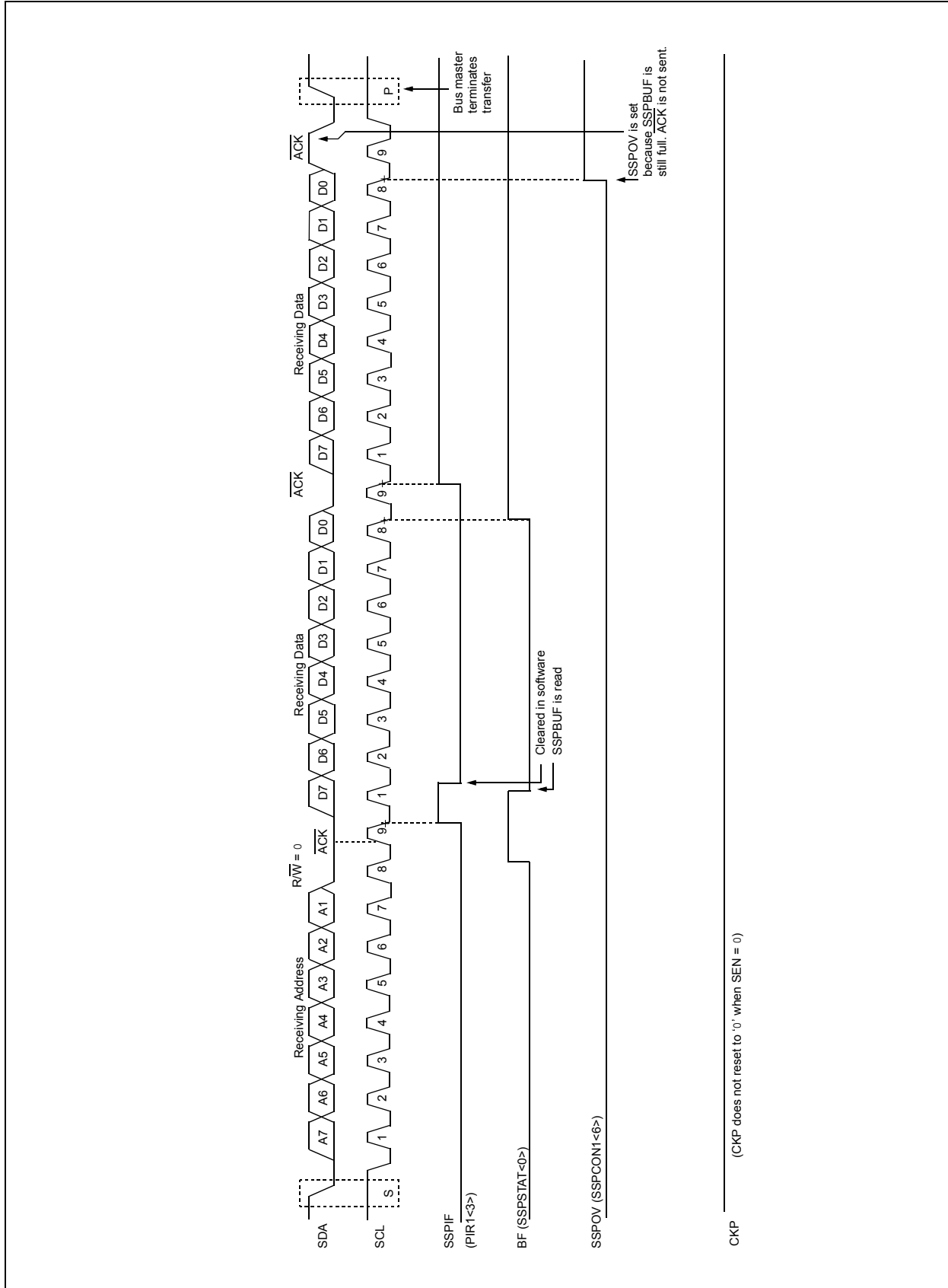
When the  $\overline{R/W}$  bit of the incoming address byte is set and an address match occurs, the  $\overline{R/W}$  bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The  $\overline{ACK}$  pulse will be sent on the ninth bit and pin, RC3/SCK/SCL, is held low regardless of SEN (see [Section 17.4.4 “Clock Stretching”](#) for more detail). 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 8 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 17-9](#)).

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 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.

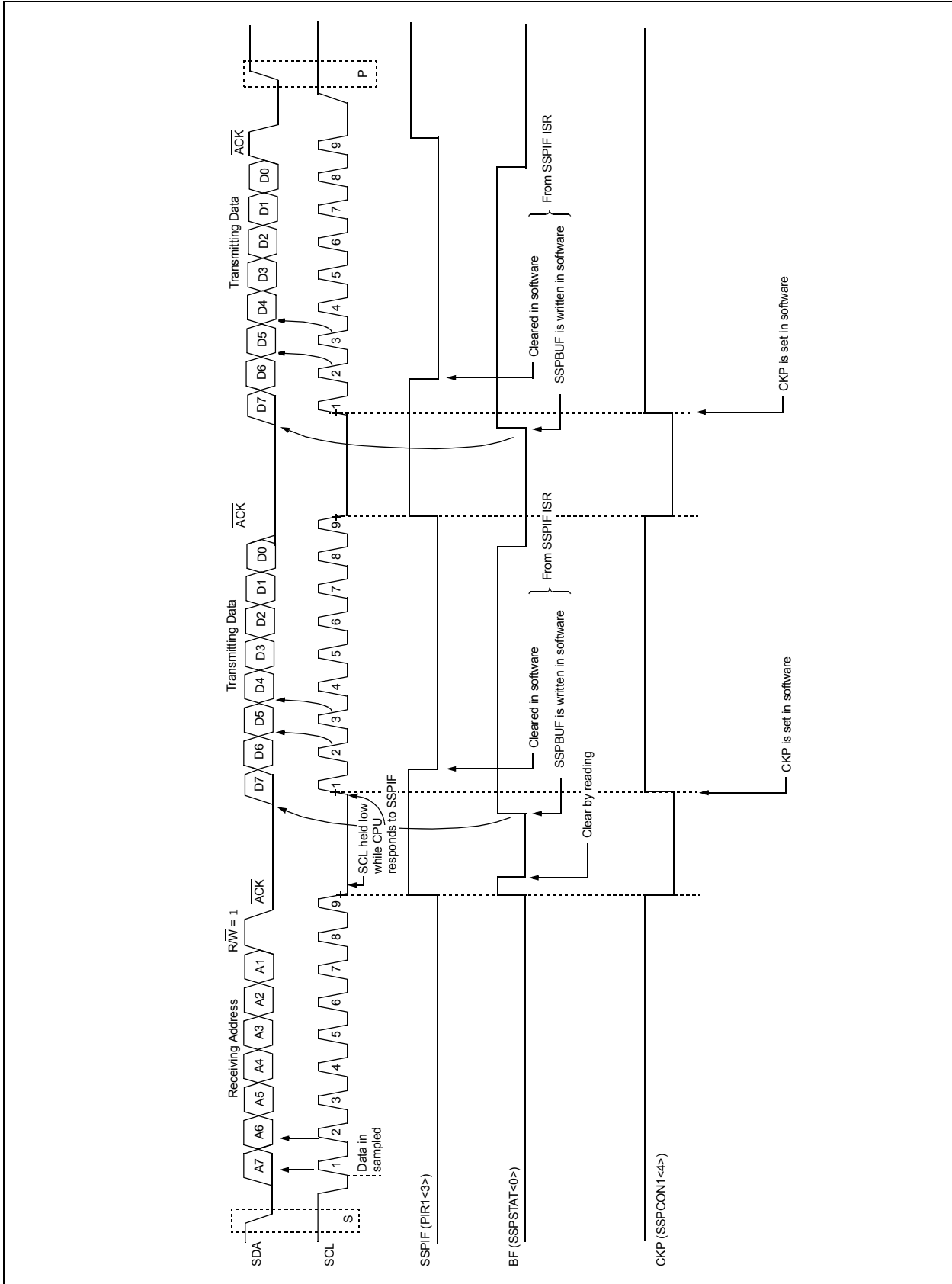
# PIC18F6310/6410/8310/8410

FIGURE 17-8: I<sup>2</sup>C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)



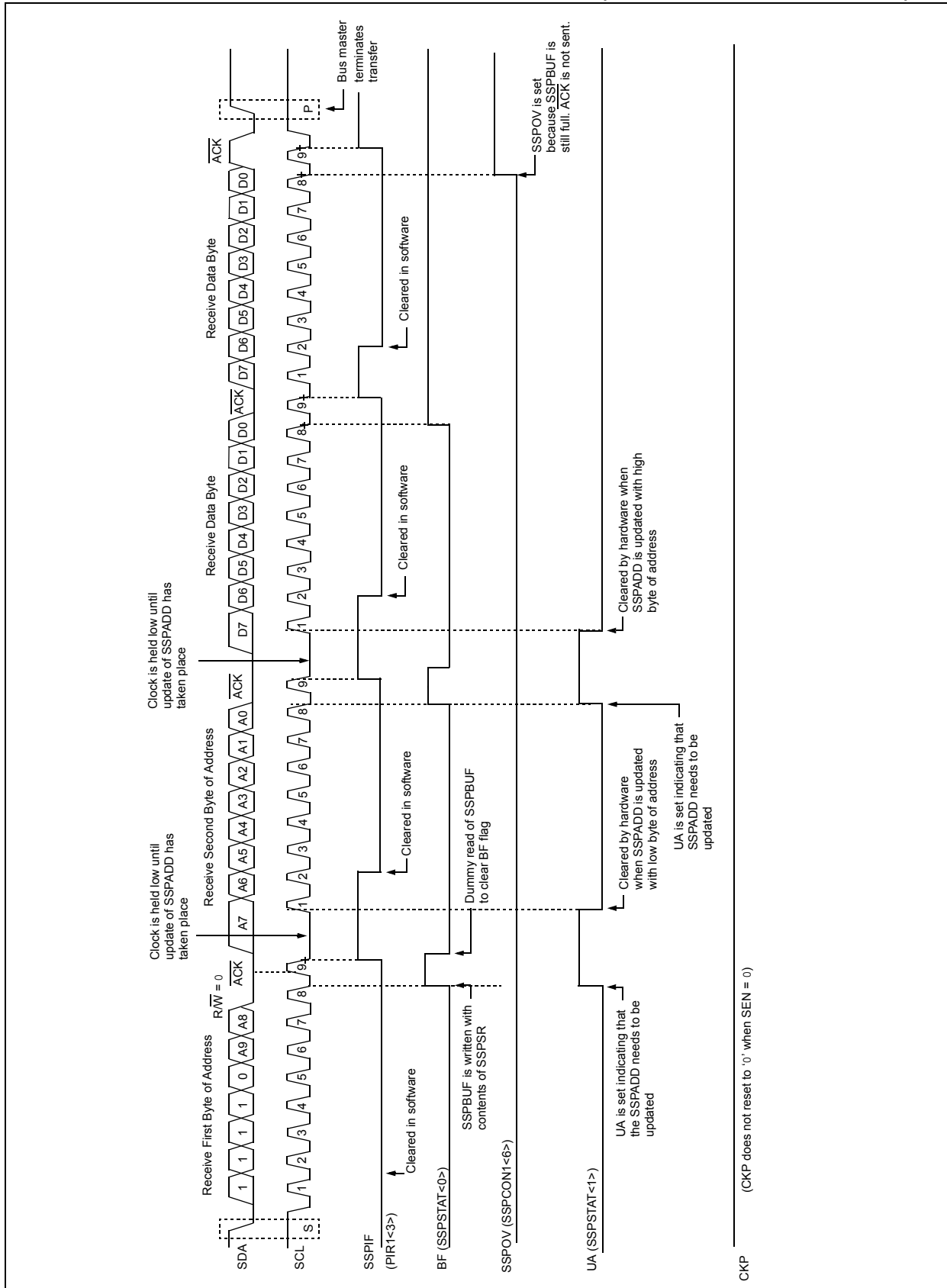


**FIGURE 17-9: I<sup>2</sup>C™ SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)**

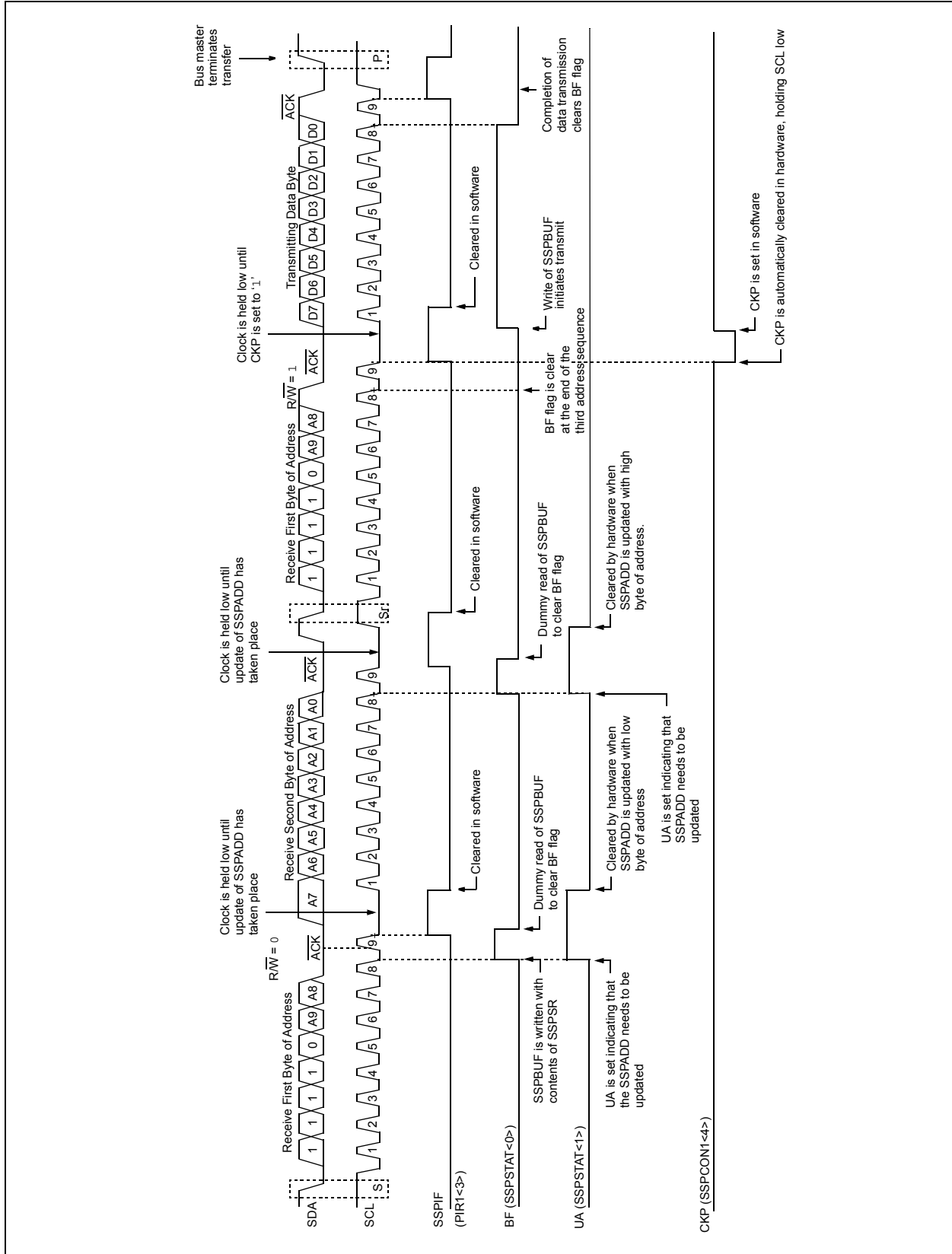


# PIC18F6310/6410/8310/8410

FIGURE 17-10: I<sup>2</sup>C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)



**FIGURE 17-11: I<sup>2</sup>C™ SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)**



# PIC18F6310/6410/8310/8410

## 17.4.4 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.

### 17.4.4.1 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 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 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 17-13](#)).

**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.

### 17.4.4.2 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/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.

### 17.4.4.3 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 17-9](#)).

**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.

### 17.4.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

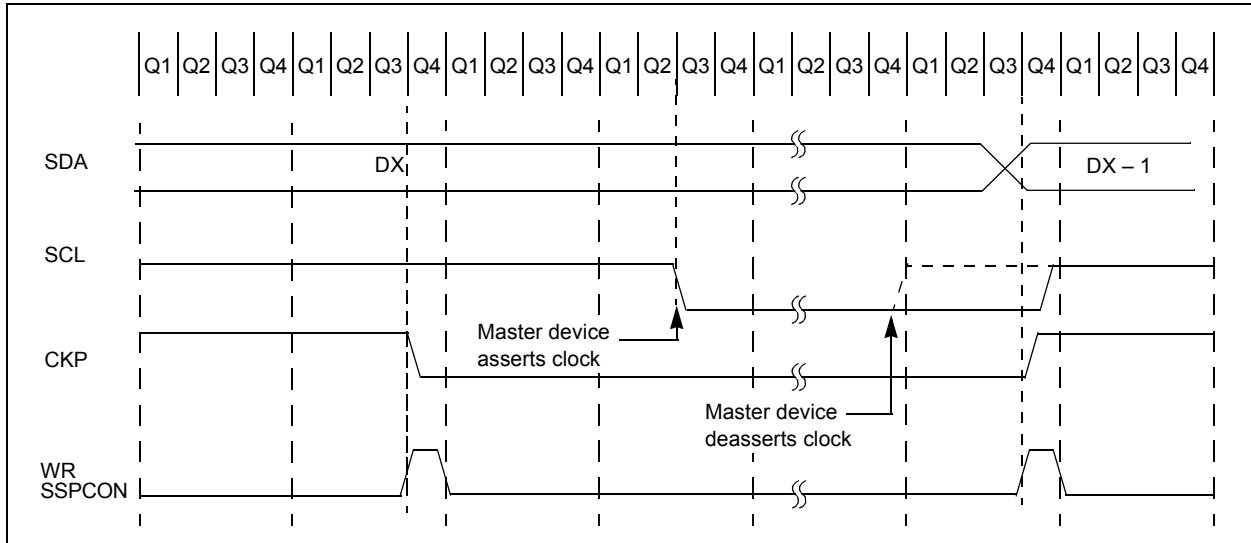
In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence which contains the high-order bits of the 10-bit address and the R/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 17-11](#)).

## 17.4.4.5 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<sup>2</sup>C master device has

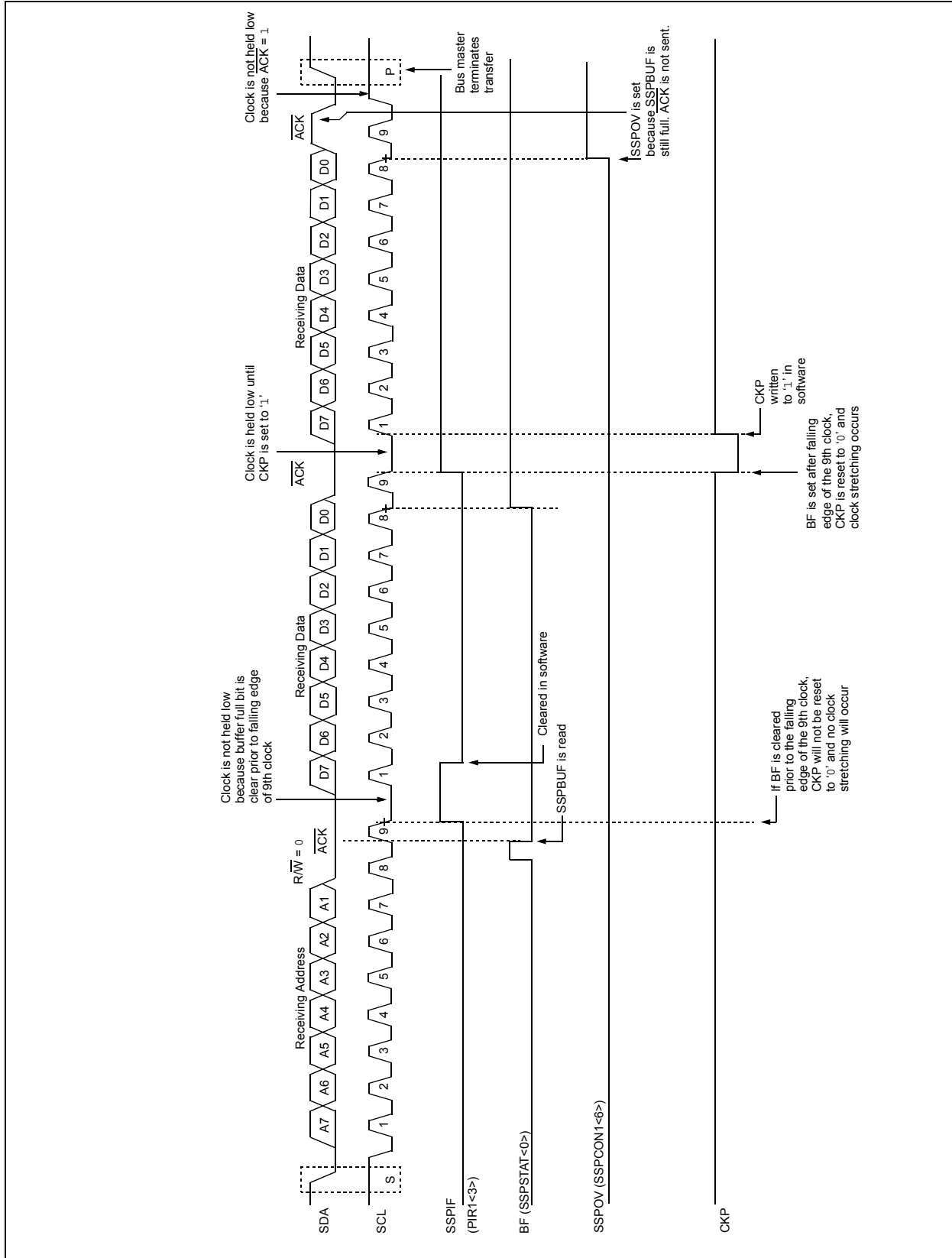
already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I<sup>2</sup>C bus have deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see [Figure 17-12](#)).

**FIGURE 17-12: CLOCK SYNCHRONIZATION TIMING**

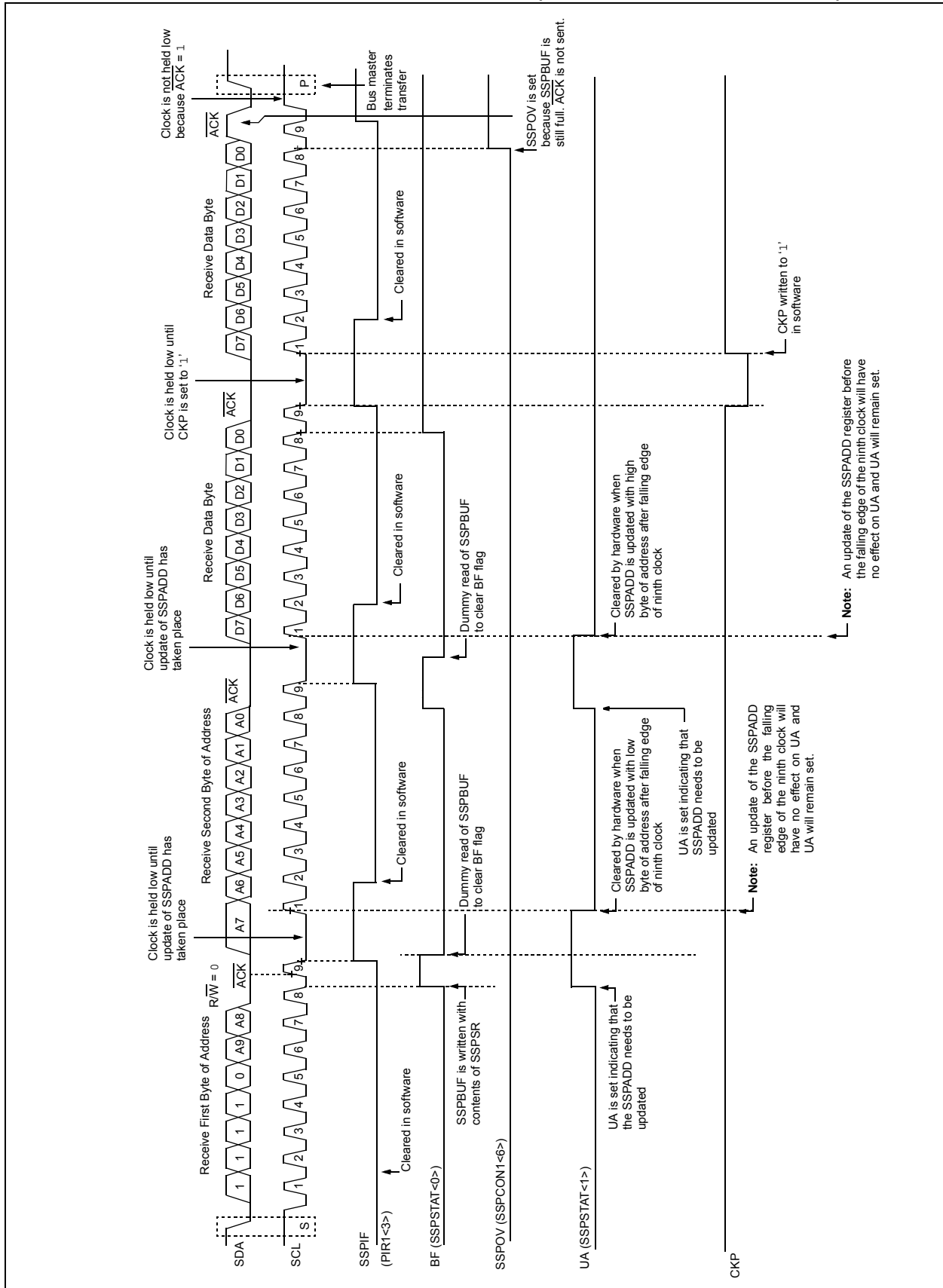


# PIC18F6310/6410/8310/8410

FIGURE 17-13: I<sup>2</sup>C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)



**FIGURE 17-14: I<sup>2</sup>C™ SLAVE MODE TIMING SEN = 1 (RECEPTION, 10-BIT ADDRESS)**



# PIC18F6310/6410/8310/8410

## 17.4.5 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I<sup>2</sup>C 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<sup>2</sup>C protocol. It consists of all '0's with R/W = 0.

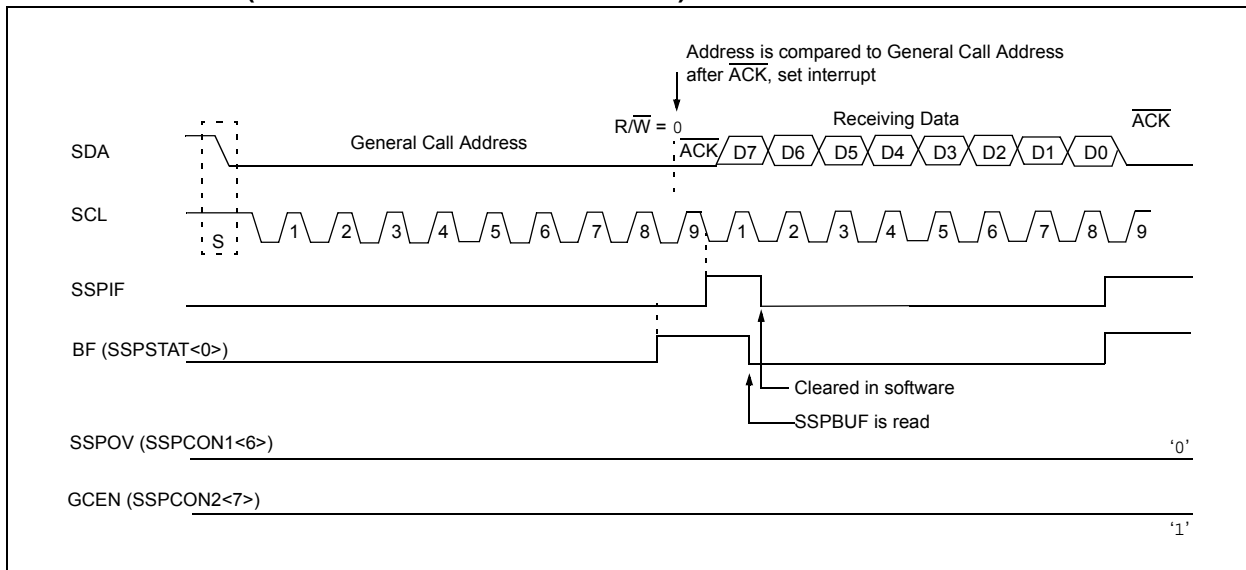
The general call address is recognized when the General Call Enable bit (GCEN) is enabled (SSPCON2<7> set). Following a Start bit detect, 8 bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

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 17-15).

**FIGURE 17-15: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESSING MODE)**





# PIC18F6310/6410/8310/8410

## 17.4.6 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<sup>2</sup>C 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<sup>2</sup>C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

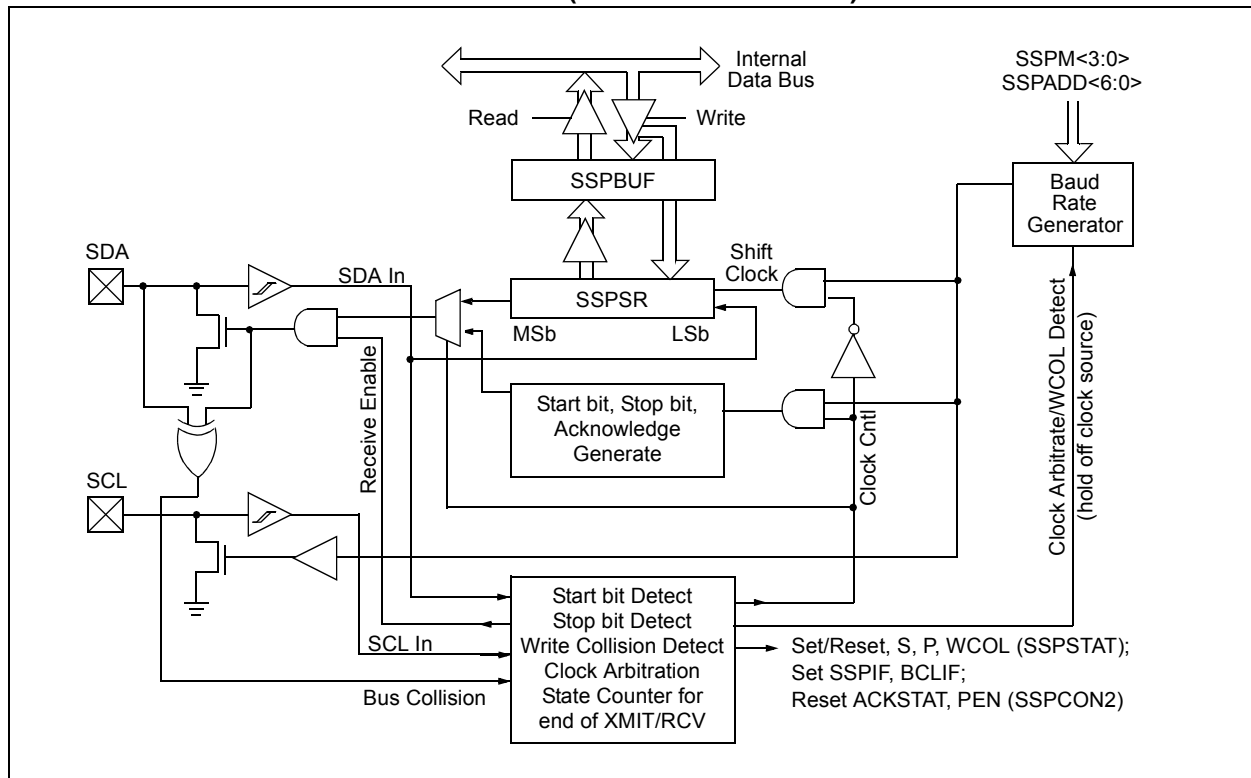
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the I<sup>2</sup>C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDA and SCL.

**Note:** The MSSP module, when configured in I<sup>2</sup>C 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 MSSP Interrupt Flag bit, SSPIF, to be set (MSSP interrupt, if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmit
- Repeated Start

**FIGURE 17-16: MSSP BLOCK DIAGRAM (I<sup>2</sup>C™ MASTER MODE)**



# PIC18F6310/6410/8310/8410

---

## 17.4.6.1 I<sup>2</sup>C 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 I<sup>2</sup>C 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/W) bit. In this case, the R/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/W bit. In this case, the R/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 I<sup>2</sup>C operation. See [Section 17.4.7 "Baud Rate"](#) for more detail.

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 8 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.

# PIC18F6310/6410/8310/8410

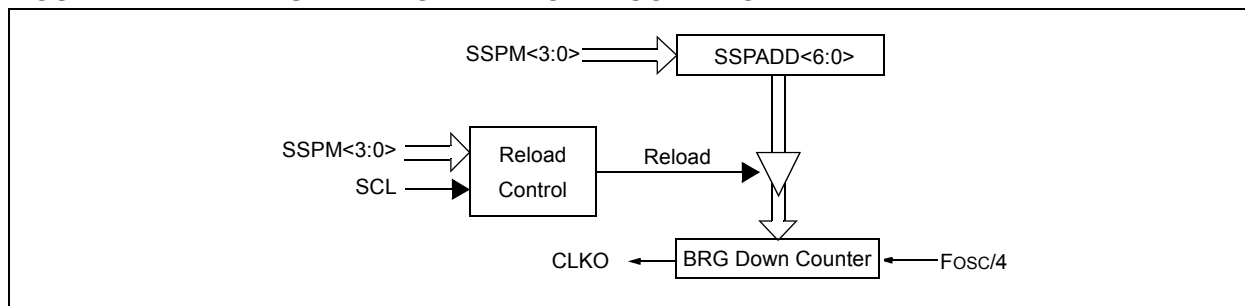
## 17.4.7 BAUD RATE

In I<sup>2</sup>C Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPADD register (Figure 17-17). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to '0' 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<sup>2</sup>C 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 17-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD. Table 17-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD. The SSPADD BRG value of '0x00' is not supported.

**FIGURE 17-17: BAUD RATE GENERATOR BLOCK DIAGRAM**



**TABLE 17-3: I<sup>2</sup>C™ 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	3Fh	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

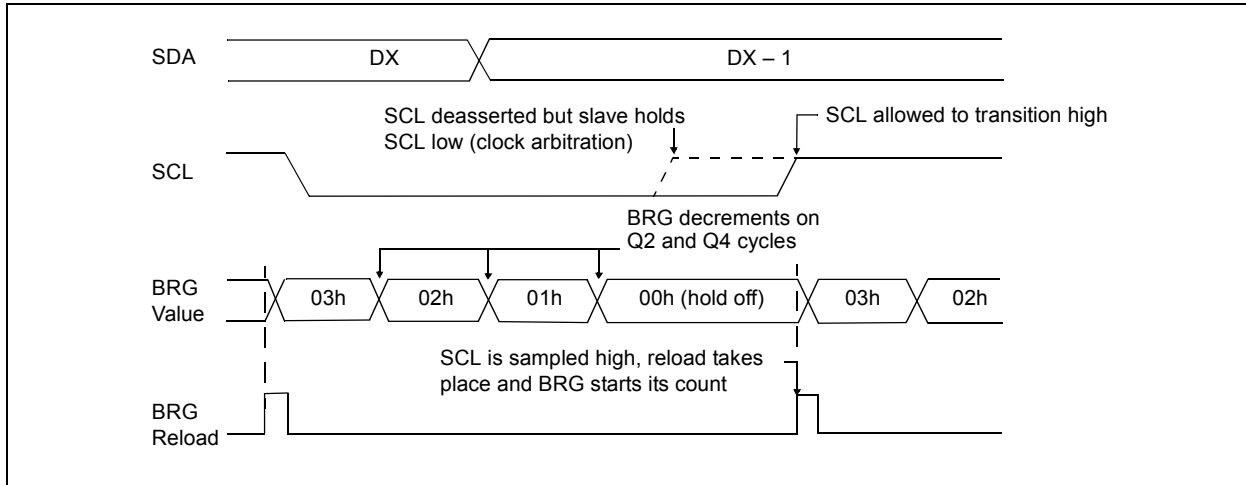
# PIC18F6310/6410/8310/8410

## 17.4.7.1 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 17-18).

**FIGURE 17-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**



## 17.4.8 I<sup>2</sup>C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start 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.

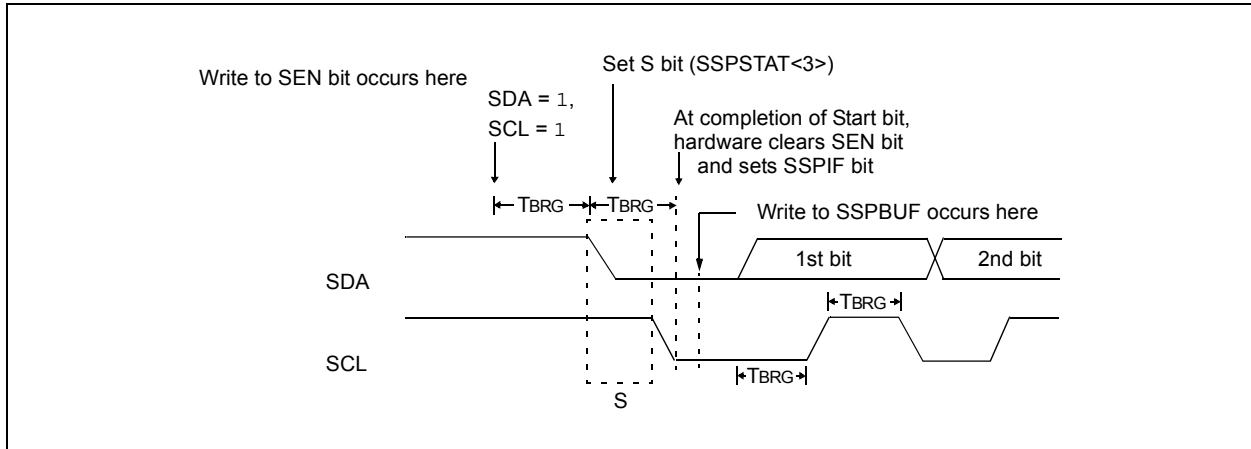
**Note:** If, at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted and the I<sup>2</sup>C module is reset into its Idle state.

### 17.4.8.1 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 17-19: FIRST START BIT TIMING**



# PIC18F6310/6410/8310/8410

## 17.4.9 I<sup>2</sup>C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) is programmed high and the I<sup>2</sup>C 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, 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'.

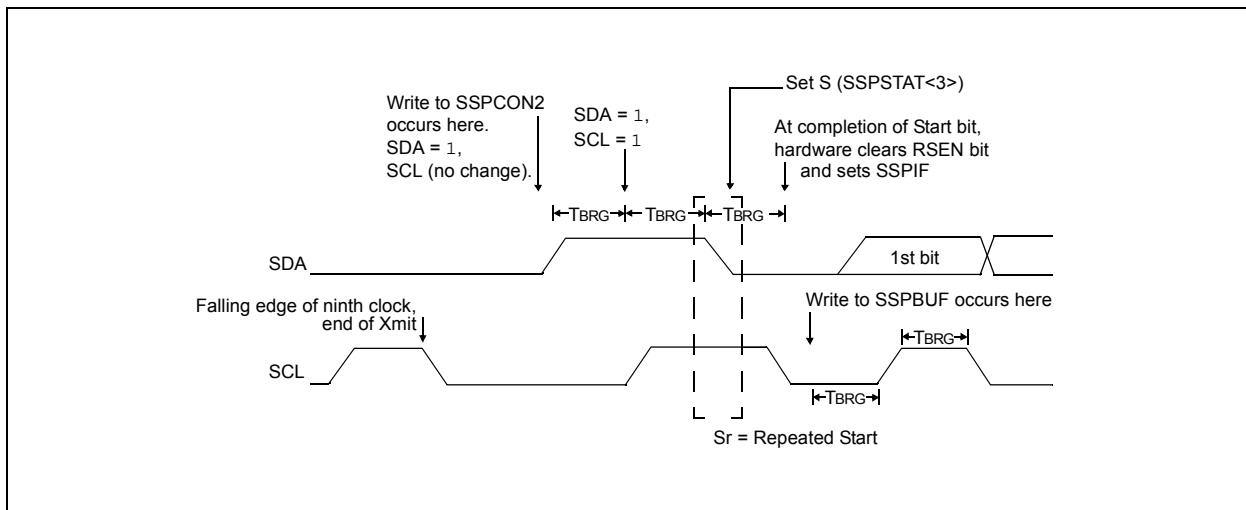
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first 8 bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or 8 bits of data (7-bit mode).

### 17.4.9.1 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 queuing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated Start condition is complete.

**FIGURE 17-20: REPEATED START CONDITION WAVEFORM**



## 17.4.10 I<sup>2</sup>C 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  $\overline{\text{ACK}}$  bit during the ninth bit time if an address match occurred, or if data was received properly. The status of  $\overline{\text{ACK}}$  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 17-21).

After the write to the SSPBUF, each bit of address will be shifted out on the falling edge of SCL until all 7 address bits and the 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  $\overline{\text{ACK}}$  bit is loaded into the ACKSTAT status bit (SSPCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPIF 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.

### 17.4.10.1 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.

### 17.4.10.2 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) after 2 T<sub>cy</sub> after the SSPBUF write. If SSPBUF is rewritten within 2 T<sub>cy</sub>, the WCOL bit is set and SSPBUF is updated. This may result in a corrupted transfer. The user should verify that the WCOL flag is clear after each write to SSPBUF to ensure the transfer is correct.

### 17.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPCON2<6>) is cleared when the slave has sent an Acknowledge ( $\overline{\text{ACK}} = 0$ ) and is set when the slave does not Acknowledge ( $\overline{\text{ACK}} = 1$ ). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

## 17.4.11 I<sup>2</sup>C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPCON2<3>).

<b>Note:</b> 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>).

### 17.4.11.1 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.

### 17.4.11.2 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.

### 17.4.11.3 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).

# PIC18F6310/6410/8310/8410

FIGURE 17-21: I<sup>2</sup>C™ MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)

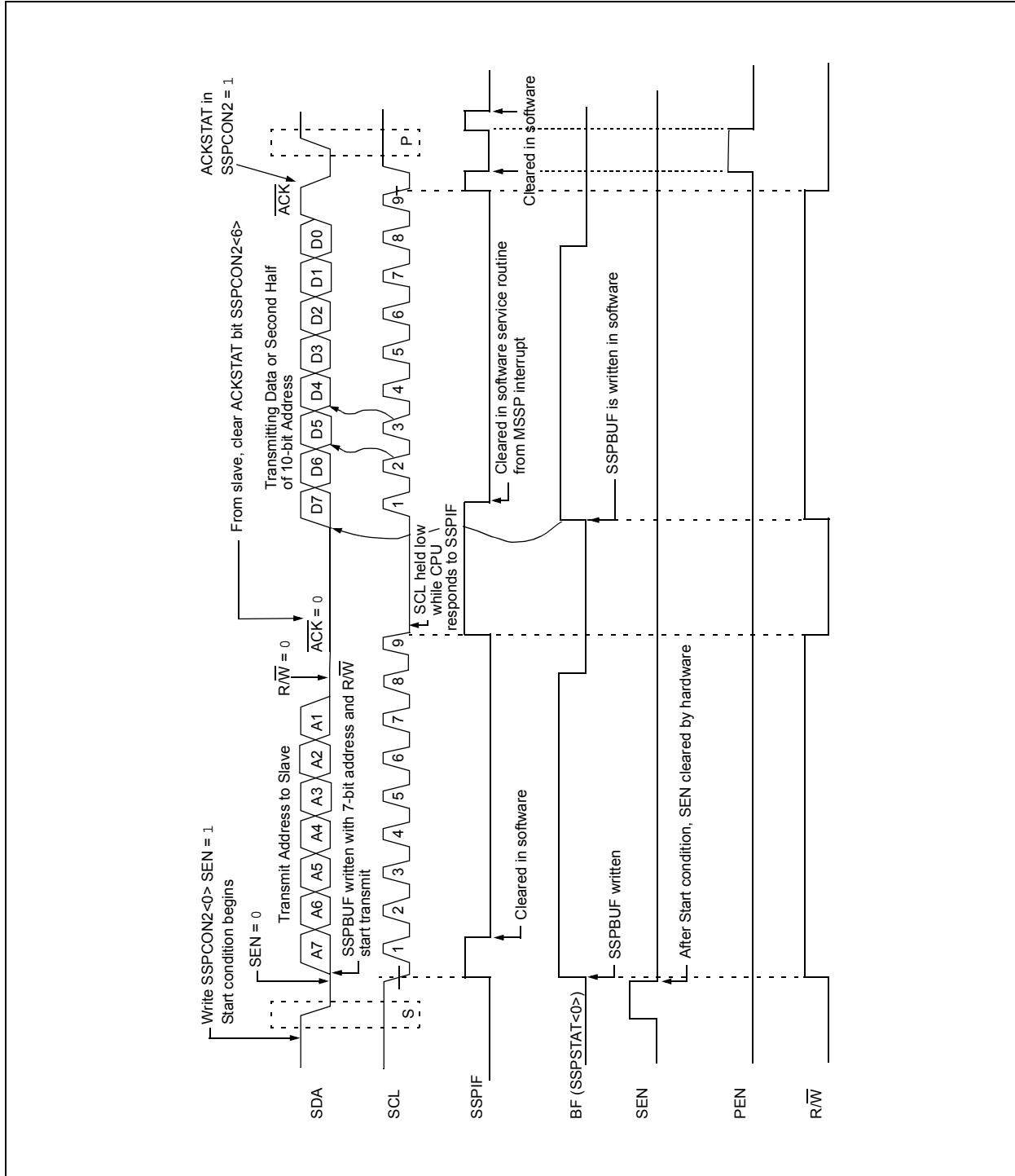
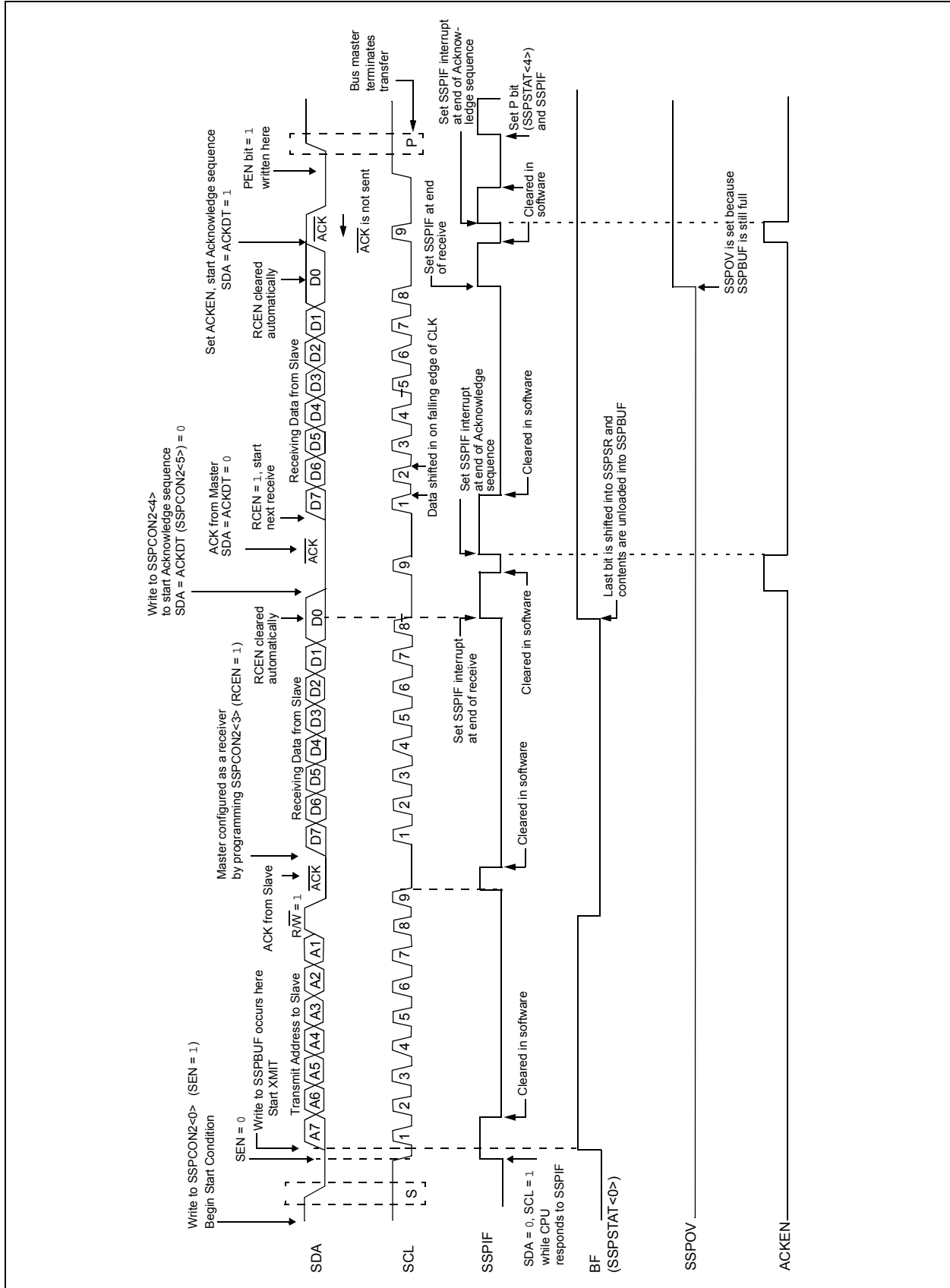




FIGURE 17-22: I<sup>2</sup>C™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



# PIC18F6310/6410/8310/8410

## 17.4.12 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 17-23).

### 17.4.12.1 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).

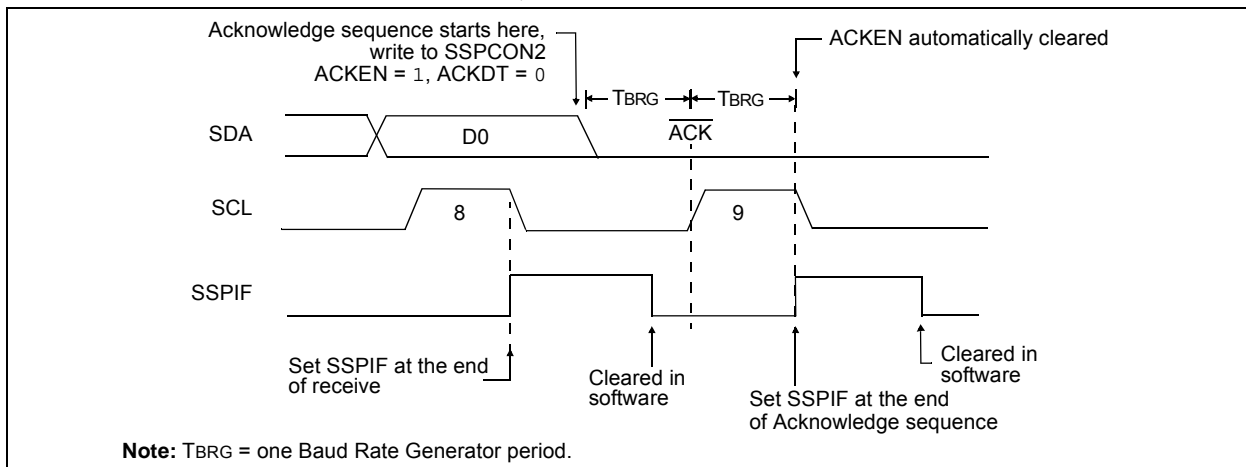
## 17.4.13 STOP CONDITION TIMING

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 17-24).

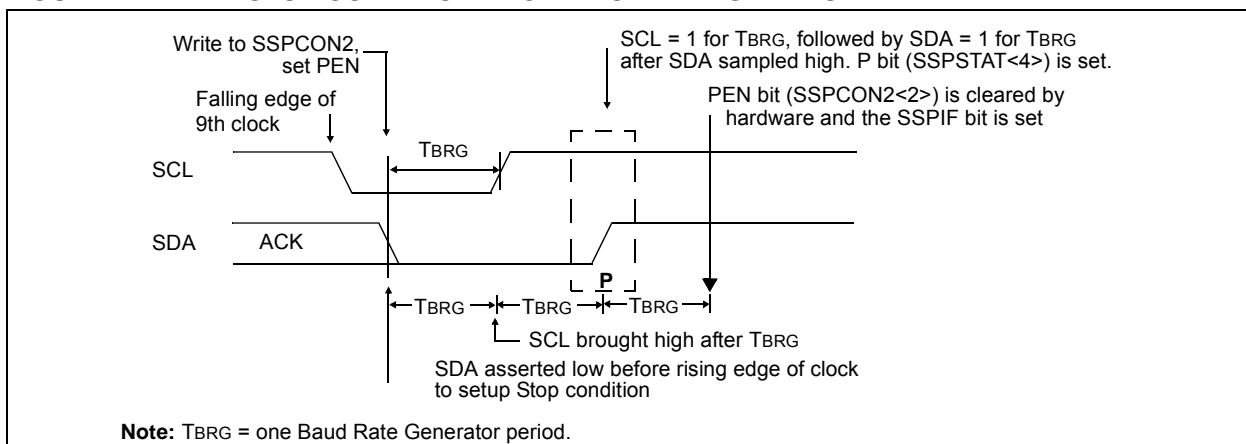
### 17.4.13.1 WCOL Status Flag

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 17-23: ACKNOWLEDGE SEQUENCE WAVEFORM**



**FIGURE 17-24: STOP CONDITION RECEIVE OR TRANSMIT MODE**



## 17.4.14 SLEEP OPERATION

While in Sleep mode, the I<sup>2</sup>C 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).

## 17.4.15 EFFECT OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

## 17.4.16 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<sup>2</sup>C 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

## 17.4.17 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<sup>2</sup>C port to its Idle state (Figure 17-25).

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<sup>2</sup>C bus is free, the user can resume communication by asserting a Start condition.

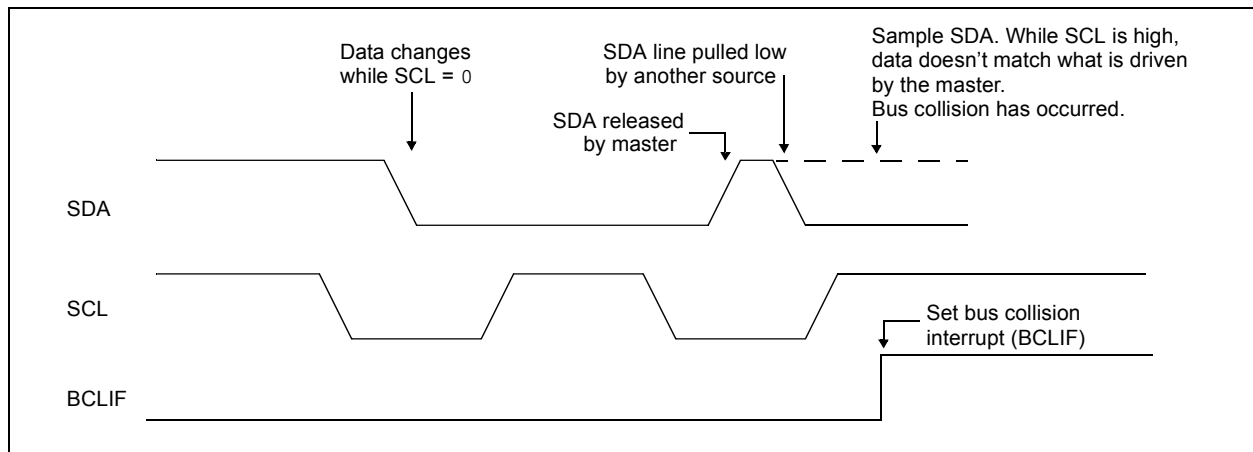
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I<sup>2</sup>C 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<sup>2</sup>C 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 17-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**



# PIC18F6310/6410/8310/8410

## 17.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

- SDA or SCL are sampled low at the beginning of the Start condition (Figure 17-26).
- SCL is sampled low before SDA is asserted low (Figure 17-27).

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:

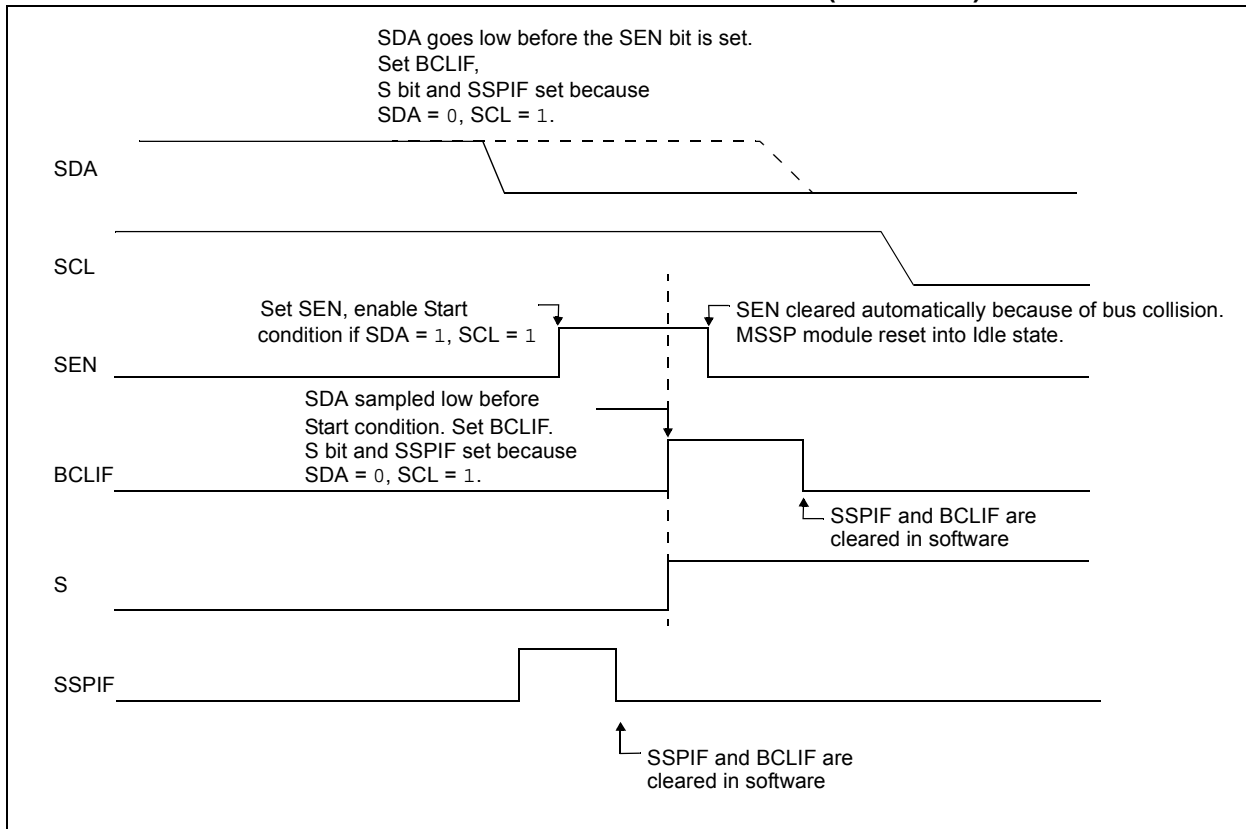
- the Start condition is aborted,
- the BCLIF flag is set and
- the MSSP module is reset to its Idle state (Figure 17-26).

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 17-28). 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.

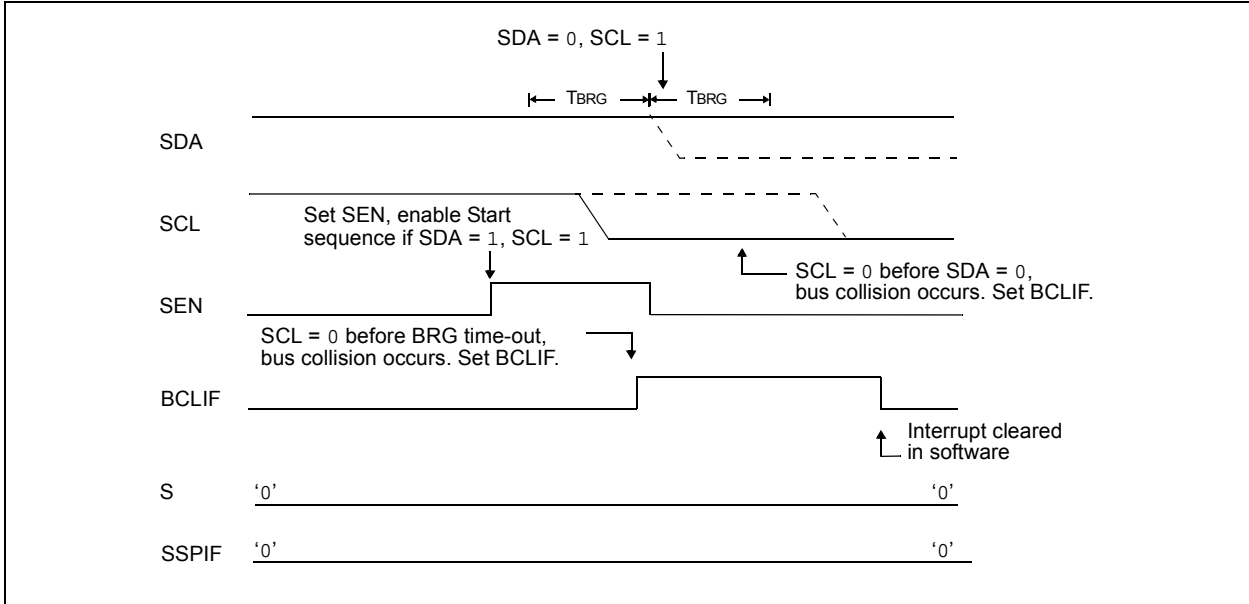
**Note:** The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be 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 17-26: BUS COLLISION DURING START CONDITION (SDA ONLY)**

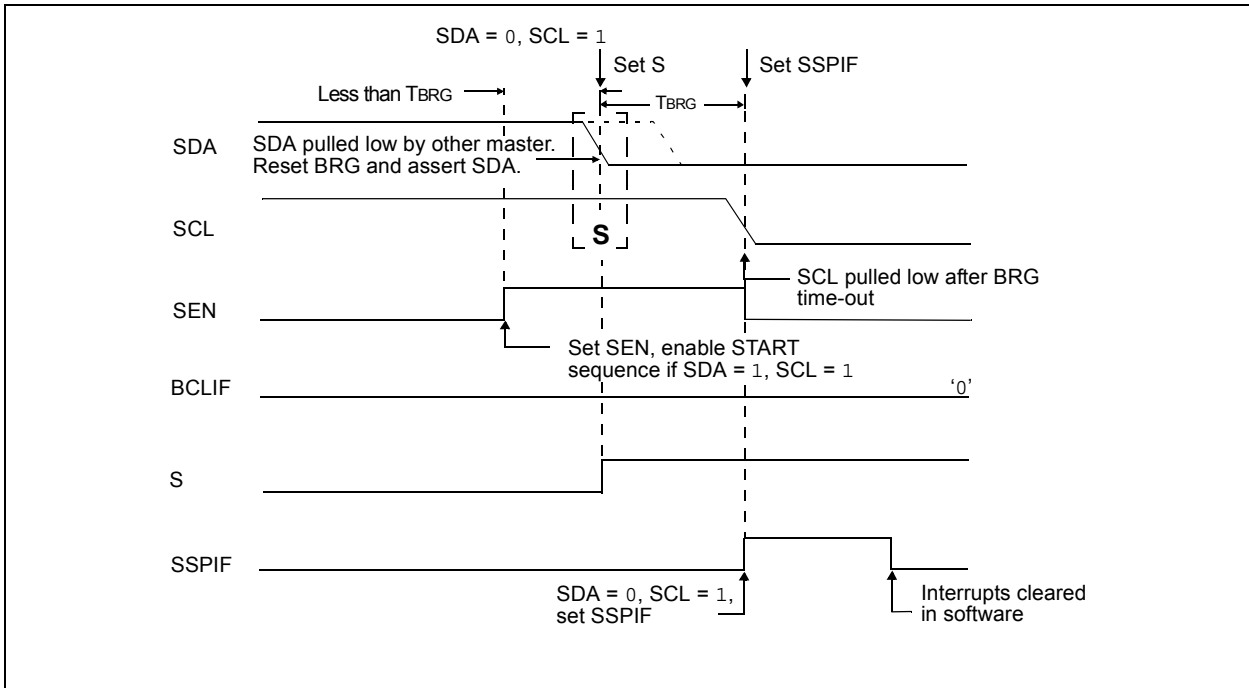


# PIC18F6310/6410/8310/8410

**FIGURE 17-27: BUS COLLISION DURING A START CONDITION (SCL = 0)**



**FIGURE 17-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION**



# PIC18F6310/6410/8310/8410

## 17.4.17.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

- A low level is sampled on SDA when SCL goes from low level to high level.
- SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

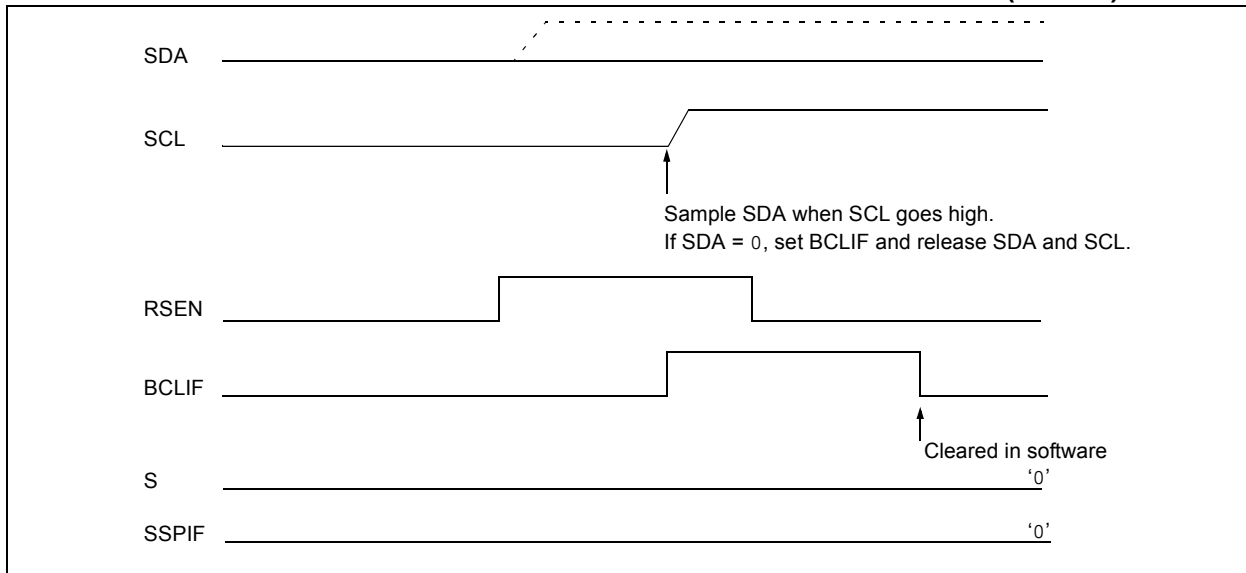
When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to '0'. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', [Figure 17-29](#)). 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.

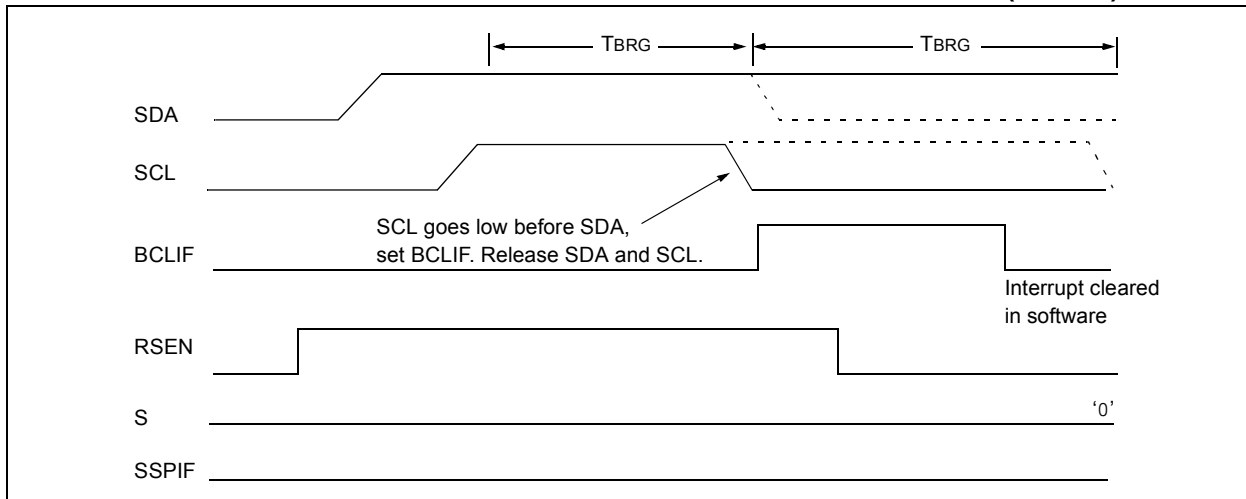
If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition (see [Figure 17-30](#)).

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 17-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**



**FIGURE 17-30: BUS COLLISION DURING A REPEATED START CONDITION (CASE 2)**



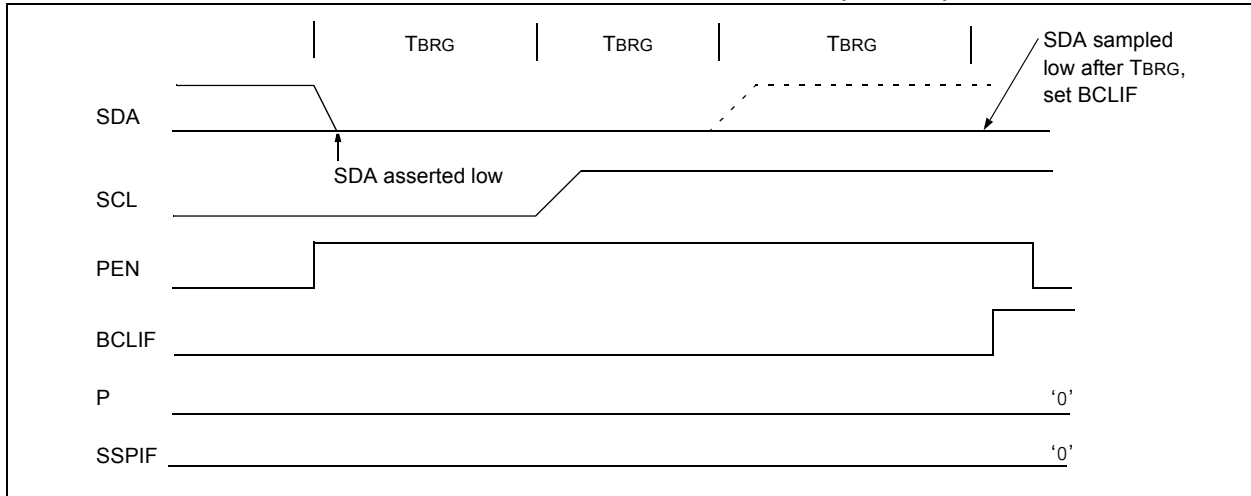
## 17.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

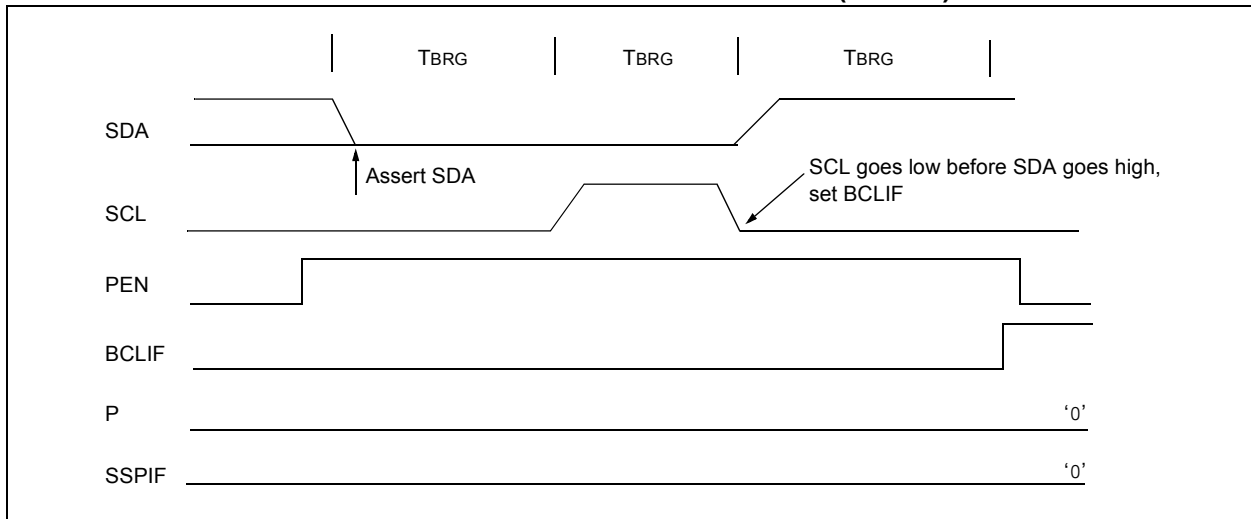
- After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.
- After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

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 17-31). 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 17-32).

**FIGURE 17-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)**



**FIGURE 17-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)**



# PIC18F6310/6410/8310/8410

**TABLE 17-4: REGISTERS ASSOCIATED WITH I<sup>2</sup>C™ 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
TRISC	PORTC Data Direction Register								66
SSPBUF	Master Synchronous Serial Port Receive Buffer/Transmit Register								64
SSPADD	Master Synchronous Serial Port Receive Buffer/Transmit Register								64
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	64
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	64
SSPSTAT	SMP	CKE	D/ $\bar{A}$	P	S	R/ $\bar{W}$	UA	BF	64

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the MSSP in I<sup>2</sup>C mode.



## 18.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

PIC18F6310/6410/8310/8410 devices have three serial I/O modules: the MSSP module, discussed in the previous chapter and two Universal Synchronous Asynchronous Receiver Transmitter (USART) modules. (Generically, the 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, etc.

There are two distinct implementations of the USART module in these devices: the Enhanced USART (EUSART), discussed here and the Addressable USART (AUSART), discussed in the next chapter. For this device family, USART1 always refers to the EUSART, while USART2 is always the AUSART.

The EUSART and AUSART modules implement the same core features for serial communications; their basic operation is essentially the same. The EUSART module provides additional features, including automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These features make it ideally suited for use in Local Interconnect Network 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 TX1/CK1 and RX1/DT1 as a USART:

- SPEN bit (RCSTA1<7>) must be set (= 1)
- TRISC<7> bit must be set (= 1)
- TRISC<6> bit must be set (= 1)

**Note:** The USART 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 Register 1 (TXSTA1)
- Receive Status and Control Register 1 (RCSTA1)
- Baud Rate Control Register 1 (BAUDCON1)

The registers are described in [Register 18-1](#), [Register 18-2](#) and [Register 18-3](#).

# PIC18F6310/6410/8310/8410

## REGISTER 18-1: TXSTA1: EUSART1 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 <sup>(1)</sup>	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<sup>(1)</sup>  
1 = Transmit is enabled  
0 = Transmit is disabled
- bit 4      **SYNC:** AUSART 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 is empty  
0 = TSR is 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.

# PIC18F6310/6410/8310/8410

**REGISTER 18-2: RCSTA1: EUSART1 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 is enabled  
           0 = Serial port is disabled
  
- 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> are set  
           0 = Disables address detection, all bytes are received and ninth bit can be used as a parity bit  
           Asynchronous mode 8-Bit (RX9 = 0):  
           Don't care.
  
- bit 2      **FERR:** Framing Error bit  
           1 = Framing error (can be cleared by reading RCREG1 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 bit  
           This can be address/data bit or a parity bit and must be calculated by user firmware.

# PIC18F6310/6410/8310/8410

## REGISTER 18-3: BAUDCON1: BAUD RATE CONTROL REGISTER 1

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	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      **RXDTP**: Received Data Polarity Select bit  
Asynchronous mode:  
 1 = Receive data (RXx) is inverted (active-low)  
 0 = Receive data (RXx) is not inverted (active-high)  
Synchronous mode:  
 No affect.
- bit 4      **TXCKP**: Clock and Data Polarity Select bit  
Asynchronous mode:  
 1 = Idle state for transmit (TXx) is a low level  
 0 = Idle state for transmit (TXx) is a high level  
Synchronous mode:  
 1 = Idle state for clock (CKx) is a high level  
 0 = Idle state for clock (CKx) is a low level
- bit 3      **BRG16**: 16-Bit Baud Rate Register Enable bit  
 1 = 16-bit Baud Rate Generator – SPBRGH1 and SPBRG1  
 0 = 8-bit Baud Rate Generator – SPBRG1 only (Compatible mode); SPBRGH1 value ignored
- bit 2      **Unimplemented**: Read as '0'
- bit 1      **WUE**: Wake-up Enable bit  
Asynchronous mode:  
 1 = EUSART will continue to sample the RXx pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge  
 0 = RXx 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.

## 18.1 EUSART Baud Rate Generator (BRG)

The BRG is a dedicated, 8-bit or 16-bit generator that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode; setting the BRG16 bit (BAUDCON1<3>) selects 16-bit mode.

The SPBRGH1:SPBRG1 register pair controls the period of a free running timer. In Asynchronous mode, bits, BRGH (TXSTA1<2>) and BRG16 (BAUDCON1<3>), also control the baud rate. In Synchronous mode, BRGH is ignored. Table 18-1 shows the formula for computation of the baud rate for different EUSART modes that only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRGH1:SPBRG1 registers can be calculated using the formulas in Table 18-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 18-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 18-2. 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 SPBRGH1:SPBRG1 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:** The BRG value of '0' is not supported.

### 18.1.1 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 SPBRG1 register pair.

### 18.1.2 SAMPLING

The data on the RXx pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RXx pin when SYNC is clear or when both BRG16 and BRGH are not set. The data on the RXx pin is sampled once when SYNC is set or when BRGH16 and BRGH are both set.

**TABLE 18-1: BAUD RATE FORMULAS**

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	$F_{OSC}/[64 (n + 1)]$
0	0	1	8-bit/Asynchronous	$F_{OSC}/[16 (n + 1)]$
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	$F_{OSC}/[4 (n + 1)]$
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

**Legend:** x = Don't care, n = Value of SPBRGH1:SPBRG1 register pair

### EXAMPLE 18-1: 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	= $F_{OSC}/(64 ([SPBRGH1:SPBRG1] + 1))$
Solving for SPBRGH1:SPBRG1:	
X	= $((F_{OSC}/\text{Desired Baud Rate})/64) - 1$
	= $((16000000/9600)/64) - 1$
	= $[25.042] = 25$
Calculated Baud Rate	= $16000000/(64 (25 + 1))$
	= 9615
Error	= $(\text{Calculated Baud Rate} - \text{Desired Baud Rate})/\text{Desired Baud Rate}$
	= $(9615 - 9600)/9600 = 0.16\%$

**TABLE 18-2: REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	65
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	65
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	66
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte								66
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte								65

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the BRG.

# PIC18F6310/6410/8310/8410

**TABLE 18-3: 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	—	—	—	—	—	—

# PIC18F6310/6410/8310/8410

**TABLE 18-3: 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	—	—	—	—	—	—

# PIC18F6310/6410/8310/8410

## 18.1.3 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 18-1) 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 RX1 signal, the RX1 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 Detect 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 SPBRG1 begins counting up, using the preselected clock source on the first rising edge of RX1. After eight bits on the RX1 pin or the fifth rising edge, an accumulated value totalling the proper BRG period is left in the SPBRGH1:SPBRG1 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 (BAUDCON1<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 18-2).

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 SPBRGH1 register. Refer to Table 18-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the EUSART state machine is held in Idle. The RC1IF interrupt is set once the fifth rising edge on RX1 is detected. The value in the RCREG1 needs to be read to clear the RC1IF interrupt. The contents of RCREG1 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 18-4: 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

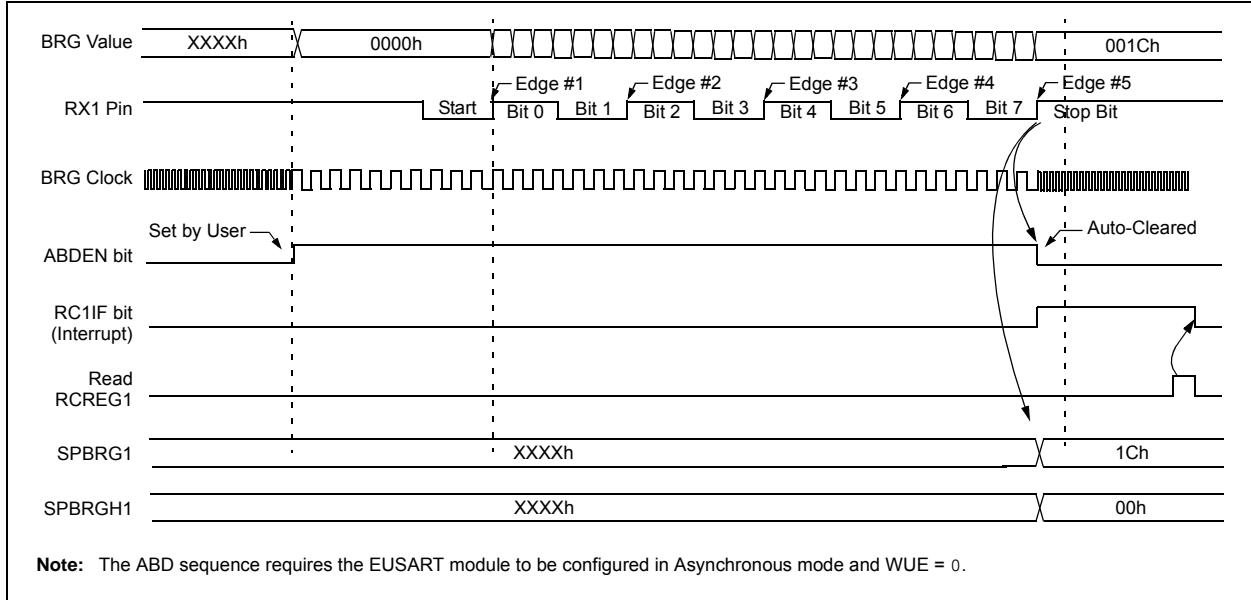
### 18.1.3.1 ABD and EUSART Transmission

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, TXREG1 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.

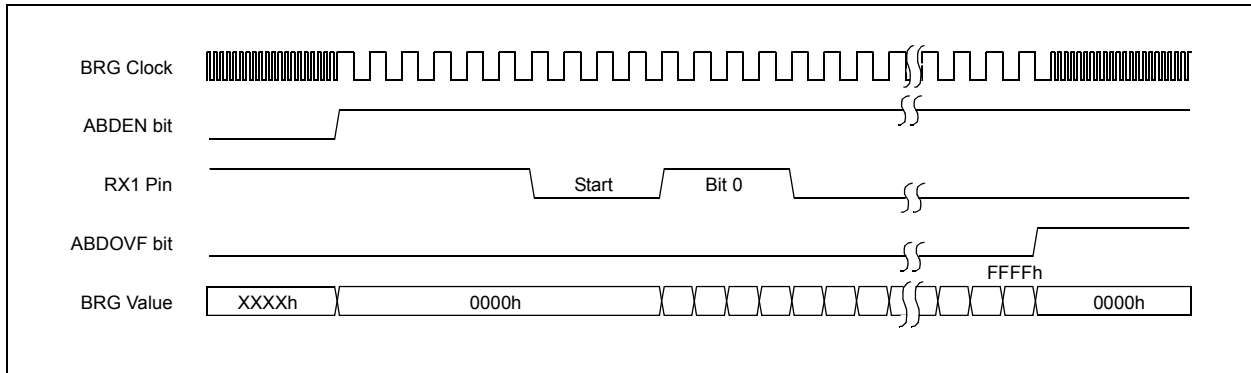


# PIC18F6310/6410/8310/8410

**FIGURE 18-1: AUTOMATIC BAUD RATE CALCULATION**



**FIGURE 18-2: BRG OVERFLOW SEQUENCE**



# PIC18F6310/6410/8310/8410

## 18.2 EUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTA1<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 (TXSTA1<2> and BAUDCON1<3>). Parity is not supported by the hardware but can be implemented in software and stored as the 9th data bit. The TXCKP (BAUDCON<4>) and RXDTP (BAUDCON<5>) bits allow the TX and RX signals to be inverted (polarity reversed). Devices that buffer signals between TTL and RS-232 levels also invert the signal. Setting the TXCKP and RXDTP bits allows for the use of circuits that provide buffering without inverting the signal.

When operating in Asynchronous mode, the EUSART module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver
- Auto-Wake-up on Sync Break Character
- 12-Bit Break Character Transmit
- Auto-Baud Rate Detection

### 18.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in [Figure 18-3](#). 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, TXREG1. The TXREG1 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 TXREG1 register (if available).

Once the TXREG1 register transfers the data to the TSR register (occurs in one Tcy), the TXREG1 register is empty and the TX1IF flag bit (PIR1<4>) is set. This

interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TX1IE (PIE1<4>). TX1IF will be set regardless of the state of TX1IE; it cannot be cleared in software. TX1IF is also not cleared immediately upon loading TXREG1, but becomes valid in the second instruction cycle following the load instruction. Polling TX1IF immediately following a load of TXREG1 will return invalid results.

While TX1IF indicates the status of the TXREG1 register, another bit, TRMT (TXSTA1<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. The TXCKP bit (BAUDCON<4>) allows the TX signal to be inverted (polarity reversed). Devices that buffer signals from TTL to RS-232 levels also invert the signal (when TTL = 1, RS-232 = negative). Inverting the polarity of the TXx pin data by setting the TXCKP bit allows for use of circuits that provide buffering without inverting the signal.

**Note 1:** The TSR register is not mapped in data memory so it is not available to the user.

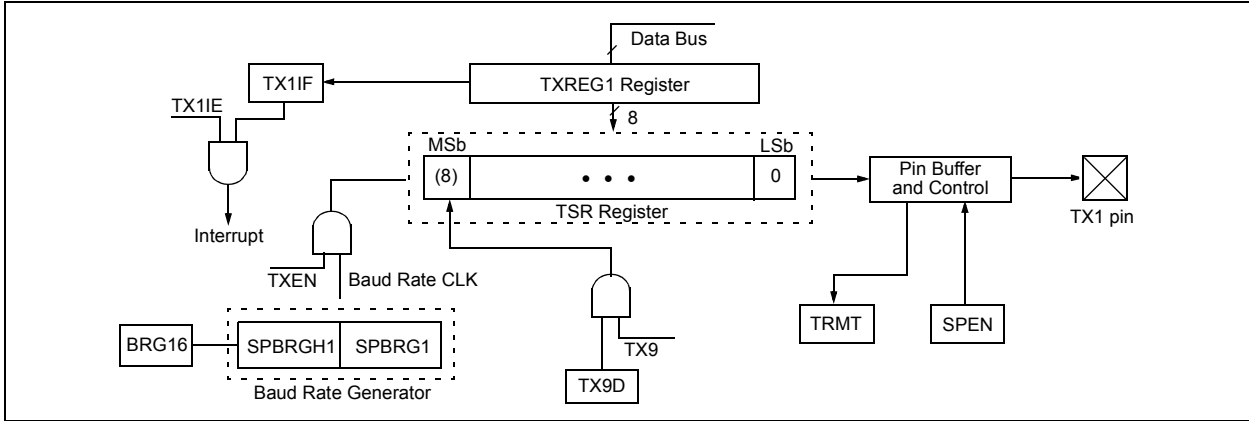
**2:** Flag bit, TX1IF, is set when enable bit, TXEN, is set.

To set up an Asynchronous Transmission:

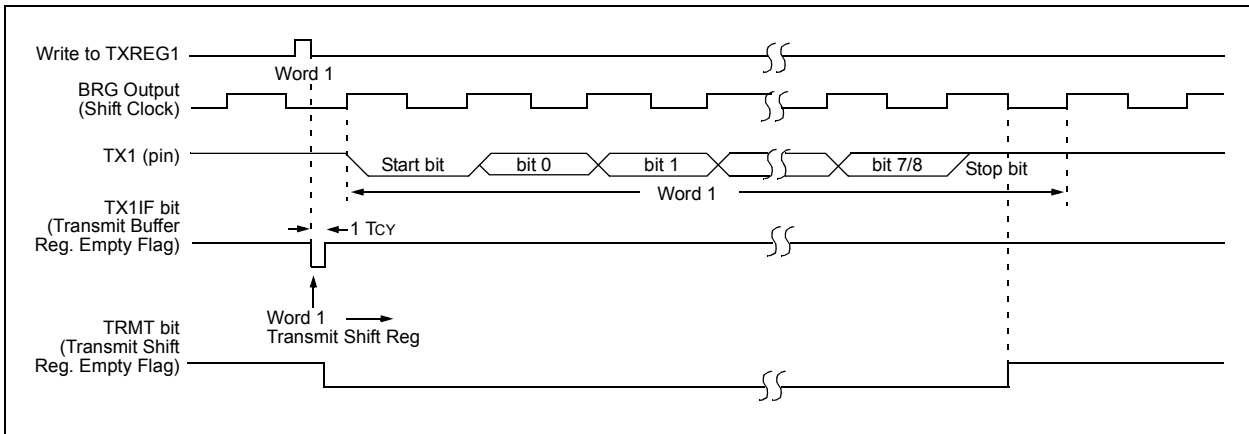
1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If the signal from the TXx pin is to be inverted, set the TXCKP bit.
4. If interrupts are desired, set enable bit, TXIE.
5. If 9-bit transmission is desired, set transmit bit, TX9. Can be used as an address/data bit.
6. Enable the transmission by setting bit, TXEN, which will also set bit, TXIF.
7. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
8. Load data to the TXREG register (starts transmission).
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

# PIC18F6310/6410/8310/8410

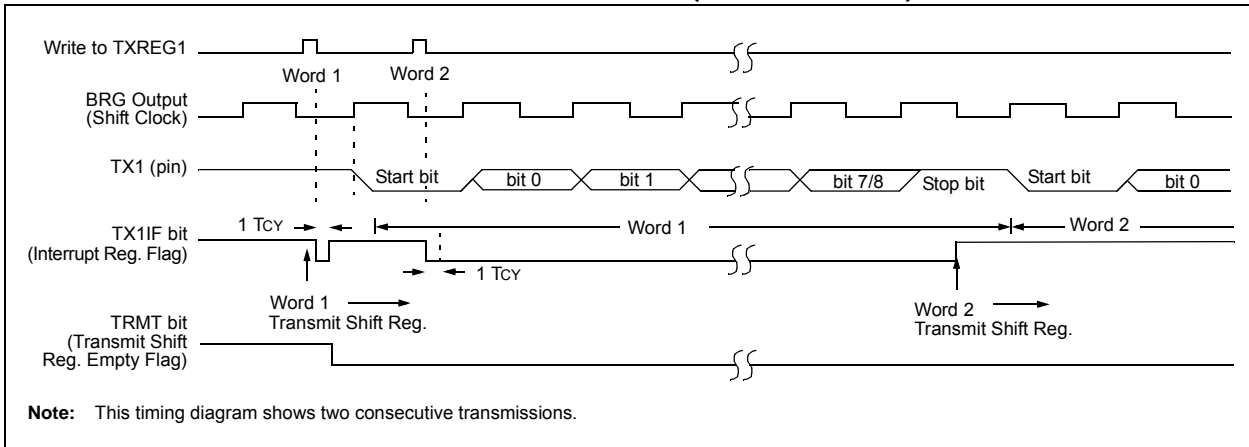
**FIGURE 18-3: EUSART TRANSMIT BLOCK DIAGRAM**



**FIGURE 18-4: ASYNCHRONOUS TRANSMISSION**



**FIGURE 18-5: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)**



# PIC18F6310/6410/8310/8410

**TABLE 18-5: 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	65
TXREG1	EUSART1 Transmit Register								65
TXSTA1	CSRC	TX9	TXEN	SYNC	SEnDB	BRGH	TRMT	TX9D	65
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	66
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte								66
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte								65

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

## 18.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in [Figure 18-6](#). The data is received on the RX1 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.

The RXDTP bit (BAUDCON<5>) allows the RX signal to be inverted (polarity reversed). Devices that buffer signals from RS-232 to TTL levels also perform an inversion of the signal (when RS-232 = positive, TTL = 0). Inverting the polarity of the RXx pin data by setting the RXDTP bit allows for the use of circuits that provide buffering without inverting the signal.

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 the signal at the RXx pin is to be inverted, set the RXDTP bit.
4. If interrupts are desired, set enable bit, RCIE.
5. If 9-bit reception is desired, set bit, RX9.
6. Enable the reception by setting bit, CREN.
7. Flag bit, RCIF, will be set when reception is complete and an interrupt will be generated if 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 enable bit, CREN.
11. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

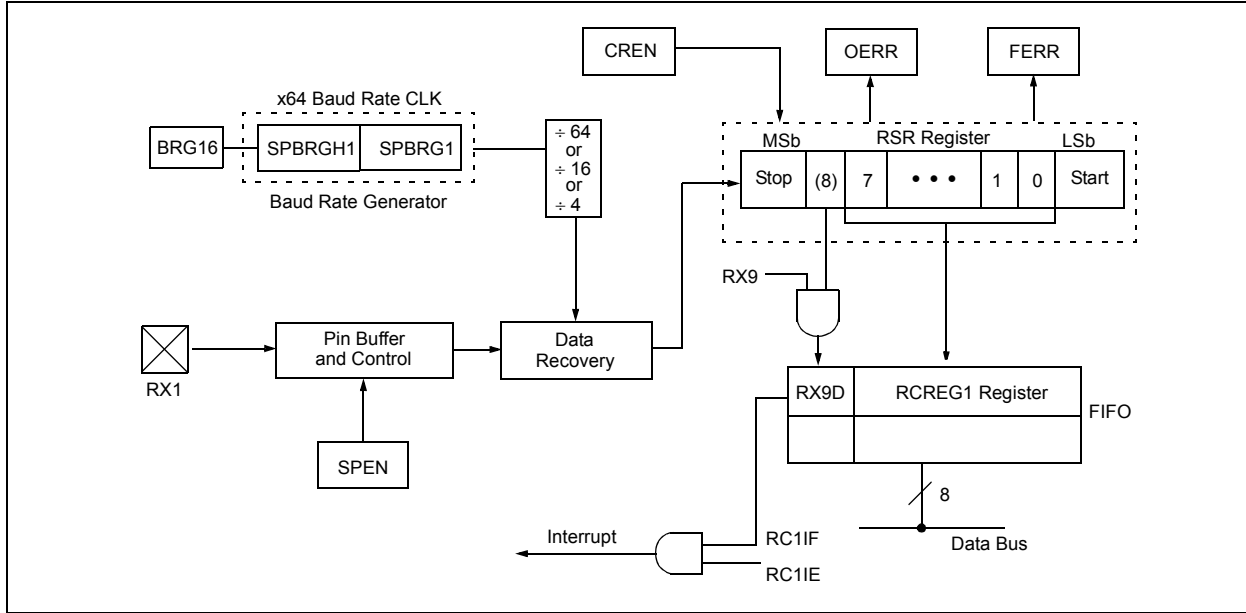
## 18.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

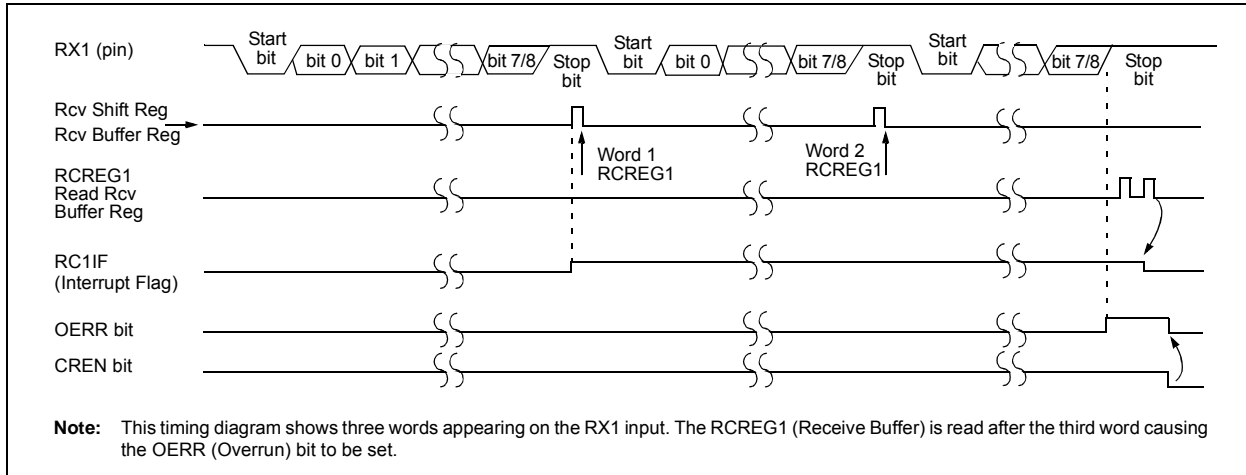
1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If the signal at the RXx pin is to be inverted, set the RXDTP bit. If the signal from the TXx pin is to be inverted, set the TXCKP bit.
4. If interrupts are required, set the RCEN bit and select the desired priority level with the RCIP bit.
5. Set the RX9 bit to enable 9-bit reception.
6. Set the ADDEN bit to enable address detect.
7. Enable reception by setting the CREN bit.
8. The RCIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCIE and GIE bits are set.
9. Read the RCSTA register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
10. Read RCREG to determine if the device is being addressed.
11. If any error occurred, clear the CREN bit.
12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

# PIC18F6310/6410/8310/8410

**FIGURE 18-6: EUSART RECEIVE BLOCK DIAGRAM**



**FIGURE 18-7: ASYNCHRONOUS RECEPTION**



# PIC18F6310/6410/8310/8410

**TABLE 18-6: 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	65
RCREG1	EUSART1 Receive Register								65
TXSTA1	CSRC	TX9	TXEN	SYNC	SEnDB	BRGH	TRMT	TX9D	65
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	66
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte								66
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte								65

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

# PIC18F6310/6410/8310/8410

---

## 18.2.4 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 RX1/DT1 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 RX1/DT1 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 RX1/DT1 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 RC1IF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 18-8) and asynchronously, if the device is in Sleep mode (Figure 18-9). The interrupt condition is cleared by reading the RCREG1 register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RX1 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.

### 18.2.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RX1/DT1, information with any state changes before the Stop bit may signal a false End-of-Character (EOC) and cause data or framing errors. Therefore, to work properly, 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 the 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.

### 18.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RC1IF 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 RC1IF bit. The WUE bit is cleared after this when a rising edge is seen on RX1/DT1. The interrupt condition is then cleared by reading the RCREG1 register. Ordinarily, the data in RCREG1 will be dummy data and should be discarded.

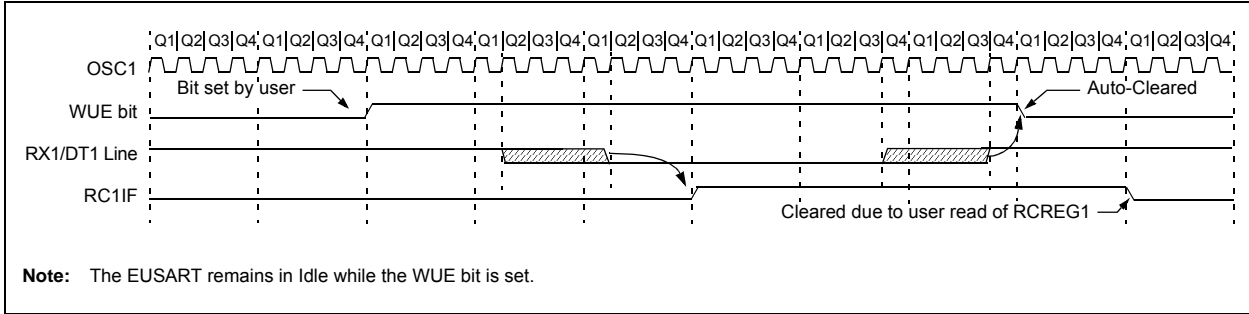
The fact that the WUE bit has been cleared (or is still set) and the RC1IF flag is set should not be used as an indicator of the integrity of the data in RCREG1. Users should consider implementing a parallel method in firmware to verify received data integrity.

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.

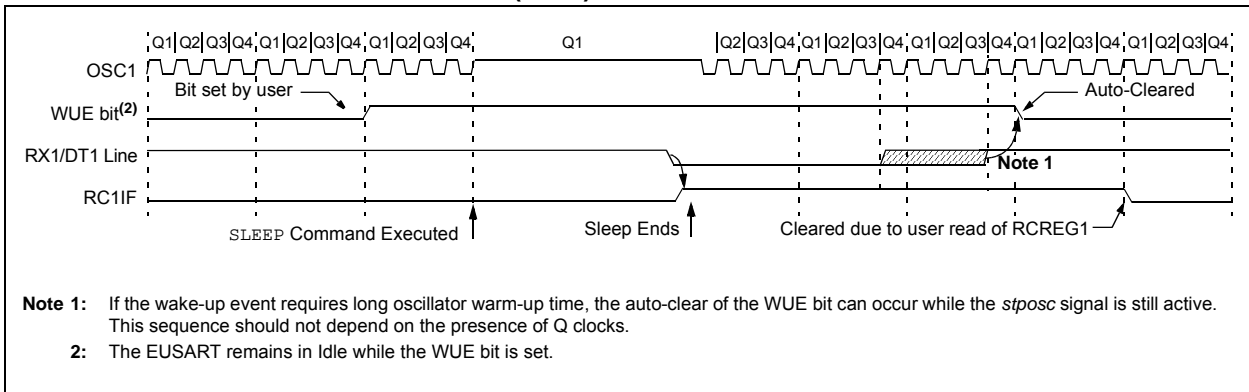


# PIC18F6310/6410/8310/8410

**FIGURE 18-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION**



**FIGURE 18-9: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**



# PIC18F6310/6410/8310/8410

## 18.2.5 BREAK CHARACTER SEQUENCE

The Enhanced USART 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 TXREG1 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 TXREG1 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 18-10](#) for the timing of the Break character sequence.

### 18.2.5.1 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 TXREG1 with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG1 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 TXREG1 becomes empty, as indicated by the TX1IF bit, the next data byte can be written to TXREG1.

## 18.2.6 RECEIVING A BREAK CHARACTER

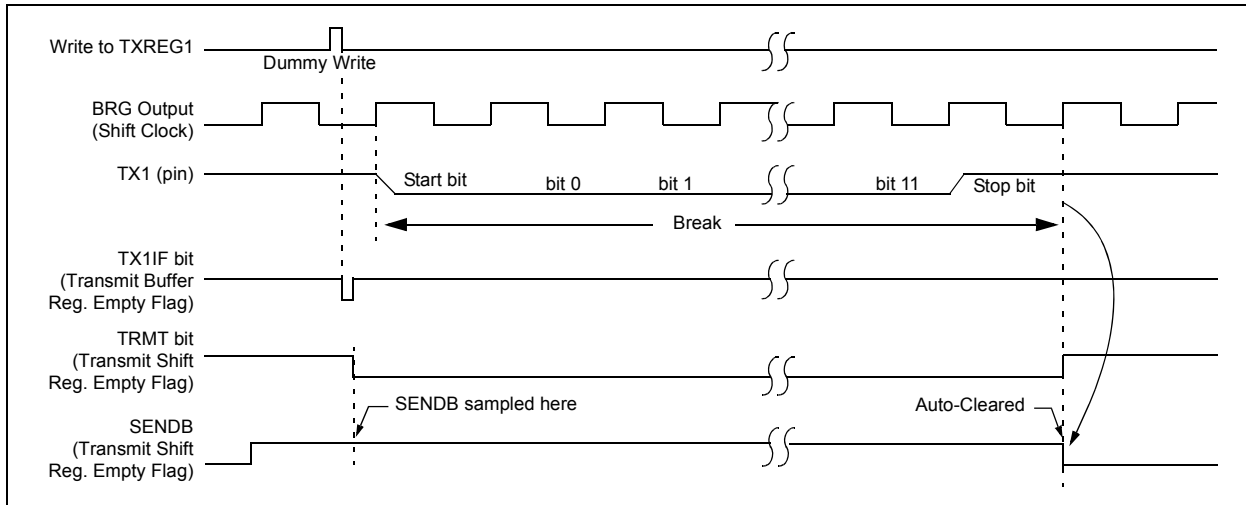
The Enhanced USART module can receive a Break character in two ways.

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 18.2.4 "Auto-Wake-up on Sync Break Character"](#). By enabling this feature, the EUSART will sample the next two transitions on RX1/DT1, cause an RC1IF 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 TX1IF interrupt is observed.

**FIGURE 18-10: SEND BREAK CHARACTER SEQUENCE**



## 18.3 EUSART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTA<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit, SYNC (TXSTA<4>). In addition, enable bit, SPEN (RCSTA1<7>), is set in order to configure the TX1 and RX1 pins to CK1 (clock) and DT1 (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CK1 line. Clock polarity (CK1) is selected with the TXCKP bit (BAUDCON<4>). Setting TXCKP sets the Idle state on CK1 as high, while clearing the bit sets the Idle state as low.

### 18.3.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in [Figure 18-3](#). 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, TXREG1. The TXREG1 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 TXREG1 register (if available).

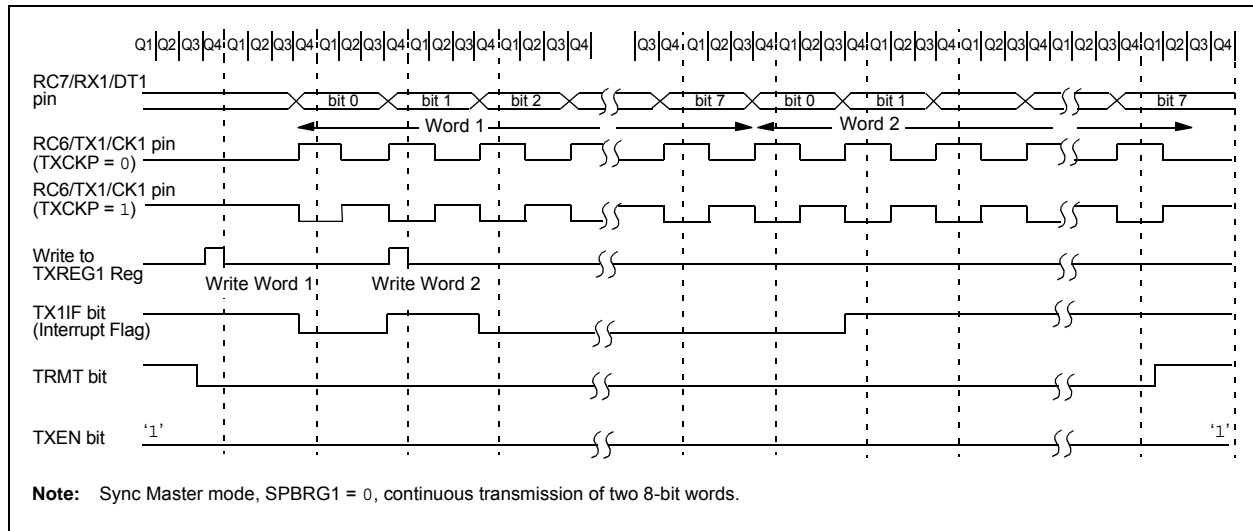
Once the TXREG1 register transfers the data to the TSR register (occurs in one T<sub>CYCLE</sub>), the TXREG1 is empty and the TX1IF flag bit (PIR1<4>) is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TX1IE (PIE1<4>). TX1IF is set regardless of the state of enable bit, TX1IE; it cannot be cleared in software. It will reset only when new data is loaded into the TXREG1 register.

While flag bit, TX1IF, indicates the status of the TXREG1 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:

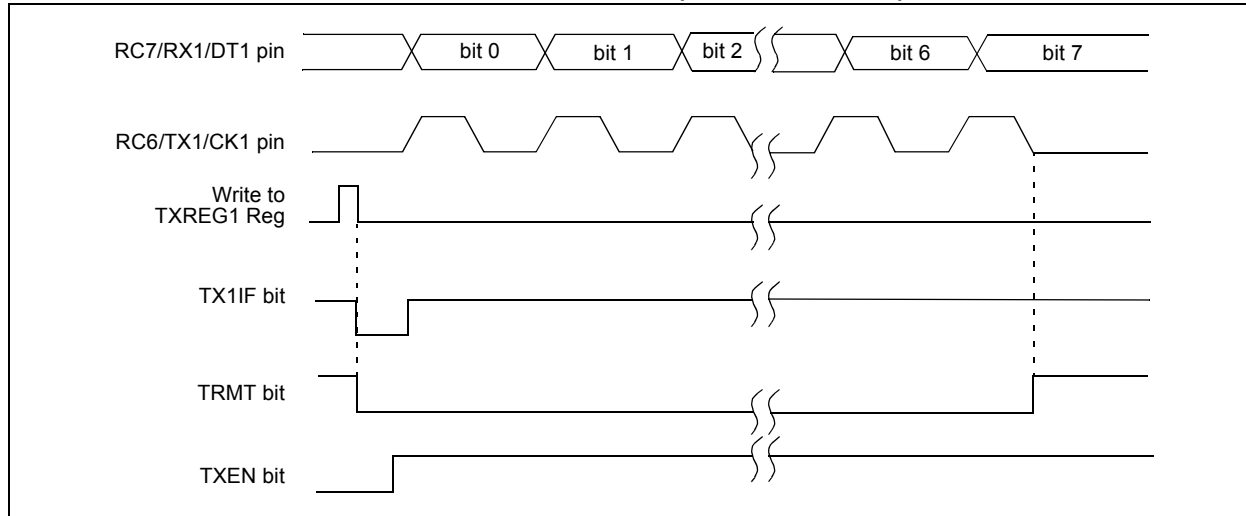
1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. If the signal from the CKx pin is to be inverted, set the TXCKP bit.
4. If interrupts are desired, set enable bit, TXIE.
5. If 9-bit transmission is desired, set bit, TX9.
6. Enable the transmission by setting bit, TXEN.
7. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
8. Start transmission by loading data to the TXREG register.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 18-11: SYNCHRONOUS TRANSMISSION**



# PIC18F6310/6410/8310/8410

**FIGURE 18-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**



**TABLE 18-7: 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	65
TXREG1	EUSART1 Transmit Register								65
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	65
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	66
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte								66
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte								65

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

## 18.3.2 EUSART SYNCHRONOUS MASTER RECEPTION

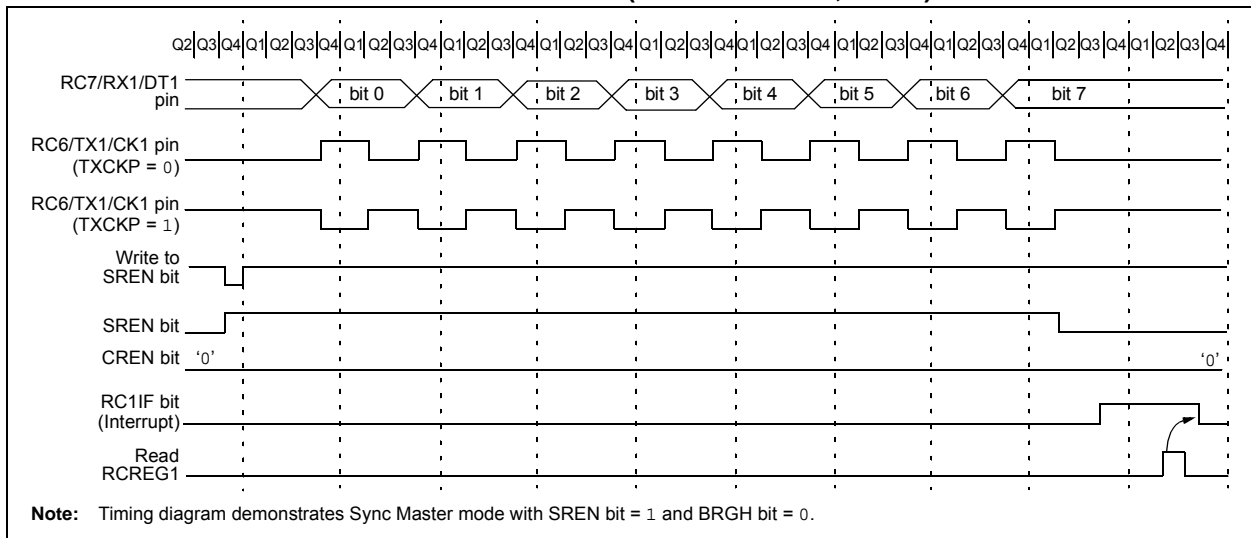
Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTA1<5>), or the Continuous Receive Enable bit, CREN (RCSTA1<4>). Data is sampled on the RX1 pin on the falling edge of the clock.

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 the signal from the CKx pin is to be inverted, set the TXCKP bit.
5. If interrupts are desired, set enable bit, RCIE.
6. If 9-bit reception is desired, set bit, RX9.
7. If a single reception is required, set bit, SREN. For continuous reception, set bit, CREN.
8. Interrupt flag bit, RCIF, will be set when reception is complete and an interrupt will be generated if the enable bit, RCIE, was set.
9. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
10. Read the 8-bit received data by reading the RCREG register.
11. If any error occurred, clear the error by clearing bit, CREN.
12. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 18-13: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**



# PIC18F6310/6410/8310/8410

**TABLE 18-8: 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	65
RCREG1	EUSART1 Receive Register								65
TXSTA1	CSRC	TX9	TXEN	SYNC	SEnDB	BRGH	TRMT	TX9D	65
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	66
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte								66
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte								65

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.

# PIC18F6310/6410/8310/8410

## 18.4 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 CK1 pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

### 18.4.1 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 TXREG1 and then the SLEEP instruction is executed, the following will occur:

- The first word will immediately transfer to the TSR register and transmit.
- The second word will remain in the TXREG1 register.
- Flag bit, TX1IF, will not be set.
- When the first word has been shifted out of TSR, the TXREG1 register will transfer the second word to the TSR and flag bit, TX1IF, will now be set.
- If enable bit, TX1IE, 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:

- Enable the synchronous slave serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
- Clear bits, CREN and SREN.
- If interrupts are desired, set enable bit, TXIE.
- If the signal from the CKx pin is to be inverted, set the TXCKP bit.
- If 9-bit transmission is desired, set bit, TX9.
- Enable the transmission by setting enable bit, TXEN.
- If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
- Start transmission by loading data to the TXREGx register.
- If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 18-9: 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	65
TXREG1	EUSART1 Transmit Register								65
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	65
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	66
SPBRGH1	EUSART1 Baud Rate Generator Register High Byte								66
SPBRG1	EUSART1 Baud Rate Generator Register Low Byte								65

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

# PIC18F6310/6410/8310/8410

## 18.4.2 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 RCREG1 register; if the RC1IE 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 the signal from the CKx pin is to be inverted, set the TXCKP bit.
4. If 9-bit reception is desired, set bit, RX9.
5. To enable reception, set enable bit, CREN.
6. Flag bit, RCIF, will be set when reception is complete. 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 bit, CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 18-10: 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	65
RCREG1	EUSART1 Receive Register								65
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	65
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	66
SPBRGH1	Baud Rate Generator Register High Byte								66
SPBRG1	Baud Rate Generator Register Low Byte								65

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.



## 19.0 ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (AUSART)

The Addressable Universal Synchronous Asynchronous Receiver Transmitter (AUSART) module is very similar in function to the Enhanced USART module, discussed in the previous chapter. It is provided as an additional channel for serial communication with external devices, for those situations that do not require Auto-Baud Detection (ABD) or LIN/J2602 bus support.

The AUSART can be configured in the following modes:

- Asynchronous (full-duplex)
- Synchronous – Master (half-duplex)
- Synchronous – Slave (half-duplex)

The pins of the AUSART module are multiplexed with the functions of PORTG (RG1/TX2/CK2 and RG2/RX2/DT2, respectively). In order to configure these pins as an AUSART:

- SPEN bit (RCSTA2<7>) must be set (= 1)
- TRISG<2> bit must be set (= 1)
- TRISG<1> bit must be cleared (= 0) for Asynchronous and Synchronous Master modes
- TRISG<1> bit must be set (= 1) for Synchronous Slave mode

<b>Note:</b> The USART control will automatically reconfigure the pin from input to output as needed.
---

The operation of the Addressable USART module is controlled through two registers: TXSTA2 and RXSTA2. These are detailed in [Register 19-1](#) and [Register 19-2](#) respectively.

# PIC18F6310/6410/8310/8410

## REGISTER 19-1: TXSTA2: AUSART2 TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN <sup>(1)</sup>	SYNC	—	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<sup>(1)</sup>  
 1 = Transmit is enabled  
 0 = Transmit is disabled
- bit 4      **SYNC:** AUSART Mode Select bit  
 1 = Synchronous mode  
 0 = Asynchronous mode
- bit 3      **Unimplemented:** Read as '0'
- 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 is empty  
 0 = TSR is full
- bit 0      **TX9D:** 9th bit of Transmit Data bit  
 Can be address/data bit or a parity bit.

**Note 1:** SREN/CREN overrides TXEN in Sync mode.

# PIC18F6310/6410/8310/8410

## REGISTER 19-2: RCSTA2: AUSART2 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 is enabled (configures RXx/DTx and TXx/CKx pins as serial port pins)  
 0 = Serial port is 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> are set  
 0 = Disables address detection, all bytes are received and ninth bit can be used as a 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 RCREG1 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 bit  
 This can be address/data bit or a parity bit and must be calculated by user firmware.

# PIC18F6310/6410/8310/8410

## 19.1 AUSART Baud Rate Generator (BRG)

The BRG is a dedicated, 8-bit generator that supports both the Asynchronous and Synchronous modes of the AUSART.

The SPBRG2 register controls the period of a free-running timer. In Asynchronous mode, BRGH bit (TXSTA<2>) also controls the baud rate. In Synchronous mode, BRGH is ignored. Table 19-1 shows the formula for computation of the baud rate for different AUSART modes, which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRG2 register 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) to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRG2 register 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.

### 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 SPBRG2 register.

### 19.1.2 SAMPLING

The data on the RX2 pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX2 pin.

**TABLE 19-1: BAUD RATE FORMULAS**

Configuration Bits		BRG/AUSART Mode	Baud Rate Formula
SYNC	BRGH		
0	0	Asynchronous	$F_{OSC}/[64 (n + 1)]$
0	1	Asynchronous	$F_{OSC}/[16 (n + 1)]$
1	x	Synchronous	$F_{OSC}/[4 (n + 1)]$

**Legend:** x = Don't care, n = Value of SPBRG2 register

### EXAMPLE 19-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, BRGH = 0:

$$\text{Desired Baud Rate} = F_{OSC}/(64 ([SPBRG2] + 1))$$

Solving for SPBRG2:

$$\begin{aligned} X &= ((F_{OSC}/\text{Desired Baud Rate})/64) - 1 \\ &= ((16000000/9600)/64) - 1 \\ &= [25.042] = 25 \end{aligned}$$

$$\begin{aligned} \text{Calculated Baud Rate} &= 16000000/(64 (25 + 1)) \\ &= 9615 \end{aligned}$$

$$\begin{aligned} \text{Error} &= (\text{Calculated Baud Rate} - \text{Desired Baud Rate})/\text{Desired Baud Rate} \\ &= (9615 - 9600)/9600 = 0.16\% \end{aligned}$$

**TABLE 19-2: REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	66
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	66
SPBRG2	AUSART2 Baud Rate Generator Register								66

**Legend:** Shaded cells are not used by the BRG.

# PIC18F6310/6410/8310/8410

**TABLE 19-3: BAUD RATES FOR ASYNCHRONOUS MODES**

BAUD RATE (K)	BRGH = 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)	BRGH = 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)	BRGH = 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	—	—	—	—	—	—	—	—	—	—	—	—
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)	BRGH = 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	-0.16	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	—	—	—	—	—	—

# PIC18F6310/6410/8310/8410

## 19.2 AUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTA2<4>). In this mode, the AUSART 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 Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The AUSART transmits and receives the LSb first. The AUSART'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 bit (TXSTA2<2>). 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 AUSART module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver

### 19.2.1 AUSART ASYNCHRONOUS TRANSMITTER

The AUSART transmitter block diagram is shown in Figure 19-1. 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, TXREG2. The TXREG2 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 TXREG2 register (if available).

Once the TXREG2 register transfers the data to the TSR register (occurs in one Tcy), the TXREG2 register is empty and the TX2IF flag bit (PIR3<4>) is set. This

interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TX2IE (PIE3<4>). TX2IF will be set regardless of the state of TX2IE; it cannot be cleared in software. TX2IF is also not cleared immediately upon loading TXREG2, but becomes valid in the second instruction cycle following the load instruction. Polling TX2IF immediately following a load of TXREG2 will return invalid results.

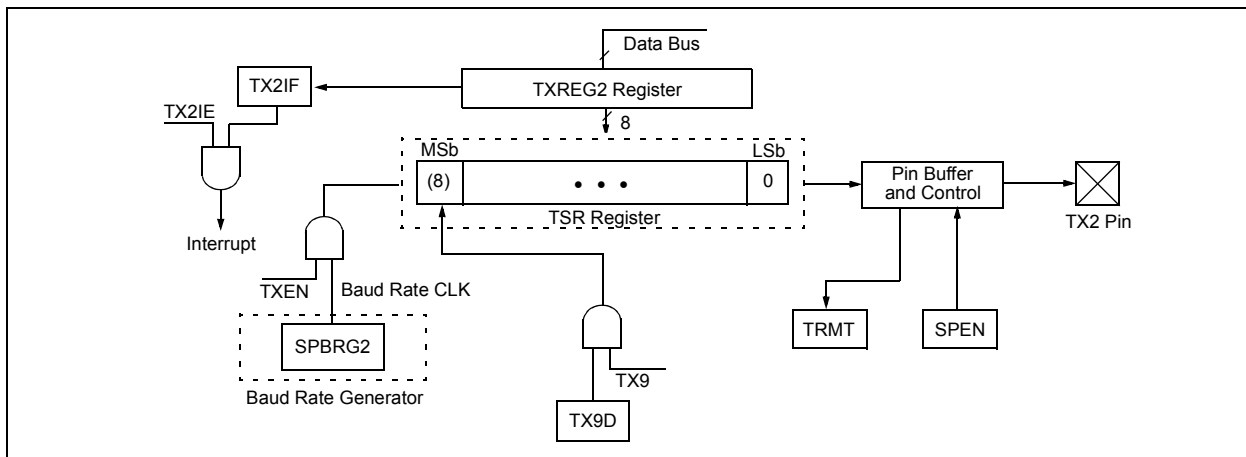
While TX2IF indicates the status of the TXREG2 register, another bit, TRMT (TXSTA2<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty.

**Note 1:** The TSR register is not mapped in data memory so it is not available to the user.  
**2:** Flag bit, TX2IF, is set when enable bit, TXEN is set.

To set up an Asynchronous Transmission:

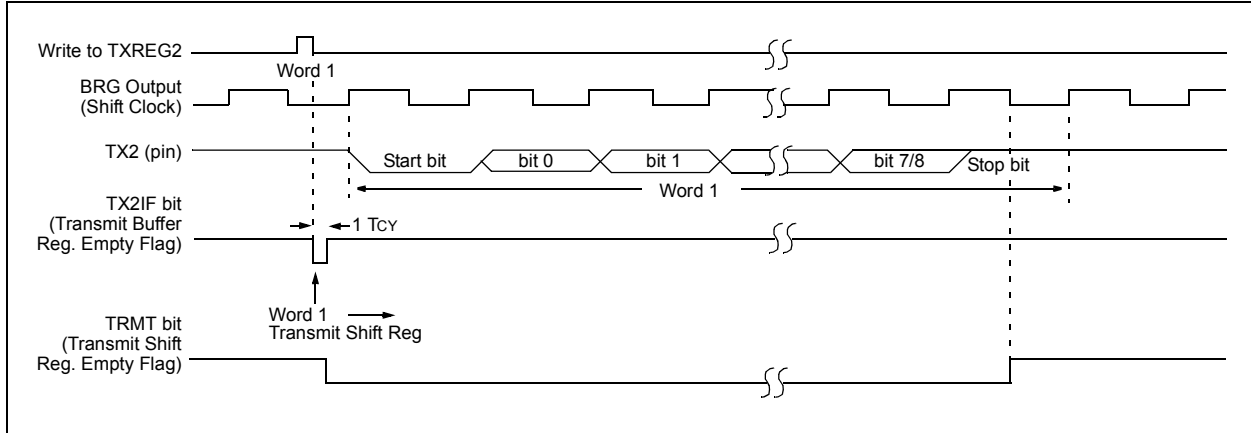
1. Initialize the SPBRG2 register for the appropriate baud rate. Set or clear the BRGH bit, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, TX2IE.
4. If 9-bit transmission 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, TX2IF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Load data to the TXREG2 register (starts transmission).
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 19-1: AUSART TRANSMIT BLOCK DIAGRAM

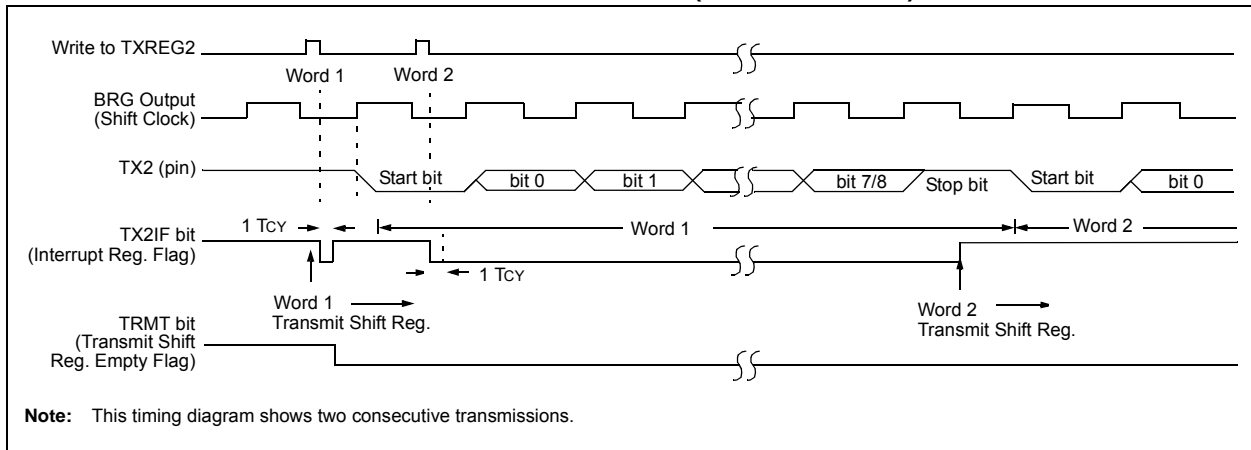


# PIC18F6310/6410/8310/8410

**FIGURE 19-2: ASYNCHRONOUS TRANSMISSION**



**FIGURE 19-3: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)**



**Note:** This timing diagram shows two consecutive transmissions.

**TABLE 19-4: 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	63
PIR3	—	—	RC2IF	TX2IF	—	—	—	CCP3IF	65
PIE3	—	—	RC2IE	TX2IE	—	—	—	CCP3IE	65
IPR3	—	—	RC2IP	TX2IP	—	—	—	CCP3IP	65
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	66
TXREG2	AUSART2 Transmit Register								66
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	66
SPBRG2	AUSART2 Baud Rate Generator Register								66

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

# PIC18F6310/6410/8310/8410

## 19.2.2 AUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in [Figure 19-4](#). The data is received on the RX2 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:

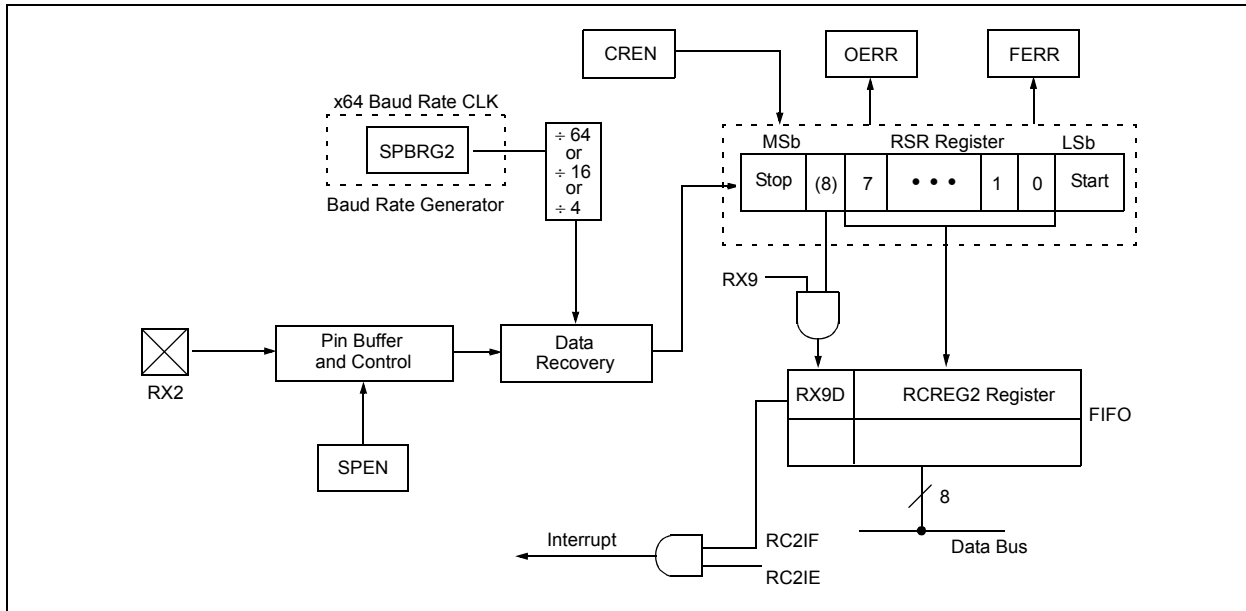
1. Initialize the SPBRG2 register for the appropriate baud rate. Set or clear the BRGH bit, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RC2IE.
4. If 9-bit reception is desired, set bit, RX9.
5. Enable the reception by setting bit, CREN.
6. Flag bit, RC2IF, will be set when reception is complete and an interrupt will be generated if enable bit, RC2IE, was set.
7. Read the RCSTA2 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 RCREG2 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 SPBRG2 register 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 RC2IP 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 RC2IF bit will be set when reception is complete. The interrupt will be Acknowledged if the RC2IE and GIE bits are set.
8. Read the RCSTA2 register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREG2 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.

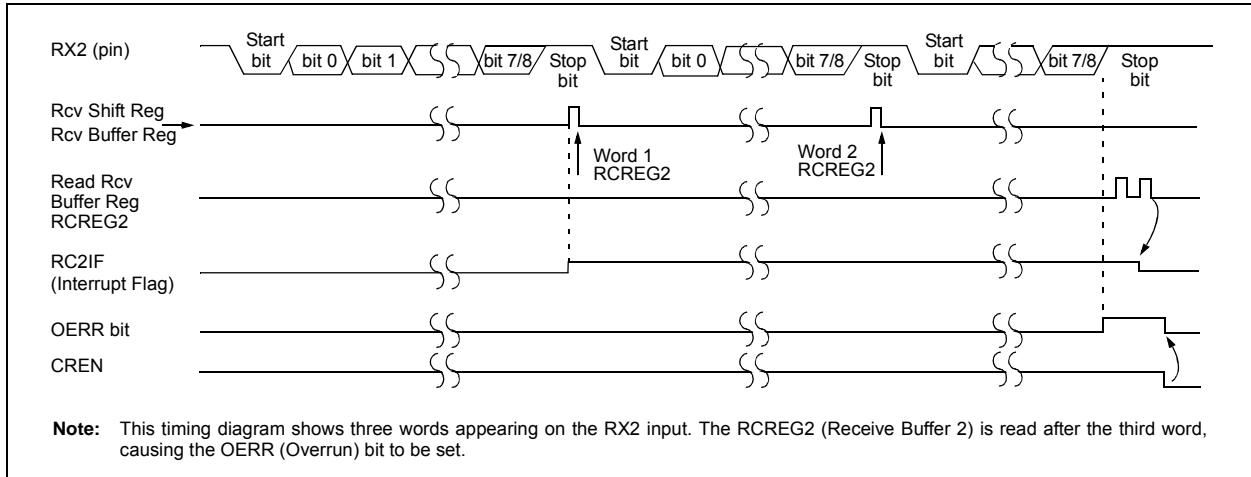
**FIGURE 19-4: AUSART RECEIVE BLOCK DIAGRAM**





# PIC18F6310/6410/8310/8410

**FIGURE 19-5: ASYNCHRONOUS RECEPTION**



**TABLE 19-5: 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	63
PIR3	—	—	RC2IF	TX2IF	—	—	—	CCP3IF	65
PIE3	—	—	RC2IE	TX2IE	—	—	—	CCP3IE	65
IPR3	—	—	RC2IP	TX2IP	—	—	—	CCP3IP	65
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	66
RCREG2	AUSART2 Receive Register								66
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	66
SPBRG2	AUSART2 Baud Rate Generator Register								66

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

# PIC18F6310/6410/8310/8410

## 19.3 AUSART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTA2<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 (TXSTA2<4>). In addition, enable bit, SPEN (RCSTA2<7>), is set in order to configure the TX2 and RX2 pins to CK2 (clock) and DT2 (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CK2 line.

### 19.3.1 AUSART SYNCHRONOUS MASTER TRANSMISSION

The AUSART transmitter block diagram is shown in Figure 19-1. 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, TXREG2. The TXREG2 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 TXREG2 (if available).

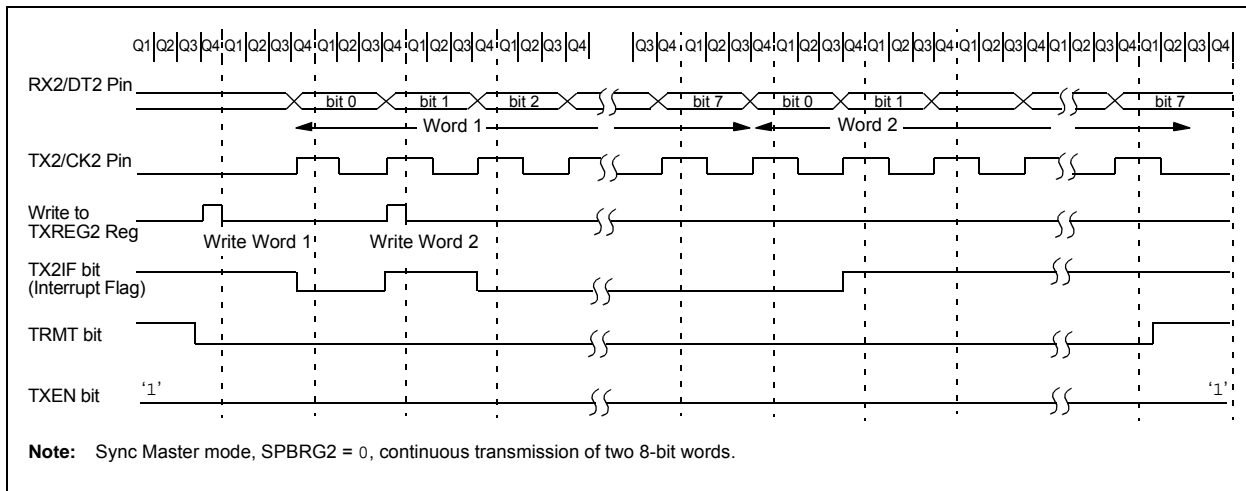
Once the TXREG2 register transfers the data to the TSR register (occurs in one T<sub>CYCLE</sub>), the TXREG2 is empty and the TX2IF flag bit (PIR3<4>) is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TX2IE (PIE3<4>). TX2IF is set regardless of the state of enable bit, TX2IE; it cannot be cleared in software. It will reset only when new data is loaded into the TXREG2 register.

While flag bit, TX2IF, indicates the status of the TXREG2 register, another bit, TRMT (TXSTA2<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:

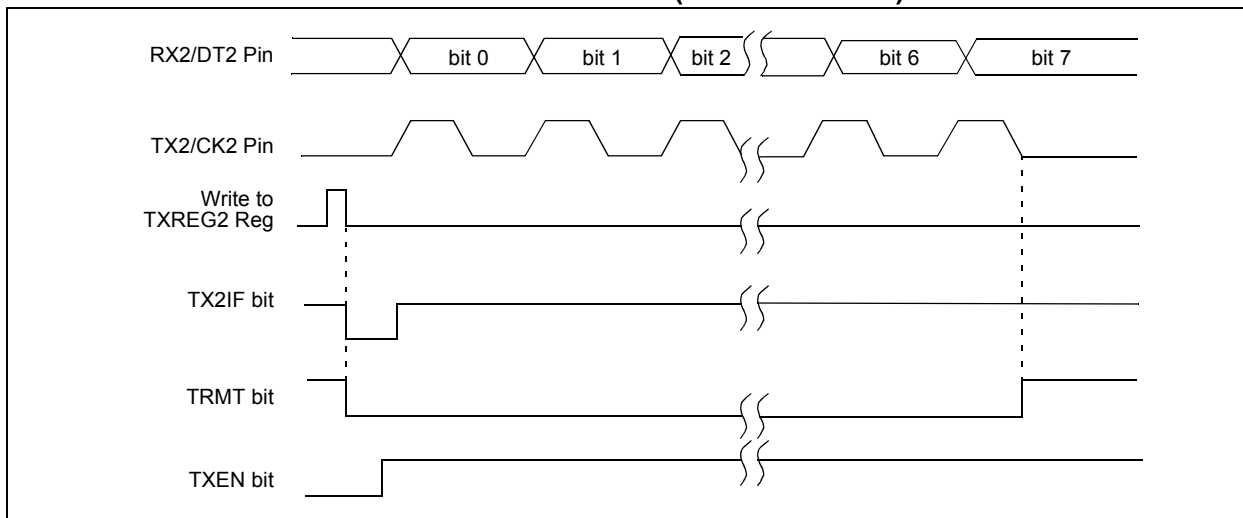
1. Initialize the SPBRG2 register for the appropriate baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit, TX2IE.
4. If 9-bit transmission is desired, set bit, TX9.
5. Enable the transmission by setting bit, TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Start transmission by loading data to the TXREG2 register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 19-6: SYNCHRONOUS TRANSMISSION**



# PIC18F6310/6410/8310/8410

**FIGURE 19-7: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**



**TABLE 19-6: 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	63
PIR3	—	—	RC2IF	TX2IF	—	—	—	CCP3IF	65
PIE3	—	—	RC2IE	TX2IE	—	—	—	CCP3IE	65
IPR3	—	—	RC2IP	TX2IP	—	—	—	CCP3IP	65
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	66
TXREG2	AUSART2 Transmit Register								66
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	66
SPBRG2	AUSART2 Baud Rate Generator Register								66

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

# PIC18F6310/6410/8310/8410

## 19.3.2 AUSART SYNCHRONOUS MASTER RECEPTION

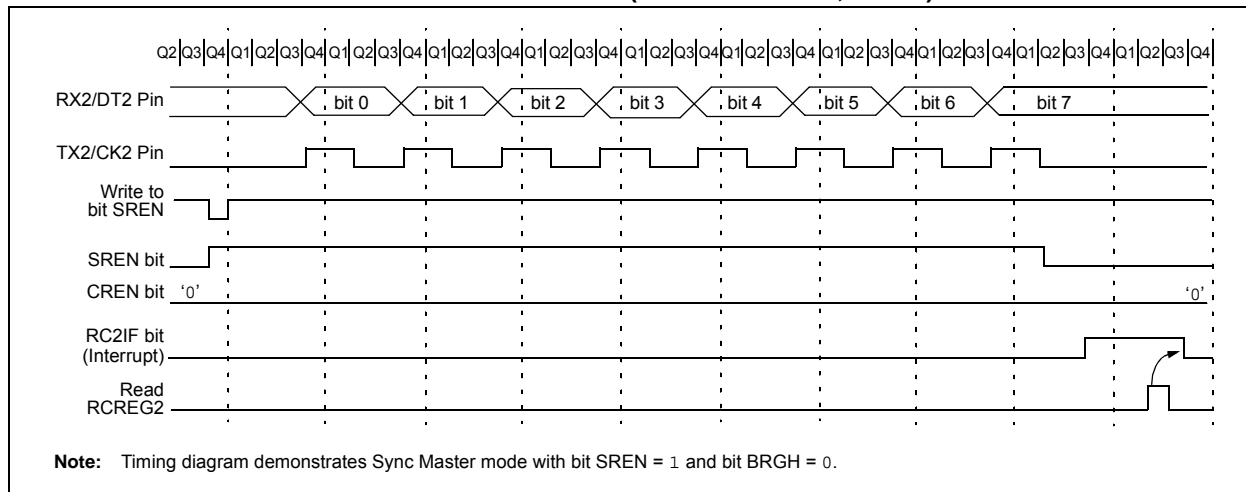
Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTA2<5>), or the Continuous Receive Enable bit, CREN (RCSTA2<4>). Data is sampled on the RX2 pin on the falling edge of the clock.

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 SPBRG2 register for the appropriate 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, RC2IE.
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, RC2IF, will be set when reception is complete and an interrupt will be generated if the enable bit, RC2IE, was set.
8. Read the RCSTA2 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 RCREG2 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 19-8: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**



**TABLE 19-7: 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	63
PIR3	—	—	RC2IF	TX2IF	—	—	—	CCP3IF	65
PIE3	—	—	RC2IE	TX2IE	—	—	—	CCP3IE	65
IPR3	—	—	RC2IP	TX2IP	—	—	—	CCP3IP	65
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	66
RCREG2	AUSART2 Receive Register								66
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	66
SPBRG2	AUSART2 Baud Rate Generator Register Low Byte								66

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.

# PIC18F6310/6410/8310/8410

## 19.4 AUSART Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit, CSRC (TXSTA2<7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the CK2 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 AUSART 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 TXREG2 and then the SLEEP instruction is executed, the following will occur:

- The first word will immediately transfer to the TSR register and transmit.
- The second word will remain in TXREG2 register.
- Flag bit, TX2IF, will not be set.
- When the first word has been shifted out of TSR, the TXREG2 register will transfer the second word to the TSR and flag bit, TX2IF, will now be set.
- If enable bit, TX2IE, 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:

- Enable the synchronous slave serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
- Clear bits, CREN and SREN.
- If interrupts are desired, set enable bit, TX2IE.
- If 9-bit transmission is desired, set bit, TX9.
- Enable the transmission by setting enable bit, TXEN.
- If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
- Start transmission by loading data to the TXREG2 register.
- If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 19-8: 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	63
PIR3	—	—	RC2IF	TX2IF	—	—	—	CCP3IF	65
PIE3	—	—	RC2IE	TX2IE	—	—	—	CCP3IE	65
IPR3	—	—	RC2IP	TX2IP	—	—	—	CCP3IP	65
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	66
TXREG2	AUSART2 Transmit Register								66
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	66
SPBRG2	AUSART2 Baud Rate Generator Register Low Byte								66

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

# PIC18F6310/6410/8310/8410

## 19.4.2 AUSART 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 RCREG2 register; if the RC2IE enable bit is set, the interrupt generated will wake the chip from low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. If interrupts are desired, set enable bit, RC2IE.
3. If 9-bit reception is desired, set bit, RX9.
4. To enable reception, set enable bit, CREN.
5. Flag bit, RC2IF, will be set when reception is complete. An interrupt will be generated if enable bit, RC2IE, was set.
6. Read the RCSTA2 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 RCREG2 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 19-9: 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	63
PIR3	—	—	RC2IF	TX2IF	—	—	—	CCP3IF	65
PIE3	—	—	RC2IE	TX2IE	—	—	—	CCP3IE	65
IPR3	—	—	RC2IP	TX2IP	—	—	—	CCP3IP	65
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	66
RCREG2	AUSART2 Receive Register								66
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	66
SPBRG2	AUSART2 Baud Rate Generator Register Low Byte								66

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave reception.

# PIC18F6310/6410/8310/8410

## 20.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has 12 inputs for the PIC18FX310/X410 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/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)

0110 = Channel 6 (AN6)

0111 = Channel 7 (AN7)

1000 = Channel 8 (AN8)

1001 = Channel 9 (AN9)

1010 = Channel 10 (AN10)

1011 = Channel 11 (AN11)

1100 = Unimplemented<sup>(1)</sup>

1101 = Unimplemented<sup>(1)</sup>

1110 = Unimplemented<sup>(1)</sup>

1111 = Unimplemented<sup>(1)</sup>

bit 1

**GO/DONE:** A/D Conversion Status bit

When ADON = 1:

1 = A/D conversion is in progress

0 = A/D is Idle

bit 0

**ADON:** A/D On bit

1 = A/D Converter module is enabled

0 = A/D Converter module is disabled

**Note 1:** Performing a conversion on unimplemented channels will return a floating input measurement.

# PIC18F6310/6410/8310/8410

## REGISTER 20-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-q	R/W-q	R/W-q	R/W-q	
—	—	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 (VREF- source):  
 1 = VREF- (AN2)  
 0 = AVSS

bit 4                      **VCFG0:** Voltage Reference Configuration bit (VREF+ source):  
 1 = VREF+ (AN3)  
 0 = AVDD

bit 3-0                      **PCFG<3:0>:** A/D Port Configuration Control bits:

PCFG<3:0>	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A
0011	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input

D = Digital I/O



# PIC18F6310/6410/8310/8410

## 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<sup>(1)</sup>
- bit 2-0                      **ADCS<2:0>:** A/D Conversion Clock Select bits  
                             111 = FRC (clock derived from A/D RC oscillator)<sup>(1)</sup>  
                             110 = FOSC/64  
                             101 = FOSC/16  
                             100 = FOSC/4  
                             011 = FRC (clock derived from A/D RC oscillator)<sup>(1)</sup>  
                             010 = FOSC/32  
                             001 = FOSC/8  
                             000 = FOSC/2

**Note 1:** If the A/D FRC clock source is selected, a delay of one T<sub>CY</sub> (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

# PIC18F6310/6410/8310/8410

The analog reference voltage is software-selectable to either the device's positive and negative supply voltage (AVDD and AVSS), or the voltage level on the RA3/AN3/VREF+ and RA2/AN2/VREF- pins.

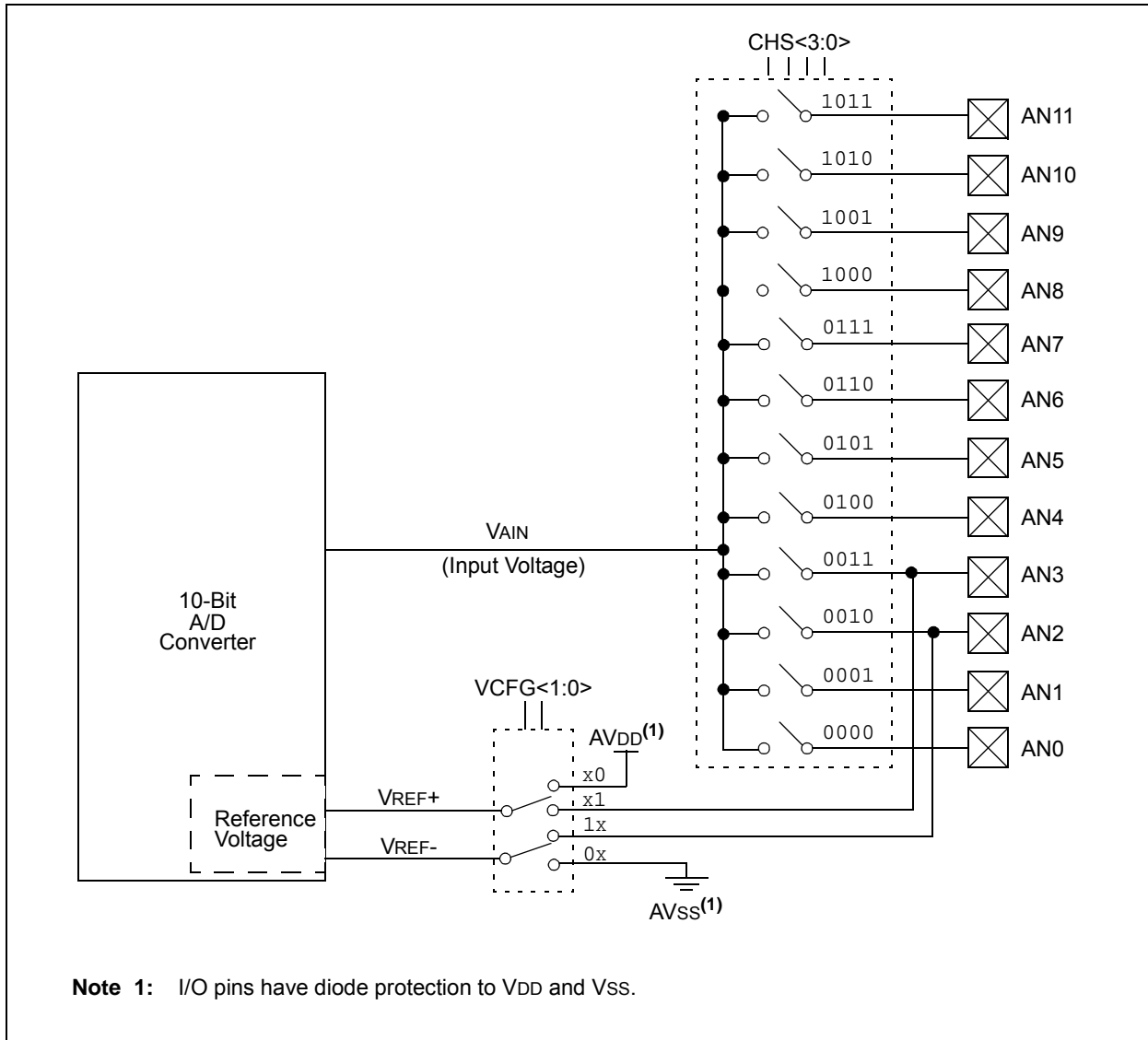
The A/D Converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion in progress is aborted.

Each port pin associated with the A/D Converter can be configured as an analog input or as a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH/ADRESL registers, the GO/DONE bit (ADCON0 register) is cleared and the 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**



# PIC18F6310/6410/8310/8410

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.

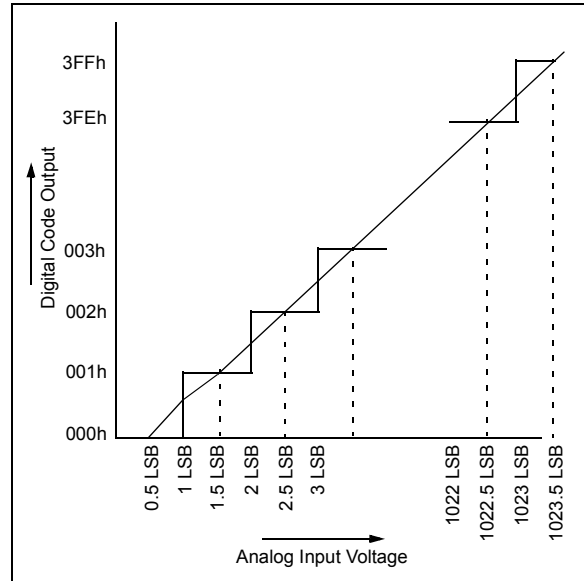
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see [Section 20.1 "A/D Acquisition Requirements"](#). After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the  $\overline{\text{GO/DONE}}$  bit and the actual start of the conversion.

The following steps should be followed to perform an A/D conversion:

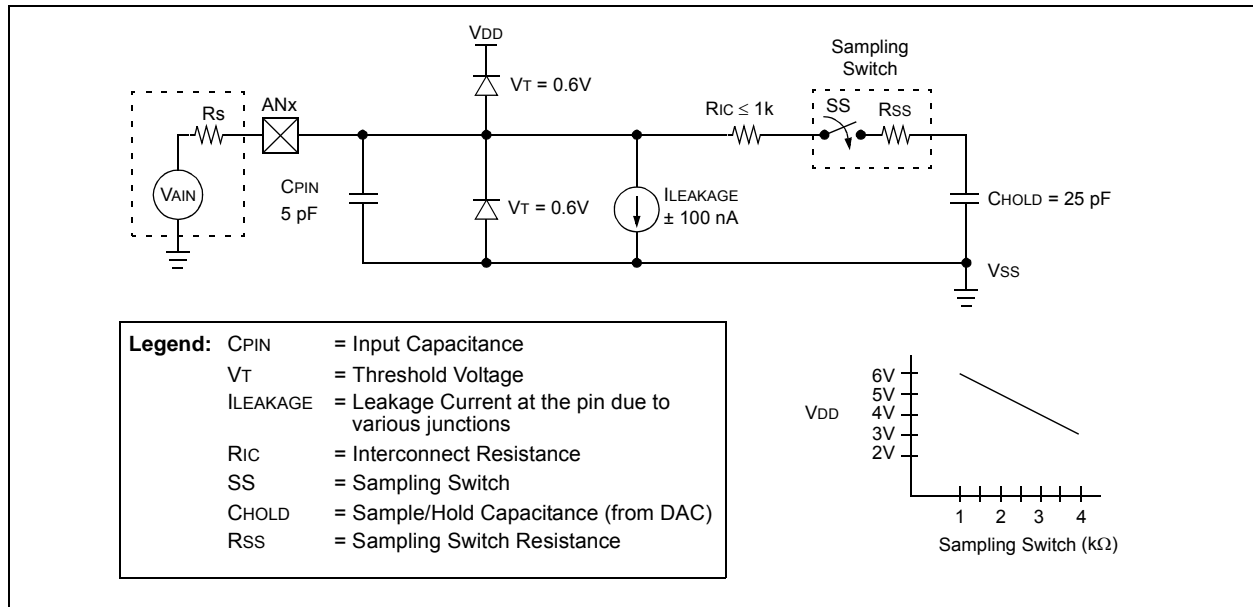
1. Configure the A/D module:
  - Configure analog pins, voltage reference and digital I/O (ADCON1)
  - Select A/D input channel (ADCON0)
  - Select A/D acquisition time (ADCON2)
  - Select A/D conversion clock (ADCON2)
  - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
  - Clear ADIF bit
  - Set ADIE bit
  - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
  - Set  $\overline{\text{GO/DONE}}$  bit (ADCON0 register)

5. Wait for A/D conversion to complete, by either:
  - Polling for the  $\overline{\text{GO/DONE}}$  bit to be cleared
  - OR
  - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit, ADIF, if required.
7. For 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 3 TAD is required before the next acquisition starts.

**FIGURE 20-2: A/D TRANSFER FUNCTION**



**FIGURE 20-3: ANALOG INPUT MODEL**



# PIC18F6310/6410/8310/8410

## 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 20-3](#). 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	=	25 pF
Rs	=	2.5 kΩ
Conversion Error	≤	1/2 LSB
VDD	=	5V → Rss = 2 kΩ
Temperature	=	85°C (system max.)

### EQUATION 20-1: ACQUISITION TIME

$$\begin{aligned} \text{TACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= \text{TAMP} + \text{TC} + \text{TCOFF} \end{aligned}$$

### EQUATION 20-2: A/D MINIMUM CHARGING TIME

$$\begin{aligned} \text{VHOLD} &= (\text{VREF} - (\text{VREF}/2048)) \cdot (1 - e^{-(\text{TC}/\text{CHOLD})(\text{RIC} + \text{RSS} + \text{RS})}) \\ \text{or} \\ \text{TC} &= -(\text{CHOLD})(\text{RIC} + \text{RSS} + \text{RS}) \ln(1/2048) \end{aligned}$$

### EQUATION 20-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$$\begin{aligned} \text{TACQ} &= \text{TAMP} + \text{TC} + \text{TCOFF} \\ \text{TAMP} &= 0.2 \mu\text{s} \\ \text{TCOFF} &= (\text{Temp} - 25^\circ\text{C})(0.02 \mu\text{s}/^\circ\text{C}) \\ &\quad (85^\circ\text{C} - 25^\circ\text{C})(0.02 \mu\text{s}/^\circ\text{C}) \\ &\quad 1.2 \mu\text{s} \end{aligned}$$

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, TCOFF = 0 ms.

$$\begin{aligned} \text{TC} &= -(\text{CHOLD})(\text{RIC} + \text{RSS} + \text{RS}) \ln(1/2047) \\ &\quad -(25 \text{ pF})(1 \text{ k}\Omega + 2 \text{ k}\Omega + 2.5 \text{ k}\Omega) \ln(0.0004883) \\ &\quad 1.05 \mu\text{s} \\ \text{TACQ} &= 0.2 \mu\text{s} + 1 \mu\text{s} + 1.2 \mu\text{s} \\ &\quad 2.4 \mu\text{s} \end{aligned}$$

# PIC18F6310/6410/8310/8410

## 20.2 Selecting and Configuring Automatic Acquisition Time

The ADCON2 register allows the user to select an acquisition time that occurs each time the  $\overline{\text{GO/DONE}}$  bit is set. It also gives users the option to use an automatically determined acquisition time. Acquisition time may be set with the ACQT<2:0> bits (ADCON2<5:3>), which provides a range of 2 to 20 TAD.

When the  $\overline{\text{GO/DONE}}$  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  $\overline{\text{GO/DONE}}$  bit. Manual acquisition is selected when ACQT<2:0> = 000. When the  $\overline{\text{GO/DONE}}$  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  $\overline{\text{GO/DONE}}$  bit. This option is also the default Reset state of the ACQT<2:0> bits and is compatible with devices that do not offer programmable acquisition times.

In either case, when the conversion is completed, the  $\overline{\text{GO/DONE}}$  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  $\mu\text{s}$ , 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	ADCS<2:0>	PIC18F6X10/8X10	PIC18LF6X10/8X10 <sup>(4)</sup>
2 TOSC	000	1.25 MHz	666 kHz
4 TOSC	100	2.50 MHz	1.33 MHz
8 TOSC	001	5.00 MHz	2.66 MHz
16 TOSC	101	10.0 MHz	5.33 MHz
32 TOSC	010	20.0 MHz	10.65 MHz
64 TOSC	110	40.0 MHz	21.33 MHz
RC <sup>(3)</sup>	x11	1.00 MHz <sup>(1)</sup>	1.00 MHz <sup>(2)</sup>

- Note 1:** The RC source has a typical TAD time of 4  $\mu\text{s}$ .
- Note 2:** The RC source has a typical TAD time of 6  $\mu\text{s}$ .
- Note 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.
- Note 4:** Low-power (PIC18LFXXXX) devices only.

# PIC18F6310/6410/8310/8410

---

## 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 and TRISF 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 (VOH or VOL) 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.

## 20.6 A/D Conversions

Figure 20-4 shows the operation of the A/D Converter after the GO 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 20-5 shows the operation of the A/D Converter after the GO/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/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).

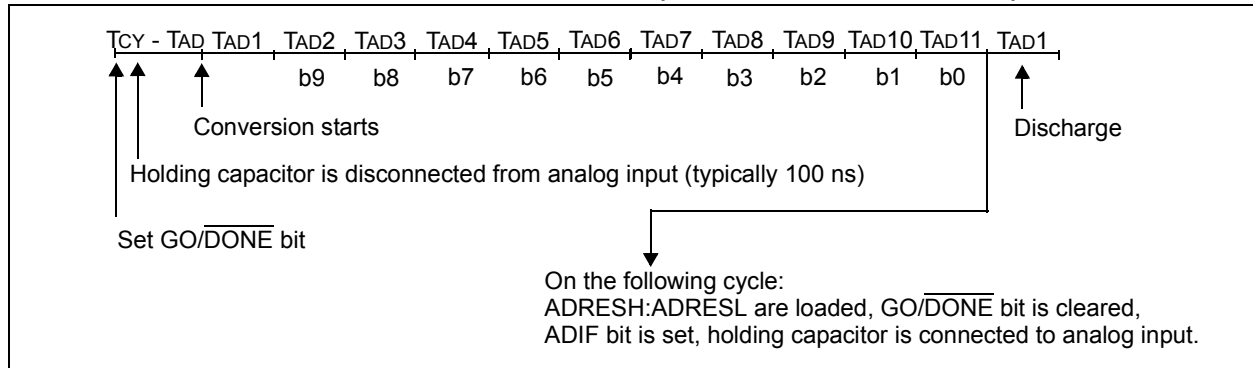
After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

**Note:** The GO/DONE bit should **NOT** be set in the same instruction that turns on the A/D.

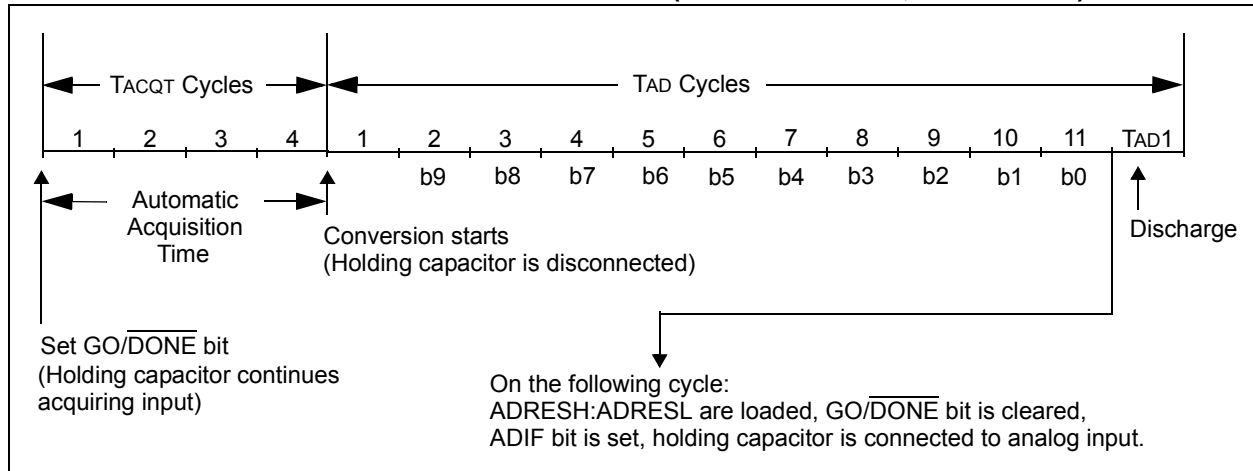
## 20.7 Discharge

The discharge phase is used to initialize the value of the capacitor array. The array is discharged before every sample. This feature helps to optimize the unity-gain amplifier as the circuit always needs to charge the capacitor array, rather than charge/discharge based on previous measure values.

**FIGURE 20-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**



**FIGURE 20-5: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**



# PIC18F6310/6410/8310/8410

## 20.8 Use of the CCP2 Trigger

An A/D conversion can be started by the “Special Event Trigger” of the CCP2 module. This requires that the CCP2M<3:0> bits (CCP2CON<3:0>) be programmed as ‘1011’ and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/DONE bit will be set, starting the A/D acquisition and conversion, and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead

(moving ADRESH/ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition period is either timed by the user, or an appropriate TACQ time selected before the “Special Event Trigger” sets the GO/DONE bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the “Special Event Trigger” will be ignored by the A/D module, but will still reset the Timer1 (or Timer3) counter.

**TABLE 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	63
PIR1	PSPIF	ADIF	RC1IF	TX1IF	SSPIF	CCP1IF	TMR2IF	TMR1IF	65
PIE1	PSPIE	ADIE	RC1IE	TX1IE	SSPIE	CCP1IE	TMR2IE	TMR1IE	65
IPR1	PSPIP	ADIP	RC1IP	TX1IP	SSPIP	CCP1IP	TMR2IP	TMR1IP	65
PIR2	OSCFIF	CMIF	—	—	BCLIF	HLVDIF	TMR3IF	CCP2IF	65
PIE2	OSCFIE	CMIE	—	—	BCLIE	HLVDIE	TMR3IE	CCP2IE	65
IPR2	OSCFIP	CMIP	—	—	BCLIP	HLVDIP	TMR3IP	CCP2IP	65
ADRESH	A/D Result Register High Byte								64
ADRESL	A/D Result Register Low Byte								64
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	64
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	64
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	64
PORTA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5	RA4	RA3	RA2	RA1	RA0	66
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	PORTA Data Direction Register						66
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	66
TRISF	PORTF Data Direction Register								66
LATF	LATF Output Latch Register								66

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used for A/D conversion.

**Note 1:** These pins may be configured as port pins depending on the oscillator mode selected.



# PIC18F6310/6410/8310/8410

## 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 RF3 through RF6, 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-1	R/W-1	R/W-1
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
bit 7				bit 0			

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'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 is inverted  
 0 = C2 output is not inverted
- bit 4      **C1INV:** Comparator 1 Output Inversion bit  
 1 = C1 output is inverted  
 0 = C1 output is not inverted
- bit 3      **CIS:** Comparator Input Switch bit  
When CM<2:0> = 110:  
 1 = C1 VIN- connects to RF5/AN10  
      C2 VIN- connects to RF3/AN8  
 0 = C1 VIN- connects to RF6/AN11  
      C2 VIN- connects to RF4/AN9
- bit 2-0    **CM<2:0>:** Comparator Mode bits  
[Figure 21-1](#) shows the Comparator modes and the CM<2:0> bit settings.

# PIC18F6310/6410/8310/8410

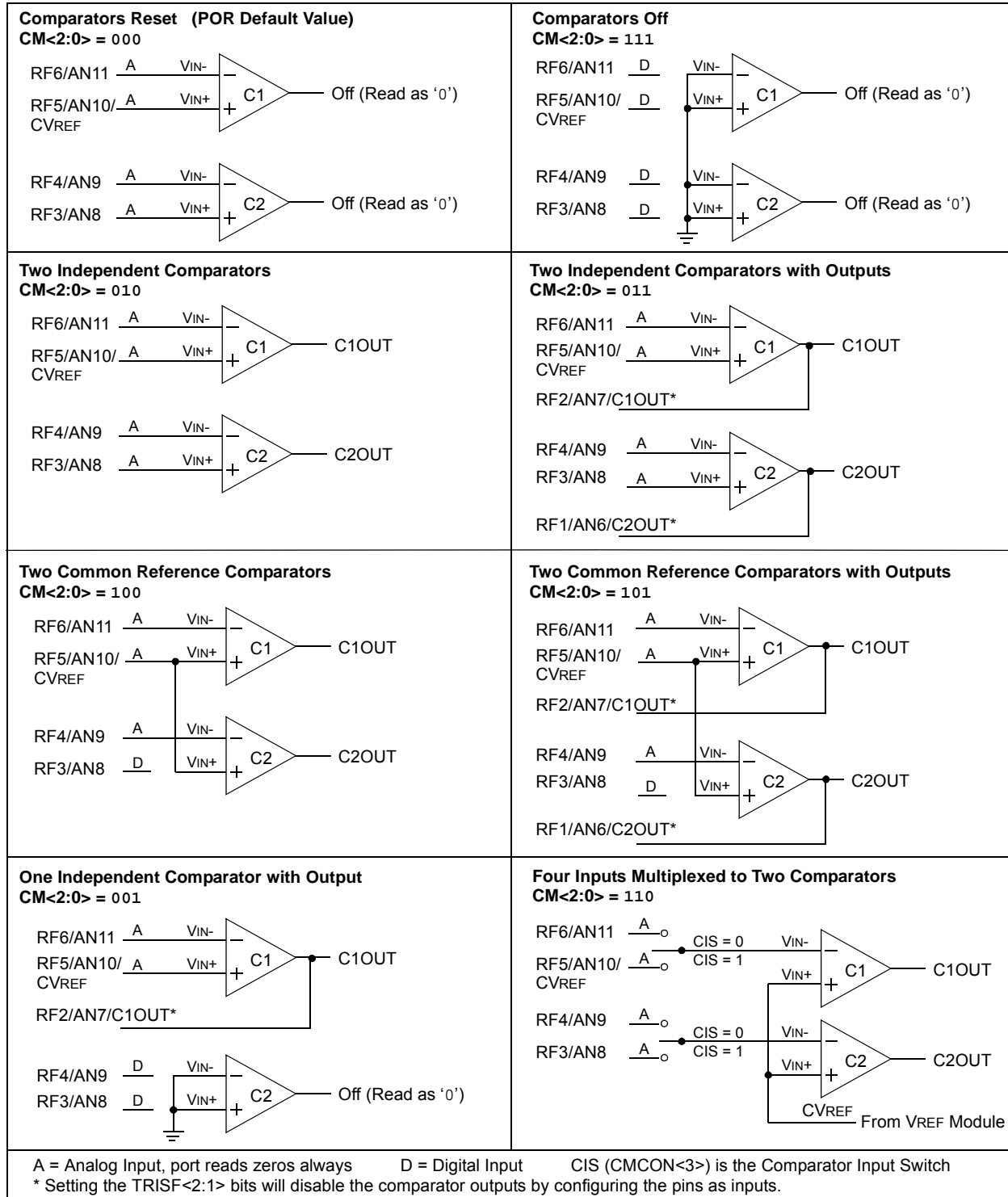
## 21.1 Comparator Configuration

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 TRISF 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 27.0 “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**



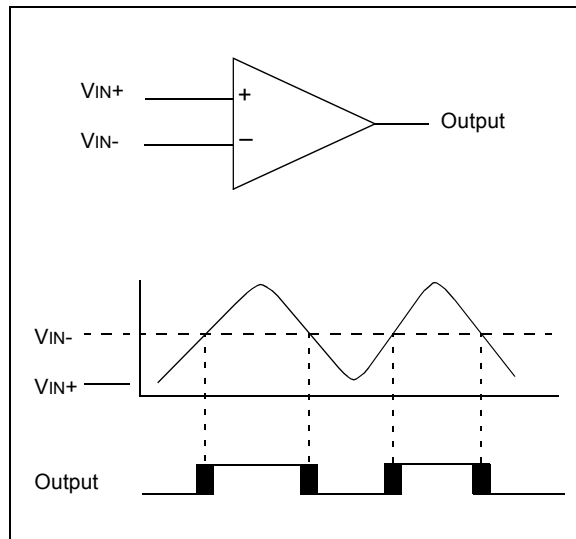
## 21.2 Comparator Operation

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  $V_{IN+}$  is less than the analog input,  $V_{IN-}$ , the output of the comparator is a digital low level. When the analog input at  $V_{IN+}$  is greater than the analog input,  $V_{IN-}$ , 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  $V_{IN-}$  is compared to the signal at  $V_{IN+}$  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  $V_{SS}$  and  $V_{DD}$  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 ( $CM\langle 2:0 \rangle = 110$ ). In this mode, the internal voltage reference is applied to the  $V_{IN+}$  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 27.0 “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  $RF2$  and  $RF1$  I/O pins. When enabled, multiplexors in the output path of the  $RF2$  and  $RF1$  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.

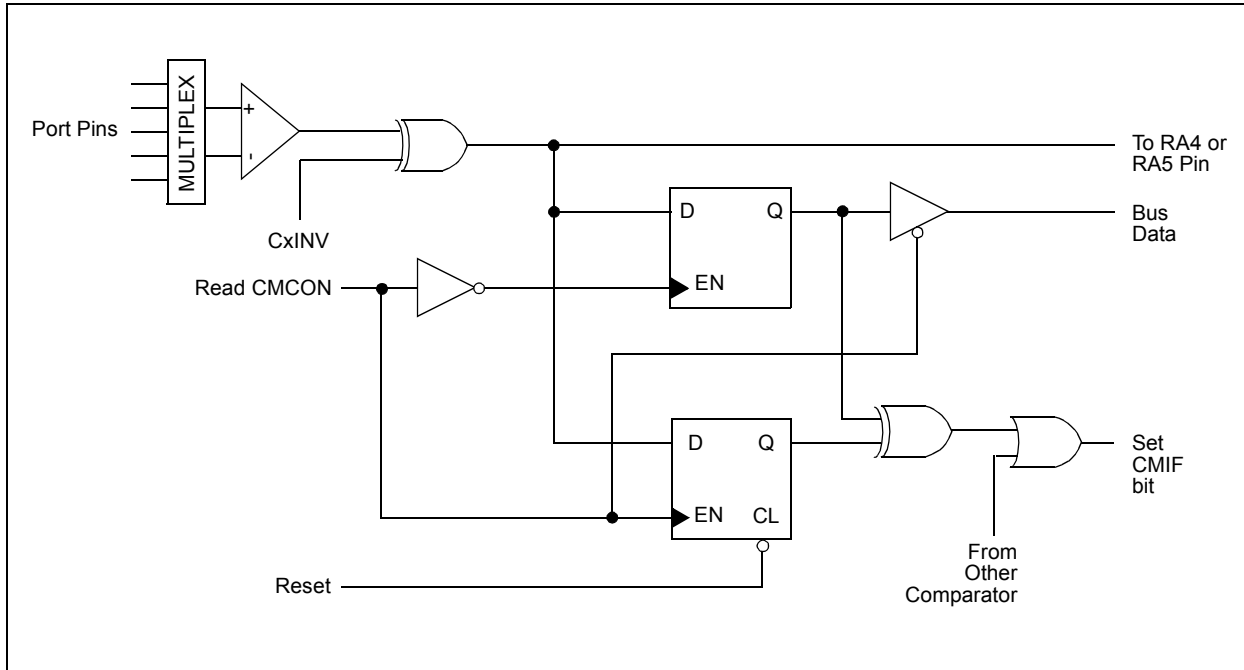
The  $TRISF$  bits will still function as an output enable/disable for the  $RF2$  and  $RF1$  pins while in this mode.

The polarity of the comparator outputs can be changed using the  $C2INV$  and  $C1INV$  bits ( $CMCON\langle 5:4 \rangle$ ).

- 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.

# PIC18F6310/6410/8310/8410

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  $CMCON\langle 7:6 \rangle$ , to determine the actual change that occurred. The CMIF bit ( $PIR2\langle 6 \rangle$ ) 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 ( $PIE2\langle 6 \rangle$ ) and the PEIE bit ( $INTCON\langle 6 \rangle$ ) must be set to enable the interrupt. In addition, the GIE bit ( $INTCON\langle 7 \rangle$ ) 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:

- Any read or write of CMCON will end the mismatch condition.
- 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 ( $CM\langle 2:0 \rangle = 111$ ) 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

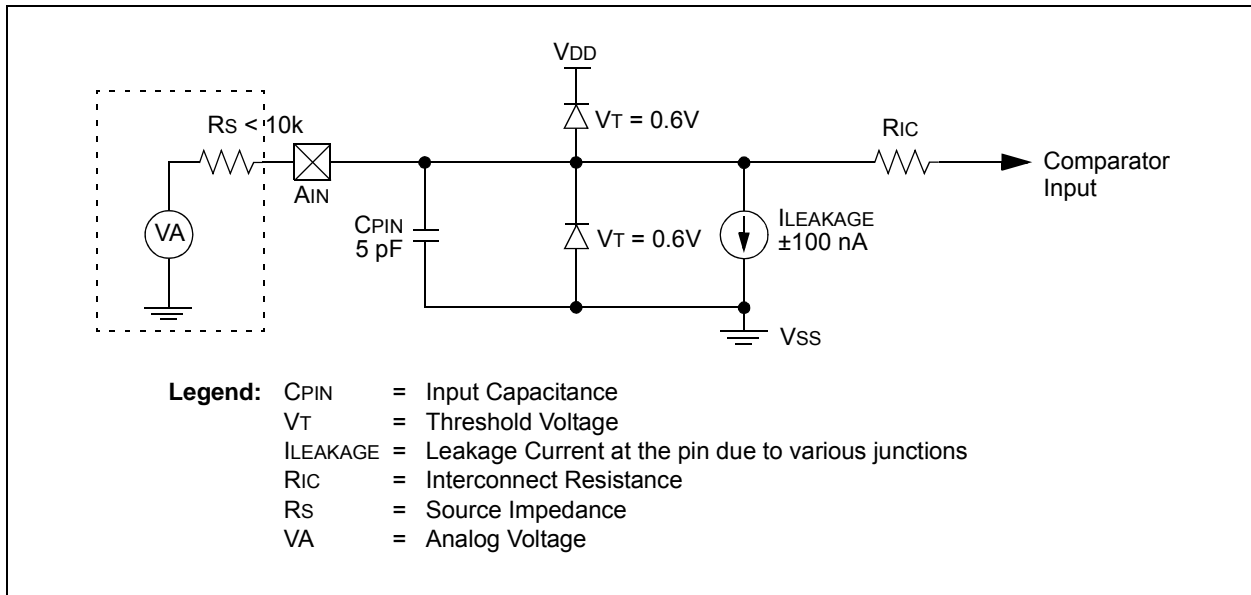
A device Reset forces the CMCON register to its Reset state, causing the comparator module to be in the Comparator Reset mode ( $CM\langle 2:0 \rangle = 000$ ). This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at Reset time. The comparators are powered down during the Reset interval.

## 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	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	65
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	65
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	63
PIR2	OSCFIF	CMIF	—	—	BCLIF	HLVDIF	TMR3IF	CCP2IF	65
PIE2	OCSFIE	CMIE	—	—	BCLIE	HLVDIE	TMR3IE	CCP2IE	65
IPR2	OSCFIP	CMIP	—	—	BCLIP	HLVDIP	TMR3IP	CCP2IP	65
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	66
LATF	LATF Output Latch Register								66
TRISF	PORTF Data Direction Register								66

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

# PIC18F6310/6410/8310/8410

---

NOTES:

## 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 of the module is 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:

$$CVREF = ((CVR<3:0>)/24) \times CVRSRC$$

If CVRR = 0:

$$CVREF = (CVDD \times 1/4) + (((CVR<3:0>)/32) \times CVRSRC)$$

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 27-3](#) in [Section 27.0 "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 <sup>(1)</sup>	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0
bit 7							bit 0

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented bit, read as '0'  
'0' = Bit is cleared

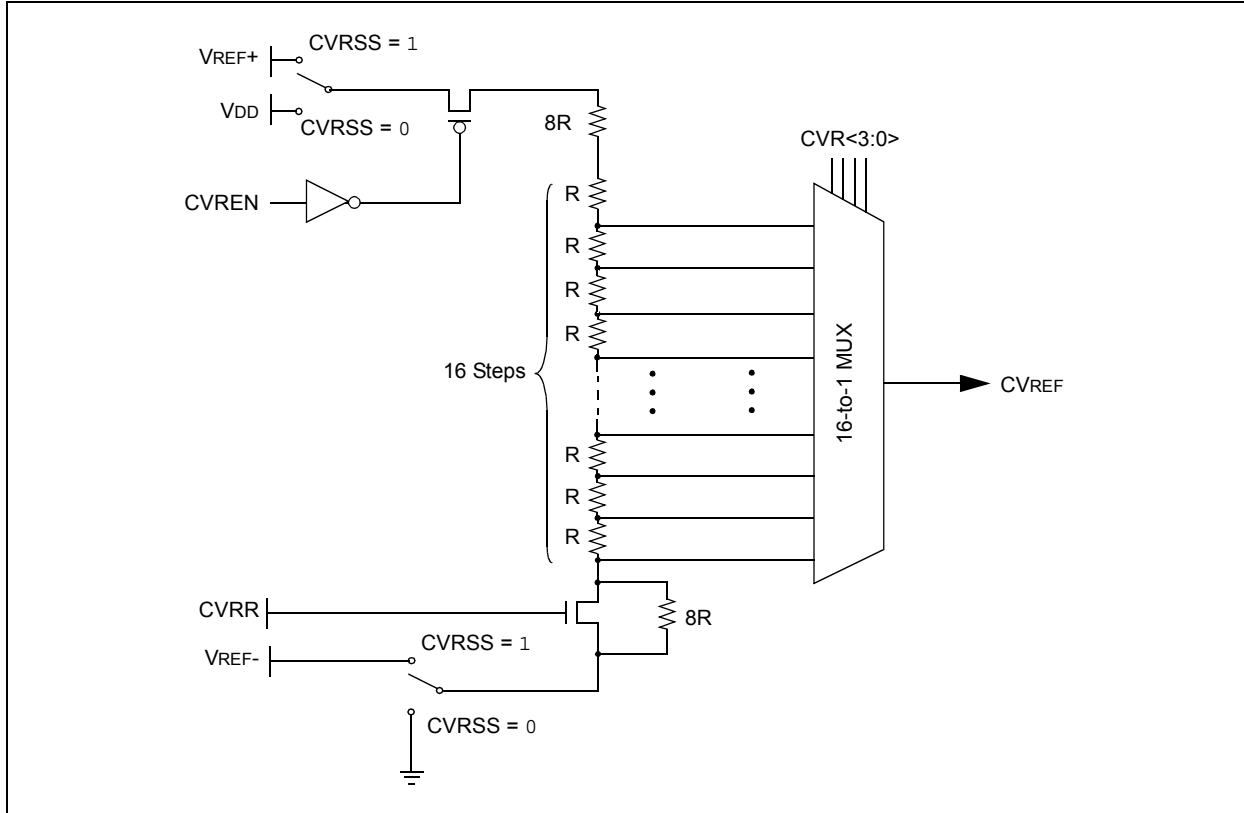
x = Bit is unknown

- 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<sup>(1)</sup>  
1 = CVREF voltage level is also output on the RF5/AN10/CVREF pin  
0 = CVREF voltage is disconnected from the RF5/AN10/CVREF pin
- bit 5 **CVRR:** Comparator VREF Range Selection bit  
1 = 0 CVRSRC to 0.667 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 ≤ (CVR<3:0>) ≤ 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 TRISF<5> bit setting if enabled for output; RF5 must also be configured as an input by setting TRISF<5> to '1'.

# PIC18F6310/6410/8310/8410

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 27.0 “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 RA2 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

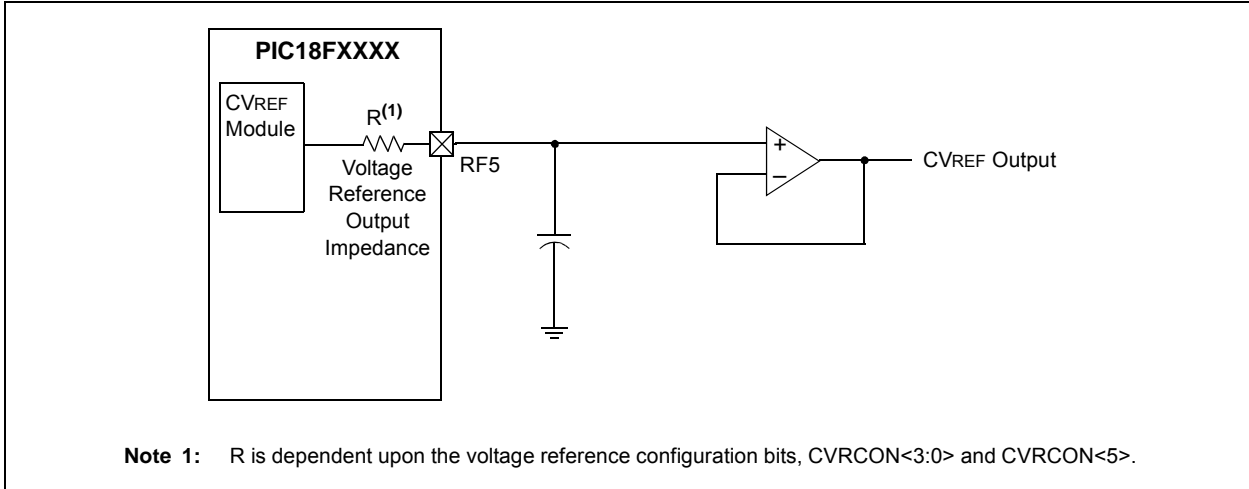
The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RF5 pin if the TRISF<5> bit and the CVROE bit are both set. Enabling the voltage reference output onto the RF5 pin, with an input signal present, will increase current consumption. Connecting RF5 as a digital output with CVRSS enabled will also increase current consumption.

The RF5 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.



# PIC18F6310/6410/8310/8410

**FIGURE 22-2: COMPARATOR VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE**



**TABLE 22-1: REGISTERS ASSOCIATED WITH THE 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	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	65
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	65
TRISF	PORTF Data Direction Register								66

**Legend:** Shaded cells are not used with the comparator voltage reference.

# PIC18F6310/6410/8310/8410

---

NOTES:

# PIC18F6310/6410/8310/8410

## 23.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

PIC18F6310/6410/8310/8410 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 <sup>(1)</sup>	HLVDL2 <sup>(1)</sup>	HLVDL1 <sup>(1)</sup>	HLVDL0 <sup>(1)</sup>
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 is enabled  
           0 = HLVD is disabled
- bit 3-0    **HLVDL<3:0>:** Voltage Detection Limit bits<sup>(1)</sup>  
           1110 = Maximum setting  
           •  
           •  
           •  
           0001 = Minimum setting

**Note 1:** HLVDL<3:0> modes that result in a trip point, below the valid operating voltage of the device, are not tested.

# PIC18F6310/6410/8310/8410

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

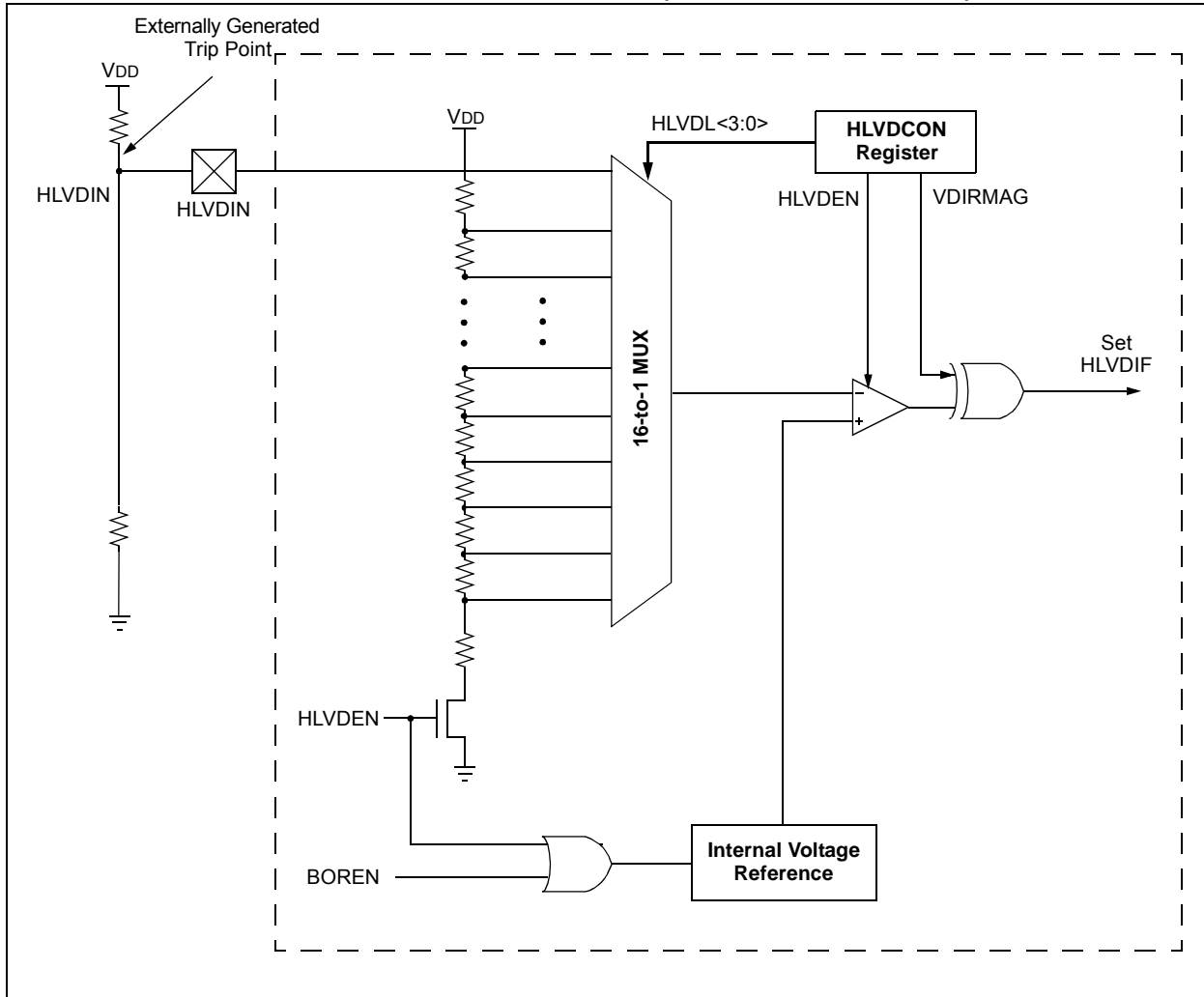
When the HLVD module is enabled, a comparator uses an internally generated reference voltage as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The “trip point” voltage is the voltage

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)**



## 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.

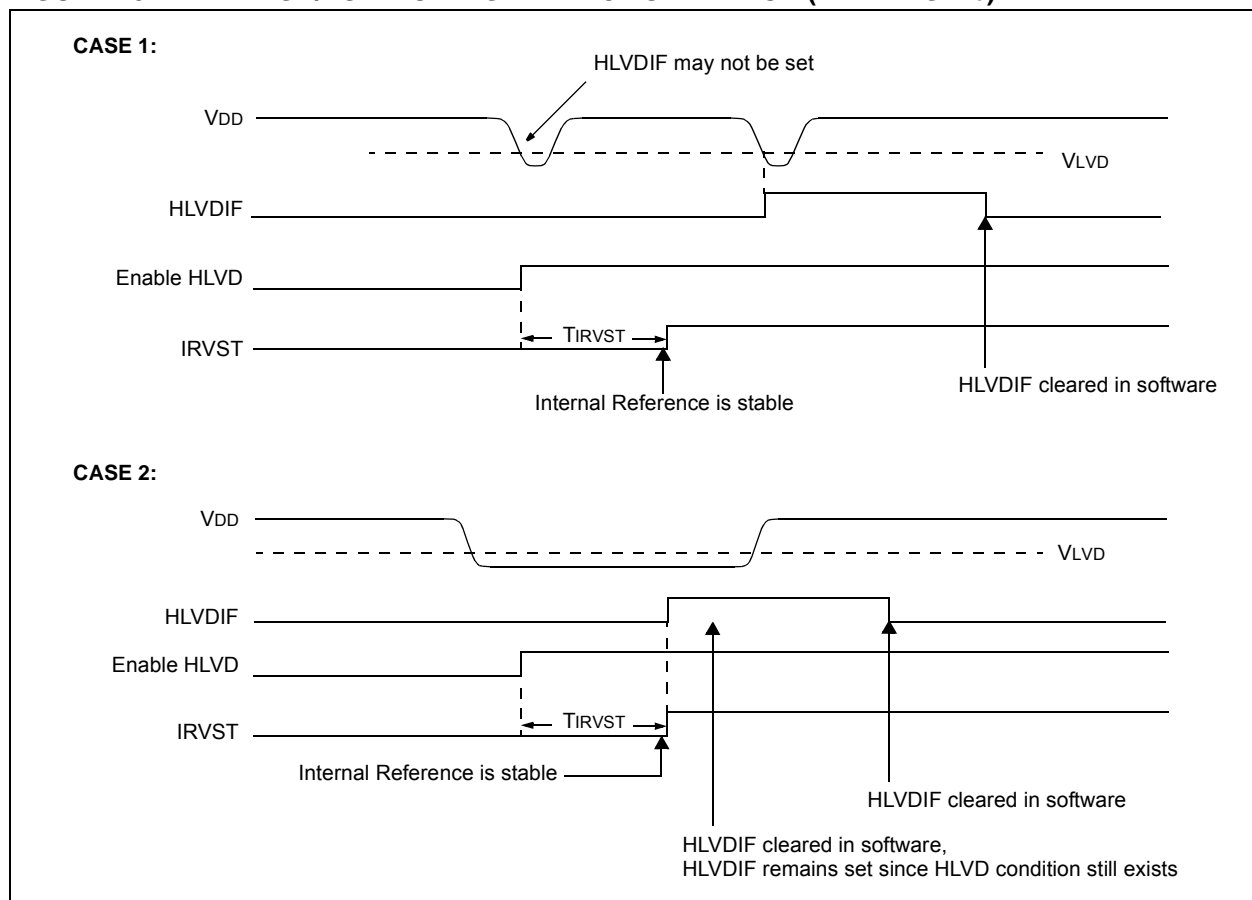
Depending on the application, the HLVD module does not need to be operating constantly. To decrease the current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After doing the check, the HLVD module may be disabled.

## 23.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification Parameter D420B, 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, T<sub>IRVST</sub>, is an interval that is independent of device clock speed. It is specified in electrical specification Parameter 36 (Table 27-12).

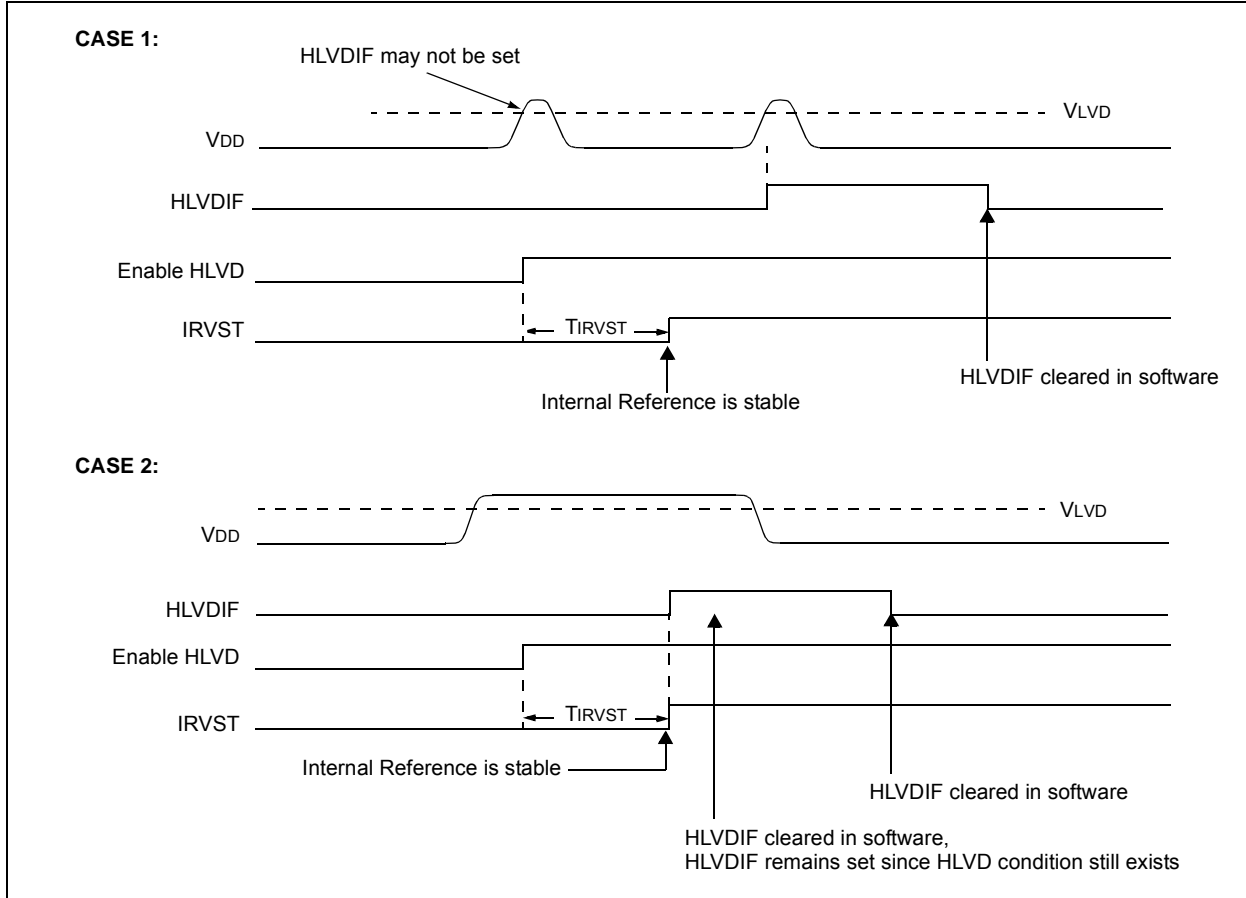
The HLVD interrupt flag is not enabled until T<sub>IRVST</sub> has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval. Refer to Figure 23-2 or Figure 23-3.

**FIGURE 23-2: HIGH/LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 0)**



# PIC18F6310/6410/8310/8410

**FIGURE 23-3: HIGH/LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 1)**

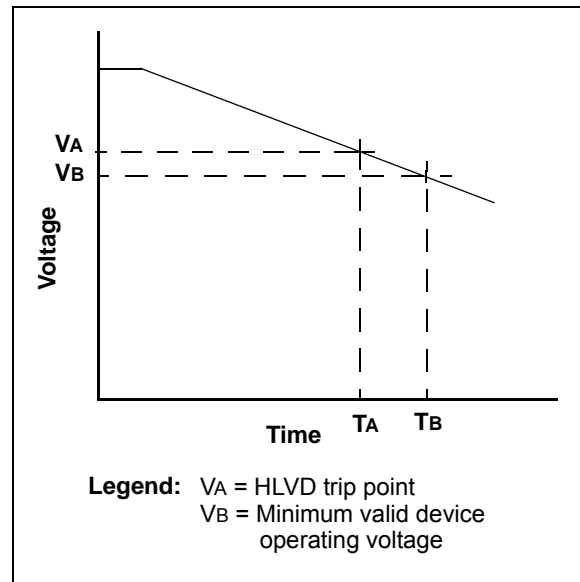


## 23.5 Applications

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 USB attach or detach. This assumes the device is powered by a lower voltage source than the Universal Serial Bus (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, V<sub>A</sub>, the HLVD logic generates an interrupt at time, T<sub>A</sub>. The interrupt could cause the execution of an ISR, which would allow the application to perform “house-keeping tasks” and perform a controlled shutdown before the device voltage exits the valid operating range at T<sub>B</sub>. The HLVD thus, would give the application a time window, represented by the difference between T<sub>A</sub> and T<sub>B</sub>, to safely exit.

**FIGURE 23-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION**



# PIC18F6310/6410/8310/8410

## 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	64
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	63
PIR2	OSCFIF	CMIF	—	—	BCLIF	HLVDIF	TMR3IF	CCP2IF	65
PIE2	OCSFIE	CMIE	—	—	BCLIE	HLVDIE	TMR3IE	CCP2IE	65
IPR2	OSCFIP	CMIP	—	—	BCLIP	HLVDIP	TMR3IP	CCP2IP	65

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the HLVD module.

# PIC18F6310/6410/8310/8410

---

NOTES:



# PIC18F6310/6410/8310/8410

## 24.0 SPECIAL FEATURES OF THE CPU

PIC18F6310/6410/8310/8410 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 (ICSP)

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, PIC18F6310/6410/8310/8410 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.

### 24.1 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.

**TABLE 24-1: 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	PWRTEN	---1 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	---1 1111
300004h	CONFIG3L	WAIT	BW	—	—	—	—	PM1	PM0	11-- --11
300005h	CONFIG3H	MCLRE	—	—	—	—	LPT1OSC	—	CCP2MX	1--- -0-1
300006h	CONFIG4L	DEBUG	XINST	—	—	—	—	—	STVREN	10-- ---1
300008h	CONFIG5L	—	—	—	—	—	—	—	CP	---- ---1
30000Ch	CONFIG7L <sup>(1)</sup>	—	—	—	—	—	—	—	EBTR	---- ---1
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	11qx xxxx <sup>(2)</sup>
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 qq1q <sup>(2)</sup>

**Legend:** x = unknown, u = unchanged, - = unimplemented, q = value depends on individual device. Shaded cells are unimplemented, read as ‘0’.

**Note 1:** Unimplemented in PIC18F6310/6410 devices; maintain this bit set.

**Note 2:** See [Register 24-9](#) for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

# PIC18F6310/6410/8310/8410

## REGISTER 24-1: 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 at erase bit                '1' = Bit is set                                '0' = Bit is cleared                                x = Bit is unknown

- bit 7                      **IESO:** Internal/External Oscillator Switchover bit  
                                     1 = Oscillator Switchover mode is enabled  
                                     0 = Oscillator Switchover mode is disabled
- bit 6                      **FCMEN:** Fail-Safe Clock Monitor Enable bit  
                                     1 = Fail-Safe Clock Monitor is enabled  
                                     0 = Fail-Safe Clock Monitor is 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 is enabled (Clock Frequency = 4 x FOSC1)  
                                     0101 = EC oscillator, CLKO 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

# PIC18F6310/6410/8310/8410

## REGISTER 24-2: 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 <sup>(1)</sup>	BOREN0 <sup>(1)</sup>	$\overline{\text{PWRTEN}}^{\text{(1)}}$
bit 7							bit 0

### Legend:

R = Readable bit                      P = Programmable bit                      U = Unimplemented bit, read as '0'  
 -n = Value at erase bit                      '1' = Bit is set                      '0' = Bit is cleared                      x = Bit is unknown

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

bit 4-3                      **BORV<1:0>:** Brown-out Reset Voltage bits

11 = Minimum setting

•  
•  
•

00 = Maximum setting

bit 2-1                      **BOREN<1:0>** Brown-out Reset Enable bits<sup>(1)</sup>

11 = Brown-out Reset is enabled in hardware only (SBOREN is disabled)

10 = Brown-out Reset is enabled in hardware only and disabled in Sleep mode (SBOREN is disabled)

01 = Brown-out Reset is enabled and controlled by software (SBOREN is enabled)

00 = Brown-out Reset is disabled in hardware and software

bit 0                      **PWRTEN:** Power-up Timer Enable bit<sup>(1)</sup>

1 = PWRT is disabled

0 = PWRT is enabled

**Note 1:** The Power-up Timer (PWRT) is decoupled from Brown-out Reset, allowing these features to be independently controlled.

# PIC18F6310/6410/8310/8410

## REGISTER 24-3: 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 at erase bit                      '1' = Bit is set                      '0' = Bit is cleared                      x = Bit is unknown

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 is enabled
- 0 = WDT is disabled (control is placed on the SWDTEN bit)

# PIC18F6310/6410/8310/8410

## REGISTER 24-4: CONFIG3L: CONFIGURATION REGISTER 3 LOW (BYTE ADDRESS 300004h)

R/P-1	R/P-1	U-0	U-0	U-0	U-0	R/P-1	R/P-1
WAIT	BW	—	—	—	—	PM1	PM0
bit 7						bit 0	

### Legend:

R = Readable bit                      P = Programmable bit                      U = Unimplemented bit, read as '0'  
 -n = Value at erase bit                '1' = Bit is set                                '0' = Bit is cleared                                x = Bit is unknown

- bit 7                      **WAIT:** External Bus Data Wait Enable bit  
                                     1 = Wait selections are unavailable, device will not wait  
                                     0 = Wait is programmed by the WAIT1 and WAIT0 bits of the MEMCOM register (MEMCOM<5:4>)
- bit 6                      **BW:** External Bus Data Width Select bit  
                                     1 = 16-bit external bus data width  
                                     0 = 8-bit external bus data width
- bit 5-2                    **Unimplemented:** Read as '0'
- bit 1-0                    **PM<1:0>:** Processor Data Memory Mode Select bits  
                                     11 = Microcontroller mode  
                                     10 = Microprocessor mode<sup>(1)</sup>  
                                     01 = Microcontroller with Boot Block mode<sup>(1)</sup>  
                                     00 = Extended Microcontroller mode<sup>(1)</sup>

**Note 1:** This mode is only available on PIC18F8310/8410 devices.

# PIC18F6310/6410/8310/8410

## REGISTER 24-5: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

R/P-1	U-0	U-0	U-0	U-0	R/P-0	U-0	R/P-1
MCLRE	—	—	—	—	LPT1OSC	—	CCP2MX
bit 7							bit 0

### Legend:

R = Readable bit                      P = Programmable bit                      U = Unimplemented bit, read as '0'  
 -n = Value at erase bit                '1' = Bit is set                                '0' = Bit is cleared                                x = Bit is unknown

- bit 7                      **MCLRE:**  $\overline{\text{MCLR}}$  Pin Enable bit  
                                  1 =  $\overline{\text{MCLR}}$  pin is enabled; RG5 input pin is disabled  
                                  0 = RG5 input pin is enabled;  $\overline{\text{MCLR}}$  is disabled
- bit 6-3                    **Unimplemented:** Read as '0'
- bit 2                      **LPT1OSC:** Low-Power Timer 1 Oscillator Enable bit  
                                  1 = Timer1 is configured for low-power operation  
                                  0 = Timer1 is configured for higher power operation
- bit 1                      **Unimplemented:** Read as '0'
- bit 0                      **CCP2MX:** CCP2 MUX bit  
                                  In Microcontroller Mode only (all devices):  
                                  1 = CCP2 input/output is multiplexed with RC1  
                                  0 = CCP2 input/output is multiplexed with RE7  
                                  In Microprocessor, Extended Microcontroller and Microcontroller with Boot Block Modes (PIC18F8310/8410 devices only):  
                                  1 = CCP2 input/output is multiplexed with RC1  
                                  0 = CCP2 input/output is multiplexed with RB3

# PIC18F6310/6410/8310/8410

## REGISTER 24-6: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/P-1	R/P-0	U-0	U-0	U-0	U-0	U-0	R/P-1
<u>DEBUG</u>	XINST	—	—	—	—	—	STVREN
bit 7							bit 0

### Legend:

R = Readable bit                      C = Clearable bit                      U = Unimplemented bit, read as '0'  
 -n = Value when device is unprogrammed bit                      u = Unchanged from programmed state

- bit 7                      **DEBUG:** Background Debugger Enable bit  
 1 = Background debugger is disabled, RB6 and RB7 are configured as general purpose I/O pins  
 0 = Background debugger is 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 are enabled  
 0 = Instruction set extension and Indexed Addressing mode are disabled (Legacy mode)
- bit 5-1                      **Unimplemented:** Read as '0'
- bit 0                      **STVREN:** Stack Full/Underflow Reset Enable bit  
 1 = Stack full/underflow will cause a Reset  
 0 = Stack full/underflow will not cause a Reset

## REGISTER 24-7: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/C-1
—	—	—	—	—	—	—	CP
bit 7							bit 0

### Legend:

R = Readable bit                      C = Clearable bit                      U = Unimplemented bit, read as '0'  
 -n = Value when device is unprogrammed bit                      u = Unchanged from programmed state

- bit 7-1                      **Unimplemented:** Read as '0'
- bit 0                      **CP:** Code Protection bit  
 1 = Program memory block is not code-protected  
 0 = Program memory block is code-protected

# PIC18F6310/6410/8310/8410

## REGISTER 24-8: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 3000Ch)<sup>(1)</sup>

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/C-1
—	—	—	—	—	—	—	EBTR <sup>(2,3)</sup>
bit 7							bit 0

### Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed bit

u = Unchanged from programmed state

bit 7-1      **Unimplemented:** Read as '0'

bit 0      **EBTR:** Table Read Protection bit<sup>(2,3)</sup>

1= Internal program memory block is not protected from table reads executed from external memory block

0= Internal program memory block is protected from table reads executed from external memory block

**Note 1:** Unimplemented on PIC18F6310/6410 devices; maintain the bit set.

**2:** Valid for the entire internal program memory block in Extended Microcontroller mode and for only the boot block (0000h to 07FFh) in Microcontroller with Boot Block mode. This bit has no effect in Microcontroller and Microprocessor modes.

**3:** It is recommended to enable the CP bit to protect the block from external read operations.



# PIC18F6310/6410/8310/8410

## REGISTER 24-9: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F6310/6410/8310/8410 DEVICES

R	R	R	R	R	R	R	R
DEV2 <sup>(1)</sup>	DEV1 <sup>(1)</sup>	DEV0 <sup>(1)</sup>	REV4	REV3	REV2	REV1	REV0
bit 7							bit 0

### Legend:

R = Readable bit                      P = Programmable bit                      U = Unimplemented bit, read as '0'  
 -n = Value at erase bit                '1' = Bit is set                                '0' = Bit is cleared                                x = Bit is unknown

bit 7-5                **DEV<2:0>**: Device ID bits<sup>(1)</sup>  
                           110 = PIC18F8310, PIC18F8410  
                           111 = PIC18F6310, PIC18F6410

bit 4-0                **REV<4:0>**: Revision ID bits  
                           These bits are used to indicate the device revision.

**Note 1:** These values for DEV<2:0> may be shared with other devices. The specific device is always identified by using the entire DEV<10:0> bit sequence.

## REGISTER 24-10: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F6310/6410/8310/8410 DEVICES

R	R	R	R	R	R	R	R
DEV10 <sup>(1)</sup>	DEV9 <sup>(1)</sup>	DEV8 <sup>(1)</sup>	DEV7 <sup>(1)</sup>	DEV6 <sup>(1)</sup>	DEV5 <sup>(1)</sup>	DEV4 <sup>(1)</sup>	DEV3 <sup>(1)</sup>
bit 7							bit 0

### Legend:

R = Readable bit                      P = Programmable bit                      U = Unimplemented bit, read as '0'  
 -n = Value at erase bit                '1' = Bit is set                                '0' = Bit is cleared                                x = Bit is unknown

bit 7-0                **DEV<10:3>**: Device ID bits  
                           These bits are used with the DEV<2:0> bits in the Device ID Register 1 to identify the part number.  
                           0000 0110 = PIC18F6410/8410 devices  
                           0000 1011 = PIC18F6310/8310 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.



# PIC18F6310/6410/8310/8410

## REGISTER 24-11: WDTCON: WATCHDOG TIMER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN <sup>(1)</sup>
bit 7							bit 0

### Legend:

R = Readable bit                      P = Programmable bit                      U = Unimplemented bit, read as '0'  
 -n = Value at erase bit                      '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<sup>(1)</sup>  
 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 24-2: 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	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	64
WDTCON	—	—	—	—	—	—	—	SWDTEN	64

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Watchdog Timer.

# PIC18F6310/6410/8310/8410

## 24.3 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 a OST start-up 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.

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.

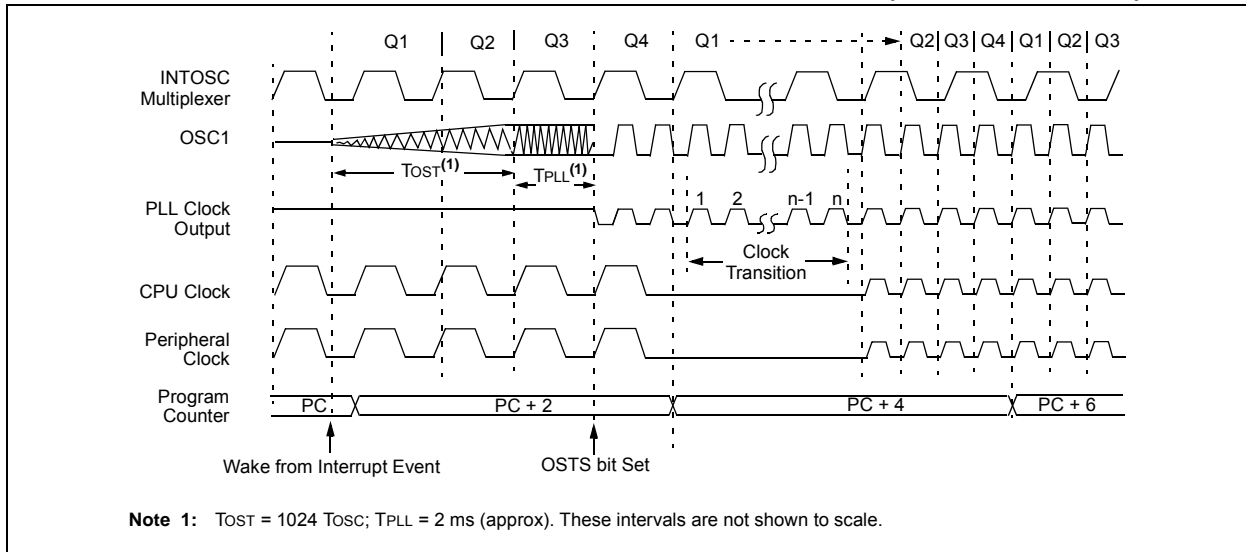
In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

### 24.3.1 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 4.1.2 “Entering Power-Managed Modes”](#)). In practice, this means that user code can change the SCS<1:0> bits setting 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 OST 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 24-2: TIMING TRANSITION FOR TWO-SPEED START-UP (INTOSC TO HSPLL)**

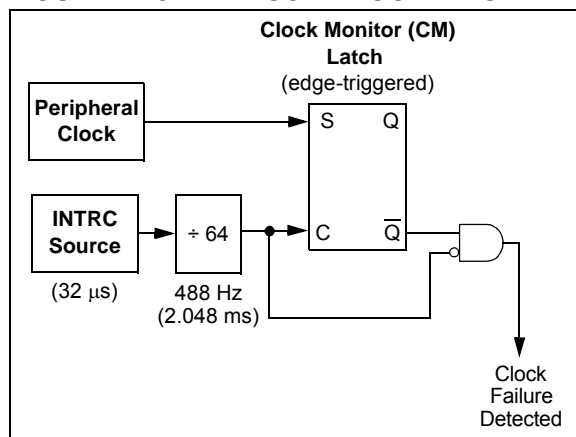


## 24.4 Fail-Safe Clock Monitor

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 24-3) 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 latch (CM). The CM is set on the falling edge of the device clock source, but cleared on the rising edge of the sample clock.

**FIGURE 24-3: 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 24-4). 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 shut-down. See Section 4.1.2 “Entering Power-Managed Modes” and Section 24.3.1 “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.

### 24.4.1 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.

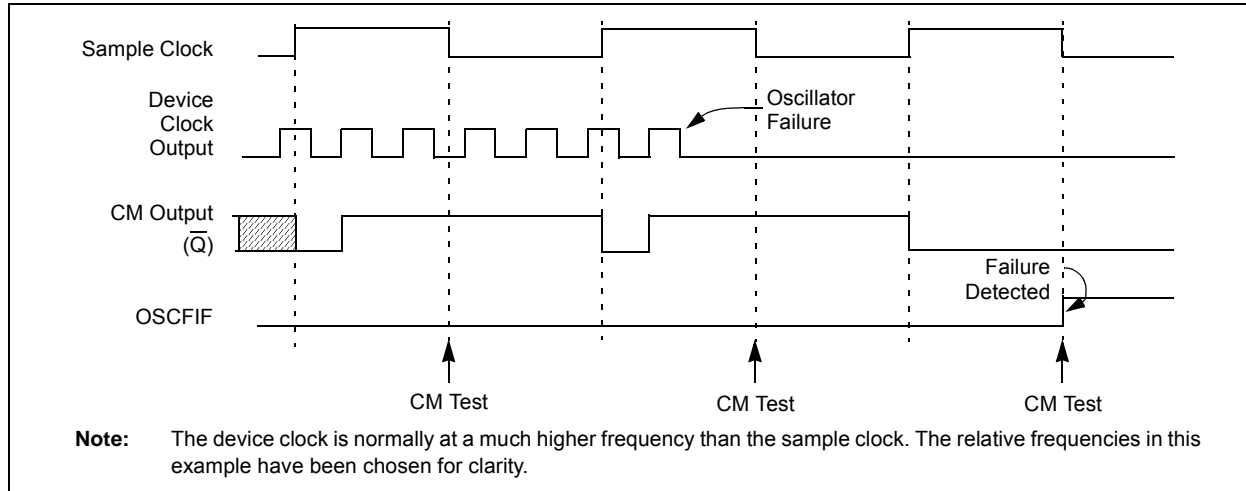
### 24.4.2 EXITING FAIL-SAFE OPERATION

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 the 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.

# PIC18F6310/6410/8310/8410

FIGURE 24-4: FSCM TIMING DIAGRAM



## 24.4.3 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, the device will not exit the power-managed mode on oscillator failure. Instead, the device will continue to operate as before, but clocked by the INTOSC multiplexer. While in Idle mode, subsequent interrupts will cause the CPU to begin executing instructions while being clocked by the INTOSC multiplexer.

## 24.4.4 POR OR WAKE FROM SLEEP

The FSCM is designed to detect oscillator failure at any point after the device has exited Power-on Reset (POR) or low-power Sleep mode. When the primary device clock is in 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 FSCM 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 24.3.1 "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 powered-managed mode is selected, the primary clock is disabled.

# PIC18F6310/6410/8310/8410

## 24.5 Program Verification and Code Protection

The overall structure of the code protection on the PIC18F6310/6410/8310/8410 Flash devices differs from previous PIC18 devices.

For all devices in the PIC18FX310/X410 family, the user program memory is made of a single block. Figure 24-5 shows the program memory organization for individual devices. Code protection for this block is controlled by a single bit, CP (CONFIG5L<0>). The CP bit inhibits external reads and writes; it has no direct effect in normal execution mode.

### 24.5.1 CODE PROTECTION FROM EXTERNAL TABLE READS

The program memory may be read to any location using the table read instructions. The Device ID and the Configuration registers may be read with the table read instructions.

For devices with the external memory interface, it is possible to execute a table read from an external program memory space and read the contents of the on-chip memory. An additional code protection bit,

EBTR (CONFIG7L<0>), is used to protect the on-chip program memory space from this possibility. Setting EBTR prevents table read commands from executing on any address in the on-chip program memory space.

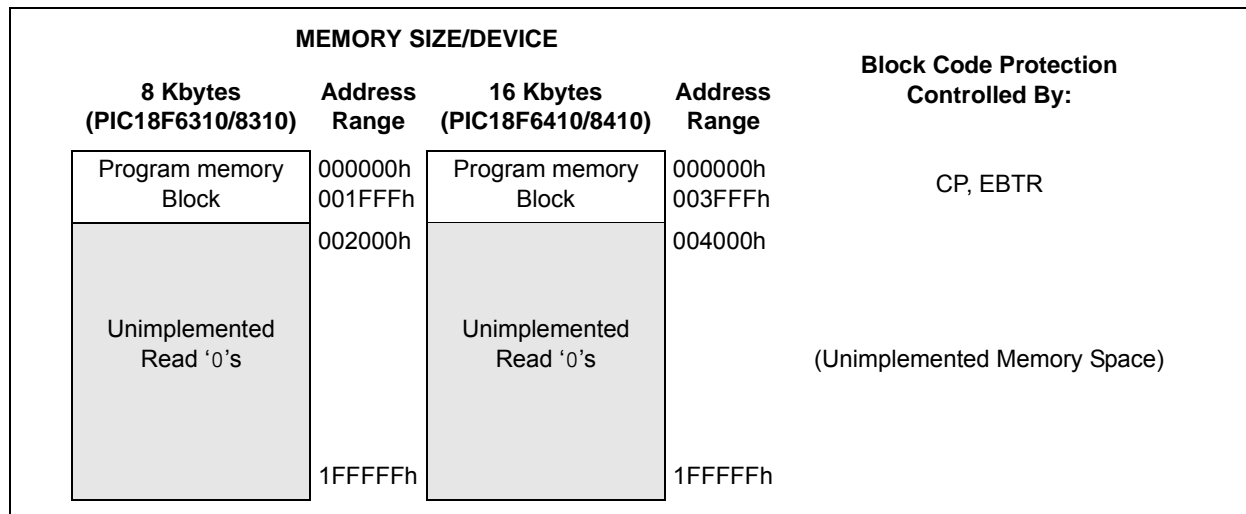
EBTR is implemented only on devices with the external memory interface. Its operation also depends on the particular mode of operation selected. In Extended Microcontroller mode, programming EBTR enables protection from external table reads for the entire program memory. In Microcontroller with Boot Block mode, only the first 2 Kbytes of on-chip memory (000h to 7FFh) are protected. This is because, only this range of internal program memory is accessible by the microcontroller in this operating mode.

When the device is in Microcontroller or Microprocessor modes, EBTR has no effect on code protection.

### 24.5.2 CONFIGURATION REGISTER PROTECTION

The Configuration registers can only be written via ICSP using an external programmer. No separate protection bit is associated with them.

**FIGURE 24-5: CODE-PROTECTED PROGRAM MEMORY FOR PIC18F6310/6410/8310/8410**



**TABLE 24-3: 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	—	—	—	—	—	—	CP
30000Ch	CONFIG7L*	—	—	—	—	—	—	EBTR

**Legend:** Shaded cells are unimplemented.

\* Unimplemented in PIC18F6310/8310 devices; maintain this bit set.

# PIC18F6310/6410/8310/8410

---

## 24.6 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 readable during normal execution through the `TBLRD` instruction. During program/verify, these locations are readable and writable. The ID locations can be read when the device is code-protected.

## 24.7 In-Circuit Serial Programming

PIC18F6310/6410/8310/8410 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.

## 24.8 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 24-4 shows which resources are required by the background debugger.

**TABLE 24-4: DEBUGGER RESOURCES**

I/O Pins:	RB6, RB7
Stack:	2 levels
Program Memory:	<1 Kbyte
Data Memory:	<16 bytes

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to `MCLR/VPP`, `VDD`, `VSS`, `RB7` and `RB6`. This will interface to the In-Circuit Debugger module available from Microchip or one of the third party development tool companies.



## 25.0 INSTRUCTION SET SUMMARY

PIC18F6310/6410/8310/8410 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.

### 25.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC<sup>®</sup> device instruction sets, while maintaining an easy migration from these PIC device 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 25-2](#) lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. [Table 25-1](#) shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The destination of the result (specified by 'd')
3. The accessed memory (specified by 'a')

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

All **bit-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The bit in the file register (specified by 'b')
3. The accessed memory (specified by 'a')

The bit field designator, 'b', selects the number of the bit affected by the operation, while the file register designator, 'f', represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by 'k')
- The desired FSR register to load the literal value into (specified by 'f')
- No operand required (specified by '—')

The **control** instructions may use some of the following operands:

- A program memory address (specified by 'n')
- The mode of the call or return instructions (specified by 's')
- The mode of the table read and table write instructions (specified by 'm')
- No operand required (specified by '—')

All instructions are a single word, except for 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  $\mu$ s. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2  $\mu$ s. Two-word branch instructions (if true) would take 3  $\mu$ s.

[Figure 25-1](#) 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 25-2](#), lists the standard instructions recognized by the Microchip Assembler (MPASM<sup>™</sup>).

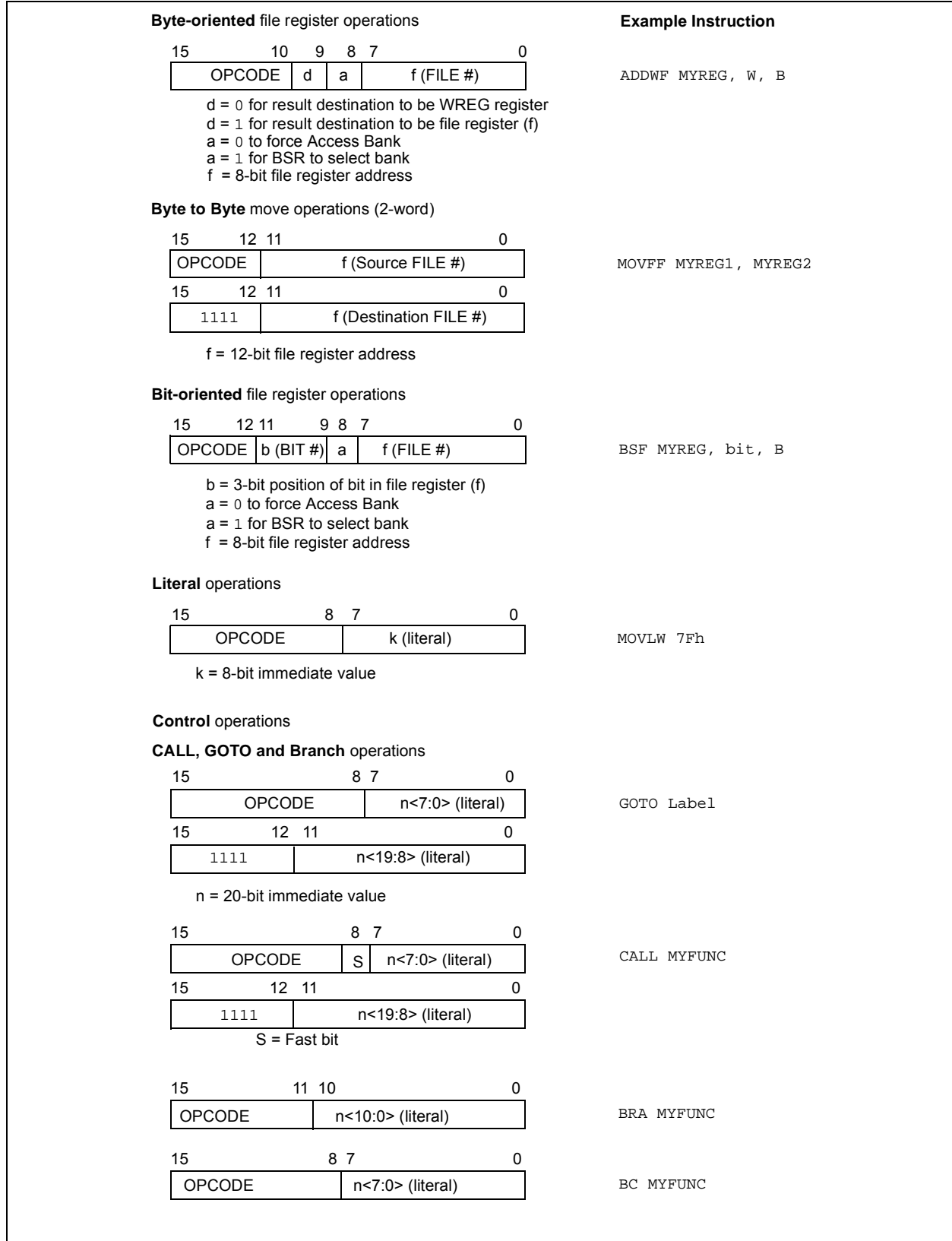
[Section 25.1.1 "Standard Instruction Set"](#) provides a description of each instruction.

# PIC18F6310/6410/8310/8410

**TABLE 25-1: OPCODE FIELD DESCRIPTIONS**

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) 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: <b>C</b> arry, <b>D</b> igit <b>C</b> arry, <b>Z</b> ero, <b>O</b> verflow, <b>N</b> egative.
d	Destination select bit d = 0: store result in WREG 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 <sub>s</sub>	12-bit register file address (000h to FFFh). This is the source address.
f <sub>d</sub>	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	The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions:
*	No change to register (such as TBLPTR with table reads and writes).
*+	Post-Increment register (such as TBLPTR with table reads and writes).
*-	Post-Decrement register (such as TBLPTR with table reads and writes).
++	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 s = 0: do not update into/from shadow registers 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 <sub>s</sub>	7-bit offset value for indirect addressing of register files (source).
z <sub>d</sub>	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.
<i>italics</i>	User-defined term (font is Courier New).

**FIGURE 25-1: GENERAL FORMAT FOR INSTRUCTIONS**



# PIC18F6310/6410/8310/8410

**TABLE 25-2: PIC18FXXXX INSTRUCTION SET**

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb			LSb			
<b>BYTE-ORIENTED OPERATIONS</b>									
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 <sub>s</sub> , f <sub>d</sub>	Move f <sub>s</sub> (source) to f <sub>d</sub> (destination)	2	1100	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:** Table write instructions are unavailable in 64-pin devices in normal operating modes. See [Section 7.4 "Writing to Program Memory Space \(PIC18F8310/8410 only\)"](#) and [Section 7.6 "Writing and Erasing On-Chip Program Memory \(ICSP Mode\)"](#) for more information.

# PIC18F6310/6410/8310/8410

**TABLE 25-2: PIC18FXXX INSTRUCTION SET (CONTINUED)**

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb	LSb					
<b>BIT-ORIENTED OPERATIONS</b>									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
<b>CONTROL OPERATIONS</b>									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	4
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 2nd word	2	1110	110s	kkkk	kkkk	None	
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 2nd word	2	1110	1111	kkkk	kkkk	None	
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	
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., `MOVWF 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:** Table write instructions are unavailable in 64-pin devices in normal operating modes. See [Section 7.4 "Writing to Program Memory Space \(PIC18F8310/8410 only\)"](#) and [Section 7.6 "Writing and Erasing On-Chip Program Memory \(ICSP Mode\)"](#) for more information.

# PIC18F6310/6410/8310/8410

**TABLE 25-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)**

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb			LSb			
<b>LITERAL OPERATIONS</b>									
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	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	
<b>DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS</b>									
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:** Table write instructions are unavailable in 64-pin devices in normal operating modes. See [Section 7.4 "Writing to Program Memory Space \(PIC18F8310/8410 only\)"](#) and [Section 7.6 "Writing and Erasing On-Chip Program Memory \(ICSP Mode\)"](#) for more information.

**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)

# PIC18F6310/6410/8310/8410

## 25.1.1 STANDARD INSTRUCTION SET

### ADDLW ADD literal to W

Syntax: ADDLW k

Operands:  $0 \leq k \leq 255$

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:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(W) + (f) \rightarrow \text{dest}$

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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## ADDWFC      ADD W and Carry bit to f

**Syntax:**            ADDWFC    f {,d {,a}}

**Operands:**         $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**         $(W) + (f) + (C) \rightarrow \text{dest}$

**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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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:**         $0 \leq k \leq 255$

**Operation:**         $(W) .\text{AND} .k \rightarrow W$

**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



# PIC18F6310/6410/8310/8410

ANDWF	AND W with f								
Syntax:	ANDWF f {,d {,a}}								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	(W) .AND. (f) → dest								
Status Affected:	N, Z								
Encoding:	<table border="1"> <tr> <td>0001</td> <td>01da</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0001	01da	ffff	ffff				
0001	01da	ffff	ffff						
Description:	<p>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'.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <a href="#">Section 25.2.3</a> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
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:	$-128 \leq n \leq 127$																				
Operation:	if Carry bit is '1', $(PC) + 2 + 2n \rightarrow PC$																				
Status Affected:	None																				
Encoding:	<table border="1"> <tr> <td>1110</td> <td>0010</td> <td>nnnn</td> <td>nnnn</td> </tr> </table>	1110	0010	nnnn	nnnn																
1110	0010	nnnn	nnnn																		
Description:	<p>If the Carry bit is '1', then the program will branch.</p> <p>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 <math>PC + 2 + 2n</math>. This instruction is then a two-cycle instruction.</p>																				
Words:	1																				
Cycles:	1(2)																				
Q Cycle Activity:	<p>If Jump:</p> <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>Write to PC</td> </tr> <tr> <td>No operation</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </tbody> </table> <p>If No Jump:</p> <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>No operation</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation
Q1	Q2	Q3	Q4																		
Decode	Read literal 'n'	Process Data	Write to PC																		
No operation	No operation	No operation	No operation																		
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)

# PIC18F6310/6410/8310/8410

## BCF Bit Clear f

**Syntax:** BCF f, b {,a}

**Operands:**  $0 \leq f \leq 255$   
 $0 \leq b \leq 7$   
 $a \in [0,1]$

**Operation:**  $0 \rightarrow f \leftarrow b$

**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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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:**  $-128 \leq n \leq 127$

**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)

# PIC18F6310/6410/8310/8410

## BNC Branch if Not Carry

**Syntax:** BNC n

**Operands:**  $-128 \leq n \leq 127$

**Operation:** if Carry bit is '0',  
 $(PC) + 2 + 2n \rightarrow PC$

**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:**  $-128 \leq n \leq 127$

**Operation:** if Negative bit is '0',  
 $(PC) + 2 + 2n \rightarrow PC$

**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)

# PIC18F6310/6410/8310/8410

## BNOV Branch if Not Overflow

Syntax: BNOV n

Operands:  $-128 \leq n \leq 127$

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:  $-128 \leq n \leq 127$

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)

# PIC18F6310/6410/8310/8410

## BRA Unconditional Branch

Syntax: BRA n  
 Operands:  $-1024 \leq n \leq 1023$   
 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:  $0 \leq f \leq 255$   
 $0 \leq b \leq 7$   
 $a \in [0,1]$   
 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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## BTFSK Bit Test File, Skip if Clear

**Syntax:** BTFSK f, b {,a}

**Operands:**  $0 \leq f \leq 255$   
 $0 \leq b \leq 7$   
 $a \in [0,1]$

**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 '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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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    BTFSK    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)

## BTFSB Bit Test File, Skip if Set

**Syntax:** BTFSB f, b {,a}

**Operands:**  $0 \leq f \leq 255$   
 $0 \leq b < 7$   
 $a \in [0,1]$

**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 '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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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    BTFSB    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)

# PIC18F6310/6410/8310/8410

## BTG

### Bit Toggle f

**Syntax:** BTG f, b {,a}

**Operands:**  $0 \leq f \leq 255$   
 $0 \leq b < 7$   
 $a \in [0,1]$

**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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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:**  $-128 \leq n \leq 127$

**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)

# PIC18F6310/6410/8310/8410

## BZ Branch if Zero

Syntax: BZ n

Operands:  $-128 \leq n \leq 127$

Operation: if Zero bit is '1',  
 $(PC) + 2 + 2n \rightarrow PC$

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:  $0 \leq k \leq 1048575$   
 $s \in [0,1]$

Operation:  $(PC) + 4 \rightarrow TOS$ ,  
 $k \rightarrow PC<20:1>$ ,  
 if  $s = 1$   
 $(W) \rightarrow WS$ ,  
 $(STATUS) \rightarrow STATUSS$ ,  
 $(BSR) \rightarrow BSRS$

Status Affected: None

Encoding: 

1110	110s	k <sub>7</sub> k <sub>6</sub> k <sub>5</sub> k <sub>4</sub>	k <sub>3</sub> k <sub>2</sub> k <sub>1</sub> k <sub>0</sub>
1111	k <sub>19</sub> k <sub>18</sub> k <sub>17</sub> k <sub>16</sub>	k <sub>15</sub> k <sub>14</sub> k <sub>13</sub> k <sub>12</sub>	k <sub>11</sub> k <sub>10</sub> k <sub>9</sub> k <sub>8</sub>

1st word ( $k<7:0>$ )  
 2nd word ( $k<19:8>$ )

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	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



# PIC18F6310/6410/8310/8410

## CLRF Clear f

**Syntax:** CLRF f{,a}

**Operands:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**Operation:** 000h  $\rightarrow$  f,  
 1  $\rightarrow$  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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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  $\rightarrow$  WDT,  
 000h  $\rightarrow$  WDT postscaler,  
 1  $\rightarrow$   $\overline{\text{TO}}$ ,  
 1  $\rightarrow$   $\overline{\text{PD}}$

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

**Encoding:**

0000	0000	0000	0100
------	------	------	------

**Description:** CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits,  $\overline{\text{TO}}$  and  $\overline{\text{PD}}$ , 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  
 $\overline{\text{TO}}$  = 1  
 $\overline{\text{PD}}$  = 1

# PIC18F6310/6410/8310/8410

## COMF Complement f

Syntax: COMF f{,d{,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $\overline{f} \rightarrow \text{dest}$

Status Affected: N, Z

Encoding: 

0001	11da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. 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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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:  $0 \leq f \leq 255$   
 $a \in [0,1]$

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 '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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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  $\neq$  W;  
 PC = Address (NEQUAL)

# PIC18F6310/6410/8310/8410

## CPFSGT Compare f with W, skip if f > W

**Syntax:** CPFSGT f {,a}

**Operands:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**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)

# PIC18F6310/6410/8310/8410

## DAW Decimal Adjust W Register

**Syntax:** DAW

**Operands:** None

**Operation:** If  $[W<3:0> > 9]$  or  $[DC = 1]$  then,  $(W<3:0>) + 6 \rightarrow W<3:0>$ ;  
else,  
 $(W<3:0>) \rightarrow W<3:0>$ ;

If  $[W<7:4> > 9]$  or  $[C = 1]$  then,  
 $(W<7:4>) + 6 \rightarrow W<7:4>$ ;  
 $C = 1$ ;  
else,  
 $(W<7:4>) \rightarrow W<7:4>$

**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:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**  $(f) - 1 \rightarrow \text{dest}$

**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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## DECFSZ Decrement f, skip if 0

**Syntax:** DECFSZ f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**  $(f) - 1 \rightarrow \text{dest}$ ,  
 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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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  $\neq$  0;  
 PC = Address (HERE + 2)

## DCFSNZ Decrement f, skip if not 0

**Syntax:** DCFSNZ f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**  $(f) - 1 \rightarrow \text{dest}$ ,  
 skip if result  $\neq$  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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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  $\neq$  0;  
 PC = Address (NZERO)

# PIC18F6310/6410/8310/8410

## GOTO Unconditional Branch

Syntax: GOTO k

Operands:  $0 \leq k \leq 1048575$

Operation:  $k \rightarrow PC<20:1>$

Status Affected: None

Encoding:

1st word ( $k<7:0>$ )

2nd word ( $k<19:8>$ )

1110	1111	$k_7kkk$	$kkkk_0$
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:  $0 \leq f \leq 255$

$d \in [0,1]$

$a \in [0,1]$

Operation:  $(f) + 1 \rightarrow \text{dest}$

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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## INCFSZ Increment f, skip if 0

**Syntax:** INCFSZ f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**  $(f) + 1 \rightarrow \text{dest}$ ,  
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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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  $\neq$  0;  
PC = Address (NZERO)

## INFSNZ Increment f, skip if not 0

**Syntax:** INFSNZ f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

**Operation:**  $(f) + 1 \rightarrow \text{dest}$ ,  
skip if result  $\neq 0$

**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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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  
                  NZERO   :  
                  NZERO

Before Instruction  
PC = Address (HERE)

After Instruction  
REG = REG + 1  
If REG  $\neq$  0;  
PC = Address (NZERO)  
If REG = 0;  
PC = Address (ZERO)

# PIC18F6310/6410/8310/8410

## IORLW Inclusive OR literal with W

Syntax: IORLW k  
 Operands:  $0 \leq k \leq 255$   
 Operation: (W) .OR. k  $\rightarrow$  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:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$   
 Operation: (W) .OR. (f)  $\rightarrow$  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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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



# PIC18F6310/6410/8310/8410

## LFSR Load FSR

Syntax: LFSR f, k  
 Operands:  $0 \leq f \leq 2$   
 $0 \leq k \leq 4095$   
 Operation:  $k \rightarrow \text{FSRf}$   
 Status Affected: None  
 Encoding: 

1110	1110	00ff	k <sub>11</sub> kkk
1111	0000	k <sub>7</sub> 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:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$   
 Operation:  $f \rightarrow \text{dest}$   
 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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## MOVFF Move f to f

Syntax: MOVFF  $f_s, f_d$

Operands:  $0 \leq f_s \leq 4095$   
 $0 \leq f_d \leq 4095$

Operation:  $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)

2nd word (destin.)

1100	ffff	ffff	ffff <sub>s</sub>
1111	ffff	ffff	ffff <sub>d</sub>

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 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:  $0 \leq k \leq 255$

Operation:  $k \rightarrow \text{BSR}$

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

# PIC18F6310/6410/8310/8410

## MOVLW Move literal to W

Syntax: MOVLW k  
 Operands:  $0 \leq k \leq 255$   
 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:  $0 \leq f \leq 255$   
 $a \in [0,1]$   
 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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## MULLW Multiply literal with W

**Syntax:** MULLW k

**Operands:**  $0 \leq k \leq 255$

**Operation:**  $(W) \times k \rightarrow \text{PRODH}:\text{PRODL}$

**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 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:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**Operation:**  $(W) \times (f) \rightarrow \text{PRODH}:\text{PRODL}$

**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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## NEGF

### Negate f

**Syntax:** NEGF f {,a}

**Operands:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**Operation:**  $(\bar{f}) + 1 \rightarrow f$

**Status Affected:** N, OV, C, DC, Z

**Encoding:**

0110	110a	ffff	ffff
------	------	------	------

**Description:** Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'.  
 If 'a' is '0', the Access Bank 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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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	0000	0000	0000
1111	xxxx	xxxx	xxxx

**Description:** No operation.

**Words:** 1

**Cycles:** 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

**Example:**

None.

# PIC18F6310/6410/8310/8410

## 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

# PIC18F6310/6410/8310/8410

## RCALL

### Relative Call

**Syntax:** RCALL n

**Operands:**  $-1024 \leq n \leq 1023$

**Operation:**  $(PC) + 2 \rightarrow TOS$ ,  
 $(PC) + 2 + 2n \rightarrow PC$

**Status Affected:** None

**Encoding:**

1101	1nnn	nnnn	nnnn
------	------	------	------

**Description:** Subroutine call with a jump up to 1K from the current location. First, return address  $(PC + 2)$  is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be  $PC + 2 + 2n$ . This instruction is a two-cycle instruction.

**Words:** 1

**Cycles:** 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' 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 MCLR Reset.

**Status Affected:** All

**Encoding:**

0000	0000	1111	1111
------	------	------	------

**Description:** This instruction provides a way to execute a 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

# PIC18F6310/6410/8310/8410

## RETFIE Return from Interrupt

Syntax: RETFIE {s}

Operands:  $s \in [0,1]$

Operation: (TOS) → PC,  
 1 → GIE/GIEH or PEIE/GIEL;  
 if  $s = 1$ ,  
 (WS) → W,  
 (STATUS) → STATUS,  
 (BSRS) → BSR,  
 PCLATU, PCLATH are unchanged

Status Affected: GIE/GIEH, PEIE/GIEL.

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, STATUS and BSR, 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 Set GIEH or GIEL
No operation	No operation	No operation	No operation

**Example:** RETFIE 1

After Interrupt

```

PC           = TOS
W            = WS
BSR         = BSR
STATUS      = STATUS
GIE/GIEH, PEIE/GIEL = 1
    
```

## RETLW Return literal to W

Syntax: RETLW k

Operands:  $0 \leq k \leq 255$

Operation:  $k \rightarrow W$ ,  
 (TOS) → PC,  
 PCLATU, PCLATH are unchanged

Status Affected: None

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



# PIC18F6310/6410/8310/8410

## RETURN Return from Subroutine

**Syntax:** RETURN {s}

**Operands:** s ∈ [0,1]

**Operation:** (TOS) → PC;  
if s = 1,  
(WS) → W,  
(STATUS) → 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, STATUS 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	Decode	No operation	Process Data	Pop PC from stack
No operation	No operation	No operation	No operation	No operation

**Example:** RETURN

After Instruction:  
PC = TOS

## RLCF Rotate Left f through Carry

**Syntax:** RLCF f {,d {,a}}

**Operands:** 0 ≤ f ≤ 255  
d ∈ [0,1]  
a ∈ [0,1]

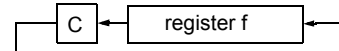
**Operation:** (f<n>) → dest<n + 1>,  
(f<7>) → C,  
(C) → dest<0>

**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 25.2.3](#) for details.



**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

	Q1	Q2	Q3	Q4
Decode	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

# PIC18F6310/6410/8310/8410

## RLNCF Rotate Left f (no carry)

Syntax: RLNCF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(f \langle n \rangle) \rightarrow \text{dest} \langle n + 1 \rangle$ ,  
 $(f \langle 7 \rangle) \rightarrow \text{dest} \langle 0 \rangle$

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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

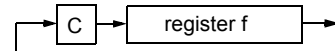
Operation:  $(f \langle n \rangle) \rightarrow \text{dest} \langle n - 1 \rangle$ ,  
 $(f \langle 0 \rangle) \rightarrow C$ ,  
 $(C) \rightarrow \text{dest} \langle 7 \rangle$

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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## RRNCF Rotate Right f (no carry)

**Syntax:** RRNCF f {,d {,a}}

**Operands:**  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

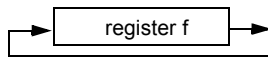
**Operation:**  $(f < n >) \rightarrow \text{dest} < n - 1 >$ ,  
 $(f < 0 >) \rightarrow \text{dest} < 7 >$

**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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**Operation:** FFh  $\rightarrow$  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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## 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:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(W) - (f) - (\overline{C}) \rightarrow \text{dest}$

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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## SUBLW Subtract W from literal

Syntax: SUBLW k  
 Operands:  $0 \leq k \leq 255$   
 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:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$   
 Operation:  $(f) - (W) \rightarrow \text{dest}$   
 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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

**SUBWFB Subtract W from f with Borrow**

Syntax: SUBWFB f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

Operation:  $(f) - (W) - (\bar{C}) \rightarrow \text{dest}$

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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

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

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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## 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 nn=0 * =1 *+ =2 *- =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	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	=	AAh
TBLPTR	=	01A357h
MEMORY(01A357h)	=	12h
MEMORY(01A358h)	=	34h

After Instruction

TABLAT	=	34h
TBLPTR	=	01A358h

# PIC18F6310/6410/8310/8410

## 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 nn=0 * =1 *+ =2 *- =3 +*
------	------	------	---

**Description:** This instruction uses the 3 LSBs of 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 “Program Memory”](#) for additional details on programming Flash memory.)

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range. The LSB of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

**Words:** 1

**Cycles:** 2

**Q Cycle Activity:**

	Q1	Q2	Q3	Q4
Decode	No operation	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

**Note:** The TBLWT instruction is not available in PIC18F6310/6410 devices (i.e., 64-pin devices) in normal operating modes. TBLWT can only be used by PIC18F8310/8410 devices with the external memory interface and only when writing to an external memory device. For more information, refer to [Section 7.4 “Writing to Program Memory Space \(PIC18F8310/8410 only\)”](#) and [Section 7.6 “Writing and Erasing On-Chip Program Memory \(ICSP Mode\)”](#).



# PIC18F6310/6410/8310/8410

## TSTFSZ Test f, skip if 0

**Syntax:** TSTFSZ f {,a}

**Operands:**  $0 \leq f \leq 255$   
 $a \in [0,1]$

**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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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  $\neq$  00h,  
PC = Address (NZERO)

## XORLW Exclusive OR literal with W

**Syntax:** XORLW k

**Operands:**  $0 \leq k \leq 255$

**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

# PIC18F6310/6410/8310/8410

---

## XORWF Exclusive OR W with f

---

Syntax: XORWF f {,d {,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$

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  $f \leq 95$  (5Fh). See [Section 25.2.3](#) 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

# PIC18F6310/6410/8310/8410

## 25.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18F6310/6410/8310/8410 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 for many of the standard PIC18 instructions.

The additional features of the extended instruction set 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 25-3. Detailed descriptions are provided in Section 25.2.2 “Extended Instruction Set”. The opcode field descriptions in Table 25-1 (page 298) 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.

### 25.2.1 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. The 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 25.2.3.1 “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 25-3: 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 <sub>s</sub> , f <sub>d</sub>	Move z <sub>s</sub> (source) to 1st word f <sub>d</sub> (destination) 2nd word	2	1110	1011	0zzz	zzzz	None
MOVSS z <sub>s</sub> , z <sub>d</sub>	Move z <sub>s</sub> (source) to 1st word z <sub>d</sub> (destination) 2nd word	2	1110	1011	1zzz	zzzz	None
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

**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)

# PIC18F6310/6410/8310/8410

## 25.2.2 EXTENDED INSTRUCTION SET

### ADDFSR Add Literal to FSR

Syntax: ADDFSR f, k  
 Operands:  $0 \leq k \leq 63$   
 $f \in [0, 1, 2]$   
 Operation:  $FSR(f) + k \rightarrow FSR(f)$   
 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:  $0 \leq k \leq 63$   
 Operation:  $FSR2 + k \rightarrow FSR2$ ,  
 (TOS)  $\rightarrow$  PC  
 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  
 After Instruction  
 FSR2 = 0422h  
 PC = (TOS)

# PIC18F6310/6410/8310/8410

**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<sub>s</sub>], f<sub>d</sub>

Operands: 0 ≤ z<sub>s</sub> ≤ 127  
0 ≤ f<sub>d</sub> ≤ 4095

Operation: ((FSR2) + z<sub>s</sub>) → f<sub>d</sub>

Status Affected: None

Encoding: 

1110	1011	0zzz	zzzz <sub>s</sub>
1111	ffff	ffff	ffff <sub>d</sub>

Description: The contents of the source register are moved to destination register 'f<sub>d</sub>'. The actual address of the source register is determined by adding the 7-bit literal offset 'z<sub>s</sub>' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal 'f<sub>d</sub>' 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 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

# PIC18F6310/6410/8310/8410

## MOVSS Move Indexed to Indexed

Syntax: MOVSS [z<sub>s</sub>], [z<sub>d</sub>]  
 Operands: 0 ≤ z<sub>s</sub> ≤ 127  
 0 ≤ z<sub>d</sub> ≤ 127  
 Operation: ((FSR2) + z<sub>s</sub>) → ((FSR2) + z<sub>d</sub>)  
 Status Affected: None  
 Encoding: 

1110	1011	1zzz	zzzz <sub>s</sub>
1111	xxxx	xzzz	zzzz <sub>d</sub>

  
 1st word (source)  
 2nd word (dest.)  
 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<sub>s</sub>' or 'z<sub>d</sub>', 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 ≤ k ≤ 255  
 Operation: k → (FSR2),  
 FSR2 - 1 → FSR2  
 Status Affected: None  
 Encoding: 

1110	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

# PIC18F6310/6410/8310/8410

## SUBFSR Subtract Literal from FSR

Syntax: SUBFSR f, k  
 Operands:  $0 \leq k \leq 63$   
 $f \in [0, 1, 2]$   
 Operation:  $FSRf - k \rightarrow FSRf$   
 Status Affected: None  
 Encoding: 

1110	1001	f fkk	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:  $0 \leq k \leq 63$   
 Operation:  $FSR2 - k \rightarrow FSR2$   
 (TOS)  $\rightarrow$  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)

# PIC18F6310/6410/8310/8410

---

## 25.2.3 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 ([Section 6.5.1 “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 25.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”](#)).

Although the Indexed Literal Offset 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 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.

### 25.2.3.1 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.

## 25.2.4 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 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 PIC18F6310/6410/8310/8410, 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.



# PIC18F6310/6410/8310/8410

## ADDWF ADD W to Indexed (Indexed Literal Offset mode)

Syntax: ADDWF [k] {,d}

Operands:  $0 \leq k \leq 95$   
 $d \in [0,1]$

Operation:  $(W) + ((FSR2) + k) \rightarrow \text{dest}$

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:  $0 \leq f \leq 95$   
 $0 \leq b \leq 7$

Operation:  $1 \rightarrow ((FSR2) + k) < b >$

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:  $0 \leq k \leq 95$

Operation:  $FFh \rightarrow ((FSR2) + k)$

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

# PIC18F6310/6410/8310/8410

---

## 25.2.5 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 PIC18F6310/6410/8310/8410 family of devices. This includes the MPLAB C18 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 is '0', disabling the extended instruction set and Indexed Literal Offset Addressing. 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.

## 26.0 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

## 26.1 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.

# PIC18F6310/6410/8310/8410

---

## 26.2 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.

## 26.3 HI-TECH C for Various Device Families

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

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

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

## 26.4 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

## 26.5 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

## 26.6 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

## 26.7 MPLAB SIM Software Simulator

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

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

## 26.8 MPLAB REAL ICE In-Circuit Emulator System

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

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

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

## 26.9 MPLAB ICD 3 In-Circuit Debugger System

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

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

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

The MPLAB PICkit 3 allows debugging and programming of PIC<sup>®</sup> and dsPIC<sup>®</sup> Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB Integrated Development Environment (IDE). The MPLAB PICkit 3 is connected to the design engineer's PC using a full speed USB interface and can be connected to the target via 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<sup>™</sup>.

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.

# PIC18F6310/6410/8310/8410

---

## 26.11 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.

## 26.12 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX 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.

## 26.13 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](http://www.microchip.com)) for the complete list of demonstration, development and evaluation kits.

# PIC18F6310/6410/8310/8410

## 27.0 ELECTRICAL CHARACTERISTICS

### Absolute Maximum Ratings<sup>(†)</sup>

Ambient temperature under bias .....	-40°C to +125°C
Storage temperature .....	-65°C to +150°C
Voltage on any pin with respect to V <sub>SS</sub> (except V <sub>DD</sub> , $\overline{\text{MCLR}}$ and RA4) .....	-0.3V to (V <sub>DD</sub> + 0.3V)
Voltage on V <sub>DD</sub> with respect to V <sub>SS</sub> .....	-0.3V to +7.5V
Voltage on $\overline{\text{MCLR}}$ with respect to V <sub>SS</sub> ( <b>Note 2</b> ) .....	0V to +13.25V
Voltage on RA4 with respect to V <sub>SS</sub> .....	0V to +8.5V
Total power dissipation ( <b>Note 1</b> ) .....	1.0W
Maximum current out of V <sub>SS</sub> pin .....	300 mA
Maximum current into V <sub>DD</sub> pin .....	250 mA
Input clamp current, I <sub>IK</sub> (V <sub>I</sub> < 0 or V <sub>I</sub> > V <sub>DD</sub> ) .....	±20 mA
Output clamp current, I <sub>OK</sub> (V <sub>O</sub> < 0 or V <sub>O</sub> > V <sub>DD</sub> ) .....	±20 mA
Maximum output current sunk by any I/O pin .....	25 mA
Maximum output current sourced by any I/O pin .....	25 mA
Maximum current sunk by all ports .....	200 mA
Maximum current sourced by all ports .....	200 mA

**Note 1:** Power dissipation is calculated as follows:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$$

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

† **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.

# PIC18F6310/6410/8310/8410

FIGURE 27-1: PIC18F6310/6410/8310/8410 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

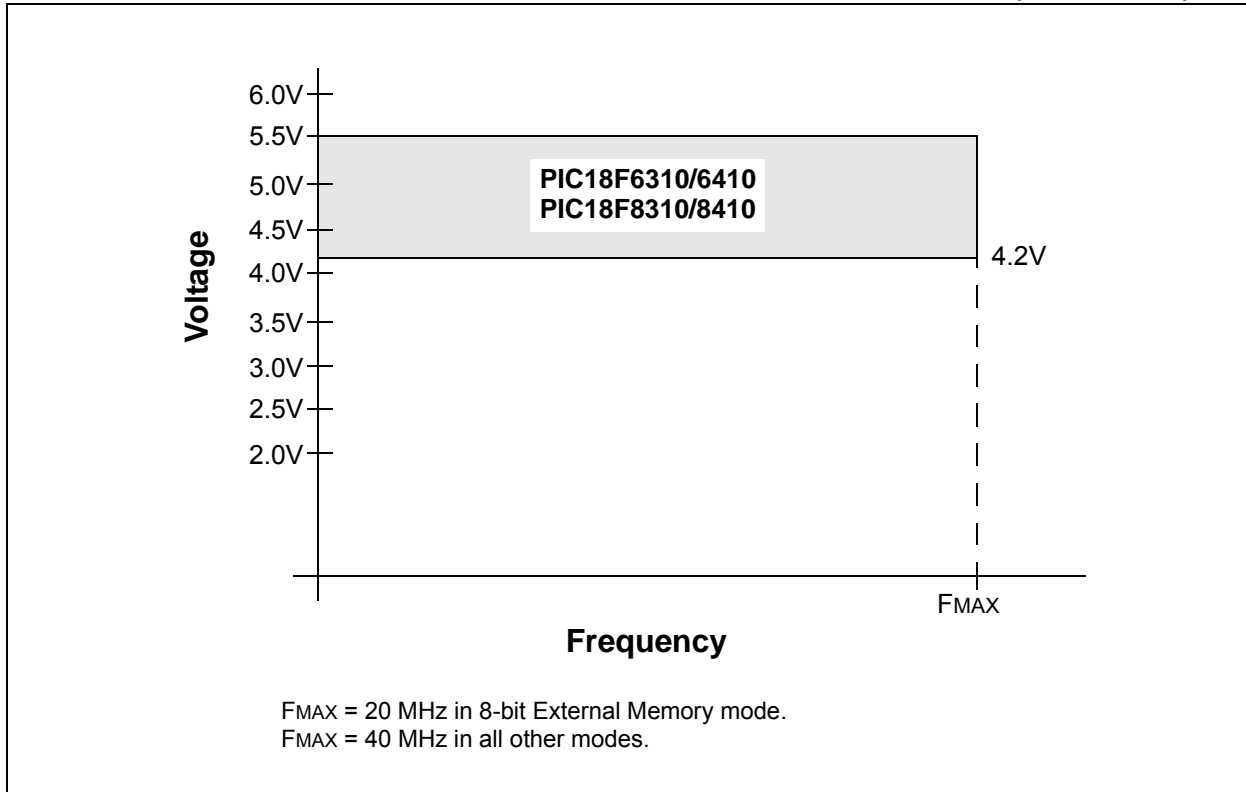
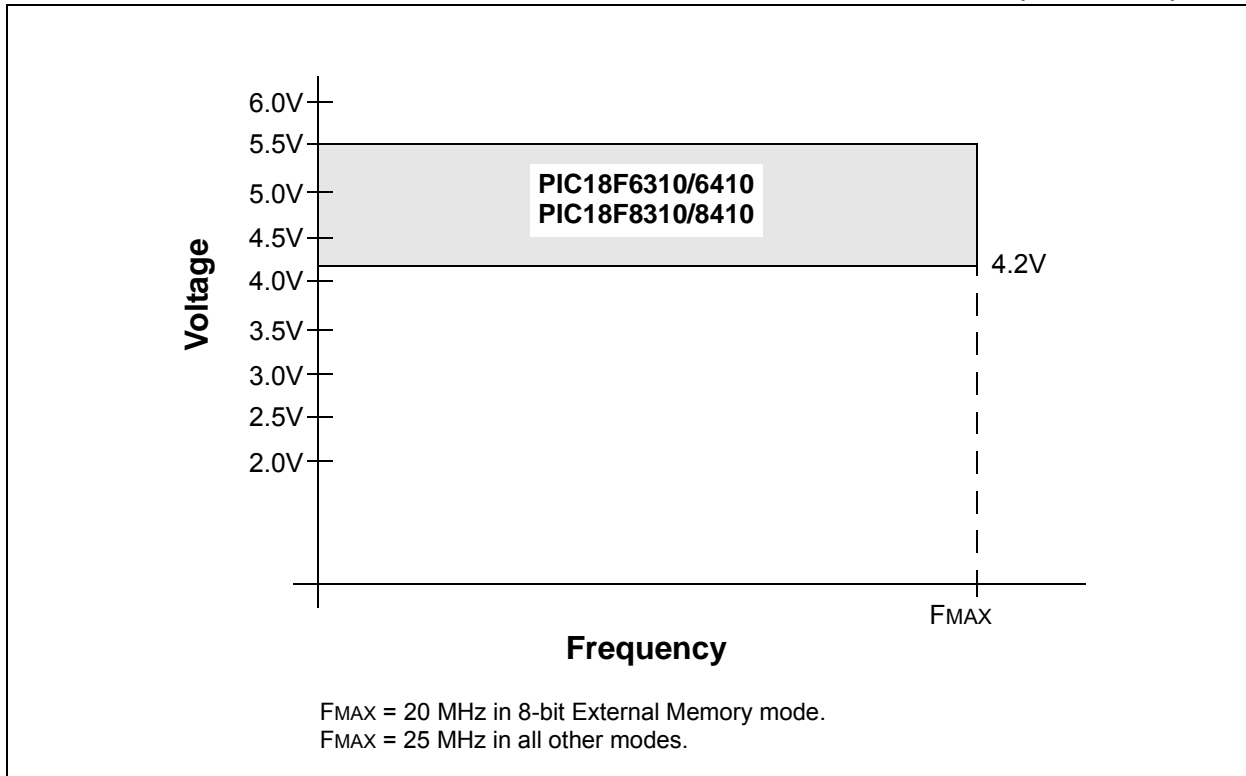


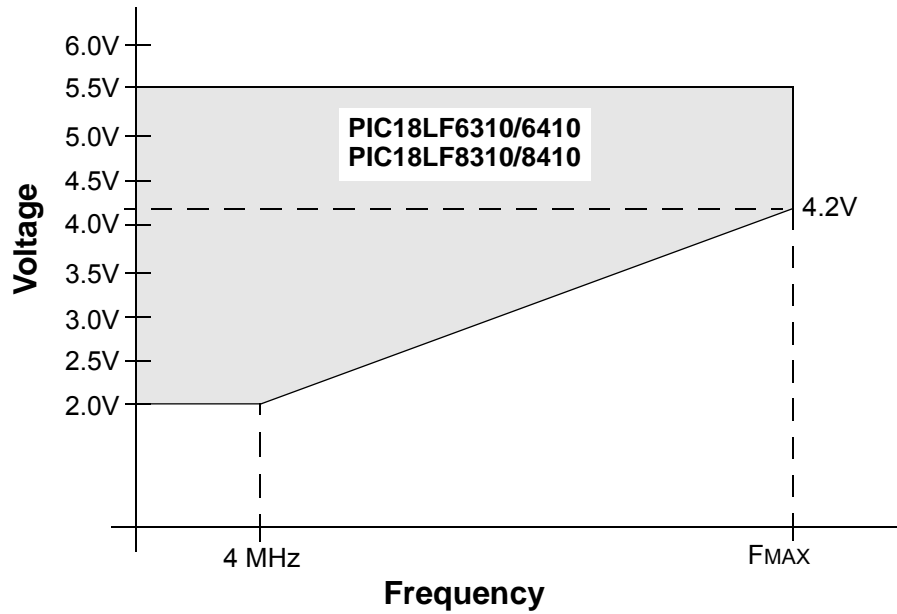
FIGURE 27-2: PIC18F6310/6410/8310/8410 VOLTAGE-FREQUENCY GRAPH (EXTENDED)





# PIC18F6310/6410/8310/8410

FIGURE 27-3: PIC18LF6310/6410/8310/8410 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



In 8-bit External Memory mode:

$F_{MAX} = (9.55 \text{ MHz/V}) (V_{DDAPP_{MIN}} - 2.0\text{V}) + 4 \text{ MHz}$ , if  $V_{DDAPP_{MIN}} \leq 4.2\text{V}$ ;

$F_{MAX} = 25 \text{ MHz}$ , if  $V_{DDAPP_{MIN}} > 4.2\text{V}$ .

In all other modes:

$F_{MAX} = (16.36 \text{ MHz/V}) (V_{DDAPP_{MIN}} - 2.0\text{V}) + 4 \text{ MHz}$ ;

$F_{MAX} = 40 \text{ MHz}$ , if  $V_{DDAPP_{MIN}} > 4.2\text{V}$ .

**Note:**  $V_{DDAPP_{MIN}}$  is the minimum voltage of the PIC<sup>®</sup> device in the application.

# PIC18F6310/6410/8310/8410

## 27.1 DC Characteristics: Supply Voltage PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial)

PIC18LF6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
PIC18F6310/6410/8310/8410 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
D001	VDD	<b>Supply Voltage</b>					
		PIC18LFX310/X410	2.0	—	5.5	V	
		PIC18F6310/6410/8310/8410	4.2	—	5.5	V	
D001B	AVDD	<b>Analog Supply Voltage</b>	$V_{DD} - 0.3$	$V_{DD} + 0.3$	—	V	
D001C	AVSS	<b>AVss Analog Ground Voltage</b>	$V_{SS} - 0.3$	$V_{SS} + 0.3$	—	V	
D002	VDR	<b>RAM Data Retention Voltage<sup>(1)</sup></b>	1.5	—	—	V	
D003	VPOR	<b>VDD Start Voltage</b> to Ensure Internal Power-on Reset Signal	—	—	0.7	V	See <a href="#">Section 5.3 “Power-on Reset (POR)”</a> for details
D004	SVDD	<b>VDD Rise Rate</b> to Ensure Internal Power-on Reset Signal	0.05	—	—	V/ms	See <a href="#">Section 5.3 “Power-on Reset (POR)”</a> for details
D005	VBOR	<b>Brown-out Reset Voltage</b>					
		BORV<1:0> = 11	1.96	2.06	2.16	V	
		BORV<1:0> = 10	2.64	2.78	2.92	V	
		BORV<1:0> = 01 <sup>(2)</sup>	4.11	4.33	4.55	V	
		BORV<1:0> = 00	4.41	4.64	4.87	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.

**Note 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.

# PIC18F6310/6410/8310/8410

## 27.2 DC Characteristics: Power-Down and Supply Current PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial)

PIC18LF6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial				
PIC18F6310/6410/8310/8410 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended				
Param No.	Device	Typ	Max	Units	Conditions	
<b>Power-Down Current (<math>I_{PD}</math>)<sup>(1)</sup></b>						
	PIC18LFX310/X410	0.1	1.0	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{DD} = 2.0\text{V}$ (Sleep mode)
		0.1	1.0	$\mu\text{A}$	$+25^{\circ}\text{C}$	
		0.3	5.0	$\mu\text{A}$	$+85^{\circ}\text{C}$	
	PIC18LFX310/X410	0.1	2.0	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{DD} = 3.0\text{V}$ (Sleep mode)
		0.1	2.0	$\mu\text{A}$	$+25^{\circ}\text{C}$	
		0.3	8.0	$\mu\text{A}$	$+85^{\circ}\text{C}$	
	All devices	0.1	2.0	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{DD} = 5.0\text{V}$ (Sleep mode)
		0.1	2.0	$\mu\text{A}$	$+25^{\circ}\text{C}$	
		0.4	15	$\mu\text{A}$	$+85^{\circ}\text{C}$	
		11	50	$\mu\text{A}$	$+125^{\circ}\text{C}$	

**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  $V_{DD}$  or  $V_{SS}$ , 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  $I_{DD}$  measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are tri-stated, pulled to  $V_{DD}$  or  $V_{SS}$ ; MCLR =  $V_{DD}$ ; WDT is enabled/disabled as specified.
- 3:** When operation below  $-10^{\circ}\text{C}$  is expected, use the T1OSC High-Power mode, where LPT1OSC (CONFIG3H<2>) = 0. When operation will always be above  $-10^{\circ}\text{C}$ , then the low-power Timer1 oscillator may be selected.
- 4:** BOR and HLVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F6310/6410/8310/8410

## 27.2 DC Characteristics: Power-Down and Supply Current PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial) (Continued)

PIC18LF6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18F6310/6410/8310/8410 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (IDD)<sup>(2,3)</sup></b>							
	PIC18LFX310/X410	12	26	μA	-40°C	VDD = 2.0V	FOSC = 31 kHz (RC_RUN mode, Internal oscillator source)
		12	24	μA	+25°C		
		12	23	μA	+85°C		
	PIC18LFX310/X410	32	50	μA	-40°C	VDD = 3.0V	
		27	48	μA	+25°C		
		22	46	μA	+85°C		
	All devices	84	134	μA	-40°C	VDD = 5.0V	
		82	128	μA	+25°C		
		72	122	μA	+85°C		
		90	145	μA	+125°C		
	PIC18LFX310/X410	.26	.8	mA	-40°C	VDD = 2.0V	
		.26	.8	mA	+25°C		
		.26	.8	mA	+85°C		
	PIC18LFX310/X410	.48	1.04	mA	-40°C	VDD = 3.0V	
		.44	.96	mA	+25°C		
		.48	.88	mA	+85°C		
	All devices	.88	1.84	mA	-40°C	VDD = 5.0V	
		.88	1.76	mA	+25°C		
		.8	1.68	mA	+85°C		
		1.25	2.2	mA	+125°C		
	PIC18LFX310/X410	0.6	1.7	mA	-40°C	VDD = 2.0V	
		0.6	1.6	mA	+25°C		
		0.6	1.5	mA	+85°C		
	PIC18LFX310/X410	1.0	2.4	mA	-40°C	VDD = 3.0V	
		1.0	2.4	mA	+25°C		
		1.0	2.4	mA	+85°C		
	All devices	2.0	4.2	mA	-40°C	VDD = 5.0V	
		2.0	4	mA	+25°C		
		2.0	3.8	mA	+85°C		
		2.7	4.3	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 are tri-stated, pulled to VDD or VSS; MCLR = VDD; WDT is enabled/disabled as specified.
- 3:** When operation below -10°C is expected, use the T1OSC High-Power mode, where LPT1OSC (CONFIG3H<2>) = 0. When operation will always be above -10°C, then the low-power Timer1 oscillator may be selected.
- 4:** BOR and HLVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F6310/6410/8310/8410

## 27.2 DC Characteristics: Power-Down and Supply Current PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial) (Continued)

PIC18LF6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature		-40°C ≤ TA ≤ +85°C for industrial			
PIC18F6310/6410/8310/8410 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated)					
		Operating temperature		-40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended			
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (IDD)<sup>(2,3)</sup></b>							
	PIC18LFX310/X410	2.3	6.4	μA	-40°C	VDD = 2.0V	FOSC = 31 kHz (RC_IDLE mode, Internal oscillator source)
		2.5	6.4	μA	+25°C		
		2.9	8.8	μA	+85°C		
PIC18LFX310/X410	3.6	8.8	μA	-40°C	VDD = 3.0V		
	3.8	8.8	μA	+25°C			
	4.6	12	μA	+85°C			
All devices	All devices	7.4	16	μA	-40°C	VDD = 5.0V	
		7.8	13	μA	+25°C		
		9.1	29	μA	+85°C		
		21	97	μA	+125°C		
PIC18LFX310/X410	PIC18LFX310/X410	132	450	μA	-40°C	VDD = 2.0V	FOSC = 1 MHz (RC_IDLE mode, Internal oscillator source)
		140	450	μA	+25°C		
		152	450	μA	+85°C		
PIC18LFX310/X410	PIC18LFX310/X410	200	600	μA	-40°C	VDD = 3.0V	
		216	600	μA	+25°C		
		252	600	μA	+85°C		
All devices	All devices	400	990	μA	-40°C	VDD = 5.0V	
		420	990	μA	+25°C		
		440	990	μA	+85°C		
		850	1.2	μA	+125°C		
PIC18LFX310/X410	PIC18LFX310/X410	272	690	μA	-40°C	VDD = 2.0V	FOSC = 4 MHz (RC_IDLE mode, Internal oscillator source)
		280	690	μA	+25°C		
		288	690	μA	+85°C		
PIC18LFX310/X410	PIC18LFX310/X410	416	990	μA	-40°C	VDD = 3.0V	
		432	990	μA	+25°C		
		464	990	μA	+85°C		
All devices	All devices	.8	1.9	mA	-40°C	VDD = 5.0V	
		.9	1.9	mA	+25°C		
		.9	1.9	mA	+85°C		
		1.6	2.2	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 are tri-stated, pulled to VDD or VSS; MCLR = VDD; WDT is enabled/disabled as specified.

**3:** When operation below -10°C is expected, use the T1OSC High-Power mode, where LPT1OSC (CONFIG3H<2>) = 0. When operation will always be above -10°C, then the low-power Timer1 oscillator may be selected.

**4:** BOR and HLVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F6310/6410/8310/8410

## 27.2 DC Characteristics: Power-Down and Supply Current PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial) (Continued)

PIC18LF6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18F6310/6410/8310/8410 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (IDD)<sup>(2,3)</sup></b>							
	PIC18LFX310/X410	250	500	μA	-40°C	VDD = 2.0V	FOSC = 1 MHz (PRI_RUN, EC oscillator)
		260	500	μA	+25°C		
		250	500	μA	+85°C		
	PIC18LFX310/X410	550	650	μA	-40°C	VDD = 3.0V	
		480	650	μA	+25°C		
		460	650	μA	+85°C		
	All devices	1.2	1.6	mA	-40°C	VDD = 5.0V	
		1.1	1.5	mA	+25°C		
		1.0	1.4	mA	+85°C		
		1.5	1.9	mA	+125°C		
	PIC18LFX310/X410	0.72	2.0	mA	-40°C	VDD = 2.0V	
		0.74	2.0	mA	+25°C		
		0.74	2.0	mA	+85°C		
	PIC18LFX310/X410	1.3	3.0	mA	-40°C	VDD = 3.0V	
		1.3	3.0	mA	+25°C		
		1.3	3.0	mA	+85°C		
	All devices	2.7	6.0	mA	-40°C	VDD = 5.0V	
		2.6	6.0	mA	+25°C		
		2.5	6.0	mA	+85°C		
		4.2	8	mA	+125°C		
	All devices	15	35	mA	-40°C	VDD = 4.2V	
		16	35	mA	+25°C		
		16	35	mA	+85°C		
	All devices	21	40	mA	-40°C	VDD = 5.0V	
		21	40	mA	+25°C		
		21	40	mA	+85°C		
		30	50	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 are tri-stated, pulled to VDD or VSS; MCLR = VDD; WDT is enabled/disabled as specified.
- 3:** When operation below -10°C is expected, use the T1OSC High-Power mode, where LPT1OSC (CONFIG3H<2>) = 0. When operation will always be above -10°C, then the low-power Timer1 oscillator may be selected.
- 4:** BOR and HLVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.



# PIC18F6310/6410/8310/8410

## 27.2 DC Characteristics: Power-Down and Supply Current PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial) (Continued)

Param No.	Device	Typ	Max	Units	Conditions	
PIC18LF6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial				
PIC18F6310/6410/8310/8410 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended				
<b>Supply Current (IDD)<sup>(2,3)</sup></b>						
	All devices	7.5	16	mA	$-40^{\circ}\text{C}$	VDD = 4.2V  FOSC = 4 MHz, 16 MHz internal (PRI_RUN HSPLL mode)
		7.4	15	mA	$+25^{\circ}\text{C}$	
		7.3	14	mA	$+85^{\circ}\text{C}$	
	All devices	10	21	mA	$-40^{\circ}\text{C}$	VDD = 5.0V  FOSC = 4 MHz, 16 MHz internal (PRI_RUN HSPLL mode)
		10	20	mA	$+25^{\circ}\text{C}$	
		9.7	19	mA	$+85^{\circ}\text{C}$	
	All devices	17	35	mA	$-40^{\circ}\text{C}$	VDD = 4.2V  FOSC = 10 MHz, 40 MHz internal (PRI_RUN HSPLL mode)
		17	35	mA	$+25^{\circ}\text{C}$	
		17	35	mA	$+85^{\circ}\text{C}$	
	All devices	23	40	mA	$-40^{\circ}\text{C}$	VDD = 5.0V  FOSC = 10 MHz, 40 MHz internal (PRI_RUN HSPLL mode)
		23	40	mA	$+25^{\circ}\text{C}$	
		23	40	mA	$+85^{\circ}\text{C}$	

**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 are tri-stated, pulled to VDD or VSS; MCLR = VDD; WDT is enabled/disabled as specified.
- 3:** When operation below  $-10^{\circ}\text{C}$  is expected, use the T1OSC High-Power mode, where LPT1OSC (CONFIG3H<2>) = 0. When operation will always be above  $-10^{\circ}\text{C}$ , then the low-power Timer1 oscillator may be selected.
- 4:** BOR and HLVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.



# PIC18F6310/6410/8310/8410

## 27.2 DC Characteristics: Power-Down and Supply Current PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial) (Continued)

PIC18LF6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
PIC18F6310/6410/8310/8410 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (I<sub>DD</sub>)<sup>(2)</sup></b>							
PIC18LFX310/X410		13	40	$\mu\text{A}$	$-10^{\circ}\text{C}$	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 32 kHz <sup>(4)</sup> ( <b>SEC_RUN</b> mode, Timer1 as clock)
		14	40	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		16	40	$\mu\text{A}$	$+70^{\circ}\text{C}$		
PIC18LFX310/X410		34	74	$\mu\text{A}$	$-10^{\circ}\text{C}$	V <sub>DD</sub> = 3.0V	
		31	70	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		28	67	$\mu\text{A}$	$+70^{\circ}\text{C}$		
All devices		72	150	$\mu\text{A}$	$-10^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V	
		65	150	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		59	150	$\mu\text{A}$	$+70^{\circ}\text{C}$		
		90	170	$\mu\text{A}$	$+125^{\circ}\text{C}$		
PIC18LFX310/X410		5.5	15	$\mu\text{A}$	$-10^{\circ}\text{C}$	V <sub>DD</sub> = 2.0V	F <sub>OSC</sub> = 32 kHz <sup>(4)</sup> ( <b>SEC_IDLE</b> mode, Timer1 as clock)
		5.8	15	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		6.1	18	$\mu\text{A}$	$+70^{\circ}\text{C}$		
PIC18LFX310/X410		8.2	30	$\mu\text{A}$	$-10^{\circ}\text{C}$	V <sub>DD</sub> = 3.0V	
		8.6	30	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		8.8	35	$\mu\text{A}$	$+70^{\circ}\text{C}$		
All devices		13	80	$\mu\text{A}$	$-10^{\circ}\text{C}$	V <sub>DD</sub> = 5.0V	
		13	80	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		13	85	$\mu\text{A}$	$+70^{\circ}\text{C}$		
		22	90	$\mu\text{A}$	$+125^{\circ}\text{C}$		

**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 V<sub>DD</sub> or V<sub>SS</sub>, 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 I<sub>DD</sub> measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are tri-stated, pulled to V<sub>DD</sub> or V<sub>SS</sub>; MCLR = V<sub>DD</sub>; WDT is enabled/disabled as specified.

**3:** When operation below  $-10^{\circ}\text{C}$  is expected, use the T1OSC High-Power mode, where LPT1OSC (CONFIG3H<2>) = 0. When operation will always be above  $-10^{\circ}\text{C}$ , then the low-power Timer1 oscillator may be selected.

**4:** BOR and HLVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F6310/6410/8310/8410

## 27.2 DC Characteristics: Power-Down and Supply Current PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial) (Continued)

PIC18LF6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial					
PIC18F6310/6410/8310/8410 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended					
Param No.	Device	Typ	Max	Units	Conditions		
<b>Module Differential Currents (<math>\Delta I_{\text{WDT}}</math>, <math>\Delta I_{\text{BOR}}</math>, <math>\Delta I_{\text{LVD}}</math>, <math>\Delta I_{\text{OSCB}}</math>, <math>\Delta I_{\text{AD}}</math>)</b>							
D022 ( $\Delta I_{\text{WDT}}$ )	Watchdog Timer	1.7	4.0	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{\text{DD}} = 2.0\text{V}$	
		2.1	4.0	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		2.6	5.0	$\mu\text{A}$	$+85^{\circ}\text{C}$		
		2.2	6.0	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{\text{DD}} = 3.0\text{V}$	
		2.4	6.0	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		2.8	7.0	$\mu\text{A}$	$+85^{\circ}\text{C}$		
		2.9	10.0	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{\text{DD}} = 5.0\text{V}$	
		3.1	10.0	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		3.3	13.0	$\mu\text{A}$	$+85^{\circ}\text{C}$		
		20	190	$\mu\text{A}$	$+125^{\circ}\text{C}$		
D022A ( $\Delta I_{\text{BOR}}$ )	Brown-out Reset <sup>(4)</sup>	17	50.0	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	$V_{\text{DD}} = 3.0\text{V}$	
		47	60.0	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	$V_{\text{DD}} = 5.0\text{V}$	
		90	200	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$		
D022B ( $\Delta I_{\text{LVD}}$ )	High/Low-Voltage Detect <sup>(4)</sup>	14	38.0	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	$V_{\text{DD}} = 2.0\text{V}$	
		18	40.0	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	$V_{\text{DD}} = 3.0\text{V}$	
		21	45.0	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	$V_{\text{DD}} = 5.0\text{V}$	
		90	2000	$\mu\text{A}$	$-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$		
D025 ( $\Delta I_{\text{OSCB}}$ )	Timer1 Oscillator	1.0	3.5	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{\text{DD}} = 2.0\text{V}$	32 kHz on Timer1 <sup>(4)</sup>
		1.1	3.5	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		1.1	4.5	$\mu\text{A}$	$+70^{\circ}\text{C}$		
		1.2	4.5	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{\text{DD}} = 3.0\text{V}$	32 kHz on Timer1 <sup>(4)</sup>
		1.3	4.5	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		1.2	5.5	$\mu\text{A}$	$+70^{\circ}\text{C}$		
		1.8	6.0	$\mu\text{A}$	$-40^{\circ}\text{C}$	$V_{\text{DD}} = 5.0\text{V}$	32 kHz on Timer1 <sup>(4)</sup>
		1.9	6.0	$\mu\text{A}$	$+25^{\circ}\text{C}$		
1.9	7.0	$\mu\text{A}$	$+85^{\circ}\text{C}$				
D026 ( $\Delta I_{\text{AD}}$ )	A/D Converter	1.0	3.0	$\mu\text{A}$	—	$V_{\text{DD}} = 2.0\text{V}$	A/D on, not converting, $1.6 \mu\text{s} \leq T_{\text{AD}} \leq 6.4 \mu\text{s}$
		1.0	4.0	$\mu\text{A}$	—	$V_{\text{DD}} = 3.0\text{V}$	
		1.0	8.0	$\mu\text{A}$	—	$V_{\text{DD}} = 5.0\text{V}$	
		15	60	$\mu\text{A}$	$+125^{\circ}\text{C}$		

**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  $V_{\text{DD}}$  or  $V_{\text{SS}}$ , 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  $I_{\text{DD}}$  measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are tri-stated, pulled to  $V_{\text{DD}}$  or  $V_{\text{SS}}$ ;  $\text{MCLR} = V_{\text{DD}}$ ; WDT is enabled/disabled as specified.
- 3:** When operation below  $-10^{\circ}\text{C}$  is expected, use the T1OSC High-Power mode, where  $\text{LPT1OSC}(\text{CONFIG3H}\langle 2 \rangle) = 0$ . When operation will always be above  $-10^{\circ}\text{C}$ , then the low-power Timer1 oscillator may be selected.
- 4:** BOR and HLVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F6310/6410/8310/8410

## 27.3 DC Characteristics: PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D030 D030A D031 D031A D031B D032 D033 D033A D033B D034	V <sub>IL</sub>	<b>Input Low Voltage</b> I/O Ports: with TTL Buffer with Schmitt Trigger Buffer RC3 and RC4 $\overline{\text{MCLR}}$ OSC1 OSC1 OSC1 T13CKI	V <sub>SS</sub> — V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub> V <sub>SS</sub>	0.15 V <sub>DD</sub> 0.8 0.2 V <sub>DD</sub> 0.3 V <sub>DD</sub> 0.8 0.2 V <sub>DD</sub> 0.3 V <sub>DD</sub> 0.2 V <sub>DD</sub> 0.3 0.3	V V V V V V V V V V	V <sub>DD</sub> < 4.5V 4.5V ≤ V <sub>DD</sub> ≤ 5.5V I <sup>2</sup> C™ enabled SMBus enabled HS, HSPLL modes RC, EC modes <sup>(1)</sup> XT, LP modes
D040 D040A D041 D041A D041B D042 D043 D043A D043B D043C D044	V <sub>IH</sub>	<b>Input High Voltage</b> I/O Ports: with TTL Buffer with Schmitt Trigger Buffer RC3 and RC4 $\overline{\text{MCLR}}$ OSC1 OSC1 OSC1 T13CKI	0.25 V <sub>DD</sub> + 0.8V 2.0 0.8 V <sub>DD</sub> 0.7 V <sub>DD</sub> 2.1 0.8 V <sub>DD</sub> 0.7 V <sub>DD</sub> 0.8 V <sub>DD</sub> 0.9 V <sub>DD</sub> 1.6 1.6	V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub> V <sub>DD</sub>	V V V V V V V V V V V	V <sub>DD</sub> < 4.5V 4.5V ≤ V <sub>DD</sub> ≤ 5.5V I <sup>2</sup> C enabled SMBus enabled HS, HSPLL modes EC mode RC mode <sup>(1)</sup> XT, LP modes
D060  D061 D063	I <sub>IL</sub>	<b>Input Leakage Current<sup>(2,3)</sup></b> I/O Ports $\overline{\text{MCLR}}$ OSC1	— — —	±200 ±50 ±1 ±1	nA nA μA μA	V <sub>DD</sub> < 5.5V V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub> , Pin at high-impedance V <sub>DD</sub> < 3V V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub> , Pin at high-impedance V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub> V <sub>SS</sub> ≤ V <sub>PIN</sub> ≤ V <sub>DD</sub>
D070	I <sub>PU</sub> I <sub>PURB</sub>	<b>Weak Pull-up Current</b> PORTB Weak Pull-up Current	50	400	μA	V <sub>DD</sub> = 5V, V <sub>PIN</sub> = V <sub>SS</sub>

**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  $\overline{\text{MCLR}}$  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.

# PIC18F6310/6410/8310/8410

## 27.3 DC Characteristics: PIC18F6310/6410/8310/8410 (Industrial, Extended) PIC18LF6310/6410/8310/8410 (Industrial) (Continued)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D080	VOL	<b>Output Low Voltage</b> I/O Ports	—	0.6	V	$I_{OL} = 8.5 \text{ mA}$ , $V_{DD} = 4.5\text{V}$ , $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D083		OSC2/CLKO (RC, RCIO, EC, ECIO modes)	—	0.6	V	$I_{OL} = 1.6 \text{ mA}$ , $V_{DD} = 4.5\text{V}$ , $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D090	VOH	<b>Output High Voltage<sup>(3)</sup></b> I/O Ports	$V_{DD} - 0.7$	—	V	$I_{OH} = -3.0 \text{ mA}$ , $V_{DD} = 4.5\text{V}$ , $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D092		OSC2/CLKO (RC, RCIO, EC, ECIO modes)	$V_{DD} - 0.7$	—	V	$I_{OH} = -1.3 \text{ mA}$ , $V_{DD} = 4.5\text{V}$ , $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
<b>Capacitive Loading Specs on Output Pins</b>						
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 <sup>2</sup> C™ 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  $\overline{\text{MCLR}}$  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.

# PIC18F6310/6410/8310/8410

**TABLE 27-1: MEMORY PROGRAMMING REQUIREMENTS**

DC Characteristics			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended				
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
<b>Program Flash Memory</b>							
D110	VPP	Voltage on $\overline{\text{MCLR}}$ /VPP pin	10.0	—	12.0	V	
D113	IDDP	Supply Current during Programming	—	—	1	mA	
D130	EP	Cell Endurance	—	1K	—	E/W	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D131	VPR	VDD for Read	V <sub>MIN</sub>	—	5.5	V	V <sub>MIN</sub> = Minimum operating voltage
D132	VIE	VDD for Block Erase	2.75	—	5.5	V	Using ICSP port
D132A	VIW	VDD for Externally Timed Erase or Write	2.75	—	5.5	V	Using ICSP port
D132B	VPEW	VDD for Self-timed Write	V <sub>MIN</sub>	—	5.5	V	V <sub>MIN</sub> = Minimum operating voltage
D133	TIE	ICSP™ Block Erase Cycle Time	—	4	—	ms	V <sub>DD</sub> > 4.5V
D133A	TIW	ICSP Erase or Write Cycle Time (externally timed)	2	—	—	ms	V <sub>DD</sub> > 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.

# PIC18F6310/6410/8310/8410

**TABLE 27-2: COMPARATOR SPECIFICATIONS**

Operating Conditions: $3.0V < V_{DD} < 5.5V$ , $-40^{\circ}C < T_A < +85^{\circ}C$ , unless otherwise stated.							
Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D300	V <sub>IOFF</sub>	Input Offset Voltage	—	±5.0	±10	mV	
D301	V <sub>ICM</sub>	Input Common Mode Voltage	0	—	$V_{DD} - 1.5$	V	
D302	CMRR	Common Mode Rejection Ratio	55	—	—	dB	
D303	T <sub>RESP</sub>	Response Time <sup>(1)</sup>	—	150	400	ns	PIC18FXXXX
D303A			—	150	600	ns	PIC18LFXXXX, $V_{DD} = 2.0V$
D304	T <sub>M2OV</sub>	Comparator Mode Change to Output Valid	—	—	10	μs	

**Note 1:** Response time measured with one comparator input at  $(V_{DD} - 1.5)/2$ , while the other input transitions from V<sub>SS</sub> to V<sub>DD</sub>.

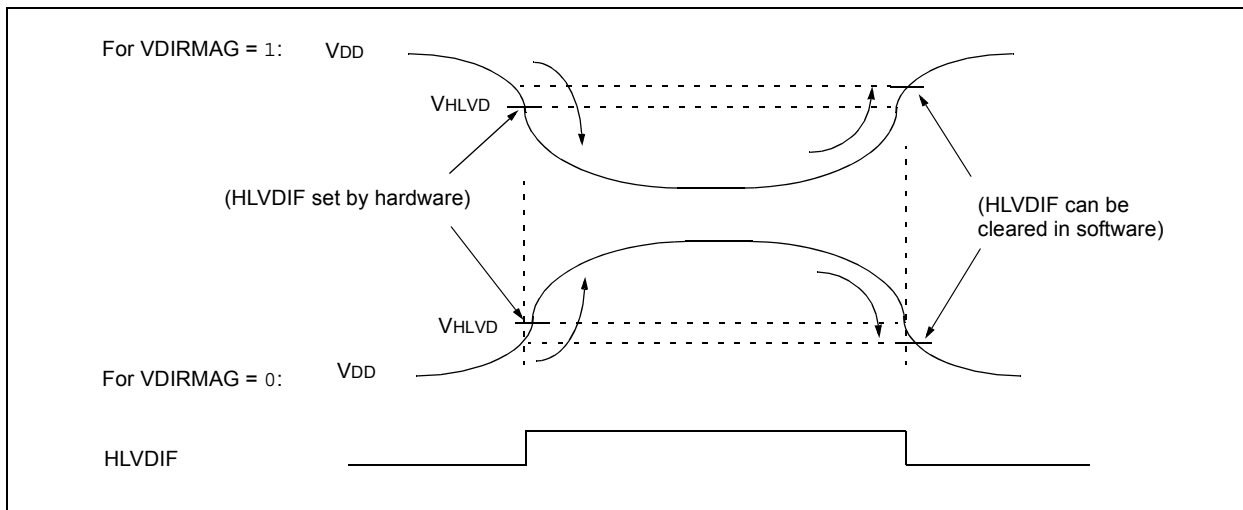
**TABLE 27-3: VOLTAGE REFERENCE SPECIFICATIONS**

Operating Conditions: $3.0V < V_{DD} < 5.5V$ , $-40^{\circ}C < T_A < +85^{\circ}C$ , unless otherwise stated.							
Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D310	V <sub>RES</sub>	Resolution	$V_{DD}/24$	—	$V_{DD}/32$	LSb	
D311	V <sub>RAA</sub>	Absolute Accuracy	—	—	1/4	LSb	Low Range (CVRR = 1)
			—	—	1/2	LSb	High Range (CVRR = 0)
D312	V <sub>RUR</sub>	Unit Resistor Value (R)	—	2k	—	Ω	
310	T <sub>SET</sub>	Settling Time <sup>(1)</sup>	—	—	10	μs	

**Note 1:** Settling time measured while CVRR = 1 and CVR<3:0> transitions from '0000' to '1111'.

# PIC18F6310/6410/8310/8410

**FIGURE 27-4: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS**



**TABLE 27-4: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS**

				Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial				
Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
D420		HLVD Voltage on VDD Transition	LVV = 0000	1.80	1.86	1.91	V	
			LVV = 0001	1.96	2.06	2.06	V	
			LVV = 0010	2.16	2.27	2.38	V	
			LVV = 0011	2.35	2.47	2.59	V	
			LVV = 0100	2.43	2.56	2.69	V	
			LVV = 0101	2.64	2.78	2.92	V	
			LVV = 0110	2.75	2.89	3.03	V	
			LVV = 0111	2.95	3.10	3.26	V	
			LVV = 1000	3.24	3.41	3.58	V	
			LVV = 1001	3.43	3.61	3.79	V	
			LVV = 1010	3.53	3.72	3.91	V	
			LVV = 1011	3.72	3.92	4.12	V	
			LVV = 1100	3.92	4.13	4.34	V	
			LVV = 1101	4.11	4.33	4.55	V	
			LVV = 1110	4.41	4.64	4.87	V	
D420B	VBG	Band Gap Reference Voltage Value	LVV = 1111	—	1.20	—	V	HLVDIN input external

† Production tested at  $T_{AMB} = 25^{\circ}\text{C}$ . Specifications over temperature limits ensured by characterization.

# PIC18F6310/6410/8310/8410

## 27.4 AC (Timing) Characteristics

### 27.4.1 TIMING PARAMETER SYMBOLOGY

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

1. TppS2ppS
2. TppS
3. TCC:ST (I<sup>2</sup>C specifications only)
4. Ts (I<sup>2</sup>C specifications only)

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

Lowercase letters (pp) and their meanings:

<p>pp</p> <p>cc      CCP1</p> <p>ck      CLKO</p> <p>cs      <math>\overline{CS}</math></p> <p>di      SDI</p> <p>do      SDO</p> <p>dt      Data in</p> <p>io      I/O port</p> <p>mc      <math>\overline{MCLR}</math></p>	<p>osc      OSC1</p> <p>rd      <math>\overline{RD}</math></p> <p>rw      <math>\overline{RD}</math> or <math>\overline{WR}</math></p> <p>sc      SCK</p> <p>ss      <math>\overline{SS}</math></p> <p>t0      T0CKI</p> <p>t1      T13CKI</p> <p>wr      <math>\overline{WR}</math></p>
--	--

Uppercase letters and their meanings:

<p>S</p> <p>F      Fall</p> <p>H      High</p> <p>I      Invalid (High-impedance)</p> <p>L      Low</p> <p>I<sup>2</sup>C only</p> <p>AA      output access</p> <p>BUF      Bus free</p>	<p>P      Period</p> <p>R      Rise</p> <p>V      Valid</p> <p>Z      High-impedance</p> <p>High      High</p> <p>Low      Low</p>
--	---

TCC:ST (I<sup>2</sup>C specifications only)

<p>CC</p> <p>HD      Hold</p> <p>ST</p> <p>DAT      DATA input hold</p> <p>STA      Start condition</p>	<p>SU      Setup</p> <p>STO      Stop condition</p>
---	--



# PIC18F6310/6410/8310/8410

## 27.4.2 TIMING CONDITIONS

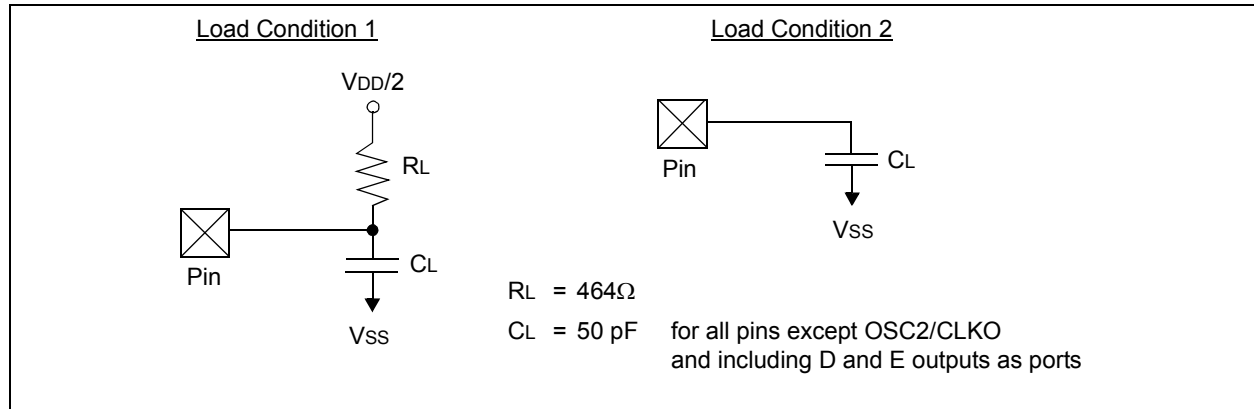
The temperature and voltages specified in [Table 27-5](#) apply to all timing specifications unless otherwise noted. [Figure 27-5](#) specifies the load conditions for the timing specifications.

**Note:** Because of space limitations, the generic terms “PIC18FXXXX” and “PIC18LFXXXX” are used throughout this section to refer to the PIC18F6310/6410/8310/8410 and PIC18LF6310/6410/8310/8410 families of devices specifically and only those devices.

**TABLE 27-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

<b>AC CHARACTERISTICS</b>	<b>Standard Operating Conditions (unless otherwise stated)</b>
	Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended
	Operating voltage $V_{DD}$ range as described in DC spec, <a href="#">Section 27.1</a> and <a href="#">Section 27.3</a> .
	LF parts operate for industrial temperatures only.

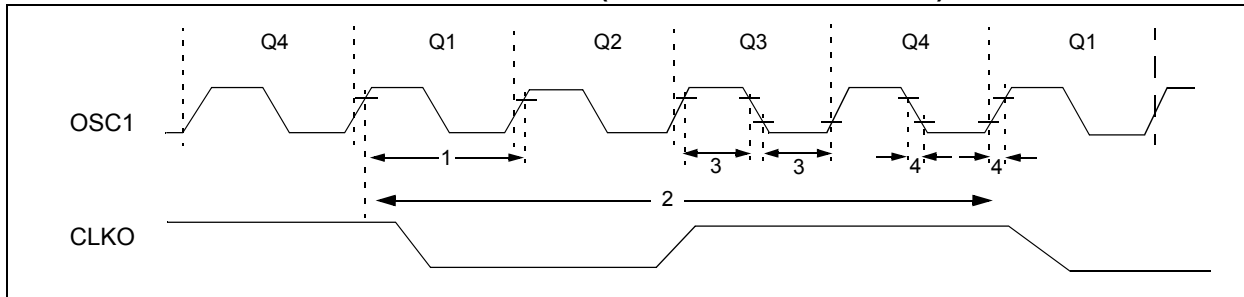
**FIGURE 27-5: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**



# PIC18F6310/6410/8310/8410

## 27.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 27-6: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)**



**TABLE 27-6: EXTERNAL CLOCK TIMING REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	Fosc	External CLKI Frequency <sup>(1)</sup>	DC	1	MHz	XT, RC Oscillator mode
			DC	25	MHz	HS Oscillator mode
			DC	31.25	kHz	LP Oscillator mode
		Oscillator Frequency <sup>(1)</sup>	DC	40	MHz	EC Oscillator mode
			DC	4	MHz	RC Oscillator mode
			0.1	4	MHz	XT Oscillator mode
			4	25	MHz	HS Oscillator mode
			4	10	MHz	HS + PLL Oscillator mode
			5	200	kHz	LP Oscillator mode
1	Tosc	External CLKI Period <sup>(1)</sup>	1000	—	ns	XT, RC Oscillator mode
			40	—	ns	HS Oscillator mode
			32	—	μs	LP Oscillator mode
		Oscillator Period <sup>(1)</sup>	25	—	ns	EC Oscillator mode
			250	—	ns	RC Oscillator mode
			0.25	10	μs	XT Oscillator mode
			40	250	ns	HS Oscillator mode
			100	250	ns	HS + PLL Oscillator mode
			5	200	μs	LP Oscillator mode
2	Tcy	Instruction Cycle Time <sup>(1)</sup>	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.

# PIC18F6310/6410/8310/8410

**TABLE 27-7: PLL CLOCK TIMING SPECIFICATIONS (V<sub>DD</sub> = 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 <sub>rc</sub>	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 27-8: AC CHARACTERISTICS: INTERNAL RC ACCURACY**  
**PIC18F6310/6410/8310/8410 (INDUSTRIAL)**  
**PIC18LF6310/6410/8310/8410 (INDUSTRIAL)**

PIC18LF6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18F6310/6410/8310/8410 (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param No.	Device	Min	Typ	Max	Units	Conditions	
<b>INTOSC Accuracy @ Freq = 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz, 125 kHz<sup>(1)</sup></b>							
	PIC18LF6310/6410/8310/8410	-2	+/-1	2	%	+25°C	V <sub>DD</sub> = 2.7-3.3 V
		-5	—	5	%	-10°C to +85°C	V <sub>DD</sub> = 2.7-3.3 V
		-10	+/-1	10	%	-40°C to +85°C	V <sub>DD</sub> = 2.7-3.3 V
	PIC18F6310/6410/8310/8410	-2	+/-1	2	%	+25°C	V <sub>DD</sub> = 4.5-5.5 V
		-5	—	5	%	-10°C to +85°C	V <sub>DD</sub> = 4.5-5.5 V
		-10	+/-1	10	%	-40°C to +85°C	V <sub>DD</sub> = 4.5-5.5 V
<b>INTRC Accuracy @ Freq = 31 kHz<sup>(2)</sup></b>							
	PIC18LF6310/6410/8310/8410	26.562	—	35.938	kHz	-40°C to +85°C	V <sub>DD</sub> = 2.7-3.3 V
	PIC18F6310/6410/8310/8410	26.562	—	35.938	kHz	-40°C to +85°C	V <sub>DD</sub> = 4.5-5.5 V

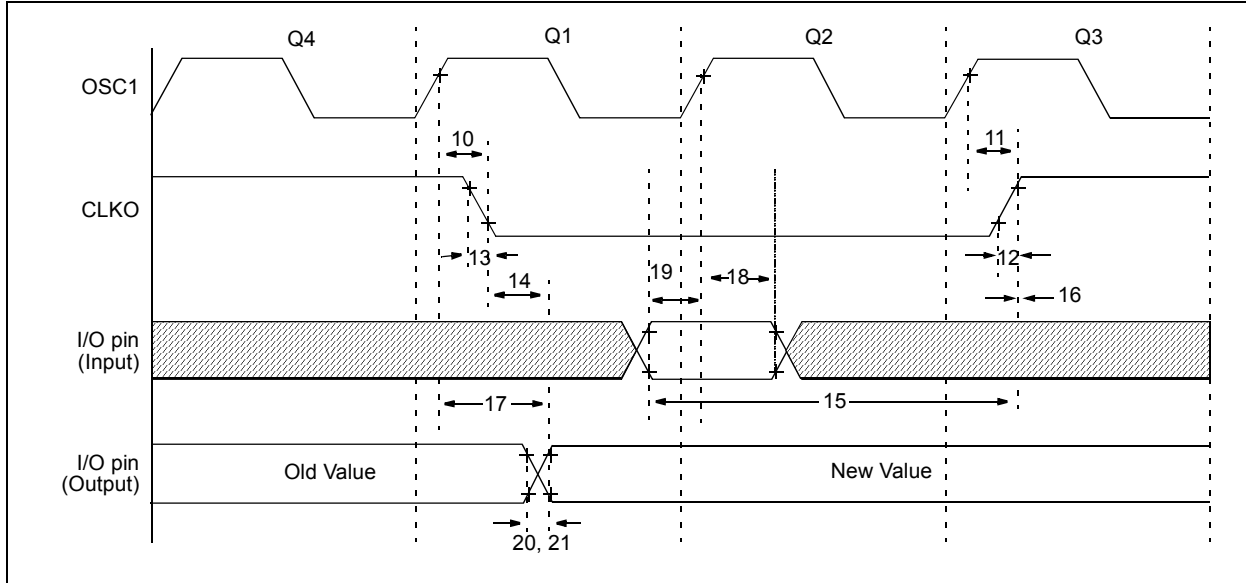
**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** Frequency calibrated at 25°C. OSCTUNE register can be used to compensate for temperature drift.

**2:** INTRC frequency after calibration.

# PIC18F6310/6410/8310/8410

**FIGURE 27-7: CLKO AND I/O TIMING**



**TABLE 27-9: CLKO AND I/O TIMING REQUIREMENTS**

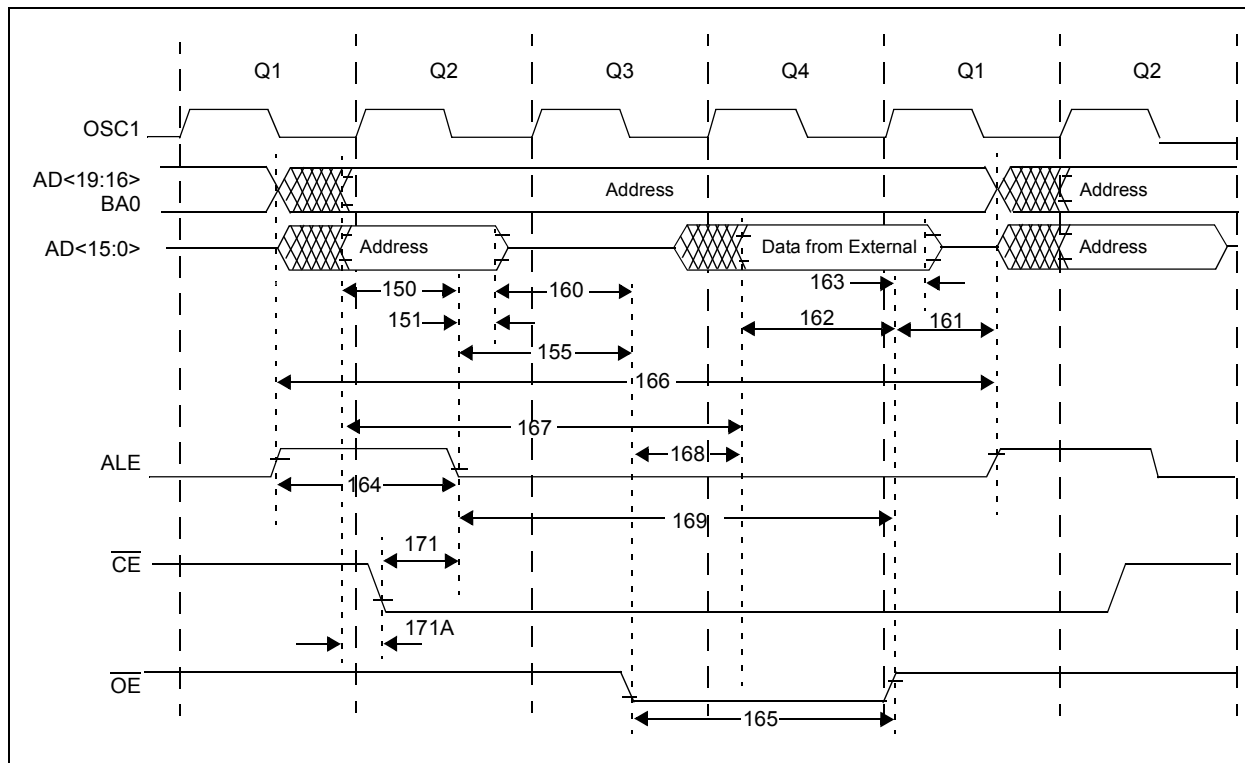
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions	
10	TosH2ckL	OSC1 ↑ to CLKO ↓	—	75	200	ns	(Note 1)	
11	TosH2ckH	OSC1 ↑ to CLKO ↑	—	75	200	ns	(Note 1)	
12	TckR	CLKO Rise Time	—	35	100	ns	(Note 1)	
13	TckF	CLKO Fall Time	—	35	100	ns	(Note 1)	
14	TckL2ioV	CLKO ↓ to Port Out Valid	—	—	0.5 T <sub>CY</sub> + 20	ns	(Note 1)	
15	TioV2ckH	Port In Valid before CLKO ↑	0.25 T <sub>CY</sub> + 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)	PIC18FXXXX	100	—	—	ns	
18A			PIC18LFXXXX	200	—	—	ns	V <sub>DD</sub> = 2.0V
19	TioV2osH	Port Input Valid to OSC1↑ (I/O in setup time)	0	—	—	ns		
20	TioR	Port Output Rise Time	PIC18FXXXX	—	10	25	ns	
20A			PIC18LFXXXX	—	—	60	ns	V <sub>DD</sub> = 2.0V
21	TioF	Port Output Fall Time	PIC18FXXXX	—	10	25	ns	
21A			PIC18LFXXXX	—	—	60	ns	V <sub>DD</sub> = 2.0V
22†	TINP	INTx pin High or Low Time	T <sub>CY</sub>	—	—	ns		
23†	TRBP	RB<7:4> Change INTx High or Low Time	T <sub>CY</sub>	—	—	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 T<sub>osc</sub>.

# PIC18F6310/6410/8310/8410

**FIGURE 27-8: PROGRAM MEMORY READ TIMING DIAGRAM**

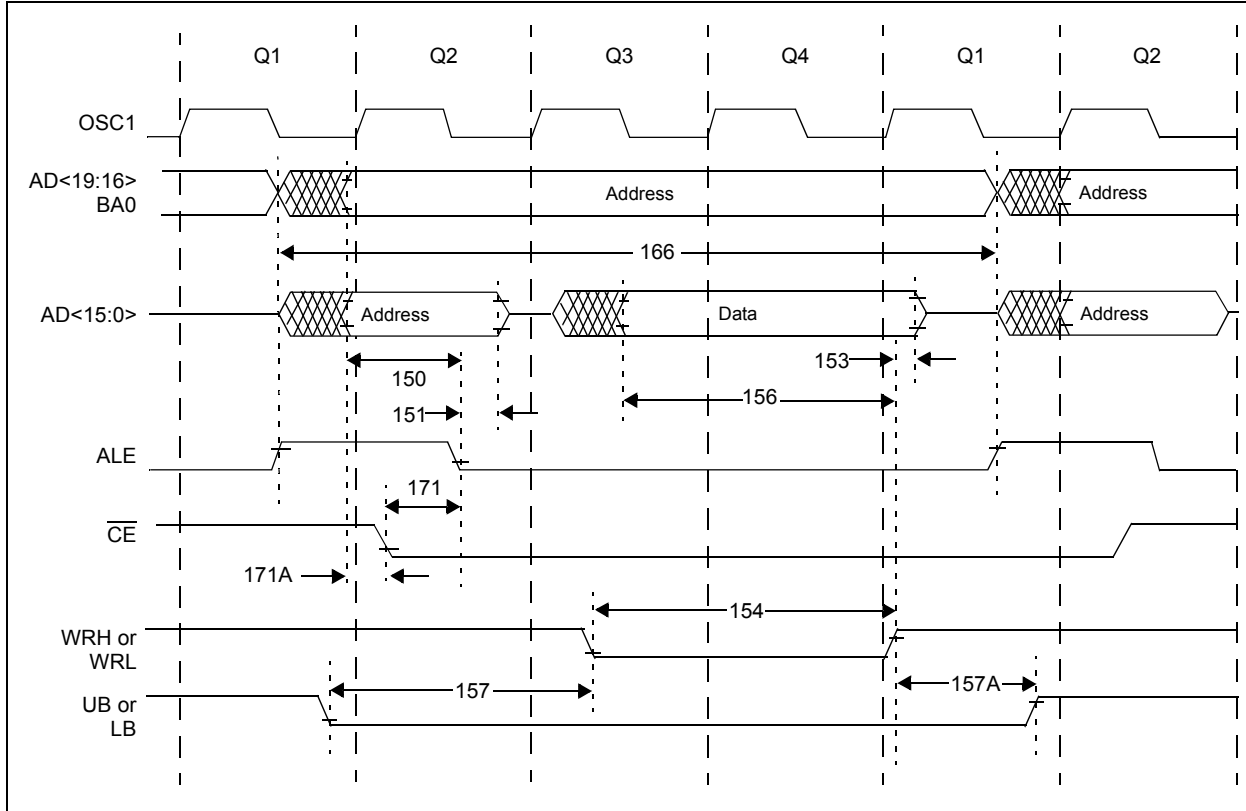


**TABLE 27-10: PROGRAM MEMORY READ TIMING REQUIREMENTS**

Param. No	Symbol	Characteristics	Min	Typ	Max	Units
150	TadV2alL	Address Out Valid to ALE ↓ (address setup time)	$0.25 T_{CY} - 10$	—	—	ns
151	TalL2adI	ALE ↓ to Address Out Invalid (address hold time)	5	—	—	ns
155	TalL2oeL	ALE ↓ to $\overline{OE}$ ↓	10	$0.125 T_{CY}$	—	ns
160	TadZ2oeL	AD high-Z to $\overline{OE}$ ↓ (bus release to $\overline{OE}$ )	0	—	—	ns
161	ToeH2adD	$\overline{OE}$ ↑ to AD Driven	$0.125 T_{CY} - 5$	—	—	ns
162	TadV2oeH	LS Data Valid before $\overline{OE}$ ↑ (data setup time)	20	—	—	ns
163	ToeH2adI	$\overline{OE}$ ↑ to Data In Invalid (data hold time)	0	—	—	ns
164	TalH2alL	ALE Pulse Width	—	$T_{CY}$	—	ns
165	ToeL2oeH	$\overline{OE}$ Pulse Width	$0.5 T_{CY} - 5$	$0.5 T_{CY}$	—	ns
166	TalH2alH	ALE ↑ to ALE ↑ (cycle time)	—	$0.25 T_{CY}$	—	ns
167	Tacc	Address Valid to Data Valid	$0.75 T_{CY} - 25$	—	—	ns
168	Toe	$\overline{OE}$ ↓ to Data Valid	—	—	$0.5 T_{CY} - 25$	ns
169	TalL2oeH	ALE ↓ to $\overline{OE}$ ↑	$0.625 T_{CY} - 10$	—	$0.625 T_{CY} + 10$	ns
171	TalH2csL	Chip Enable Active to ALE ↓	$0.25 T_{CY} - 20$	—	—	ns
171A	TubL2oeH	AD Valid to Chip Enable Active	—	—	10	ns

# PIC18F6310/6410/8310/8410

**FIGURE 27-9: PROGRAM MEMORY WRITE TIMING DIAGRAM**

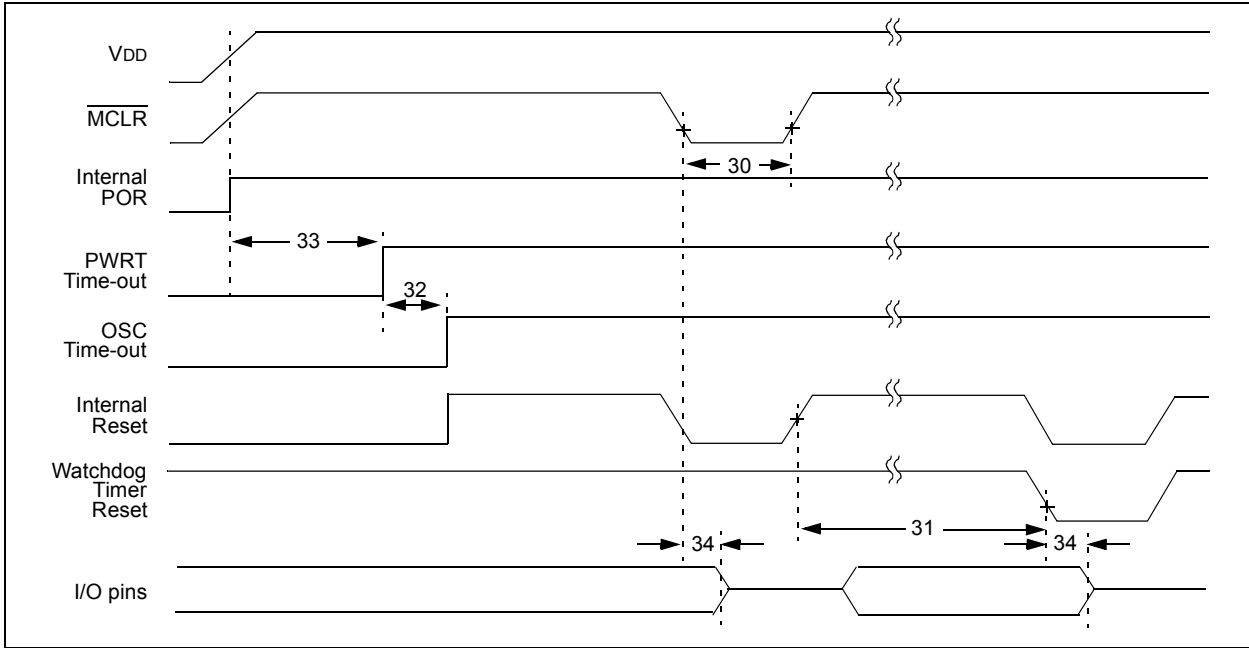


**TABLE 27-11: PROGRAM MEMORY WRITE TIMING REQUIREMENTS**

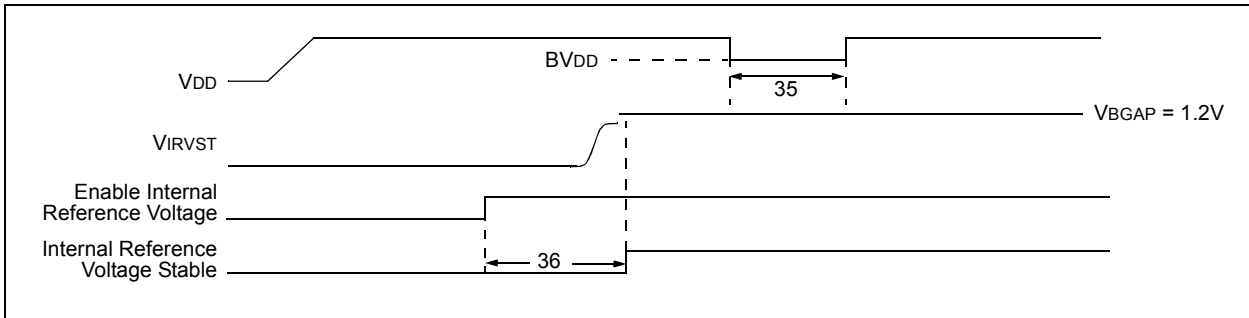
Param. No	Symbol	Characteristics	Min	Typ	Max	Units
150	TadV2alL	Address Out Valid to ALE ↓ (address setup time)	0.25 Tcy - 10	—	—	ns
151	TalL2adI	ALE ↓ to Address Out Invalid (address hold time)	5	—	—	ns
153	TwrH2adI	WRn ↑ to Data Out Invalid (data hold time)	5	—	—	ns
154	TwrL	WRn Pulse Width	0.5 Tcy - 5	0.5 Tcy	—	ns
156	TadV2wrH	Data Valid before WRn ↑ (data setup time)	0.5 Tcy - 10	—	—	ns
157	TbsV2wrL	Byte Select Valid before WRn ↓ (byte select setup time)	0.25 Tcy	—	—	ns
157A	TwrH2bsI	WRn ↑ to Byte Select Invalid (byte select hold time)	0.125 Tcy - 5	—	—	ns
166	TalH2alH	ALE ↑ to ALE ↑ (cycle time)	—	0.25 Tcy	—	ns
171	TalH2csL	Chip Enable Active to ALE ↓	0.25 Tcy - 20	—	—	ns
171A	TubL2oeH	AD Valid to Chip Enable Active	—	—	10	ns

# PIC18F6310/6410/8310/8410

**FIGURE 27-10: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**



**FIGURE 27-11: BROWN-OUT RESET TIMING**

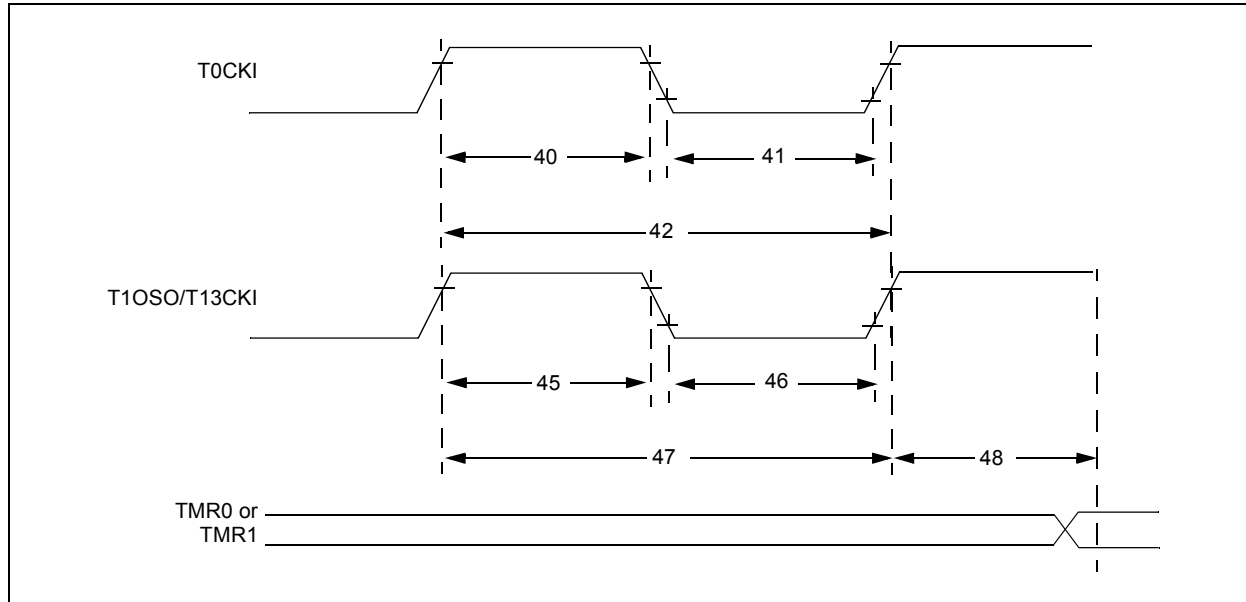


**TABLE 27-12: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
30	T <sub>MCLR</sub>	MCLR Pulse Width (low)	2	—	—	μs	
31	T <sub>WDT</sub>	Watchdog Timer Time-out Period (no postscaler)	3.4	4.1	4.71	ms	
32	T <sub>OST</sub>	Oscillator Start-up Timer Period	1024 T <sub>osc</sub>	—	1024 T <sub>osc</sub>	—	T <sub>osc</sub> = OSC1 period
33	T <sub>PWRT</sub>	Power-up Timer Period	55.5	65.5	75	ms	
34	T <sub>IOZ</sub>	I/O High-Impedance from MCLR Low or Watchdog Timer Reset	—	2	—	μs	
35	T <sub>BOR</sub>	Brown-out Reset Pulse Width	200	—	—	μs	V <sub>DD</sub> ≤ BV <sub>DD</sub> (see D005)
36	T <sub>IRVST</sub>	Time for Internal Reference Voltage to become stable	—	20	50	μs	
37	T <sub>LVDT</sub>	Low-Voltage Detect Pulse Width	200	—	—	μs	V <sub>DD</sub> ≤ VL <sub>VD</sub>
38	T <sub>CSD</sub>	CPU Start-up Time	—	10	—	μs	
39	T <sub>IOBST</sub>	Time for INTRC Block to stabilize	—	1	—	ms	

# PIC18F6310/6410/8310/8410

**FIGURE 27-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**



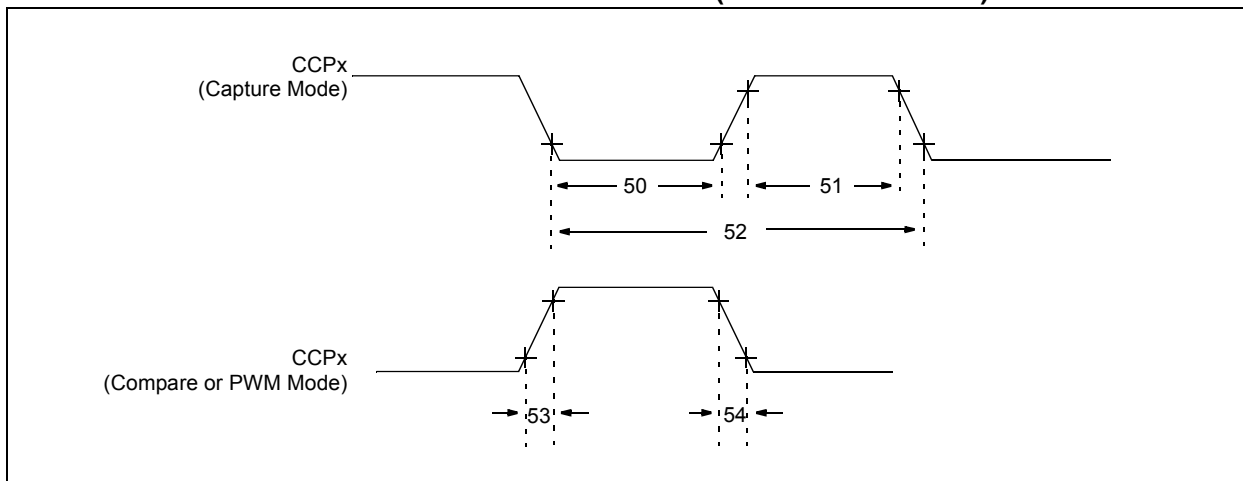
**TABLE 27-13: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions	
40	T <sub>T0H</sub>	T0CKI High Pulse Width	No prescaler	$0.5 T_{CY} + 20$	—	ns		
			With prescaler	10	—	ns		
41	T <sub>T0L</sub>	T0CKI Low Pulse Width	No prescaler	$0.5 T_{CY} + 20$	—	ns		
			With prescaler	10	—	ns		
42	T <sub>T0P</sub>	T0CKI Period	No prescaler	$T_{CY} + 10$	—	ns		
			With prescaler	Greater of: 20 ns or $(T_{CY} + 40)/N$	—	ns		N = prescale value (1, 2, 4, ..., 256)
45	T <sub>T1H</sub>	T13CKI High Time	Synchronous, no prescaler	$0.5 T_{CY} + 20$	—	ns		
			Synchronous, with prescaler	PIC18FXXXX	10	—		ns
				PIC18LFXXXX	25	—		ns
			Asynchronous	PIC18FXXXX	30	—		ns
PIC18LFXXXX	50	—		ns				
46	T <sub>T1L</sub>	T13CKI Low Time	Synchronous, no prescaler	$0.5 T_{CY} + 5$	—	ns		
			Synchronous, with prescaler	PIC18FXXXX	10	—		ns
				PIC18LFXXXX	25	—		ns
			Asynchronous	PIC18FXXXX	30	—		ns
PIC18LFXXXX	50	—		ns				
47	T <sub>T1P</sub>	T13CKI Input Period	Synchronous	Greater of: 20 ns or $(T_{CY} + 40)/N$	—	ns	N = prescale value (1, 2, 4, 8)	
			Asynchronous	60	—	ns		
	F <sub>T1</sub>	T13CKI Oscillator Input Frequency Range		DC	50	kHz		
48	T <sub>CKE2TMR1</sub>	Delay from External T13CKI Clock Edge to Timer Increment		$2 T_{OSC}$	$7 T_{OSC}$	—		



# PIC18F6310/6410/8310/8410

**FIGURE 27-13: CAPTURE/COMPARE/PWM TIMINGS (ALL CCP MODULES)**

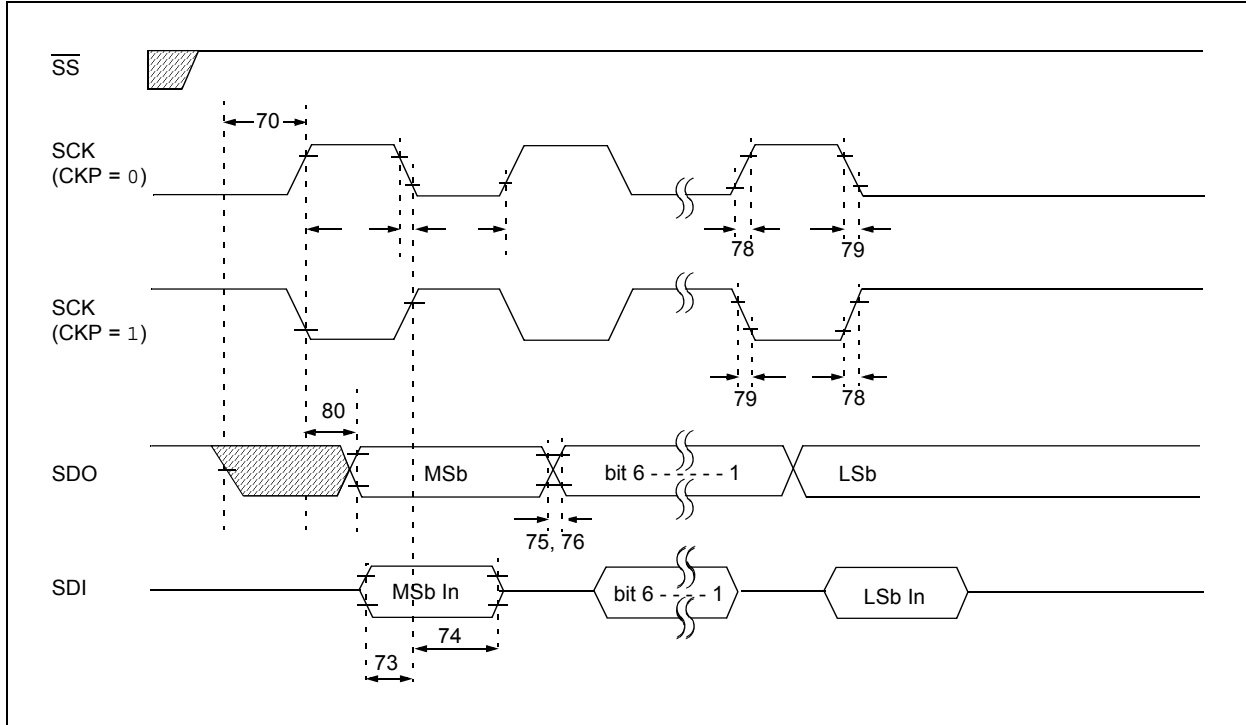


**TABLE 27-14: CAPTURE/COMPARE/PWM REQUIREMENTS (ALL CCP MODULES)**

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
50	TccL	CCPx Input Low Time	No prescaler	$0.5 T_{CY} + 20$	—	ns	
			With prescaler	PIC18FXXXXX	10	—	
			PIC18LFXXXXX	20	—	ns	VDD = 2.0V
51	TccH	CCPx Input High Time	No prescaler	$0.5 T_{CY} + 20$	—	ns	
			With prescaler	PIC18FXXXXX	10	—	
			PIC18LFXXXXX	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	PIC18FXXXXX	—	25	ns	
			PIC18LFXXXXX	—	45	ns	
							VDD = 2.0V
54	TccF	CCPx Output Fall Time	PIC18FXXXXX	—	25	ns	
			PIC18LFXXXXX	—	45	ns	
							VDD = 2.0V

# PIC18F6310/6410/8310/8410

**FIGURE 27-14: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)**

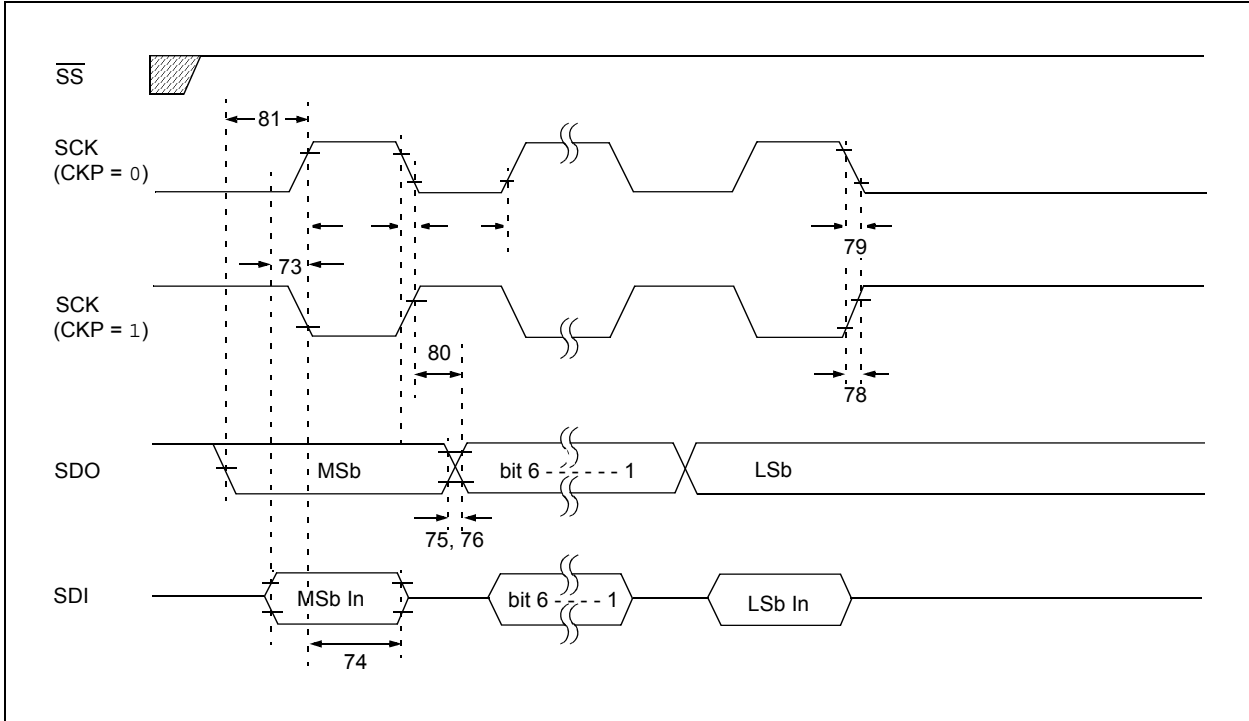


**TABLE 27-15: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)**

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions	
70	TssL2sCH, TssL2sCL	$\overline{SS} \downarrow$ to SCK $\downarrow$ or SCK $\uparrow$ Input	Tcy	—	ns		
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	40	—	ns		
75	TdoR	SDO Data Output Rise Time	PIC18FXXXX	—	25	ns	
			PIC18LFXXXX	—	45	ns	VDD = 2.0V
76	TdoF	SDO Data Output Fall Time	—	25	ns		
78	TscR	SCK Output Rise Time	PIC18FXXXX	—	25	ns	
			PIC18LFXXXX	—	45	ns	VDD = 2.0V
79	TscF	SCK Output Fall Time	—	25	ns		
80	Tsch2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXXXX	—	50	ns	
			PIC18LFXXXX	—	100	ns	VDD = 2.0V

# PIC18F6310/6410/8310/8410

**FIGURE 27-15: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)**

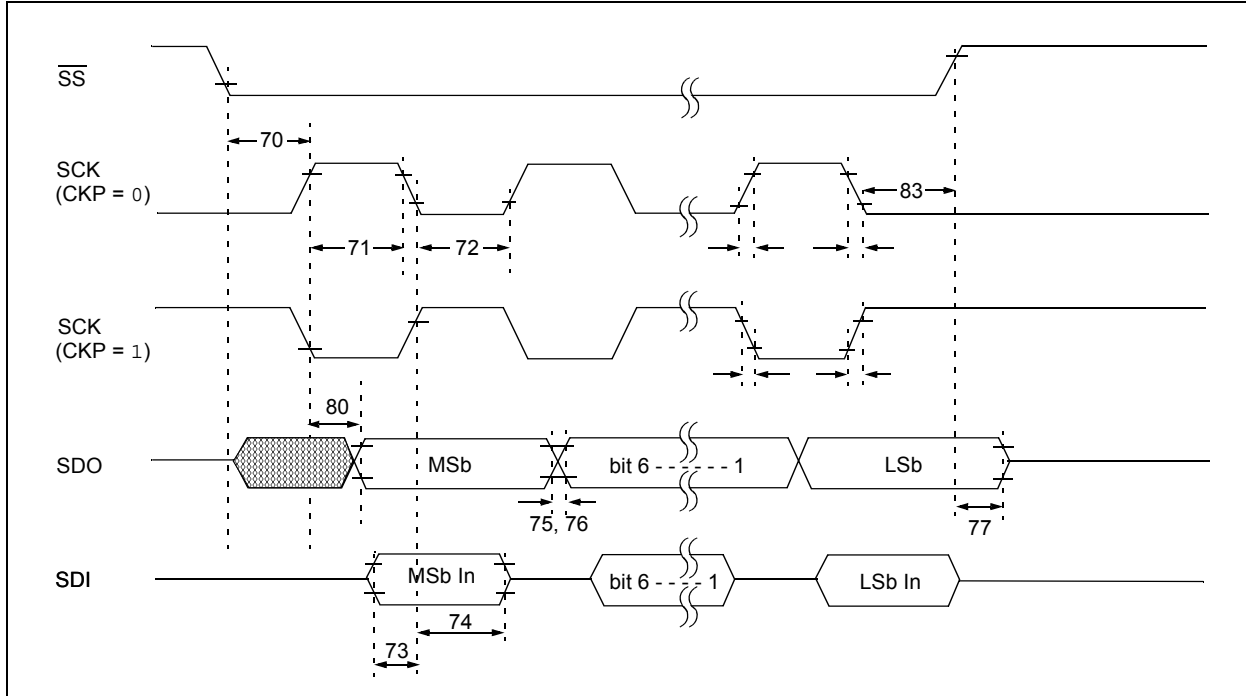


**TABLE 27-16: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions	
73	TdIV2sCH, TdIV2sCL	Setup Time of SDI Data Input to SCK Edge	20	—	ns		
74	TsCH2dIL, TsCL2dIL	Hold Time of SDI Data Input to SCK Edge	40	—	ns		
75	TdoR	SDO Data Output Rise Time	PIC18FXXXX	—	25	ns	V <sub>DD</sub> = 2.0V
			PIC18LFXXXX	—	45	ns	
76	TdoF	SDO Data Output Fall Time	—	25	ns		
78	TscR	SCK Output Rise Time	PIC18FXXXX	—	25	ns	V <sub>DD</sub> = 2.0V
			PIC18LFXXXX	—	45	ns	
79	TscF	SCK Output Fall Time	—	25	ns		
80	TsCH2doV, TsCL2doV	SDO Data Output Valid after SCK Edge	PIC18FXXXX	—	50	ns	V <sub>DD</sub> = 2.0V
			PIC18LFXXXX	—	100	ns	
81	TdoV2sCH, TdoV2sCL	SDO Data Output Setup to SCK Edge	T <sub>cy</sub>	—	ns		

# PIC18F6310/6410/8310/8410

**FIGURE 27-16: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)**



**TABLE 27-17: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)**

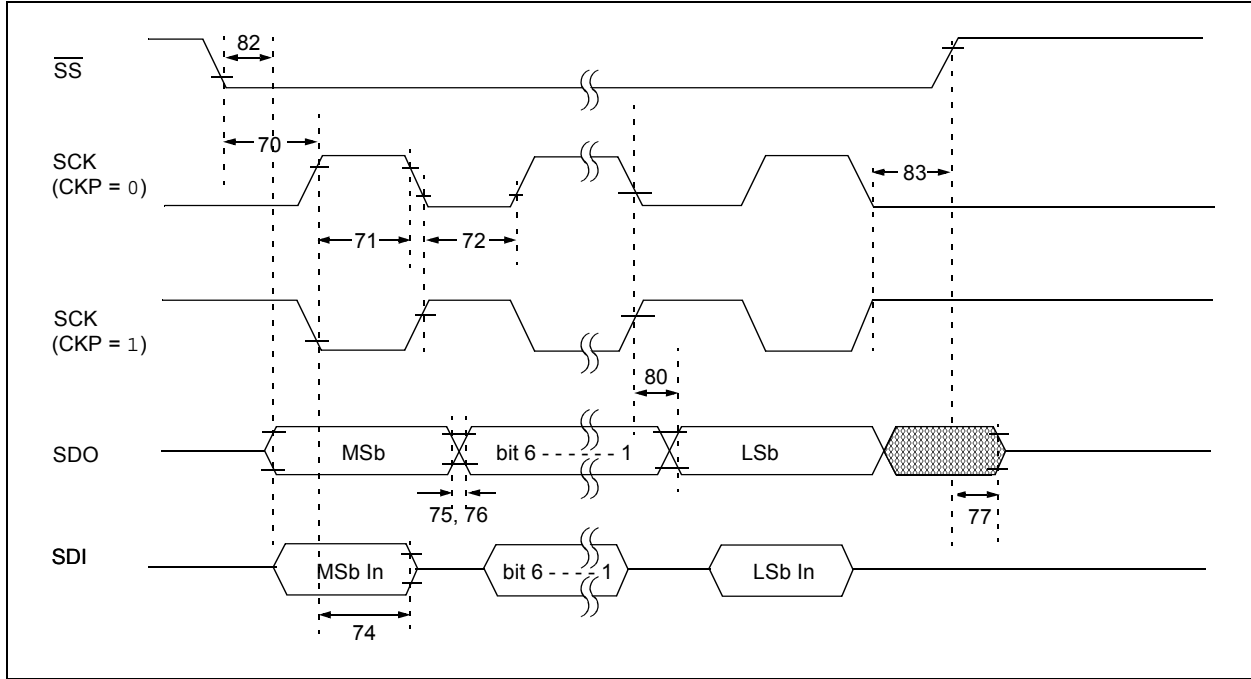
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}$ $\downarrow$ to SCK $\downarrow$ or SCK $\uparrow$ Input	T <sub>CY</sub>	—	ns	
71	Tsch	SCK Input High Time	Continuous	1.25 T <sub>CY</sub> + 30	—	ns
71A			Single Byte	40	—	ns
72	TscL	SCK Input Low Time	Continuous	1.25 T <sub>CY</sub> + 30	—	ns
72A			Single Byte	40	—	ns
73	TdIV2scH, TdIV2scL	Setup Time of SDI Data Input to SCK Edge	20	—	ns	
73A	TB2B	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2	1.5 T <sub>CY</sub> + 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	PIC18FXXXX	—	25	ns
76			PIC18LFXXXX	—	45	ns
76	TdoF	SDO Data Output Fall Time	—	25	ns	
77	TssH2doZ	$\overline{SS}$ $\uparrow$ to SDO Output High-impedance	10	50	ns	
80	Tsch2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXXXX	—	50	ns
83			PIC18LFXXXX	—	100	ns
83	Tsch2ssH, TscL2ssH	$\overline{SS}$ $\uparrow$ after SCK Edge	1.5 T <sub>CY</sub> + 40	—	ns	

**Note 1:** Requires the use of Parameter #73A.

**2:** Only if Parameter #71A and #72A are used.

# PIC18F6310/6410/8310/8410

**FIGURE 27-17: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)**



**TABLE 27-18: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)**

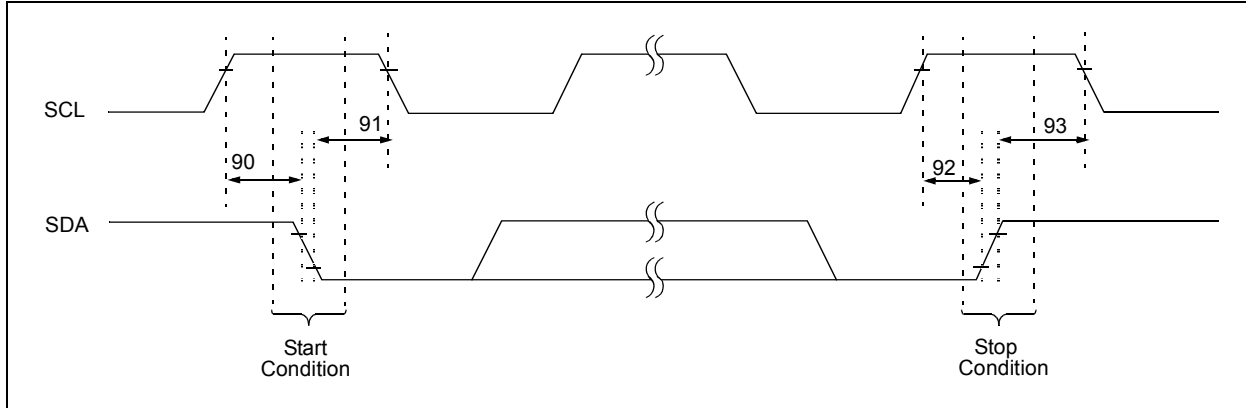
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
70	TssL2scl, TssL2sch	$\overline{SS}$ $\downarrow$ to SCK $\downarrow$ or SCK $\uparrow$ Input	T <sub>cy</sub>	—	ns	
71	Tsch	SCK Input High Time	1.25 T <sub>cy</sub> + 30	—	ns	
71A		Single Byte	40	—	ns	(Note 1)
72	Tscl	SCK Input Low Time	1.25 T <sub>cy</sub> + 30	—	ns	
72A		Single Byte	40	—	ns	(Note 1)
73A	Tb2b	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2	1.5 T <sub>cy</sub> + 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	PIC18FXXXX —	25	ns	
76	TDOF	SDO Data Output Fall Time	—	25	ns	V <sub>DD</sub> = 2.0V
77	TssH2doZ	$\overline{SS}$ $\uparrow$ to SDO Output High-Impedance	10	50	ns	
80	Tsch2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXXXX —	50 100	ns	V <sub>DD</sub> = 2.0V
82	TssL2doV	SDO Data Output Valid after $\overline{SS}$ $\downarrow$ Edge	PIC18FXXXX —	50 100	ns	V <sub>DD</sub> = 2.0V
83	Tsch2ssH, TscL2ssH	$\overline{SS}$ $\uparrow$ after SCK Edge	1.5 T <sub>cy</sub> + 40	—	ns	

**Note 1:** Requires the use of Parameter #73A.

**2:** Only if Parameter #71A and #72A are used.

# PIC18F6310/6410/8310/8410

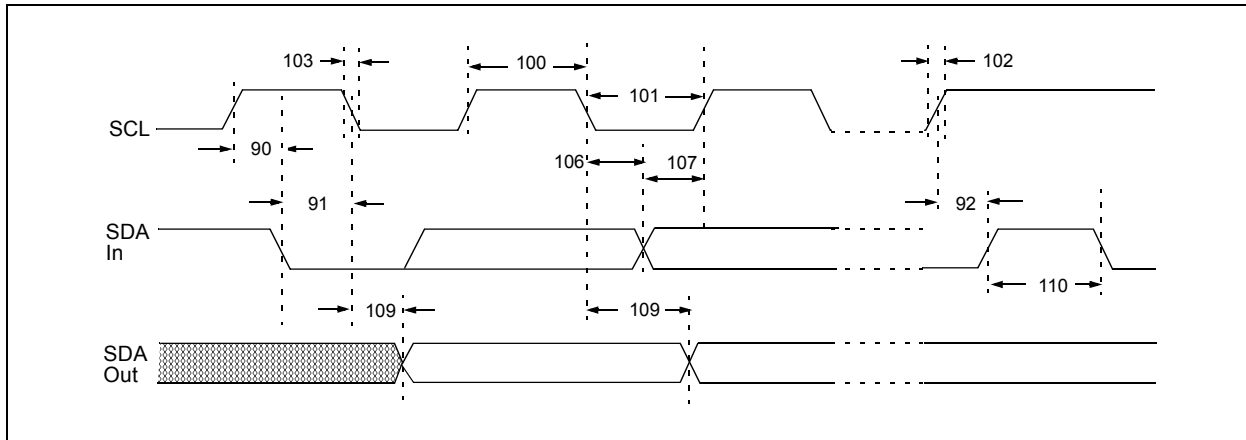
**FIGURE 27-18: I<sup>2</sup>C™ BUS START/STOP BITS TIMING**



**TABLE 27-19: I<sup>2</sup>C™ BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions	
90	TSU:STA	Start Condition Setup Time	100 kHz mode	4700	—	ns	Only relevant for Repeated Start condition
			400 kHz mode	600	—		
91	THD:STA	Start Condition Hold Time	100 kHz mode	4000	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—		
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	4700	—	ns	
			400 kHz mode	600	—		
93	THD:STO	Stop Condition Hold Time	100 kHz mode	4000	—	ns	
			400 kHz mode	600	—		

**FIGURE 27-19: I<sup>2</sup>C™ BUS DATA TIMING**



# PIC18F6310/6410/8310/8410

**TABLE 27-20: I<sup>2</sup>C™ BUS DATA REQUIREMENTS (SLAVE MODE)**

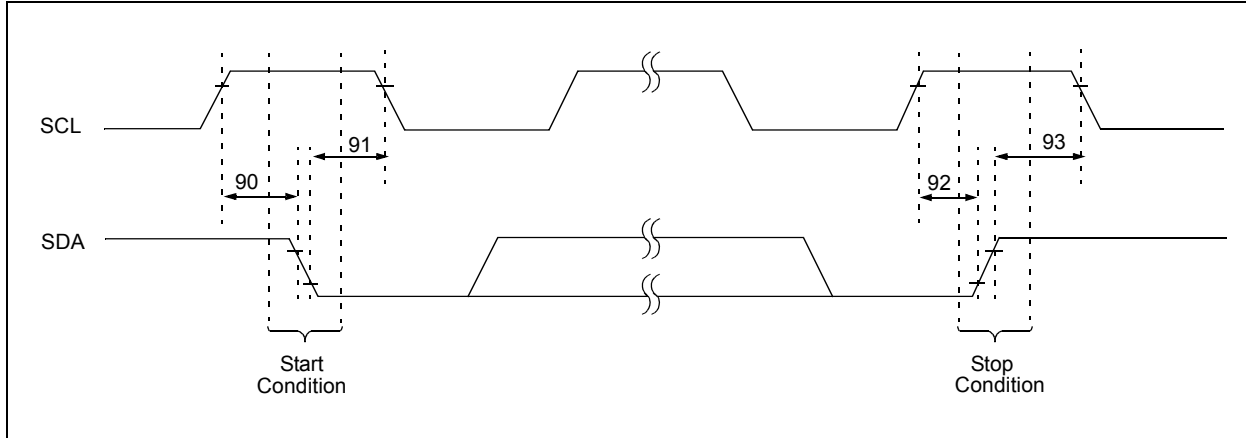
Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions	
100	THIGH	Clock High Time	100 kHz mode	4.0	—	μs	PIC18FXXXX must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	μs	PIC18FXXXX must operate at a minimum of 10 MHz
			MSSP Module	1.5 T <sub>CY</sub>	—		
101	TLOW	Clock Low Time	100 kHz mode	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 T <sub>CY</sub>	—		
102	TR	SDA and SCL Rise Time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1 C <sub>B</sub>	300	ns	C <sub>B</sub> 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 C <sub>B</sub>	300	ns	C <sub>B</sub> 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	<b>(Note 2)</b>
			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	<b>(Note 1)</b>
			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 I<sup>2</sup>C bus device can be used in a Standard mode I<sup>2</sup>C 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 I<sup>2</sup>C bus specification), before the SCL line is released.

# PIC18F6310/6410/8310/8410

**FIGURE 27-20: MASTER SSP I<sup>2</sup>C™ BUS START/STOP BITS TIMING WAVEFORMS**

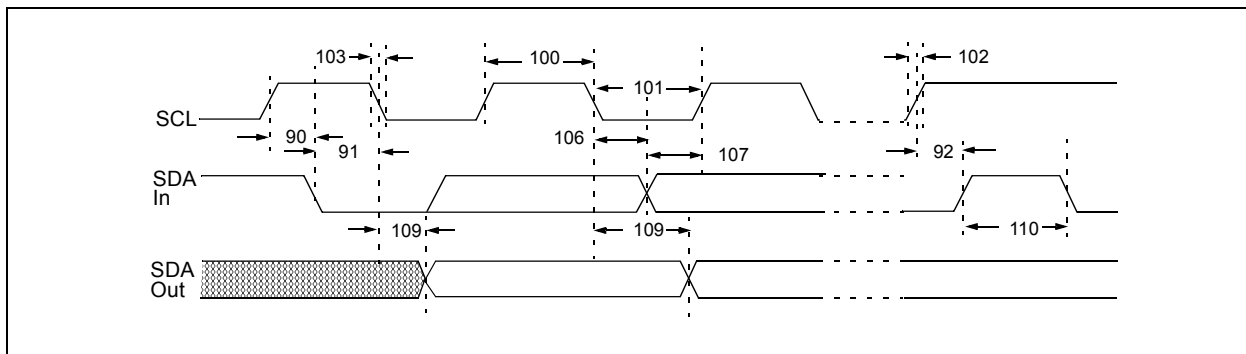


**TABLE 27-21: MASTER SSP I<sup>2</sup>C™ BUS START/STOP BITS REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
90	TSU:STA	Start Condition Setup Time	100 kHz mode	$2(T_{osc})(BRG + 1)$	—	ns Only relevant for Repeated Start condition
			400 kHz mode	$2(T_{osc})(BRG + 1)$	—	
			1 MHz mode <sup>(1)</sup>	$2(T_{osc})(BRG + 1)$	—	
91	THD:STA	Start Condition Hold Time	100 kHz mode	$2(T_{osc})(BRG + 1)$	—	ns After this period, the first clock pulse is generated
			400 kHz mode	$2(T_{osc})(BRG + 1)$	—	
			1 MHz mode <sup>(1)</sup>	$2(T_{osc})(BRG + 1)$	—	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	$2(T_{osc})(BRG + 1)$	—	ns
			400 kHz mode	$2(T_{osc})(BRG + 1)$	—	
			1 MHz mode <sup>(1)</sup>	$2(T_{osc})(BRG + 1)$	—	
93	THD:STO	Stop Condition Hold Time	100 kHz mode	$2(T_{osc})(BRG + 1)$	—	ns
			400 kHz mode	$2(T_{osc})(BRG + 1)$	—	
			1 MHz mode <sup>(1)</sup>	$2(T_{osc})(BRG + 1)$	—	

**Note 1:** Maximum pin capacitance = 10 pF for all I<sup>2</sup>C pins.

**FIGURE 27-21: MASTER SSP I<sup>2</sup>C™ BUS DATA TIMING**





# PIC18F6310/6410/8310/8410

**TABLE 27-22: MASTER SSP I<sup>2</sup>C™ BUS DATA REQUIREMENTS**

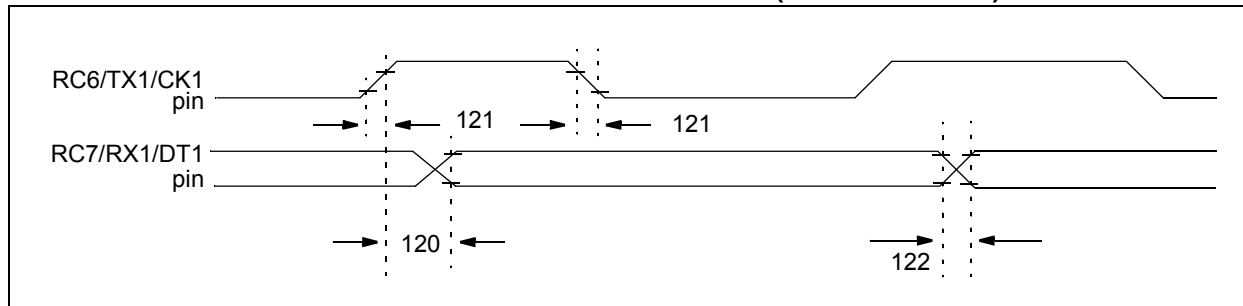
Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	ms
101	TLOW	Clock Low Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	ms
102	TR	SDA and SCL Rise Time	100 kHz mode	—	1000	ns
			400 kHz mode	20 + 0.1 C <sub>B</sub>	300	ns
			1 MHz mode <sup>(1)</sup>	—	300	ns
103	TF	SDA and SCL Fall Time	100 kHz mode	—	300	ns
			400 kHz mode	20 + 0.1 C <sub>B</sub>	300	ns
			1 MHz mode <sup>(1)</sup>	—	100	ns
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	ms
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms
			1 MHz mode <sup>(1)</sup>	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
			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 <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	ms
109	TAA	Output Valid from Clock	100 kHz mode	—	3500	ns
			400 kHz mode	—	1000	ns
			1 MHz mode <sup>(1)</sup>	—	—	ns
110	TBUF	Bus Free Time	100 kHz mode	4.7	—	ms
			400 kHz mode	1.3	—	ms
D102	CB	Bus Capacitive Loading	—	400	pF	

**Note 1:** Maximum pin capacitance = 10 pF for all I<sup>2</sup>C pins.

**2:** A Fast mode I<sup>2</sup>C bus device can be used in a Standard mode I<sup>2</sup>C 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.

# PIC18F6310/6410/8310/8410

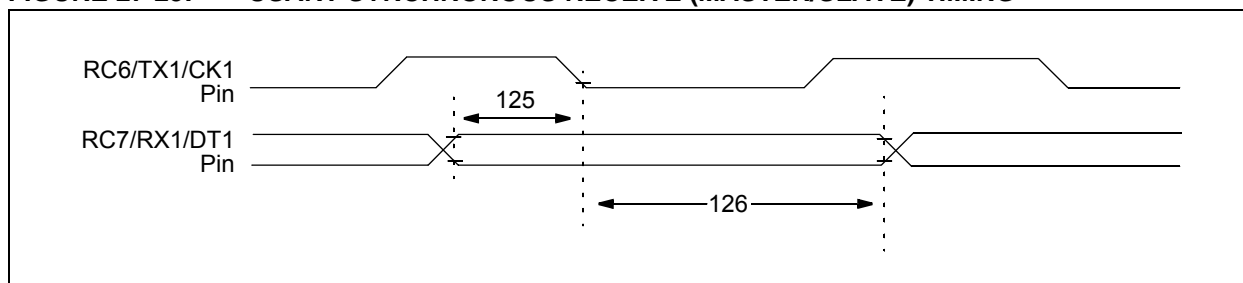
**FIGURE 27-22: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**



**TABLE 27-23: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
120	T <sub>CKH2DTV</sub>	SYNC XMIT (MASTER and SLAVE) Clock High to Data Out Valid	PIC18FXXXX	—	40	ns
			PIC18LFXXXX	—	100	ns
121	T <sub>CKRF</sub>	Clock Out Rise Time and Fall Time (Master mode)	PIC18FXXXX	—	20	ns
			PIC18LFXXXX	—	50	ns
122	T <sub>DTRF</sub>	Data Out Rise Time and Fall Time	PIC18FXXXX	—	20	ns
			PIC18LFXXXX	—	50	ns

**FIGURE 27-23: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**



**TABLE 27-24: USART SYNCHRONOUS RECEIVE REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	T <sub>DTV2CKL</sub>	SYNC RCV (MASTER and SLAVE) Data Hold before CKx ↓ (DTx hold time)	10	—	ns	
126	T <sub>CKL2DTL</sub>	Data Hold after CKx ↓ (DTx hold time)	15	—	ns	

# PIC18F6310/6410/8310/8410

**TABLE 27-25: A/D CONVERTER CHARACTERISTICS: PIC18F6310/6410/8310/8410 (INDUSTRIAL)  
PIC18LF6310/6410/8310/8410 (INDUSTRIAL)**

Param No.	Sym	Characteristic	Min	Typ	Max	Units	Conditions	
A01	NR	Resolution	—	—	10	bit	$\Delta V_{REF} \geq 3.0V$	
A03	EIL	Integral Linearity Error	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$	
A04	EDL	Differential Linearity Error	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$	
A06	E <sub>OFF</sub>	Offset Error	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$	
A07	E <sub>GN</sub>	Gain Error	—	—	$< \pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$	
A10	—	Monotonicity	Guaranteed <sup>(1)</sup>			—		
A20	$\Delta V_{REF}$	Reference Voltage Range (V <sub>REFH</sub> – V <sub>REFL</sub> )	3	—	AV <sub>DD</sub> – AV <sub>SS</sub>	V	V <sub>DD</sub> ≥ 3.0V	
			1.8	—	V <sub>DD</sub> – V <sub>SS</sub>	V	V <sub>DD</sub> < 3.0V	
A21	V <sub>REFH</sub>	Reference Voltage High	AV <sub>SS</sub> + $\Delta V_{REF}$	—	AV <sub>DD</sub>	V	For 10-bit resolution	
A22	V <sub>REFL</sub>	Reference Voltage Low	AV <sub>SS</sub>	—	AV <sub>DD</sub> – $\Delta V_{REF}$	V	For 10-bit resolution	
A25	V <sub>AIN</sub>	Analog Input Voltage	V <sub>REFL</sub>	—	V <sub>REFH</sub>	V		
A28	AV <sub>DD</sub>	Analog Supply Voltage	V <sub>DD</sub> – 0.3	—	V <sub>DD</sub> + 0.3	V		
A29	AV <sub>SS</sub>	Analog Supply Voltage	V <sub>SS</sub> – 0.3	—	V <sub>SS</sub> + 0.3	V		
A30	Z <sub>AIN</sub>	Recommended Impedance of Analog Voltage Source	—	—	2.5	k $\Omega$		
A40	I <sub>AD</sub>	A/D Conversion Current (V <sub>DD</sub> )	PIC18FXXXX	—	180	—	$\mu A$	Average current consumption when A/D is on ( <b>Note 2</b> )
			PIC18LFXXXX	—	90	—	$\mu A$	V <sub>DD</sub> = 2.0V; Average current consumption when A/D is on ( <b>Note 2</b> )
A50	I <sub>REF</sub>	V <sub>REF</sub> Input Current ( <b>Note 3</b> )	—	—	$\pm 5$	$\mu A$	During V <sub>AIN</sub> acquisition. During A/D conversion cycle.	
			—	—	$\pm 150$	$\mu A$		

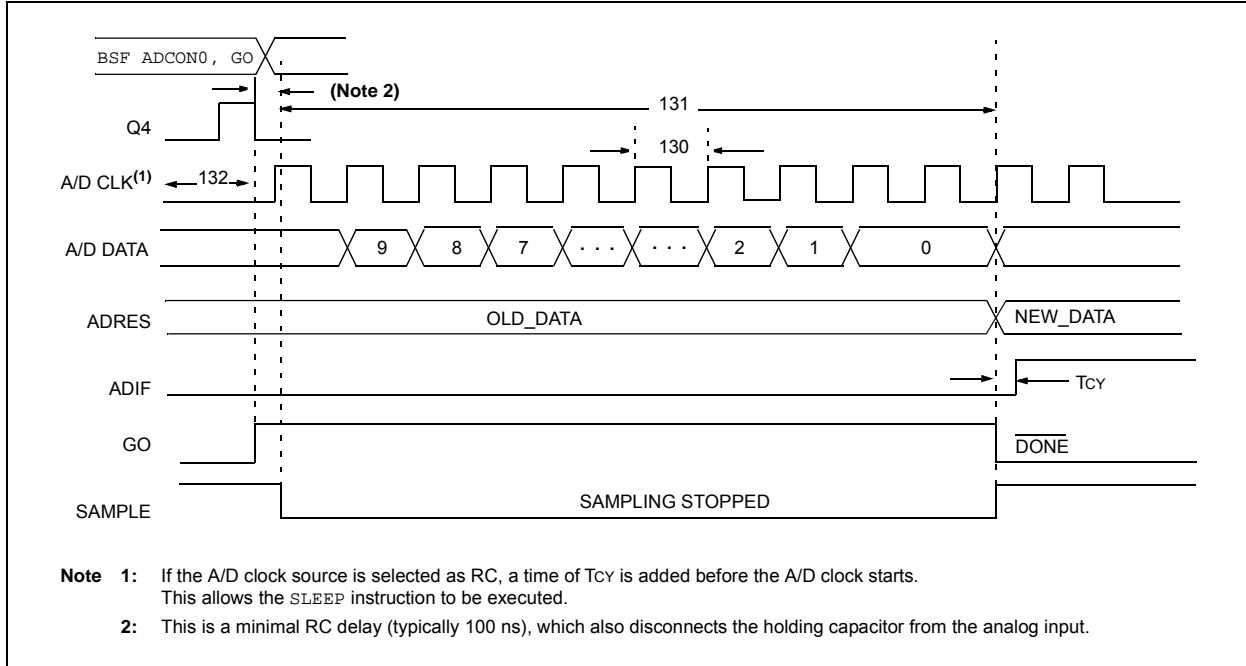
**Note 1:** The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

**2:** When A/D is off, it will not consume any current other than minor leakage current. The power-down current specification includes any such leakage from the A/D module.

**3:** V<sub>REFH</sub> current is from the RA3/AN3/V<sub>REF+</sub> pin or AV<sub>DD</sub>, whichever is selected as the V<sub>REFH</sub> source. V<sub>REFL</sub> current is from the RA2/AN2/V<sub>REF-</sub> pin or AV<sub>SS</sub>, whichever is selected as the V<sub>REFL</sub> source.

# PIC18F6310/6410/8310/8410

**FIGURE 27-24: A/D CONVERSION TIMING**



**TABLE 27-26: A/D CONVERSION REQUIREMENTS**

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions	
130	TAD	A/D Clock Period	PIC18FXXXX	0.7	25.0 <sup>(1)</sup>	μS	TOSC based, VREF ≥ 3.0V
			PIC18LFXXXX	1.4	25.0 <sup>(1)</sup>	μS	VDD = 2.0V; TOSC based, VREF full range
			PIC18FXXXX	—	1	μS	A/D RC mode
			PIC18LFXXXX	—	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)			
137	Tdis	Discharge Time	0.2	—	μS		

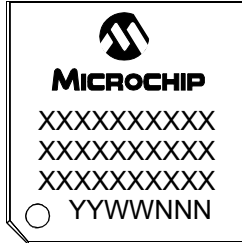
- Note 1:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.
- Note 2:** ADRES register may be read on the following Tcy cycle.
- Note 3:** The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (AVDD to AVSS or AVSS to AVDD). The source impedance (Rs) on the input channels is 50Ω.
- Note 4:** On the following cycle of the device clock.

# PIC18F6310/6410/8310/8410

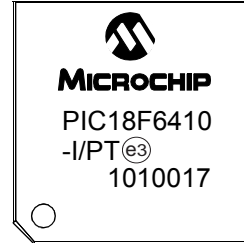
## 28.0 PACKAGING INFORMATION

### 28.1 Package Marking Information

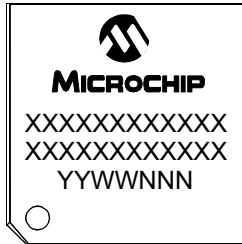
64-Lead TQFP



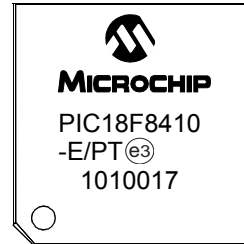
Example



80-Lead TQFP



Example



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

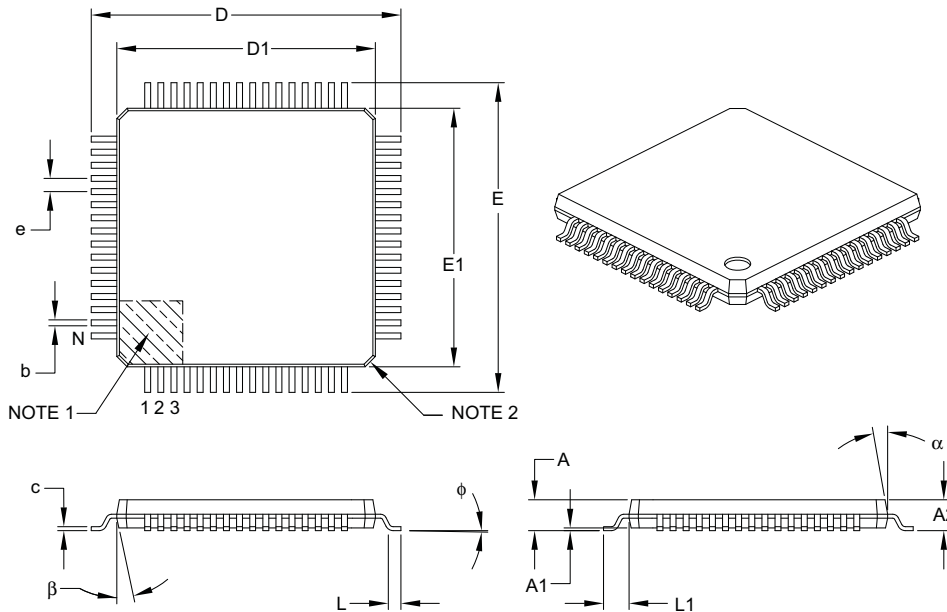
# PIC18F6310/6410/8310/8410

## 28.2 Package Details

The following sections give the technical details of the packages.

### 64-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>



Dimension Limits	Units	MILLIMETERS		
		MIN	NOM	MAX
Number of Leads	N	64		
Lead Pitch	e	0.50 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	$\phi$	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	0.17	0.22	0.27
Mold Draft Angle Top	$\alpha$	11°	12°	13°
Mold Draft Angle Bottom	$\beta$	11°	12°	13°

**Notes:**

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Chamfers at corners are optional; size may vary.
- Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
- Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

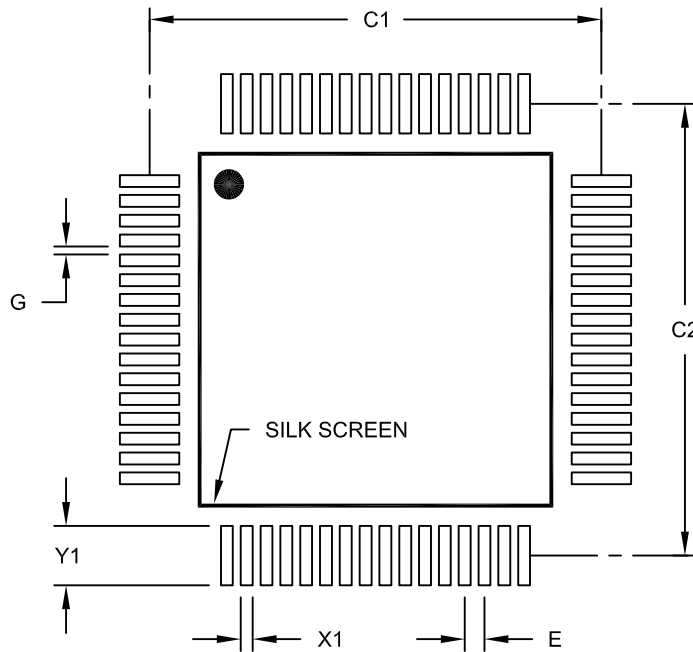
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-085B

# PIC18F6310/6410/8310/8410

## 64-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		
		MIN	NOM	MAX
Dimension Limits				
Contact Pitch	E	0.50 BSC		
Contact Pad Spacing	C1		11.40	
Contact Pad Spacing	C2		11.40	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			1.50
Distance Between Pads	G	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

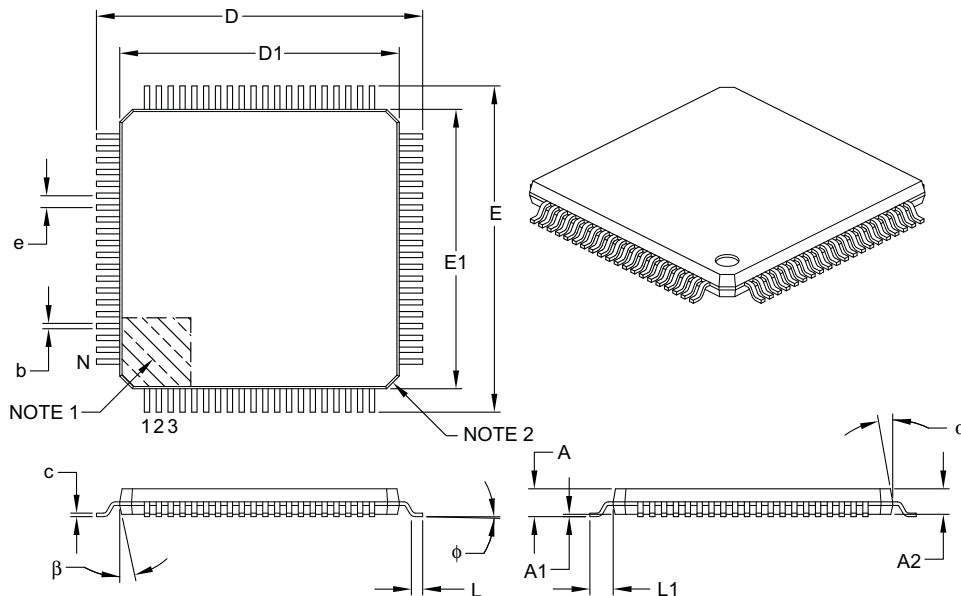
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2085A

# PIC18F6310/6410/8310/8410

## 80-Lead Plastic Thin Quad Flatpack (PT) – 12x12x1 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	80		
Lead Pitch	e	0.50 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	$\phi$	0°	3.5°	7°
Overall Width	E	14.00 BSC		
Overall Length	D	14.00 BSC		
Molded Package Width	E1	12.00 BSC		
Molded Package Length	D1	12.00 BSC		
Lead Thickness	c	0.09	–	0.20
Lead Width	b	0.17	0.22	0.27
Mold Draft Angle Top	$\alpha$	11°	12°	13°
Mold Draft Angle Bottom	$\beta$	11°	12°	13°

### Notes:

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Chamfers at corners are optional; size may vary.
- Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
- 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.

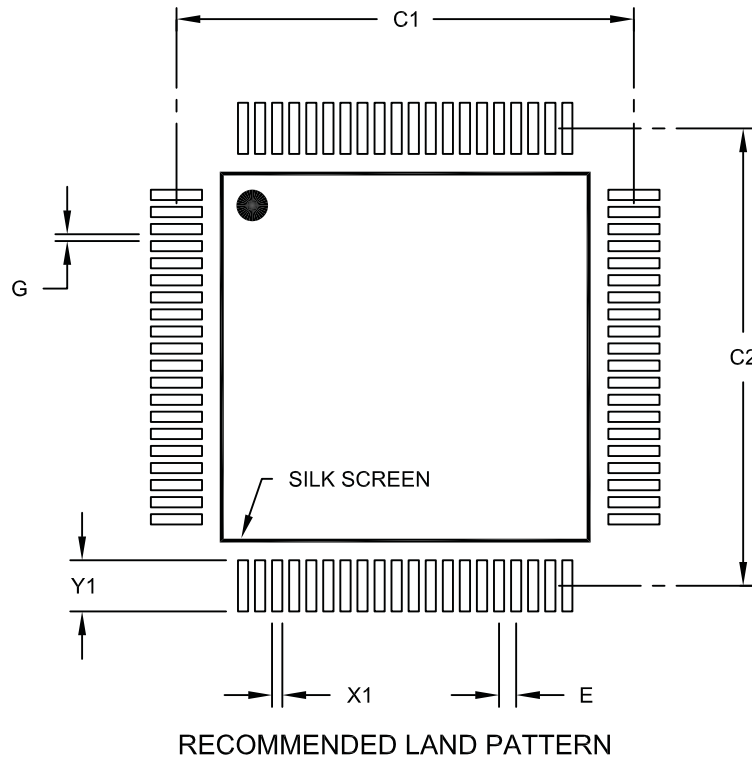
Microchip Technology Drawing C04-092B



# PIC18F6310/6410/8310/8410

## 80-Lead Plastic Thin Quad Flatpack (PT) – 12x12x1 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>



		MILLIMETERS		
		MIN	NOM	MAX
	Units			
	Dimension Limits			
Contact Pitch	E	0.50 BSC		
Contact Pad Spacing	C1		13.40	
Contact Pad Spacing	C2		13.40	
Contact Pad Width (X80)	X1			0.30
Contact Pad Length (X80)	Y1			1.50
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. C04-2092A

# PIC18F6310/6410/8310/8410

---

NOTES:

# PIC18F6310/6410/8310/8410

## APPENDIX A: REVISION HISTORY

### Revision A (June 2004)

Original data sheet for PIC18F6310/6410/8310/8410 devices.

### Revision B (May 2007)

Updated Electrical Characteristics and packaging diagrams.

### Revision C (October 2010)

Changes to electricals in **Section 27.0 “Electrical Characteristics”** and minor text edits throughout document.

## 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	PIC18F6310	PIC18F6410	PIC18F8310	PIC18F8410
Program Memory (Bytes)	8K	16K	8K	16K
Program Memory (Instructions)	4096	8192	4096	8192
External Memory Interface	No	No	Yes	Yes
I/O Ports	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G, H, J	Ports A, B, C, D, E, F, G, H, J
Packages	64-Pin TQFP	64-Pin TQFP	80-Pin TQFP	80-Pin TQFP

# PIC18F6310/6410/8310/8410

---

## APPENDIX C: CONVERSION CONSIDERATIONS

This appendix discusses the considerations for converting from previous versions of a device to the ones listed in this data sheet. Typically, these changes are due to the differences in the process technology used. An example of this type of conversion is from a PIC16C74A to a PIC16C74B.

**Not Applicable**

## APPENDIX D: MIGRATION FROM BASELINE TO ENHANCED DEVICES

This section discusses how to migrate from a Baseline device (i.e., PIC16C5X) to an Enhanced MCU device (i.e., PIC18FXXX).

The following are the list of modifications over the PIC16C5X microcontroller family:

**Not Currently Available**

## **APPENDIX E: MIGRATION FROM MID-RANGE TO ENHANCED DEVICES**

A detailed discussion of the differences between the mid-range MCU devices (i.e., PIC16CXXX) and the enhanced devices (i.e., PIC18FXXX) is provided in AN716, “*Migrating Designs from PIC16C74A/74B to PIC18C442*”. The changes discussed, while device specific, are generally applicable to all mid-range to enhanced device migrations.

This Application Note is available as Literature Number DS00716.

## **APPENDIX F: MIGRATION FROM HIGH-END TO ENHANCED DEVICES**

A detailed discussion of the migration pathway and differences between the high-end MCU devices (i.e., PIC17CXXX) and the enhanced devices (i.e., PIC18FXXX) is provided in AN726, “*PIC17CXXX to PIC18CXXX Migration*”. This Application Note is available as Literature Number DS00726.

# PIC18F6310/6410/8310/8410

---

NOTES:

# PIC18F6310/6410/8310/8410

## INDEX

### A

A/D	255
A/D Converter Interrupt, Configuring	259
Acquisition Requirements	260
ADCON0 Register	255
ADCON1 Register	255
ADCON2 Register	255
ADRESH Register	255, 258
ADRESL Register	255
Analog Port Pins	148
Analog Port Pins, Configuring	262
Associated Registers	264
Calculating the Minimum Required	
Acquisition Time	260
Configuring the Module	259
Conversion Clock (TAD)	261
Conversion Status (GO/DONE Bit)	258
Conversions	263
Converter Characteristics	387
Discharge	263
Operation in Power-Managed Modes	262
Selecting, Configuring Automatic	
Acquisition Time	261
Special Event Trigger (CCP)	264
Use of the CCP2 Trigger	264
Absolute Maximum Ratings	351
AC (Timing) Characteristics	368
Load Conditions for Device Timing	
Specifications	369
Parameter Symbolology	368
Temperature and Voltage Specifications	369
Timing Conditions	369
Access Bank	77
ACKSTAT	207
ACKSTAT Status Flag	207
ADCON0 Register	255
GO/DONE Bit	258
ADCON1 Register	255
ADCON2 Register	255
ADDFSR	340
ADDLW	303
Addressable Universal Synchronous Asynchronous Receiver Transmitter (AUSART). <i>See</i> AUSART.	
ADDULNK	340
ADDWF	303
ADDWFC	304
ADRESH Register	255
ADRESL Register	255, 258
Analog-to-Digital Converter. <i>See</i> A/D.	
ANDLW	304
ANDWF	305
Assembler	
MPASM Assembler	348
AUSART	
Asynchronous Mode	246
Associated Registers, Receive	249
Associated Registers, Transmit	247
Receiver	248
Setting up 9-Bit Mode with	
Address Detect	248
Transmitter	246

Baud Rate Generator (BRG)	244
Associated Registers	244
Baud Rate Error, Calculating	244
Baud Rates, Asynchronous Modes	245
High Baud Rate Select (BRGH Bit)	244
Operation in Power-Managed Modes	244
Sampling	244
Synchronous Master Mode	250
Associated Registers, Receive	252
Associated Registers, Transmit	251
Reception	252
Transmission	250
Synchronous Slave Mode	253
Associated Registers, Receive	254
Associated Registers, Transmit	253
Reception	254
Transmission	253
Auto-Wake-up on Sync Break Character	232

### B

Bank Select Register (BSR)	75
Baud Rate Generator	203
BC	305
BCF	306
BF	207
BF Status Flag	207
Block Diagrams	
16-Bit Byte Select Mode	99
16-Bit Byte Write Mode	97
16-Bit Word Write Mode	98
8-Bit Multiplexed Mode	102
A/D	258
Analog Input Model	259
AUSART Receive	248
AUSART Transmit	246
Baud Rate Generator	203
Capture Mode Operation	169
Comparator	
I/O Operating Modes	266
Comparator Analog Input Model	269
Comparator Output	268
Comparator Voltage Reference	272
Comparator Voltage Reference	
Output Buffer Example	273
Compare Mode Operation	171
Device Clock	40
EUSART Receive	230
EUSART Transmit	227
External Clock Input, EC Oscillator	36
External Clock Input, HS Oscillator	36
External Power-on Reset Circuit	
(Slow VDD Power-up)	57
Fail-Safe Clock Monitor	293
Generic I/O Port Operation	125
High/Low-Voltage Detect with External Input	276
Interrupt Logic	110
MSSP (I <sup>2</sup> C Master Mode)	201
MSSP (I <sup>2</sup> C Mode)	186
MSSP (SPI Mode)	177
On-Chip Reset Circuit	55
PIC18F6310/6410	12
PIC18F8310/8410	13
PLL (HS Mode)	37
PORTD and PORTE (Parallel Slave Port)	148
PWM Operation (Simplified)	173
RC Oscillator Mode	37

# PIC18F6310/6410/8310/8410

RCIO Oscillator Mode .....	37	Initializing PORTB .....	128
Reads From Program Memory .....	91	Initializing PORTC .....	131
Single Comparator .....	267	Initializing PORTD .....	134
Table Read and Table Write Operations .....	89	Initializing PORTE .....	137
Timer0 in 16-Bit Mode .....	152	Initializing PORTF .....	140
Timer0 in 8-Bit Mode .....	152	Initializing PORTG .....	142
Timer1 .....	156	Initializing PORTH .....	144
Timer1 (16-Bit Read/Write Mode) .....	156	Initializing PORTJ .....	146
Timer2 .....	162	Loading the SSPBUF (SSPSR) Register .....	180
Timer3 .....	164	Reading a Flash Program Memory Word .....	91
Timer3 (16-Bit Read/Write Mode) .....	164	Saving STATUS, WREG and BSR	
Watchdog Timer .....	290	Registers in RAM .....	124
BN .....	306	Code Protection .....	281
BNC .....	307	COMF .....	314
BNN .....	307	Comparator .....	265
BNOV .....	308	Analog Input Connection Considerations .....	269
BNZ .....	308	Associated Registers .....	269
BOR. See Brown-out Reset.		Configuration .....	266
BOV .....	311	Effects of a Reset .....	268
BRA .....	309	Interrupts .....	268
Break Character (12-Bit) Transmit and Receive .....	234	Operation .....	267
BRG. See Baud Rate Generator.		Operation During Sleep .....	268
Brown-out Reset (BOR) .....	58, 281	Outputs .....	267
Detecting .....	58	Reference .....	267
Disabling in Sleep Mode .....	58	External Signal .....	267
Software Enabled .....	58	Internal Signal .....	267
BSF .....	309	Response Time .....	267
BTFSC .....	310	Comparator Specifications .....	366
BTFSS .....	310	Comparator Voltage Reference .....	271
BTG .....	311	Accuracy and Error .....	272
BZ .....	312	Associated Registers .....	273
<b>C</b>		Configuring .....	271
C Compilers		Connection Considerations .....	272
MPLAB C18 .....	348	Effects of a Reset .....	272
CALL .....	312	Operation During Sleep .....	272
Capture (CCP Module) .....	169	Compare (CCP Module) .....	170
Associated Registers .....	172	Associated Registers .....	172
CCP Pin Configuration .....	169	CCP Pin Configuration .....	170
CCPR2H:CCPR2L Registers .....	169	CCPR2 Register .....	170
Software Interrupt .....	170	Software Interrupt Mode .....	170
Timer1/Timer3 Mode Selection .....	169	Special Event Trigger .....	165, 170, 264
Capture/Compare/PWM (CCP) .....	167	Timer1/Timer3 Mode Selection .....	170
Capture Mode. See Capture.		Computed GOTO .....	72
CCP Mode and Timer Resources .....	168	CONFIG2L (Configuration 2 Low) .....	283
CCPRxH Register .....	168	Configuration Bits .....	281
CCPRxL Register .....	168	Configuration Register Protection .....	295
Compare Mode. See Compare.		Conversion Considerations .....	396
Interconnect Configurations .....	168	CPFSEQ .....	314
Module Configuration .....	168	CPFSGT .....	315
CLRF .....	313	CPFSLT .....	315
CLRWDT .....	313	Crystal Oscillator/Ceramic Resonator .....	35
Code Examples		Customer Change Notification Service .....	409
16 x 16 Signed Multiply Routine .....	108	Customer Notification Service .....	409
16 x 16 Unsigned Multiply Routine .....	108	Customer Support .....	409
8 x 8 Signed Multiply Routine .....	107	<b>D</b>	
8 x 8 Unsigned Multiply Routine .....	107	Data Addressing Modes .....	84
Changing Between Capture Prescalers .....	170	Comparing Addressing Modes with the	
Computed GOTO Using an Offset Value .....	72	Extended Instruction Set Enabled .....	87
Executing Back to Back Sleep Instructions .....	46	Direct .....	84
Fast Register Stack .....	72	Indexed Literal Offset .....	86
How to Clear RAM (Bank 1) Using Indirect		Indirect .....	84
Addressing .....	84	Inherent and Literal .....	84
Implementing a Real-Time Clock Using a			
Timer1 Interrupt Service .....	159		
Initializing PORTA .....	125		



# PIC18F6310/6410/8310/8410

Data Memory .....	75	Extended Instruction Set	
Access Bank .....	77	ADDFSR .....	340
and the Extended Instruction Set .....	86	ADDULNK .....	340
Bank Select Register (BSR) .....	75	and Using MPLAB IDE Tools .....	346
General Purpose Registers .....	77	CALLW .....	341
Map for PIC18F6310/6410/8310/8410 Devices .....	76	Considerations for Use .....	344
Special Function Registers .....	78	MOVSF .....	341
DAW .....	316	MOVSS .....	342
DC Characteristics .....	363	PUSHL .....	342
Power-Down and Supply Current .....	355	SUBFSR .....	343
Supply Voltage .....	354	SUBULNK .....	343
DCFSNZ .....	317	External Memory Interface .....	95
DECF .....	316	16-Bit Byte Select Mode .....	99
DECFSZ .....	317	16-Bit Byte Write Mode .....	97
Development Support .....	347	16-Bit Mode .....	97
Device Differences .....	395	16-Bit Mode Timing .....	100
Device Overview .....	9	16-Bit Word Write Mode .....	98
Features (table) .....	11	8-Bit Mode .....	102
New Core Features .....	9	8-Bit Mode Timing .....	103
Device Reset Timers .....	59	and the Program Memory Modes .....	96
PLL Lock Time-out .....	59	Associated Registers .....	105
Power-up Timer (PWRT) .....	59	PIC18F8310/8410 External Bus, I/O Port Functions .....	96
Time-out Sequence .....	59		
Device Reset Timer Oscillator Start-up Timer (OST) .....	59	<b>F</b>	
Direct Addressing .....	85	Fail-Safe Clock Monitor .....	281, 293
<b>E</b>		Interrupts in Power-Managed Modes .....	294
Effect on Standard PIC Instructions .....	344	POR or Wake from Sleep .....	294
Effects of Power-Managed Modes on Various Clock Sources .....	43	WDT During Oscillator Failure .....	293
Electrical Characteristics .....	351	Fast Register Stack .....	72
Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART). See EUSART.		Firmware Instructions .....	297
Equations		Flash Program Memory	
16 x 16 Signed Multiplication Algorithm .....	108	Associated Registers .....	93
16 x 16 Unsigned Multiplication Algorithm .....	108	Operation During Code-Protect .....	92
A/D Acquisition Time .....	260	Reading .....	90
A/D Minimum Charging Time .....	260	FSCM. See Fail-Safe Clock Monitor.	
Errata .....	7	<b>G</b>	
EUSART		GOTO .....	318
Asynchronous Mode .....	226	<b>H</b>	
12-Bit Break Transmit and Receive .....	234	Hardware Multiplier .....	107
Associated Registers, Receive .....	231	Introduction .....	107
Associated Registers, Transmit .....	228	Operation .....	107
Auto-Wake-up on Sync Break .....	232	Performance Comparison .....	107
Receiver .....	229	High/Low-Voltage Detect .....	275
Setting up 9-Bit Mode with Address Detect .....	229	Applications .....	278
Transmitter .....	226	Associated Registers .....	279
Baud Rate Generator (BRG) .....	221	Characteristics .....	367
Associated Registers .....	221	Current Consumption .....	277
Auto-Baud Rate Detect .....	224	Effects of a Reset .....	279
Baud Rate Error, Calculating .....	221	Operation .....	276
Baud Rates, Asynchronous Modes .....	222	During Sleep .....	279
High Baud Rate Select (BRGH Bit) .....	221	Start-up Time .....	277
Operation in Power-Managed Modes .....	221	Setup .....	277
Sampling .....	221	Typical Application .....	278
Synchronous Master Mode .....	235	HLVD. See High/Low-Voltage Detect. ....	275
Associated Registers, Receive .....	238		
Associated Registers, Transmit .....	236		
Reception .....	237		
Transmission .....	235		
Synchronous Slave Mode .....	239		
Associated Registers, Receive .....	240		
Associated Registers, Transmit .....	239		
Reception .....	240		
Transmission .....	239		

# PIC18F6310/6410/8310/8410

<b>I</b>	
I/O Ports .....	125
I <sup>2</sup> C Mode (MSSP)	
Acknowledge Sequence Timing .....	210
Associated Registers .....	216
Baud Rate Generator .....	203
Bus Collision	
During a Repeated Start Condition .....	214
During a Start Condition .....	212
During a Stop Condition .....	215
Clock Arbitration .....	204
Clock Stretching .....	196
10-Bit Slave Receive Mode (SEN = 1) .....	196
7-Bit Slave Receive Mode (SEN = 1) .....	196
Effect of a Reset .....	211
General Call Address Support .....	200
I <sup>2</sup> C Clock Rate w/BRG .....	203
Master Mode .....	201
Operation .....	202
Reception .....	207
Repeated Start Condition Timing .....	206
Start Condition .....	205
Transmission .....	207
Transmit Sequence .....	202
Multi-Master Communication, Bus Collision and Arbitration .....	211
Multi-Master Mode .....	211
Operation .....	190
Read/Write Bit Information (R/W Bit) .....	190, 191
Registers .....	186
Serial Clock (RC3/SCK/SCL) .....	191
Slave Mode .....	190
Addressing .....	190
Reception .....	191
Sleep Operation .....	211
Stop Condition Timing .....	210
Transmission .....	191
ID Locations .....	281, 296
Idle Modes	
PRI_IDLE .....	51
INCF .....	318
INCFSZ .....	319
In-Circuit Debugger .....	296
In-Circuit Serial Programming (ICSP) .....	281, 296
Indexed Literal Offset Addressing	
and Standard PIC18 Instructions .....	344
Indexed Literal Offset Mode .....	86, 344
Effect on Standard PIC18 Instructions .....	86
Mapping the Access Bank .....	88
Indirect Addressing .....	85
INFSNZ .....	319
Initialization Conditions for all Registers .....	63–66
Instruction Cycle .....	73
Clocking Scheme .....	73
Instruction Flow/Pipelining .....	73
Instruction Set .....	297
ADDLW .....	303
ADDWF .....	303
ADDWF (Indexed Literal Offset mode) .....	345
ADDWFC .....	304
ANDLW .....	304
ANDWF .....	305
BC .....	305
BCF .....	306
BN .....	306
BNC .....	307
BNN .....	307
BNOV .....	308
BNZ .....	308
BOV .....	311
BRA .....	309
BSF .....	309
BSF (Indexed Literal Offset mode) .....	345
BTFSC .....	310
BTFSS .....	310
BTG .....	311
BZ .....	312
CALL .....	312
CLRf .....	313
CLRWDT .....	313
COMF .....	314
CPFSEQ .....	314
CPFSGT .....	315
CPFSLT .....	315
DAW .....	316
DCFSNZ .....	317
DECF .....	316
DECFSZ .....	317
Extended Instructions .....	339
Syntax .....	339
General Format .....	299
GOTO .....	318
INCF .....	318
INCFSZ .....	319
INFSNZ .....	319
IORLW .....	320
IORWF .....	320
LFSR .....	321
MOVf .....	321
MOVFF .....	322
MOVLB .....	322
MOVLW .....	323
MOVWF .....	323
MULLW .....	324
MULWF .....	324
NEGF .....	325
NOP .....	325
Opcode Field Descriptions .....	298
POP .....	326
PUSH .....	326
RCALL .....	327
RESET .....	327
RETFIE .....	328
RETLW .....	328
RETURN .....	329
RLCF .....	329
RLNCF .....	330
RRCF .....	330
RRNCF .....	331
SETF .....	331
SETF (Indexed Literal Offset mode) .....	345
SLEEP .....	332
SUBFWB .....	332
SUBLW .....	333
SUBWF .....	333
SUBWFB .....	334
SWAPF .....	334

# PIC18F6310/6410/8310/8410

TBLRD .....	335	MSSP	
TBLWT .....	336	ACK Pulse .....	190, 191
TSTFSZ .....	337	Control Registers (general) .....	177
XORLW .....	337	I <sup>2</sup> C Mode. See I <sup>2</sup> C Mode.	
XORWF .....	338	Module Overview .....	177
Summary Table .....	300	SPI Master/Slave Connection .....	181
INTCON Register		SPI Mode. See SPI Mode.	
RBIF Bit .....	128	SSPBUF .....	182
INTCON Registers .....	111	SSPSR .....	182
Inter-Integrated Circuit. See I <sup>2</sup> C.		MULLW .....	324
Internal Oscillator Block .....	38	MULWF .....	324
Adjustment .....	38	<b>N</b>	
INTIO Modes .....	38	NEGF .....	325
INTOSC Frequency Drift .....	38	NOP .....	325
INTOSC Output Frequency .....	38	<b>O</b>	
OSCTUNE Register .....	38	Oscillator	
Internal RC Oscillator		Clock Sources .....	40
Use with WDT .....	290	Selecting the 31 kHz Source .....	41
Internet Address .....	409	Selection Using OSCCON Register .....	41
Interrupt Sources .....	281	External Clock Input .....	36
A/D Conversion Complete .....	259	RC .....	37
Context Saving During Interrupts .....	124	RCIO Mode .....	37
Interrupt-on-Change (RB7:RB4) .....	128	Switching .....	40
INTx Pin .....	124	Transitions .....	41
PORTB, Interrupt-on-Change .....	124	Oscillator Configuration .....	35
TMR0 .....	124	EC .....	35
TMR0 Overflow .....	153	ECIO .....	35
TMR1 Overflow .....	155	HS .....	35
TMR2 to PR2 Match (PWM) .....	173	HSPLL .....	35
TMR3 Overflow .....	163, 165	Internal Oscillator Block .....	38
Interrupts .....	109	INTIO1 .....	35
Interrupts, Flag Bits		INTIO2 .....	35
Interrupt-on-Change (RB7:RB4) Flag		LP .....	35
(RBIF Bit) .....	128	RC .....	35
INTOSC, INTRC. See Internal Oscillator Block.		RCIO .....	35
IORLW .....	320	XT .....	35
IORWF .....	320	Oscillator Selection .....	281
IPR Registers .....	120	Oscillator Start-up Timer (OST) .....	43, 281
<b>L</b>		Oscillator, Timer1 .....	155, 165
LFSR .....	321	Oscillator, Timer3 .....	163
<b>M</b>		<b>P</b>	
Master Clear ( $\overline{\text{MCLR}}$ ) .....	57	Packaging .....	389
Master Synchronous Serial Port (MSSP). See MSSP.		Details .....	390
Memory Organization .....	67	Marking .....	389
Data Memory .....	75	Parallel Slave Port (PSP) .....	148
Program Memory .....	67	Associated Registers .....	150
Memory Programming Requirements .....	365	RE0/RD Pin .....	148
Microchip Internet Web Site .....	409	RE1/ $\overline{\text{WR}}$ Pin .....	148
Migration from Baseline to Enhanced Devices .....	396	RE2/ $\overline{\text{CS}}$ Pin .....	148
Migration from High-End to Enhanced Devices .....	397	Select (PSPMODE Bit) .....	148
Migration from Mid-Range to Enhanced Devices .....	397	PIC18 Instruction Execution, Extended .....	88
MOVF .....	321	PIE Registers .....	117
MOVFF .....	322	Pin Functions	
MOVLB .....	322	AVDD .....	30
MOVLW .....	323	AVDD .....	21
MOVSS .....	342	AVss .....	21
MOVWF .....	323	AVss .....	30
MPLAB ASM30 Assembler, Linker, Librarian .....	348	OSC1/CLKI/RA7 .....	14, 22
MPLAB Integrated Development Environment		OSC2/CLKO/RA6 .....	14, 22
Software .....	347	RA0/AN0 .....	15, 23
MPLAB PM3 Device Programmer .....	350	RA1/AN1 .....	15, 23
MPLAB REAL ICE In-Circuit Emulator System .....	349	RA2/AN2/VREF- .....	15, 23
MPLINK Object Linker/MPLIB Object Librarian .....	348	RA3/AN3/VREF+ .....	15, 23

# PIC18F6310/6410/8310/8410

RA4/T0CKI .....	15, 23	RH0/AD16 .....	29
RA5/AN4/HLVDIN .....	15, 23	RH1/AD17 .....	29
RB0/INT0 .....	16, 24	RH2/AD18 .....	29
RB1/INT1 .....	16, 24	RH3/AD19 .....	29
RB2/INT2 .....	16, 24	RH4 .....	29
RB3/INT3 .....	16	RH5 .....	29
RB3/INT3/CCP2 .....	24	RH6 .....	29
RB4/KBI0 .....	16, 24	RH7 .....	29
RB5/KBI1 .....	16, 24	RJ0/ALE .....	30
RB6/KBI2/PGC .....	16, 24	RJ1/OE .....	30
RB7/KBI3/PGD .....	16, 24	RJ2/WRL .....	30
RC0/T1OSO/T13CKI .....	17, 25	RJ3/WRH .....	30
RC1/T1OSI/CCP2 .....	17, 25	RJ4/BA0 .....	30
RC2/CCP1 .....	17, 25	RJ5/CE .....	30
RC3/SCK/SCL .....	17, 25	RJ6/LB .....	30
RC4/SDI/SDA .....	17, 25	RJ7/UB .....	30
RC5/SDO .....	17, 25	VDD .....	21
RC6/TX1/CK1 .....	17, 25	VDD .....	30
RC7/RX1/DT1 .....	17, 25	Vss .....	21
RD0/AD0/PSP0 .....	26	Vss .....	30
RD0/PSP0 .....	18	Pinout I/O Descriptions	
RD1/AD1/PSP1 .....	26	PIC18F6310/6410 .....	14
RD1/PSP1 .....	18	PIC18F8310/8410 .....	22
RD2/AD2/PSP2 .....	26	PIR Registers .....	114
RD2/PSP2 .....	18	PLL .....	37
RD3/AD3/PSP3 .....	26	HSPLL Oscillator Mode .....	37
RD3/PSP3 .....	18	Use with INTOSC .....	37, 38
RD4/AD4/PSP4 .....	26	POP .....	326
RD4/PSP4 .....	18	POR. See Power-on Reset.	
RD5/AD5/PSP5 .....	26	PORTA	
RD5/PSP5 .....	18	Associated Registers .....	127
RD6/AD6/PSP6 .....	26	Functions .....	126
RD6/PSP6 .....	18	LATA Register .....	125
RD7/AD7/PSP7 .....	26	PORTA Register .....	125
RD7/PSP7 .....	18	TRISA Register .....	125
RE0/AD8/RD .....	27	PORTB	
RE0/RD .....	19	Associated Registers .....	130
RE1/AD9/WR .....	27	Functions .....	129
RE1/WR .....	19	LATB Register .....	128
RE2/AD10/CS .....	27	PORTB Register .....	128
RE2/CS .....	19	RB7:RB4 Interrupt-on-Change Flag (RBIF Bit) .....	128
RE3 .....	19	TRISB Register .....	128
RE3/AD11 .....	27	PORTC	
RE4 .....	19	Associated Registers .....	133
RE4/AD12 .....	27	Functions .....	132
RE5 .....	19	LATC Register .....	131
RE5/AD13 .....	27	PORTC Register .....	131
RE6 .....	19	RC3/SCK/SCL Pin .....	191
RE6/AD14 .....	27	TRISC Register .....	131
RE7/CCP2 .....	19	PORTD .....	148
RE7/CCP2/AD15 .....	27	Associated Registers .....	136
RF0/AN5 .....	20, 28	Functions .....	135
RF1/AN6/C2OUT .....	20, 28	LATD Register .....	134
RF2/AN7/C1OUT .....	20, 28	PORTD Register .....	134
RF3/AN8 .....	20, 28	TRISD Register .....	134
RF4/AN9 .....	20, 28	PORTE	
RF5/AN10/CVREF .....	20, 28	Analog Port Pins .....	148
RF6/AN11 .....	20, 28	Associated Registers .....	139
RF7/SS .....	20, 28	Functions .....	138
RG0/CCP3 .....	21, 29	LATE Register .....	137
RG1/TX2/CK2 .....	21, 29	PORTE Register .....	137
RG2/RX2/DT2 .....	21, 29	PSP Mode Select (PSPMODE Bit) .....	148
RG3 .....	21, 29	RE0/RD Pin .....	148
RG4 .....	21, 29	RE1/WR Pin .....	148
RG5 .....	21, 29	RE2/CS Pin .....	148
RG5/MCLR/VPP .....	14, 22	TRISE Register .....	137

# PIC18F6310/6410/8310/8410

PORTF		Instructions	74
Associated Registers	141	Two-Word Instructions	74
Functions	141	Interrupt Vector	67
LATF Register	140	Look-up Tables	72
PORTF Register	140	Map and Stack (diagram)	67
TRISF Register	140	Memory Access for PIC18F8310/8410 Modes	69
PORTG		Memory Maps for PIC18FX310/X410 Modes	69
Associated Registers	143	PIC18F8310/8410 Memory Modes	68
Functions	143	Reset Vector	67
LATG Register	142	Table Reads and Table Writes	89
PORTG Register	142	Writing and Erasing On-Chip Program	
TRISG Register	142	Memory (ICSP Mode)	92
PORTH		Writing To	
Associated Registers	145	Unexpected Termination	92
Functions	145	Write Verify	92
LATH Register	144	Writing to Memory Space (PIC18F8X10)	92
PORTH Register	144	Program Memory Modes	
TRISH Register	144	Extended Microcontroller	96
PORTJ		Microcontroller	96
Associated Registers	147	Microprocessor	96
Functions	147	Microprocessor with Boot Block	96
LATJ Register	146	Program Verification and Code Protection	295
PORTJ Register	146	Associated Registers	295
TRISJ Register	146	Programming, Device Instructions	297
Postscaler, WDT		PSP. See Parallel Slave Port.	
Assignment (PSA Bit)	153	Pulse-Width Modulation. See PWM (CCP Module).	
Rate Select (TOPS2:TOPS0 Bits)	153	PUSH	326
Switching Between Timer0 and WDT	153	PUSH and POP Instructions	71
Power-Managed Modes	45	PUSHL	342
and Multiple Sleep Commands	46	PWM (CCP Module)	
Clock Sources	45	Associated Registers	175
Clock Transitions, Status Indicators	46	Duty Cycle	174
Entering	45	Example Frequencies/Resolutions	174
Exiting Idle and Sleep Modes	53	Period	173
by Interrupt	53	Setup for PWM Operation	174
by Reset	53	TMR2 to PR2 Match	173
by WDT Time-out	53		
Without an Oscillator Start-up Delay	53	<b>Q</b>	
Idle Modes	50	Q Clock	174
Operation	105	<b>R</b>	
Run Modes	46	RAM. See Data Memory.	
Selecting	45	RCALL	327
Sleep Mode	50	RCON Register	
Summary (table)	45	Bit Status During Initialization	62
Power-on Reset (POR)	57, 281	Reader Response	410
Oscillator Start-up Timer (OST)	59	Register File	77
Power-up Timer (PWRT)	59	Register File Summary	79–82
Time-out Sequence	59	Registers	
Power-up Delays	43	ADCON0 (A/D Control 0)	255
Power-up Timer (PWRT)	43, 281	ADCON1 (A/D Control 1)	256
Prescaler, Capture	170	ADCON2 (A/D Control 2)	257
Prescaler, Timer0	153	BAUDCON1 (Baud Rate Control 1)	220
Assignment (PSA Bit)	153	CCPxCON (Capture/Compare/PWM Control)	167
Rate Select (TOPS2:TOPS0 Bits)	153	CMCON (Comparator Control)	265
Switching Between Timer0 and WDT	153	CONFIG1H (Configuration 1 High Byte)	282
Prescaler, TMR2	174	CONFIG2H (Configuration 2 High)	284
Program Counter	70	CONFIG3H (Configuration 3 High)	286
PCL, PCH and PCU Registers	70	CONFIG3L (Configuration 3 Low)	285
PCLATH and PCLATU Registers	70	CONFIG4L (Configuration 4 Low)	287
Program Memory	89	CONFIG5L (Configuration 5 Low)	287
Code Protection, from Table Reads	295	CONFIG7L (Configuration 7 Low)	288
Control Registers	90	CVRCON (Comparator Voltage	
TABLAT (Table Latch) Register	90	Reference Control)	271
TBLPTR (Table Pointer) Register	90	DEVID1 (Device ID 1)	289
Erasing External Memory (PIC18F8X10)	92	DEVID2 (Device ID 2)	289

# PIC18F6310/6410/8310/8410

HLVDCON (HLVD Control) .....	275	<b>S</b>	
INTCON (Interrupt Control) .....	111	SCK .....	177
INTCON2 (Interrupt Control 2) .....	112	SDI .....	177
INTCON3 (Interrupt Control 3) .....	113	SDO .....	177
IPR1 (Peripheral Interrupt Priority 1) .....	120	Serial Clock, SCK .....	177
IPR2 (Peripheral Interrupt Priority 2) .....	121	Serial Data In (SDI) .....	177
IPR3 (Peripheral Interrupt Priority 3) .....	122	Serial Data Out (SDO) .....	177
MEMCON (Memory Control) .....	95	Serial Peripheral Interface. <i>See</i> SPI Mode.	
OSCCON (Oscillator Control) .....	42	SETF .....	331
OSCTUNE (Oscillator Tuning) .....	39	Slave Select ( $\overline{SS}$ ) .....	177
PIE1 (Peripheral Interrupt Enable 1) .....	117	SLEEP .....	332
PIE2 (Peripheral Interrupt Enable 2) .....	118	Sleep Mode	
PIE3 (Peripheral Interrupt Enable 3) .....	119	OSC1 and OSC2 Pin States .....	43
PIR1 (Peripheral Interrupt Request (Flag) 1) .....	114	Software Simulator (MPLAB SIM) .....	349
PIR2 (Peripheral Interrupt Request (Flag) 2) .....	115	Special Event Trigger. <i>See</i> Compare (CCP Module).	
PIR3 (Peripheral Interrupt Request (Flag) 3) .....	116	Special Features of the CPU .....	281
PSPCON (Parallel Slave Port Control) .....	149	Special Function Registers .....	78
RCON (Reset Control) .....	56, 123	Map .....	78
RCSTA2 (AUSART2 Receive Status and Control) .....	243	SPI Mode (MSSP)	
SSPCON1 (MSSP Control 1, SPI Mode) .....	179	Associated Registers .....	185
SSPCON2, (I <sup>2</sup> C Mode) .....	189	Bus Mode Compatibility .....	185
SSPSTAT (MSSP Status, I <sup>2</sup> C Mode) .....	187, 188	Effects of a Reset .....	185
SSPSTAT (MSSP Status, SPI Mode) .....	178, 219	Enabling SPI I/O .....	181
T0CON (Timer0 Control) .....	151	Master Mode .....	182
T1CON (Timer1 Control) .....	155	Master/Slave Connection .....	181
T2CON (Timer2 Control) .....	161	Operation .....	180
T3CON (Timer3 Control) .....	163	Serial Clock .....	177
TXSTA1 (EUSART1 Transmit Status and Control) .....	218	Serial Data In .....	177
TXSTA2 (AUSART2 Transmit Status and Control) .....	242	Serial Data Out .....	177
WDTCON (Watchdog Timer Control) .....	291	Slave Mode .....	183
RESET .....	327	Slave Select .....	177
Reset .....	55	Slave Select Synchronization .....	183
MCLR Reset, Normal Operation .....	55	Sleep Operation .....	185
MCLR Reset, Power Managed Modes .....	55	SPI Clock .....	182
Power-on Reset (POR) .....	55	Typical Connection .....	181
Programmable Brown-out Reset (BOR) .....	55	$\overline{SS}$ .....	177
RESET Instruction .....	55	SSPOV .....	207
Stack Full Reset .....	55	SSPOV Status Flag .....	207
Stack Underflow Reset .....	55	SSPSTAT Register	
Watchdog Timer (WDT) Reset .....	55	R/W Bit .....	190, 191
Resets .....	281	Stack Full/Underflow Resets .....	72
RETFIE .....	328	Standard Instructions .....	297
RETLW .....	328	SUBFSR .....	343
RETURN .....	329	SUBFWB .....	332
Return Address Stack .....	70	SUBLW .....	333
Return Stack Pointer (STKPTR) .....	71	SUBULNK .....	343
Revision History .....	395	SUBWF .....	333
RLCF .....	329	SUBWFB .....	334
RLNCF .....	330	SWAPF .....	334
RRCF .....	330	<b>T</b>	
RRNCF .....	331	T0CON Register	
Run Modes		PSA Bit .....	153
PRI_RUN .....	46	T0CS Bit .....	152
RC_RUN .....	48	T0PS2:T0PS0 Bits .....	153
SEC_RUN .....	46	T0SE Bit .....	152
		Table Pointer Operations (table) .....	90
		Table Reads/Table Writes .....	72
		TBLRD .....	335
		TBLWT .....	336
		Time-out in Various Situations (table) .....	59

# PIC18F6310/6410/8310/8410

Timer0 .....	151	Bus Collision for Transmit and Acknowledge .....	211
16-Bit Mode Timer Reads and Writes .....	152	Capture/Compare/PWM (All CCP Modules) .....	377
Associated Registers .....	153	CLKO and I/O .....	372
Clock Source Edge Select (T0SE Bit) .....	152	Clock Synchronization .....	197
Clock Source Select (T0CS Bit) .....	152	Clock/Instruction Cycle .....	73
Operation .....	152	Example SPI Master Mode (CKE = 0) .....	378
Overflow Interrupt .....	153	Example SPI Master Mode (CKE = 1) .....	379
Prescaler. See Prescaler, Timer0.		Example SPI Slave Mode (CKE = 0) .....	380
Timer1 .....	155	Example SPI Slave Mode (CKE = 1) .....	381
16-Bit Read/Write Mode .....	157	External Clock (All Modes Except PLL) .....	370
Associated Registers .....	159	External Memory Bus for SLEEP (16-Bit Microprocessor Mode) .....	101
Interrupt .....	158	External Memory Bus for SLEEP (8-Bit Microprocessor Mode) .....	104
Low-Power Option .....	157	External Memory Bus for TBLRD (16-Bit Extended Microcontroller Mode) .....	100
Operation .....	156	External Memory Bus for TBLRD (16-Bit Microprocessor Mode) .....	100
Oscillator .....	155, 157	External Memory Bus for TBLRD (8-Bit Extended Microcontroller Mode) .....	103
Oscillator Layout Considerations .....	158	External Memory Bus for TBLRD (8-Bit Microprocessor Mode) .....	103
Overflow Interrupt .....	155	Fail-Safe Clock Monitor .....	294
Resetting, Using a Special Event Trigger Output (CCP) .....	158	High/Low-Voltage Detect (VDIRMAG = 1) .....	278
TMR1H Register .....	155	High/Low-Voltage Detect Characteristics .....	367
TMR1L Register .....	155	High/Low-Voltage Detect Operation (VDIRMAG = 0) .....	277
Use as a Real-Time Clock .....	158	I <sup>2</sup> C Bus Data .....	382
Using as a Clock Source .....	157	I <sup>2</sup> C Bus Start/Stop Bits .....	382
Timer2 .....	161	I <sup>2</sup> C Master Mode (7 or 10-Bit Transmission) .....	208
Associated Registers .....	162	I <sup>2</sup> C Master Mode (7-Bit Reception) .....	209
Interrupt .....	162	I <sup>2</sup> C Master Mode First Start Bit .....	205
Operation .....	161	I <sup>2</sup> C Slave Mode (10-Bit Reception, SEN = 0) .....	194
Output .....	162	I <sup>2</sup> C Slave Mode (10-Bit Reception, SEN = 1) .....	199
PR2 Register .....	173	I <sup>2</sup> C Slave Mode (10-Bit Transmission) .....	195
TMR2 to PR2 Match Interrupt .....	173	I <sup>2</sup> C Slave Mode (7-bit Reception, SEN = 0) .....	192
Timer3 .....	163	I <sup>2</sup> C Slave Mode (7-Bit Reception, SEN = 1) .....	198
16-Bit Read/Write Mode .....	165	I <sup>2</sup> C Slave Mode (7-Bit Transmission) .....	193
Associated Registers .....	165	I <sup>2</sup> C Slave Mode General Call Address Sequence (7 or 10-Bit Address Mode) .....	200
Operation .....	164	I <sup>2</sup> C Stop Condition Receive or Transmit Mode .....	210
Oscillator .....	163, 165	Master SSP I <sup>2</sup> C Bus Data .....	384
Overflow Interrupt .....	163, 165	Master SSP I <sup>2</sup> C Bus Start/Stop Bits .....	384
Special Event Trigger (CCP) .....	165	Parallel Slave Port (PSP) Read .....	150
TMR3H Register .....	163	Parallel Slave Port (PSP) Write .....	149
TMR3L Register .....	163	Program Memory Read .....	373
Timing Diagrams		Program Memory Write .....	374
A/D Conversion .....	388	PWM Output .....	173
Acknowledge Sequence .....	210	Repeated Start Condition .....	206
Asynchronous Reception .....	230, 249	Reset, Watchdog Timer (WDT), Oscillator Start-up Timer (OST) and Power-up Timer (PWRT) .....	375
Asynchronous Transmission .....	227, 247	Send Break Character Sequence .....	234
(Back to Back) .....	227, 247	Slave Synchronization .....	183
Automatic Baud Rate Calculation .....	225	Slow Rise Time (MCLR Tied to VDD, VDD Rise > TPWRT) .....	61
Auto-Wake-up Bit (WUE) During Normal Operation .....	233	SPI Mode (Master Mode) .....	182
Auto-Wake-up Bit (WUE) During Sleep .....	233	SPI Mode (Slave Mode, CKE = 0) .....	184
Baud Rate Generator with Clock Arbitration .....	204	SPI Mode (Slave Mode, CKE = 1) .....	184
BRG Overflow Sequence .....	225	Synchronous Reception (Master Mode, SREN) .....	237, 252
BRG Reset Due to SDA Arbitration During Start Condition .....	213	Synchronous Transmission .....	235, 250
Brown-out Reset (BOR) .....	375	Synchronous Transmission (Through TXEN) .....	236, 251
Bus Collision During a Repeated Start Condition (Case 1) .....	214	Time-out Sequence on POR w/PLL Enabled (MCLR Tied to VDD) .....	61
Bus Collision During a Repeated Start Condition (Case 2) .....	214		
Bus Collision During a Start Condition (SCL = 0) .....	213		
Bus Collision During a Start Condition (SDA Only) .....	212		
Bus Collision During a Stop Condition (Case 1) .....	215		
Bus Collision During a Stop Condition (Case 2) .....	215		

# PIC18F6310/6410/8310/8410

Time-out Sequence on Power-up (MCLR Not Tied to VDD, Case 1) .....	60	Master SSP I <sup>2</sup> C Bus Data Requirements .....	385
Time-out Sequence on Power-up (MCLR Not Tied to VDD, Case 2) .....	60	Master SSP I <sup>2</sup> C Bus Start/Stop Bits Requirements .....	384
Time-out Sequence on Power-up (MCLR Tied to VDD, VDD Rise TPWRT) .....	60	PLL Clock .....	371
Timer0 and Timer1 External Clock .....	376	Program Memory Read Requirements .....	373
Transition for Entry to PRI_IDLE Mode .....	51	Program Memory Write Requirements .....	374
Transition for Entry to SEC_RUN Mode .....	47	Reset, Watchdog Timer, Oscillator Start-up Timer, Power-up Timer and Brown-out Reset Requirements .....	375
Transition for Entry to Sleep Mode .....	50	Timer0 and Timer1 External Clock Requirements .....	376
Transition for Two-Speed Start-up (INTOSC to HSPLL) .....	292	USART Synchronous Receive Requirements .....	386
Transition for Wake From Idle to Run Mode .....	51	USART Synchronous Transmission Requirements .....	386
Transition for Wake From Sleep (HSPLL) .....	50	Top-of-Stack Access .....	70
Transition From RC_RUN Mode to PRI_RUN Mode .....	49	TRISE Register PSPMODE Bit .....	148
Transition From SEC_RUN Mode to PRI_RUN Mode (HSPLL) .....	47	TSTFSZ .....	337
Transition to RC_RUN Mode .....	49	Two-Speed Start-up .....	281, 292
USART Synchronous Receive (Master/Slave) .....	386	Two-Word Instructions Example Cases .....	74
USART Synchronous Transmission (Master/Slave) .....	386	TXSTA1 Register BRGH Bit .....	221
Timing Diagrams and Specifications		TXSTA2 Register BRGH Bit .....	244
A/D Conversion Requirements .....	388	<b>V</b>	
AC Characteristics		Voltage Reference Specifications .....	366
Internal RC Accuracy .....	371	<b>W</b>	
Capture/Compare/PWM Requirements (All CCP Modules) .....	377	Watchdog Timer (WDT) .....	281, 290
CLKO and I/O Requirements .....	372	Associated Registers .....	291
Example SPI Mode Requirements (Master Mode, CKE = 0) .....	378	Control Register .....	290
Example SPI Mode Requirements (Master Mode, CKE = 1) .....	379	During Oscillator Failure .....	293
Example SPI Mode Requirements (Slave Mode, CKE = 0) .....	380	Programming Considerations .....	290
Example SPI Slave Mode Requirements (CKE = 1) .....	381	WCOL .....	205, 206, 207, 210
External Clock Requirements .....	370	WCOL Status Flag .....	205, 206, 207, 210
I <sup>2</sup> C Bus Data Requirements (Slave Mode) .....	383	WWW Address .....	409
I <sup>2</sup> C Bus Start/Stop Bits Requirements (Slave Mode) .....	382	WWW, On-Line Support .....	7
		<b>X</b>	
		XORLW .....	337
		XORWF .....	338



## THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at [www.microchip.com](http://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](http://www.microchip.com). Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support
- 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>**

# PIC18F6310/6410/8310/8410

---

---

## 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: PIC18F6310/6410/8310/8410

Literature Number: DS39635C

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?

\_\_\_\_\_  
\_\_\_\_\_

# PIC18F6310/6410/8310/8410

## PIC18F6310/6410/8310/8410 PRODUCT IDENTIFICATION SYSTEM

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

<u>PART NO.</u>	<u>X</u>	<u>/XX</u>	<u>XXX</u>
Device	Temperature Range	Package	Pattern
Device	PIC18F6310/6410/8310/8410 <sup>(1)</sup> , PIC18F6310/6410/8310/8410T <sup>(2)</sup> ; VDD range 4.2V to 5.5V PIC18LF6310/6410/8310/8410 <sup>(1)</sup> , PIC18LF6310/6410/8310/8410T <sup>(2)</sup> ; VDD range 2.0V to 5.5V		
Temperature Range	I = -40°C to +85°C (Industrial) E = -40°C to +125°C (Extended)		
Package	PT = TQFP (Thin Quad Flatpack)		
Pattern	QTP, SQTP, Code or Special Requirements (blank otherwise)		

**Examples:**

- a) PIC18LF6410-I/PT 301 = Industrial temp., TQFP package, Extended VDD limits, QTP pattern #301.
- b) PIC18F8410-I/PT = Industrial temp., TQFP package, normal VDD limits.
- c) PIC18F8410-E/PT = Extended temp., TQFP package, normal VDD limits.

**Note 1:** F = Standard Voltage Range  
LF = Wide Voltage Range  
**2:** T = in tape and reel



# MICROCHIP

## 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](http://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 - Chongqing**  
Tel: 86-23-8980-9588  
Fax: 86-23-8980-9500

**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 - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4123

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**  
Tel: 91-20-2566-1512  
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: 886-7-213-7830  
Fax: 886-7-330-9305

**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

08/04/10

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

## Microchip:

[PIC18F6310T-I/PT](#) [PIC18F6410T-I/PT](#) [PIC18LF8310T-I/PT](#) [PIC18LF8410T-I/PT](#) [PIC18F8310T-I/PT](#) [PIC18LF6410-I/PT](#) [PIC18LF6310-I/PT](#) [PIC18F8410-E/PT](#) [PIC18F8310-E/PT](#) [PIC18F6310-E/PT](#) [PIC18LF8410-I/PT](#) [PIC18F8410-I/PT](#) [PIC18F8310-I/PT](#) [PIC18F6410-I/PT](#) [PIC18F6310-I/PT](#) [PIC18LF8310-I/PT](#) [PIC18F6410-E/PT](#) [PIC18F8410T-I/PT](#) [PIC18LF6310T-I/PT](#) [PIC18LF6410T-I/PT](#)