



---

# **PIC18F2525/2620/4525/4620**

## **Data Sheet**

28/40/44-Pin  
Enhanced Flash Microcontrollers  
with 10-Bit A/D and nanoWatt 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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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, Accuron, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL, SmartSensor 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, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, PICkit, PICDEM, PICDEM.net, PICtail, PIC<sup>32</sup> logo, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Total Endurance, UNI/O, 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.

© 2008, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

---

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

*Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

**MICROCHIP****PIC18F2525/2620/4525/4620**

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

### Power Management Features:

- Run: CPU on, Peripherals on
- Idle: CPU off, Peripherals on
- Sleep: CPU off, Peripherals off
- Ultra Low 50nA Input Leakage
- Run mode Currents Down to 11  $\mu$ A Typical
- Idle mode Currents Down to 2.5  $\mu$ A Typical
- Sleep mode Current Down to 100 nA Typical
- Timer1 Oscillator: 900 nA, 32 kHz, 2V
- Watchdog Timer: 1.4  $\mu$ A, 2V Typical
- Two-Speed Oscillator Start-up

### Flexible Oscillator Structure:

- Four Crystal modes, up to 40 MHz
- 4x Phase Lock Loop (PLL) – Available for Crystal and Internal Oscillators
- Two External RC modes, up to 4 MHz
- Two External Clock modes, up to 40 MHz
- Internal Oscillator Block:
  - Fast wake from Sleep and Idle, 1  $\mu$ s typical
  - 8 use-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

### Peripheral Highlights:

- High-Current Sink/Source 25 mA/25 mA
- Three Programmable External Interrupts
- Four Input Change Interrupts
- Up to 2 Capture/Compare/PWM (CCP) modules, one with Auto-Shutdown (28-pin devices)
- Enhanced Capture/Compare/PWM (ECCP) module (40/44-pin devices only):
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart

### Peripheral Highlights (Continued):

- Master Synchronous Serial Port (MSSP) module Supporting 3-Wire SPI (all 4 modes) and I<sup>2</sup>C<sup>TM</sup> Master and Slave modes
- Enhanced Addressable USART module:
  - Supports RS-485, RS-232 and LIN/J2602
  - RS-232 operation using internal oscillator block (no external crystal required)
  - Auto-wake-up on Start bit
  - Auto-Baud Detect
- 10-Bit, up to 13-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:

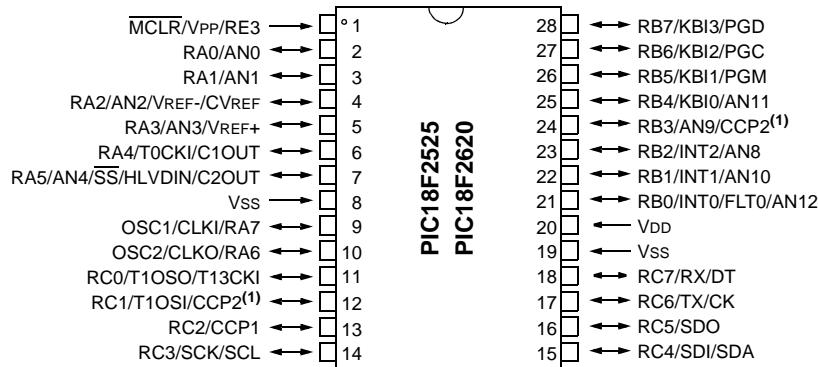
- C Compiler Optimized Architecture:
  - Optional extended instruction set designed to optimize re-entrant code
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory Typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory Typical
- Flash/Data EEPROM Retention: 100 Years Typical
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-Supply 5V In-Circuit Serial Programming<sup>TM</sup> (ICSP<sup>TM</sup>) 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		Data Memory		I/O	10-Bit A/D (ch)	CCP/ECCP (PWM)	MSSP		USART	Comp.	Timers 8/16-Bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I <sup>2</sup> C <sup>TM</sup>			
PIC18F2525	48K	24576	3968	1024	25	10	2/0	Y	Y	1	2	1/3
PIC18F2620	64K	32768	3968	1024	25	10	2/0	Y	Y	1	2	1/3
PIC18F4525	48K	24576	3968	1024	36	13	1/1	Y	Y	1	2	1/3
PIC18F4620	64K	32768	3968	1024	36	13	1/1	Y	Y	1	2	1/3

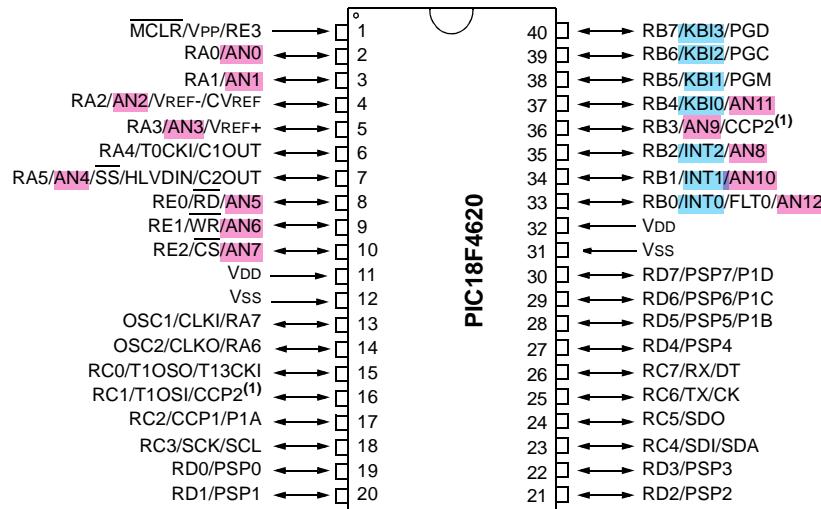
# PIC18F2525/2620/4525/4620

## Pin Diagrams

28-Pin SPDIP, SOIC



40-Pin PDIP

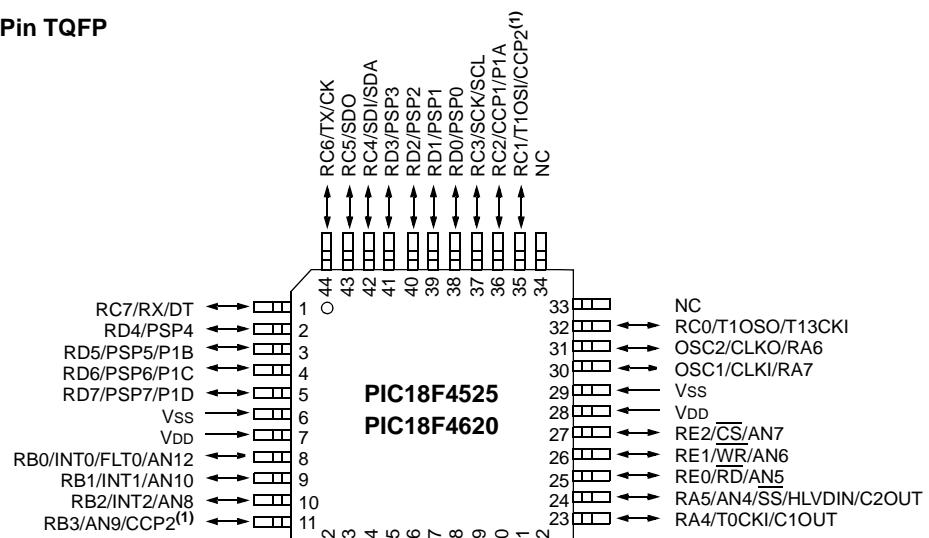


Note 1: RB3 is the alternate pin for CCP2 multiplexing.

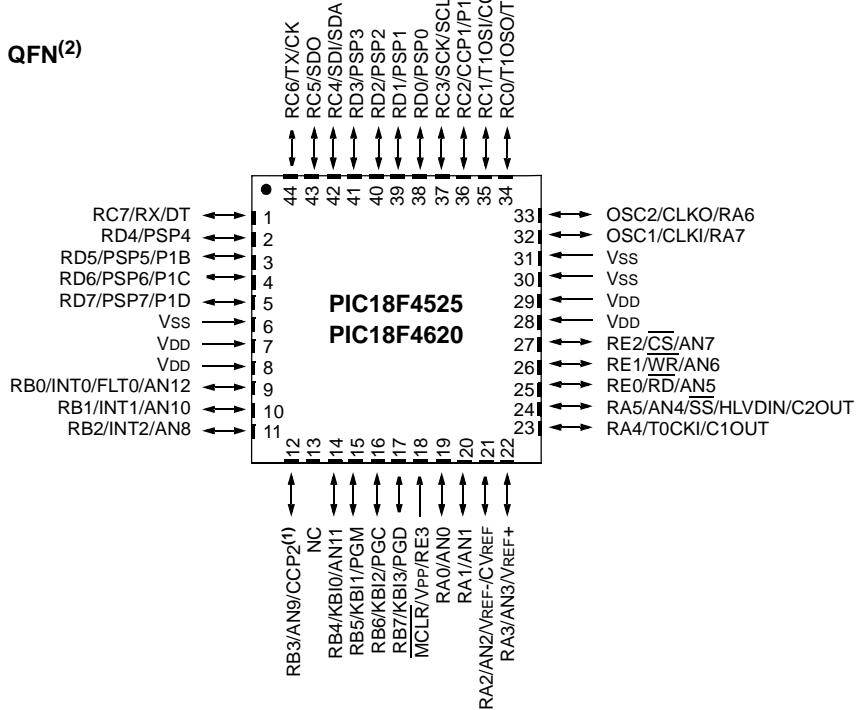
## **PIC18F2525/2620/4525/4620**

## Pin Diagrams (Cont.'d)

### 44-Pin TQFP



44-Pin QFN<sup>(2)</sup>



**Note 1:** RB3 is the alternate pin for CCP2 multiplexing.

2: For the QFN package, it is recommended that the bottom pad be connected to Vss.

# PIC18F2525/2620/4525/4620

---

---

## Table of Contents

1.0	Device Overview .....	7
2.0	Oscillator Configurations .....	23
3.0	Power-Managed Modes .....	33
4.0	Reset .....	41
5.0	Memory Organization .....	53
6.0	Data EEPROM Memory .....	73
7.0	Flash Program Memory .....	79
8.0	8 x 8 Hardware Multiplier .....	89
9.0	I/O Ports .....	91
10.0	Interrupts .....	109
11.0	Timer0 Module .....	123
12.0	Timer1 Module .....	127
13.0	Timer2 Module .....	133
14.0	Timer3 Module .....	135
15.0	Capture/Compare/PWM (CCP) Modules .....	139
16.0	Enhanced Capture/Compare/PWM (ECCP) Module .....	147
17.0	Master Synchronous Serial Port (MSSP) Module .....	161
18.0	Enhanced Universal Synchronous Receiver Transmitter (EUSART) .....	201
19.0	10-Bit Analog-to-Digital Converter (A/D) Module .....	223
20.0	Comparator Module .....	233
21.0	Comparator Voltage Reference Module .....	239
22.0	High/Low-Voltage Detect (HLVD) .....	243
23.0	Special Features of the CPU .....	249
24.0	Instruction Set Summary .....	267
25.0	Development Support .....	317
26.0	Electrical Characteristics .....	321
27.0	DC and AC Characteristics Graphs and Tables .....	361
28.0	Packaging Information .....	383
	Appendix A: Revision History .....	393
	Appendix B: Device Differences .....	394
	Appendix C: Conversion Considerations .....	395
	Appendix D: Migration from Baseline to Enhanced Devices .....	395
	Appendix E: Migration from Mid-Range TO Enhanced Devices .....	396
	Appendix F: Migration from High-End to Enhanced Devices .....	396
	Index .....	397
	The Microchip Web Site .....	407
	Customer Change Notification Service .....	407
	Customer Support .....	407
	Reader Response .....	408
	PIC18F2525/2620/4525/4620 Product Identification System .....	409

## TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at [docerrors@microchip.com](mailto:docerrors@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)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

### Customer Notification System

Register on our web site at [www.microchip.com](http://www.microchip.com) to receive the most current information on all of our products.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 1.0 DEVICE OVERVIEW

This document contains device-specific information for the following devices:

- PIC18F2525
- PIC18F2620
- PIC18F4525
- PIC18F4620
- PIC18LF2525
- PIC18LF2620
- PIC18LF4525
- PIC18LF4620

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Enhanced Flash program memory. On top of these features, the PIC18F2525/2620/4525/4620 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 PIC18F2525/2620/4525/4620 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.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 26.0 "Electrical Characteristics"** for values.

#### 1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F2525/2620/4525/4620 family offer ten different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes, using crystals or ceramic resonators
- Two External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O)
- Two External RC Oscillator modes with the same pin options as the External Clock modes
- An internal oscillator block which provides an 8 MHz clock and an INTRC source (approximately 31 kHz), as well as a range of 6 user-selectable clock frequencies, between 125 kHz to 4 MHz, for a total of 8 clock frequencies. This option frees the two oscillator pins for use as additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the High-Speed Crystal and Internal Oscillator modes, which allows clock speeds of up to 40 MHz. Used with the internal oscillator, the PLL gives users a complete selection of clock speeds, from 31 kHz to 32 MHz – all without using an external crystal or clock circuit.

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

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

# PIC18F2525/2620/4525/4620

---

## 1.2 Other Special Features

- **Memory Endurance:** The Enhanced Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 100,000 for program memory and 1,000,000 for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-Programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18F2525/2620/4525/4620 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **Enhanced CCP Module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include auto-shutdown, for disabling PWM outputs on interrupt or other select conditions and auto-restart, to reactivate outputs once the condition has cleared.
- **Enhanced Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include automatic baud rate detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-Bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reducing code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit prescaler, allowing an extended time-out range that is stable across operating voltage and temperature. See **Section 26.0 “Electrical Characteristics”** for time-out periods.

## 1.3 Details on Individual Family Members

Devices in the PIC18F2525/2620/4525/4620 family are available in 28-pin and 40/44-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in five ways:

1. Flash program memory (48 Kbytes for PIC18FX525 devices, 64 Kbytes for PIC18FX620 devices).
2. A/D channels (10 for 28-pin devices, 13 for 40/44-pin devices).
3. I/O ports (3 bidirectional ports on 28-pin devices, 5 bidirectional ports on 40/44-pin devices).
4. CCP and Enhanced CCP implementation (28-pin devices have 2 standard CCP modules, 40/44-pin devices have one standard CCP module and one ECCP module).
5. Parallel Slave Port (present only on 40/44-pin devices).

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 PIC18F2525/2620/4525/4620 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an “F” in the part number (such as PIC18F2620), accommodate an operating VDD range of 4.2V to 5.5V. Low-voltage parts, designated by “LF” (such as PIC18LF2620), function over an extended VDD range of 2.0V to 5.5V.

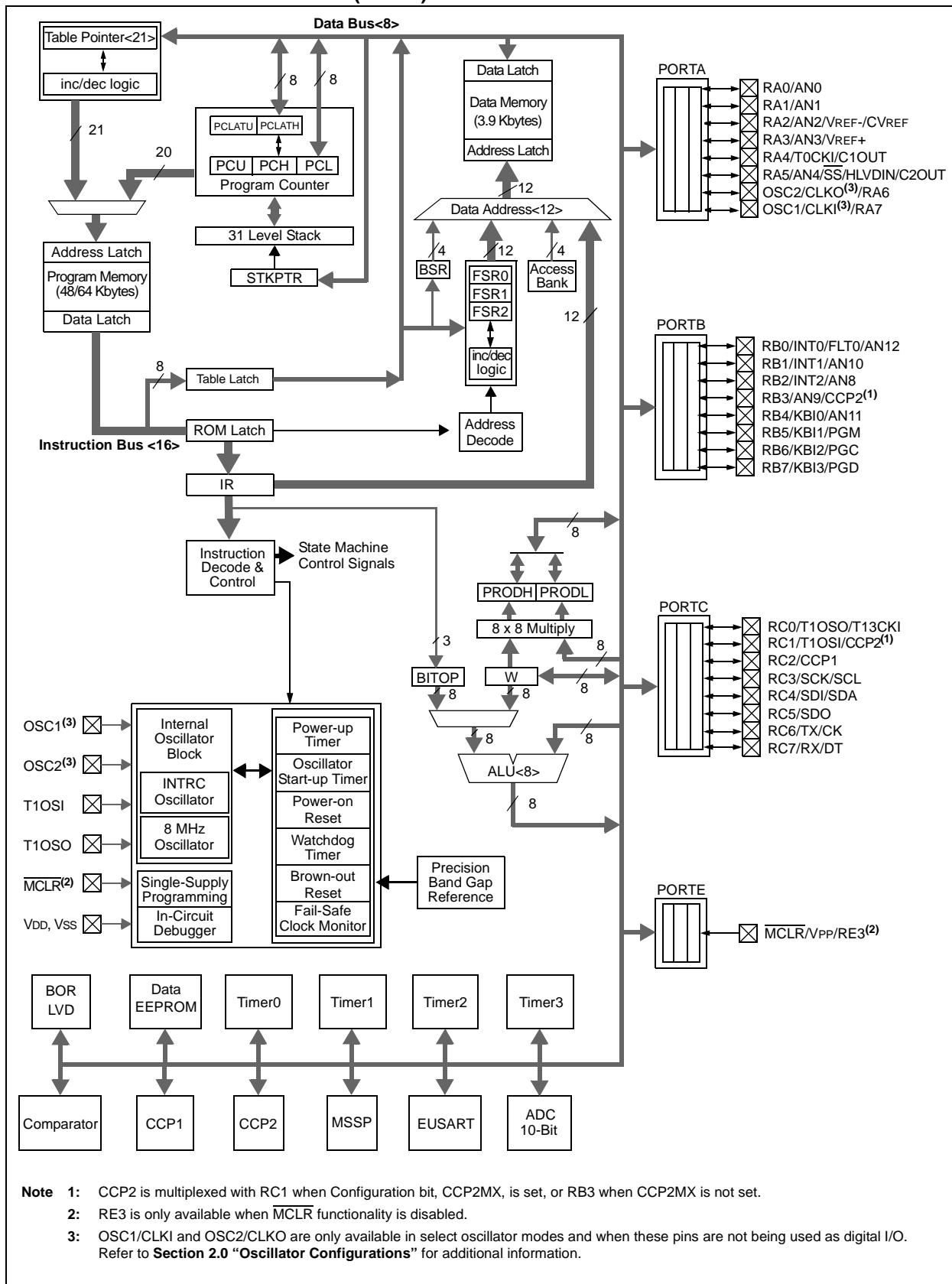
# PIC18F2525/2620/4525/4620

TABLE 1-1: DEVICE FEATURES

Features	PIC18F2525	PIC18F2620	PIC18F4525	PIC18F4620
Operating Frequency	DC – 40 MHz			
Program Memory (Bytes)	49152	65536	49152	65536
Program Memory (Instructions)	24576	32768	24576	32768
Data Memory (Bytes)	3968	3968	3968	3968
Data EEPROM Memory (Bytes)	1024	1024	1024	1024
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Parallel Communications (PSP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Resets (and Delays)	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			
Packages	28-Pin SPDIP 28-Pin SOIC	28-Pin SPDIP 28-Pin SOIC	40-Pin PDIP 44-Pin QFN 44-Pin TQFP	40-Pin PDIP 44-Pin QFN 44-Pin TQFP

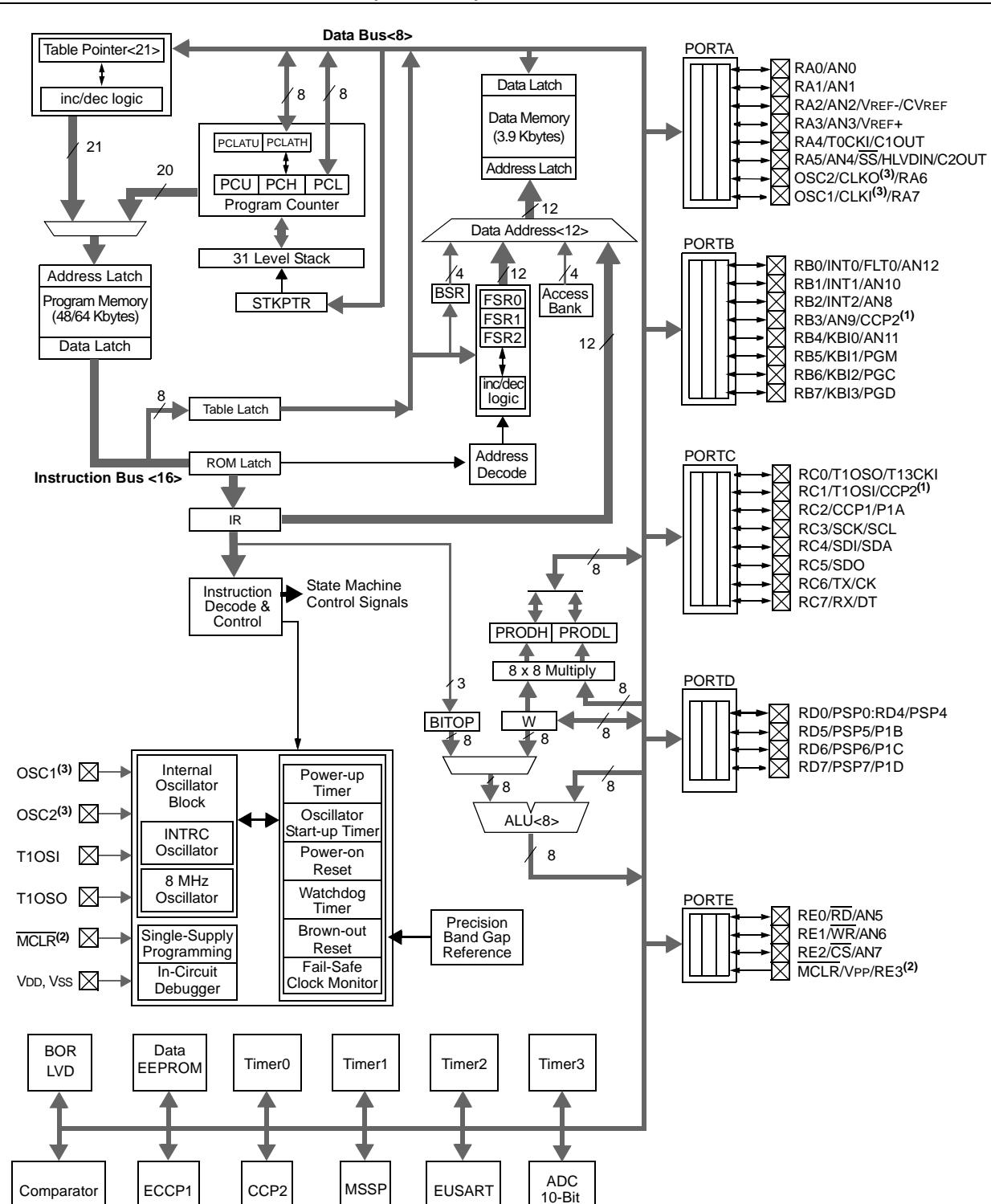
# PIC18F2525/2620/4525/4620

FIGURE 1-1: PIC18F2525/2620 (28-PIN) BLOCK DIAGRAM



# PIC18F2525/2620/4525/4620

FIGURE 1-2: PIC18F4525/4620 (40/44-PIN) BLOCK DIAGRAM



**Note 1:** CCP2 is multiplexed with RC1 when Configuration bit, CCP2MX, is set, or RB3 when CCP2MX is not set.

**2:** RE3 is only available when MCLR functionality is disabled.

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

# PIC18F2525/2620/4525/4620

---

TABLE 1-2: PIC18F2525/2620 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	SPDIP, SOIC			
MCLR/VPP/RE3 MCLR	1	I	ST	Master Clear (input) or programming voltage (input). Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP RE3		P I	ST	Programming voltage input. Digital input.
OSC1/CLK1/RA7 OSC1	9	I	ST	Oscillator crystal or external clock input. Oscillator crystal input or external clock source input.
CLK1		I	CMOS	ST buffer when configured in RC mode; CMOS otherwise.
RA7		I/O	TTL	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLK1, OSC2/CLKO pins.) General purpose I/O pin.
OSC2/CLKO/RA6 OSC2	10	O	—	Oscillator crystal or clock output. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO		O	—	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6		I/O	TTL	General purpose I/O pin.

**Legend:** TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

CMOS = CMOS compatible input or output

I = Input

P = Power

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

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

# PIC18F2525/2620/4525/4620

TABLE 1-2: PIC18F2525/2620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	SPDIP, SOIC			
RA0/AN0	2			PORTA is a bidirectional I/O port.
RA0		I/O	TTL	Digital I/O.
AN0		I	Analog	Analog input 0.
RA1/AN1	3			Digital I/O.
RA1		I/O	TTL	Digital I/O.
AN1		I	Analog	Analog input 1.
RA2/AN2/VREF-/CVREF	4			Digital I/O.
RA2		I/O	TTL	Digital I/O.
AN2		I	Analog	Analog input 2.
VREF-		I	Analog	A/D reference voltage (low) input.
CVREF		O	Analog	Comparator reference voltage output.
RA3/AN3/VREF+	5			Digital I/O.
RA3		I/O	TTL	Digital I/O.
AN3		I	Analog	Analog input 3.
VREF+		I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT	6			Digital I/O.
RA4		I/O	ST	Digital I/O.
T0CKI		I	ST	Timer0 external clock input.
C1OUT		O	—	Comparator 1 output.
RA5/AN4/SS/HLDIN/C2OUT	7			Digital I/O.
RA5		I/O	TTL	Digital I/O.
AN4		I	Analog	Analog input 4.
SS		I	TTL	SPI slave select input.
HLDIN		I	Analog	High/Low-Voltage Detect input.
C2OUT		O	—	Comparator 2 output.
RA6				See the OSC2/CLKO/RA6 pin.
RA7				See the OSC1/CLKI/RA7 pin.

**Legend:** TTL = TTL compatible input

CMOS = CMOS compatible input or output

ST = Schmitt Trigger input with CMOS levels

I = Input

O = Output

P = Power

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

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

# PIC18F2525/2620/4525/4620

TABLE 1-2: PIC18F2525/2620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	SPDIP, SOIC			
RB0/INT0/FLT0/AN12 RB0 INT0 FLT0 AN12	21	I/O	TTL	PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
		I	ST	Digital I/O.
		I	ST	External interrupt 0.
		I	Analog	PWM Fault input for CCP1.
				Analog input 12.
RB1/INT1/AN10 RB1 INT1 AN10	22	I/O	TTL	Digital I/O.
		I	ST	External interrupt 1.
		I	Analog	Analog input 10.
RB2/INT2/AN8 RB2 INT2 AN8	23	I/O	TTL	Digital I/O.
		I	ST	External interrupt 2.
		I	Analog	Analog input 8.
RB3/AN9/CCP2 RB3 AN9 CCP2 <sup>(1)</sup>	24	I/O	TTL	Digital I/O.
		I	Analog	Analog input 9.
		I/O	ST	Capture 2 input/Compare 2 output/PWM2 output.
RB4/KBI0/AN11 RB4 KBI0 AN11	25	I/O	TTL	Digital I/O.
		I	TTL	Interrupt-on-change pin.
		I	Analog	Analog input 11.
RB5/KBI1/PGM RB5 KBI1 PGM	26	I/O	TTL	Digital I/O.
		I	TTL	Interrupt-on-change pin.
		I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC RB6 KBI2 PGC	27	I/O	TTL	Digital I/O.
		I	TTL	Interrupt-on-change pin.
		I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD RB7 KBI3 PGD	28	I/O	TTL	Digital I/O.
		I	TTL	Interrupt-on-change pin.
		I/O	ST	In-Circuit Debugger and ICSP programming data pin.

**Legend:** TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

CMOS = CMOS compatible input or output

I = Input

P = Power

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

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

# PIC18F2525/2620/4525/4620

TABLE 1-2: PIC18F2525/2620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	SPDIP, SOIC			
RC0/T1OSO/T13CKI RC0 T1OSO T13CKI	11	I/O O I	ST — ST	PORTC is a bidirectional I/O port.  Digital I/O. Timer1 oscillator output. Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2 RC1 T1OSI CCP2 <sup>(2)</sup>	12	I/O I I/O	ST Analog ST	Digital I/O. Timer1 oscillator input. Capture 2 input/Compare 2 output/PWM2 output.
RC2/CCP1 RC2 CCP1	13	I/O I/O	ST ST	Digital I/O. Capture 1 input/Compare 1 output/PWM1 output.
RC3/SCK/SCL RC3 SCK SCL	14	I/O I/O I/O	ST ST ST	Digital I/O. Synchronous serial clock input/output for SPI mode. Synchronous serial clock input/output for I <sup>2</sup> C <sup>TM</sup> mode.
RC4/SDI/SDA RC4 SDI SDA	15	I/O I I/O	ST ST ST	Digital I/O. SPI data in. I <sup>2</sup> C data I/O.
RC5/SDO RC5 SDO	16	I/O O	ST —	Digital I/O. SPI data out.
RC6/TX/CK RC6 TX CK	17	I/O O I/O	ST — ST	Digital I/O. EUSART asynchronous transmit. EUSART synchronous clock (see related RX/DT).
RC7/RX/DT RC7 RX DT	18	I/O I I/O	ST ST ST	Digital I/O. EUSART asynchronous receive. EUSART synchronous data (see related TX/CK).
RE3	—	—	—	See MCLR/VPP/RE3 pin.
Vss	8, 19	P	—	Ground reference for logic and I/O pins.
Vdd	20	P	—	Positive supply for logic and I/O pins.

**Legend:** TTL = TTL compatible input

CMOS = CMOS compatible input or output

ST = Schmitt Trigger input with CMOS levels

I = Input

O = Output

P = Power

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

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

# PIC18F2525/2620/4525/4620

TABLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
MCLR/VPP/RE3 MCLR VPP RE3	1	18	18	I P I	ST ST	Master Clear (input) or programming voltage (input). Master Clear (Reset) input. This pin is an active-low Reset to the device. Programming voltage input. Digital input.
OSC1/CLKI/RA7 OSC1 CLKI RA7	13	32	30	I I I/O	ST CMOS TTL	Oscillator crystal or external clock input. Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; analog otherwise. External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.) General purpose I/O pin.
OSC2/CLKO/RA6 OSC2 CLKO RA6	14	33	31	O O I/O	— — TTL	Oscillator crystal or clock output. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. General purpose I/O pin.

**Legend:** TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

CMOS = CMOS compatible input or output

I = Input

P = Power

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

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

**3:** For the QFN package, it is recommended that the bottom pad be connected to Vss.

# PIC18F2525/2620/4525/4620

TABLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
RA0/AN0	2	19	19	I/O	TTL	PORTA is a bidirectional I/O port.
RA0				I	Analog	Digital I/O.
AN0						Analog input 0.
RA1/AN1	3	20	20	I/O	TTL	Digital I/O.
RA1				I	Analog	Analog input 1.
AN1						
RA2/AN2/VREF-/CVREF	4	21	21	I/O	TTL	Digital I/O.
RA2				I	Analog	Analog input 2.
AN2						
VREF-				I	Analog	A/D reference voltage (low) input.
CVREF				O	Analog	Comparator reference voltage output.
RA3/AN3/VREF+	5	22	22	I/O	TTL	Digital I/O.
RA3				I	Analog	Analog input 3.
AN3						
VREF+				I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT	6	23	23	I/O	ST	Digital I/O.
RA4				I	ST	Timer0 external clock input.
T0CKI				O	—	Comparator 1 output.
C1OUT						
RA5/AN4/SS/HLDIN/C2OUT	7	24	24	I/O	TTL	Digital I/O.
RA5				I	Analog	Analog input 4.
AN4						
SS				I	TTL	SPI slave select input.
HLDIN				I	Analog	High/Low-Voltage Detect input.
C2OUT				O	—	Comparator 2 output.
RA6						See the OSC2/CLK0/RA6 pin.
RA7						See the OSC1/CLK1/RA7 pin.

**Legend:** TTL = TTL compatible input CMOS = CMOS compatible input or output  
 ST = Schmitt Trigger input with CMOS levels I = Input  
 O = Output P = Power

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

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

**3:** For the QFN package, it is recommended that the bottom pad be connected to Vss.

# PIC18F2525/2620/4525/4620

TABLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
RB0/INT0/FLT0/AN12 RB0 INT0 FLT0 AN12	33	9	8	I/O I I I	TTL ST ST Analog	PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.  Digital I/O. External interrupt 0. PWM Fault input for Enhanced CCP1. Analog input 12.
RB1/INT1/AN10 RB1 INT1 AN10	34	10	9	I/O I I	TTL ST Analog	Digital I/O. External interrupt 1. Analog input 10.
RB2/INT2/AN8 RB2 INT2 AN8	35	11	10	I/O I I	TTL ST Analog	Digital I/O. External interrupt 2. Analog input 8.
RB3/AN9/CCP2 RB3 AN9 CCP2 <sup>(1)</sup>	36	12	11	I/O I I/O	TTL Analog ST	Digital I/O. Analog input 9. Capture 2 input/Compare 2 output/PWM2 output.
RB4/KBI0/AN11 RB4 KBI0 AN11	37	14	14	I/O I I	TTL TTL Analog	Digital I/O. Interrupt-on-change pin. Analog input 11.
RB5/KBI1/PGM RB5 KBI1 PGM	38	15	15	I/O I I/O	TTL TTL ST	Digital I/O. Interrupt-on-change pin. Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC RB6 KBI2 PGC	39	16	16	I/O I I/O	TTL TTL ST	Digital I/O. Interrupt-on-change pin. In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD RB7 KBI3 PGD	40	17	17	I/O I I/O	TTL TTL ST	Digital I/O. Interrupt-on-change pin. In-Circuit Debugger and ICSP programming data pin.

**Legend:** TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

CMOS = CMOS compatible input or output

I = Input

P = Power

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

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

**3:** For the QFN package, it is recommended that the bottom pad be connected to Vss.

# PIC18F2525/2620/4525/4620

TABLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
RC0/T1OSO/T13CKI RC0 T1OSO T13CKI	15	34	32	I/O O I	ST — ST	PORTC is a bidirectional I/O port. Digital I/O. Timer1 oscillator output. Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2 RC1 T1OSI CCP2 <sup>(2)</sup>	16	35	35	I/O I I/O	ST CMOS ST	Digital I/O. Timer1 oscillator input. Capture 2 input/Compare 2 output/PWM2 output.
RC2/CCP1/P1A RC2 CCP1 P1A	17	36	36	I/O I/O O	ST ST —	Digital I/O. Capture 1 input/Compare 1 output/PWM1 output. Enhanced CCP1 output.
RC3/SCK/SCL RC3 SCK SCL	18	37	37	I/O I/O I/O	ST ST ST	Digital I/O. Synchronous serial clock input/output for SPI mode. Synchronous serial clock input/output for I <sup>2</sup> C <sup>TM</sup> mode.
RC4/SDI/SDA RC4 SDI SDA	23	42	42	I/O I I/O	ST ST ST	Digital I/O. SPI data in. I <sup>2</sup> C data I/O.
RC5/SDO RC5 SDO	24	43	43	I/O O	ST —	Digital I/O. SPI data out.
RC6/TX/CK RC6 TX CK	25	44	44	I/O O I/O	ST — ST	Digital I/O. EUSART asynchronous transmit. EUSART synchronous clock (see related RX/DT).
RC7/RX/DT RC7 RX DT	26	1	1	I/O I I/O	ST ST ST	Digital I/O. EUSART asynchronous receive. EUSART synchronous data (see related TX/CK).

Legend: TTL = TTL compatible input

CMOS = CMOS compatible input or output

ST = Schmitt Trigger input with CMOS levels

I = Input

O = Output

P = Power

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

2: Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

3: For the QFN package, it is recommended that the bottom pad be connected to Vss.

# PIC18F2525/2620/4525/4620

TABLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP	QFN	TQFP			
						PORTD is a bidirectional I/O port or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when the PSP module is enabled.
RD0/PSP0 RD0 PSP0	19	38	38	I/O I/O	ST TTL	Digital I/O. Parallel Slave Port data.
RD1/PSP1 RD1 PSP1	20	39	39	I/O I/O	ST TTL	Digital I/O. Parallel Slave Port data.
RD2/PSP2 RD2 PSP2	21	40	40	I/O I/O	ST TTL	Digital I/O. Parallel Slave Port data.
RD3/PSP3 RD3 PSP3	22	41	41	I/O I/O	ST TTL	Digital I/O. Parallel Slave Port data.
RD4/PSP4 RD4 PSP4	27	2	2	I/O I/O	ST TTL	Digital I/O. Parallel Slave Port data.
RD5/PSP5/P1B RD5 PSP5 P1B	28	3	3	I/O I/O O	ST TTL —	Digital I/O. Parallel Slave Port data. Enhanced CCP1 output.
RD6/PSP6/P1C RD6 PSP6 P1C	29	4	4	I/O I/O O	ST TTL —	Digital I/O. Parallel Slave Port data. Enhanced CCP1 output.
RD7/PSP7/P1D RD7 PSP7 P1D	30	5	5	I/O I/O O	ST TTL —	Digital I/O. Parallel Slave Port data. Enhanced CCP1 output.

Legend: TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

CMOS = CMOS compatible input or output

I = Input

P = Power

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

2: Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

3: For the QFN package, it is recommended that the bottom pad be connected to Vss.

# PIC18F2525/2620/4525/4620

TABLE 1-3: PIC18F4525/4620 PINOUT I/O DESCRIPTIONS (CONTINUED)

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

**Legend:** TTL = TTL compatible input CMOS = CMOS compatible input or output  
 ST = Schmitt Trigger input with CMOS levels I = Input  
 O = Output P = Power

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

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

**3:** For the QFN package, it is recommended that the bottom pad be connected to Vss.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 2.0 OSCILLATOR CONFIGURATIONS

### 2.1 Oscillator Types

PIC18F2525/2620/4525/4620 devices can be operated in ten different oscillator modes. The user can program the Configuration bits, FOSC3:FOSC0, in Configuration Register 1H to select one of these ten modes:

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

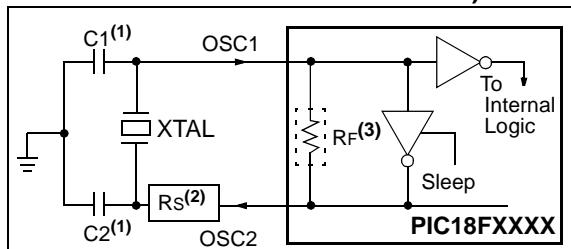
### 2.2 Crystal Oscillator/Ceramic Resonators

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

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

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

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



- Note 1:** See Table 2-1 and Table 2-2 for initial values of C1 and C2.  
**Note 2:** A series resistor (Rs) may be required for AT strip cut crystals.  
**Note 3:** RF varies with the oscillator mode chosen.

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

Typical Capacitor Values Used:			
Mode	Freq	OSC1	OSC2
XT	3.58 MHz 4.19 MHz 4 MHz 4 MHz	15 pF 15 pF 30 pF 50 pF	15 pF 15 pF 30 pF 50 pF

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

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

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

# PIC18F2525/2620/4525/4620

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

Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	C2
LP	32 kHz	30 pF	30 pF
XT	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	10 MHz	15 pF	15 pF
	20 MHz	15 pF	15 pF
	25 MHz	15 pF	15 pF

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

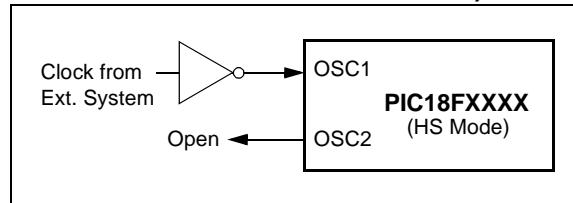
Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following this table for additional information.

- Note 1:** Higher capacitance increases the stability of the oscillator but also increases the start-up time.
- 2:** When operating below 3V VDD, or when using certain ceramic resonators at any voltage, it may be necessary to use the HS mode or switch to a crystal oscillator.
- 3:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
- 4:** Rs may be required to avoid overdriving crystals with low drive level specification.
- 5:** Always verify oscillator performance over the VDD and temperature range that is expected for the application.

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

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

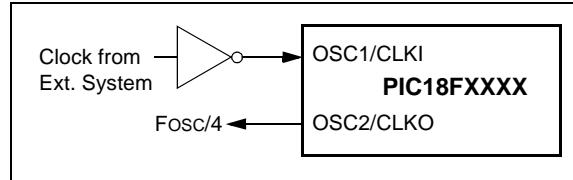


## 2.3 External Clock Input

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.

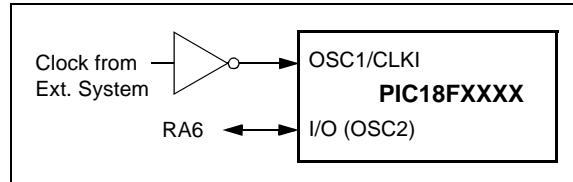
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-3 shows the pin connections for the EC Oscillator mode.

**FIGURE 2-3: 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 2-4 shows the pin connections for the ECIO Oscillator mode.

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



## 2.4 RC Oscillator

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

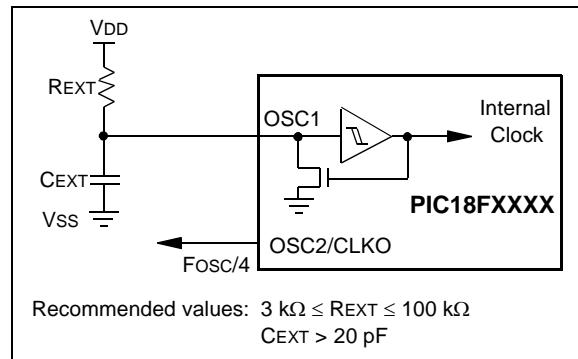
- supply voltage
- values of the external resistor (REXT) and capacitor (CEXT)
- operating temperature

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

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

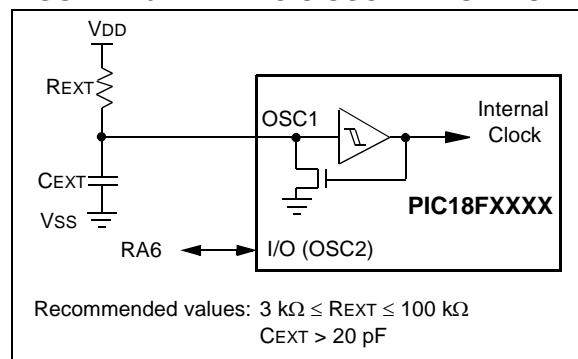
In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-5 shows how the R/C combination is connected.

**FIGURE 2-5: RC OSCILLATOR MODE**



The RCIO Oscillator mode (Figure 2-6) 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 2-6: RCIO OSCILLATOR MODE**



## 2.5 PLL Frequency Multiplier

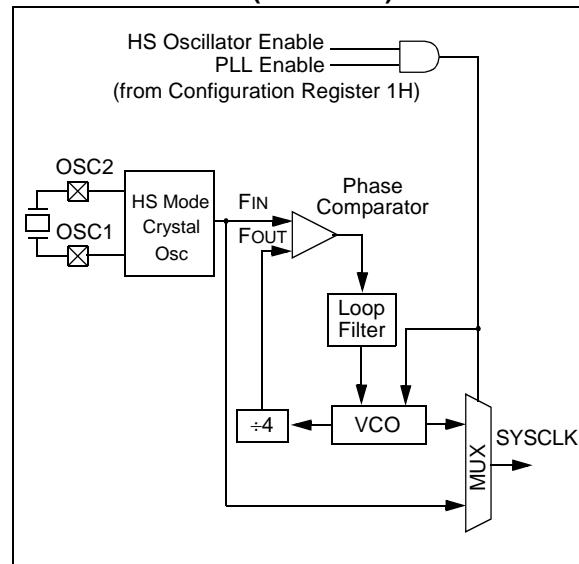
A Phase Locked Loop (PLL) circuit is provided as an option for users who wish to use a lower frequency oscillator circuit or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals or users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

### 2.5.1 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 bit is not available in this oscillator mode.

The PLL is only available to the crystal oscillator when the FOSC3:FOSC0 Configuration bits are programmed for HSPLL mode (= 0110).

**FIGURE 2-7: PLL BLOCK DIAGRAM (HS MODE)**



### 2.5.2 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 2.6.4 "PLL in INTOSC Modes"**.

## 2.6 Internal Oscillator Block

The PIC18F2525/2620/4525/4620 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 23.0 “Special Features of the CPU”](#).

The clock source frequency (INTOSC direct, INTRC direct or INTOSC postscaler) is selected by configuring the IRCC bits of the OSCCON register (page 30).

### 2.6.1 INTIO MODES

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

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

### 2.6.2 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.

### 2.6.3 OSCTUNE REGISTER

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

When the OSCTUNE register is modified, the INTOSC frequency will begin shifting to the new frequency. The INTRC clock will reach the new frequency within 8 clock cycles (approximately  $8 * 32 \mu\text{s} = 256 \mu\text{s}$ ). 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 PLL bit, 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 2.7.1 “Oscillator Control Register”](#).

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

### 2.6.4 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, PLL (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 (FOSC3:FOSC0 = 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 PLL 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.

### 2.6.5 INTOSC FREQUENCY DRIFT

The factory calibrates the internal oscillator block output (INTOSC) for 8 MHz. However, this frequency may drift as VDD or temperature changes, which can affect the controller operation in a variety of ways. It is possible to adjust the INTOSC frequency by modifying the value in the OSCTUNE register. This has no effect on the INTRC clock source frequency.

Tuning the INTOSC source requires knowing when to make the adjustment, in which direction it should be made, and in some cases, how large a change is needed. Three compensation techniques are discussed in [Section 2.6.5.1 “Compensating with the EUSART”](#), [Section 2.6.5.2 “Compensating with the Timers”](#) and [Section 2.6.5.3 “Compensating with the CCP Module in Capture Mode”](#), but other techniques may be used.

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

R/W-0	R/W-0 <sup>(1)</sup>	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	<b>INTSRC:</b> 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 INTOSC internal oscillator
bit 6	<b>PLLEN:</b> 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	<b>Unimplemented:</b> Read as '0'
bit 4-0	<b>TUN4:TUN0:</b> Frequency Tuning bits 011111 = Maximum frequency • • 000001 000000 = Center frequency. Oscillator module is running at the calibrated frequency. 111111 • • 100000 = Minimum frequency

**Note 1:** Available only in certain oscillator configurations; otherwise, this bit is unavailable and reads as '0'. See [Section 2.6.4 "PLL in INTOSC Modes"](#) for details.

### 2.6.5.1 Compensating with the EUSART

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

### 2.6.5.2 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.

### 2.6.5.3 Compensating with the CCP Module in Capture Mode

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

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

# PIC18F2525/2620/4525/4620

## 2.7 Clock Sources and Oscillator Switching

Like previous PIC18 devices, the PIC18F2525/2620/4525/4620 family includes a feature that allows the device clock source to be switched from the main oscillator to an alternate, low-frequency clock source. PIC18F2525/2620/4525/4620 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 FOSC3:FOSC0 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.

PIC18F2525/2620/4525/4620 devices offer the Timer1 oscillator as a secondary oscillator. This oscillator, in all power-managed modes, is often the time base for functions such as a Real-Time Clock (RTC).

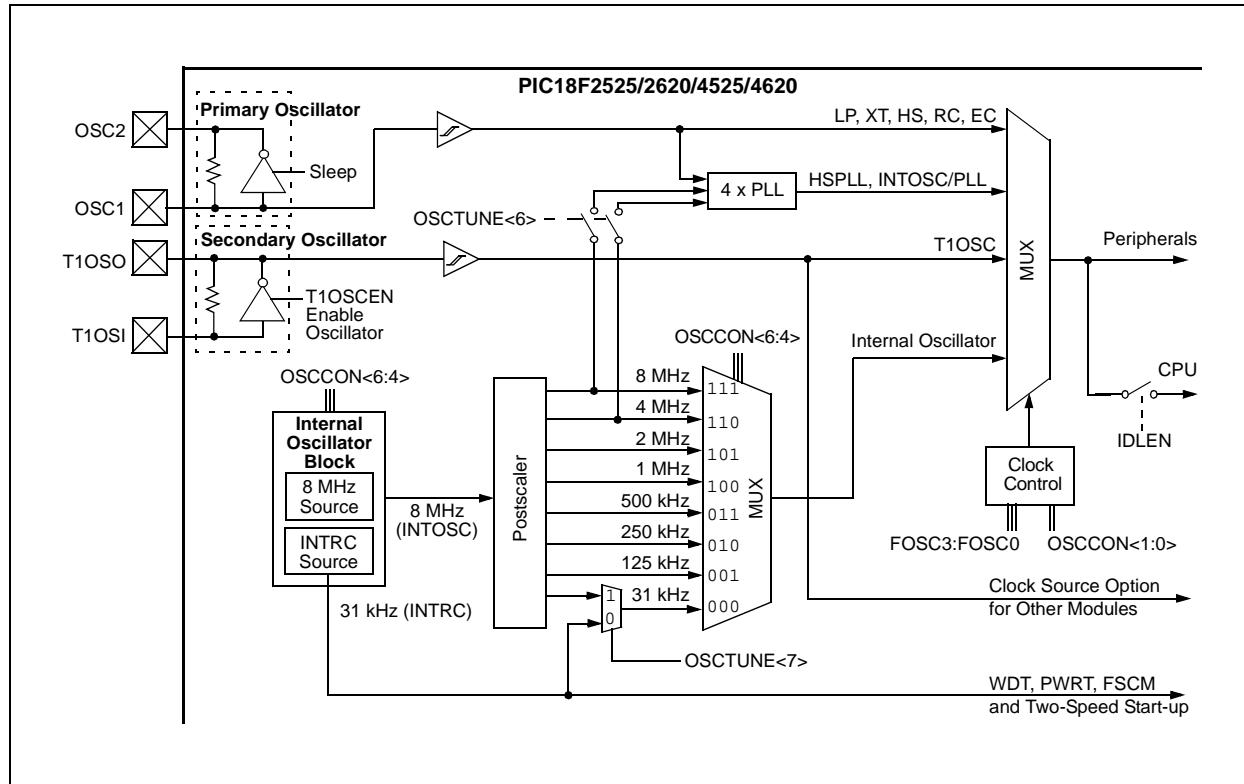
Most often, a 32.768 kHz watch crystal is connected between the RC0/T1OSO/T13CKI and RC1/T1OSI pins. Like the LP Oscillator mode circuit, loading capacitors are also connected from each pin to ground.

The Timer1 oscillator is discussed in greater detail in **Section 12.3 “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 PIC18F2525/2620/4525/4620 devices are shown in Figure 2-8. See **Section 23.0 “Special Features of the CPU”** for Configuration register details.

**FIGURE 2-8: PIC18F2525/2620/4525/4620 CLOCK DIAGRAM**



## 2.7.1 OSCILLATOR CONTROL REGISTER

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

The System Clock Select bits, SCS1:SCS0, select the clock source. The available clock sources are the primary clock (defined by the FOSC3:FOSC0 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 (IRCF2:IRCF0) select the frequency output of the internal oscillator block to drive the device clock. The choices are the INTRC source, the INTOSC source (8 MHz) or one of the frequencies derived from the INTOSC postscaler (31.25 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. On device Resets, the default output frequency of the internal oscillator block is set at 1 MHz.

When a nominal output frequency of 31 kHz is selected (IRCF2:IRCF0 = 000), users may choose which internal oscillator acts as the source. This is done with the INTSRC bit in the OSCTUNE register (OSCTUNE<7>). Setting this bit selects INTOSC as a 31.25 kHz clock source by enabling the divide-by-256 output of the INTOSC postscaler. Clearing INTSRC selects INTRC (nominally 31 kHz) as the clock source.

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

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

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

The use of the flag and control bits in the OSCCON register is discussed in more detail in **Section 3.0 “Power-Managed Modes”**.

**Note 1:** The Timer1 oscillator must be enabled to select the secondary clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 Control register (T1CON<3>). If the Timer1 oscillator is not enabled, then any attempt to select a secondary clock source will be ignored.

**2:** It is recommended that the Timer1 oscillator be operating and stable before selecting the secondary clock source or a very long delay may occur while the Timer1 oscillator starts.

## 2.7.2 OSCILLATOR TRANSITIONS

PIC18F2525/2620/4525/4620 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 3.1.2 “Entering Power-Managed Modes”**.

# PIC18F2525/2620/4525/4620

## REGISTER 2-2: 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	<b>IDLEN:</b> Idle Enable bit 1 = Device enters Idle mode on SLEEP instruction 0 = Device enters Sleep mode on SLEEP instruction
bit 6-4	<b>IRCF2:IRCF0:</b> 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	<b>OSTS:</b> Oscillator Start-up Timer Time-out Status bit <sup>(1)</sup> 1 = Oscillator Start-up Timer (OST) time-out has expired; primary oscillator is running 0 = Oscillator Start-up Timer (OST) time-out is running; primary oscillator is not ready
bit 2	<b>IOFS:</b> INTOSC Frequency Stable bit 1 = INTOSC frequency is stable 0 = INTOSC frequency is not stable
bit 1-0	<b>SCS1:SCS0:</b> System Clock Select bits 1x = Internal oscillator block 01 = Secondary (Timer1) oscillator 00 = Primary oscillator

**Note 1:** Reset state depends on state of the IESO Configuration bit.

**2:** Source selected by the INTSRC bit (OSCTUNE<7>), see text.

**3:** Default output frequency of INTOSC on Reset.

## 2.8 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 23.2 “Watchdog Timer (WDT)**, **Section 23.3 “Two-Speed Start-up”** and **Section 23.4 “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 26.2 “DC Characteristics”**.

## 2.9 Power-up Delays

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

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up (parameter 33, Table 26-10). It is enabled by clearing (= 0) the PWRTE 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 26-10), following POR, while the controller becomes ready to execute instructions. This delay runs concurrently with any other delays. This may be the only delay that occurs when any of the EC, RC or INTIO modes are used as the primary clock source.

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

OSC Mode	OSC1 Pin	OSC2 Pin
RC, INTIO1	Floating, external resistor should pull high	At logic low (clock/4 output)
RCIO	Floating, external resistor should pull high	Configured as PORTA, bit 6
INTIO2	Configured as PORTA, bit 7	Configured as PORTA, bit 6
ECIO	Floating, pulled by external clock	Configured as PORTA, bit 6
EC	Floating, pulled by external clock	At logic low (clock/4 output)
LP, XT and HS	Feedback inverter disabled at quiescent voltage level	Feedback inverter disabled at quiescent voltage level

**Note:** See Table 4-2 in **Section 4.0 “Reset”** for time-outs due to Sleep and MCLR Reset.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 3.0 POWER-MANAGED MODES

PIC18F2525/2620/4525/4620 devices offer a total of seven operating modes for more efficient power management. These modes provide a variety of options for selective power conservation in applications where resources may be limited (i.e., battery-powered devices).

There are three categories of power-managed modes:

- Run modes
- Idle modes
- Sleep mode

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

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

### 3.1 Selecting Power-Managed Modes

Selecting a power-managed mode requires two decisions: if the CPU is to be clocked or not and the selection of a clock source. The IDLEN bit (OSCCON<7>) controls CPU clocking, while the SCS1:SCS0 bits (OSCCON<1:0>) select the clock source. The individual modes, bit settings, clock sources and affected modules are summarized in Table 3-1.

TABLE 3-1: POWER-MANAGED MODES

Mode	OSCCON Bits<7,1:0>		Module Clocking		Available Clock and Oscillator Source
	IDLEN <sup>(1)</sup>	SCS1:SCS0	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 and Internal Oscillator Block <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.

### 3.1.1 CLOCK SOURCES

The SCS1:SCS0 bits allow the selection of one of three clock sources for power-managed modes. They are:

- the primary clock, as defined by the FOSC3:FOSC0 Configuration bits
- the secondary clock (the Timer1 oscillator)
- the internal oscillator block (for RC modes)

### 3.1.2 ENTERING POWER-MANAGED MODES

Switching from one power-managed mode to another begins by loading the OSCCON register. The SCS1:SCS0 bits select the clock source and determine which Run or Idle mode is to be used. Changing these bits causes an immediate switch to the new clock source, assuming that it is running. The switch may also be subject to clock transition delays. These are discussed in **Section 3.1.3 “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.

### 3.1.3 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 FOSC3:FOSC0 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.

### 3.1.4 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.

## 3.2 Run Modes

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

### 3.2.1 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 23.3 “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 2.7.1 “Oscillator Control Register”**).

### 3.2.2 SEC\_RUN MODE

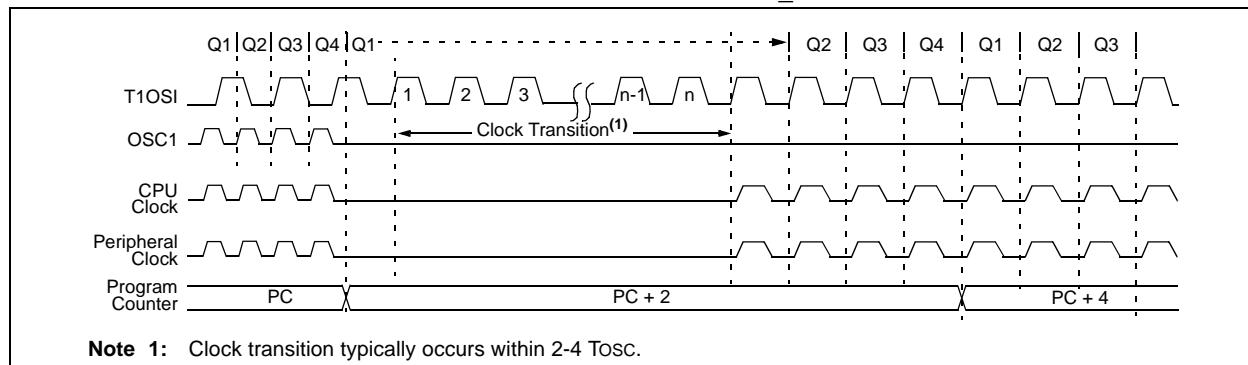
The SEC\_RUN mode is the compatible mode to the “clock switching” feature offered in other PIC18 devices. In this mode, the CPU and peripherals are clocked from the Timer1 oscillator. This gives users the option of lower power consumption while still using a high-accuracy clock source.

SEC\_RUN mode is entered by setting the SCS1:SCS0 bits to ‘01’. The device clock source is switched to the Timer1 oscillator (see Figure 3-1), the primary oscillator is shut down, the T1RUN bit (T1CON<6>) is set and the OSTS bit is cleared.

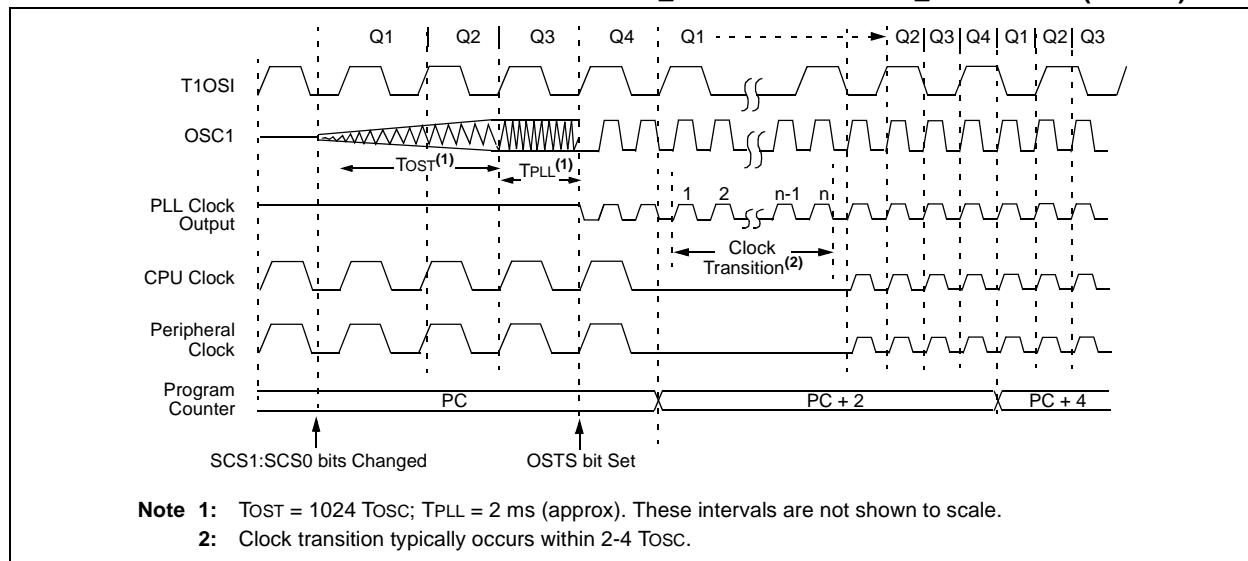
**Note:** The Timer1 oscillator should already be running prior to entering SEC\_RUN mode. If the T1OSCEN bit is not set when the SCS1:SCS0 bits are set to ‘01’, entry to SEC\_RUN mode will not occur. If the Timer1 oscillator is enabled, but not yet running, device clocks will be delayed until the oscillator has started. In such situations, initial oscillator operation is far from stable and unpredictable operation may result.

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 3-2). 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 3-1: TRANSITION TIMING FOR ENTRY TO SEC\_RUN MODE**



**FIGURE 3-2: TRANSITION TIMING FROM SEC\_RUN MODE TO PRI\_RUN MODE (HSPLL)**



### 3.2.3 RC\_RUN MODE

In RC\_RUN mode, the CPU and peripherals are clocked from the internal oscillator block using the INTOSC multiplexer. In this mode, 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 3-3), 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.

# PIC18F2525/2620/4525/4620

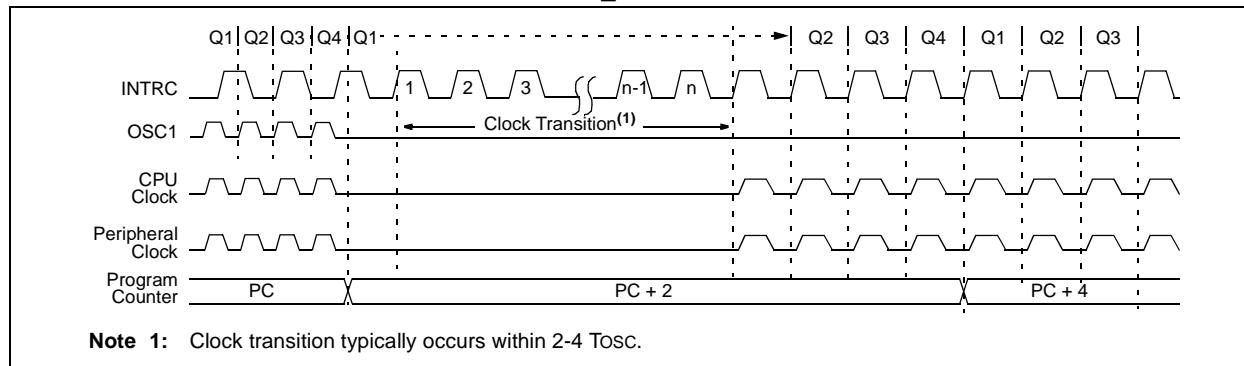
If the IRCF bits and the INTSRC bit are all clear, the INTOSC output is not enabled and the IOFS bit will remain clear; there will be no indication of the current clock source. The INTRC source is providing the device clocks.

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

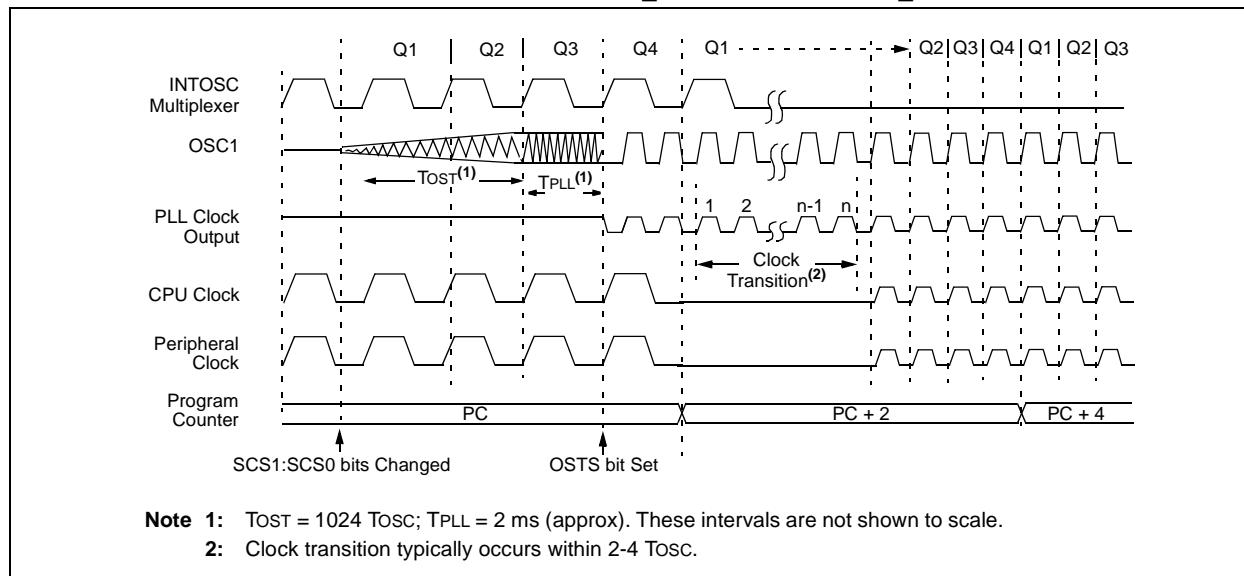
If the IRCF bits were previously at a non-zero value, or if INTSRC was set before setting SCS1 and the INTOSC source was already stable, the IOFS bit will remain set.

On transitions from RC\_RUN mode to PRI\_RUN mode, the device continues to be clocked from the INTOSC multiplexer while the primary clock is started. When the primary clock becomes ready, a clock switch to the primary clock occurs (see Figure 3-4). When the clock switch is complete, the IOFS bit is cleared, the OSTS bit is set and the primary clock is providing the device clock. The IDLEN and SCS bits are not affected by the switch. The INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

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



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



### 3.3 Sleep Mode

The power-managed Sleep mode in the PIC18F2525/2620/4525/4620 devices is identical to the legacy Sleep mode offered in all other PIC devices. It is entered by clearing the IDLEN bit (the default state on device Reset) and executing the *SLEEP* instruction. This shuts down the selected oscillator (Figure 3-5). All clock source status bits are cleared.

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

When a wake event occurs in Sleep mode (by interrupt, Reset or WDT time-out), the device will not be clocked until the clock source selected by the SCS1:SCS0 bits becomes ready (see Figure 3-6), 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 23.0 “Special Features of the CPU”**). In either case, the OSTST bit is set when the primary clock is providing the device clocks. The IDLEN and SCS bits are not affected by the wake-up.

### 3.4 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.

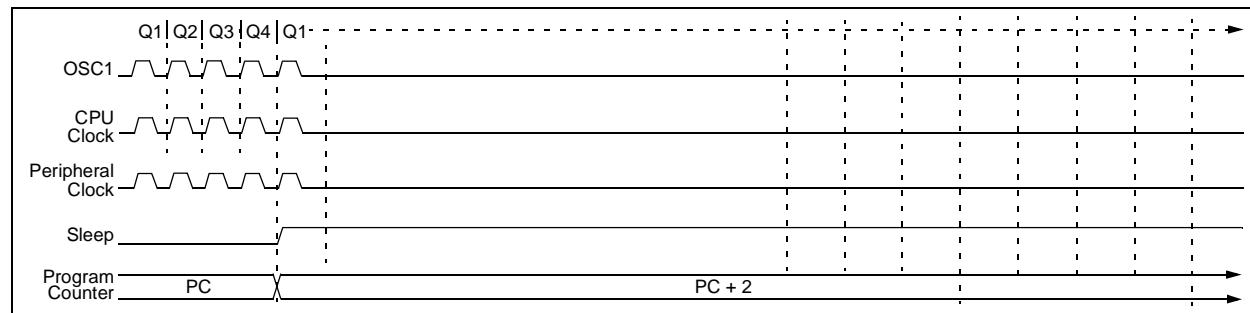
If the IDLEN bit is set to a '1' when a *SLEEP* instruction is executed, the peripherals will be clocked from the clock source selected using the SCS1:SCS0 bits; however, the CPU will not be clocked. The clock source status bits are not affected. Setting IDLEN and executing a *SLEEP* instruction provides a quick method of switching from a given Run mode to its corresponding Idle mode.

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

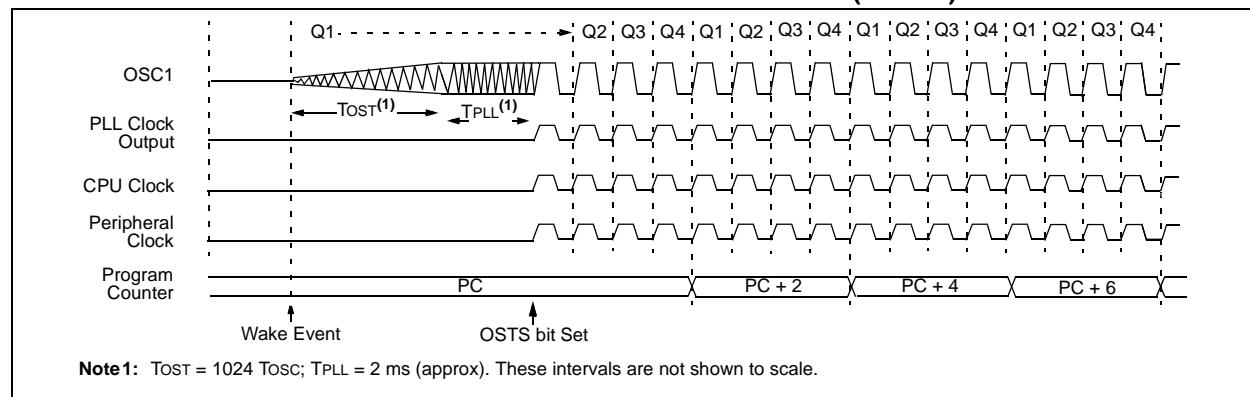
Since the CPU is not executing instructions, the only exits from any of the Idle modes are by interrupt, WDT time-out or a Reset. When a wake event occurs, CPU execution is delayed by an interval of *T<sub>CSD</sub>* (parameter 38, Table 26-10) while it becomes ready to execute code. When the CPU begins executing code, it resumes with the same clock source for the current Idle mode. For example, when waking from RC\_IDLE mode, the internal oscillator block will clock the CPU and peripherals (in other words, RC\_RUN mode). The IDLEN and SCS bits are not affected by the wake-up.

While in any Idle mode or Sleep mode, a WDT time-out will result in a WDT wake-up to the Run mode currently specified by the SCS1:SCS0 bits.

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



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



# PIC18F2525/2620/4525/4620

### 3.4.1 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 FOSC3:FOSC0 Configuration bits. The OSTS bit remains set (see Figure 3-7).

When a wake event occurs, the CPU is clocked from the primary clock source. A delay of interval TcSD is required between the wake event and when code execution starts. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTS bit remains set. The IDLEN and SCS bits are not affected by the wake-up (see Figure 3-8).

### 3.4.2 SEC\_IDLE MODE

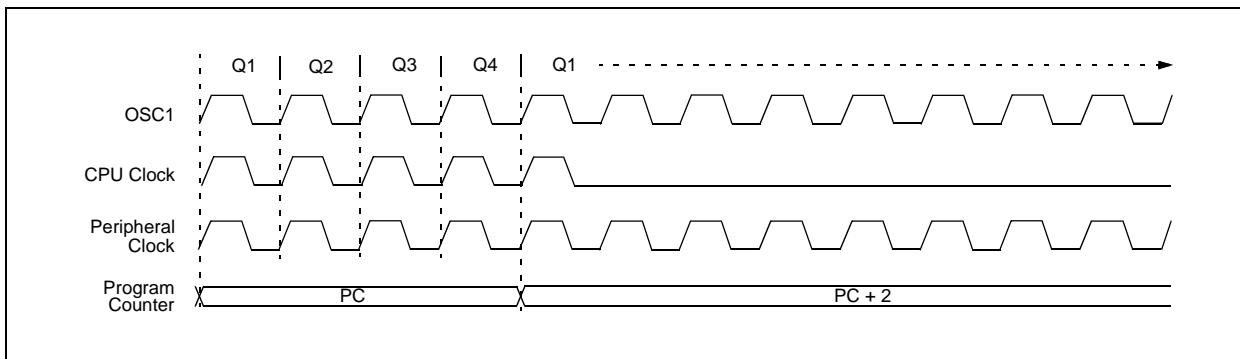
In SEC\_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the Timer1 oscillator. This mode is entered from SEC\_RUN by

setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set the IDLEN bit first, then set the SCS1:SCS0 bits to ‘01’ and execute SLEEP. When the clock source is switched to the Timer1 oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the T1RUN bit is set.

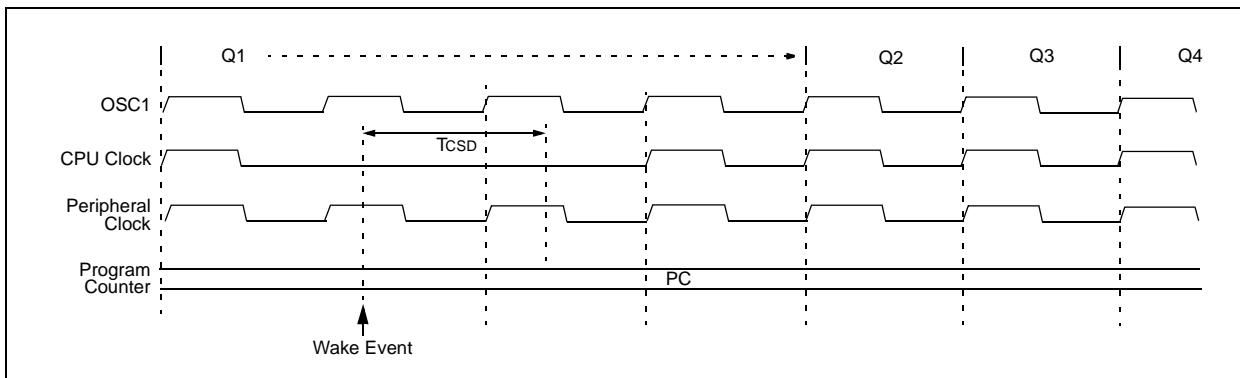
When a wake event occurs, the peripherals continue to be clocked from the Timer1 oscillator. After an interval of TcSD following the wake event, the CPU begins executing code being clocked by the Timer1 oscillator. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run (see Figure 3-8).

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

**FIGURE 3-7: TRANSITION TIMING FOR ENTRY TO IDLE MODE**



**FIGURE 3-8: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE**



### 3.4.3 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 IRFC 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 IRFC 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 TIOB<sub>ST</sub> (parameter 39, Table 26-10). Clocks to the peripherals continue while the INTOSC source stabilizes. If the IRFC 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 IRFC 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.

## 3.5 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 3.2 “Run Modes”**, **Section 3.3 “Sleep Mode”** and **Section 3.4 “Idle Modes”**).

### 3.5.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit from an Idle mode 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 TCSD following the wake event is required when leaving Sleep and Idle modes. This delay is required for the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

### 3.5.2 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 3.2 “Run Modes”** and **Section 3.3 “Sleep Mode”**). If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see **Section 23.2 “Watchdog Timer (WDT)”**).

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

### 3.5.3 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 3-2.

Code execution can begin before the primary clock becomes ready. If either the Two-Speed Start-up (see **Section 23.3 “Two-Speed Start-up”**) or Fail-Safe Clock Monitor (see **Section 23.4 “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.

# PIC18F2525/2620/4525/4620

## 3.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

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

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

In these instances, the primary clock source either does not require an oscillator start-up delay since it is already running (PRI\_IDLE), or normally does not require an oscillator start-up delay (RC, EC and INTIO Oscillator modes). However, a fixed delay of interval Tcsd following the wake event is still required when leaving Sleep and Idle modes to allow the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

**TABLE 3-2: 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	Tcsd <sup>(1)</sup>	OSTS
	HSPLL		
	EC, RC		IOFS
	INTOSC <sup>(2)</sup>		
T1OSC	LP, XT, HS	Tost <sup>(3)</sup>	OSTS
	HSPLL	Tost + t <sub>rc</sub> <sup>(3)</sup>	
	EC, RC	Tcsd <sup>(1)</sup>	
	INTOSC <sup>(2)</sup>	TIOBST <sup>(4)</sup>	IOFS
INTOSC <sup>(3)</sup>	LP, XT, HS	Tost <sup>(3)</sup>	OSTS
	HSPLL	Tost + t <sub>rc</sub> <sup>(3)</sup>	
	EC, RC	Tcsd <sup>(1)</sup>	IOFS
	INTOSC <sup>(2)</sup>	None	
None (Sleep mode)	LP, XT, HS	Tost <sup>(3)</sup>	OSTS
	HSPLL	Tost + t <sub>rc</sub> <sup>(3)</sup>	
	EC, RC	Tcsd <sup>(1)</sup>	
	INTOSC <sup>(2)</sup>	TIOBST <sup>(4)</sup>	IOFS

**Note 1:** Tcsd (parameter 38) is a required delay when waking from Sleep and all Idle modes and runs concurrently with any other required delays (see **Section 3.4 “Idle Modes”**). On Reset, INTOSC defaults to 1 MHz.

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

**3:** Tost 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 TPLL.

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

## 4.0 RESET

The PIC18F2525/2620/4525/4620 devices differentiate between various kinds of Reset:

- Power-on Reset (POR)
- MCLR Reset during normal operation
- 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 MCLR, POR and BOR and covers the operation of the various start-up timers. Stack Reset events are covered in **Section 5.1.2.4 “Stack Full and Underflow Resets”**. WDT Resets are covered in **Section 23.2 “Watchdog Timer (WDT)”**.

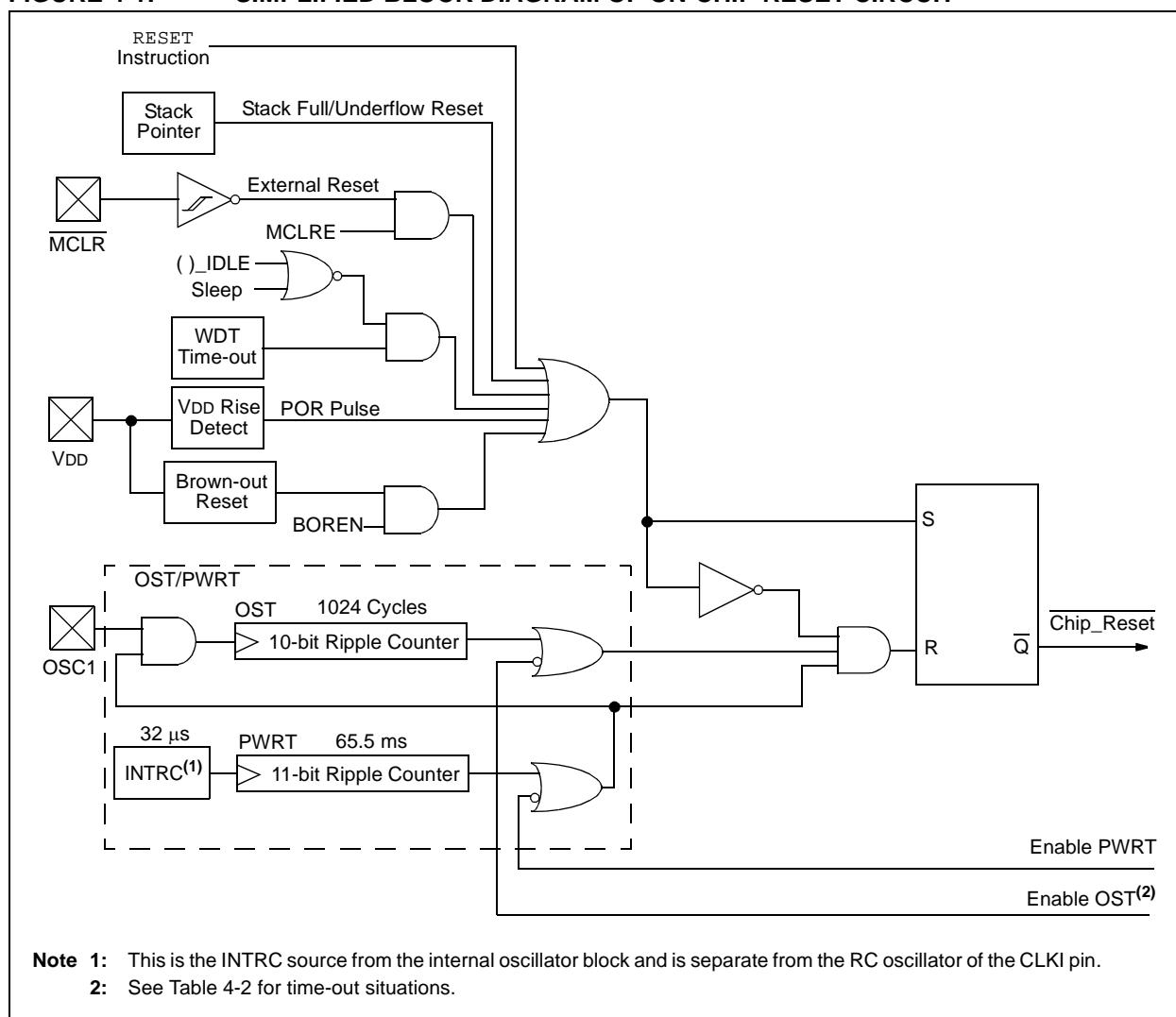
A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 4-1.

## 4.1 RCON Register

Device Reset events are tracked through the RCON register (Register 4-1). The lower five bits of the register indicate that a specific Reset event has occurred. In most cases, these bits can only be cleared by the event and must be set by the application after the event. The state of these flag bits, taken together, can be read to indicate the type of Reset that just occurred. This is described in more detail in **Section 4.6 “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 4.4 “Brown-out Reset (BOR)”**.

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



# PIC18F2525/2620/4525/4620

## REGISTER 4-1: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1 <sup>(1)</sup>	U-0	R/W-1	R-1	R-1	R/W-0 <sup>(2)</sup>	R/W-0
IPEN	SBOREN	—	RI	TO	PD	POR	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	<b>IPEN:</b> Interrupt Priority Enable bit 1 = Enable priority levels on interrupts 0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)
bit 6	<b>SBOREN:</b> BOR Software Enable bit <sup>(1)</sup> <u>If BOREN1:BOREN0 = 01:</u> 1 = BOR is enabled 0 = BOR is disabled <u>If BOREN1:BOREN0 = 00, 10 or 11:</u> Bit is disabled and read as '0'.
bit 5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>RI:</b> 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	<b>TO:</b> Watchdog Time-out Flag bit 1 = Set by power-up, CLRWDT instruction or SLEEP instruction 0 = A WDT time-out occurred
bit 2	<b>PD:</b> 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	<b>POR:</b> 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	<b>BOR:</b> Brown-out Reset Status bit 1 = A Brown-out Reset has not occurred (set by firmware only) 0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

- Note 1:** If SBOREN is enabled, its Reset state is '1'; otherwise, it is '0'.  
**2:** The actual Reset value of POR is determined by the type of device Reset. See the notes following this register and **Section 4.6 "Reset State of Registers"** for additional information.

- Note 1:** It is recommended that the 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 BOR is '0' and POR is '1' (assuming that POR was set to '1' by software immediately after Power-on Reset).

## 4.2 Master Clear (MCLR)

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

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

In PIC18F2525/2620/4525/4620 devices, the MCLR input can be disabled with the MCLRE Configuration bit. When MCLR is disabled, the pin becomes a digital input. See **Section 9.5 “PORTE, TRISE and LATE Registers”** for more information.

## 4.3 Power-on Reset (POR)

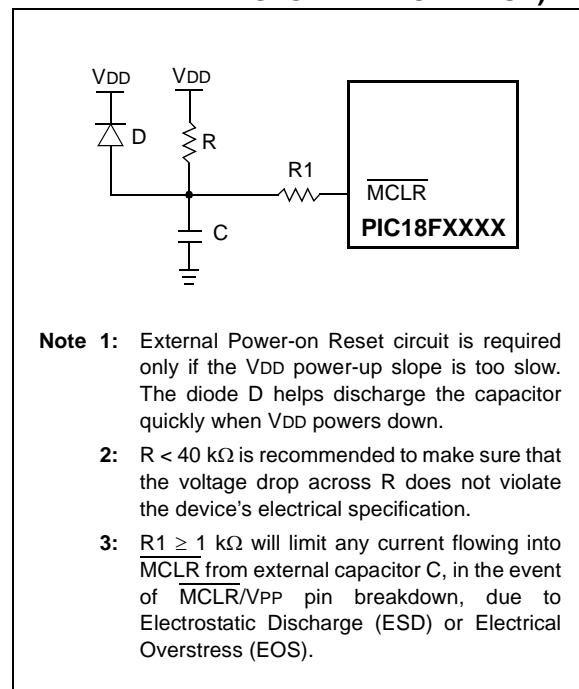
A Power-on Reset pulse is generated on-chip whenever VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.

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

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

POR events are captured by the 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. 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 4-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)**



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

**2:**  $R < 40\text{ k}\Omega$  is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

**3:**  $R1 \geq 1\text{ k}\Omega$  will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

## 4.4 Brown-out Reset (BOR)

PIC18F2525/2620/4525/4620 devices implement a BOR circuit that provides the user with a number of configuration and power-saving options. The BOR is controlled by the BORV1:BORV0 and BOREN1:BOREN0 Configuration bits. There are a total of four BOR configurations which are summarized in Table 4-1.

The BOR threshold is set by the BORV1:BORV0 bits. If BOR is enabled (any values of BOREN1:BOREN0, except '00'), any drop of VDD below VBOR (parameter D005) for greater than TBOR (parameter 35) will reset the device. A Reset may or may not occur if VDD falls below VBOR for less than TBOR. The chip will remain in Brown-out Reset until VDD rises above VBOR.

If the Power-up Timer is enabled, it will be invoked after VDD rises above VBOR; it then will keep the chip in Reset for an additional time delay, TPWRT (parameter 33). If VDD drops below VBOR while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be initialized. Once VDD rises above VBOR, the Power-up Timer will execute the additional time delay.

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

### 4.4.1 SOFTWARE ENABLED BOR

When BOREN1:BOREN0 = 01, the BOR can be enabled or disabled by the user in software. This is done with the control bit, SBOREN (RCON<6>). Setting SBOREN enables the BOR to function as previously described. Clearing SBOREN disables the BOR entirely. The SBOREN bit operates only in this mode; otherwise it is read as '0'.

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

**Note:** Even when BOR is under software control, the BOR Reset voltage level is still set by the BORV1:BORV0 Configuration bits. It cannot be changed in software.

### 4.4.2 DETECTING BOR

When BOR is enabled, the BOR bit always resets to '0' on any BOR or POR event. This makes it difficult to determine if a BOR event has occurred just by reading the state of 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 BOR is '0' while POR is '1', it can be reliably assumed that a BOR event has occurred.

### 4.4.3 DISABLING BOR IN SLEEP MODE

When BOREN1:BOREN0 = 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 4-1: BOR CONFIGURATIONS

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

## 4.5 Device Reset Timers

PIC18F2525/2620/4525/4620 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

### 4.5.1 POWER-UP TIMER (PWRT)

The Power-up Timer (PWRT) of PIC18F2525/2620/4525/4620 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 PWRTEN Configuration bit.

### 4.5.2 OSCILLATOR START-UP TIMER (OST)

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

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

### 4.5.3 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.

### 4.5.4 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 4-3, Figure 4-4, Figure 4-5, Figure 4-6 and Figure 4-7 all depict time-out sequences on power-up, with the Power-up Timer enabled and the device operating in HS Oscillator mode. Figures 4-3 through 4-6 also apply to devices operating in XT or LP modes. For devices in RC mode and with the PWRT disabled, there will be no time-out at all.

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

**TABLE 4-2: TIME-OUT IN VARIOUS SITUATIONS**

Oscillator Configuration	Power-up <sup>(2)</sup> and Brown-out Reset		Exit from Power-Managed Mode
	<u>PWRTE</u> N = 0	<u>PWRTE</u> N = 1	
HSPLL	66 ms <sup>(1)</sup> + 1024 Tosc + 2 ms <sup>(2)</sup>	1024 Tosc + 2 ms <sup>(2)</sup>	1024 Tosc + 2 ms <sup>(2)</sup>
HS, XT, LP	66 ms <sup>(1)</sup> + 1024 Tosc	1024 Tosc	1024 Tosc
EC, ECIO	66 ms <sup>(1)</sup>	—	—
RC, RCIO	66 ms <sup>(1)</sup>	—	—
INTIO1, INTIO2	66 ms <sup>(1)</sup>	—	—

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

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

# PIC18F2525/2620/4525/4620

FIGURE 4-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO V<sub>DD</sub>, V<sub>DD</sub> RISE < T<sub>PWRT</sub>)

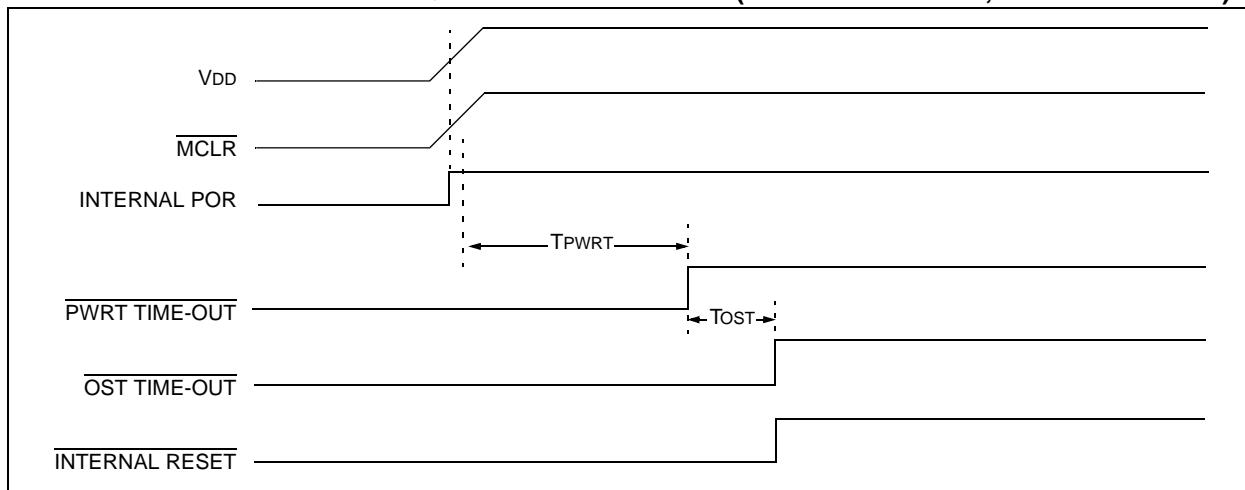


FIGURE 4-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO V<sub>DD</sub>): CASE 1

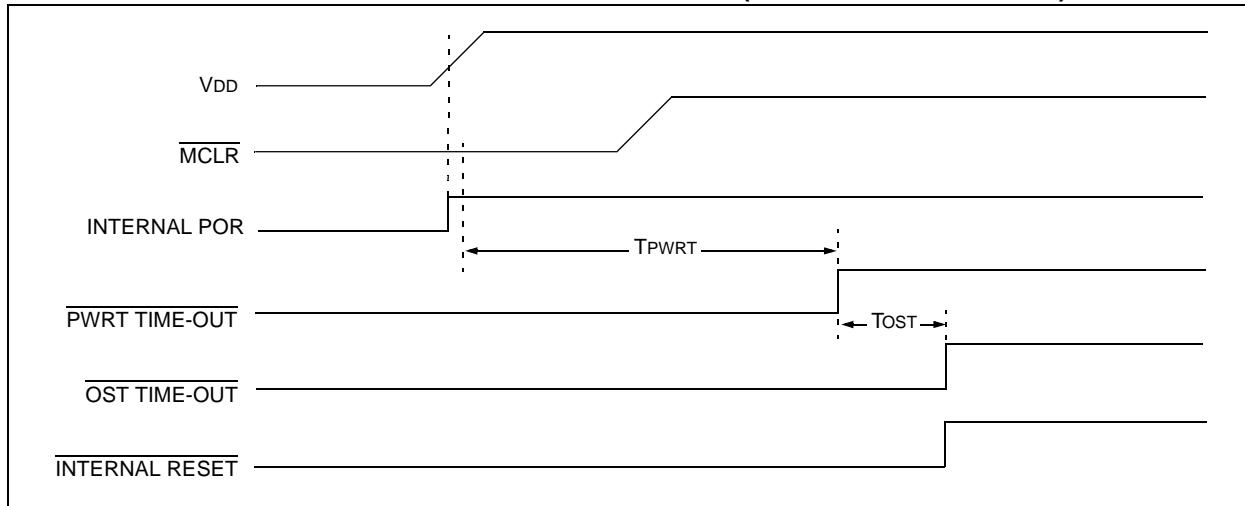
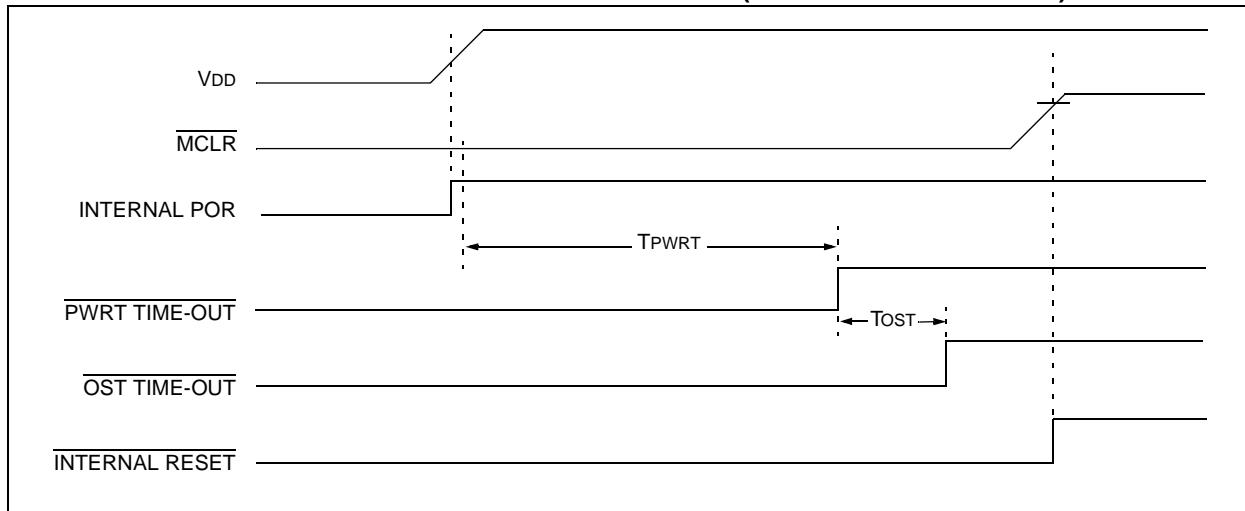


FIGURE 4-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO V<sub>DD</sub>): CASE 2



# PIC18F2525/2620/4525/4620

FIGURE 4-6: SLOW RISE TIME (MCLR TIED TO VDD, VDD RISE > TPWRT)

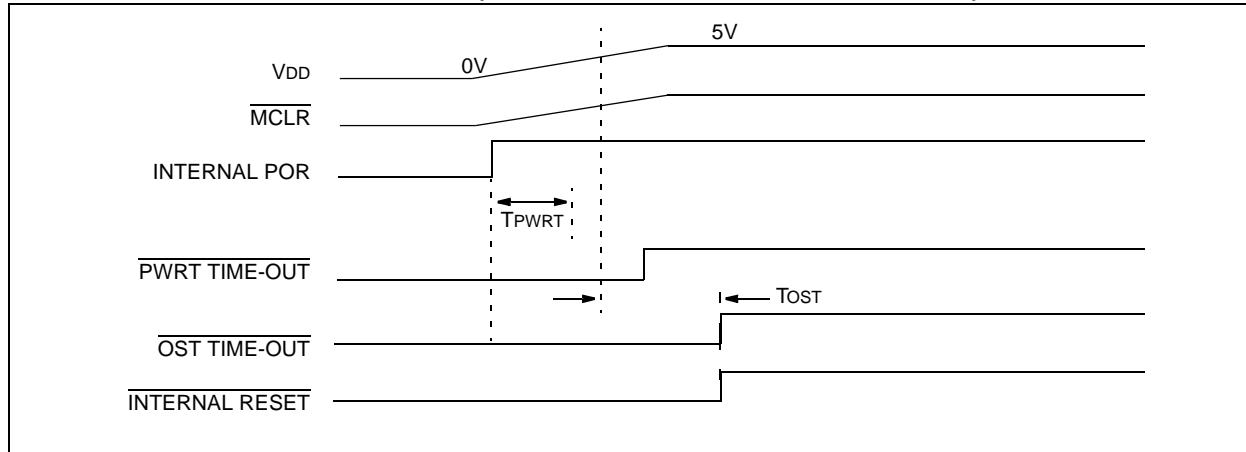
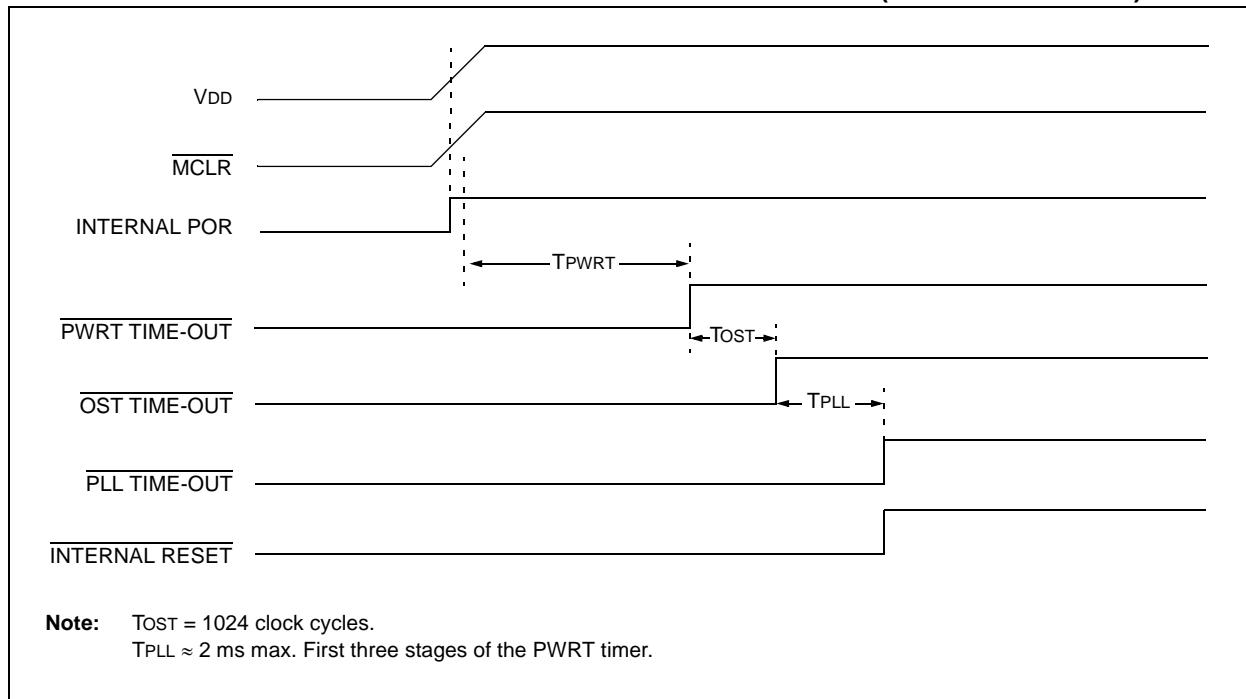


FIGURE 4-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED (MCLR TIED TO VDD)



# PIC18F2525/2620/4525/4620

## 4.6 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, RI, TO, PD, POR and BOR, are set or cleared differently in different Reset situations, as indicated in Table 4-3. These bits are used in software to determine the nature of the Reset.

Table 4-4 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 4-3: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER**

Condition	Program Counter	RCON Register						STKPTR Register	
		SBOREN	RI	TO	PD	POR	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
MCLR during power-managed Run Modes	0000h	u <sup>(2)</sup>	u	1	u	u	u	u	u
MCLR 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 mode	0000h	u <sup>(2)</sup>	u	0	u	u	u	u	u
MCLR 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 (BOREN1:BOREN0 Configuration bits = 01 and SBOREN = 1); otherwise, the Reset state is '0'.

# PIC18F2525/2620/4525/4620

TABLE 4-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Applicable Devices				Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt	
TOSU	2525	2620	4525	4620	---	0 0000	---	0 uuuu <sup>(3)</sup>
TOSH	2525	2620	4525	4620	0000	0000	0000 0000	uuuu uuuu <sup>(3)</sup>
TOSL	2525	2620	4525	4620	0000	0000	0000 0000	uuuu uuuu <sup>(3)</sup>
STKPTR	2525	2620	4525	4620	00	-0 0000	uu-0 0000	uu-u uuuu <sup>(3)</sup>
PCLATU	2525	2620	4525	4620	---	0 0000	---	0 0000
PCLATH	2525	2620	4525	4620	0000	0000	0000 0000	uuuu uuuu
PCL	2525	2620	4525	4620	0000	0000	0000 0000	PC + 2 <sup>(2)</sup>
TBLPTRU	2525	2620	4525	4620	--	00 0000	--00 0000	--uu uuuu
TBLPTRH	2525	2620	4525	4620	0000	0000	0000 0000	uuuu uuuu
TBLPTRL	2525	2620	4525	4620	0000	0000	0000 0000	uuuu uuuu
TABLAT	2525	2620	4525	4620	0000	0000	0000 0000	uuuu uuuu
PRODH	2525	2620	4525	4620	xxxx	xxxx	uuuu uuuu	uuuu uuuu
PRODL	2525	2620	4525	4620	xxxx	xxxx	uuuu uuuu	uuuu uuuu
INTCON	2525	2620	4525	4620	0000	000x	0000 000u	uuuu uuuu <sup>(1)</sup>
INTCON2	2525	2620	4525	4620	1111	-1-1	1111 -1-1	uuuu -u-u <sup>(1)</sup>
INTCON3	2525	2620	4525	4620	11	-0 0-00	11-0 0-00	uu-u u-uu <sup>(1)</sup>
INDF0	2525	2620	4525	4620	N/A	N/A	N/A	N/A
POSTINCO	2525	2620	4525	4620	N/A	N/A	N/A	N/A
POSTDEC0	2525	2620	4525	4620	N/A	N/A	N/A	N/A
PREINC0	2525	2620	4525	4620	N/A	N/A	N/A	N/A
PLUSW0	2525	2620	4525	4620	N/A	N/A	N/A	N/A
FSR0H	2525	2620	4525	4620	----	0000	----	0000
FSR0L	2525	2620	4525	4620	xxxx	xxxx	uuuu uuuu	uuuu uuuu
WREG	2525	2620	4525	4620	xxxx	xxxx	uuuu uuuu	uuuu uuuu
INDF1	2525	2620	4525	4620	N/A	N/A	N/A	N/A
POSTINC1	2525	2620	4525	4620	N/A	N/A	N/A	N/A
POSTDEC1	2525	2620	4525	4620	N/A	N/A	N/A	N/A
PREINC1	2525	2620	4525	4620	N/A	N/A	N/A	N/A
PLUSW1	2525	2620	4525	4620	N/A	N/A	N/A	N/A

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

Shaded cells indicate conditions do not apply for the designated device.

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

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

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

**4:** See Table 4-3 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'.

# PIC18F2525/2620/4525/4620

TABLE 4-4: 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
FSR1H	2525	2620	4525	4620	---- 0000	---- 0000	---- uuuu
FSR1L	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	2525	2620	4525	4620	---- 0000	---- 0000	---- uuuu
INDF2	2525	2620	4525	4620	N/A	N/A	N/A
POSTINC2	2525	2620	4525	4620	N/A	N/A	N/A
POSTDEC2	2525	2620	4525	4620	N/A	N/A	N/A
PREINC2	2525	2620	4525	4620	N/A	N/A	N/A
PLUSW2	2525	2620	4525	4620	N/A	N/A	N/A
FSR2H	2525	2620	4525	4620	---- 0000	---- 0000	---- uuuu
FSR2L	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	2525	2620	4525	4620	--x xxxx	--u uuuu	--u uuuu
TMR0H	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
TMR0L	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	2525	2620	4525	4620	1111 1111	1111 1111	uuuu uuuu
OSCCON	2525	2620	4525	4620	0100 q000	0100 q000	uuuu uaqu
HLVDCON	2525	2620	4525	4620	0-00 0101	0-00 0101	u-uu uuuu
WDTCON	2525	2620	4525	4620	---- ---0	---- ---0	---- ---u
RCON <sup>(4)</sup>	2525	2620	4525	4620	0q-1 11q0	0q-q qquu	uq-u qquu
TMR1H	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	2525	2620	4525	4620	0000 0000	u0uu uuuu	uuuu uuuu
TMR2	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
PR2	2525	2620	4525	4620	1111 1111	uuuu uuuu	uuuu uuuu
T2CON	2525	2620	4525	4620	-000 0000	-000 0000	-uuu uuuu
SSPBUF	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
SSPCON1	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
SSPCON2	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu

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

Shaded cells indicate conditions do not apply for the designated device.

- Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).
- 2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4:** See Table 4-3 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'.

# PIC18F2525/2620/4525/4620

TABLE 4-4: 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
ADRESH	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	2525	2620	4525	4620	--00 0000	--00 0000	--uu uuuu
ADCON1	2525	2620	4525	4620	--00 0qqq	--00 0qqq	--uu uuuu
ADCON2	2525	2620	4525	4620	0-00 0000	0-00 0000	u-uu uuuu
CCPR1H	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
	2525	2620	4525	4620	--00 0000	--00 0000	--uu uuuu
CCPR2H	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	2525	2620	4525	4620	--00 0000	--00 0000	--uu uuuu
BAUDCON	2525	2620	4525	4620	0100 0-00	0100 0-00	uuuu u-uu
PWM1CON	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
ECCP1AS	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
	2525	2620	4525	4620	0000 00--	0000 00--	uuuu uu--
CVRCON	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
CMCON	2525	2620	4525	4620	0000 0111	0000 0111	uuuu uuuu
TMR3H	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	2525	2620	4525	4620	0000 0000	uuuu uuuu	uuuu uuuu
SPBRGH	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
SPBRG	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
RCREG	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
TXREG	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
TXSTA	2525	2620	4525	4620	0000 0010	0000 0010	uuuu uuuu
RCSTA	2525	2620	4525	4620	0000 000x	0000 000x	uuuu uuuu
EEADRH	2585	2680	4585	4680	---- --00	---- --00	---- --uu
EEADR	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
EEDATA	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
EECON2	2525	2620	4525	4620	0000 0000	0000 0000	0000 0000
EECON1	2525	2620	4525	4620	xx-0 x000	uu-0 u000	uu-0 u000

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

- Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).
- 2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
  - 3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
  - 4: See Table 4-3 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'.

# PIC18F2525/2620/4525/4620

TABLE 4-4: 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
IPR2	2525	2620	4525	4620	11-1 1111	11-1 1111	uu-u uuuu
PIR2	2525	2620	4525	4620	00-0 0000	00-0 0000	uu-u uuuu <sup>(1)</sup>
PIE2	2525	2620	4525	4620	00-0 0000	00-0 0000	uu-u uuuu
IPR1	2525	2620	4525	4620	1111 1111	1111 1111	uuuu uuuu
	2525	2620	4525	4620	-111 1111	-111 1111	-uuu uuuu
PIR1	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu <sup>(1)</sup>
	2525	2620	4525	4620	-000 0000	-000 0000	-uuu uuuu <sup>(1)</sup>
PIE1	2525	2620	4525	4620	0000 0000	0000 0000	uuuu uuuu
	2525	2620	4525	4620	-000 0000	-000 0000	-uuu uuuu
OSCTUNE	2525	2620	4525	4620	00-0 0000	00-0 0000	uu-u uuuu
TRISE	2525	2620	4525	4620	0000 -111	0000 -111	uuuu -uuu
TRISD	2525	2620	4525	4620	1111 1111	1111 1111	uuuu uuuu
TRISC	2525	2620	4525	4620	1111 1111	1111 1111	uuuu uuuu
TRISB	2525	2620	4525	4620	1111 1111	1111 1111	uuuu uuuu
TRISA <sup>(5)</sup>	2525	2620	4525	4620	1111 1111 <sup>(5)</sup>	1111 1111 <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>
LATE	2525	2620	4525	4620	---- -xxx	---- -uuu	---- -uuu
LATD	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA <sup>(5)</sup>	2525	2620	4525	4620	xxxx xxxx <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>
PORTE	2525	2620	4525	4620	---- xxxx	---- uuuu	---- uuuu
	2525	2620	4525	4620	---- x---	---- u---	---- u---
PORTE	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE	2525	2620	4525	4620	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTE	2525	2620	4525	4620	xx0x 0000 <sup>(5)</sup>	uu0u 0000 <sup>(5)</sup>	uuuu uuuu <sup>(5)</sup>

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

Shaded cells indicate conditions do not apply for the designated device.

- Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).
- 2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4:** See Table 4-3 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'.

## 5.0 MEMORY ORGANIZATION

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

- Program Memory
- Data RAM
- Data EEPROM

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

Additional detailed information on the operation of the Flash program memory is provided in **Section 7.0 "Flash Program Memory"**. Data EEPROM is discussed separately in **Section 6.0 "Data EEPROM Memory"**.

## 5.1 Program Memory Organization

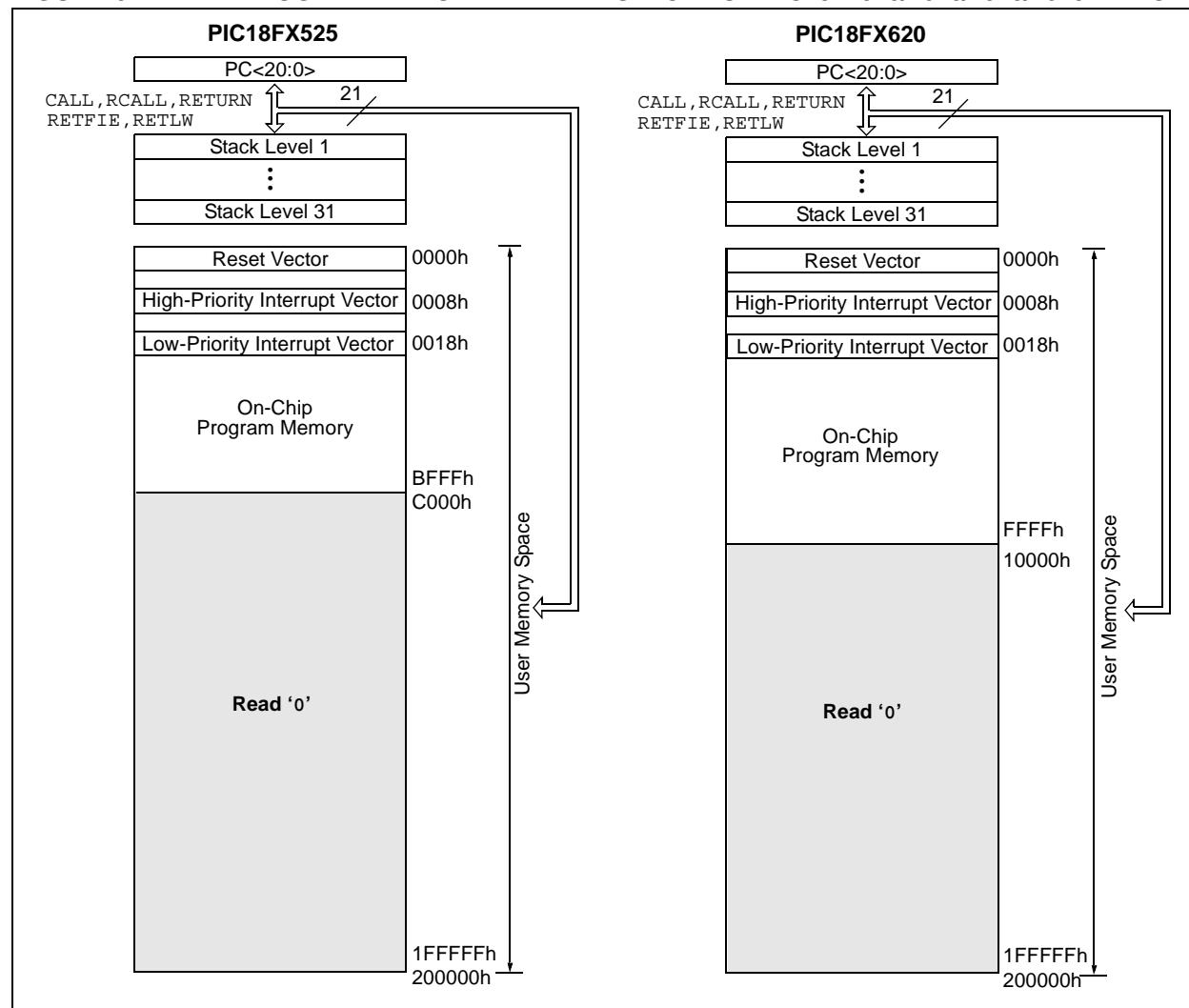
PIC18 microcontrollers implement a 21-bit program counter, which is capable of addressing a 2-Mbyte program memory space. Accessing a location between the upper boundary of the physically implemented memory and the 2-Mbyte address will return all '0's (a NOP instruction).

The PIC18F2525 and PIC18F4525 each have 48 Kbytes of Flash memory and can store up to 24,576 single-word instructions. The PIC18F2620 and PIC18F4620 each have 64 Kbytes of Flash memory and can store up to 32,768 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 PIC18FX525 and PIC18FX620 devices are shown in Figure 5-1.

**FIGURE 5-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F2525/2620/4525/4620 DEVICES**



# PIC18F2525/2620/4525/4620

## 5.1.1 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<sub>15:8</sub> 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<sub>20:16</sub> 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 5.1.4.1 “Computed GOTO”**).

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

The CALL, RCALL, 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.

## 5.1.2 RETURN ADDRESS STACK

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

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, STKPTR. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the top-of-stack Special File Registers. Data can also be pushed to, or popped from the stack, using these registers.

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

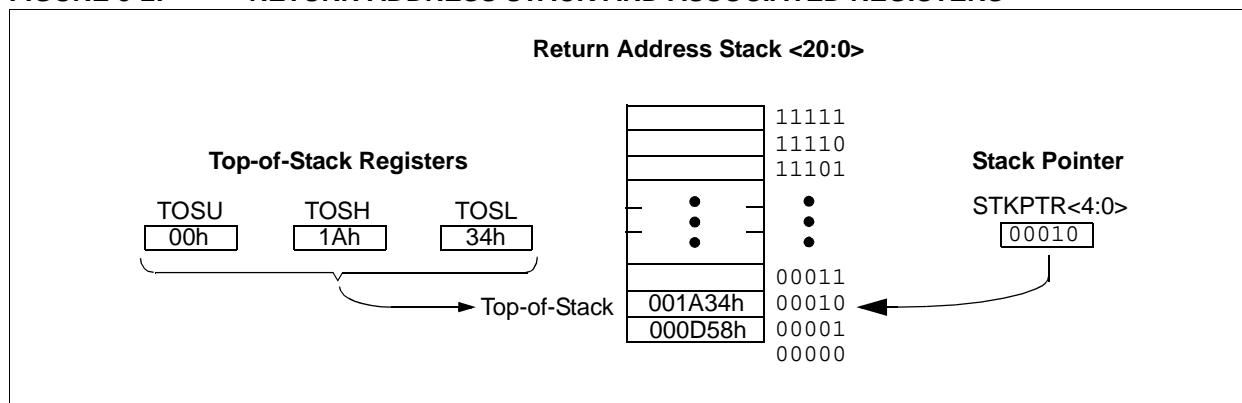
The Stack Pointer is initialized to ‘00000’ after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of ‘00000’; this is only a Reset value. Status bits indicate if the stack is full or has overflowed or has underflowed.

### 5.1.2.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 5-2). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

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

**FIGURE 5-2: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**



## 5.1.2.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 5-1) contains the Stack Pointer value, the STKFUL (Stack Full) status bit and the STKUNF (Stack Underflow) status bits. The value of the Stack Pointer can be 0 through 31. The Stack Pointer increments before values are pushed onto the stack and decrements after values are popped off the stack. 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 (RTOS) 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 23.1 “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.

## 5.1.2.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off of 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 5-1: STKPTR: STACK POINTER REGISTER

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

### Legend:

R = Readable bit

-n = Value at POR

C = Clearable only bit

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>STKFUL:</b> Stack Full Flag bit <sup>(1)</sup> 1 = Stack became full or overflowed 0 = Stack has not become full or overflowed
bit 6	<b>STKUNF:</b> Stack Underflow Flag bit <sup>(1)</sup> 1 = Stack underflow occurred 0 = Stack underflow did not occur
bit 5	<b>Unimplemented:</b> Read as '0'
bit 4-0	<b>SP4:SP0:</b> Stack Pointer Location bits

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

## 5.1.2.4 Stack Full and Underflow Resets

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

## 5.1.3 FAST REGISTER STACK

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

If both low and high-priority interrupts are enabled, the stack registers cannot be used reliably to return from low-priority interrupts. If a high-priority interrupt occurs while servicing a low-priority interrupt, the stack register values stored by the low-priority interrupt will be overwritten. In these cases, users must save the key registers in software during a low-priority interrupt.

If interrupt priority is not used, all interrupts may use the Fast Register Stack for returns from interrupt. If no interrupts are used, the Fast Register Stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the Fast Register Stack for a subroutine call, a CALL label, FAST instruction must be executed to save the STATUS, WREG and BSR registers to the Fast Register Stack. A RETURN, FAST instruction is then executed to restore these registers from the Fast Register Stack.

Example 5-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

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

```
CALL SUB1, FAST      ;STATUS, WREG, BSR
                      ;SAVED IN FAST REGISTER
                      ;STACK
.
.
.
SUB1
.
.
.
RETURN, FAST        ;RESTORE VALUES SAVED
                      ;IN FAST REGISTER STACK
```

## 5.1.4 LOOK-UP TABLES IN PROGRAM MEMORY

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

- Computed GOTO
- Table Reads

### 5.1.4.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 5-2.

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 5-2: COMPUTED GOTO USING AN OFFSET VALUE

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

### 5.1.4.2 Table Reads and Table Writes

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

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

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

## 5.2 PIC18 Instruction Cycle

### 5.2.1 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 5-3.

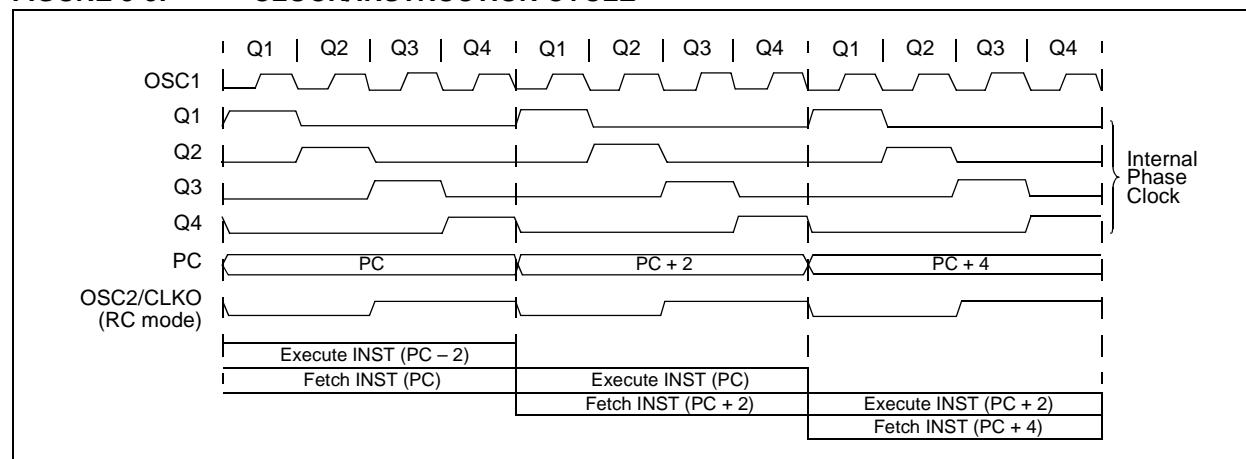
### 5.2.2 INSTRUCTION FLOW/PIPELINING

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 5-3).

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 5-3: CLOCK/INSTRUCTION CYCLE**



**EXAMPLE 5-3: INSTRUCTION PIPELINE FLOW**

	Tcy0	Tcy1	Tcy2	Tcy3	Tcy4	Tcy5
1. MOVLW 55h		Fetch 1	Execute 1			
2. MOVWF PORTB			Fetch 2	Execute 2		
3. BRA SUB_1				Fetch 3	Execute 3	
4. BSF PORTA, BIT3 (Forced NOP)					Fetch 4	Flush (NOP)
5. Instruction @ address SUB_1						Fetch SUB_1 Execute SUB_1

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

# PIC18F2525/2620/4525/4620

## 5.2.3 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 5.1.1 "Program Counter"**).

Figure 5-4 shows an example of how instruction words are stored in the program memory.

The **CALL** and **GOTO** instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 5-4 shows how the instruction **GOTO 0006h** is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 24.0 "Instruction Set Summary"** provides further details of the instruction set.

**FIGURE 5-4: INSTRUCTIONS IN PROGRAM MEMORY**

Program Memory Byte Locations →		Word Address ↓	
		LSB = 1	LSB = 0
Instruction 1:	MOVLW 055h		000000h
Instruction 2:	GOTO 0006h		000002h
Instruction 3:	MOVFF 123h, 456h	0Fh	000004h
		EFh	000006h
		F0h	000008h
		C1h	00000Ah
		F4h	00000Ch
			00000Eh
			0000010h
			0000012h
			0000014h

## 5.2.4 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 5-4 shows how this works.

**Note:** See **Section 5.6 "PIC18 Instruction Execution and the Extended Instruction Set"** for information on two-word instructions in the extended instruction set.

## EXAMPLE 5-4: 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	;
0010 0100 0000 0000	Execute this word as a NOP
ADDWF REG3	;
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	;
0010 0100 0000 0000	2nd word of instruction
ADDWF REG3	;
	continue code

## 5.3 Data Memory Organization

**Note:** The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See **Section 5.5 “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; PIC18F2525/2620/4525/4620 devices implement all 16 banks. Figure 5-5 shows the data memory organization for the PIC18F2525/2620/4525/4620 devices.

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

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

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to SFRs and the lower portion of GPR Bank 0 without using the BSR. **Section 5.3.2 “Access Bank”** provides a detailed description of the Access RAM.

### 5.3.1 BANK SELECT REGISTER (BSR)

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

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the four 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 (BSR3:BSR0). 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 5-6.

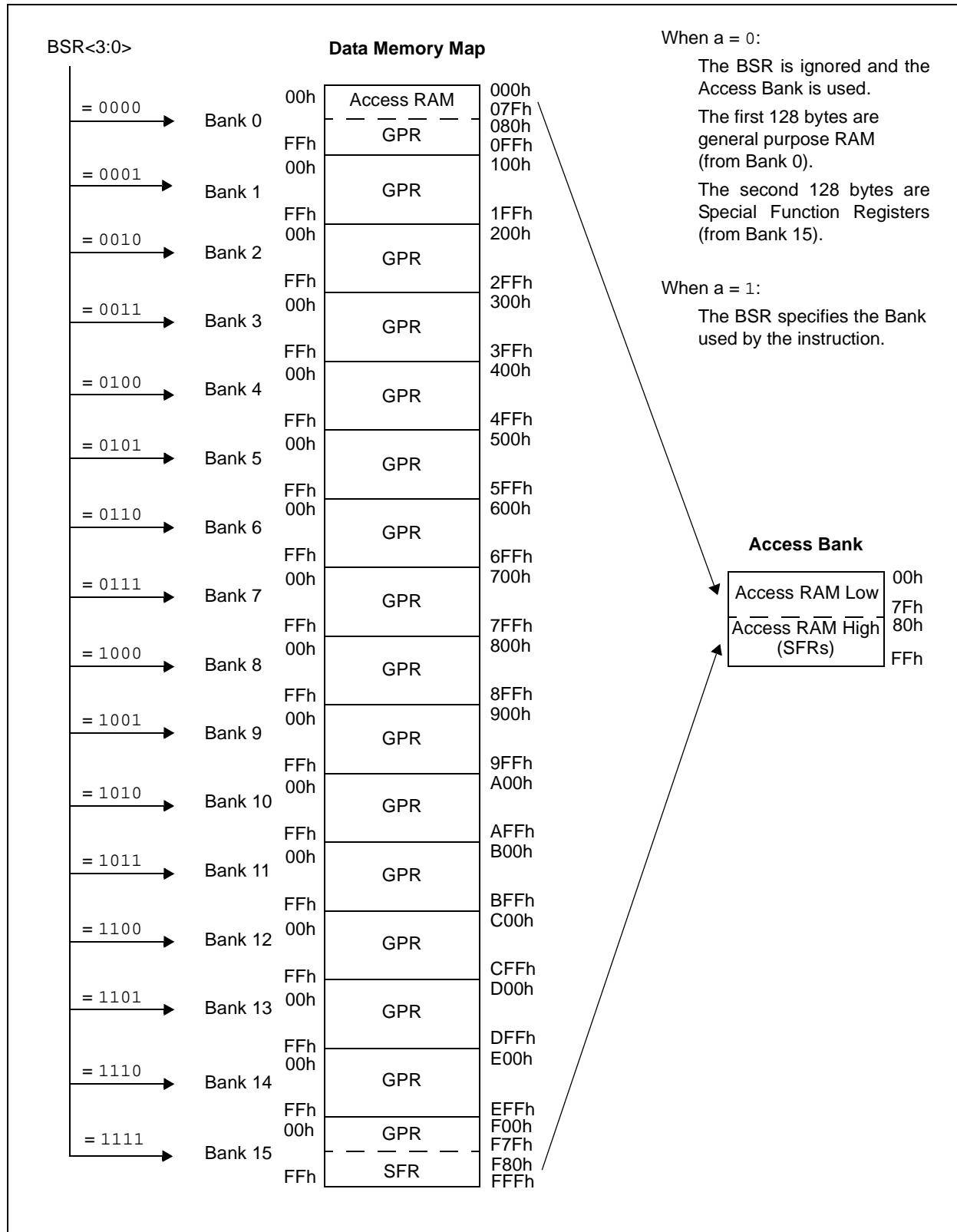
Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h, while the BSR is 0Fh, will end up resetting the 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 5-5 indicates which banks are implemented.

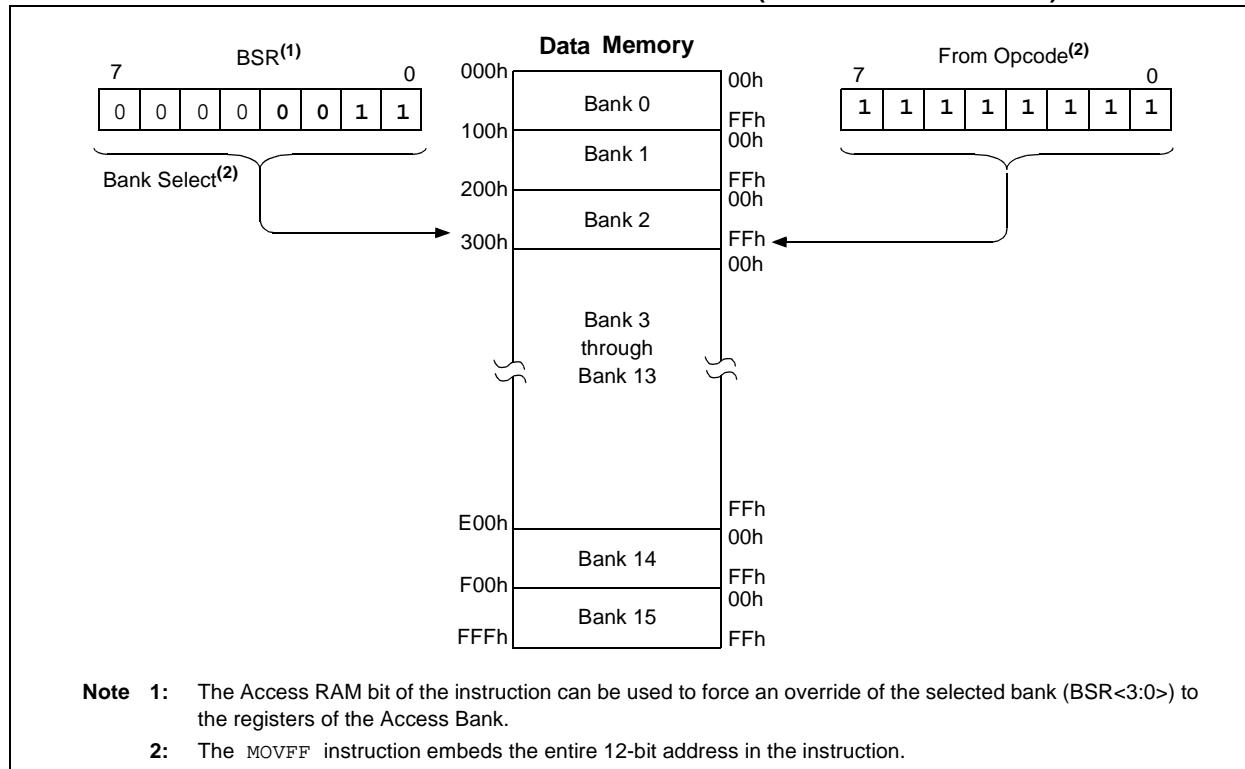
In the core PIC18 instruction set, only the MOVFF instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the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

# PIC18F2525/2620/4525/4620

FIGURE 5-5: DATA MEMORY MAP FOR PIC18F2525/2620/4525/4620 DEVICES



**FIGURE 5-6: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)**



### 5.3.2 ACCESS BANK

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

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 128 bytes of memory (00h-7Fh) in Bank 0 and the last 128 bytes of memory (80h-FFh) in Block 15. The lower half is known as the “Access RAM” and is composed of GPRs. This upper half is also 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 5-5).

The Access Bank is used by core PIC18 instructions that include the Access RAM bit (the ‘a’ parameter in the instruction). When ‘a’ is equal to ‘1’, the instruction uses the BSR and the 8-bit address included in the opcode for the data memory address. When ‘a’ is ‘0’,

however, the instruction is forced to use the Access Bank address map; the current value of the BSR is ignored entirely.

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

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = 1). This is discussed in more detail in **Section 5.5.3 “Mapping the Access Bank in Indexed Literal Offset Addressing Mode”**.

### 5.3.3 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.

# PIC18F2525/2620/4525/4620

## 5.3.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top half of Bank 15 (F80h to FFFh). A list of these registers is given in Table 5-1 and Table 5-2.

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

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

**TABLE 5-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F2525/2620/4525/4620 DEVICES**

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2 <sup>(1)</sup>	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2 <sup>(1)</sup>	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2 <sup>(1)</sup>	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2 <sup>(1)</sup>	FBCh	CCPR2H	F9Ch	__(2)
FFBh	PCLATU	FDBh	PLUSW2 <sup>(1)</sup>	FBBh	CCPR2L	F9Bh	OSCTUNE
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	__(2)
FF9h	PCL	FD9h	FSR2L	FB9h	__(2)	F99h	__(2)
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	__(2)
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	PWM1CON <sup>(3)</sup>	F97h	__(2)
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS <sup>(3)</sup>	F96h	TRISE <sup>(3)</sup>
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD <sup>(3)</sup>
FF4h	PRODH	FD4h	__(2)	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	__(2)
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	__(2)
FEFh	INDF0 <sup>(1)</sup>	FCFh	TMR1H	FAFh	SPBRG	F8Fh	__(2)
FEEh	POSTINC0 <sup>(1)</sup>	FCEh	TMR1L	FAEh	RCREG	F8Eh	__(2)
FEDh	POSTDEC0 <sup>(1)</sup>	FCDh	T1CON	FADh	TXREG	F8Dh	LATE <sup>(3)</sup>
FECh	PREINC0 <sup>(1)</sup>	FCCh	TMR2	FACH	TXSTA	F8Ch	LATD <sup>(3)</sup>
FEBh	PLUSW0 <sup>(1)</sup>	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	EEADRH	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	__(2)
FE7h	INDF1 <sup>(1)</sup>	FC7h	SSPSTAT	FA7h	EECON2 <sup>(1)</sup>	F87h	__(2)
FE6h	POSTINC1 <sup>(1)</sup>	FC6h	SSPCON1	FA6h	EECON1	F86h	__(2)
FE5h	POSTDEC1 <sup>(1)</sup>	FC5h	SSPCON2	FA5h	__(2)	F85h	__(2)
FE4h	PREINC1 <sup>(1)</sup>	FC4h	ADRESH	FA4h	__(2)	F84h	PORTE <sup>(3)</sup>
FE3h	PLUSW1 <sup>(1)</sup>	FC3h	ADRESL	FA3h	__(2)	F83h	PORTD <sup>(3)</sup>
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

**Note 1:** This is not a physical register.

**2:** Unimplemented registers are read as ‘0’.

**3:** This register is not available on 28-pin devices.

# PIC18F2525/2620/4525/4620

**TABLE 5-2: REGISTER FILE SUMMARY (PIC18F2525/2620/4525/4620)**

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	49, 54	
TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	49, 54	
TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	49, 54	
STKPTR	STKFUL <sup>(6)</sup>	STKUNF <sup>(6)</sup>	—	SP4	SP3	SP2	SP1	SP0	00-0 0000	49, 55	
PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000	49, 54	
PCLATH	Holding Register for PC<15:8>								0000 0000	49, 54	
PCL	PC Low Byte (PC<7:0>)								0000 0000	49, 54	
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--00 0000	49, 82	
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	49, 82	
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	49, 82	
TABLAT	Program Memory Table Latch								0000 0000	49, 82	
PRODH	Product Register High Byte								xxxx xxxx	49, 89	
PRODL	Product Register Low Byte								xxxx xxxx	49, 89	
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	49, 111	
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1	49, 112	
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	49, 113	
INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								N/A	49, 68	
POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								N/A	49, 68	
POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								N/A	49, 68	
PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								N/A	49, 68	
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	49, 68	
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte					---- 0000	49, 68
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	49, 68	
WREG	Working Register								xxxx xxxx	49	
INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								N/A	49, 68	
POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								N/A	49, 68	
POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								N/A	49, 68	
PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								N/A	49, 68	
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	49, 68	
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte					---- 0000	50, 68
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	50, 68	
BSR	—	—	—	—	Bank Select Register					---- 0000	50, 59
INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								N/A	50, 68	
POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								N/A	50, 68	
POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								N/A	50, 68	
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								N/A	50, 68	
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	50, 68	
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte					---- 0000	50, 68
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	50, 68	
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	50, 66	

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

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 4.4 "Brown-out Reset (BOR)".**

**2:** These registers and/or bits are not implemented on 28-pin devices and are read as '0'. Reset values are shown for 40/44-pin devices; individual unimplemented bits should be interpreted as '-'.

**3:** The PLLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See **Section 2.6.4 "PLL in INTOSC Modes".**

**4:** The RE3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0); otherwise, RE3 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:** Bit 7 and bit 6 are cleared by user software or by a POR.

# PIC18F2525/2620/4525/4620

TABLE 5-2: REGISTER FILE SUMMARY (PIC18F2525/2620/4525/4620) (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	50, 125
TMR0L	Timer0 Register Low Byte								xxxx xxxx	50, 125
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	50, 123
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0	0100 q000	30, 50
HLVDCON	VDIRMAG	—	IRVST	HLVDEN	HLVLD3	HLVLD2	HLVLD1	HLVLD0	0-00 0101	50, 243
WDTCON	—	—	—	—	—	—	—	SWDTEN	--- ---0	50, 259
RCON	IPEN	SBOREN <sup>(1)</sup>	—	RI	TO	PD	POR	BOR	0q-1 11q0	42, 48, 120
TMR1H	Timer1 Register High Byte								xxxx xxxx	50, 131
TMR1L	Timer1 Register Low Byte								xxxx xxxx	50, 131
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0000 0000	50, 127
TMR2	Timer2 Register								0000 0000	50, 134
PR2	Timer2 Period Register								1111 1111	50, 134
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	50, 133
SSPBUF	MSSP Receive Buffer/Transmit Register								xxxx xxxx	50, 169, 170
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	50, 170
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000	50, 162, 171
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	50, 163, 172
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	50, 173
ADRESH	A/D Result Register High Byte								xxxx xxxx	51, 232
ADRESL	A/D Result Register Low Byte								xxxx xxxx	51, 232
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	--00 0000	51, 223
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0qqq	51, 224
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	51, 225
CCPR1H	Capture/Compare/PWM Register 1 High Byte								xxxx xxxx	51, 140
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx	51, 140
CCP1CON	P1M1 <sup>(2)</sup>	P1M0 <sup>(2)</sup>	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	51, 139, 147
CCPR2H	Capture/Compare/PWM Register 2 High Byte								xxxx xxxx	51, 140
CCPR2L	Capture/Compare/PWM Register 2 Low Byte								xxxx xxxx	51, 140
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	51, 139
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	0100 0-00	51, 204
PWM1CON	PRSEN	PDC6 <sup>(2)</sup>	PDC5 <sup>(2)</sup>	PDC4 <sup>(2)</sup>	PDC3 <sup>(2)</sup>	PDC2 <sup>(2)</sup>	PDC1 <sup>(2)</sup>	PDC0 <sup>(2)</sup>	0000 0000	51, 156
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1 <sup>(2)</sup>	PSSBD0 <sup>(2)</sup>	0000 0000	51, 157
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	51, 239
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	51, 233
TMR3H	Timer3 Register High Byte								xxxx xxxx	51, 137
TMR3L	Timer3 Register Low Byte								xxxx xxxx	51, 137
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	0000 0000	51, 135

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

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 4.4 "Brown-out Reset (BOR)".**

- 2: These registers and/or bits are not implemented on 28-pin devices and are read as '0'. Reset values are shown for 40/44-pin devices; individual unimplemented bits should be interpreted as '-'.
- 3: The PLLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See **Section 2.6.4 "PLL in INTOSC Modes".**
- 4: The RE3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0); otherwise, RE3 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: Bit 7 and bit 6 are cleared by user software or by a POR.

# PIC18F2525/2620/4525/4620

TABLE 5-2: REGISTER FILE SUMMARY (PIC18F2525/2620/4525/4620) (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:	
SPBRGH	EUSART Baud Rate Generator Register High Byte								0000 0000	51, 206	
SPBRG	EUSART Baud Rate Generator Register Low Byte								0000 0000	51, 206	
RCREG	EUSART Receive Register								0000 0000	51, 213	
TXREG	EUSART Transmit Register								0000 0000	51, 211	
TXSTA	CSRC	TX9	TXEN	SYNC	SEDB	BRGH	TRMT	TX9D	0000 0010	51, 202	
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	51, 203	
EEADRH	—	—	—	—	—	—	—	EEPROM Addr Register High	---- --00	51, 73	
EEADR	EEPROM Address Register								0000 0000	51, 80, 73	
EEDATA	EEPROM Data Register								0000 0000	51, 80, 73	
EECON2	EEPROM Control Register 2 (not a physical register)								0000 0000	51, 80, 73	
EECON1	EEPGD	CFG5	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	51, 81, 74	
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	11-1 1111	52, 119	
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	00-0 0000	52, 115	
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	00-0 0000	52, 117	
IPR1	PSPIP <sup>(2)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	52, 118	
PIR1	PSPIF <sup>(2)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	52, 114	
PIE1	PSPIE <sup>(2)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	52, 116	
OSCTUNE	INTSRC	PLLEN <sup>(3)</sup>	—	TUN4	TUN3	TUN2	TUN1	TUN0	00-0 0000	27, 52	
TRISE <sup>(2)</sup>	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	0000 -111	52, 104	
TRISD <sup>(2)</sup>	PORTD Data Direction Control Register								1111 1111	52, 100	
TRISC	PORTC Data Direction Control Register								1111 1111	52, 97	
TRISB	PORTB Data Direction Control Register								1111 1111	52, 94	
TRISA	TRISA7 <sup>(5)</sup>	TRISA6 <sup>(5)</sup>	Data Direction Control Register for PORTA						1111 1111	52, 91	
LATE <sup>(2)</sup>	—	—	—	—	—	PORTE Data Latch Register (Read and Write to Data Latch)			---- -xxx	52, 103	
LATD <sup>(2)</sup>	PORTD Data Latch Register (Read and Write to Data Latch)						xxxx xxxx				52, 100
LATC	PORTC Data Latch Register (Read and Write to Data Latch)						xxxx xxxx				52, 97
LATB	PORTB Data Latch Register (Read and Write to Data Latch)						xxxx xxxx				52, 94
LATA	LATA7 <sup>(5)</sup>	LATA6 <sup>(5)</sup>	PORTA Data Latch Register (Read and Write to Data Latch)						xxxx xxxx	52, 91	
PORTE	—	—	—	—	RE3 <sup>(4)</sup>	RE2 <sup>(2)</sup>	RE1 <sup>(2)</sup>	RE0 <sup>(2)</sup>	---- xxxx	52, 103	
PORTD <sup>(2)</sup>	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	52, 100	
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	52, 97	
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	52, 94	
PORTA	RA7 <sup>(5)</sup>	RA6 <sup>(5)</sup>	RA5	RA4	RA3	RA2	RA1	RA0	xx0x 0000	52, 91	

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

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 4.4 "Brown-out Reset (BOR)".**

- 2: These registers and/or bits are not implemented on 28-pin devices and are read as '0'. Reset values are shown for 40/44-pin devices; individual unimplemented bits should be interpreted as '-'.
- 3: The PLLEN bit is only available in specific oscillator configurations; otherwise, it is disabled and reads as '0'. See **Section 2.6.4 "PLL in INTOSC Modes".**
- 4: The RE3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0); otherwise, RE3 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: Bit 7 and bit 6 are cleared by user software or by a POR.

# PIC18F2525/2620/4525/4620

## 5.3.5 STATUS REGISTER

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

If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, the results of the instruction are not written; instead, the STATUS register 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 uuu').

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 24-2 and Table 24-3.

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

## REGISTER 5-2: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC <sup>(1)</sup>	C <sup>(2)</sup>
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 7-5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>N:</b> 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	<b>OV:</b> 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	<b>Z:</b> 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	<b>DC:</b> 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	<b>C:</b> 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.

## 5.4 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 5.5 “Data Memory and the Extended Instruction Set”** for more information.

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 5.5.1 “Indexed Addressing with Literal Offset”**.

### 5.4.1 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.

### 5.4.2 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 5.3.3 “General Purpose Register File”**) or a location in the Access Bank (**Section 5.3.2 “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 5.3.1 “Bank Select Register (BSR)”**) are used with the address to determine the complete 12-bit address of the register. When ‘a’ is ‘0’, the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

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

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

### 5.4.3 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 Function 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 5-5.

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

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

### 5.4.3.1 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.

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

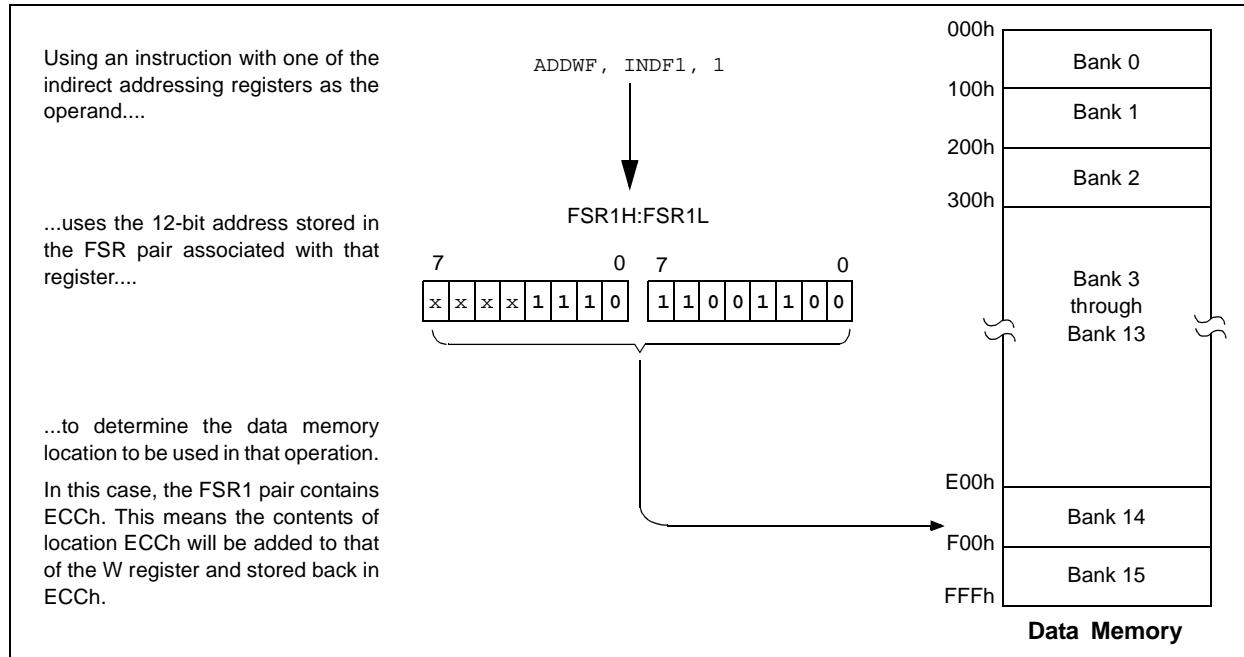
In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers that cannot be indirectly read or written to. Accessing these registers actually accesses the associated FSR register pair, but also performs a specific action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

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

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

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

**FIGURE 5-7: INDIRECT ADDRESSING**



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.

#### 5.4.3.3 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.

## 5.5 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 remains unchanged.

### 5.5.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair within Access RAM. 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.

### 5.5.2 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 5-8.

Those who desire to use bit-oriented or byte-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 24.2.1 “Extended Instruction Syntax”**.

# PIC18F2525/2620/4525/4620

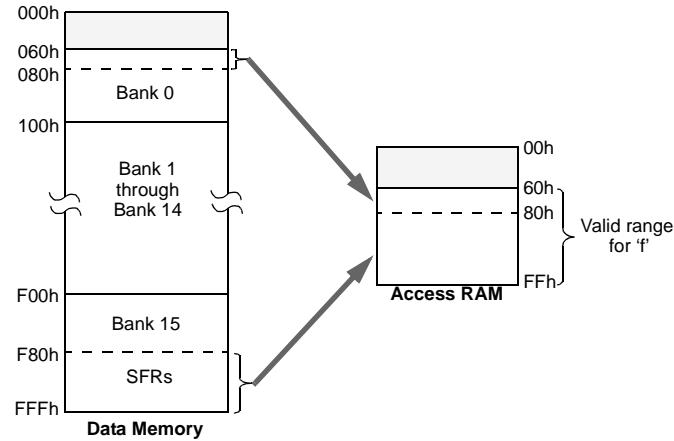
**FIGURE 5-8: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

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

**When 'a' = 0 and f  $\geq$  60h:**

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations 060h to 07Fh (Bank 0) and F80h to FFFh (Bank 15) of data memory.

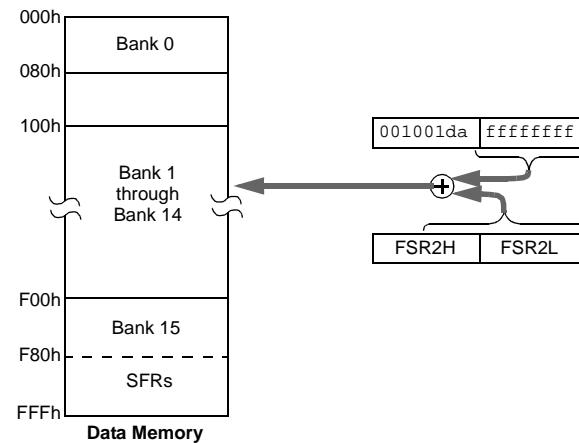
Locations below 60h are not available in this addressing mode.



**When 'a' = 0 and f  $\leq$  5Fh:**

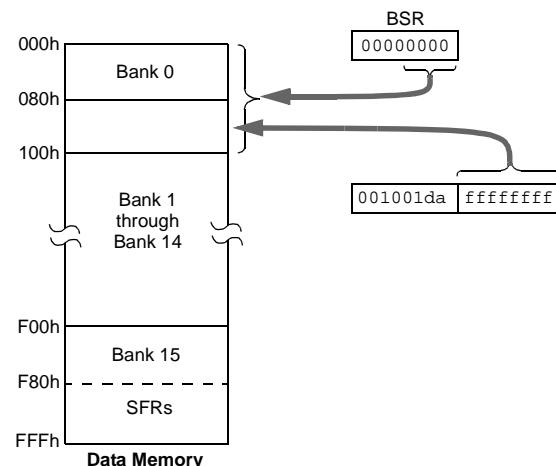
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

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.



### 5.5.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET ADDRESSING MODE

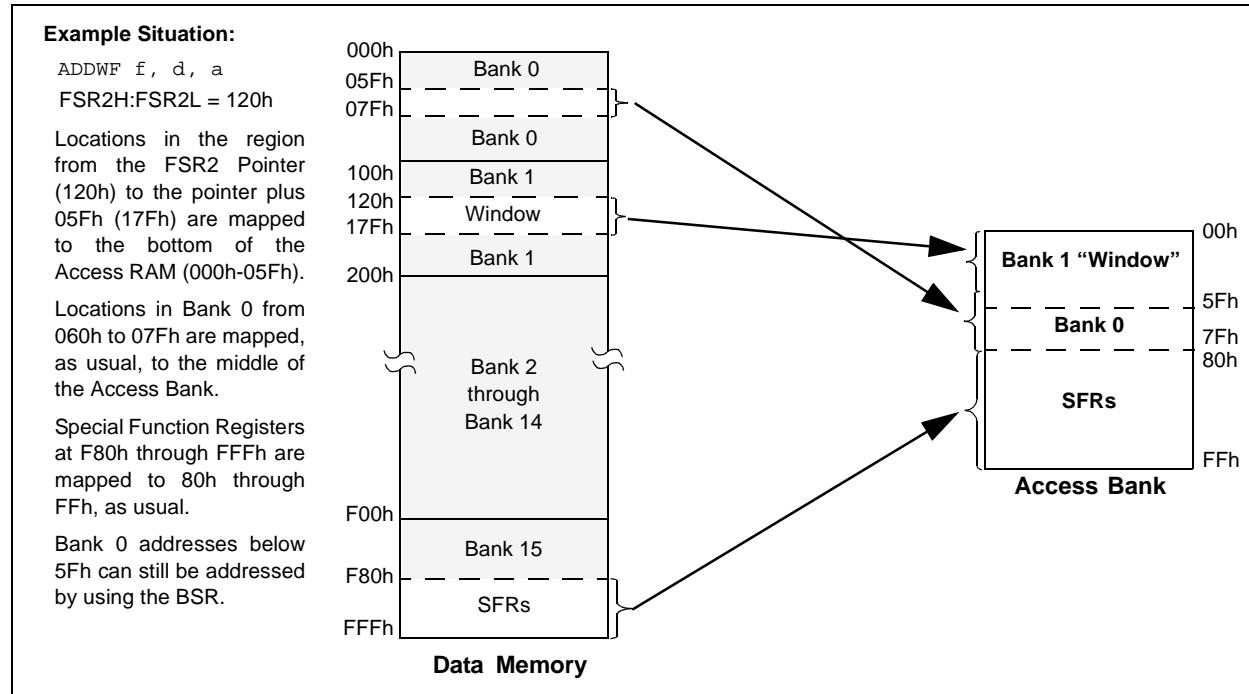
The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the bottom half of Bank 0, this mode maps the contents from Bank 0 and a user-defined "window" that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see **Section 5.3.2 "Access Bank"**). An example of Access Bank remapping in this addressing mode is shown in Figure 5-9.

Remapping of the Access Bank applies *only* to operations using the Indexed Literal Offset Addressing mode. Operations that use the BSR (Access RAM bit is '1') will continue to use Direct Addressing as before.

### 5.6 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 24.2 "Extended Instruction Set"**.

**FIGURE 5-9: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING MODE**



# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 6.0 DATA EEPROM MEMORY

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

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

- EECON1
- EECON2
- EEDATA
- EEADR
- EEADRH

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

The EEPROM data memory is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer; it will vary with voltage and temperature as well as from chip to chip. Please refer to parameter D122 (Table 26-1 in **Section 26.0 “Electrical Characteristics”**) for exact limits.

### 6.1 EEADR and EEADRH Registers

The EEADRH:EEADR register pair is used to address the data EEPROM for read and write operations. EEADRH holds the two MSbits of the address; the upper 6 bits are ignored. The 10-bit range of the pair can address a memory range of 1024 bytes (00h to 3FFh).

### 6.2 EECON1 and EECON2 Registers

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

The EECON1 register (Register 6-1) is the control register for data and program memory access. Control bit EEPGD determines if the access will be to program or data EEPROM memory. When clear, operations will access the data EEPROM memory. When set, program memory is accessed.

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

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

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

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

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

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

The RD bit cannot be set when accessing program memory (EEPGD = 1). Program memory is read using table read instructions. See **Section 7.1 “Table Reads and Table Writes”** regarding table reads.

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

# PIC18F2525/2620/4525/4620

---

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

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGs	—	FREE	WRERR <sup>(1)</sup>	WREN	WR	RD
bit 7	bit 0						

**Legend:**

S = Set only bit (cannot be cleared in software)

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7      **EEPGD:** Flash Program or Data EEPROM Memory Select bit  
1 = Access Flash program memory  
0 = Access data EEPROM memory
- bit 6      **CFGs:** Flash Program/Data EEPROM or Configuration Select bit  
1 = Access Configuration registers  
0 = Access Flash program or data EEPROM memory
- bit 5      **Unimplemented:** Read as '0'
- bit 4      **FREE:** Flash Row Erase Enable bit  
1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation)  
0 = Perform write only
- bit 3      **WRERR:** Flash Program/Data EEPROM Error Flag bit<sup>(1)</sup>  
1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation, or an improper write attempt)  
0 = The write operation completed
- bit 2      **WREN:** Flash Program/Data EEPROM Write Enable bit  
1 = Allows write cycles to Flash program/data EEPROM  
0 = Inhibits write cycles to Flash program/data EEPROM
- bit 1      **WR:** Write Control bit  
1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle  
(The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)  
0 = Write cycle to the EEPROM is complete
- bit 0      **RD:** Read Control bit  
1 = Initiates an EEPROM read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1 or CFGs = 1.)  
0 = Does not initiate an EEPROM read

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

## 6.3 Reading the Data EEPROM Memory

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

The basic process is shown in Example 6-1.

## 6.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADRH:EEADR register pair and the data written to the EEDATA register. The sequence in Example 6-2 must be followed to initiate the write cycle.

The write will not begin if this sequence is not exactly followed (write 55h to EECON2, write 0AAh to EECON2, then set WR bit) for each byte. It is strongly recommended that interrupts be disabled during this code segment.

### EXAMPLE 6-1: DATA EEPROM READ

```

MOVLW  DATA_EE_ADDRH      ;
MOVWF  EEADRH             ; Upper bits of Data Memory Address to read
MOVLW  DATA_EE_ADDR      ;
MOVWF  EEADR              ; Lower bits of Data Memory Address to read
BCF    EECON1, EEPGD       ; Point to DATA memory
BCF    EECON1, CFGS        ; Access EEPROM
BSF    EECON1, RD          ; EEPROM Read
MOVF   EEDATA, W           ; W = EEDATA

```

### EXAMPLE 6-2: DATA EEPROM WRITE

	MOVLW  DATA_EE_ADDRH      ;	
	MOVWF  EEADRH             ; Upper bits of Data Memory Address to write	
	MOVLW  DATA_EE_ADDR      ;	
	MOVWF  EEADR              ; Lower bits of Data Memory Address to write	
	MOVLW  DATA_EE_DATA      ;	
	MOVWF  EEDATA             ; Data Memory Value to write	
	BCF    EECON1, EPGD        ; Point to DATA memory	
	BCF    EECON1, CFGS        ; Access EEPROM	
	BSF    EECON1, WREN        ; Enable writes	
	BCF    INTCON, GIE         ; Disable Interrupts	
Required Sequence	MOVLW  55h                ;	
	MOVWF  EECON2              ; Write 55h	
	MOVLW  0AAh                ;	
	MOVWF  EECON2              ; Write 0AAh	
	BSF    EECON1, WR           ; Set WR bit to begin write	
BSF    INTCON, GIE          ; Enable Interrupts		
	;	
	BCF    EECON1, WREN        ; User code execution	
	BCF    EECON1, WREN        ; Disable writes on write complete (EEIF set)	

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

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

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

## 6.5 Write Verify

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

## 6.6 Operation During Code-Protect

Data EEPROM memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if code protection is enabled.

The microcontroller itself can both read and write to the internal data EEPROM, regardless of the state of the code-protect Configuration bit. Refer to **Section 23.0 “Special Features of the CPU”** for additional information.

## 6.7 Protection Against Spurious Write

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been implemented. On power-up, the WREN bit is cleared. In addition, writes to the EEPROM are blocked during the Power-up Timer period (TPWRT, parameter 33).

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

## 6.8 Using the Data EEPROM

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

A simple data EEPROM refresh routine is shown in Example 6-3.

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

### EXAMPLE 6-3: DATA EEPROM REFRESH ROUTINE

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

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

# PIC18F2525/2620/4525/4620

TABLE 6-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page		
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RBIE	TMROIF	INT0IF	RBIF	49		
EEADRH	—	—	—	—	—	—	EEPROM Address Register High Byte		51		
EEADR	EEPROM Address Register								51		
EEDATA	EEPROM Data Register								51		
EECON2	EEPROM Control Register 2 (not a physical register)								51		
EECON1	EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD	51		
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52		
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52		
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52		

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

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 7.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire V<sub>DD</sub> range.

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

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

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

## 7.1 Table Reads and Table Writes

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

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

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

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

Table write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 7.5 “Writing to Flash Program Memory”**. Figure 7-2 shows the operation of a table write with program memory and data RAM.

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

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

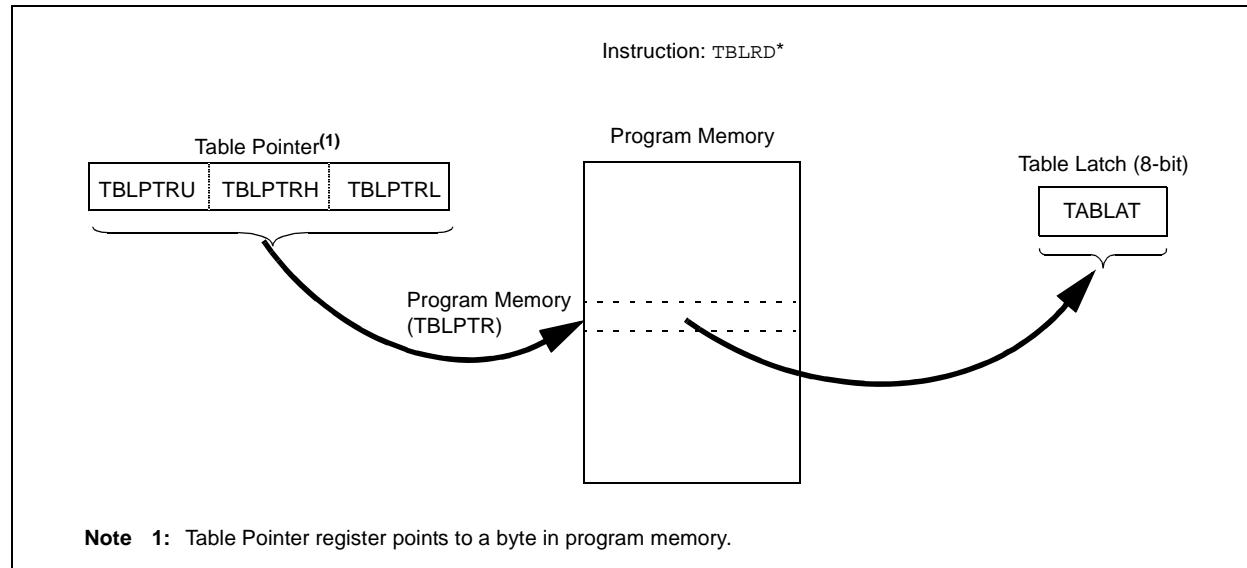
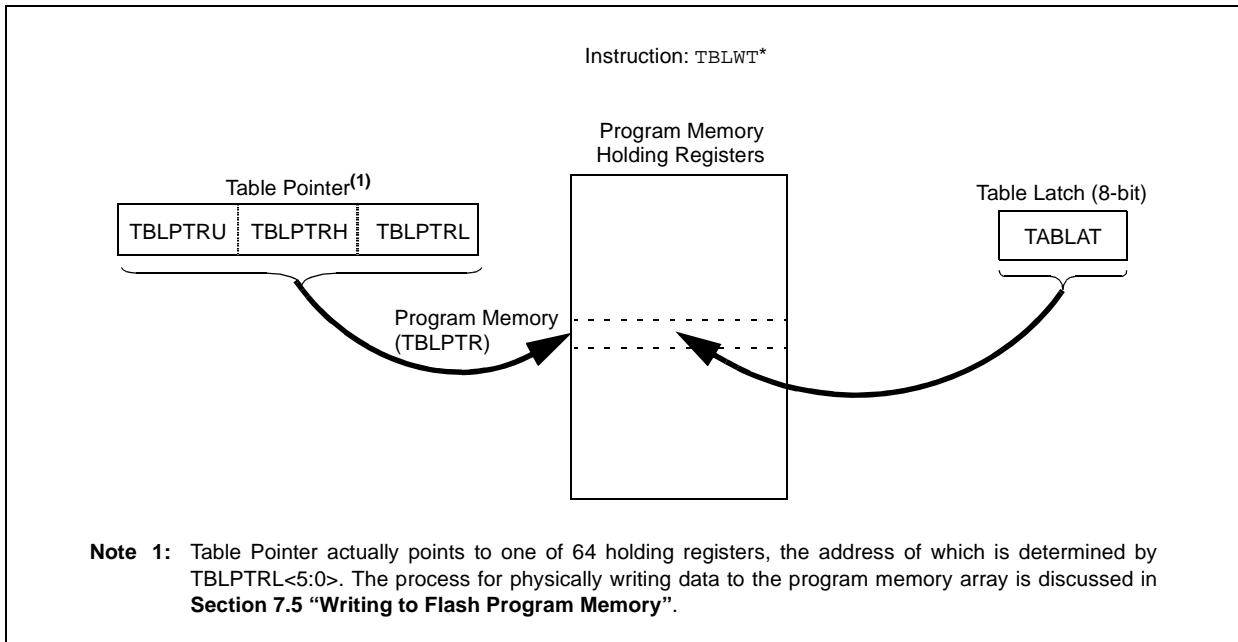


FIGURE 7-2: TABLE WRITE OPERATION



## 7.2 Control Registers

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

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

### 7.2.1 EECON1 AND EECON2 REGISTERS

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

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

The CFGS control bit determines if the access will be to the Configuration/Calibration registers or to program memory/data EEPROM memory. When set, subsequent operations will operate on Configuration registers regardless of EEPGD (see **Section 23.0 "Special Features of the CPU"**). When clear, memory selection access is determined by EEPGD.

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

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

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

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

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

# PIC18F2525/2620/4525/4620

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

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGs	—	FREE	WRERR <sup>(1)</sup>	WREN	WR	RD
bit 7	bit 0						

**Legend:**

S = Set only bit (cannot be cleared in software)

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>EEPGD:</b> Flash Program or Data EEPROM Memory Select bit 1 = Access Flash program memory 0 = Access data EEPROM memory
bit 6	<b>CFGs:</b> Flash Program/Data EEPROM or Configuration Select bit 1 = Access Configuration registers 0 = Access Flash program or data EEPROM memory
bit 5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>FREE:</b> Flash Row Erase Enable bit 1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation) 0 = Perform write only
bit 3	<b>WRERR:</b> Flash Program/Data EEPROM Error Flag bit <sup>(1)</sup> 1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation, or an improper write attempt) 0 = The write operation completed
bit 2	<b>WREN:</b> Flash Program/Data EEPROM Write Enable bit 1 = Allows write cycles to Flash program/data EEPROM 0 = Inhibits write cycles to Flash program/data EEPROM
bit 1	<b>WR:</b> Write Control bit 1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.) 0 = Write cycle to the EEPROM is complete
bit 0	<b>RD:</b> Read Control bit 1 = Initiates an EEPROM read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1 or CFGs = 1.) 0 = Does not initiate an EEPROM read

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

# PIC18F2525/2620/4525/4620

## 7.2.2 TABLAT – TABLE LATCH REGISTER

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

## 7.2.3 TBLPTR – TABLE POINTER REGISTER

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

The Table Pointer register, TBLPTR, is used by the TBLRD and TBLWT instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations are shown in Table 7-1. These operations on the TBLPTR only affect the low-order 21 bits.

## 7.2.4 TABLE POINTER BOUNDARIES

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

When a TBLRD is executed, all 22 bits of the TBLPTR determine which byte is read from program memory into TABLAT.

When a TBLWT is executed, the six LSbs of the Table Pointer register (TBLPTR<5:0>) determine which of the 64 program memory holding registers is written to. When the timed write to program memory begins (via the WR bit), the 16 MSbs of the TBLPTR (TBLPTR<21:6>) determine which program memory block of 64 bytes is written to. For more detail, see **Section 7.5 “Writing to Flash Program Memory”**.

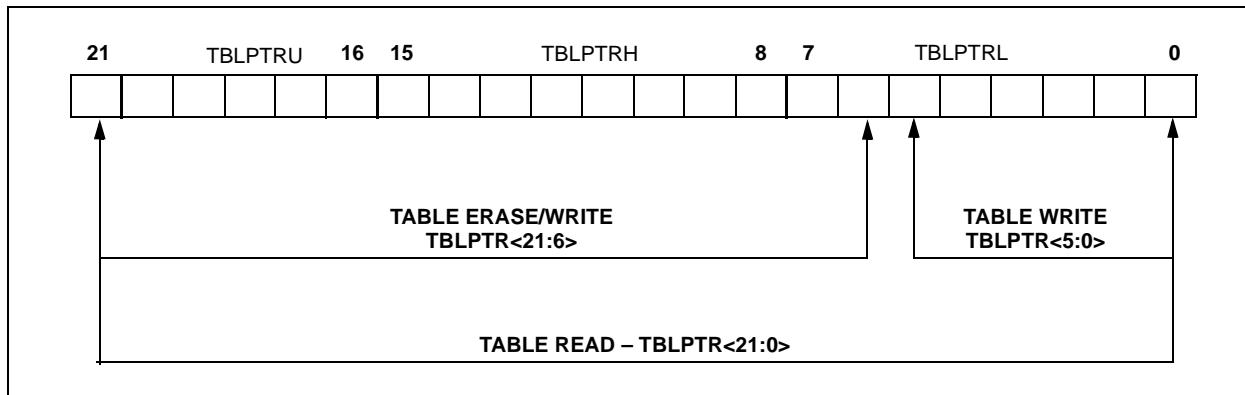
When an erase of program memory is executed, the 16 MSbs of the Table Pointer register (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

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

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

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

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



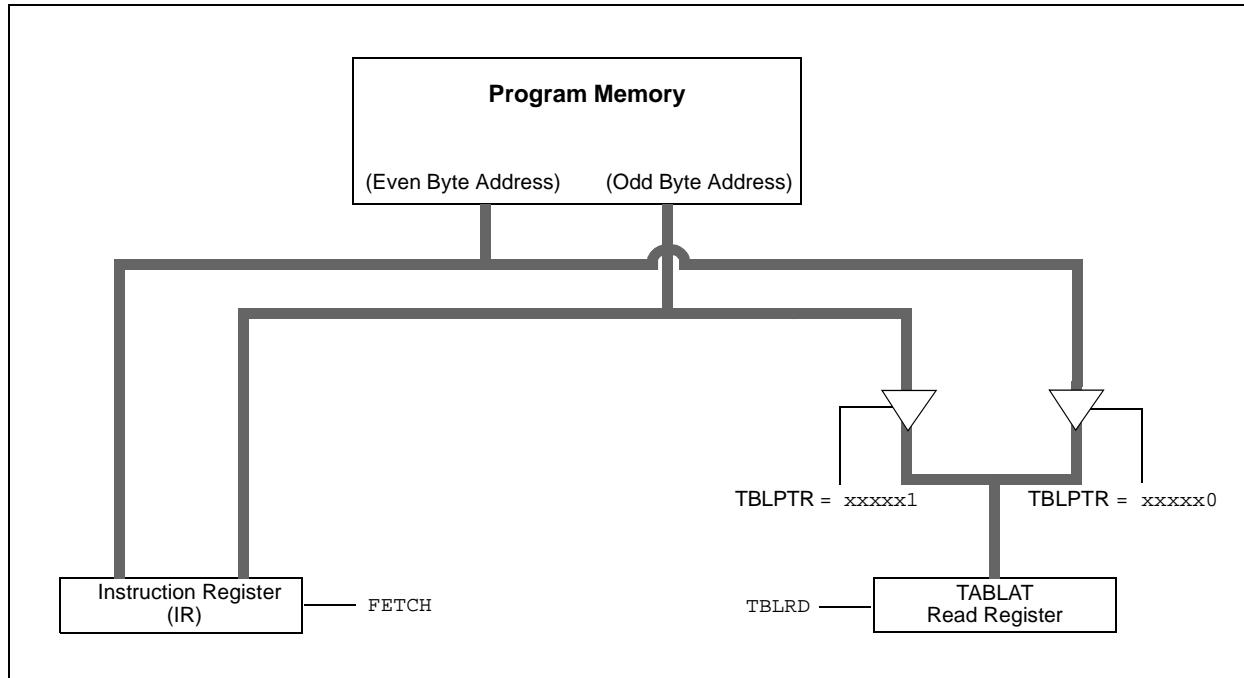
## 7.3 Reading the Flash Program Memory

The TBLRD instruction is used to retrieve data from program memory 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. In addition, TBLPTR can be modified automatically for the next table read operation.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word. Figure 7-4 shows the interface between the internal program memory and the TABLAT.

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



## EXAMPLE 7-1: READING A FLASH PROGRAM MEMORY WORD

```

MOVlw  CODE_ADDR_UPPER      ; Load TBLPTR with the base
MOVwf  TBLPTRU
MOVlw  CODE_ADDR_HIGH
MOVwf  TBLPTRH
MOVlw  CODE_ADDR_LOW
MOVwf  TBLPTRL

READ_WORD
TBLRD*+
MOVf   TABLAT, W           ; read into TABLAT and increment
MOVwf  WORD_EVEN           ; get data
TBLRD*+
MOVf   TABLAT, W           ; read into TABLAT and increment
MOVwf  WORD_ODD           ; get data

```

## 7.4 Erasing Flash Program Memory

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

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

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

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

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

### 7.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

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

1. Load Table Pointer register with address of row being erased.
2. Set the EECON1 register for the erase operation:
  - set EEPGD bit to point to program memory;
  - clear the CFGS bit to access program memory;
  - set WREN bit to enable writes;
  - set FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit. This will begin the row erase cycle.
7. The CPU will stall for duration of the erase (about 2 ms using internal timer).
8. Re-enable interrupts.

### EXAMPLE 7-2: ERASING A FLASH PROGRAM MEMORY ROW

	MOVLW	CODE_ADDR_UPPER	<i>; load TBLPTR with the base address of the memory block</i>
	MOVWF	TBLPTRU	
	MOVLW	CODE_ADDR_HIGH	
	MOVWF	TBLPTRH	
	MOVLW	CODE_ADDR_LOW	
	MOVWF	TBLPTRL	
ERASE_ROW	BSF	EECON1, EEPGD	<i>; point to Flash program memory</i>
	BCF	EECON1, CFGS	<i>; access Flash program memory</i>
	BSF	EECON1, WREN	<i>; enable write to memory</i>
	BSF	EECON1, FREE	<i>; enable Row Erase operation</i>
	BCF	INTCON, GIE	<i>; disable interrupts</i>
Required Sequence	MOVLW	55h	<i>; write 55h</i>
	MOVWF	EECON2	
	MOVLW	0AAh	
	MOVWF	EECON2	<i>; write 0AAh</i>
	BSF	EECON1, WR	<i>; start erase (CPU stall)</i>
	BSF	INTCON, GIE	<i>; re-enable interrupts</i>

## 7.5 Writing to Flash Program Memory

The minimum programming block is 32 words or 64 bytes. Word or byte programming is not supported.

Table writes are used internally to load the holding registers needed to program the Flash memory. There are 64 holding registers used by the table writes for programming.

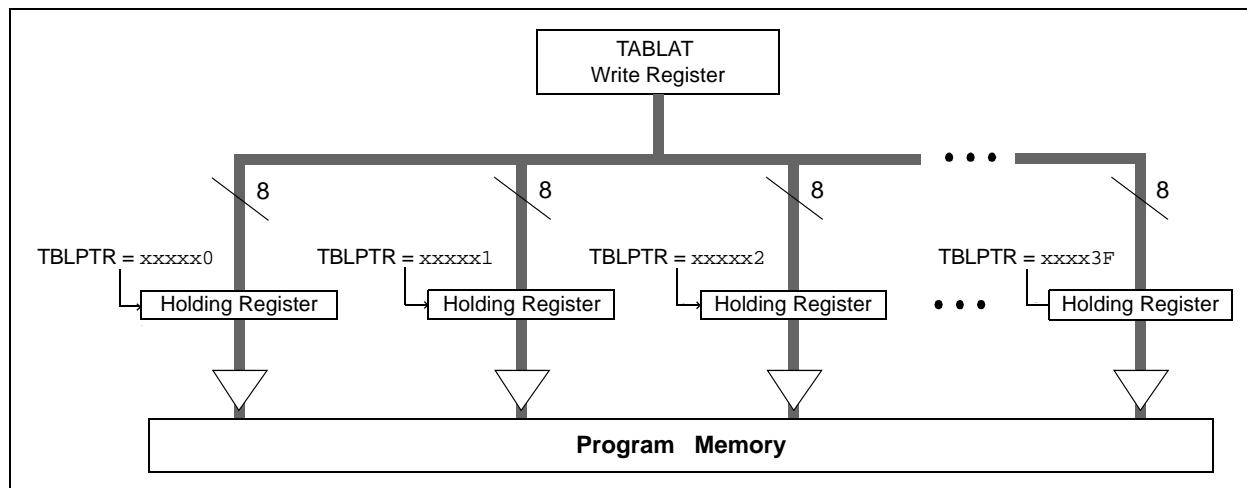
Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction may need to be executed 64 times for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. At the end of updating the 64 holding registers, the EECON1 register must be written to in order to start the programming operation with a long write.

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

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

**Note:** The default value of the holding registers on device Resets and after write operations is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all 64 holding registers before executing a write operation.

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



### 7.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

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

1. Read 64 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load Table Pointer register with address being erased.
4. Execute the row erase procedure.
5. Load Table Pointer register with address of first byte being written.
6. Write the 64 bytes into the holding registers with auto-increment.
7. Set the EECON1 register for the write operation:
  - set EEPGD bit to point to program memory;
  - clear the CFGS bit to access program memory;
  - set WREN to enable byte writes.

8. Disable interrupts.
9. Write 55h to EECON2.
10. Write 0AAh to EECON2.
11. Set the WR bit. This will begin the write cycle.
12. The CPU will stall for duration of the write (about 2 ms using internal timer).
13. Re-enable interrupts.
14. Verify the memory (table read).

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

**Note:** Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the 64 bytes in the holding register.

# PIC18F2525/2620/4525/4620

## EXAMPLE 7-3: WRITING TO FLASH PROGRAM MEMORY

	MOVLW	D'64	; number of bytes in erase block
	MOVWF	COUNTER	
	MOVLW	BUFFER_ADDR_HIGH	; point to buffer
	MOVWF	FSR0H	
	MOVLW	BUFFER_ADDR_LOW	
	MOVWF	FSR0L	
	MOVLW	CODE_ADDR_UPPER	; Load TBLPTR with the base
	MOVWF	TBLPTRU	; address of the memory block
	MOVLW	CODE_ADDR_HIGH	
	MOVWF	TBLPTRH	
	MOVLW	CODE_ADDR_LOW	
	MOVWF	TBLPTRL	
READ_BLOCK	TBLRD*+		
	MOVF	TABLAT, W	; read into TABLAT, and inc
	MOVWF	POSTINCO	; get data
	DECFSZ	COUNTER	; store data
	BRA	READ_BLOCK	; done?
			; repeat
MODIFY_WORD	MOVLW	DATA_ADDR_HIGH	; point to buffer
	MOVWF	FSR0H	
	MOVLW	DATA_ADDR_LOW	
	MOVWF	FSR0L	
	MOVLW	NEW_DATA_LOW	; update buffer word
	MOVWF	POSTINCO	
	MOVLW	NEW_DATA_HIGH	
	MOVWF	INDFO	
ERASE_BLOCK	MOVLW	CODE_ADDR_UPPER	; load TBLPTR with the base
	MOVWF	TBLPTRU	; address of the memory block
	MOVLW	CODE_ADDR_HIGH	
	MOVWF	TBLPTRH	
	MOVLW	CODE_ADDR_LOW	
	MOVWF	TBLPTRL	
	BSF	EECON1, EEPGD	; point to Flash program memory
	BCF	EECON1, CFGS	; access Flash program memory
	BSF	EECON1, WREN	; enable write to memory
	BSF	EECON1, FREE	; enable Row Erase operation
	BCF	INTCON, GIE	; disable interrupts
Required Sequence	MOVLW	55h	
	MOVWF	EECON2	; write 55h
	MOVLW	0AAh	
	MOVWF	EECON2	; write 0AAh
	BSF	EECON1, WR	; start erase (CPU stall)
	BSF	INTCON, GIE	; re-enable interrupts
TBLRD*-			; dummy read decrement
	MOVLW	BUFFER_ADDR_HIGH	; point to buffer
	MOVWF	FSR0H	
	MOVLW	BUFFER_ADDR_LOW	
	MOVWF	FSR0L	
WRITE_BUFFER_BACK	MOVLW	D'64	; number of bytes in holding register
	MOVWF	COUNTER	
WRITE_BYTE_TO_HREGS	MOVFF	POSTINCO, WREG	; get low byte of buffer data
	MOVWF	TABLAT	; present data to table latch
	TBLWT+*		; write data, perform a short write
			; to internal TBLWT holding register.
	DECFSZ	COUNTER	; loop until buffers are full
	BRA	WRITE_WORD_TO_HREGS	

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

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

#### 7.5.2 WRITE VERIFY

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

#### 7.5.3 UNEXPECTED TERMINATION OF WRITE OPERATION

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

#### 7.5.4 PROTECTION AGAINST SPURIOUS WRITES

To protect against spurious writes to Flash program memory, the write initiate sequence must also be followed. See **Section 23.0 “Special Features of the CPU”** for more detail.

#### 7.6 Flash Program Operation During Code Protection

See **Section 23.5 “Program Verification and Code Protection”** for details on code protection of Flash program memory.

TABLE 7-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page					
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					49					
TBPLTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								49					
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								49					
TABLAT	Program Memory Table Latch								49					
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49					
EECON2	EEPROM Control Register 2 (not a physical register)								51					
EECON1	EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD	51					
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52					
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52					
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52					

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

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 8.0 8 x 8 HARDWARE MULTIPLIER

### 8.1 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 8-1.

### 8.2 Operation

Example 8-1 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 8-2 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 8-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

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

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

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

**TABLE 8-1: 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 $\mu$ s	27.6 $\mu$ s	69 $\mu$ s
	Hardware multiply	1	1	100 ns	400 ns	1 $\mu$ s
8 x 8 signed	Without hardware multiply	33	91	9.1 $\mu$ s	36.4 $\mu$ s	91 $\mu$ s
	Hardware multiply	6	6	600 ns	2.4 $\mu$ s	6 $\mu$ s
16 x 16 unsigned	Without hardware multiply	21	242	24.2 $\mu$ s	96.8 $\mu$ s	242 $\mu$ s
	Hardware multiply	28	28	2.8 $\mu$ s	11.2 $\mu$ s	28 $\mu$ s
16 x 16 signed	Without hardware multiply	52	254	25.4 $\mu$ s	102.6 $\mu$ s	254 $\mu$ s
	Hardware multiply	35	40	4.0 $\mu$ s	16.0 $\mu$ s	40 $\mu$ s

# PIC18F2525/2620/4525/4620

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

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

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

## EXAMPLE 8-3: 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 ARG2H      ; ARG1H * ARG2H->
                  ; PRODH:PRODL
MOVF PRODL, W    ;
ADDWF RES1, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;
;
```

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

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

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

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

```

MOVF ARG1L, W
MULWF ARG2L      ; ARG1L * ARG2L ->
                  ; PRODH:PRODL
MOVFF PRODH, RES1 ;
MOVFF PRODL, RES0 ;
;

MOVF ARG1H, W
MULWF ARG2H      ; ARG1H * ARG2H ->
                  ; PRODH:PRODL
MOVFF PRODH, RES3 ;
MOVFF PRODL, RES2 ;
;

MOVF ARG1L, W
MULWF ARG2H      ; ARG1L * ARG2H ->
                  ; PRODH:PRODL
MOVF PRODL, W    ;
ADDWF RES1, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;
;

MOVF ARG1H, W
MULWF ARG2H      ; ARG1H * ARG2H ->
                  ; PRODH:PRODL
MOVF PRODL, W    ;
ADDWF RES1, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;
;

MOVF PRODL, W    ;
ADDWF RES1, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;
;

BTFS ARG2H, 7    ; ARG2H:ARG2L neg?
BRA SIGN_ARG1    ; no, check ARG1
MOVF ARG1L, W    ;
SUBWF RES2        ;
MOVF ARG1H, W    ;
SUBWFB RES3        ;
;

SIGN_ARG1
BTFS ARG1H, 7    ; ARG1H:ARG1L neg?
BRA CONT_CODE    ; no, done
MOVF ARG2L, W    ;
SUBWF RES2        ;
MOVF ARG2H, W    ;
SUBWFB RES3        ;
;

CONT_CODE
:
```

## 9.0 I/O PORTS

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

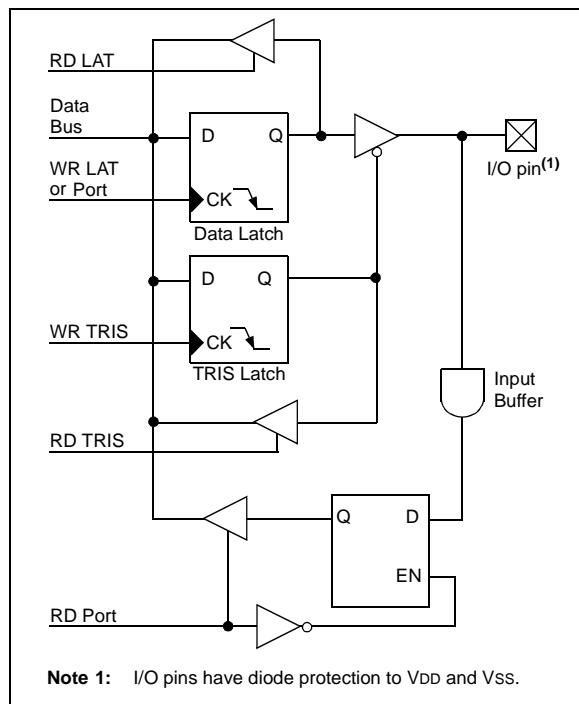
Each port has three registers for its operation. These registers are:

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

The Data Latch (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 9-1.

**FIGURE 9-1: GENERIC I/O PORT OPERATION**



## 9.1 PORTA, TRISA and LATA Registers

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

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

The Data Latch (LATA) register 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 and one of the comparator outputs to become the RA4/T0CKI/C1OUT 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 23.1 “Configuration Bits”** for details). When they are not used as port pins, RA6 and RA7 and their associated TRIS and LAT bits are read as ‘0’.

The other PORTA pins are multiplexed with analog inputs, the analog VREF+ and VREF- inputs and the comparator voltage reference output. The operation of pins RA3:RA0 and RA5 as A/D converter inputs is selected by clearing or setting the control bits in the ADCON1 register (A/D Control Register 1).

Pins RA0 through RA5 may also be used as comparator inputs or outputs by setting the appropriate bits in the CMCON register. To use RA3:RA0 as digital inputs, it is also necessary to turn off the comparators.

**Note:** On a Power-on Reset, RA5 and RA3:RA0 are configured as analog inputs and read as ‘0’. RA4 is configured as a digital input.

The RA4/T0CKI/C1OUT pin is a Schmitt Trigger input. All other PORTA pins have TTL input levels. All PORTA pins have 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 9-1: 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<7:6,3:0> as inputs
              ; RA<5:4> as outputs

```

# PIC18F2525/2620/4525/4620

TABLE 9-1: PORTA I/O SUMMARY

Pin	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 and Comparator C1- input. 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 and Comparator C2- input. Default input configuration on POR; does not affect digital output.
RA2/AN2/ VREF-/CVREF	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 and Comparator C2+ input. Default input configuration on POR; not affected by analog output.
	VREF-	1	I	ANA	A/D and comparator voltage reference low input.
	CVREF	x	O	ANA	Comparator voltage reference output. Enabling this feature disables digital I/O.
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 and Comparator C1+ input. Default input configuration on POR.
	VREF+	1	I	ANA	A/D and comparator voltage reference high input.
RA4/T0CKI/C1OUT	RA4	0	O	DIG	LATA<4> data output.
		1	I	ST	PORTA<4> data input; default configuration on POR.
	T0CKI	1	I	ST	Timer0 clock input.
	C1OUT	0	O	DIG	Comparator 1 output; takes priority over port data.
RA5/AN4/SS/ HLVDIN/C2OUT	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.
	SS	1	I	TTL	Slave Select input for MSSP module.
	HLVDIN	1	I	ANA	High/Low-Voltage Detect external trip point input.
OSC2/CLKO/RA6	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.
	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 RC, INTIO1 and EC Oscillator modes.
OSC1/CLKI/RA7	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.
	OSC1	x	I	ANA	Main oscillator input connection.
	CLKI	x	I	ANA	Main clock input connection.

**Legend:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output;  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

# PIC18F2525/2620/4525/4620

TABLE 9-2: 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	52
LATA	LATA7 <sup>(1)</sup>	LATA6 <sup>(1)</sup>	PORTA Data Latch Register (Read and Write to Data Latch)						52
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	PORTA Data Direction Control Register						52
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	51
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	51
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	51

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

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

## 9.2 PORTB, TRISB and LATB Registers

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

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

### EXAMPLE 9-2: INITIALIZING PORTB

```
CLRF  PORTB    ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF  LATB     ; Alternate method
                ; to clear output
                ; data latches
MOVLW 0Fh      ; Set RB<4:0> as
MOVWF ADCON1   ; digital I/O pins
                ; (required if config bit
                ; PBADEN is set)
MOVLW 0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

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

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

By programming the Configuration bit, PBADEN, RB4:RB0 will alternatively be configured as digital inputs on POR.

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

This interrupt can wake the device from the Sleep mode, or any of the Idle 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).
- 1 TCY.
- Clear flag bit, RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB and waiting 1 TCY will end the mismatch condition and allow flag bit, RBIF, to be cleared.

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

RB3 can be configured by the Configuration bit, CCP2MX, as the alternate peripheral pin for the CCP2 module (CCP2MX = 0).

# PIC18F2525/2620/4525/4620

TABLE 9-3: PORTB I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RB0/INT0/FLT0/ AN12	RB0	0	O	DIG	LATB<0> data output; not affected by analog input.
		1	I	TTL	PORTB<0> data input; weak pull-up when RBPU bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	INT0	1	I	ST	External interrupt 0 input.
	FLT0	1	I	ST	Enhanced PWM Fault input (ECCP1 module); enabled in software.
	AN12	1	I	ANA	A/D input channel 12. <sup>(1)</sup>
RB1/INT1/AN10	RB1	0	O	DIG	LATB<1> data output; not affected by analog input.
		1	I	TTL	PORTB<1> data input; weak pull-up when RBPU bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	INT1	1	I	ST	External interrupt 1 input.
	AN10	1	I	ANA	A/D input channel 10. <sup>(1)</sup>
RB2/INT2/AN8	RB2	0	O	DIG	LATB<2> data output; not affected by analog input.
		1	I	TTL	PORTB<2> data input; weak pull-up when RBPU bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	INT2	1	I	ST	External interrupt 2 input.
	AN8	1	I	ANA	A/D input channel 8. <sup>(1)</sup>
RB3/AN9/CCP2	RB3	0	O	DIG	LATB<3> data output; not affected by analog input.
		1	I	TTL	PORTB<3> data input; weak pull-up when RBPU bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	AN9	1	I	ANA	A/D input channel 9. <sup>(1)</sup>
	CCP2 <sup>(2)</sup>	0	O	DIG	CCP2 compare and PWM output.
		1	I	ST	CCP2 capture input.
RB4/KBI0/AN11	RB4	0	O	DIG	LATB<4> data output; not affected by analog input.
		1	I	TTL	PORTB<4> data input; weak pull-up when RBPU bit is cleared. Disabled when analog input enabled. <sup>(1)</sup>
	KBI0	1	I	TTL	Interrupt on pin change.
	AN11	1	I	ANA	A/D input channel 11. <sup>(1)</sup>
RB5/KBI1/PGM	RB5	0	O	DIG	LATB<5> data output.
		1	I	TTL	PORTB<5> data input; weak pull-up when RBPU bit is cleared.
	KBI1	1	I	TTL	Interrupt on pin change.
	PGM	x	I	ST	Single-Supply Programming mode entry (ICSP™). Enabled by LVP Configuration bit; all other pin functions disabled.
RB6/KBI2/PGC	RB6	0	O	DIG	LATB<6> data output.
		1	I	TTL	PORTB<6> data input; weak pull-up when RBPU bit is cleared.
	KBI2	1	I	TTL	Interrupt on pin change.
	PGC	x	I	ST	Serial execution (ICSP™) clock input for ICSP and ICD operation. <sup>(3)</sup>
RB7/KBI3/PGD	RB7	0	O	DIG	LATB<7> data output.
		1	I	TTL	PORTB<7> data input; weak pull-up when RBPU bit is cleared.
	KBI3	1	I	TTL	Interrupt on pin change.
	PGD	x	O	DIG	Serial execution data output for ICSP and ICD operation. <sup>(3)</sup>
		x	I	ST	Serial execution data input for ICSP and ICD operation. <sup>(3)</sup>

**Legend:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output;  
x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Configuration on POR is determined by the PBADEN Configuration bit. Pins are configured as analog inputs by default when PBADEN is set and digital inputs when PBADEN is cleared.

**2:** Alternate assignment for CCP2 when the CCP2MX Configuration bit is '0'. Default assignment is RC1.

**3:** All other pin functions are disabled when ICSP or ICD is enabled.

# PIC18F2525/2620/4525/4620

---

---

TABLE 9-4: 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	52
LATB	PORTB Data Latch Register (Read and Write to Data Latch)								
TRISB	PORTB Data Direction Control Register								
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
INTCON2	<u>RBPU</u>	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	49
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	49
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	51

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

## 9.3 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 Data 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 9-5). 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 additional information.

**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 9-3: 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
```

# PIC18F2525/2620/4525/4620

TABLE 9-5: PORTC I/O SUMMARY

Pin	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 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 enabled. Disables digital I/O.
	CCP2 <sup>(1)</sup>	0	O	DIG	CCP2 compare and PWM output; takes priority over port data.
		1	I	ST	CCP2 capture input.
RC2/CCP1/P1A	RC2	0	O	DIG	LATC<2> data output.
		1	I	ST	PORTC<2> data input.
	CCP1	0	O	DIG	ECCP1 compare or PWM output; takes priority over port data.
		1	I	ST	ECCP1 capture input.
	P1A <sup>(2)</sup>	0	O	DIG	ECCP1 Enhanced PWM output, channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data.
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 <sup>TM</sup> clock output (MSSP module); takes priority over port data.
		1	I	I <sup>2</sup> C/SMB	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	0	O	DIG	I <sup>2</sup> C data output (MSSP module); takes priority over port data.
		1	I	I <sup>2</sup> C/SMB	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/TX/CK	RC6	0	O	DIG	LATC<6> data output.
		1	I	ST	PORTC<6> data input.
	TX	0	O	DIG	Asynchronous serial transmit data output (EUSART module); takes priority over port data. User must configure as output.
	CK	0	O	DIG	Synchronous serial clock output (EUSART module); takes priority over port data.
		1	I	ST	Synchronous serial clock input (EUSART module).
RC7/RX/DT	RC7	0	O	DIG	LATC<7> data output.
		1	I	ST	PORTC<7> data input.
	RX	1	I	ST	Asynchronous serial receive data input (EUSART module).
	DT	0	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:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I<sup>2</sup>C/SMB = I<sup>2</sup>C/SMBus input buffer; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Default assignment for CCP2 when the CCP2MX Configuration bit is set. Alternate assignment is RB3.

**2:** Enhanced PWM output is available only on PIC18F4525/4620 devices.

# PIC18F2525/2620/4525/4620

---

TABLE 9-6: 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	52
LATC	PORTC Data Latch Register (Read and Write to Data Latch)								52
TRISC	PORTC Data Direction Control Register								52

## 9.4 PORTD, TRISD and LATD Registers

**Note:** PORTD is only available on 40/44-pin devices.

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

The Data 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.

Three of the PORTD pins are multiplexed with outputs P1B, P1C and P1D of the Enhanced CCP module. The operation of these additional PWM output pins is covered in greater detail in **Section 16.0 “Enhanced Capture/Compare/PWM (ECCP) Module”**.

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

PORTD can also be configured as an 8-bit wide micro-processor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL. See **Section 9.6 “Parallel Slave Port”** for additional information on the Parallel Slave Port (PSP).

**Note:** When the Enhanced PWM mode is used with either dual or quad outputs, the PSP functions of PORTD are automatically disabled.

### EXAMPLE 9-4: INITIALIZING PORTD

```
CLRF    PORTD    ; Initialize PORTD by
                ; clearing output
                ; data latches
CLRF    LATD     ; Alternate method
                ; to clear output
                ; data latches
MOVlw  0CFh     ; Value used to
                ; initialize data
                ; direction
MOVwf  TRISD    ; Set RD<3:0> as inputs
                ; RD<5:4> as outputs
                ; RD<7:6> as inputs
```

**TABLE 9-7: PORTD I/O SUMMARY**

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RD0/PSP0	RD0	0	O	DIG	LATD<0> data output.
		1	I	ST	PORTD<0> data input.
	PSP0	x	O	DIG	PSP read data output (LATD<0>); takes priority over port data.
		x	I	TTL	PSP write data input.
RD1/PSP1	RD1	0	O	DIG	LATD<1> data output.
		1	I	ST	PORTD<1> data input.
	PSP1	x	O	DIG	PSP read data output (LATD<1>); takes priority over port data.
		x	I	TTL	PSP write data input.
RD2/PSP2	RD2	0	O	DIG	LATD<2> data output.
		1	I	ST	PORTD<2> data input.
	PSP2	x	O	DIG	PSP read data output (LATD<2>); takes priority over port data.
		x	I	TTL	PSP write data input.
RD3/PSP3	RD3	0	O	DIG	LATD<3> data output.
		1	I	ST	PORTD<3> data input.
	PSP3	x	O	DIG	PSP read data output (LATD<3>); takes priority over port data.
		x	I	TTL	PSP write data input.
RD4/PSP4	RD4	0	O	DIG	LATD<4> data output.
		1	I	ST	PORTD<4> data input.
	PSP4	x	O	DIG	PSP read data output (LATD<4>); takes priority over port data.
		x	I	TTL	PSP write data input.
RD5/PSP5/P1B	RD5	0	O	DIG	LATD<5> data output.
		1	I	ST	PORTD<5> data input.
	PSP5	x	O	DIG	PSP read data output (LATD<5>); takes priority over port data.
		x	I	TTL	PSP write data input.
	P1B	0	O	DIG	ECCP1 Enhanced PWM output, channel B; takes priority over port and PSP data. May be configured for tri-state during Enhanced PWM shutdown events.
RD6/PSP6/P1C	RD6	0	O	DIG	LATD<6> data output.
		1	I	ST	PORTD<6> data input.
	PSP6	x	O	DIG	PSP read data output (LATD<6>); takes priority over port data.
		x	I	TTL	PSP write data input.
	P1C	0	O	DIG	ECCP1 Enhanced PWM output, channel C; takes priority over port and PSP data. May be configured for tri-state during Enhanced PWM shutdown events.
RD7/PSP7/P1D	RD7	0	O	DIG	LATD<7> data output.
		1	I	ST	PORTD<7> data input.
	PSP7	x	O	DIG	PSP read data output (LATD<7>); takes priority over port data.
		x	I	TTL	PSP write data input.
	P1D	0	O	DIG	ECCP1 Enhanced PWM output, channel D; takes priority over port and PSP data. May be configured for tri-state during Enhanced PWM shutdown events.

**Legend:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer;  
 x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

# PIC18F2525/2620/4525/4620

---

---

TABLE 9-8: 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	52
LATD	PORTD Data Latch Register (Read and Write to Data Latch)								52
TRISD	PORTD Data Direction Control Register								52
TRISE	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	52
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	51

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

## 9.5 PORTE, TRISE and LATE Registers

Depending on the particular PIC18F2525/2620/4525/4620 device selected, PORTE is implemented in two different ways.

For 40/44-pin devices, PORTE is a 4-bit wide port. Three pins (RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/AN7) are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. When selected as an analog input, these pins will read as '0's.

The corresponding Data Direction register is TRISE. Setting a TRISE bit (= 1) will make the corresponding PORTE pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISE bit (= 0) will make the corresponding PORTE pin an output (i.e., put the contents of the output latch on the selected pin).

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

**Note:** On a Power-on Reset, RE2:RE0 are configured as analog inputs.

The upper four bits of the TRISE register also control the operation of the Parallel Slave Port. Their operation is explained in Register 9-1.

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

The fourth pin of PORTE (MCLR/VPP/RE3) is an input only pin. Its operation is controlled by the MCLRE Configuration bit. When selected as a port pin (MCLRE = 0), it functions as a digital input only pin; as such, it does not have TRIS or LAT bits associated with its operation. Otherwise, it functions as the device's Master Clear input. In either configuration, RE3 also functions as the programming voltage input during programming.

**Note:** On a Power-on Reset, RE3 is enabled as a digital input only if Master Clear functionality is disabled.

### EXAMPLE 9-5: INITIALIZING PORTE

```
CLRF    PORTE    ; Initialize PORTE by
                  ; clearing output
                  ; data latches
CLRF    LATE     ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW  0Ah     ; Configure A/D
MOVWF  ADCON1  ; for digital inputs
MOVLW  03h     ; Value used to
                  ; initialize data
                  ; direction
MOVWF  TRISE   ; Set RE<0> as inputs
                  ; RE<1> as outputs
                  ; RE<2> as inputs
```

### 9.5.1 PORTE IN 28-PIN DEVICES

For 28-pin devices, PORTE is only available when Master Clear functionality is disabled (MCLRE = 0). In these cases, PORTE is a single bit, input only port comprised of RE3 only. The pin operates as previously described.

# PIC18F2525/2620/4525/4620

## REGISTER 9-1: TRISE REGISTER (40/44-PIN DEVICES ONLY)

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7      **IBF:** Input Buffer Full Status bit  
1 = A word has been received and waiting to be read by the CPU  
0 = No word has been received
- bit 6      **OBF:** Output Buffer Full Status bit  
1 = The output buffer still holds a previously written word  
0 = The output buffer has been read
- bit 5      **IBOV:** Input Buffer Overflow Detect bit (in Microprocessor mode)  
1 = A write occurred when a previously input word has not been read (must be cleared in software)  
0 = No overflow occurred
- bit 4      **PSPMODE:** Parallel Slave Port Mode Select bit  
1 = Parallel Slave Port mode  
0 = General purpose I/O mode
- bit 3      **Unimplemented:** Read as '0'
- bit 2      **TRISE2:** RE2 Direction Control bit  
1 = Input  
0 = Output
- bit 1      **TRISE1:** RE1 Direction Control bit  
1 = Input  
0 = Output
- bit 0      **TRISE0:** RE0 Direction Control bit  
1 = Input  
0 = Output

TABLE 9-9: PORTE I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RE0/RD/AN5	RE0	0	O	DIG	LATE<0> data output; not affected by analog input.
		1	I	ST	PORTE<0> data input; disabled when analog input enabled.
	RD	1	I	TTL	PSP read enable input (PSP enabled).
	AN5	1	I	ANA	A/D input channel 5; default input configuration on POR.
RE1/WR/AN6	RE1	0	O	DIG	LATE<1> data output; not affected by analog input.
		1	I	ST	PORTE<1> data input; disabled when analog input enabled.
	WR	1	I	TTL	PSP write enable input (PSP enabled).
	AN6	1	I	ANA	A/D input channel 6; default input configuration on POR.
RE2/CS/AN7	RE2	0	O	DIG	LATE<2> data output; not affected by analog input.
		1	I	ST	PORTE<2> data input; disabled when analog input enabled.
	CS	1	I	TTL	PSP write enable input (PSP enabled).
	AN7	1	I	ANA	A/D input channel 7; default input configuration on POR.
MCLR/VPP/RE3 <sup>(1)</sup>	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.
	RE3	— <sup>(2)</sup>	I	ST	PORTE<3> data input; enabled when MCLRE Configuration bit is clear.

**Legend:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** RE3 is available on both 28-pin and 40/44-pin devices. All other PORTE pins are only implemented on 40/44-pin devices.

**2:** RE3 does not have a corresponding TRIS bit to control data direction.

TABLE 9-10: 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	—	—	—	—	RE3 <sup>(1,2)</sup>	RE2	RE1	RE0	52
LATE <sup>(2)</sup>	—	—	—	—	—	PORTE Data Latch Register			52
TRISE	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	52
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	51

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

**Note 1:** Implemented only when Master Clear functionality is disabled (MCLRE Configuration bit = 0).

**2:** RE3 is the only PORTE bit implemented on both 28-pin and 40/44-pin devices. All other bits are implemented only when PORTE is implemented (i.e., 40/44-pin devices).

# PIC18F2525/2620/4525/4620

## 9.6 Parallel Slave Port

**Note:** The Parallel Slave Port is only available on 40/44-pin devices.

In addition to its function as a general I/O port, PORTD can also operate as an 8-bit wide Parallel Slave Port (PSP) or microprocessor port. PSP operation is controlled by the 4 upper bits of the TRISE register (Register 9-1). Setting control bit, PSPMODE (TRISE<4>), enables PSP operation as long as the Enhanced CCP module is not operating in dual output or quad output PWM mode. In Slave mode, the port is asynchronously readable and writable by the external world.

The PSP can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an 8-bit latch. Setting the control bit, PSPMODE, enables the PORTE I/O pins to become control inputs for the microprocessor port. When set, port pin RE0 is the RD input, RE1 is the WR input and RE2 is the CS (Chip Select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (set). The A/D port configuration bits, PFCG3:PFCG0 (ADCON1<3:0>), must also be set to a value in the range of '1010' through '1111'.

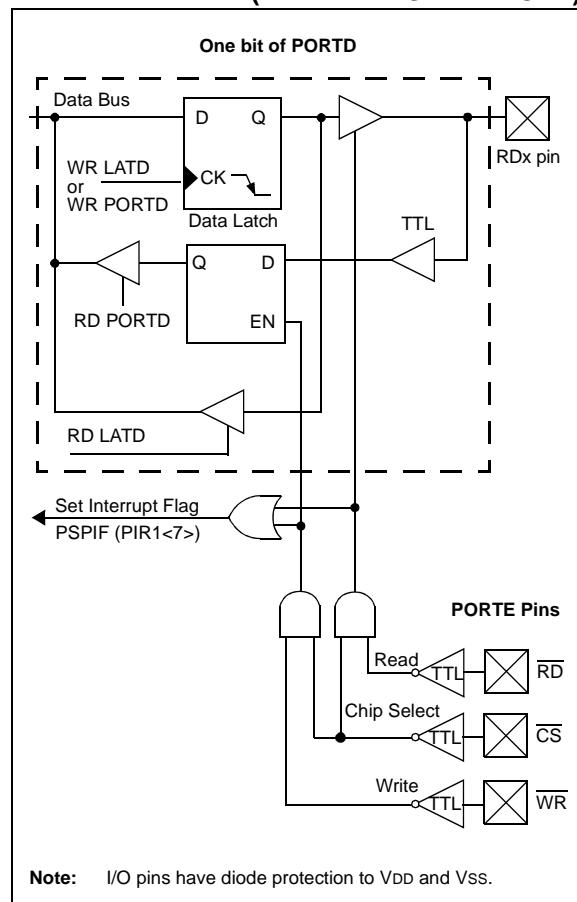
A write to the PSP occurs when both the CS and RD lines are first detected low and ends when either are detected high. The PSPIF and IBF flag bits are both set when the write ends.

A read from the PSP occurs when both the CS and RD lines are first detected low. The data in PORTD is read out and the OBF bit is clear. If the user writes new data to PORTD to set OBF, the data is immediately read out; however, the OBF bit is not set.

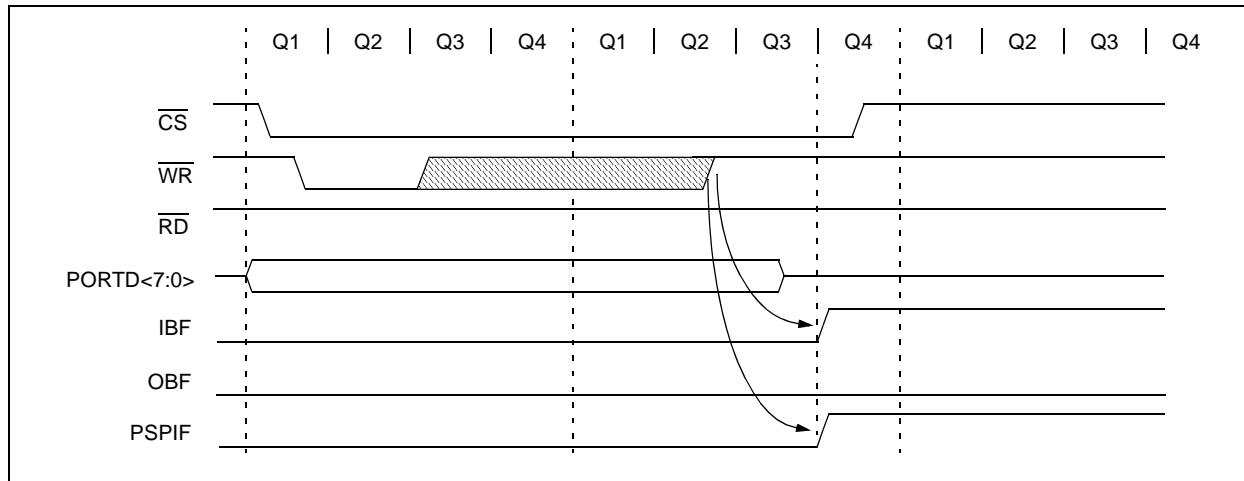
When either the CS or RD lines are detected high, the PORTD pins return to the input state and the PSPIF bit is set. User applications should wait for PSPIF to be set before servicing the PSP; when this happens, the IBF and OBF bits can be polled and the appropriate action taken.

The timing for the control signals in Write and Read modes is shown in Figure 9-3 and Figure 9-4, respectively.

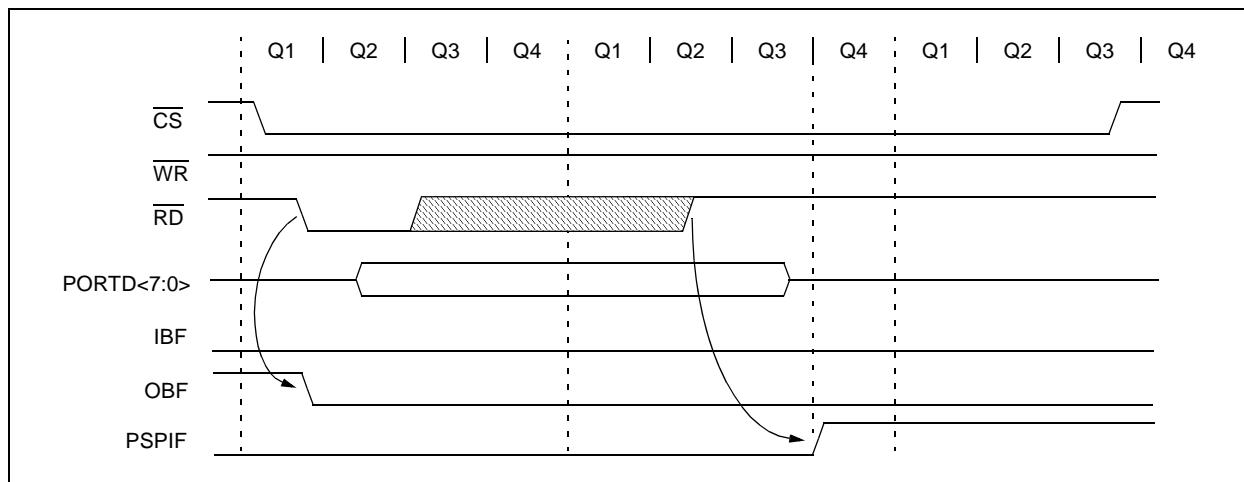
**FIGURE 9-2: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)**



**FIGURE 9-3: PARALLEL SLAVE PORT WRITE WAVEFORMS**



**FIGURE 9-4: PARALLEL SLAVE PORT READ WAVEFORMS**



**TABLE 9-11: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	52
LATD	PORTD Data Latch Register (Read and Write to Data Latch)								52
TRISD	PORTD Data Direction Control Register								52
PORTE	—	—	—	—	RE3	RE2	RE1	RE0	52
LATE	—	—	—	—	—	LATE Data Output bits			52
TRISE	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	52
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IF	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	51

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

**Note 1:** These bits are unimplemented on 28-pin devices and read as '0'.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 10.0 INTERRUPTS

The PIC18F2525/2620/4525/4620 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
- PIE1, PIE2
- IPR1, IPR2

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

In general, 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.

Compatibility mode (No priority)  
 ↳ Peripheral inten (PIEN)  
 ↳ Global inten (GIE)

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 be either the GIEH or GIEL bit. High-priority interrupt sources can interrupt a low-priority interrupt. Low-priority interrupts are not processed while high-priority interrupts are in progress.

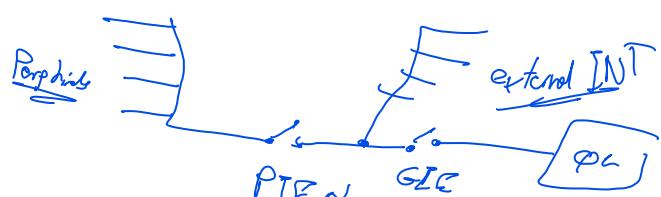
The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling 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.

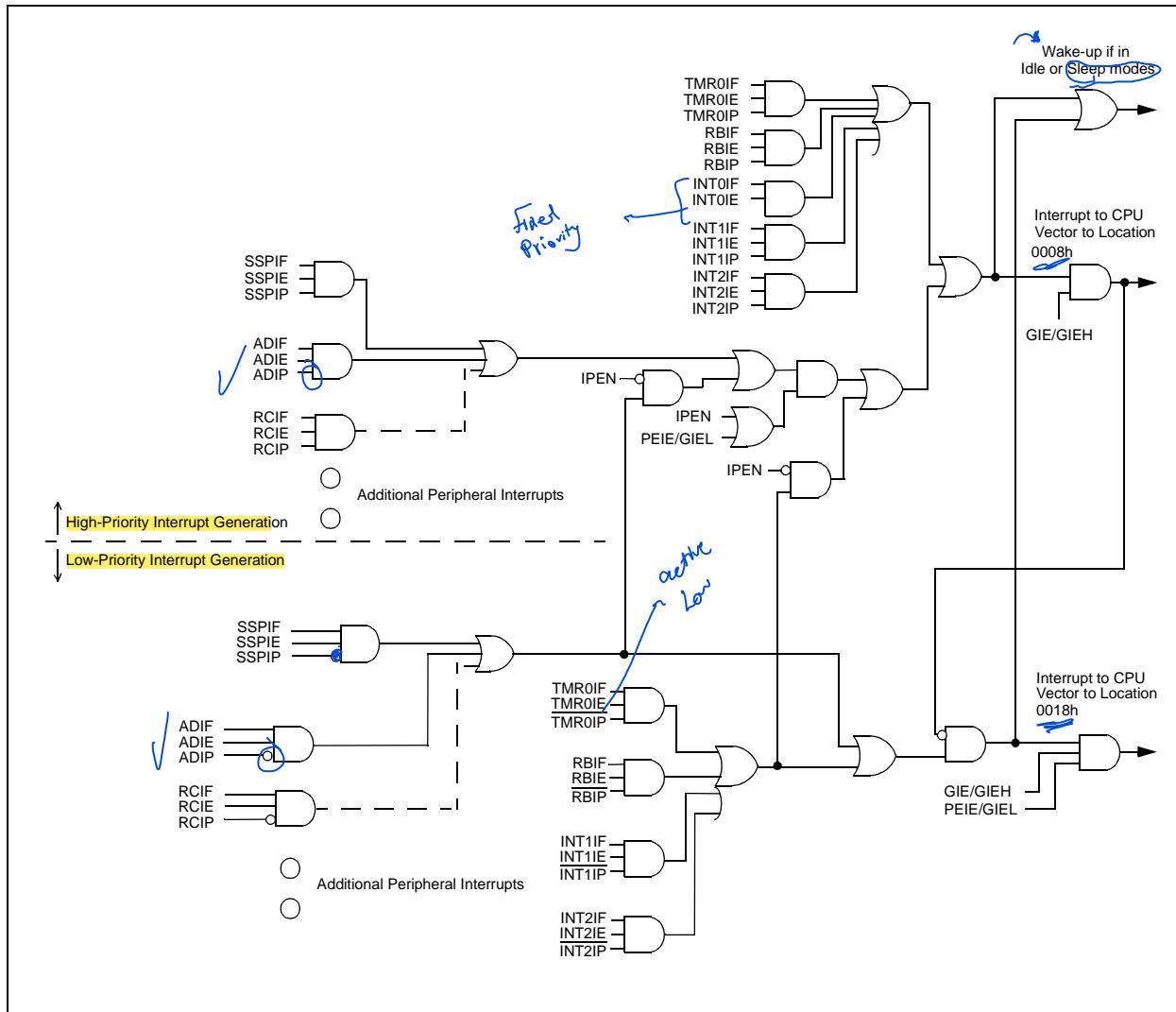
**Note:** Do not use the MOVFF instruction to modify any of the interrupt control registers while any interrupt is enabled. Doing so may cause erratic microcontroller behavior.

- ① IPEN & RCON<7>
- ② GIEL & GIEH <7> (INTON)



# PIC18F2525/2620/4525/4620

FIGURE 10-1: PIC18 INTERRUPT LOGIC



## 10.1 INTCON Registers

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

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

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

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF <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 RB7:RB4 pins changed state (must be cleared in software)
- 0 = None of the RB7:RB4 pins have changed state

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

# PIC18F2525/2620/4525/4620

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

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **RBPU:** PORTB Pull-up Enable bit

1 = All PORTB pull-ups are disabled

0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0:** External Interrupt 0 Edge Select bit

1 = Interrupt on rising edge

0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit

1 = Interrupt on rising edge

0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit

1 = Interrupt on rising edge

0 = Interrupt on falling edge

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

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit

1 = High priority

0 = Low priority

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

bit 0 **RBIP:** RB Port Change **Interrupt Priority bit**

1 = High priority

0 = Low priority

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

# PIC18F2525/2620/4525/4620

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

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

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7      **INT2IP**: INT2 External Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 6      **INT1IP**: INT1 External Interrupt Priority bit  
1 = High priority  
0 = Low priority
- bit 5      **Unimplemented**: Read as '0'
- bit 4      **INT2IE**: INT2 External Interrupt Enable bit  
1 = Enables the INT2 external interrupt  
0 = Disables the INT2 external interrupt
- bit 3      **INT1IE**: INT1 External Interrupt Enable bit  
1 = Enables the INT1 external interrupt  
0 = Disables the INT1 external interrupt
- bit 2      **Unimplemented**: Read as '0'
- bit 1      **INT2IF**: INT2 External Interrupt Flag bit  
1 = The INT2 external interrupt occurred (must be cleared in software)  
0 = The INT2 external interrupt did not occur
- bit 0      **INT1IF**: INT1 External Interrupt Flag bit  
1 = The INT1 external interrupt occurred (must be cleared in software)  
0 = The INT1 external interrupt did not occur

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

# **PIC18F2525/2620/4525/4620**

## 10.2 PIR Registers

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

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

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

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

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

**Legend:**

R = Readable bit  
-n = Value at POF

W = Writable bit  
'1' = Bit is set

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

x = Bit is unknown

bit 7	<b>PSPIF:</b> Parallel Slave Port Read/Write Interrupt Flag bit <sup>(1)</sup> 1 = A read or a write operation has taken place (must be cleared in software) 0 = No read or write has occurred
bit 6	<b>ADIF:</b> 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	<b>RCIF:</b> EUSART Receive Interrupt Flag bit 1 = The EUSART receive buffer, RCREG, is full (cleared when RCREG is read) 0 = The EUSART receive buffer is empty
bit 4	<b>TXIF:</b> EUSART Transmit Interrupt Flag bit 1 = The EUSART transmit buffer, TXREG, is empty (cleared when TXREG is written) 0 = The EUSART transmit buffer is full
bit 3	<b>SSPIF:</b> 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	<b>CCP1IF:</b> CCP1 Interrupt Flag bit <u>Capture mode:</u> 1 = A TMR1 register capture occurred (must be cleared in software) 0 = No TMR1 register capture occurred <u>Compare mode:</u> 1 = A TMR1 register compare match occurred (must be cleared in software) 0 = No TMR1 register compare match occurred <u>PWM mode:</u> Unused in this mode.
bit 1	<b>TMR2IF:</b> 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	<b>TMR1IF:</b> TMR1 Overflow Interrupt Flag bit 1 = TMR1 register overflowed (must be cleared in software) 0 = TMR1 register did not overflow

**Note 1:** This bit is unimplemented on 28-pin devices and will read as '0'.

# PIC18F2525/2620/4525/4620

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

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIF	CMIF	—	EEIF	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      **Unimplemented:** Read as '0'
- bit 4      **EEIF:** Data EEPROM/Flash Write Operation Interrupt Flag bit  
1 = The write operation is complete (must be cleared in software)  
0 = The write operation is not complete or has not been started
- bit 3      **BCLIF:** Bus Collision Interrupt Flag bit  
1 = A bus collision occurred (must be cleared in software)  
0 = No bus collision occurred
- bit 2      **HLVDIF:** High/Low-Voltage Detect Interrupt Flag bit  
1 = A high/low-voltage condition occurred (direction determined by VDIRMAG bit, HLVDCON<7>)  
0 = A high/low-voltage condition has not occurred
- 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 register capture occurred (must be cleared in software)  
0 = No TMR1 register capture occurred  
Compare mode:  
1 = A TMR1 register compare match occurred (must be cleared in software)  
0 = No TMR1 register compare match occurred  
PWM mode:  
Unused in this mode.

# PIC18F2525/2620/4525/4620

---

## 10.3 PIE Registers

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

### REGISTER 10-6: 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 <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							

#### 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      **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit<sup>(1)</sup>

1 = Enables the PSP read/write interrupt  
0 = Disables the PSP read/write interrupt

bit 6      **ADIE:** A/D Converter Interrupt Enable bit

1 = Enables the A/D interrupt  
0 = Disables the A/D interrupt

bit 5      **RCIE:** EUSART Receive Interrupt Enable bit

1 = Enables the EUSART receive interrupt  
0 = Disables the EUSART receive interrupt

bit 4      **TXIE:** EUSART Transmit Interrupt Enable bit

1 = Enables the EUSART transmit interrupt  
0 = Disables the EUSART transmit interrupt

bit 3      **SSPIE:** Master Synchronous Serial Port Interrupt Enable bit

1 = Enables the MSSP interrupt  
0 = Disables the MSSP interrupt

bit 2      **CCP1IE:** CCP1 Interrupt Enable bit

1 = Enables the CCP1 interrupt  
0 = Disables the CCP1 interrupt

bit 1      **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit

1 = Enables the TMR2 to PR2 match interrupt  
0 = Disables the TMR2 to PR2 match interrupt

bit 0      **TMR1IE:** TMR1 Overflow Interrupt Enable bit

1 = Enables the TMR1 overflow interrupt  
0 = Disables the TMR1 overflow interrupt

**Note 1:** This bit is unimplemented on 28-pin devices and will read as '0'.

## REGISTER 10-7: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIE	CMIE	—	EEIE	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      **Unimplemented:** Read as '0'
- bit 4      **EEIE:** Data EEPROM/Flash Write Operation Interrupt Enable bit  
           1 = Enabled  
           0 = Disabled
- bit 3      **BCLIE:** Bus Collision Interrupt Enable bit  
           1 = Enabled  
           0 = Disabled
- bit 2      **HLVDIE:** High/Low-Voltage Detect Interrupt Enable bit  
           1 = Enabled  
           0 = Disabled
- bit 1      **TMR3IE:** TMR3 Overflow Interrupt Enable bit  
           1 = Enabled  
           0 = Disabled
- bit 0      **CCP2IE:** CCP2 Interrupt Enable bit  
           1 = Enabled  
           0 = Disabled

# PIC18F2525/2620/4525/4620

---

---

## 10.4 IPR Registers

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

### REGISTER 10-8: 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 <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
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      **PSPIP:** Parallel Slave Port Read/Write Interrupt Priority bit<sup>(1)</sup>

1 = High priority  
0 = Low priority

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

1 = High priority  
0 = Low priority

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

1 = High priority  
0 = Low priority

bit 4      **TXIP:** EUSART Transmit Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 3      **SSPIP:** Master Synchronous Serial Port Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 2      **CCP1IP:** CCP1 Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 1      **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit

1 = High priority  
0 = Low priority

bit 0      **TMR1IP:** TMR1 Overflow Interrupt Priority bit

1 = High priority  
0 = Low priority

**Note 1:** This bit is unimplemented on 28-pin devices and will read as '0'.

# PIC18F2525/2620/4525/4620

## REGISTER 10-9: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

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

# PIC18F2525/2620/4525/4620

## 10.5 RCON Register

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

The operation of the SBOREN bit and the Reset flag bits is discussed in more detail in **Section 4.1 “RCON Register”**.

*Default* *R CON*

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

R/W-0	R/W-1 <sup>(1)</sup>	U-0	R/W-1	R-1	R-1	R/W-0 <sup>(1)</sup>	R/W-0
IPEN	SBOREN	—	RI	TO	PD	POR	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)

*Priority Feature*

bit 6

**SBOREN:** Software BOR Enable bit<sup>(1)</sup>

For details of bit operation, see Register 4-1.

bit 5

**Unimplemented:** Read as '0'

bit 4

**RI:** RESET Instruction Flag bit

For details of bit operation, see Register 4-1.

bit 3

**TO:** Watchdog Timer Time-out Flag bit

For details of bit operation, see Register 4-1.

bit 2

**PD:** Power-Down Detection Flag bit

For details of bit operation, see Register 4-1.

bit 1

**POR:** Power-on Reset Status bit<sup>(1)</sup>

For details of bit operation, see Register 4-1.

bit 0

**BOR:** Brown-out Reset Status bit

For details of bit operation, see Register 4-1.

**Note 1:** Actual Reset values are determined by device configuration and the nature of the device Reset. See Register 4-1 for additional information.

## 10.6 INTx Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1 and RB2/INT2 pins are edge-triggered. If the corresponding INTEDG<sub>x</sub> 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 RB<sub>x</sub>/INT<sub>x</sub> pin, the corresponding flag bit, INT<sub>x</sub>IF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INT<sub>x</sub>IE. Flag bit, INT<sub>x</sub>IF, must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1 and INT2) can wake-up the processor from Idle or Sleep modes if bit INT<sub>x</sub>IE was set prior to going into those modes. If the Global Interrupt Enable bit, GIE, is set, the processor will branch to the interrupt vector following wake-up.

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

## 10.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit, TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP (INTCON2<2>). See **Section 11.0 “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 5.3 “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
```

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 11.0 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 11-1) 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 11-1. Figure 11-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

### REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>TMR0ON:</b> Timer0 On/Off Control bit 1 = Enables Timer0 0 = Stops Timer0
bit 6	<b>T08BIT:</b> 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	<b>T0CS:</b> Timer0 Clock Source Select bit 1 = Transition on T0CKI pin 0 = Internal instruction cycle clock (CLKO)
bit 4	<b>T0SE:</b> 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	<b>PSA:</b> 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	<b>T0PS2:T0PS0:</b> 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

## 11.1 Timer0 Operation

Timer0 can operate as either a timer or a counter; the mode is selected with 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 11.3 “Prescaler”**). If the TMRO 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 TMRO register.

The Counter mode is selected by setting the T0CS bit (= 1). In this mode, Timer0 increments either on every rising or falling edge of pin RA4/T0CKI/C1OUT. 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

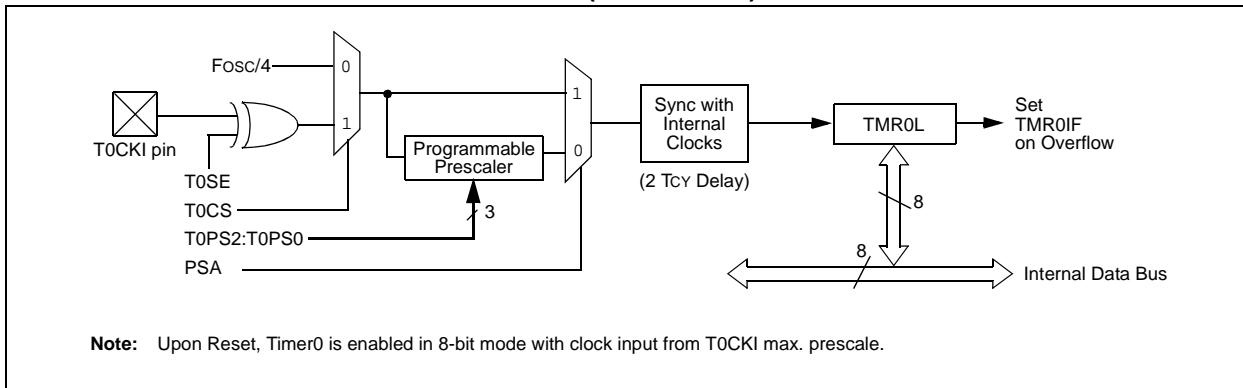
internal phase clock (Tosc). There is a delay between synchronization and the onset of incrementing the timer/counter.

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

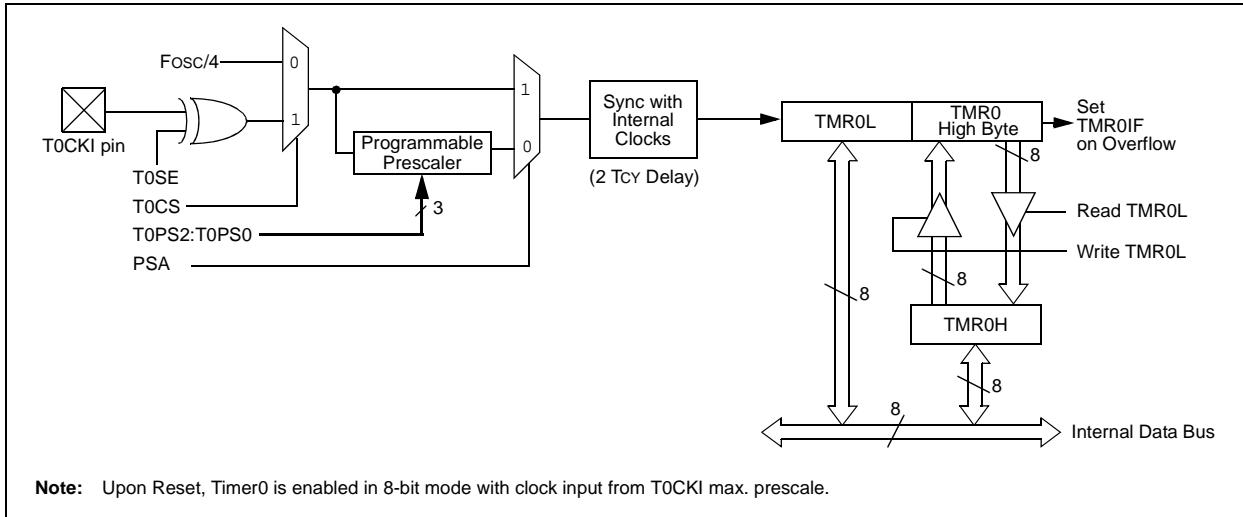
TMR0H is not the actual high byte of Timer0 in 16-bit mode; it is actually a buffered version of the real high byte of Timer0 which is not directly readable nor writable (refer to Figure 11-2). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMRO. This provides the ability 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 TMRO. This allows all 16 bits of Timer0 to be updated at once.

**FIGURE 11-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)**



**FIGURE 11-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)**



## 11.3 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 T0PS2:T0PS0 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.

### 11.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control and can be changed "on-the-fly" during program execution.

## 11.4 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 11-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page	
TMR0L	Timer0 Register Low Byte									50
TMR0H	Timer0 Register High Byte									50
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49	
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	50	
TRISA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5	RA4	RA3	RA2	RA1	RA0	52	

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

**Note 1:** PORTA<7:6> and their direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 12.0 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 12-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 12-2.

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 12-1). 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 12-1: 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	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	<b>RD16:</b> 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	<b>T1RUN:</b> Timer1 System Clock Status bit 1 = Device clock is derived from Timer1 oscillator 0 = Device clock is derived from another source
bit 5-4	<b>T1CKPS1:T1CKPS0:</b> 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	<b>T1OSCEN:</b> 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	<b>T1SYNC:</b> Timer1 External Clock Input Synchronization Select bit <u>When TMR1CS = 1:</u> 1 = Do not synchronize external clock input 0 = Synchronize external clock input <u>When TMR1CS = 0:</u> This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.
bit 1	<b>TMR1CS:</b> Timer1 Clock Source Select bit 1 = External clock from pin RC0/T1OSO/T13CKI (on the rising edge) 0 = Internal clock (Fosc/4)
bit 0	<b>TMR1ON:</b> Timer1 On bit 1 = Enables Timer1 0 = Stops Timer1

# PIC18F2525/2620/4525/4620

## 12.1 Timer1 Operation

Timer1 can operate in one of these modes:

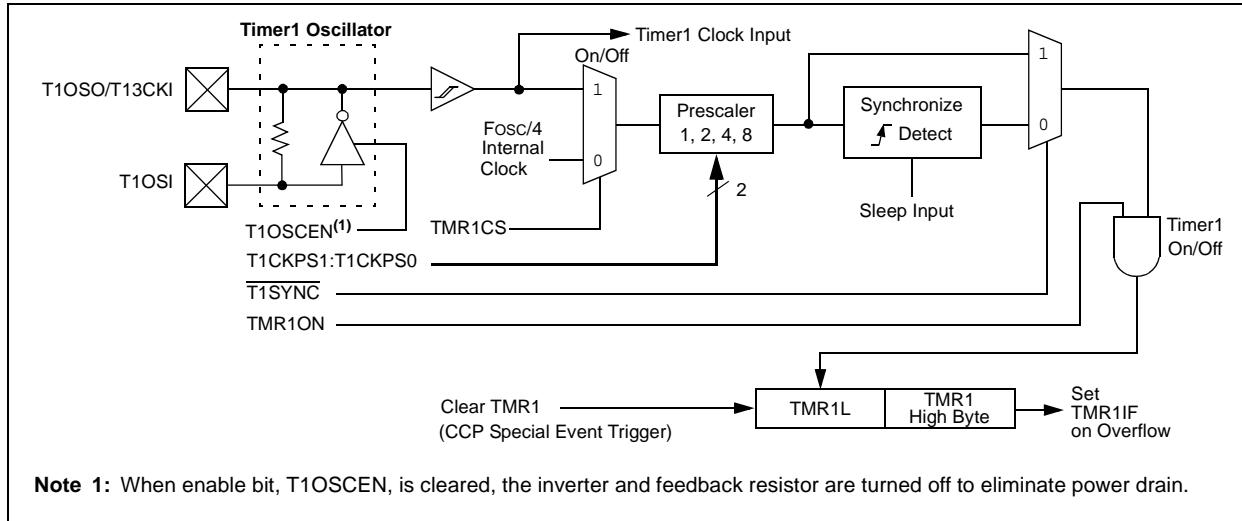
- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>). When TMR3CS is cleared (= 0), Timer1 increments on every internal instruction

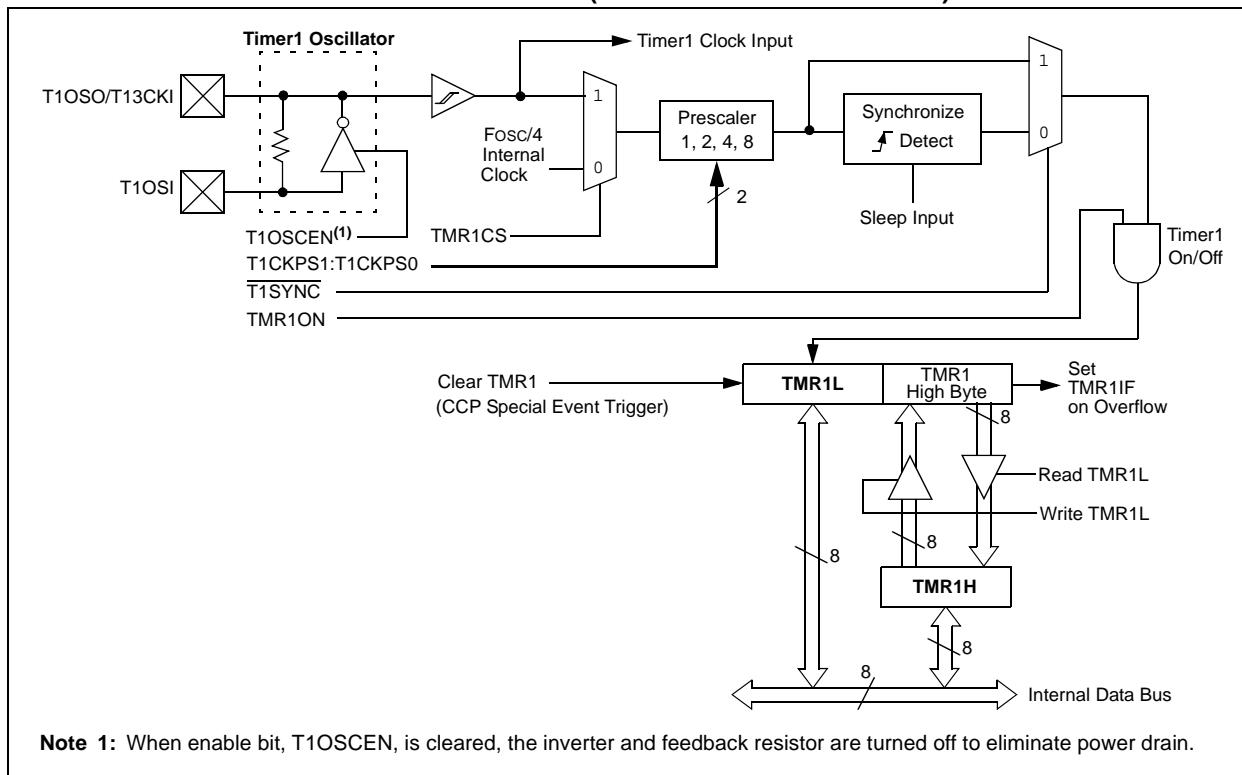
cycle (Fosc/4). When the bit is set, Timer1 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

When Timer1 is enabled, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

**FIGURE 12-1: TIMER1 BLOCK DIAGRAM**



**FIGURE 12-2: TIMER1 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**



## 12.2 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 12-2). When the RD16 control bit (T1CON<7>) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 high byte buffer. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, 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.

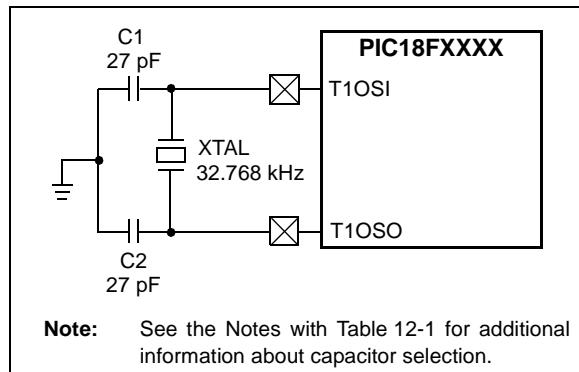
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

## 12.3 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 12-3. Table 12-1 shows the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

**FIGURE 12-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR**



**TABLE 12-1: 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.

### 12.3.1 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, SCS1:SCS0 (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 3.0 “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.

### 12.3.2 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.

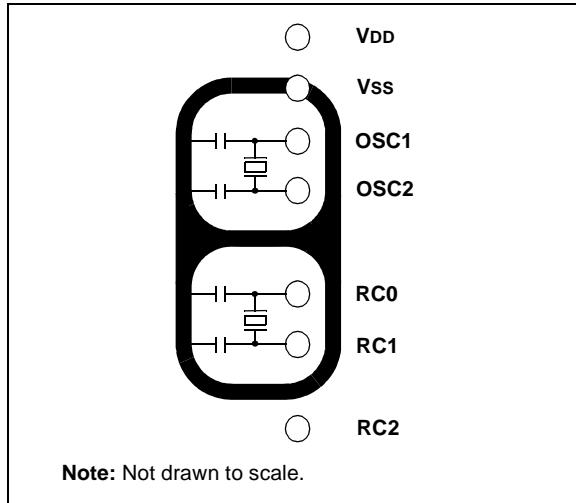
### 12.3.3 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

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 12-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than Vss or Vdd.

If a high-speed circuit must be located near the oscillator (such as the CCP1 pin in Output Compare or PWM mode, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as shown in Figure 12-4, may be helpful when used on a single-sided PCB or in addition to a ground plane.

**FIGURE 12-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING**



### 12.4 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled or disabled by setting or clearing the Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

### 12.5 Resetting Timer1 Using the CCP Special Event Trigger

If either of the CCP modules is configured to use Timer1 and generate a Special Event Trigger in Compare mode (CCP1M3:CCP1M0 or CCP2M3:CCP2M0 = 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 15.3.4 “Special Event Trigger”** for more information).

The module must be configured as either a timer or a synchronous counter to take advantage of this feature. When used this way, the CCPRH:CCPRL register pair effectively becomes a period register for Timer1.

If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a Special Event Trigger, the write operation will take precedence.

**Note:** The Special Event Triggers from the CCP2 module will not set the TMR1IF interrupt flag bit (PIR1<0>).

### 12.6 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 12.3 “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 12-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow triggers the interrupt and calls the routine, which increments the seconds counter by one; additional counters for minutes and hours are incremented as the previous counter overflow.

Since the register pair is 16 bits wide, counting up to overflow the register directly from a 32.768 kHz clock would take 2 seconds. To force the overflow at the required one-second intervals, it is necessary to preload it. The simplest method is to set the MSb of TMR1H with a BSF instruction. Note that the TMR1L register is never preloaded or altered. Doing so may introduce cumulative errors 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.

# PIC18F2525/2620/4525/4620

## EXAMPLE 12-1: 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
    RETURN
    CLRF   secs          ; Clear seconds
    INCF   mins, F        ; Increment minutes
    MOVLW  .59            ; 60 minutes elapsed?
    CPFSGT
    RETURN
    CLRF   mins          ; No, done
    INCF   hours, F       ; clear minutes
    MOVLW  .23            ; Increment hours
    CPFSGT
    RETURN
    CLRF   hours          ; 24 hours elapsed?
    CPFSGT
    RETURN
    CLRF   hours          ; No, done
    CLRF   hours          ; Reset hours
    RETURN
    CLRF   hours          ; Done
    RETURN

```

TABLE 12-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
TMR1L	Timer1 Register Low Byte								50
TMR1H	Timer1 Register High Byte								50
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	50

**Legend:** Shaded cells are not used by the Timer1 module.

**Note 1:** These bits are unimplemented on 28-pin devices and read as '0'.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 13.0 TIMER2 MODULE

The Timer2 module timer incorporates the following features:

- 8-bit Timer and Period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable postscale (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 13-1), which enables or disables the timer and configures the prescaler and postscale. 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 13-1.

## 13.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (Fosc/4). A 4-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, T2CKPS1:T2CKPS0 (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/postscale (see **Section 13.2 “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 postscale 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, MCLR Reset, Watchdog Timer Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

### REGISTER 13-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	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      **T2OUTPS3:T2OUTPS0:** 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      **T2CKPS1:T2CKPS0:** Timer2 Clock Prescale Select bits

00 = Prescaler is 1

01 = Prescaler is 4

1x = Prescaler is 16

# PIC18F2525/2620/4525/4620

## 13.2 Timer2 Interrupt

Timer2 can also 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>).

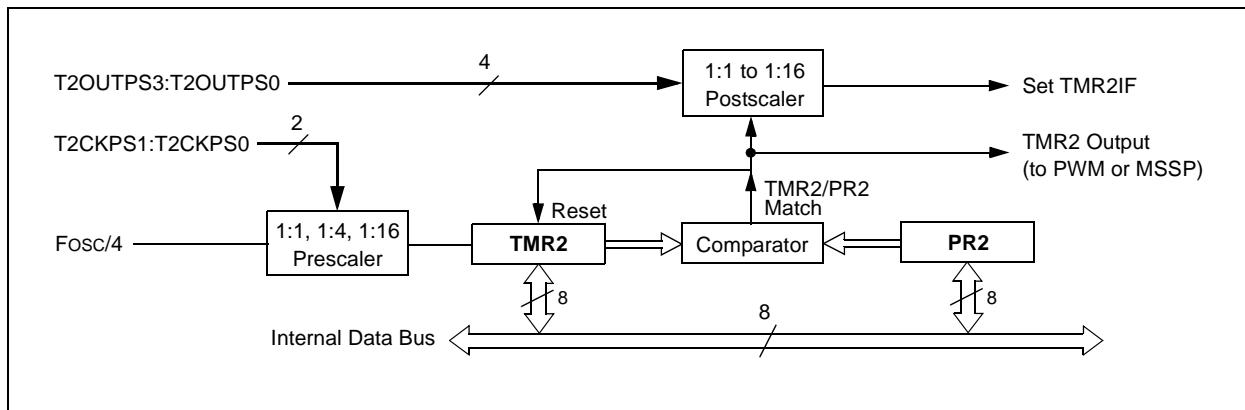
A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscale control bits, T2OUTPS3:T2OUTPS0 (T2CON<6:3>).

## 13.3 Timer2 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 13-1: TIMER2 BLOCK DIAGRAM**



**TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
TMR2	Timer2 Register								50
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	50
PR2	Timer2 Period Register								50

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

**Note 1:** These bits are unimplemented on 28-pin devices and read as '0'.



## 14.1 Timer3 Operation

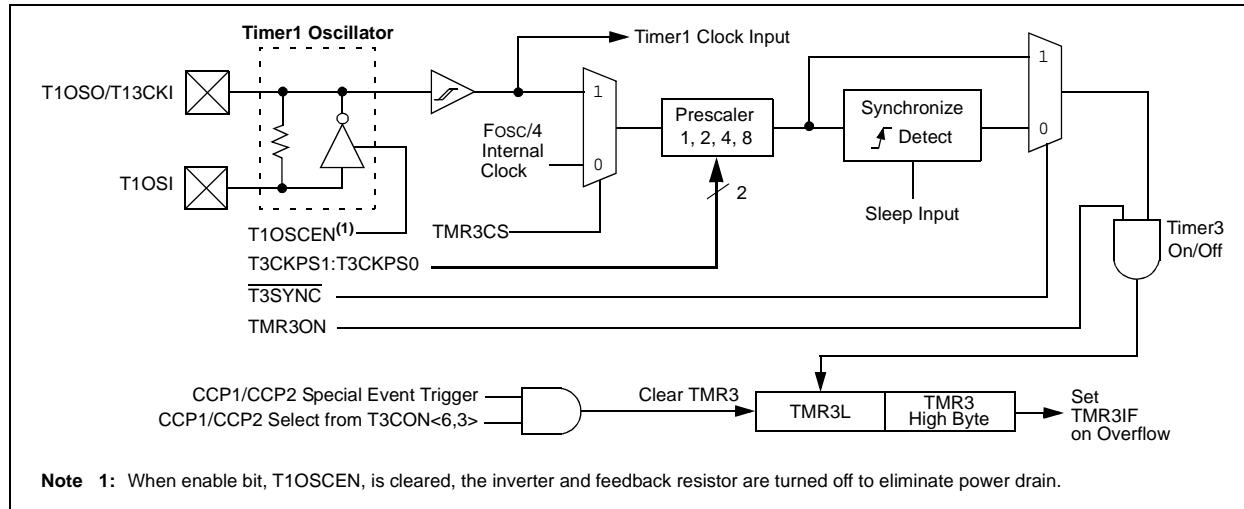
Timer3 can operate in one of three modes:

- Timer
- Synchronous Counter
- Asynchronous Counter

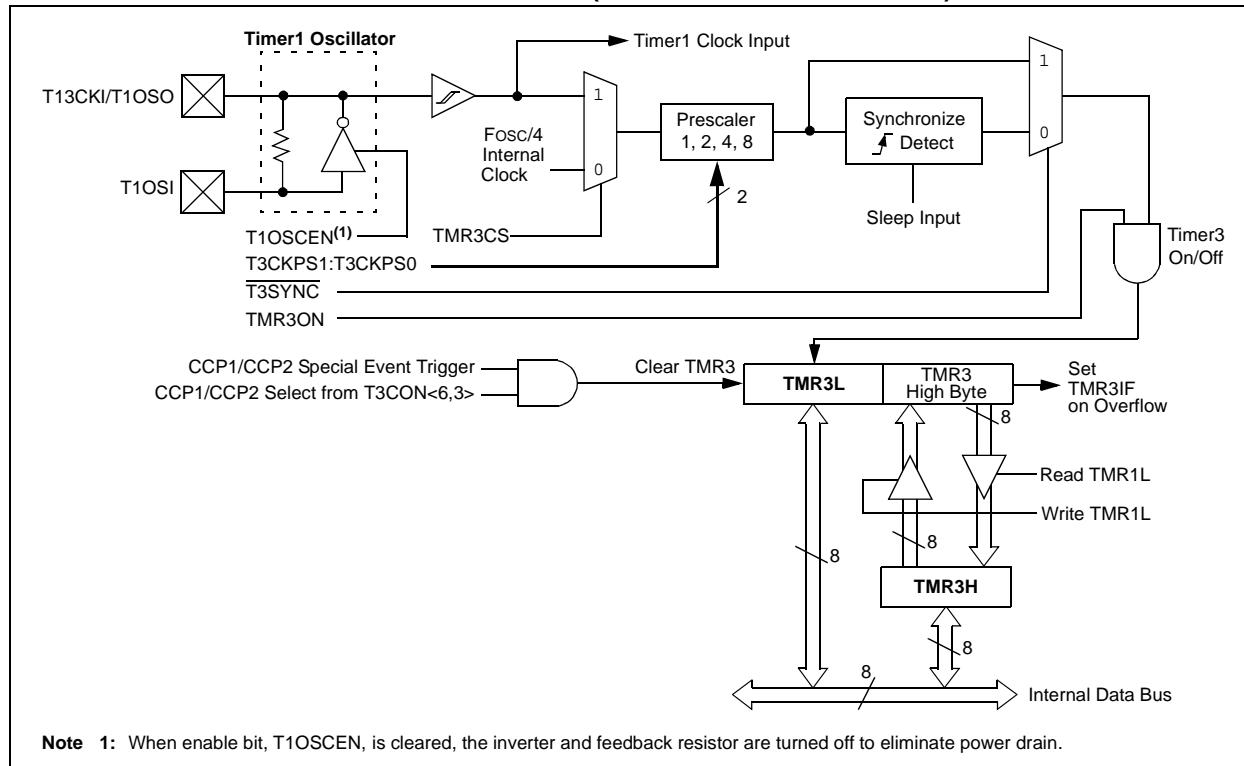
The operating mode is determined by the clock select bit, TMR3CS (T3CON<1>). When TMR3CS is cleared (= 0), Timer3 increments on every internal instruction cycle (Fosc/4). When the bit is set, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

As with Timer1, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs when the Timer1 oscillator is enabled. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

**FIGURE 14-1: TIMER3 BLOCK DIAGRAM**



**FIGURE 14-2: TIMER3 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**



## 14.2 Timer3 16-Bit Read/Write Mode

Timer3 can be configured for 16-bit reads and writes (see Figure 14-2). 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.

## 14.3 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 12.0 “Timer1 Module”**.

**TABLE 14-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RBIE	TMROIF	INT0IF	RBIF	49
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52
TMR3L	Timer3 Register Low Byte								51
TMR3H	Timer3 Register High Byte								51
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	50
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	51

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

## 14.4 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>).

## 14.5 Resetting Timer3 Using the CCP Special Event Trigger

If either of the CCP modules is configured to use Timer3 and to generate a Special Event Trigger in Compare mode (CCP1M3:CCP1M0 or CCP2M3:CCP2M0 = 1011), this signal will reset Timer3. It will also start an A/D conversion if the A/D module is enabled (see **Section 15.3.4 “Special Event Trigger”** for more information).

The module must be configured as either a timer or synchronous counter to take advantage of this feature. When used this way, the CCP2H:CCP2L 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>).

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

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

PIC18F2525/2620/4525/4620 devices all have two CCP (Capture/Compare/PWM) modules. Each 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.

In 28-pin devices, the two standard CCP modules (CCP1 and CCP2) operate as described in this chapter. In 40/44-pin devices, CCP1 is implemented as an Enhanced CCP module with standard Capture and Compare modes and Enhanced PWM modes. The ECCP implementation is discussed in **Section 16.0 “Enhanced Capture/Compare/PWM (ECCP) Module”**.

The Capture and Compare operations described in this chapter apply to all standard and Enhanced CCP modules.

**Note:** Throughout this section and **Section 16.0 “Enhanced Capture/Compare/PWM (ECCP) Module”**, references to the register and bit names for CCP modules are referred to generically by the use of 'x' or 'y' in place of the specific module number. Thus, "CCPxCON" might refer to the control register for CCP1, CCP2 or ECCP1. "CCPxCON" is used throughout these sections to refer to the module control register, regardless of whether the CCP module is a standard or enhanced implementation.

### REGISTER 15-1: CCPxCON: CCPx CONTROL REGISTER (28-PIN DEVICES)

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

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

bit 5-4      **DCxB1:DCxB0:** PWM Duty Cycle bit 1 and bit 0 for CCPx Module

#### Capture mode:

Unused.

#### Compare mode:

Unused.

#### PWM mode:

These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs (DCx9:DCx2) of the duty cycle are found in CCPRxL.

bit 3-0      **CCPxM3:CCPxM0:** CCPx Module 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; CCPx match starts A/D conversion (CCPxIF bit is set)

11xx = PWM mode

# PIC18F2525/2620/4525/4620

## 15.1 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.

### 15.1.1 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 15-1: CCP MODE – TIMER RESOURCES**

CCP/ECCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
PWM	Timer2

The assignment of a particular timer to a module is determined by the Timer to CCP enable bits in the T3CON register (Register 14-1). Both modules may be active at any given time and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time. The interactions between the two modules are summarized in Figure 15-1 and Figure 15-2. In Timer1 in Asynchronous Counter mode, the capture operation will not work.

### 15.1.2 CCP2 PIN ASSIGNMENT

The pin assignment for CCP2 (capture input, compare and PWM output) can change, based on device configuration. The CCP2MX Configuration bit determines which pin CCP2 is multiplexed to. By default, it is assigned to RC1 (CCP2MX = 1). If the Configuration bit is cleared, CCP2 is multiplexed with RB3.

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.

**TABLE 15-2: INTERACTIONS BETWEEN CCP1 AND CCP2 FOR TIMER RESOURCES**

CCP1 Mode	CCP2 Mode	Interaction
Capture	Capture	Each module can use TMR1 or TMR3 as the time base. The time base can be different for each CCP.
Capture	Compare	CCP2 can be configured for the Special Event Trigger to reset TMR1 or TMR3 (depending upon which time base is used). Automatic A/D conversions on trigger event can also be done. Operation of CCP1 could be affected if it is using the same timer as a time base.
Compare	Capture	CCP1 can be configured for the Special Event Trigger to reset TMR1 or TMR3 (depending upon which time base is used). Operation of CCP2 could be affected if it is using the same timer as a time base.
Compare	Compare	Either module can be configured for the Special Event Trigger to reset the time base. Automatic A/D conversions on CCP2 trigger event can be done. Conflicts may occur if both modules are using the same time base.
Capture	PWM <sup>(1)</sup>	None
Compare	PWM <sup>(1)</sup>	None
PWM <sup>(1)</sup>	Capture	None
PWM <sup>(1)</sup>	Compare	None
PWM <sup>(1)</sup>	PWM	Both PWMs will have the same frequency and update rate (TMR2 interrupt).

**Note 1:** Includes standard and Enhanced PWM operation.

## 15.2 Capture Mode

In Capture mode, the CCPRxH:CCPRxL register pair captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on the corresponding CCPx pin. 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, CCPxM3:CCPxM0 (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit, CCPxIF, is set; it must be cleared in software. If another capture occurs before the value in register CCPRx is read, the old captured value is overwritten by the new captured value.

### 15.2.1 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 RB3/CCP2 or RC1/CCP2 is configured as an output, a write to the port can cause a capture condition.

### 15.2.2 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 will not work. The timer to be used with each CCP module is selected in the T3CON register (see **Section 15.1.1 “CCP Modules and Timer Resources”**).

## 15.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE interrupt enable bit clear to avoid false interrupts. The interrupt flag bit, CCPxIF, should also be cleared following any such change in operating mode.

## 15.2.4 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 (CCPxM3:CCPxM0). Whenever the CCP module is turned off or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared; therefore, the first capture may be from a non-zero prescaler. Example 15-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

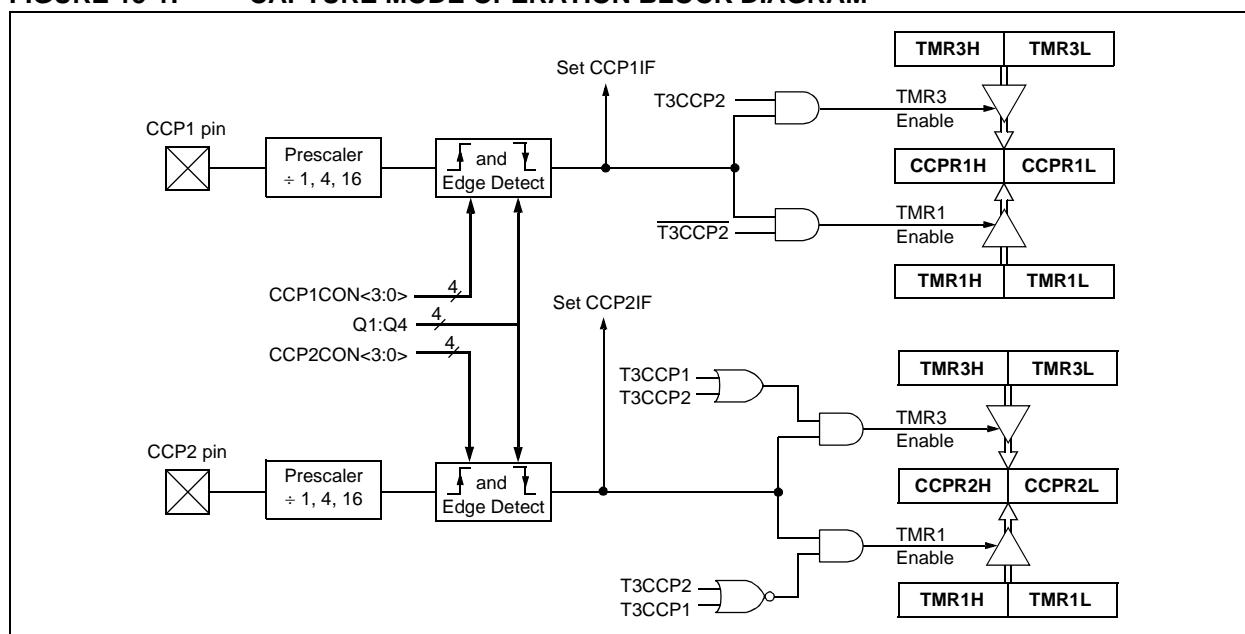
### EXAMPLE 15-1: CHANGING BETWEEN CAPTURE PRESCALERS (CCP2 SHOWN)

```

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

```

**FIGURE 15-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**



## 15.3 Compare Mode

In Compare mode, the 16-bit CCPRx register value is constantly compared against either the TMR1 or TMR3 register pair value. When a match occurs, the CCPx 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 (CCPxM3:CCPxM0). At the same time, the interrupt flag bit, CCPxIF, is set.

### 15.3.1 CCP PIN CONFIGURATION

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

**Note:** Clearing the CCP2CON register will force the RB3 or RC1 compare output latch (depending on device configuration) to the default low level. This is not the PORTB or PORTC I/O data latch.

### 15.3.2 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.

### 15.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCPxM3:CCPxM0 = 1010), the corresponding CCPx pin is not affected. Only a CCP interrupt is generated, if enabled, and the CCPxIE bit is set.

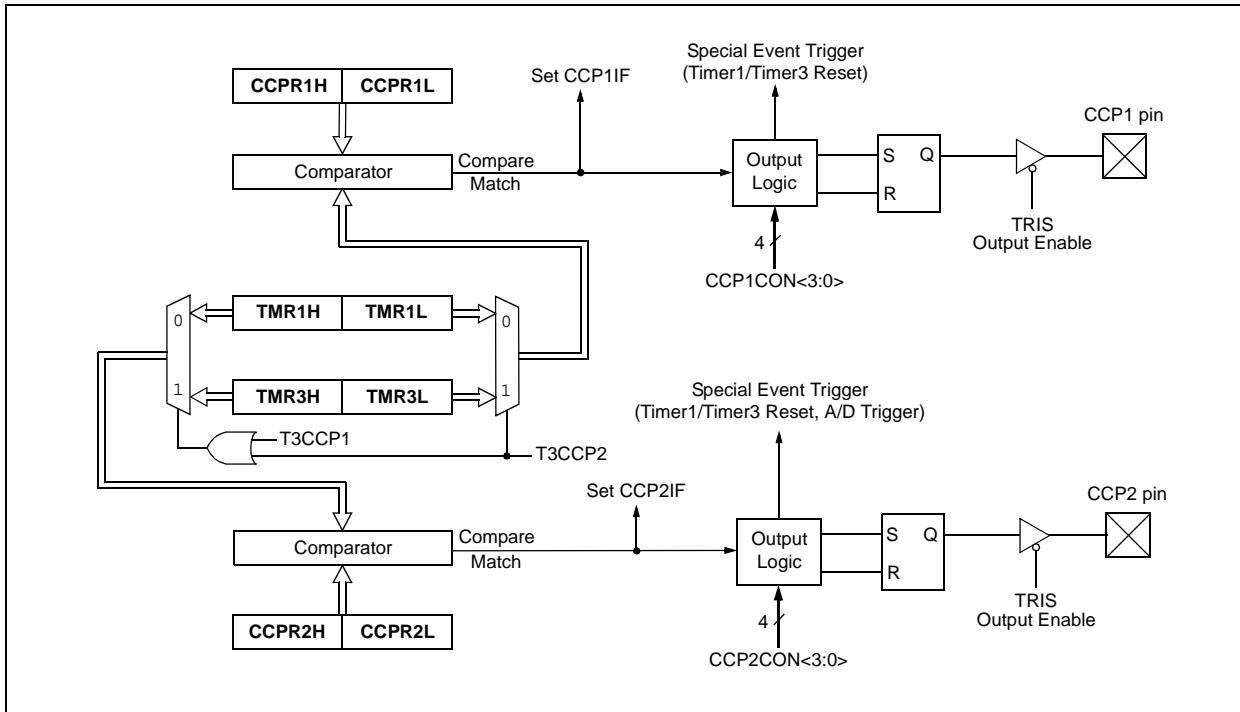
### 15.3.4 SPECIAL EVENT TRIGGER

Both CCP modules are equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode (CCPxM3:CCPxM0 = 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.

**FIGURE 15-2: COMPARE MODE OPERATION BLOCK DIAGRAM**



# PIC18F2525/2620/4525/4620

TABLE 15-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
RCON	IPEN	SBOREN <sup>(1)</sup>	—	RI	TO	PD	POR	BOR	48
PIR1	PSPIF <sup>(2)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(2)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(2)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52
TRISB	PORTB Data Direction Control Register								52
TRISC	PORTC Data Direction Control Register								52
TMR1L	Timer1 Register Low Byte								50
TMR1H	Timer1 Register High Byte								50
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	50
TMR3H	Timer3 Register High Byte								51
TMR3L	Timer3 Register Low Byte								51
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	51
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								51
CCPR1H	Capture/Compare/PWM Register 1 High Byte								51
CCP1CON	P1M1 <sup>(2)</sup>	P1M0 <sup>(2)</sup>	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	51
CCPR2L	Capture/Compare/PWM Register 2 Low Byte								51
CCPR2H	Capture/Compare/PWM Register 2 High Byte								51
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	51

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

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 4.4 “Brown-out Reset (BOR)”**.

**2:** These bits are unimplemented on 28-pin devices and read as '0'.

## 15.4 PWM Mode

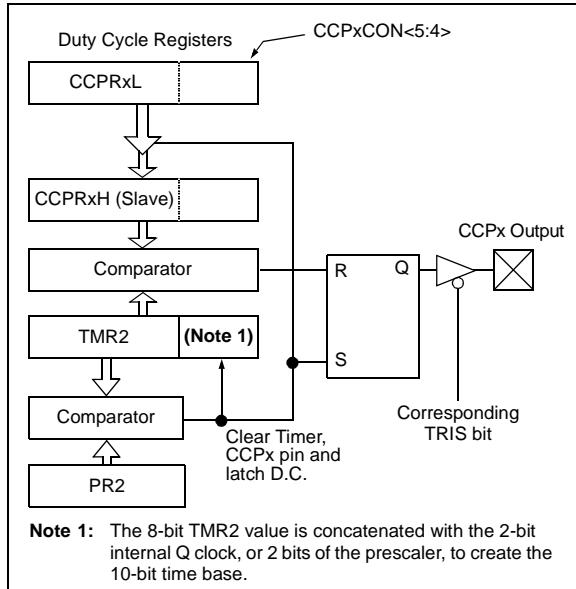
In Pulse-Width Modulation (PWM) mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP2 pin is multiplexed with a PORTB or PORTC data latch, the appropriate TRIS bit must be cleared to make the CCP2 pin an output.

**Note:** Clearing the CCP2CON register will force the RB3 or RC1 output latch (depending on device configuration) to the default low level. This is not the PORTB or PORTC I/O data latch.

Figure 15-3 shows a simplified block diagram of the CCP module in PWM mode.

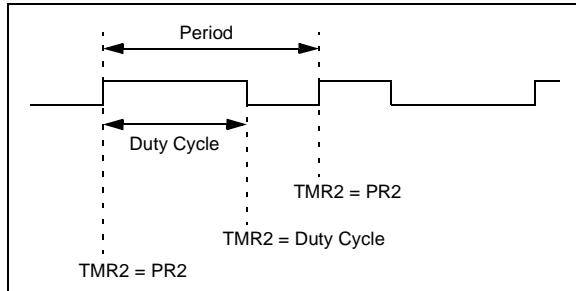
For a step-by-step procedure on how to set up the CCP module for PWM operation, see **Section 15.4.4 “Setup for PWM Operation”**.

**FIGURE 15-3: SIMPLIFIED PWM BLOCK DIAGRAM**



A PWM output (Figure 15-4) has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

**FIGURE 15-4: PWM OUTPUT**



### 15.4.1 PWM PERIOD

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

#### EQUATION 15-1:

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

PWM frequency is defined as  $1/[\text{PWM period}]$ .

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCPx pin is set (exception: if PWM duty cycle = 0%, the CCPx pin will not be set)
- The PWM duty cycle is latched from CCPRxL into CCPRxH

**Note:** The Timer2 postscalers (see **Section 13.0 “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.

### 15.4.2 PWM DUTY CYCLE

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

#### EQUATION 15-2:

$$\text{PWM Duty Cycle} = (CCPRxL:CCPxCON<5:4>) \cdot TOSC \cdot (\text{TMR2 Prescale Value})$$

CCPRxL and CCPxCON<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, CCPRxH 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 CCPRxH 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 15-3:

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

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

**TABLE 15-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz**

PWM Frequency	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	FFh	FFh	FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	10	10	10	8	7	6.58

#### 15.4.3 PWM AUTO-SHUTDOWN (CCP1 ONLY)

The PWM auto-shutdown features of the Enhanced CCP module are also available to CCP1 in 28-pin devices. The operation of this feature is discussed in detail in **Section 16.4.7 “Enhanced PWM Auto-Shutdown”**.

Auto-shutdown features are not available for CCP2.

#### 15.4.4 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCPx 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 CCPRxL register and CCPxCON<5:4> bits.
3. Make the CCPx 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 CCPx module for PWM operation.

# PIC18F2525/2620/4525/4620

TABLE 15-5: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
RCON	IPEN	SBOREN <sup>(1)</sup>	—	RI	TO	PD	POR	BOR	48
PIR1	PSPIF <sup>(2)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(2)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(2)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
TRISB	PORTB Data Direction Control Register								52
TRISC	PORTC Data Direction Control Register								52
TMR2	Timer2 Register								50
PR2	Timer2 Period Register								50
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	50
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								51
CCPR1H	Capture/Compare/PWM Register 1 High Byte								51
CCP1CON	P1M1 <sup>(2)</sup>	P1M0 <sup>(2)</sup>	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	51
CCPR2L	Capture/Compare/PWM Register 2 Low Byte								51
CCPR2H	Capture/Compare/PWM Register 2 High Byte								51
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	51
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1 <sup>(2)</sup>	PSSBD0 <sup>(2)</sup>	51
PWM1CON	PRSEN	PDC6 <sup>(2)</sup>	PDC5 <sup>(2)</sup>	PDC4 <sup>(2)</sup>	PDC3 <sup>(2)</sup>	PDC2 <sup>(2)</sup>	PDC1 <sup>(2)</sup>	PDC0 <sup>(2)</sup>	51

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

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 4.4 "Brown-out Reset (BOR)"**.

**2:** These bits are unimplemented on 28-pin devices and read as '0'.

## 16.0 ENHANCED CAPTURE/COMPARE/PWM (ECCP) MODULE

**Note:** The ECCP module is implemented only in 40/44-pin devices.

In PIC18F4525/4620 devices, CCP1 is implemented as a standard CCP module with Enhanced PWM capabilities. These include the provision for 2 or 4 output channels, user-selectable polarity, dead-band control and automatic shutdown and restart. The

Enhanced features are discussed in detail in **Section 16.4 “Enhanced PWM Mode”**. Capture, Compare and single-output PWM functions of the ECCP module are the same as described for the standard CCP module.

The control register for the Enhanced CCP module is shown in Register 16-1. It differs from the CCPxCON registers in PIC18F2525/2620 devices in that the two Most Significant bits are implemented to control PWM functionality.

### REGISTER 16-1: CCP1CON: ECCP CONTROL REGISTER (40/44-PIN DEVICES)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6	<b>P1M1:P1M0:</b> Enhanced PWM Output Configuration bits  <u>If CCP1M3:CCP1M2 = 00, 01, 10:</u> xx = P1A assigned as capture/compare input/output; P1B, P1C, P1D assigned as port pins <u>If CCP1M3:CCP1M2 = 11:</u> 00 = Single output, P1A modulated; P1B, P1C, P1D assigned as port pins 01 = Full-bridge output forward, P1D modulated; P1A active; P1B, P1C inactive 10 = Half-bridge output, P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins 11 = Full-bridge output reverse, P1B modulated; P1C active; P1A, P1D inactive
bit 5-4	<b>DC1B1:DC1B0:</b> PWM Duty Cycle bit 1 and bit 0  <u>Capture mode:</u> Unused. <u>Compare mode:</u> Unused. <u>PWM mode:</u> These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.
bit 3-0	<b>CCP1M3:CCP1M0:</b> Enhanced CCP Mode Select bits 0000 = Capture/Compare/PWM off (resets ECCP module) 0001 = Reserved 0010 = Compare mode, toggle output on match 0011 = Capture mode 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 CCP1 pin low; set output on compare match (set CCP1IF) 1001 = Compare mode, initialize CCP1 pin high; clear output on compare match (set CCP1IF) 1010 = Compare mode, generate software interrupt only; CCP1 pin reverts to I/O state 1011 = Compare mode, trigger special event (ECCP resets TMR1 or TMR3, sets CCP1IF bit) 1100 = PWM mode, P1A, P1C active-high; P1B, P1D active-high 1101 = PWM mode, P1A, P1C active-high; P1B, P1D active-low 1110 = PWM mode, P1A, P1C active-low; P1B, P1D active-high 1111 = PWM mode, P1A, P1C active-low; P1B, P1D active-low

# PIC18F2525/2620/4525/4620

In addition to the expanded range of modes available through the CCP1CON and ECCP1AS registers, the ECCP module has an additional register associated with Enhanced PWM operation and auto-shutdown features; it is:

- PWM1CON (PWM Configuration)

## 16.1 ECCP Outputs and Configuration

The Enhanced CCP module may have up to four PWM outputs, depending on the selected operating mode. These outputs, designated P1A through P1D, are multiplexed with I/O pins on PORTC and PORTD. The outputs that are active depend on the CCP operating mode selected. The pin assignments are summarized in Table 16-1.

To configure the I/O pins as PWM outputs, the proper PWM mode must be selected by setting the P1M1:P1M0 and CCP1M3:CCP1M0 bits. The appropriate TRISC and TRISD direction bits for the port pins must also be set as outputs.

### 16.1.1 ECCP MODULES AND TIMER RESOURCES

Like the standard CCP modules, the ECCP module can utilize Timers 1, 2 or 3, depending on the mode selected. Timer1 and Timer3 are available for modules in Capture or Compare modes, while Timer2 is available for modules in PWM mode. Interactions between the standard and Enhanced CCP modules are identical to those described for standard CCP modules. Additional details on timer resources are provided in **Section 15.1.1 “CCP Modules and Timer Resources”**.

## 16.2 Capture and Compare Modes

Except for the operation of the Special Event Trigger discussed below, the Capture and Compare modes of the ECCP module are identical in operation to that of CCP2. These are discussed in detail in **Section 15.2 “Capture Mode”** and **Section 15.3 “Compare Mode”**. No changes are required when moving between 28-pin and 40/44-pin devices.

### 16.2.1 SPECIAL EVENT TRIGGER

The Special Event Trigger output of ECCP1 resets the TMR1 or TMR3 register pair, depending on which timer resource is currently selected. This allows the CCP1 register to effectively be a 16-bit programmable period register for Timer1 or Timer3.

## 16.3 Standard PWM Mode

When configured in Single Output mode, the ECCP module functions identically to the standard CCP module in PWM mode, as described in **Section 15.4 “PWM Mode”**. This is also sometimes referred to as “Compatible CCP” mode, as in Table 16-1.

**Note:** When setting up single output PWM operations, users are free to use either of the processes described in **Section 15.4.4 “Setup for PWM Operation”** or **Section 16.4.9 “Setup for PWM Operation”**. The latter is more generic and will work for either single or multi-output PWM.

TABLE 16-1: PIN ASSIGNMENTS FOR VARIOUS ECCP1 MODES

ECCP Mode	CCP1CON Configuration	RC2	RD5	RD6	RD7
All 40/44-pin devices:					
Compatible CCP	00xx 11xx	CCP1	RD5/PSP5	RD6/PSP6	RD7/PSP7
Dual PWM	10xx 11xx	P1A	P1B	RD6/PSP6	RD7/PSP7
Quad PWM	x1xx 11xx	P1A	P1B	P1C	P1D

**Legend:** x = Don't care. Shaded cells indicate pin assignments not used by ECCP1 in a given mode.

## 16.4 Enhanced PWM Mode

The Enhanced PWM mode provides additional PWM output options for a broader range of control applications. The module is a backward compatible version of the standard CCP module and offers up to four outputs, designated P1A through P1D. Users are also able to select the polarity of the signal (either active-high or active-low). The module's output mode and polarity are configured by setting the P1M1:P1M0 and CCP1M3:CCP1M0 bits of the CCP1CON register.

Figure 16-1 shows a simplified block diagram of PWM operation. All control registers are double-buffered and are loaded at the beginning of a new PWM cycle (the period boundary when Timer2 resets) in order to prevent glitches on any of the outputs. The exception is the PWM Delay register, PWM1CON, which is loaded at either the duty cycle boundary or the period boundary (whichever comes first). Because of the buffering, the module waits until the assigned timer resets, instead of starting immediately. This means that Enhanced PWM waveforms do not exactly match the standard PWM waveforms, but are instead offset by one full instruction cycle (4 Tosc).

As before, the user must manually configure the appropriate TRIS bits for output.

### 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 equation.

#### EQUATION 16-1:

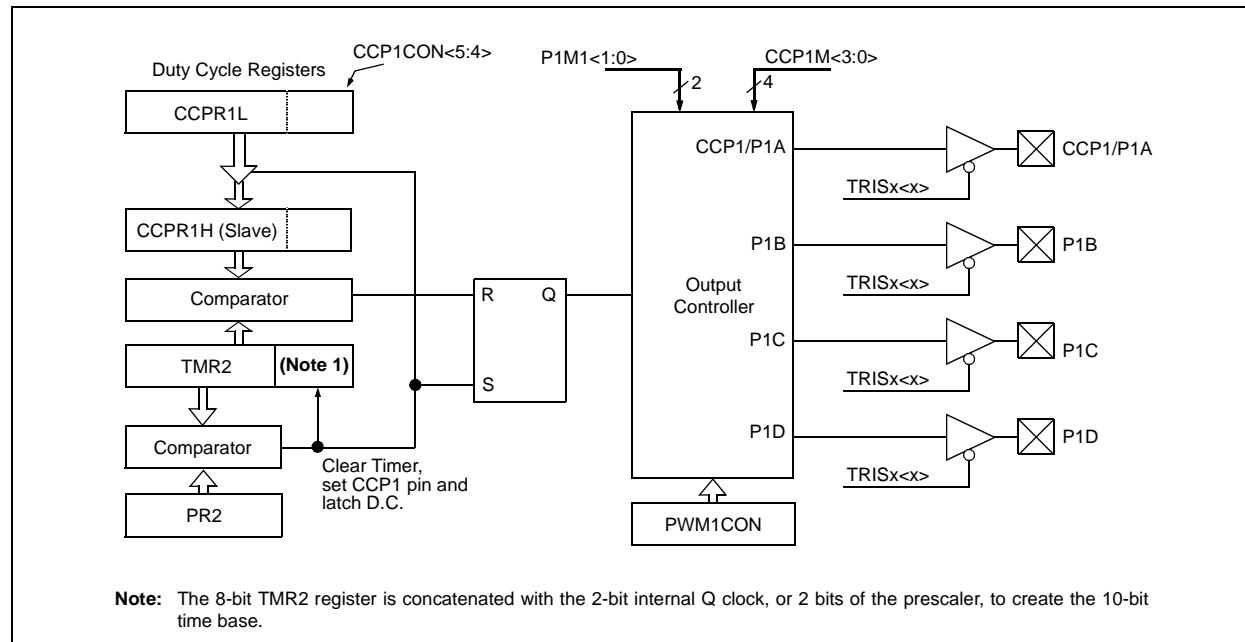
$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot TOSC \cdot (TMR2 \text{ Prescale Value})$$

PWM frequency is defined as  $1/[\text{PWM period}]$ . When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is copied from CCP1L into CCP1H

**Note:** The Timer2 postscaler (see **Section 13.0 “Timer2 Module”**) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

**FIGURE 16-1: SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODULE**



# PIC18F2525/2620/4525/4620

## 16.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCP1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCP1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCP1L:CCP1CON<5:4>. The PWM duty cycle is calculated by the following equation.

### EQUATION 16-2:

$$\text{PWM Duty Cycle} = (\text{CCP1L:CCP1CON<5:4>} \cdot \text{Tosc} \cdot (\text{TMR2 Prescale Value}))$$

CCP1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not copied into CCP1H until a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCP1H is a read-only register.

The CCP1H 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 CCP1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or two bits of the TMR2 prescaler, the CCP1 pin is cleared. The maximum PWM resolution (bits) for a given PWM frequency is given by the following equation.

### EQUATION 16-3:

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

**Note:** If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

## 16.4.3 PWM OUTPUT CONFIGURATIONS

The P1M1:P1M0 bits in the CCP1CON register allow one of four configurations:

- Single Output
- Half-Bridge Output
- Full-Bridge Output, Forward mode
- Full-Bridge Output, Reverse mode

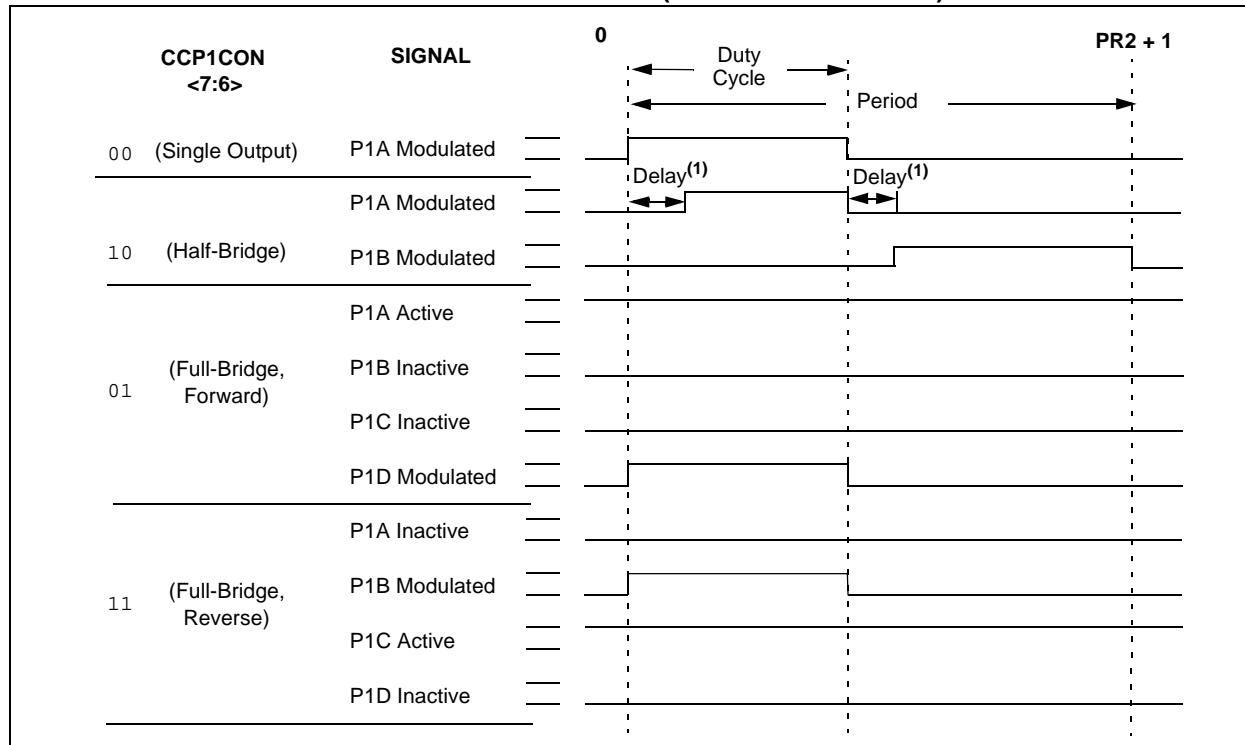
The Single Output mode is the standard PWM mode discussed in [Section 16.4 “Enhanced PWM Mode”](#). The Half-Bridge and Full-Bridge Output modes are covered in detail in the sections that follow.

The general relationship of the outputs in all configurations is summarized in Figure 16-2.

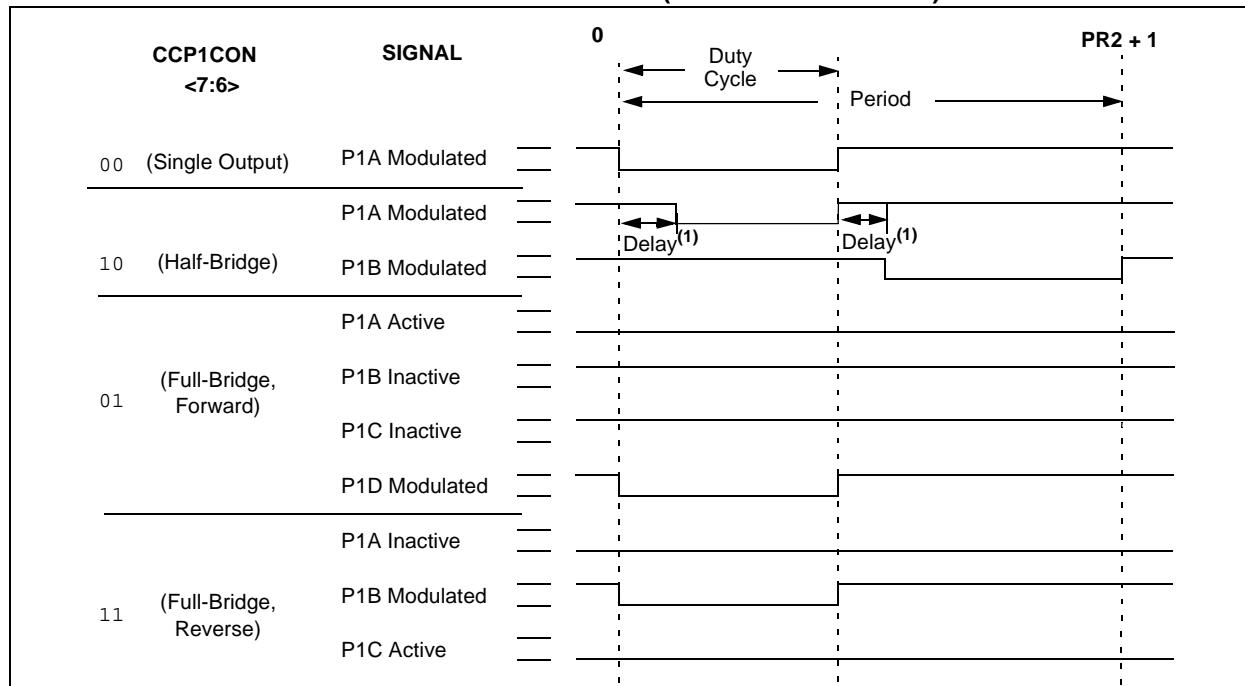
**TABLE 16-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz**

PWM Frequency	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	FFh	FFh	FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	10	10	10	8	7	6.58

**FIGURE 16-2: PWM OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)**



**FIGURE 16-3: PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)**



**Relationships:**

- Period =  $4 * Tosc * (PR2 + 1) * (TMR2 \text{ Prescale Value})$
- Duty Cycle =  $Tosc * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 \text{ Prescale Value})$
- Delay =  $4 * Tosc * (PWM1CON<6:0>)$

**Note 1:** Dead-band delay is programmed using the PWM1CON register (see **Section 16.4.6 “Programmable Dead-Band Delay”**).

# PIC18F2525/2620/4525/4620

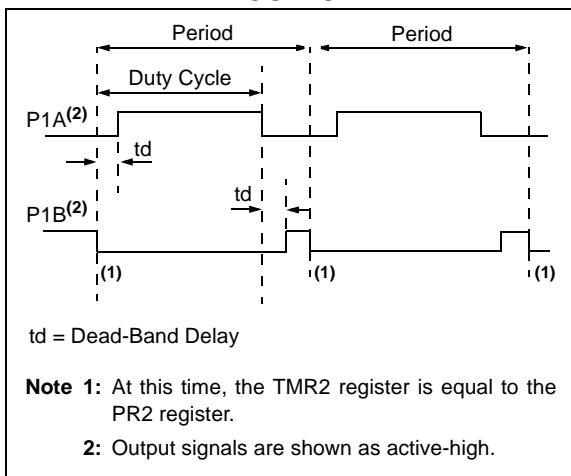
## 16.4.4 HALF-BRIDGE MODE

In the Half-Bridge Output mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the P1A pin, while the complementary PWM output signal is output on the P1B pin (Figure 16-4). This mode can be used for half-bridge applications, as shown in Figure 16-5, or for full-bridge applications where four power switches are being modulated with two PWM signals.

In Half-Bridge Output mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of bits, PDC6:PDC0, sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See **Section 16.4.6 “Programmable Dead-Band Delay”** for more details of the dead-band delay operations.

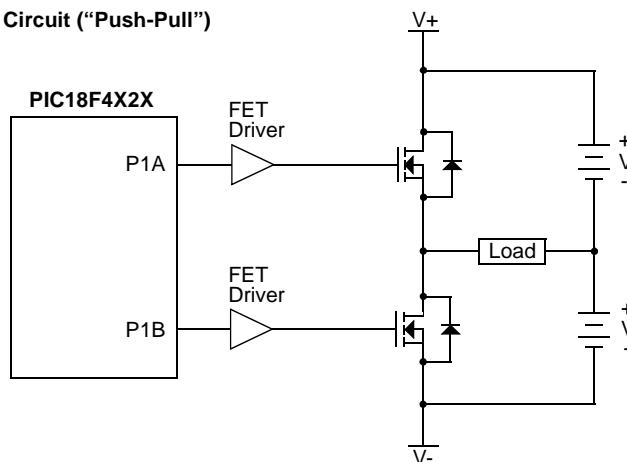
Since the P1A and P1B outputs are multiplexed with the PORTC<2> and PORTD<5> data latches, the TRISC<2> and TRISD<5> bits must be cleared to configure P1A and P1B as outputs.

**FIGURE 16-4: HALF-BRIDGE PWM OUTPUT**

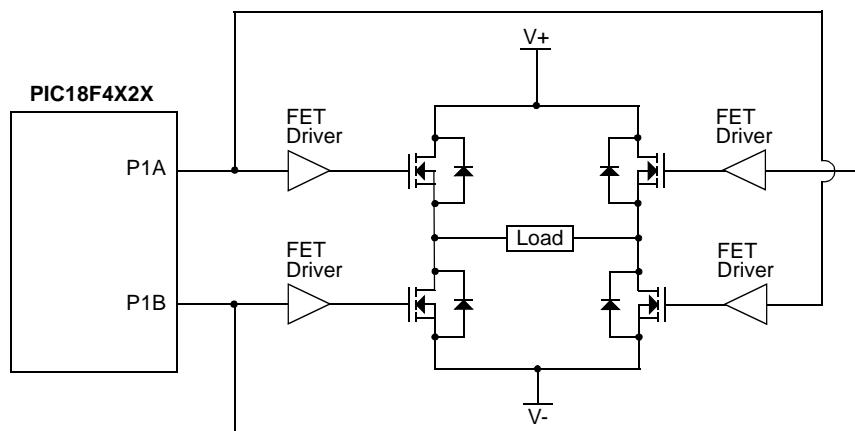


**FIGURE 16-5: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS**

Standard Half-Bridge Circuit (“Push-Pull”)



Half-Bridge Output Driving a Full-Bridge Circuit

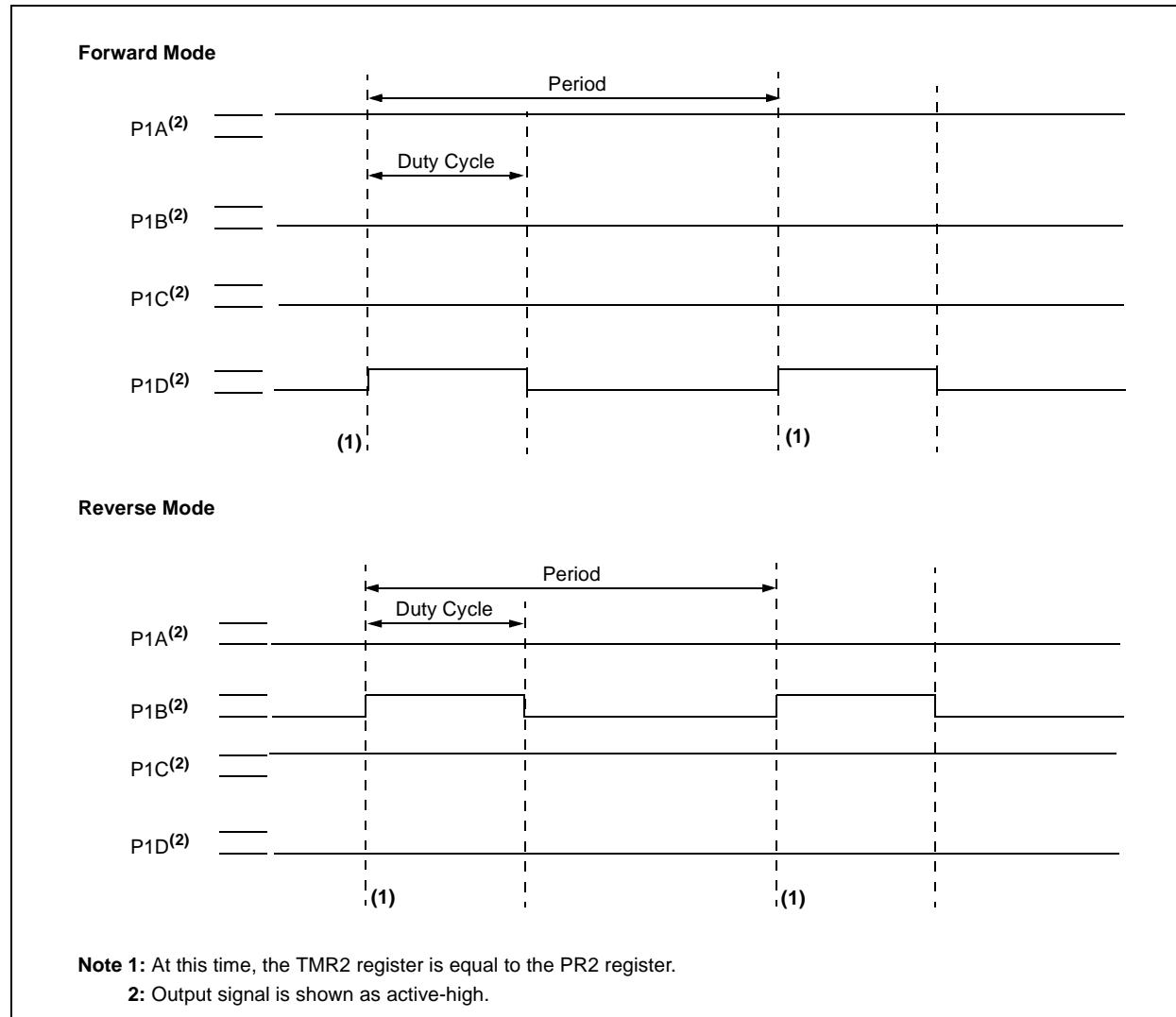


## 16.4.5 FULL-BRIDGE MODE

In Full-Bridge Output mode, four pins are used as outputs; however, only two outputs are active at a time. In the Forward mode, pin P1A is continuously active and pin P1D is modulated. In the Reverse mode, pin P1C is continuously active and pin P1B is modulated. These are illustrated in Figure 16-6.

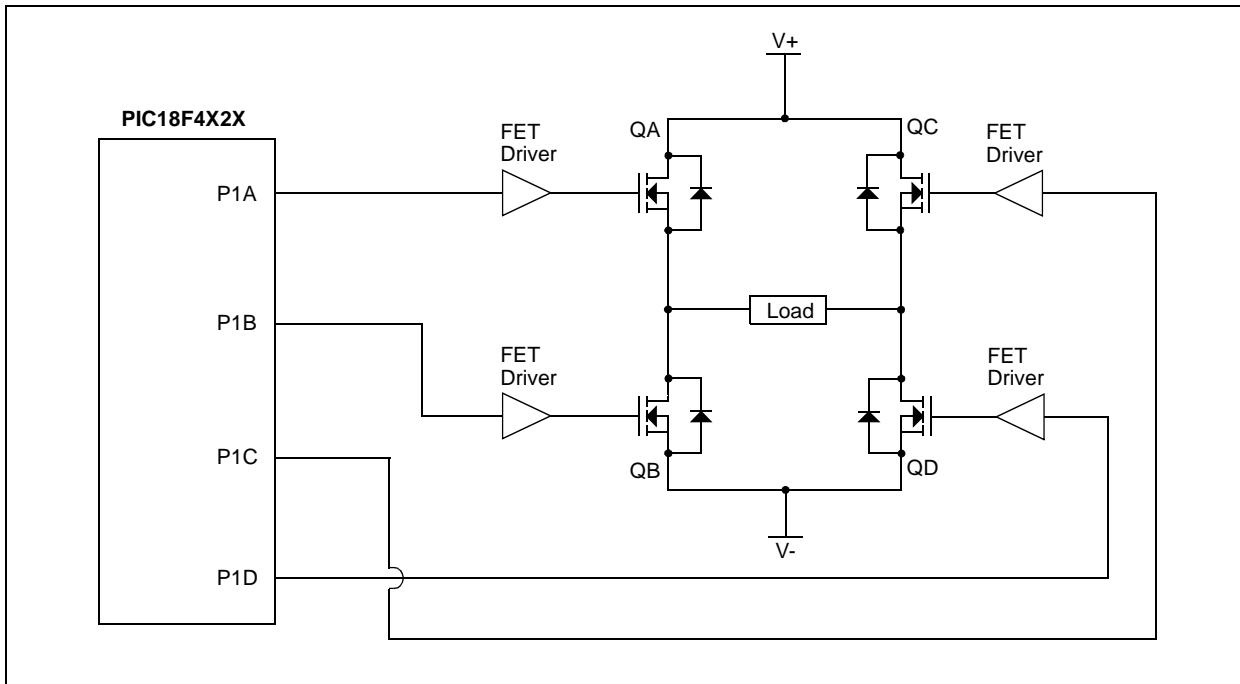
P1A, P1B, P1C and P1D outputs are multiplexed with the PORTC<2> and PORTD<7:5> data latches. The TRISC<2> and TRISD<7:5> bits must be cleared to make the P1A, P1B, P1C and P1D pins outputs.

**FIGURE 16-6: FULL-BRIDGE PWM OUTPUT**



# PIC18F2525/2620/4525/4620

FIGURE 16-7: EXAMPLE OF FULL-BRIDGE APPLICATION



#### 16.4.5.1 Direction Change in Full-Bridge Mode

In the Full-Bridge Output mode, the P1M1 bit in the CCP1CON register allows user to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will assume the new direction on the next PWM cycle.

Just before the end of the current PWM period, the modulated outputs (P1B and P1D) are placed in their inactive state, while the unmodulated outputs (P1A and P1C) are switched to drive in the opposite direction. This occurs in a time interval of  $4 \text{ Tosc} * (\text{Timer2 Prescale Value})$  before the next PWM period begins. The Timer2 prescaler will be either 1, 4 or 16, depending on the value of the T2CKPS1:T2CKPS0 bits (T2CON<1:0>). During the interval from the switch of the unmodulated outputs to the beginning of the next period, the modulated outputs (P1B and P1D) remain inactive. This relationship is shown in Figure 16-8.

Note that in the Full-Bridge Output mode, the CCP1 module does not provide any dead-band delay. In general, since only one output is modulated at all times, dead-band delay is not required. However, there is a situation where a dead-band delay might be required. This situation occurs when both of the following conditions are true:

1. The direction of the PWM output changes when the duty cycle of the output is at or near 100%.
2. The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

Figure 16-9 shows an example where the PWM direction changes from forward to reverse at a near 100% duty cycle. At time t1, the outputs P1A and P1D become inactive, while output P1C becomes active. In this example, since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current may flow through power devices, QC and QD (see Figure 16-7), for the duration of 't'. The same phenomenon will occur to power devices, QA and QB, for PWM direction change from reverse to forward.

If changing PWM direction at high duty cycle is required for an application, one of the following requirements must be met:

1. Reduce PWM for a PWM period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

Other options to prevent shoot-through current may exist.

FIGURE 16-8: PWM DIRECTION CHANGE

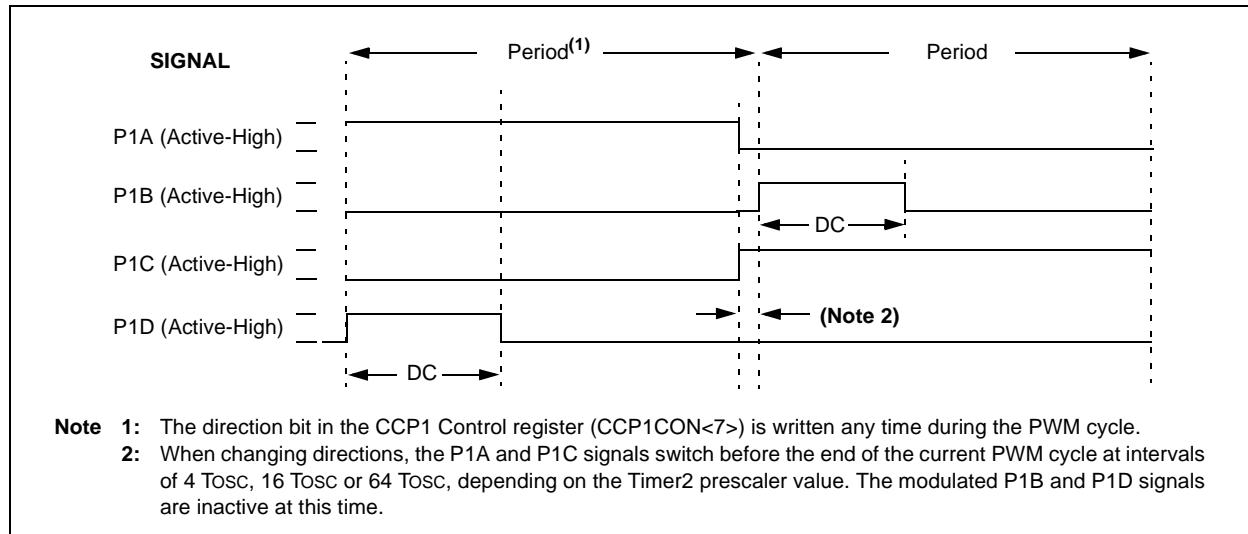
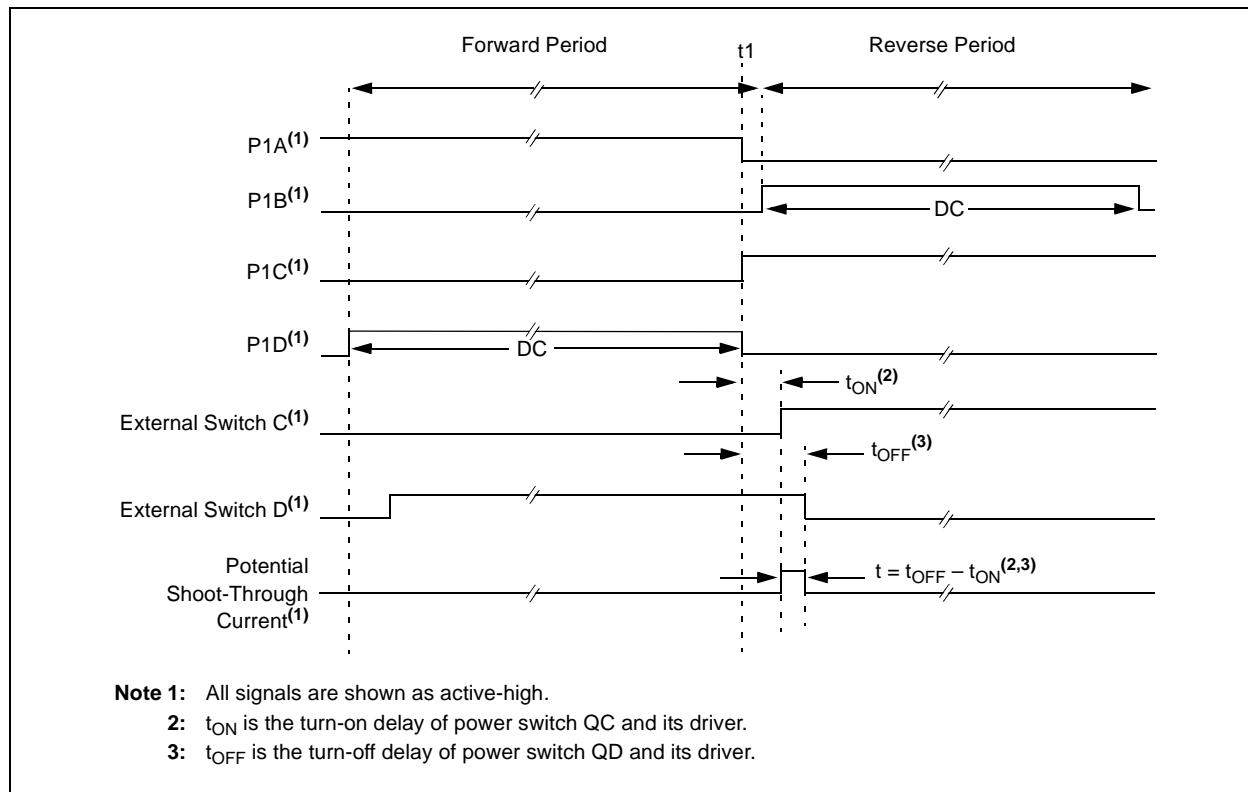


FIGURE 16-9: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



# PIC18F2525/2620/4525/4620

## 16.4.6 PROGRAMMABLE DEAD-BAND DELAY

**Note:** Programmable dead-band delay is not implemented in 28-pin devices with standard CCP modules.

In half-bridge applications where all power switches are modulated at the PWM frequency at all times, the power switches normally require more time to turn off than to turn on. If both the upper and lower power switches are switched at the same time (one turned on and the other turned off), both switches may be on for a short period of time until one switch completely turns off. During this brief interval, a very high current (*shoot-through current*) may flow through both power switches, shorting the bridge supply. To avoid this potentially destructive shoot-through current from flowing during switching, turning on either of the power switches is normally delayed to allow the other switch to completely turn off.

In the Half-Bridge Output mode, a digitally programmable dead-band delay is available to avoid shoot-through current from destroying the bridge power switches. The delay occurs at the signal transition from the nonactive state to the active state. See Figure 16-4 for illustration. Bits PDC6:PDC0 of the PWM1CON register (Register 16-2) set the delay period in terms of microcontroller instruction cycles (TCY or 4 Tosc). These bits are not available on 28-pin devices as the standard CCP module does not support half-bridge operation.

## 16.4.7 ENHANCED PWM AUTO-SHUTDOWN

When the CCP1 is programmed for any of the Enhanced PWM modes, the active output pins may be configured for auto-shutdown. Auto-shutdown immediately places the Enhanced PWM output pins into a defined shutdown state when a shutdown event occurs.

## REGISTER 16-2: PWM1CON: PWM CONFIGURATION REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PRSEN	PDC6 <sup>(1)</sup>	PDC5 <sup>(1)</sup>	PDC4 <sup>(1)</sup>	PDC3 <sup>(1)</sup>	PDC2 <sup>(1)</sup>	PDC1 <sup>(1)</sup>	PDC0 <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

**PRSEN:** PWM Restart Enable bit

1 = Upon auto-shutdown, the ECCPASE bit clears automatically once the shutdown event goes away; the PWM restarts automatically

0 = Upon auto-shutdown, ECCPASE must be cleared in software to restart the PWM

bit 6-0

**PDC6:PDC0:** PWM Delay Count bits<sup>(1)</sup>

Delay time, in number of Fosc/4 (4 \* Tosc) cycles, between the scheduled and actual time for a PWM signal to transition to active.

**Note 1:** Unimplemented on 28-pin devices; bits read as '0'.

A shutdown event can be caused by either of the comparator modules, a low level on the Fault input pin (FLT0) or any combination of these three sources. The comparators may be used to monitor a voltage input proportional to a current being monitored in the bridge circuit. If the voltage exceeds a threshold, the comparator switches state and triggers a shutdown. Alternatively, a low digital signal on FLT0 can also trigger a shutdown. The auto-shutdown feature can be disabled by not selecting any auto-shutdown sources. The auto-shutdown sources to be used are selected using the ECCPAS2:ECCPAS0 bits (ECCP1AS<6:4>).

When a shutdown occurs, the output pins are asynchronously placed in their shutdown states, specified by the PSSAC1:PSSAC0 and PSSBD1:PSSBD0 bits (ECCP1AS<3:0>). Each pin pair (P1A/P1C and P1B/P1D) may be set to drive high, drive low or be tri-stated (not driving). The ECCPASE bit (ECCP1AS<7>) is also set to hold the Enhanced PWM outputs in their shutdown states.

The ECCPASE bit is set by hardware when a shutdown event occurs. If automatic restarts are not enabled, the ECCPASE bit is cleared by firmware when the cause of the shutdown clears. If automatic restarts are enabled, the ECCPASE bit is automatically cleared when the cause of the auto-shutdown has cleared.

If the ECCPASE bit is set when a PWM period begins, the PWM outputs remain in their shutdown state for that entire PWM period. When the ECCPASE bit is cleared, the PWM outputs will return to normal operation at the beginning of the next PWM period.

**Note:** Writing to the ECCPASE bit is disabled while a shutdown condition is active.

# PIC18F2525/2620/4525/4620

## REGISTER 16-3: ECCP1AS: ECCP AUTO-SHUTDOWN 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
ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1 <sup>(1)</sup>	PSSBD0 <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      **ECCPASE:** ECCP Auto-Shutdown Event Status bit  
1 = A shutdown event has occurred; ECCP outputs are in shutdown state  
0 = ECCP outputs are operating
- bit 6-4     **ECCPAS2:ECCPAS0:** ECCP Auto-Shutdown Source Select bits  
111 = FLT0 or Comparator 1 or Comparator 2  
110 = FLT0 or Comparator 2  
101 = FLT0 or Comparator 1  
100 = FLT0  
011 = Either Comparator 1 or 2  
010 = Comparator 2 output  
001 = Comparator 1 output  
000 = Auto-shutdown is disabled
- bit 3-2     **PSSAC1:PSSAC0:** Pins A and C Shutdown State Control bits  
1x = Pins A and C are tri-state (40/44-pin devices);  
      PWM output is tri-state (28-pin devices)  
01 = Drive Pins A and C to '1'  
00 = Drive Pins A and C to '0'
- bit 1-0     **PSSBD1:PSSBD0:** Pins B and D Shutdown State Control bits<sup>(1)</sup>  
1x = Pins B and D tri-state  
01 = Drive Pins B and D to '1'  
00 = Drive Pins B and D to '0'

**Note 1:** Unimplemented on 28-pin devices; bits read as '0'.

## 16.4.7.1 Auto-Shutdown and Automatic Restart

The auto-shutdown feature can be configured to allow automatic restarts of the module following a shutdown event. This is enabled by setting the PRSEN bit of the PWM1CON register (PWM1CON<7>).

In Shutdown mode with PRSEN = 1 (Figure 16-10), the ECCPASE bit will remain set for as long as the cause of the shutdown continues. When the shutdown condition clears, the ECCP1ASE bit is cleared. If PRSEN = 0 (Figure 16-11), once a shutdown condition occurs, the ECCPASE bit will remain set until it is cleared by firmware. Once ECCPASE is cleared, the Enhanced PWM will resume at the beginning of the next PWM period.

**Note:** Writing to the ECCPASE bit is disabled while a shutdown condition is active.

Independent of the PRSEN bit setting, if the auto-shutdown source is one of the comparators, the shutdown condition is a level. The ECCPASE bit cannot be cleared as long as the cause of the shutdown persists.

The Auto-Shutdown mode can be forced by writing a '1' to the ECCPASE bit.

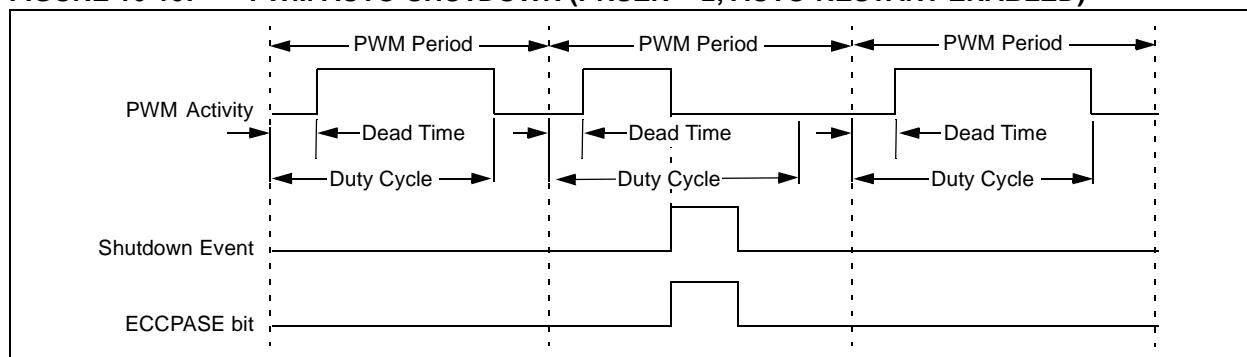
## 16.4.8 START-UP CONSIDERATIONS

When the ECCP module is used in the PWM mode, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins. When the microcontroller is released from Reset, all of the I/O pins are in the high-impedance state. The external circuits must keep the power switch devices in the off state until the microcontroller drives the I/O pins with the proper signal levels, or activates the PWM output(s).

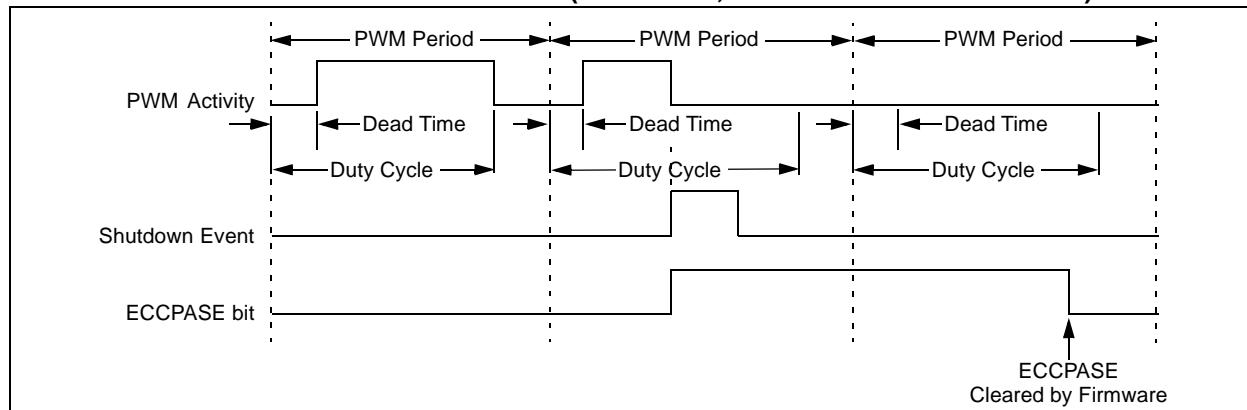
The CCP1M1:CCP1M0 bits (CCP1CON<1:0>) allow the user to choose whether the PWM output signals are active-high or active-low for each pair of PWM output pins (P1A/P1C and P1B/P1D). The PWM output polarities must be selected before the PWM pins are configured as outputs. Changing the polarity configuration while the PWM pins are configured as outputs is not recommended, since it may result in damage to the application circuits.

The P1A, P1B, P1C and P1D output latches may not be in the proper states when the PWM module is initialized. Enabling the PWM pins for output at the same time as the ECCP module may cause damage to the application circuit. The ECCP module must be enabled in the proper output mode and complete a full PWM cycle before configuring the PWM pins as outputs. The completion of a full PWM cycle is indicated by the TMR2IF bit being set as the second PWM period begins.

**FIGURE 16-10: PWM AUTO-SHUTDOWN (PRSEN = 1, AUTO-RESTART ENABLED)**



**FIGURE 16-11: PWM AUTO-SHUTDOWN (PRSEN = 0, AUTO-RESTART DISABLED)**



## 16.4.9 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the ECCP module for PWM operation:

1. Configure the PWM pins, P1A and P1B (and P1C and P1D, if used), as inputs by setting the corresponding TRIS bits.
2. Set the PWM period by loading the PR2 register.
3. If auto-shutdown is required, do the following:
  - Disable auto-shutdown (ECCP1AS = 0)
  - Configure source (FLT0, Comparator 1 or Comparator 2)
  - Wait for non-shutdown condition
4. Configure the ECCP module for the desired PWM mode and configuration by loading the CCP1CON register with the appropriate values:
  - Select one of the available output configurations and direction with the P1M1:P1M0 bits.
  - Select the polarities of the PWM output signals with the CCP1M3:CCP1M0 bits.
5. Set the PWM duty cycle by loading the CCP1L register and CCP1CON<5:4> bits.
6. For Half-Bridge Output mode, set the dead-band delay by loading PWM1CON<6:0> with the appropriate value.
7. If auto-shutdown operation is required, load the ECCP1AS register:
  - Select the auto-shutdown sources using the ECCPAS2:ECCPAS0 bits.
  - Select the shutdown states of the PWM output pins using the PSSAC1:PSSAC0 and PSSBD1:PSSBD0 bits.
  - Set the ECCPASE bit (ECCP1AS<7>).
  - Configure the comparators using the CMCON register.
  - Configure the comparator inputs as analog inputs.
8. If auto-restart operation is required, set the PRSEN bit (PWM1CON<7>).
9. Configure and start TMR2:
  - Clear the TMR2 interrupt flag bit by clearing the TMR2IF bit (PIR1<1>).
  - Set the TMR2 prescale value by loading the T2CKPS bits (T2CON<1:0>).
  - Enable Timer2 by setting the TMR2ON bit (T2CON<2>).
10. Enable PWM outputs after a new PWM cycle has started:
  - Wait until TMRx overflows (TMRxIF bit is set).
  - Enable the CCP1/P1A, P1B, P1C and/or P1D pin outputs by clearing the respective TRIS bits.
  - Clear the ECCPASE bit (ECCP1AS<7>).

## 16.4.10 OPERATION IN POWER-MANAGED MODES

In Sleep mode, all clock sources are disabled. Timer2 will not increment and the state of the module will not change. If the ECCP pin is driving a value, it will continue to drive that value. When the device wakes up, it will continue from this state. If Two-Speed Start-ups are enabled, the initial start-up frequency from INTOSC and the postscaler may not be stable immediately.

In PRI\_IDLE mode, the primary clock will continue to clock the ECCP module without change. In all other power-managed modes, the selected power-managed mode clock will clock Timer2. Other power-managed mode clocks will most likely be different than the primary clock frequency.

### 16.4.10.1 Operation with Fail-Safe Clock Monitor

If the Fail-Safe Clock Monitor is enabled, a clock failure will force the device into the power-managed RC\_RUN mode and the OSCFIF bit (PIR2<7>) will be set. The ECCP will then be clocked from the internal oscillator clock source, which may have a different clock frequency than the primary clock.

See the previous section for additional details.

### 16.4.11 EFFECTS OF A RESET

Both Power-on Reset and subsequent Resets will force all ports to Input mode and the CCP registers to their Reset states.

This forces the Enhanced CCP module to reset to a state compatible with the standard CCP module.

# PIC18F2525/2620/4525/4620

TABLE 16-3: REGISTERS ASSOCIATED WITH ECCP1 MODULE AND TIMER1 TO TIMER3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
RCON	IPEN	SBOREN <sup>(1)</sup>	—	RI	TO	PD	POR	BOR	48
PIR1	PSPIF <sup>(2)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(2)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(2)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52
TRISB	PORTB Data Direction Control Register								52
TRISC	PORTC Data Direction Control Register								52
TRISD	PORTD Data Direction Control Register								52
TMR1L	Timer1 Register Low Byte								50
TMR1H	Timer1 Register High Byte								50
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	50
TMR2	Timer2 Register								50
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	50
PR2	Timer2 Period Register								50
TMR3L	Timer3 Register Low Byte								51
TMR3H	Timer3 Register High Byte								51
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	51
CCPR1L	Capture/Compare/PWM Register 1 Low Byte								51
CCPR1H	Capture/Compare/PWM Register 1 High Byte								51
CCP1CON	P1M1 <sup>(2)</sup>	P1M0 <sup>(2)</sup>	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	51
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1 <sup>(2)</sup>	PSSBD0 <sup>(2)</sup>	51
PWM1CON	PRSEN	PDC6 <sup>(2)</sup>	PDC5 <sup>(2)</sup>	PDC4 <sup>(2)</sup>	PDC3 <sup>(2)</sup>	PDC2 <sup>(2)</sup>	PDC1 <sup>(2)</sup>	PDC0 <sup>(2)</sup>	51

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during ECCP operation.

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 4.4 "Brown-out Reset (BOR)"**.

**2:** These bits are unimplemented on 28-pin devices; always maintain these bits clear.

## 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 ( $I^2C$ )
  - Full Master mode
  - Slave mode (with general address call)

The  $I^2C$  interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

### 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  $I^2C$  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 SPI modes are supported. To accomplish communication, typically three pins are used:

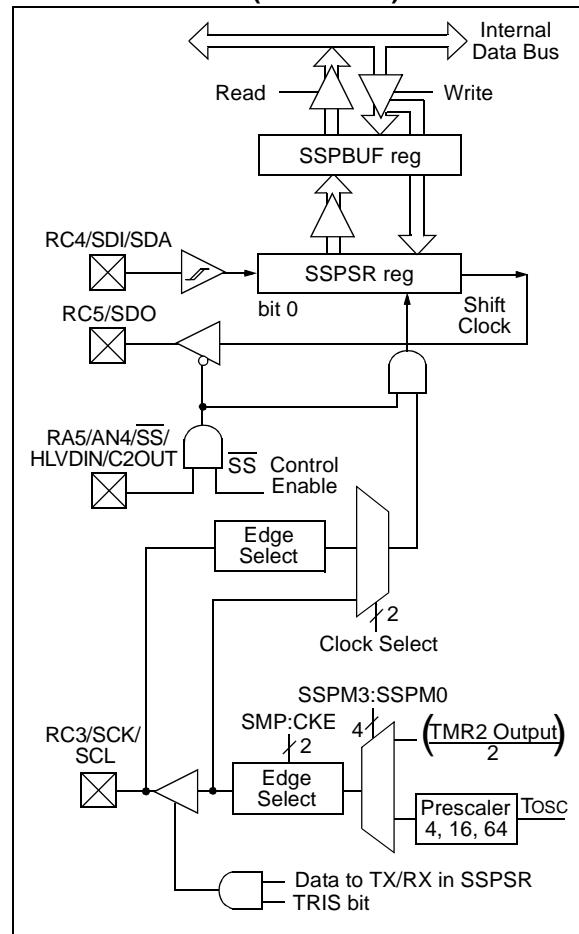
- Serial Data Out (SDO) – RC5/SDO
- Serial Data In (SDI) – RC4/SDI/SDA
- Serial Clock (SCK) – RC3/SCK/SCL

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

- Slave Select ( $\overline{SS}$ ) – RA5/AN4/ $\overline{SS}$ /HLVDIN/C2OUT

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

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



# PIC18F2525/2620/4525/4620

## 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 two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

## REGISTER 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 <sup>(1)</sup>	D/A	P	S	R/W	UA	BF
bit 7							

### 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>SMP:</b> Sample bit  <u>SPI Master mode:</u> 1 = Input data sampled at end of data output time 0 = Input data sampled at middle of data output time  <u>SPI Slave mode:</u> SMP must be cleared when SPI is used in Slave mode.
bit 6	<b>CKE:</b> SPI Clock Select bit <sup>(1)</sup>  1 = Transmit occurs on transition from active to Idle clock state 0 = Transmit occurs on transition from Idle to active clock state
bit 5	<b>D/A:</b> Data/Address bit Used in I <sup>2</sup> C mode only.
bit 4	<b>P:</b> 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	<b>S:</b> Start bit Used in I <sup>2</sup> C mode only.
bit 2	<b>R/W:</b> Read/Write Information bit Used in I <sup>2</sup> C mode only.
bit 1	<b>UA:</b> Update Address bit Used in I <sup>2</sup> C mode only.
bit 0	<b>BF:</b> Buffer Full Status bit (Receive mode only) 1 = Receive complete, SSPBUF is full 0 = Receive not complete, SSPBUF is empty

**Note 1:** Polarity of clock state is set by the CKP bit (SSPCON1<4>).

## 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	<b>WCOL:</b> Write Collision Detect bit 1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software) 0 = No collision
bit 6	<b>SSPOV:</b> Receive Overflow Indicator bit <sup>(1)</sup> <b>SPI Slave mode:</b> 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	<b>SSPEN:</b> Master Synchronous Serial Port Enable bit <sup>(2)</sup> 1 = Enables serial port and configures SCK, SDO, SDI and <u>SS</u> as serial port pins 0 = Disables serial port and configures these pins as I/O port pins
bit 4	<b>CKP:</b> 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	<b>SSPM3:SSPM0:</b> Master Synchronous Serial Port Mode Select bits <sup>(3)</sup> 0101 = SPI Slave mode, clock = SCK pin, <u>SS</u> pin control disabled, <u>SS</u> can be used as I/O pin 0100 = SPI Slave mode, clock = SCK pin, <u>SS</u> pin control enabled 0011 = SPI Master mode, clock = TMR2 output/2 0010 = SPI Master mode, clock = Fosc/64 0001 = SPI Master mode, clock = Fosc/16 0000 = SPI Master mode, clock = Fosc/4

- Note 1:** In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.
- 2:** When enabled, these pins must be properly configured as input or output.
- 3:** Bit combinations not specifically listed here are either reserved or implemented in I<sup>2</sup>C<sup>TM</sup> mode only.

## 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.

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

**Note:** To avoid lost data in Master mode, a read of the SSPBUF must be performed to clear the Buffer Full (BF) detect bit (SSPSTAT<0>) between each transmission.

## EXAMPLE 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
```

### 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 SS pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> bit cleared
- SCK (Master mode) must have TRISC<3> bit cleared
- SCK (Slave mode) must have TRISC<3> bit set
- SS must have TRISA<5> bit set

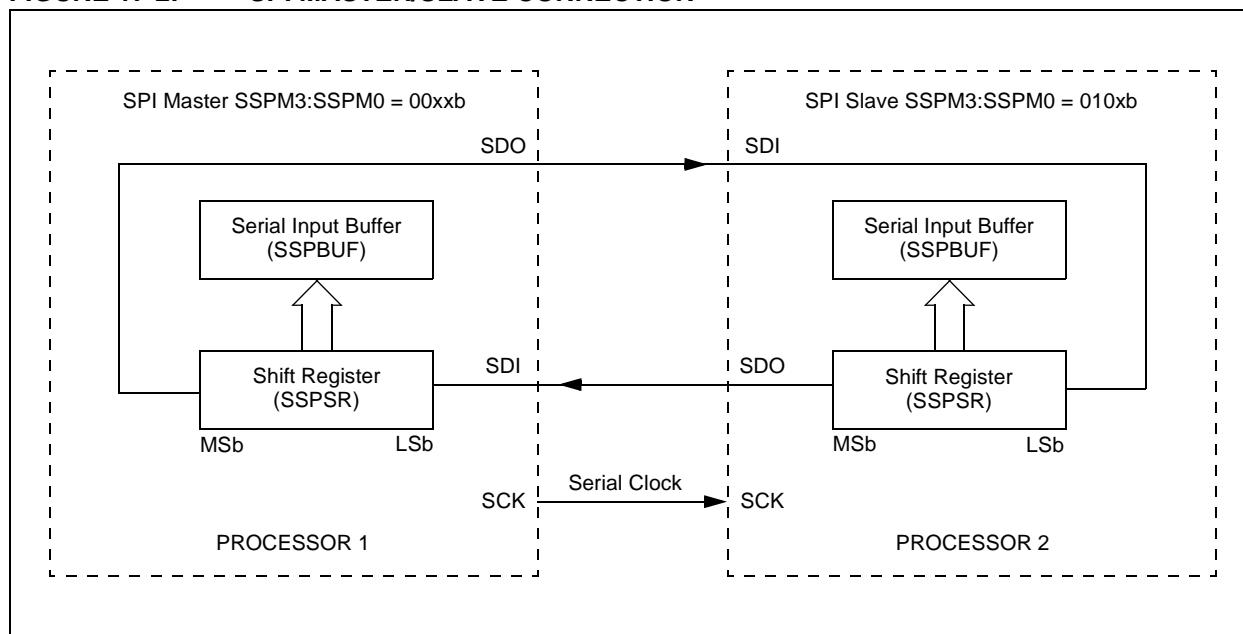
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

### 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**



## 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 operation 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.

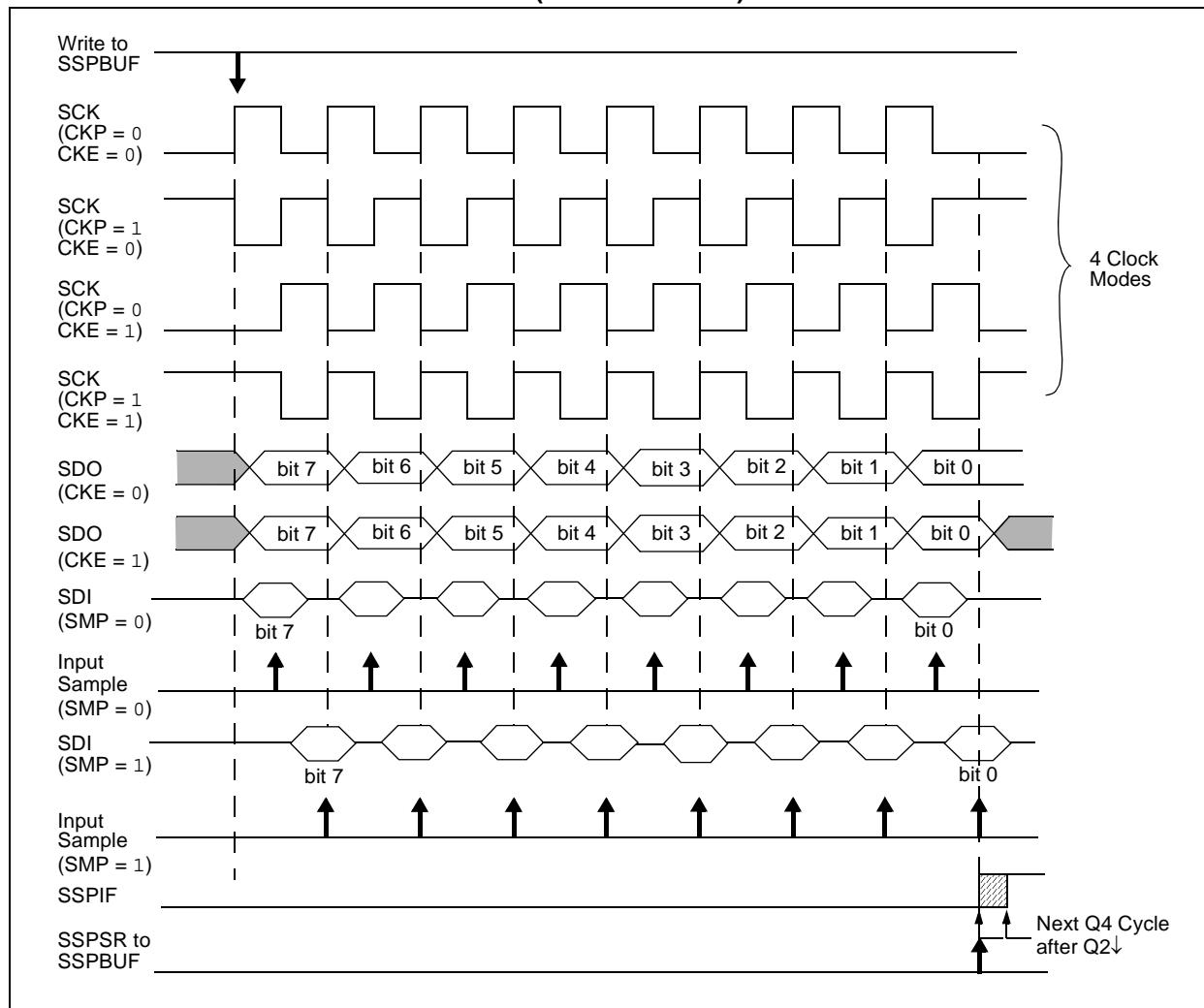
The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then, would give waveforms for SPI communication, as shown in Figure 17-3, Figure 17-5 and Figure 17-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or T<sub>CY</sub>)
- FOSC/16 (or 4 • T<sub>CY</sub>)
- FOSC/64 (or 16 • T<sub>CY</sub>)
- Timer2 output/2

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

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.

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit (SSPCON1<4>).

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. When a byte is received, the device will wake-up from Sleep.

### 17.3.7 SLAVE SELECT SYNCHRONIZATION

The SS pin allows a Synchronous Slave mode. The SPI operation must be in Slave mode with the SS pin control enabled (SSPCON1<3:0> = 04h). When the SS pin is low, transmission and reception are enabled and the

SDO pin is driven. When the 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.

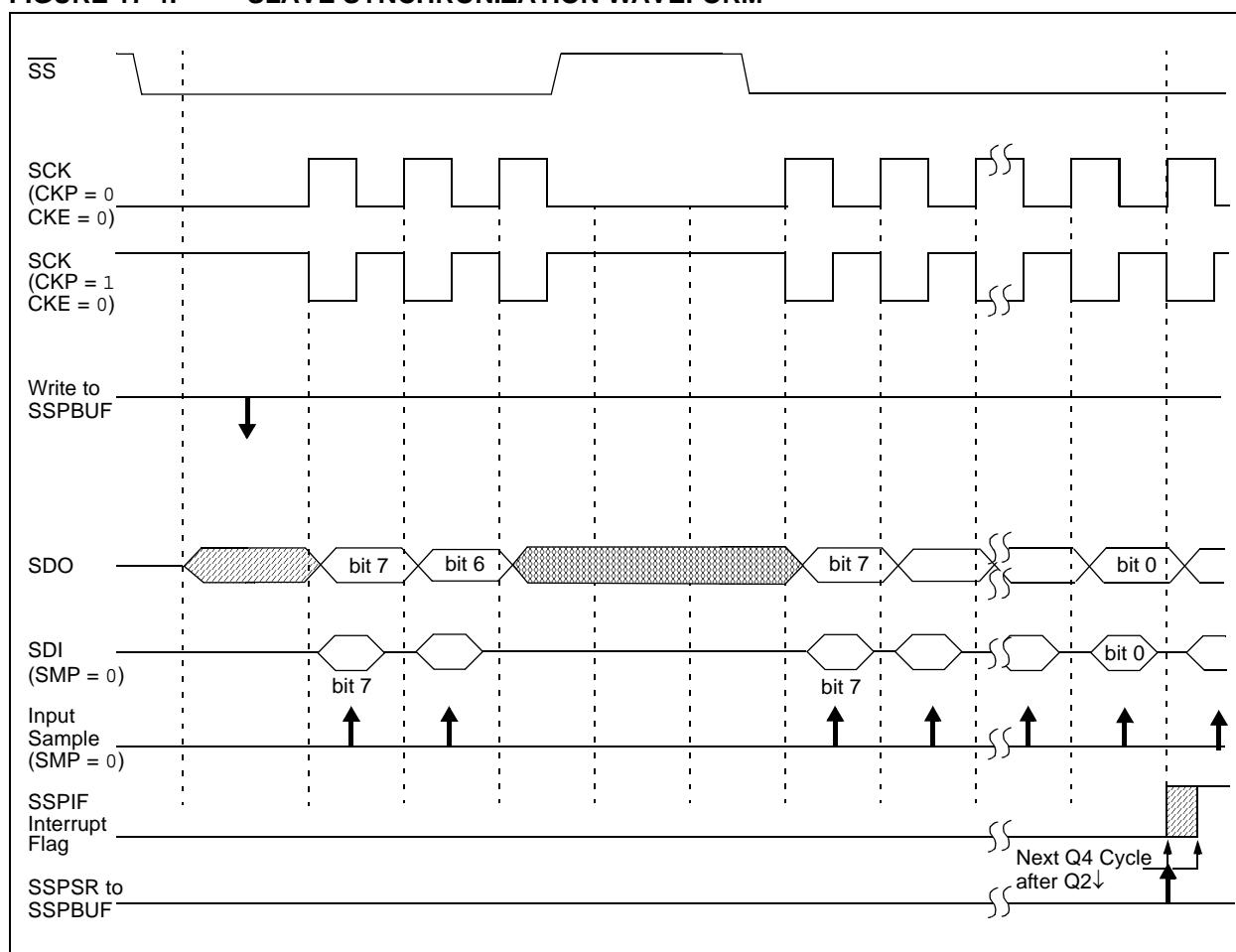
**Note 1:** When the SPI interface is in Slave mode with SS pin control enabled (SSPCON1<3:0> = 0100), the SPI module will reset if the SS pin is set to VDD.

**2:** If the SPI interface is used in Slave mode with CKE set, then the 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 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**



# PIC18F2525/2620/4525/4620

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

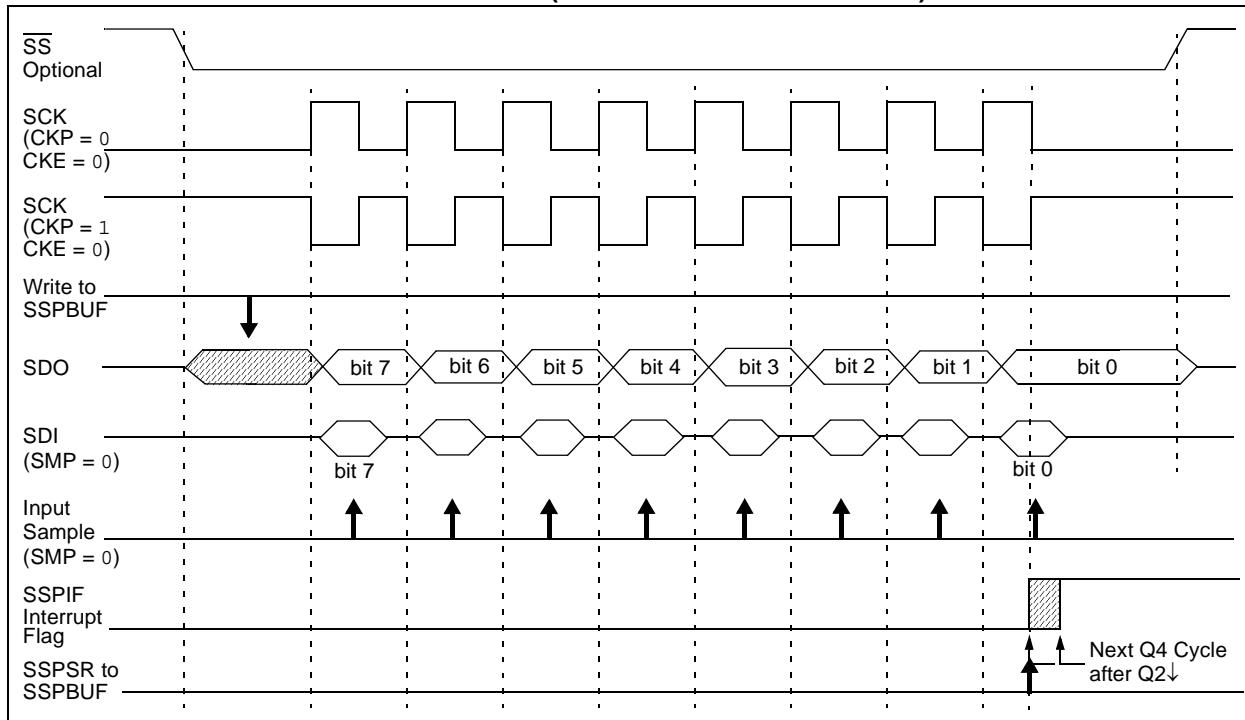
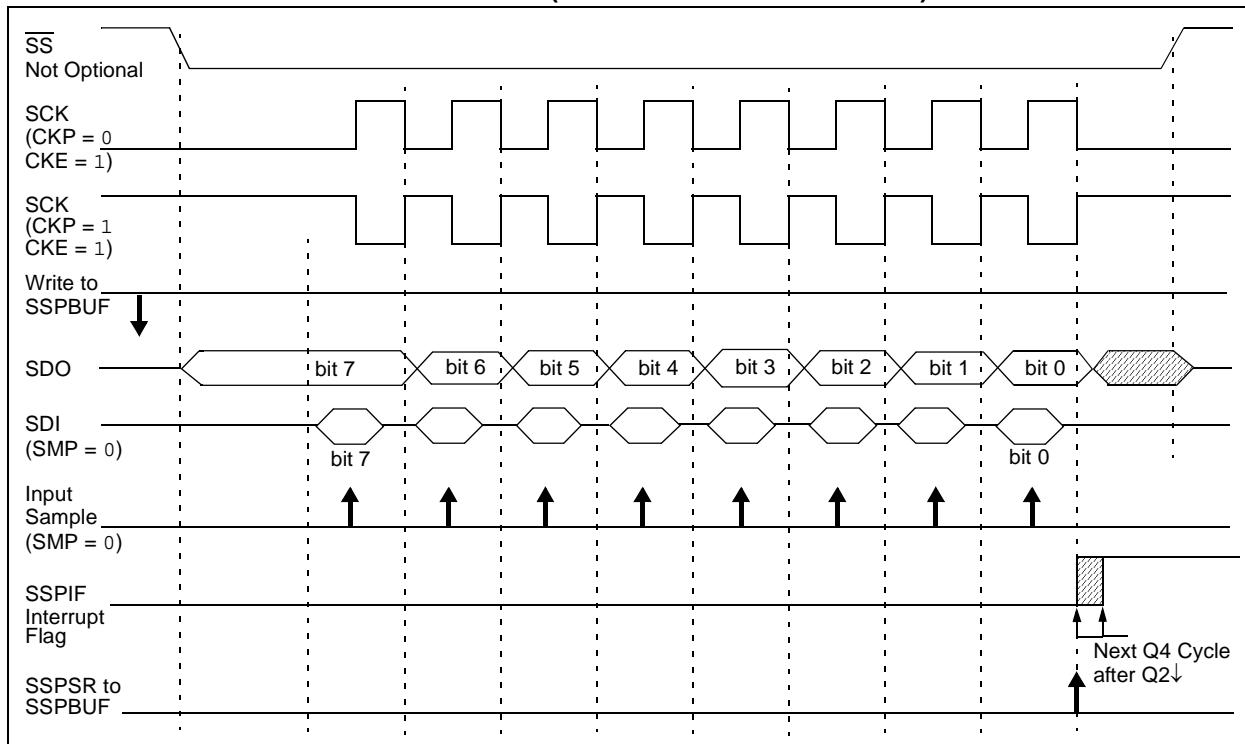


FIGURE 17-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



### 17.3.8 OPERATION IN POWER-MANAGED MODES

In SPI Master mode, module clocks may be operating at a different speed than when in full-power mode. In the case of Sleep mode, all clocks are halted.

In Idle 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 2.7 “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	TMROIE	INT0IE	RBIE	TMROIF	INT0IF	RBIF	49						
PIR1	PSP1IF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52						
PIE1	PSP1IE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52						
IPR1	PSP1IP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52						
TRISA	TRISA7 <sup>(2)</sup>	TRISA6 <sup>(2)</sup>	PORTA Data Direction Control Register						52						
TRISC	PORTC Data Direction Control Register								52						
SSPBUF	MSSP Receive Buffer/Transmit Register								50						
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	50						
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	50						

**Legend:** Shaded cells are not used by the MSSP in SPI mode.

**Note 1:** These bits are unimplemented on 28-pin devices and read as ‘0’.

**2:** PORTA<7:6> and their direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as ‘0’.

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

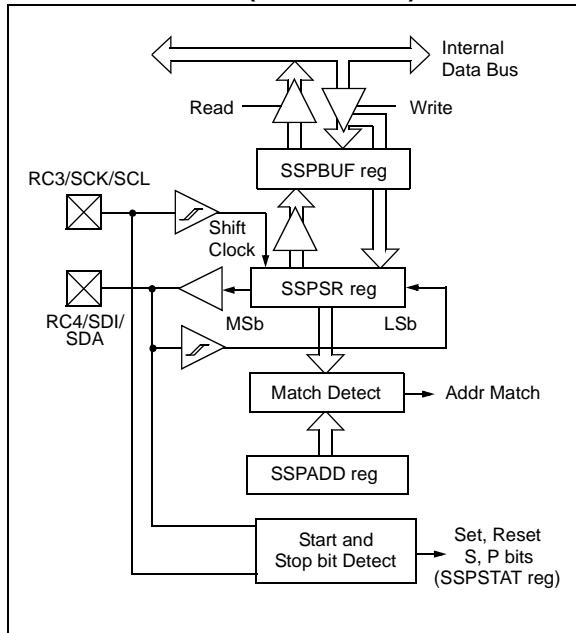
The MSSP module in I<sup>2</sup>C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit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

**FIGURE 17-7: MSSP BLOCK DIAGRAM (I<sup>2</sup>C<sup>TM</sup> 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 two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

SSPADD register holds the slave device address when the MSSP is configured in I<sup>2</sup>C Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPADD act as the Baud Rate Generator reload value.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

# PIC18F2525/2620/4525/4620

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

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-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						

### 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)

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 Information bit (I<sup>2</sup>C mode only)<sup>(2,3)</sup>

#### In Slave mode:

1 = Read

0 = Write

#### In Master mode:

1 = Transmit is in progress

0 = Transmit is not in progress

bit 1 **UA**: Update Address bit (10-Bit Slave mode only)

1 = Indicates that the user needs to update the address in the SSPADD register

0 = Address does not need to be updated

bit 0 **BF**: Buffer Full Status bit

#### In Transmit mode:

1 = SSPBUF is full

0 = SSPBUF is empty

#### In Receive mode:

1 = SSPBUF is full (does not include the ACK and Stop bits)

0 = SSPBUF is empty (does not include the ACK and Stop bits)

**Note 1:** This bit is cleared on Reset and when SSPEN is cleared.

**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 ACK bit.

**3:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Active mode.

# PIC18F2525/2620/4525/4620

## REGISTER 17-4: SSPCON1: MSSP CONTROL REGISTER 1 (I<sup>2</sup>C<sup>TM</sup> MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN <sup>(1)</sup>	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **WCOL:** Write Collision Detect bit

#### In Master Transmit mode:

1 = A write to the SSPBUF register was attempted while the I<sup>2</sup>C conditions were not valid for a transmission to be started (must be cleared in software)

0 = No collision

#### In Slave Transmit mode:

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

0 = No collision

#### In Receive mode (Master or Slave modes):

This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

#### In Receive mode:

1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared in software)

0 = No overflow

#### In Transmit mode:

This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit<sup>(1)</sup>

1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCK Release Control bit

#### In Slave mode:

1 = Releases clock

0 = Holds clock low (clock stretch), used to ensure data setup time

#### In Master mode:

Unused in this mode.

bit 3-0 **SSPM3:SSPM0:** Master Synchronous Serial Port Mode Select bits<sup>(2)</sup>

1111 = I<sup>2</sup>C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

1110 = I<sup>2</sup>C Slave mode, 7-bit address with Start and Stop bit interrupts enabled

1011 = I<sup>2</sup>C Firmware Controlled Master mode (Slave Idle)

1000 = I<sup>2</sup>C Master mode, clock = Fosc/(4 \* (SSPADD + 1))

0111 = I<sup>2</sup>C Slave mode, 10-bit address

0110 = I<sup>2</sup>C Slave mode, 7-bit address

Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

**Note 1:** When enabled, the SDA and SCL pins must be properly configured as inputs or outputs.

## REGISTER 17-5: SSPCON2: MSSP CONTROL REGISTER 2 (I<sup>2</sup>C<sup>TM</sup> 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>(2)</sup>	ACKEN <sup>(1)</sup>	RCEN <sup>(1)</sup>	PEN <sup>(1)</sup>	RSEN <sup>(1)</sup>	SEN <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	<b>GCEN:</b> General Call Enable bit (Slave mode only) 1 = Enables interrupt when a general call address (0000h) is received in the SSPSR 0 = General call address disabled.
bit 6	<b>ACKSTAT:</b> Acknowledge Status bit (Master Transmit mode only) 1 = Acknowledge was not received from slave 0 = Acknowledge was received from slave
bit 5	<b>ACKDT:</b> Acknowledge Data bit (Master Receive mode only) <sup>(2)</sup> 1 = Not Acknowledge 0 = Acknowledge
bit 4	<b>ACKEN:</b> Acknowledge Sequence Enable bit (Master Receive mode only) <sup>(1)</sup> 1 = Initiates Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware. 0 = Acknowledge sequence Idle
bit 3	<b>RCEN:</b> Receive Enable bit (Master mode only) <sup>(1)</sup> 1 = Enables Receive mode for I <sup>2</sup> C 0 = Receive Idle
bit 2	<b>PEN:</b> Stop Condition Enable bit (Master mode only) <sup>(1)</sup> 1 = Initiates Stop condition on SDA and SCL pins. Automatically cleared by hardware. 0 = Stop condition Idle
bit 1	<b>RSEN:</b> Repeated Start Condition Enable bit (Master mode only) <sup>(1)</sup> 1 = Initiates Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware. 0 = Repeated Start condition Idle
bit 0	<b>SEN:</b> Start Condition Enable/Stretch Enable bit <sup>(1)</sup> <u>In Master mode:</u> 1 = Initiates Start condition on SDA and SCL pins. Automatically cleared by hardware. 0 = Start condition Idle <u>In Slave mode:</u> 1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled) 0 = Clock stretching is disabled

**Note 1:** For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I<sup>2</sup>C module is not in the Idle mode, these bits may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

**2:** Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

## 17.4.2 OPERATION

The MSSP module functions are enabled by setting MSSP Enable bit, SSPEN (SSPCON1<5>).

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

- I<sup>2</sup>C Master mode clock
- 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 (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 ACK pulse:

- The Buffer Full bit, BF (SSPSTAT<0>), was set before the transfer was received.
- The overflow bit, SSPOV (SSPCON1<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  $\overline{\text{ACK}}$  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 (bits, SSPIF, BF and UA, are set).
5. Update the SSPADD register with the first (high) byte of address. If match releases SCL line, this will clear bit, UA.
6. Read the SSPBUF register (clears bit, BF) and clear flag bit, SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of address (bits, SSPIF and BF, are set).
9. Read the SSPBUF register (clears bit, BF) and clear flag bit, SSPIF.

### 17.4.3.2 Reception

When the R/W bit of the address byte is clear and an address match occurs, the 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 (ACK).

When the address byte overflow condition exists, then the no Acknowledge (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 detail.

### 17.4.3.3 Transmission

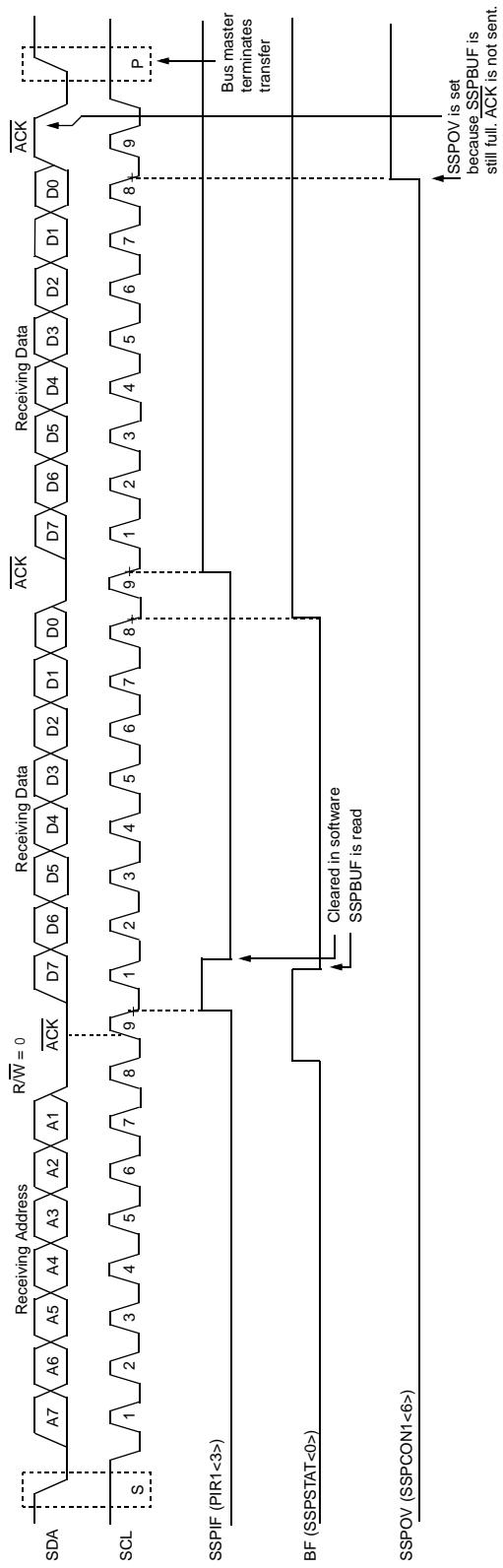
When the R/W bit of the incoming address byte is set and an address match occurs, the R/W bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The ACK pulse will be sent on the ninth bit and the RC3/SCK/SCL pin is held low regardless of SEN (see **Section 17.4.4 “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 eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time (Figure 17-9).

The 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 ACK), then the data transfer is complete. In this case, when the ACK is latched by the slave, the slave logic is reset and the slave monitors for another occurrence of the Start bit. If the SDA line was low (ACK), the next transmit data must be loaded into the SSPBUF register. Again, the RC3/SCK/SCL pin 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.

## **PIC18F2525/2620/4525/4620**

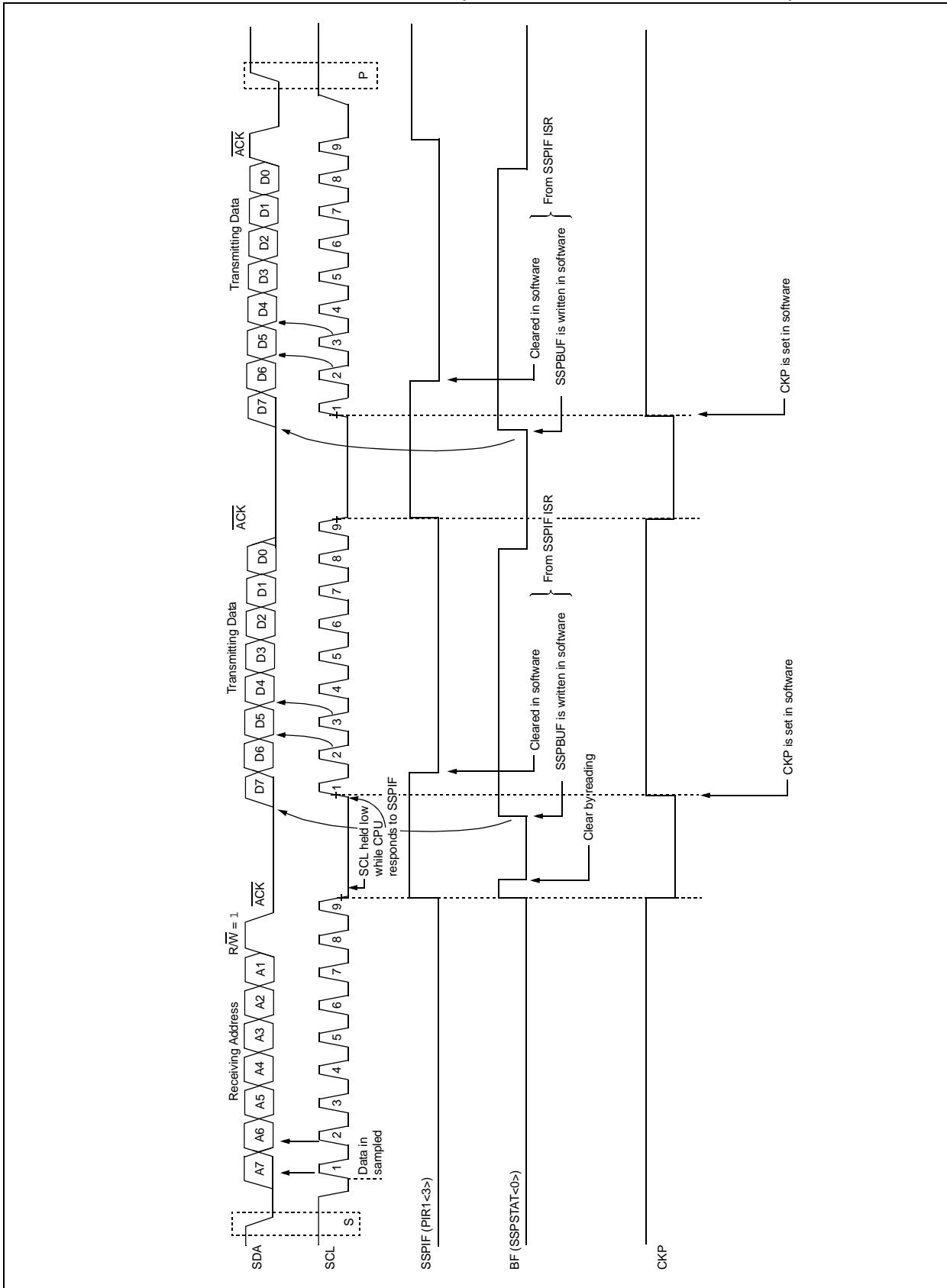
**FIGURE 17-8: I<sup>2</sup>C<sup>TM</sup> SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)**



---

CKP (CKP does not reset to '0' when SEN = 0)

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



## **PIC18F2525/2620/4525/4620**

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

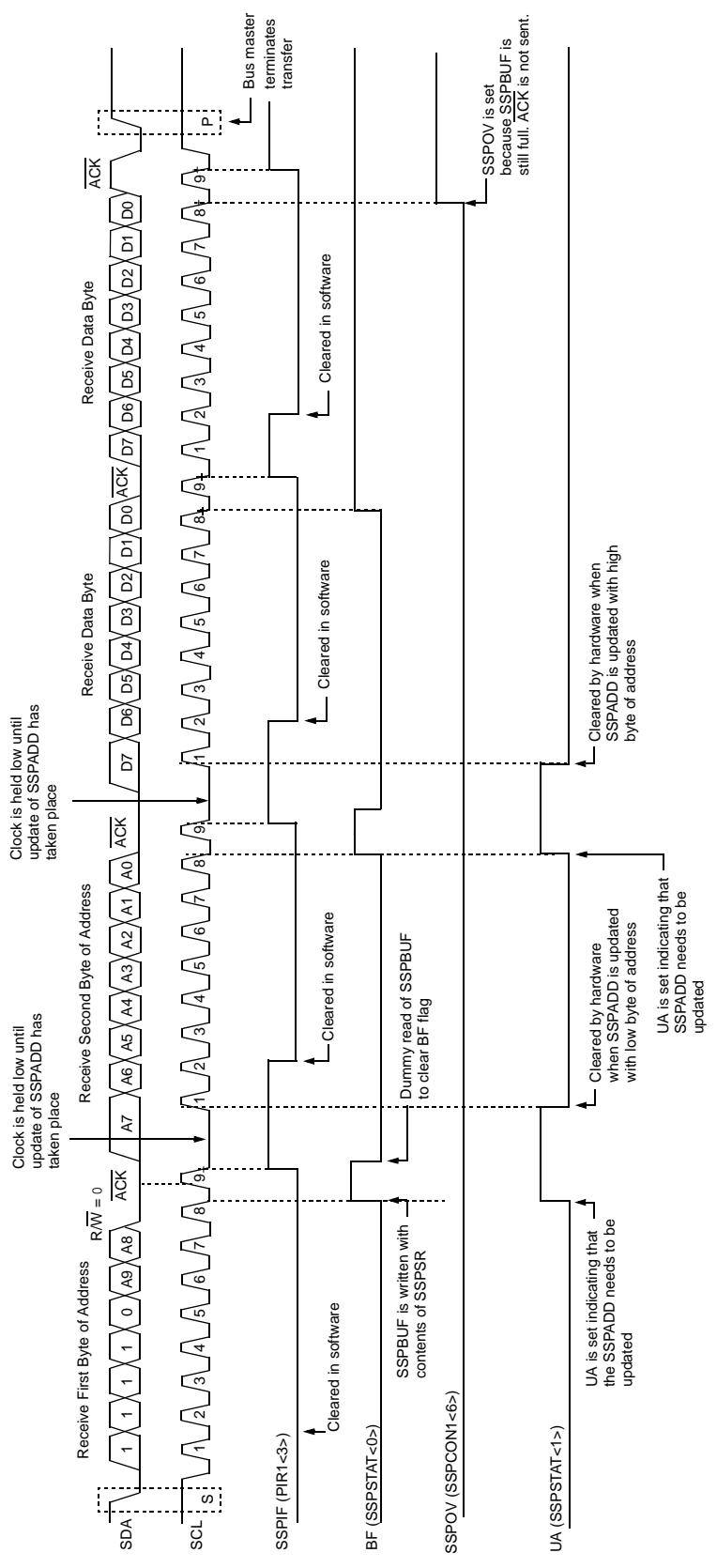
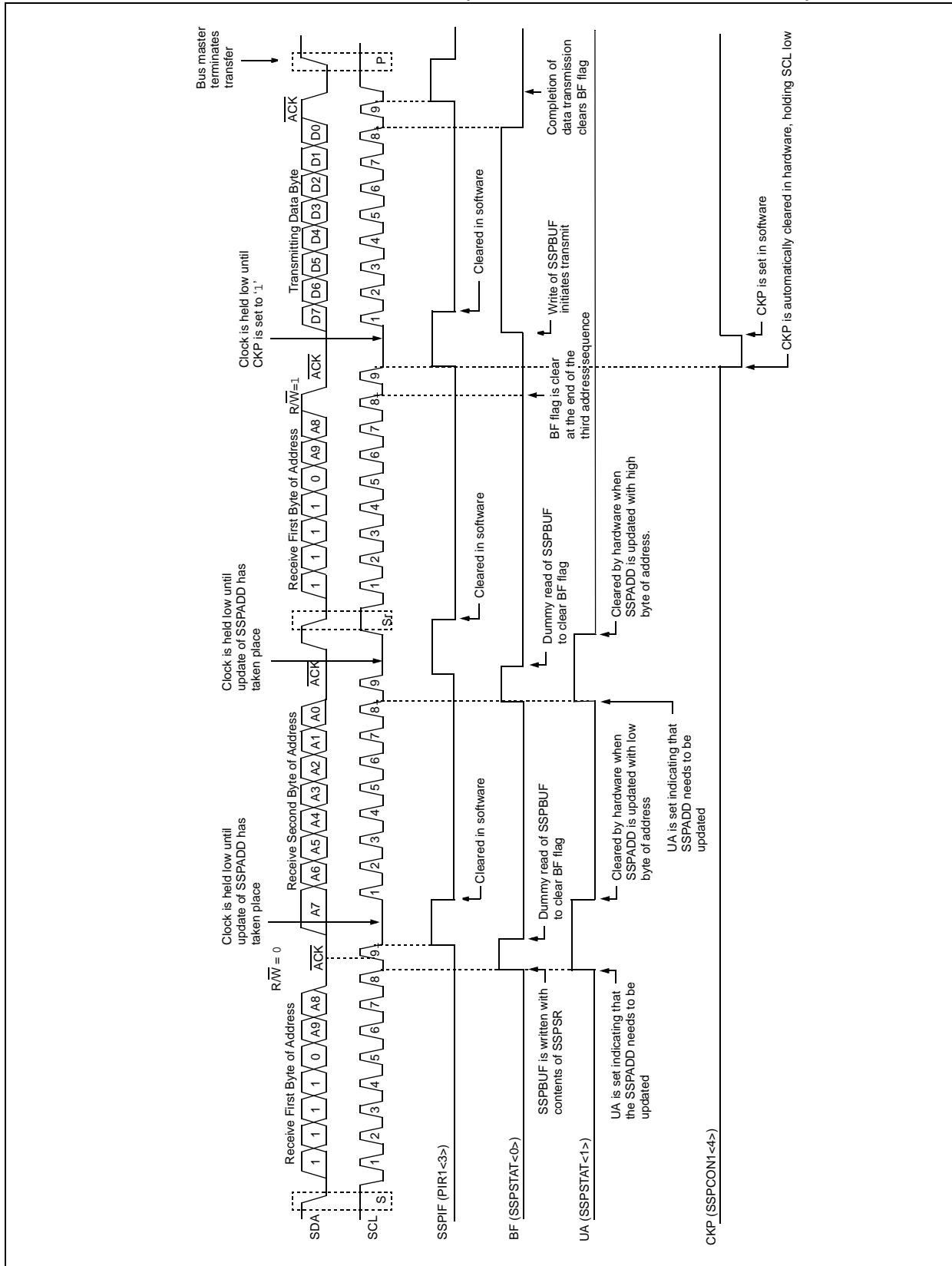


FIGURE 17-11:  $\text{I}^2\text{C}^{\text{TM}}$  SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)



## 17.4.4 CLOCK STRETCHING

Both 7-bit 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

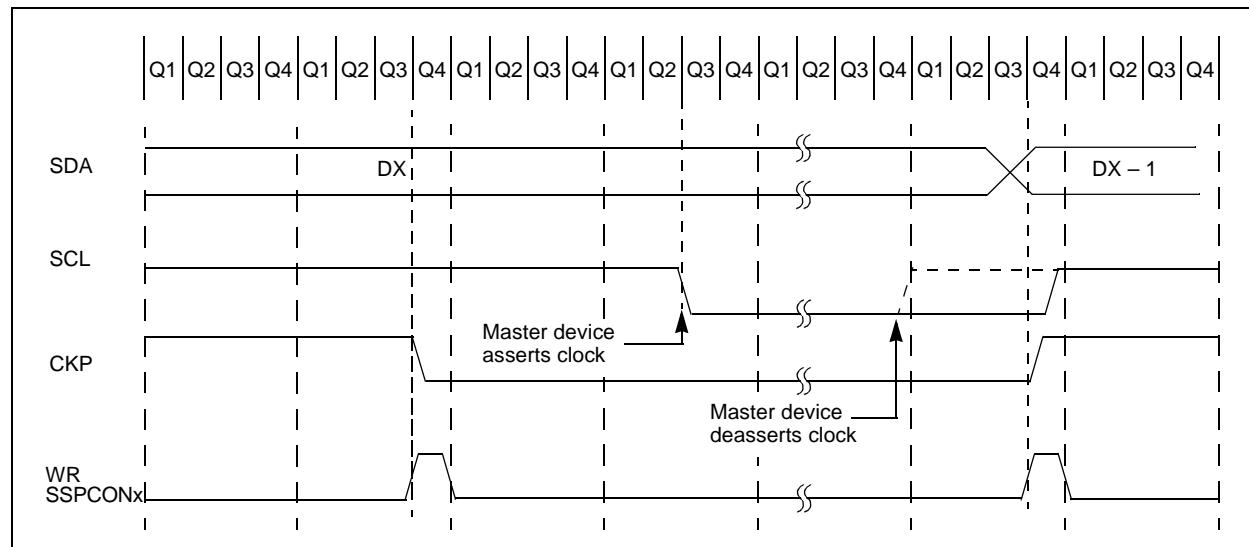
In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence which contains the high-order bits of the 10-bit address and the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



# PIC18F2525/2620/4525/4620

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

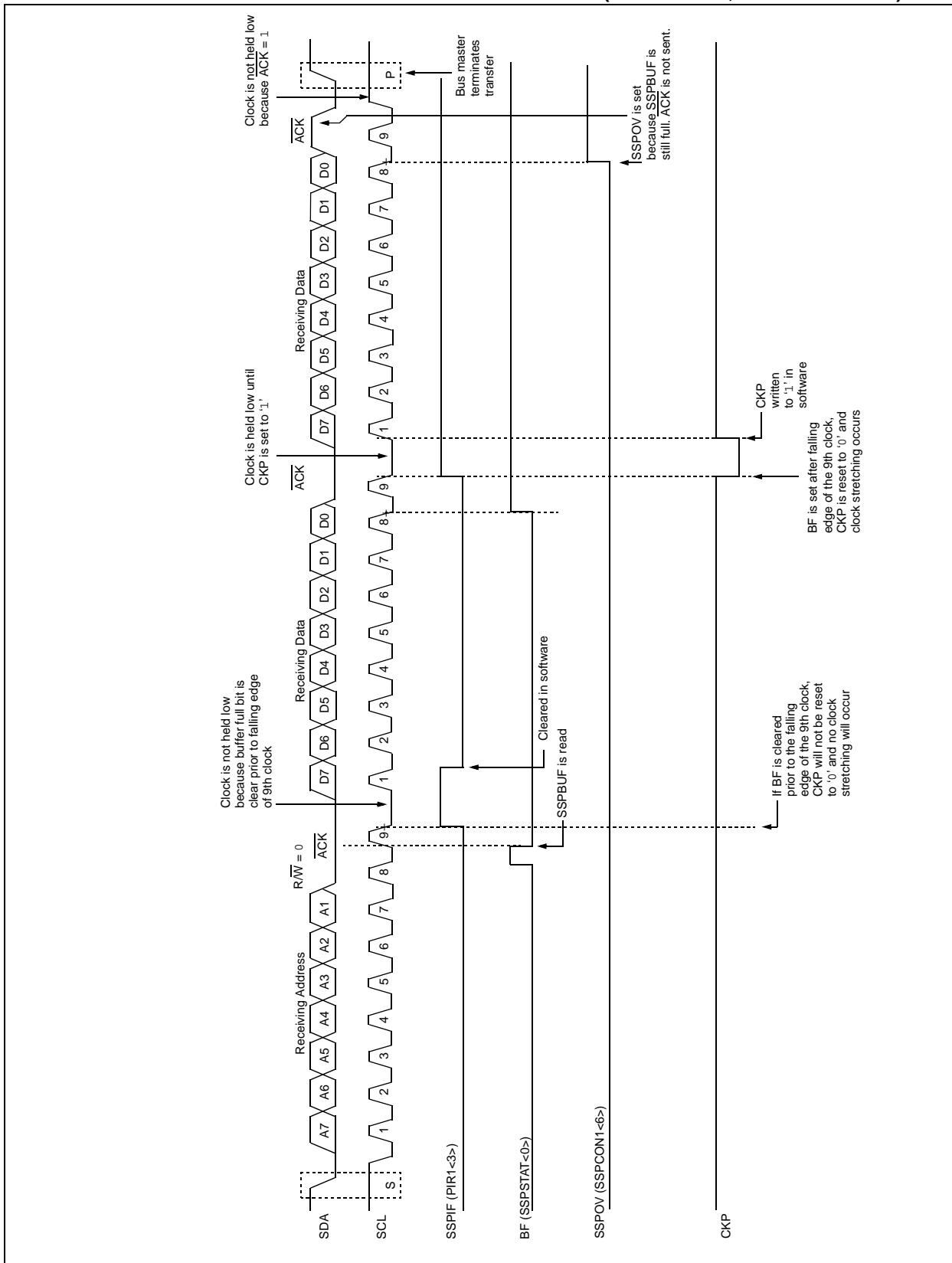
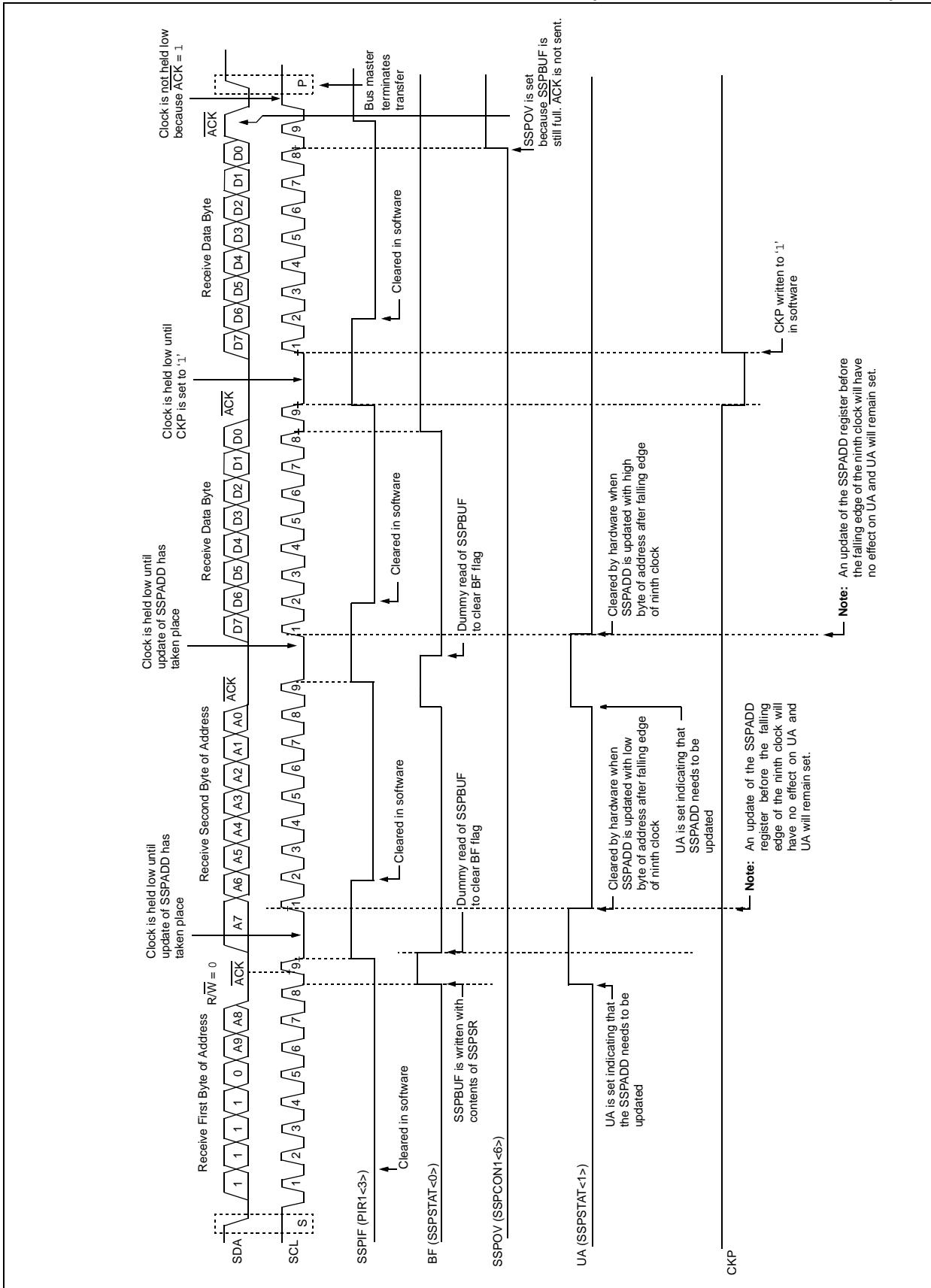


FIGURE 17-14:  $\text{I}^2\text{C}$ ™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESS)



## 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.

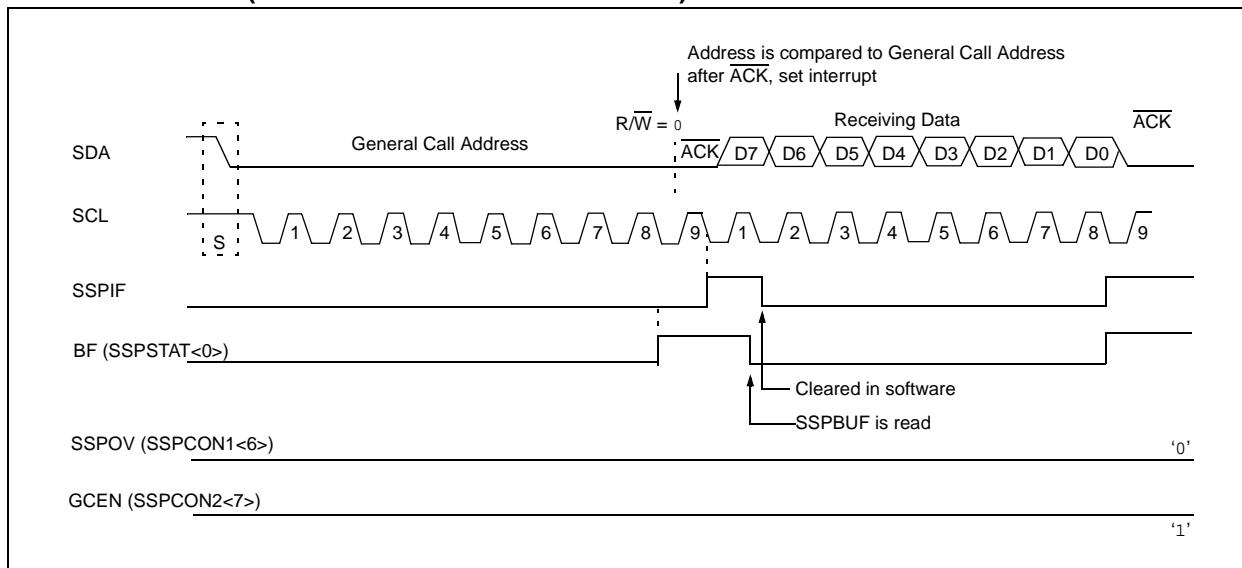
The general call address is recognized when the General Call Enable bit, GCEN, is enabled (SSPCON2<7> is set). Following a Start bit detect, 8 bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

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 (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 (SSPSTAT<1>) is set. 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)**



## 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.

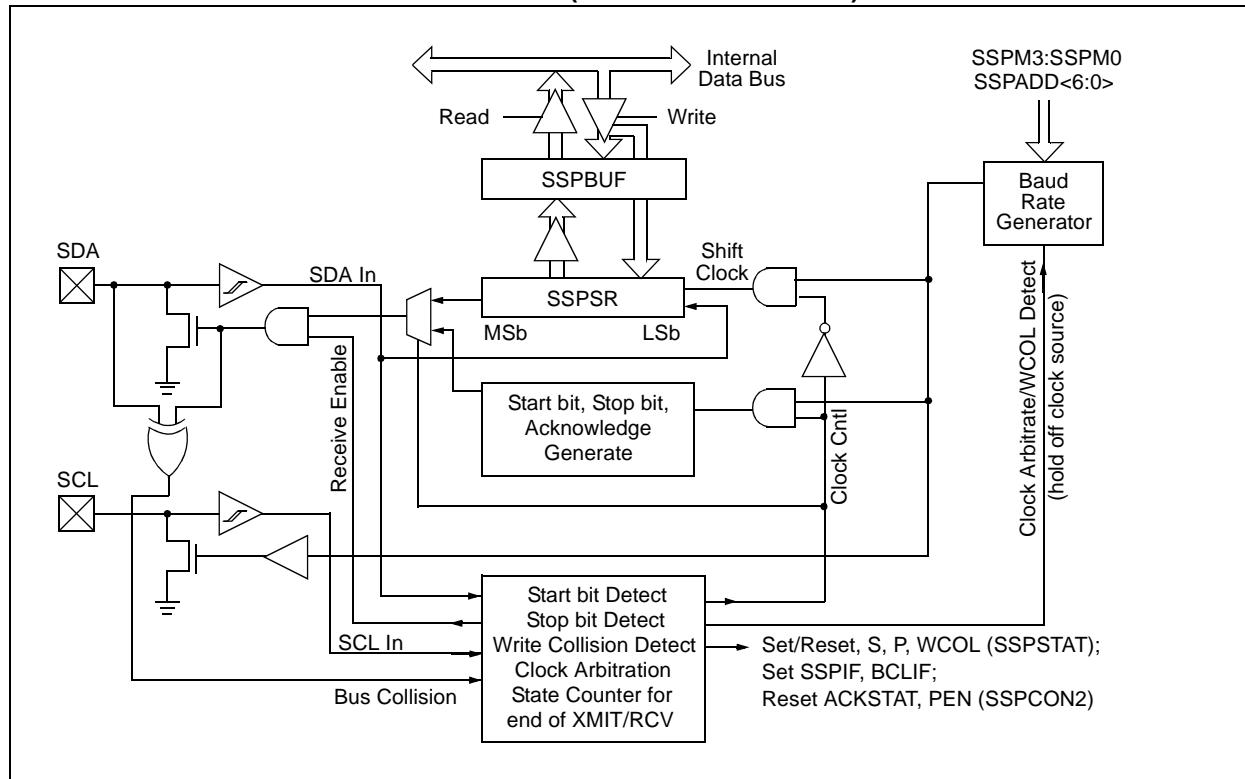
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the I<sup>2</sup>C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6.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



## 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.
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
7. The user loads the SSPBUF with eight bits of data.
8. Data is shifted out the SDA pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register.
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.

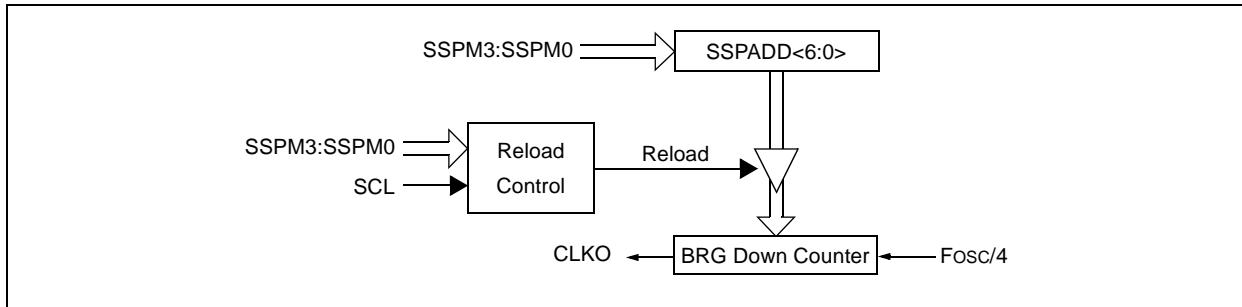
## 17.4.7 BAUD RATE

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.

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



**TABLE 17-3: I<sup>2</sup>C™ CLOCK RATE W/BRG**

Fosc	F <sub>CY</sub>	F <sub>CY</sub> * 2	BRG Value	F <sub>SCL</sub> (2 Rollovers of BRG)
40 MHz	10 MHz	20 MHz	18h	400 kHz <sup>(1)</sup>
40 MHz	10 MHz	20 MHz	1Fh	312.5 kHz
40 MHz	10 MHz	20 MHz	63h	100 kHz
16 MHz	4 MHz	8 MHz	09h	400 kHz <sup>(1)</sup>
16 MHz	4 MHz	8 MHz	0Ch	308 kHz
16 MHz	4 MHz	8 MHz	27h	100 kHz
4 MHz	1 MHz	2 MHz	02h	333 kHz <sup>(1)</sup>
4 MHz	1 MHz	2 MHz	09h	100 kHz
4 MHz	1 MHz	2 MHz	00h	1 MHz <sup>(1)</sup>

**Note 1:** The I<sup>2</sup>C interface does not conform to the 400 kHz I<sup>2</sup>C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.

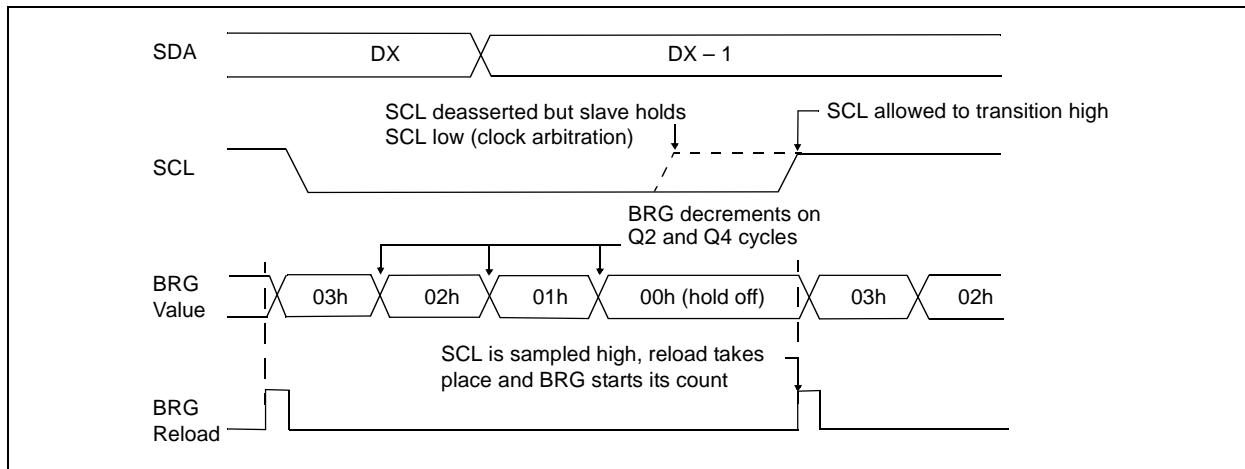
# PIC18F2525/2620/4525/4620

## 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.

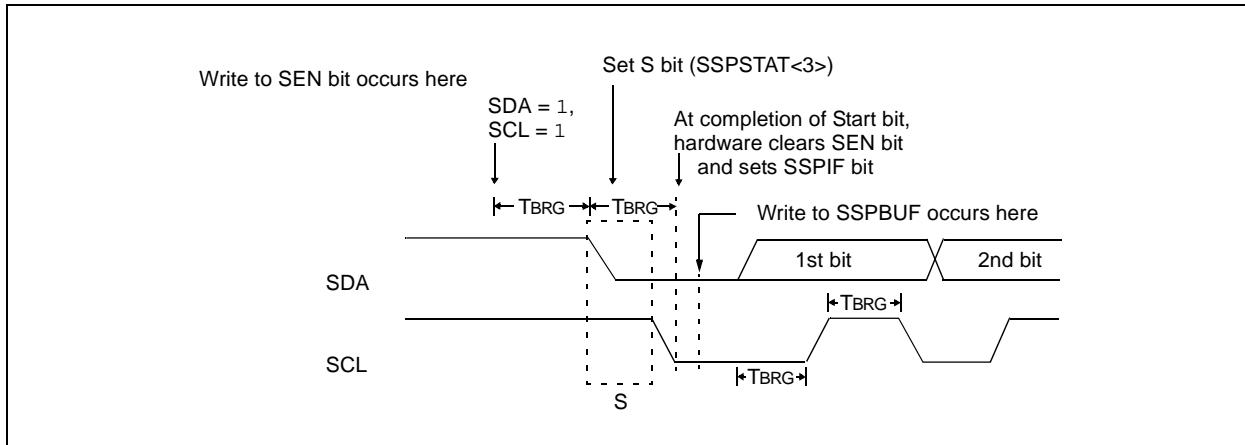
**Note:** If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted 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**



## 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'.

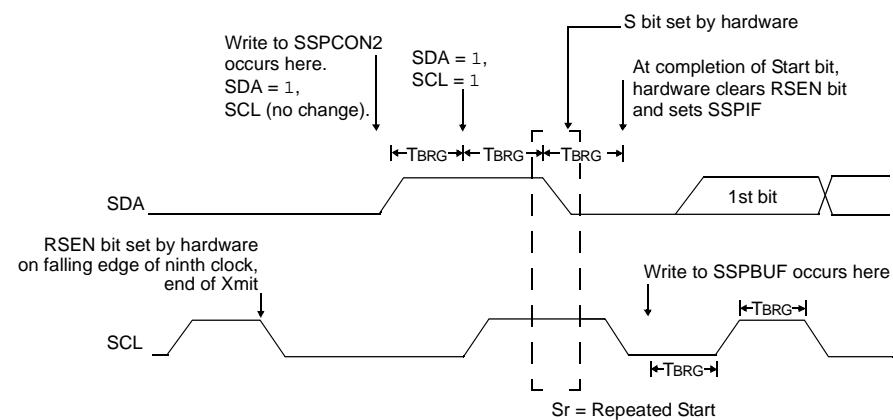
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode)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

### 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 queueing 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: REPEAT 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 ACK bit during the ninth bit time if an address match occurred, or if data was received properly. The status of 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 the address will be shifted out on the falling edge of SCL until all seven 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 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 flag is set and the contents of the buffer are unchanged (the write doesn't occur) after 2 Tcy after the SSPBUF write. If SSPBUF is rewritten within 2 Tcy, 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 ( $\overline{ACK} = 0$ ) and is set when the slave does not Acknowledge ( $\overline{ACK} = 1$ ). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

## 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>).

**Note:** The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN (SSPCON2<4>).

### 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).

# PIC18F2525/2620/4525/4620

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

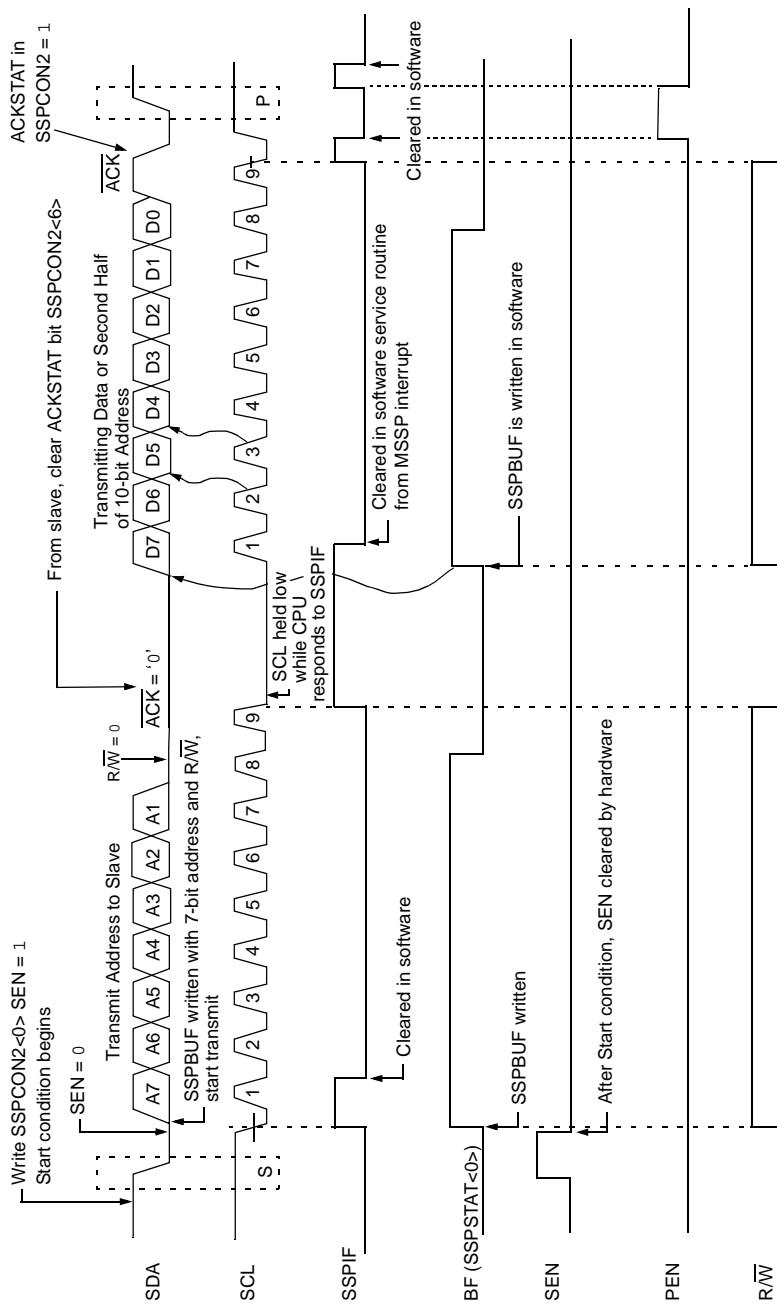
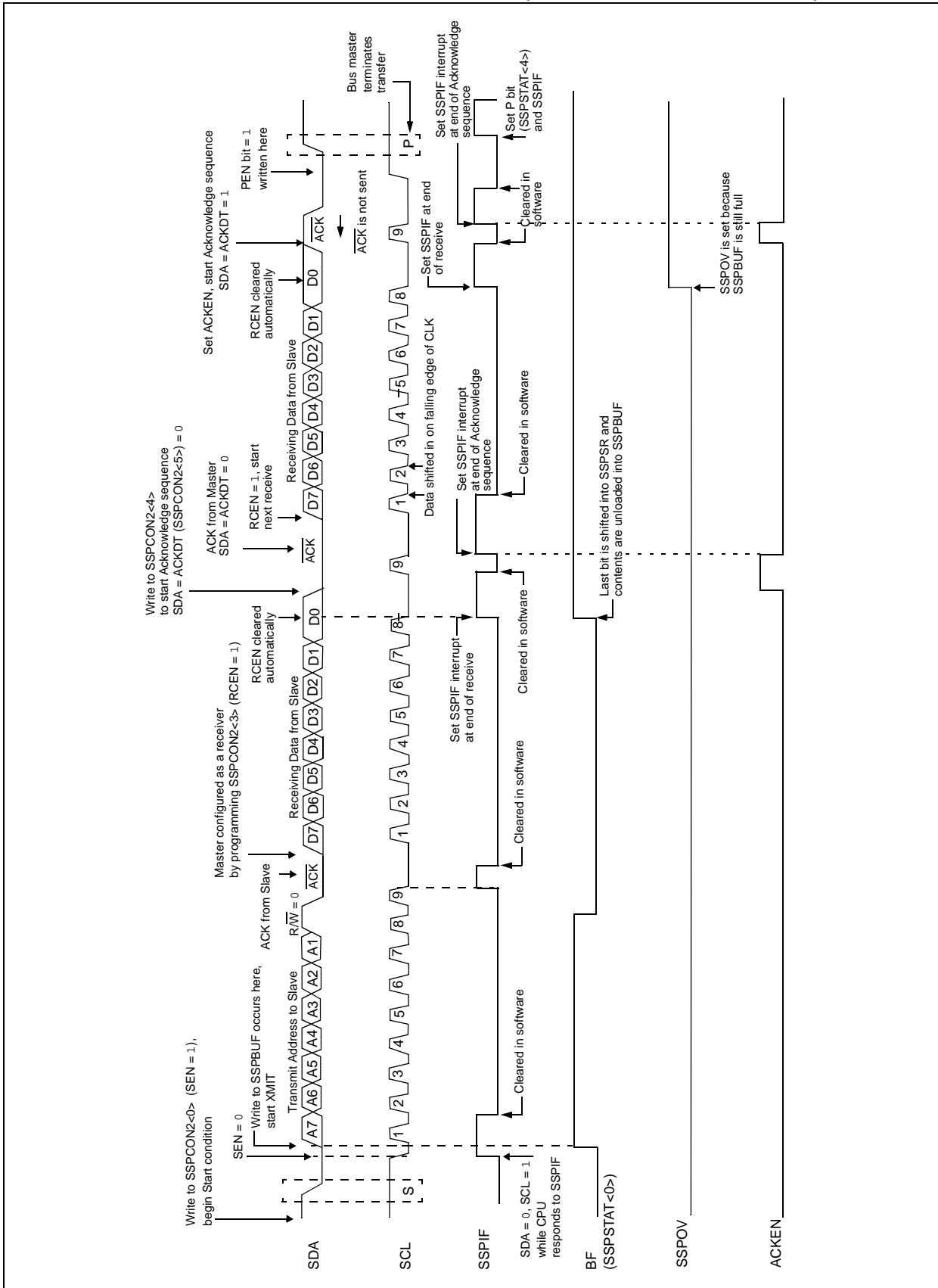


FIGURE 17-22:  $\text{I}^2\text{C}$ ™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



# PIC18F2525/2620/4525/4620

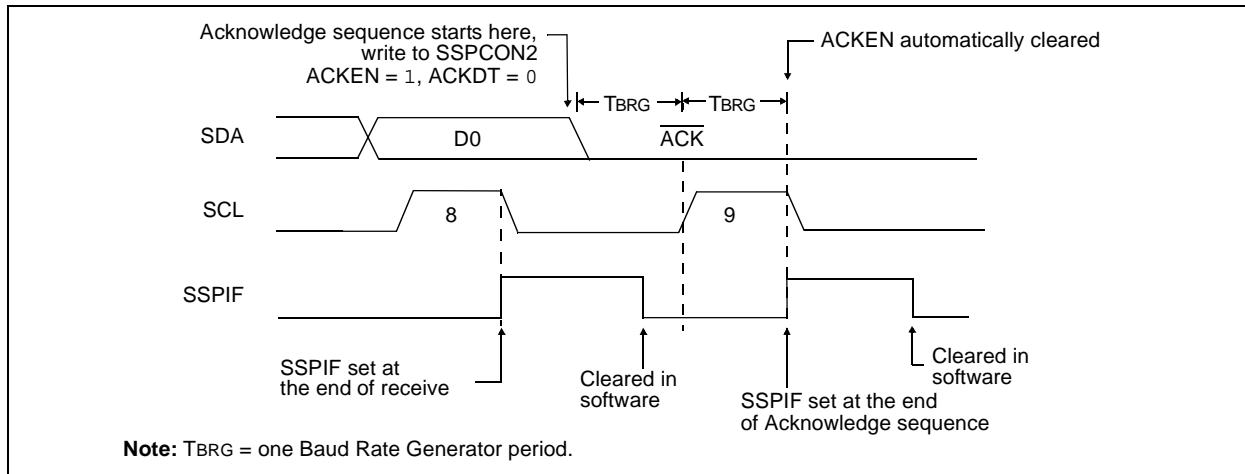
## 17.4.12 ACKNOWLEDGE SEQUENCE TIMING

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).

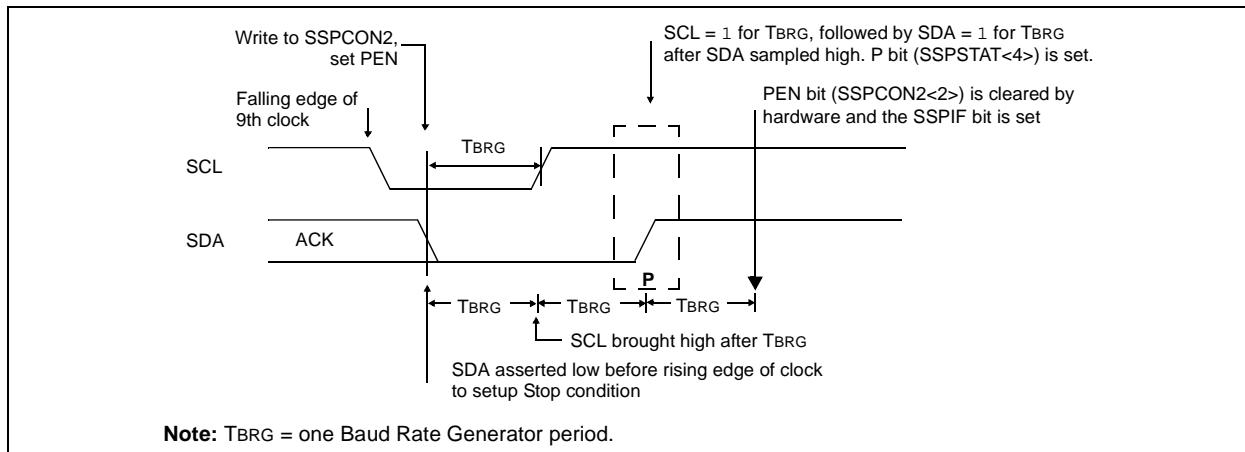
### 17.4.12.1 WCOL Status Flag

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).

**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 EFFECTS 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.

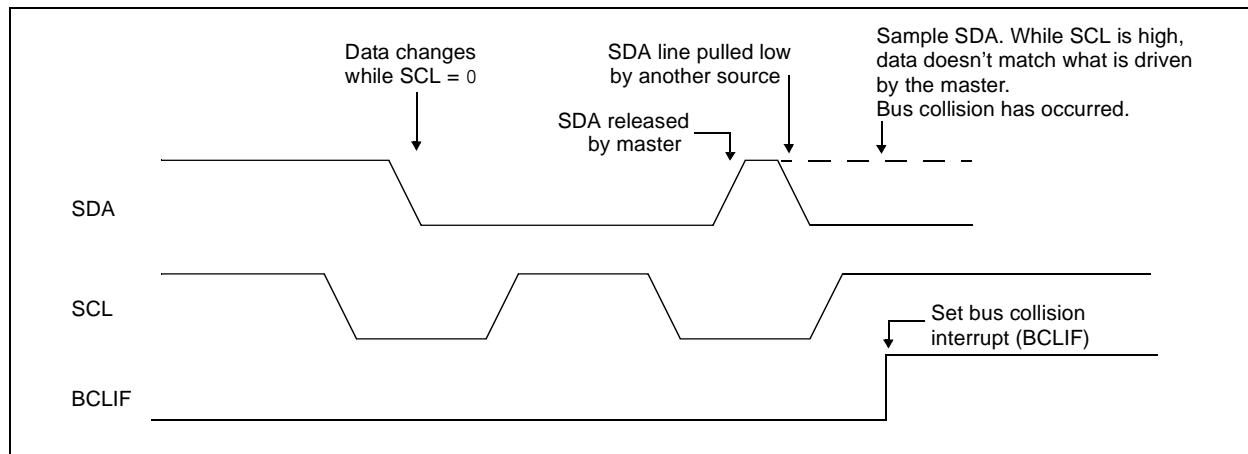
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, 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**



## 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; if the SCL pin is sampled as '0' during this time, 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)**

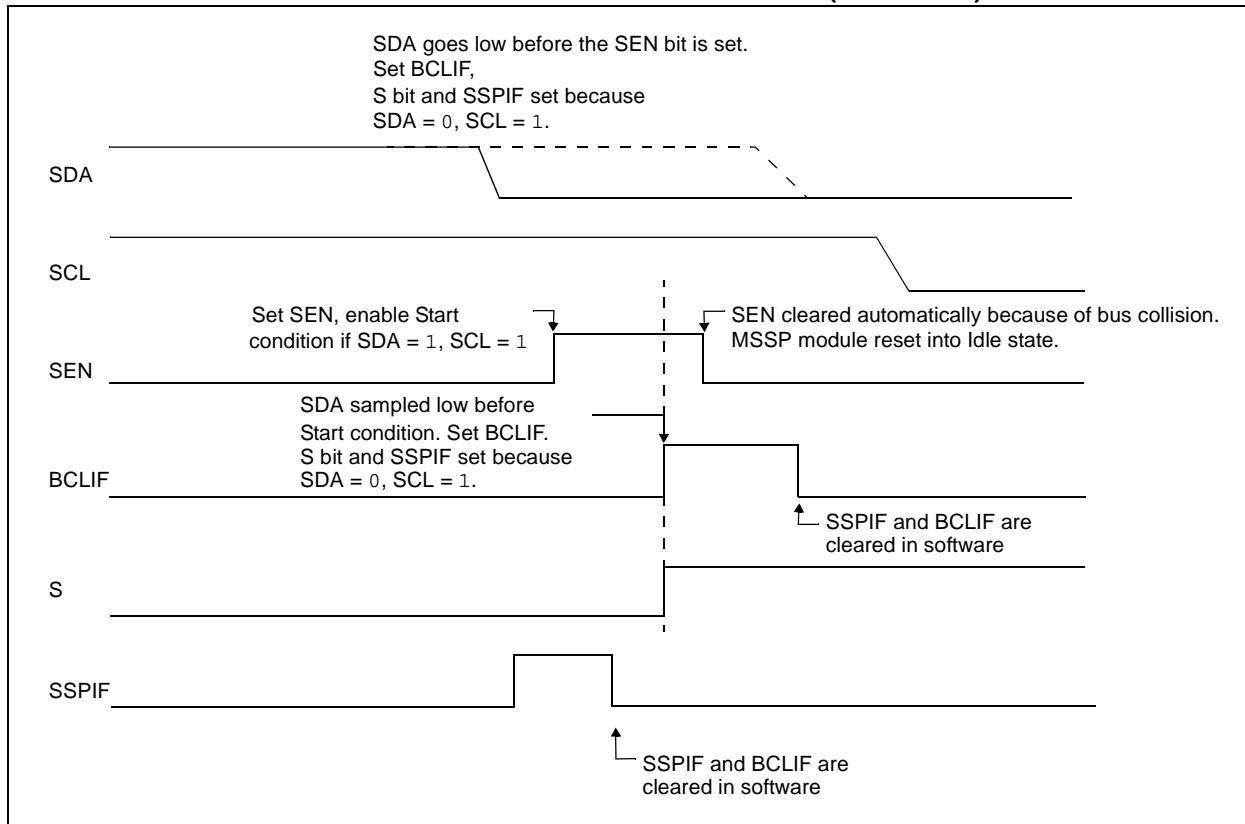


FIGURE 17-27: BUS COLLISION DURING START CONDITION (SCL = 0)

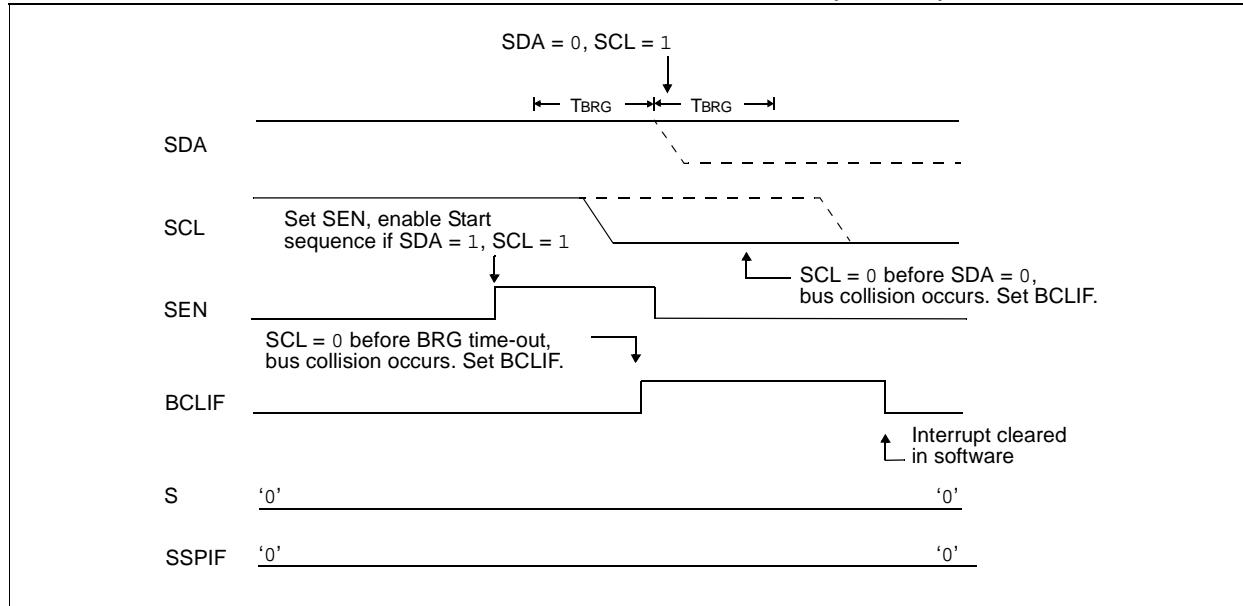
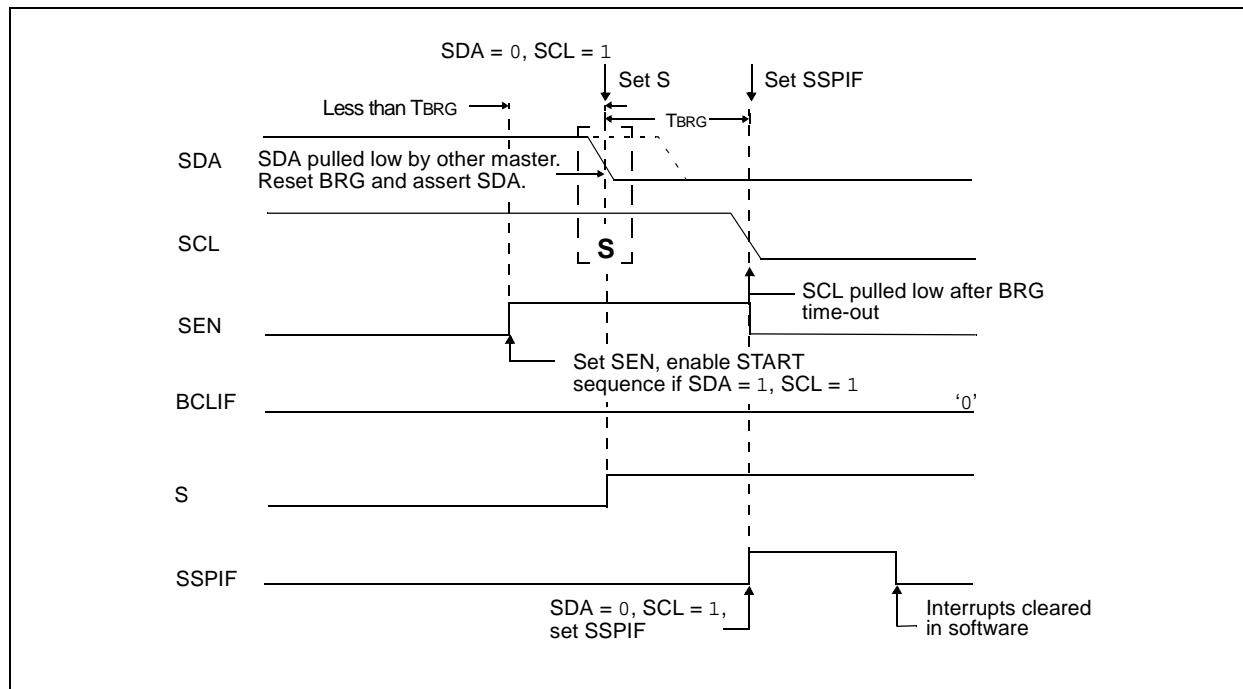


FIGURE 17-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



# PIC18F2525/2620/4525/4620

## 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'.

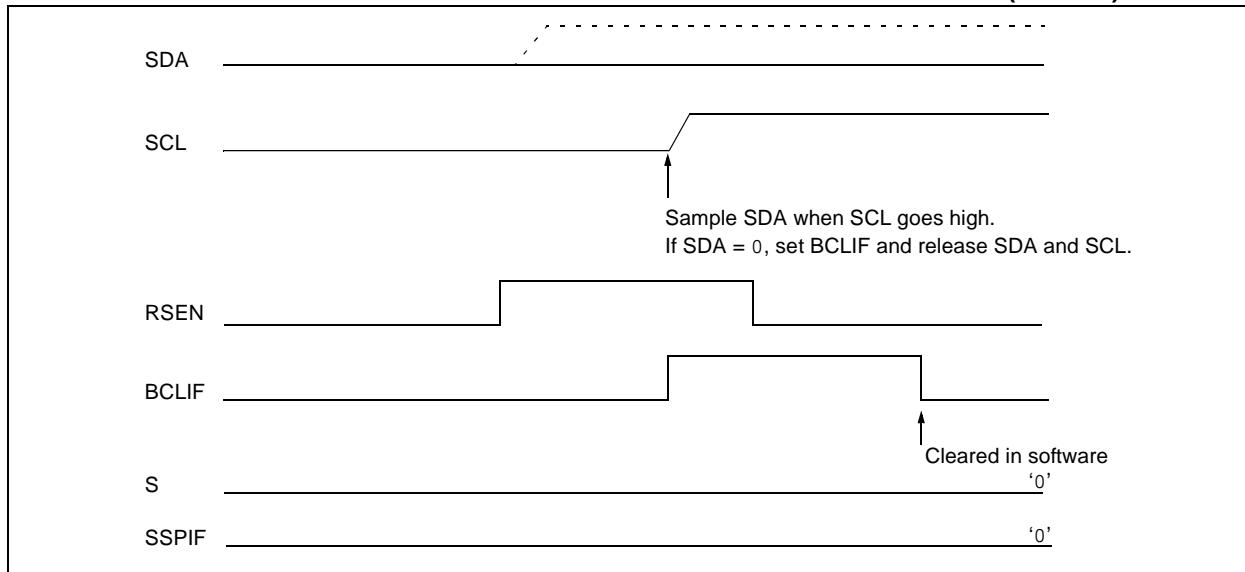
When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to 0. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

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.

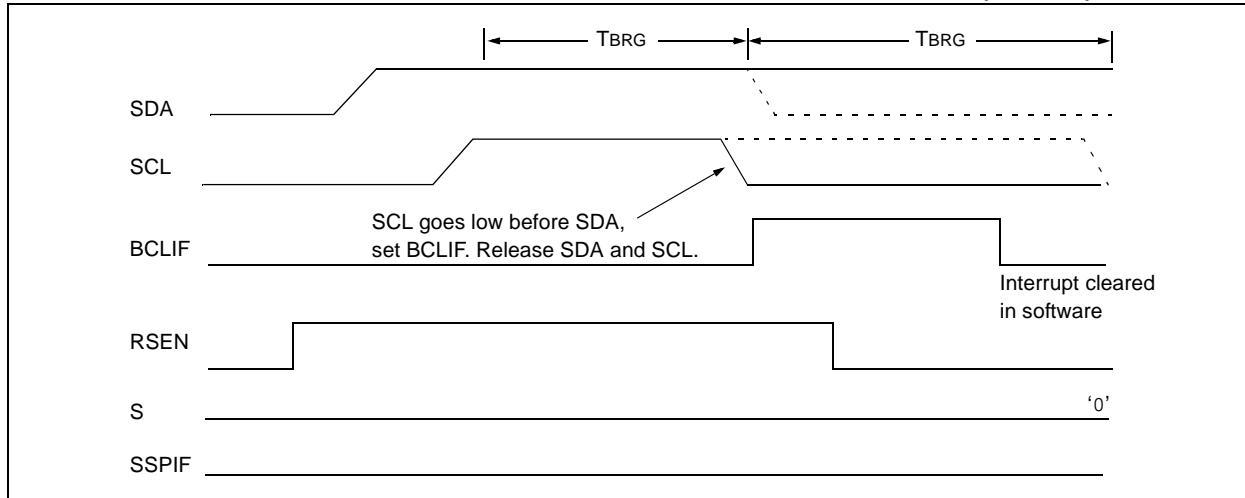
If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition, see Figure 17-30.

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 REPEATED START CONDITION (CASE 2)**



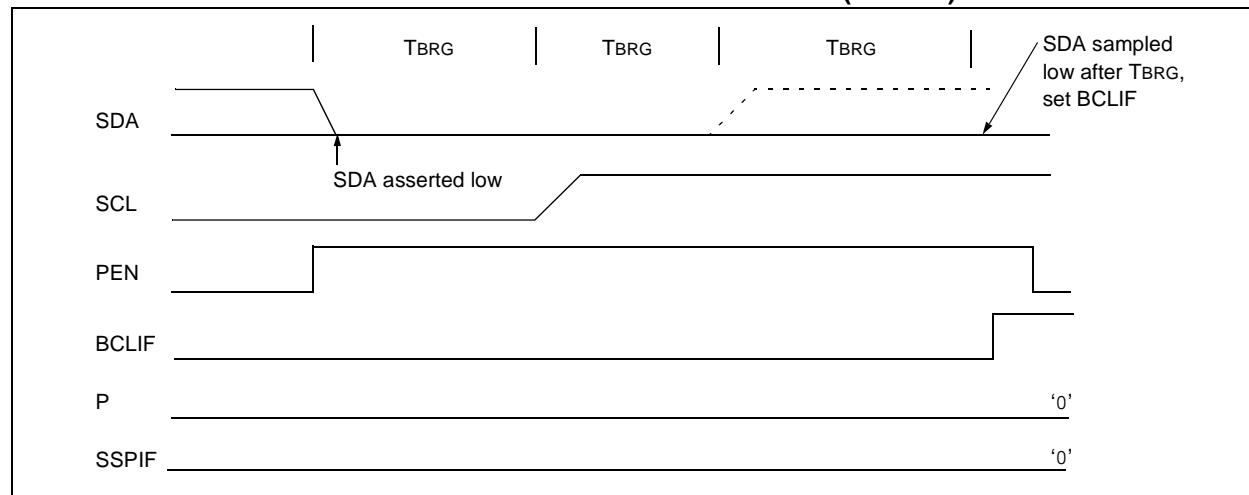
### 17.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

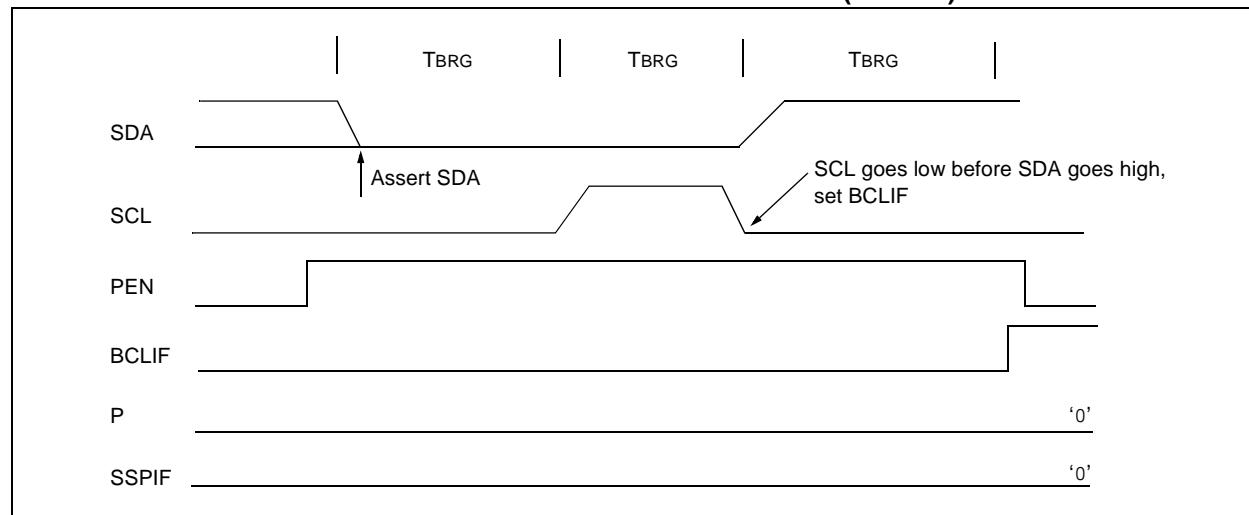
- After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.
- After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

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)**



# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 18.0 ENHANCED UNIVERSAL SYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is one of the two serial I/O modules. (Generically, the USART is also known as a Serial Communications Interface or SCI.) The EUSART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The Enhanced USART module implements additional features, including automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These make it ideally suited for use in Local Interconnect Network bus (LIN bus) systems.

The EUSART can be configured in the following modes:

- Asynchronous (full duplex) with:
  - Auto-wake-up on character reception
  - Auto-baud calibration
  - 12-bit Break character transmission
- Synchronous – Master (half duplex) with selectable clock polarity
- Synchronous – Slave (half duplex) with selectable clock polarity

The pins of the Enhanced USART are multiplexed with PORTC. In order to configure RC6/TX/CK and RC7/RX/DT as a USART:

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

**Note:** The EUSART control will automatically reconfigure the pin from input to output as needed.

The operation of the Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)

These are detailed on the following pages in Register 18-1, Register 18-2 and Register 18-3, respectively.

# PIC18F2525/2620/4525/4620

## REGISTER 18-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN <sup>(1)</sup>	SYNC	SENDDB	BRGH	TRMT	TX9D
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      **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 enabled  
0 = Transmit disabled

bit 4      **SYNC:** EUSART Mode Select bit

1 = Synchronous mode  
0 = Asynchronous mode

bit 3      **SENDDB:** Send Break Character bit

#### Asynchronous mode:

1 = Send Sync Break on next transmission (cleared by hardware upon completion)  
0 = Sync Break transmission completed

#### Synchronous mode:

Don't care.

bit 2      **BRGH:** High Baud Rate Select bit

#### Asynchronous mode:

1 = High speed  
0 = Low speed

#### Synchronous mode:

Unused in this mode.

bit 1      **TRMT:** Transmit Shift Register Status bit

1 = TSR empty  
0 = TSR full

bit 0      **TX9D:** 9th Bit of Transmit Data

Can be address/data bit or a parity bit.

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

# PIC18F2525/2620/4525/4620

## REGISTER 18-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>SPEN:</b> Serial Port Enable bit 1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins) 0 = Serial port disabled (held in Reset)
bit 6	<b>RX9:</b> 9-Bit Receive Enable bit 1 = Selects 9-bit reception 0 = Selects 8-bit reception
bit 5	<b>SREN:</b> Single Receive Enable bit <u>Asynchronous mode:</u> Don't care. <u>Synchronous mode – Master:</u> 1 = Enables single receive 0 = Disables single receive This bit is cleared after reception is complete. <u>Synchronous mode – Slave:</u> Don't care.
bit 4	<b>CREN:</b> Continuous Receive Enable bit <u>Asynchronous mode:</u> 1 = Enables receiver 0 = Disables receiver <u>Synchronous mode:</u> 1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN) 0 = Disables continuous receive
bit 3	<b>ADDEN:</b> Address Detect Enable bit <u>Asynchronous mode 9-Bit (RX9 = 1):</u> 1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set 0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit <u>Asynchronous mode 9-Bit (RX9 = 0):</u> Don't care.
bit 2	<b>FERR:</b> Framing Error bit 1 = Framing error (can be cleared by reading RCREG register and receiving next valid byte) 0 = No framing error
bit 1	<b>OERR:</b> Overrun Error bit 1 = Overrun error (can be cleared by clearing bit, CREN) 0 = No overrun error
bit 0	<b>RX9D:</b> 9th Bit of Received Data This can be address/data bit or a parity bit and must be calculated by user firmware.

# PIC18F2525/2620/4525/4620

## REGISTER 18-3: BAUDCON: BAUD RATE CONTROL REGISTER

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	<b>ABDOVF:</b> 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	<b>RCIDL:</b> Receive Operation Idle Status bit 1 = Receive operation is Idle 0 = Receive operation is active
bit 5	<b>RXDTP:</b> Received Data Polarity Select bit (Asynchronous mode only) <u>Asynchronous mode:</u> 1 = RX data is inverted 0 = RX data received is not inverted
bit 4	<b>TXCKP:</b> Clock and Data Polarity Select bit <u>Asynchronous mode:</u> 1 = Idle state for transmit (TX) is a low level 0 = Idle state for transmit (TX) is a high level <u>Synchronous mode:</u> 1 = Idle state for clock (CK) is a high level 0 = Idle state for clock (CK) is a low level
bit 3	<b>BRG16:</b> 16-Bit Baud Rate Register Enable bit 1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG 0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored
bit 2	<b>Unimplemented:</b> Read as '0'
bit 1	<b>WUE:</b> Wake-up Enable bit <u>Asynchronous mode:</u> 1 = EUSART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge 0 = RX pin not monitored or rising edge detected <u>Synchronous mode:</u> Unused in this mode.
bit 0	<b>ABDEN:</b> Auto-Baud Detect Enable bit <u>Asynchronous mode:</u> 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 <u>Synchronous mode:</u> Unused in this mode.

## 18.1 Baud Rate Generator (BRG)

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

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

Given the desired baud rate and Fosc, the nearest integer value for the SPBRGH:SPBRG registers can be calculated using the formulas in Table 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 any value (even the same value) to the SPBRGH:SPBRG registers immediately reloads the BRG timer. This may corrupt a transmission or reception already in progress. This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

### 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 SPBRG register pair.

### 18.1.2 SAMPLING

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

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

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

**Legend:** x = Don't care, n = value of SPBRGH:SPBRG register pair

# PIC18F2525/2620/4525/4620

## 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 =  $FOSC/(64 ([SPBRGH:SPBRG] + 1))$

Solving for SPBRGH:SPBRG:

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

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 BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TXSTA	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D	51
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

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

# PIC18F2525/2620/4525/4620

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)	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)	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	—	—	—	—	—	—	—	—	—

# PIC18F2525/2620/4525/4620

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)	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)	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	—	—	—	—	—	—	—	—	—

### 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 RX signal, the RX signal is timing the BRG. In ABD mode, the internal Baud Rate Generator is used as a counter to time the bit period of the incoming serial byte stream.

Once the ABDEN bit is set, the state machine will clear the BRG and look for a Start bit. The Auto-Baud Rate Detect must receive a byte with the value, 55h (ASCII "U", which is also the LIN bus Sync character), in order to calculate the proper bit rate. The measurement is taken over both a low and a high bit time in order to minimize any effects caused by asymmetry of the incoming signal. After a Start bit, the SPBRG begins counting up, using the preselected clock source on the first rising edge of RX. After eight bits on the RX pin or the fifth rising edge, an accumulated value totalling the proper BRG period is left in the SPBRGH:SPBRG register pair. Once the 5th edge is seen (this should correspond to the Stop bit), the ABDEN bit is automatically cleared.

If a rollover of the BRG occurs (an overflow from FFFFh to 0000h), the event is trapped by the ABDOVF status bit (BAUDCON<7>). It is set in hardware by BRG rollovers and can be set or cleared by the user in software. ABD mode remains active after rollover events and the ABDEN bit remains set (Figure 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 will be configured by the BRG16 and BRGH bits. Independent of the BRG16 bit setting, both the SPBRG and SPBRGH will be used as a 16-bit counter. This allows the user to verify that no carry occurred for 8-bit modes by checking for 00h in the SPBRGH register. Refer to Table 18-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the USART state machine is held in Idle. The RCIF interrupt is set once the fifth rising edge on RX is detected. The value in the RCREG needs to be read to clear the RCIF interrupt. The contents of RCREG should be discarded.

**Note 1:** If the WUE bit is set with the ABDEN bit, Auto-Baud Rate Detection will occur on the byte *following* the Break character.

**2:** It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible due to bit error rates. Overall system timing and communication baud rates must be taken into consideration when using the Auto-Baud Rate Detection feature.

**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

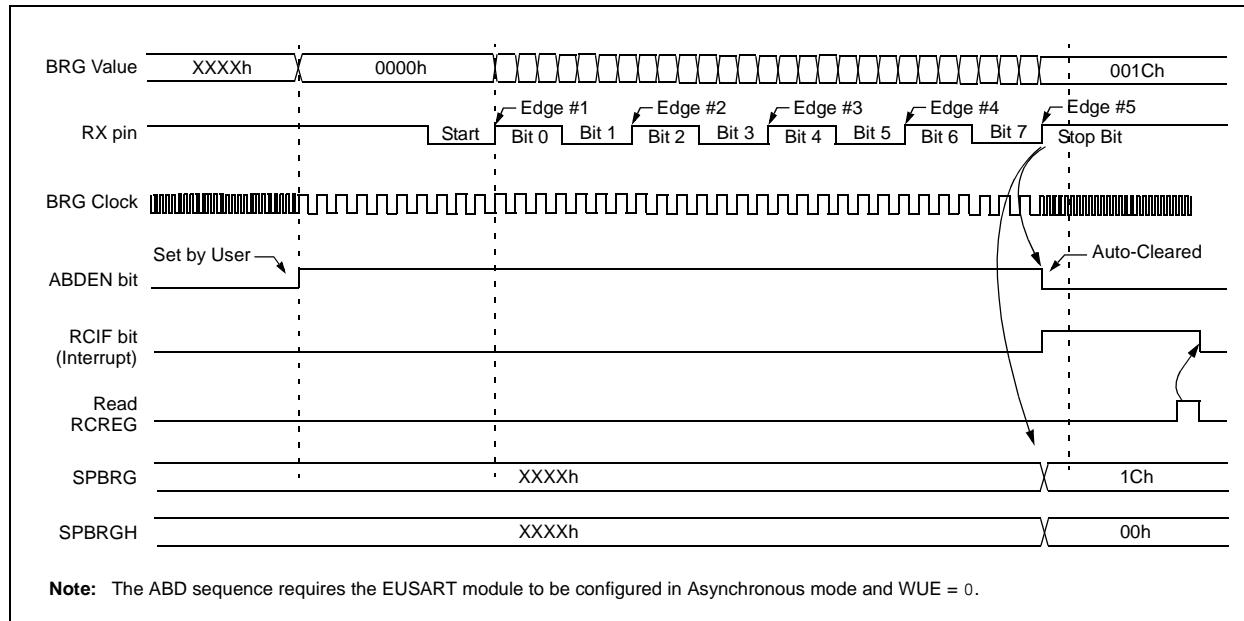
**Note:** During the ABD sequence, SPBRG and SPBRGH are both used as a 16-bit counter, independent of BRG16 setting.

### 18.1.3.1 ABD and EUSART Transmission

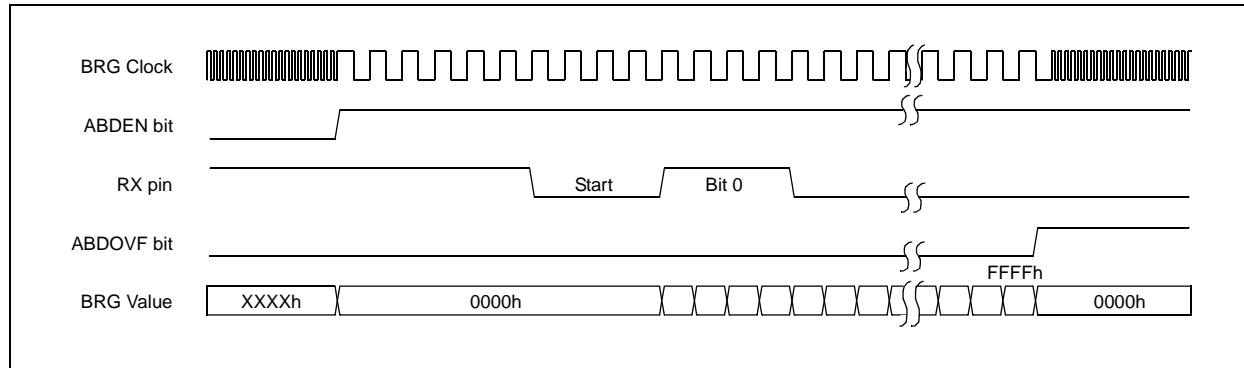
Since the BRG clock is reversed during ABD acquisition, the EUSART transmitter cannot be used during ABD. This means that whenever the ABDEN bit is set, TXREG cannot be written to. Users should also ensure that ABDEN does not become set during a transmit sequence. Failing to do this may result in unpredictable EUSART operation.

# PIC18F2525/2620/4525/4620

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



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



## 18.2 EUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTA<4>). In this mode, the EUSART uses standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine data bits and one Stop bit). The most common data format is 8 bits. An on-chip dedicated 8-bit/16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent but use the same data format and baud rate. The Baud Rate Generator produces a clock, either x16 or x64 of the bit shift rate depending on the BRGH and BRG16 bits (TXSTA<2> and BAUDCON<3>). Parity is not supported by the hardware but can be implemented in software and stored as the 9th data bit.

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

- 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, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREG register (if available).

Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and the TXIF flag bit (PIR1<4>) is set. This interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXIE (PIE1<4>). TXIF will be set regardless of the state of TXIE; it cannot be cleared in software. TXIF is also not cleared immediately upon loading TXREG, but becomes valid in the second instruction cycle following a load instruction. Polling TXIF immediately following a load of TXREG will return invalid results.

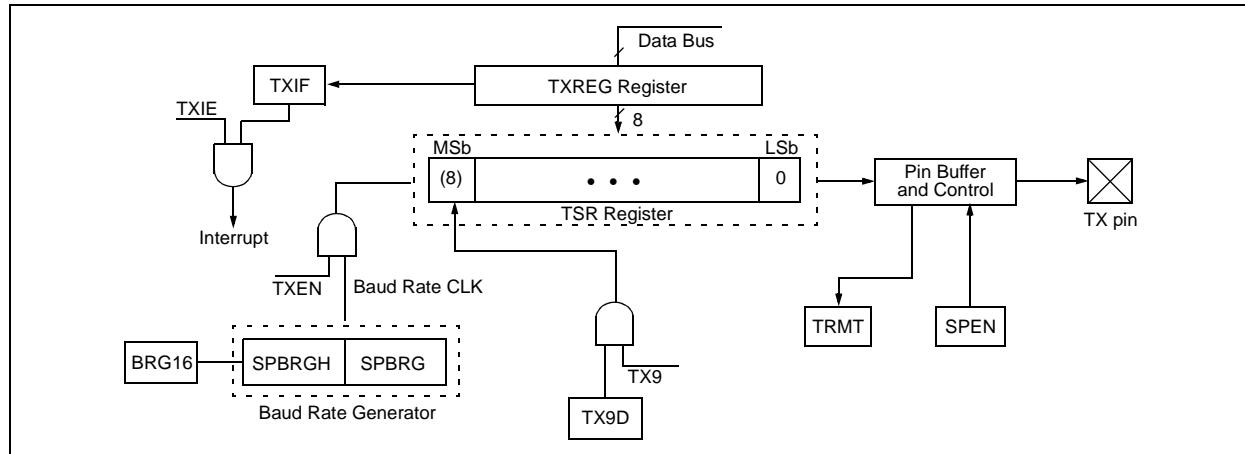
While TXIF indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register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

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

To set up an Asynchronous Transmission:

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

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



# PIC18F2525/2620/4525/4620

FIGURE 18-4: ASYNCHRONOUS TRANSMISSION

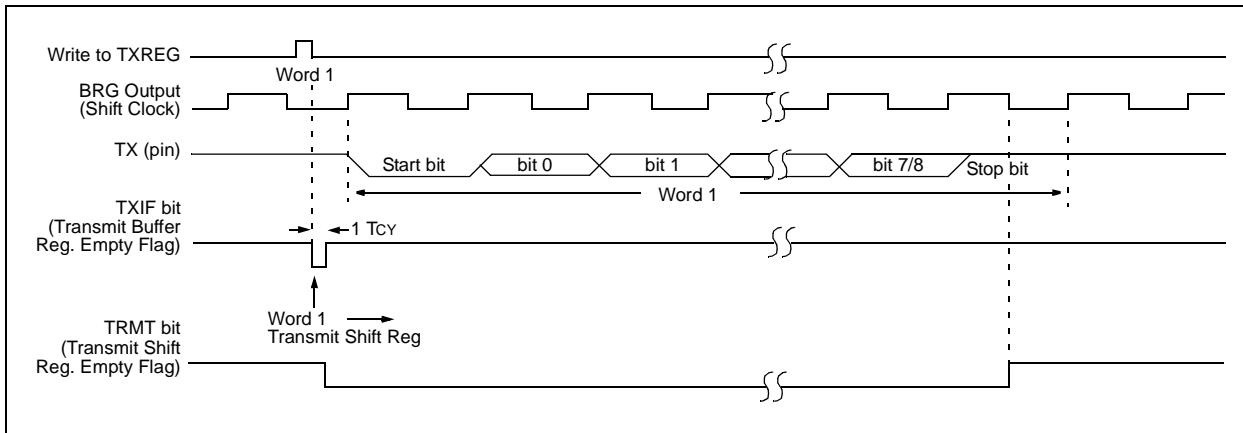


FIGURE 18-5: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)

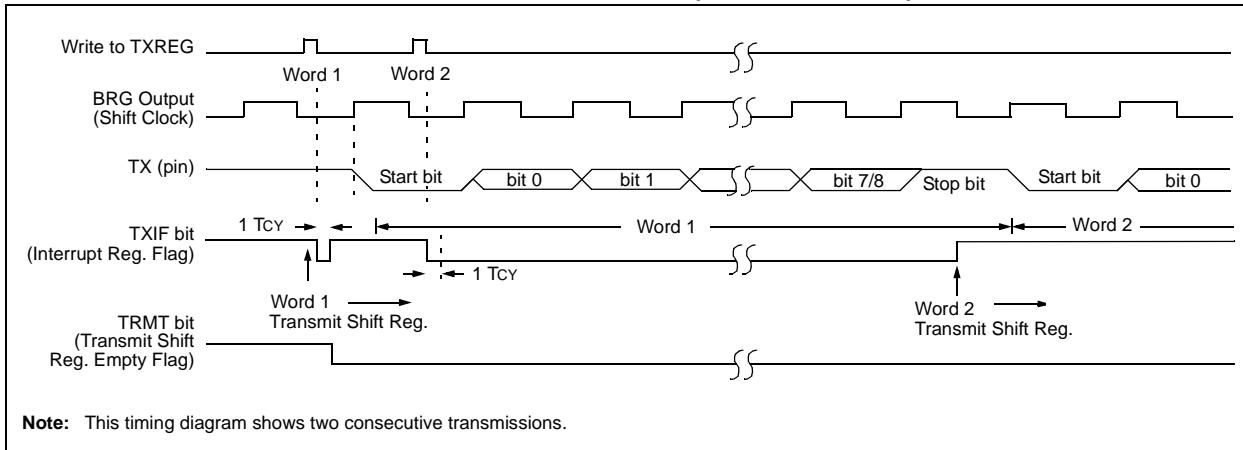


TABLE 18-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

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	TMROIE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
TXREG	EUSART Transmit Register								51
TXSTA	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D	51
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

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

Note 1: These bits are unimplemented on 28-pin devices and read as '0'.

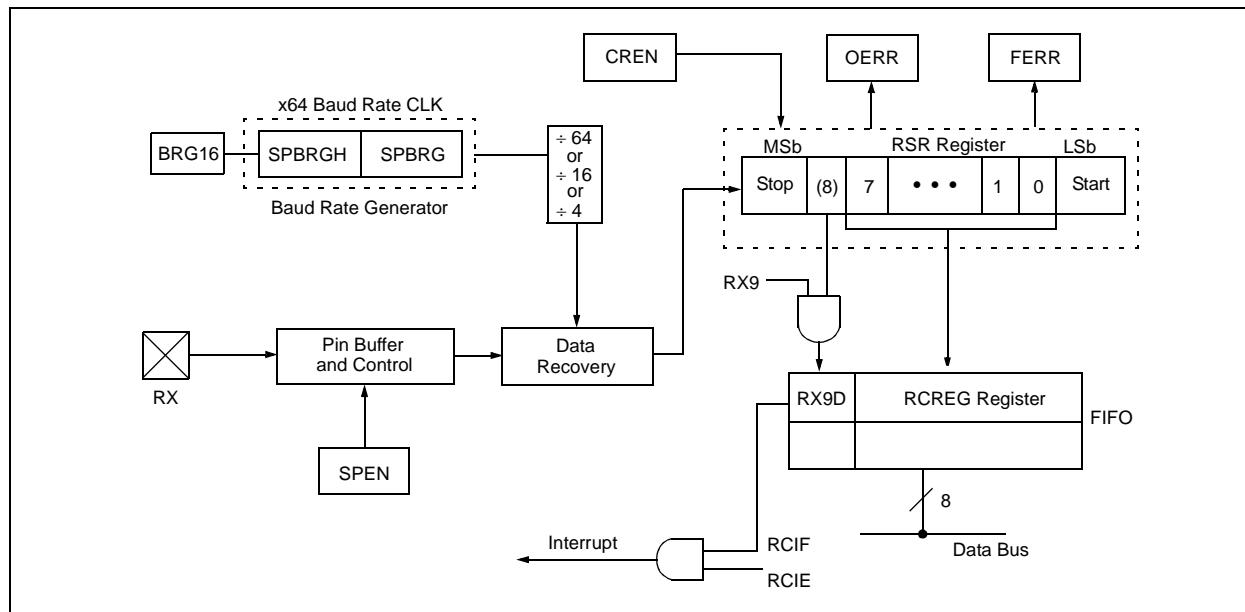
## 18.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 18-6. The data is received on the RX pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at Fosc. This mode would typically be used in RS-232 systems.

To set up an Asynchronous Reception:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RCIE.
4. If 9-bit reception is desired, set bit, RX9.
5. Enable the reception by setting bit, CREN.
6. Flag bit, RCIF, will be set when reception is complete and an interrupt will be generated if enable bit, RCIE, was set.
7. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit, CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

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



## 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 interrupts are required, set the RCIE bit and select the desired priority level with the RCIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCIE and GIE bits are set.
8. Read the RCSTA register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREG to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

# PIC18F2525/2620/4525/4620

FIGURE 18-7: ASYNCHRONOUS RECEPTION

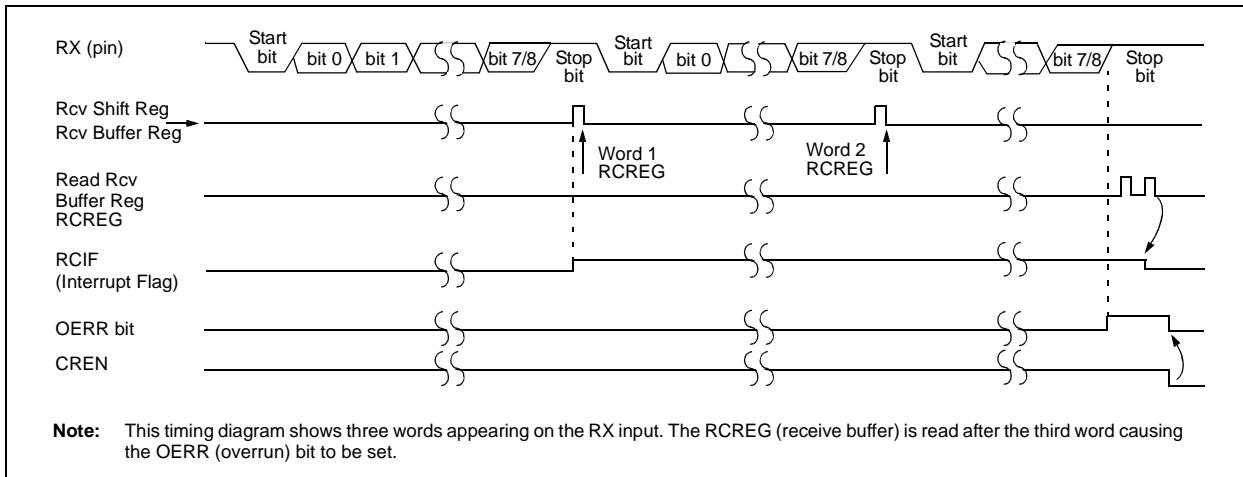


TABLE 18-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

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	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
RCREG	EUSART Receive Register								51
TXSTA	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D	51
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

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

**Note 1:** These bits are unimplemented on 28-pin devices and read as '0'.

## 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 RX/DT line while the EUSART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCON<1>). Once set, the typical receive sequence on RX/DT is disabled and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a Sync Break or a Wake-up Signal character for the LIN protocol.)

Following a wake-up event, the module generates an RCIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 18-8) and asynchronously, if the device is in Sleep mode (Figure 18-9). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RX line following the wake-up event. At this point, the EUSART module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.

## 18.2.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RX/DT, information with any state changes before the Stop bit may signal a false End-Of-Character (EOC) and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all '0's. This can be 00h (8 bytes) for standard RS-232 devices or 000h (12 bits) for LIN bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., 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.

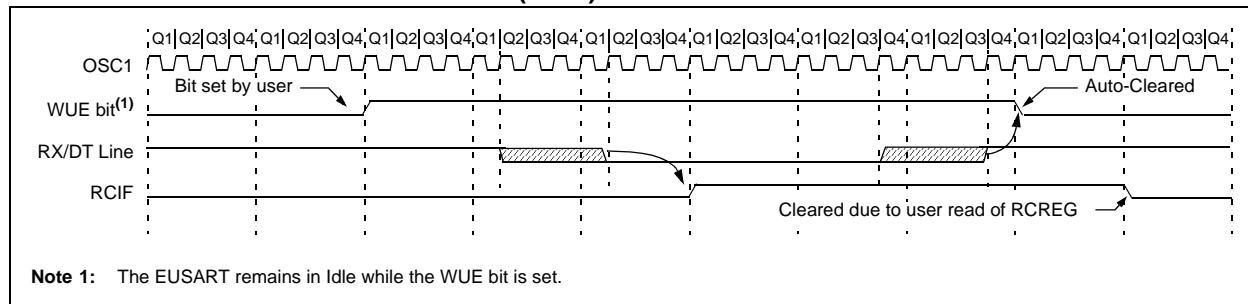
## 18.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the EUSART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared after this when a rising edge is seen on RX/DT. The interrupt condition is then cleared by reading the RCREG register. Ordinarily, the data in RCREG will be dummy data and should be discarded.

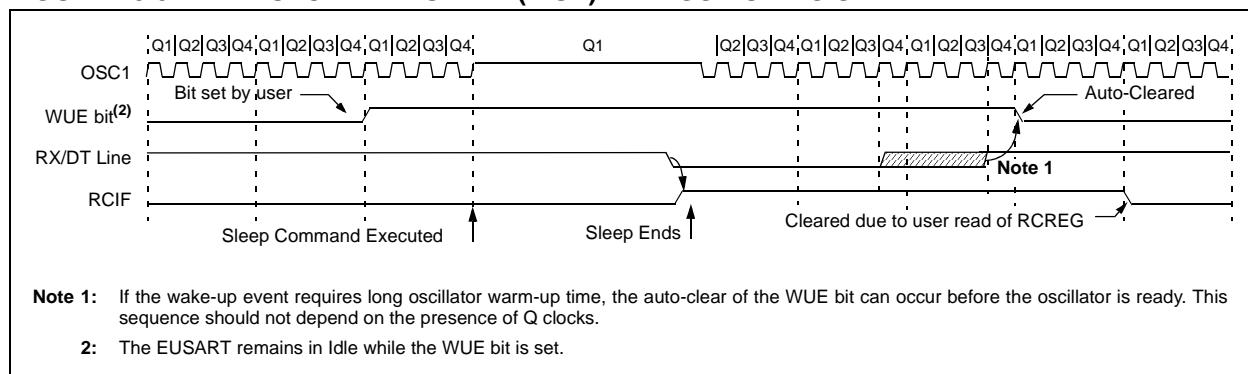
The fact that the WUE bit has been cleared (or is still set) and the RCIF flag is set should not be used as an indicator of the integrity of the data in RCREG. Users should consider implementing a parallel method in firmware to verify received data integrity.

To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

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



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



## 18.2.5 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. The Break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The Frame Break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift Register is loaded with data. Note that the value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

Note that the data value written to the TXREG for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 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 bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

## 18.2.6 RECEIVING A BREAK CHARACTER

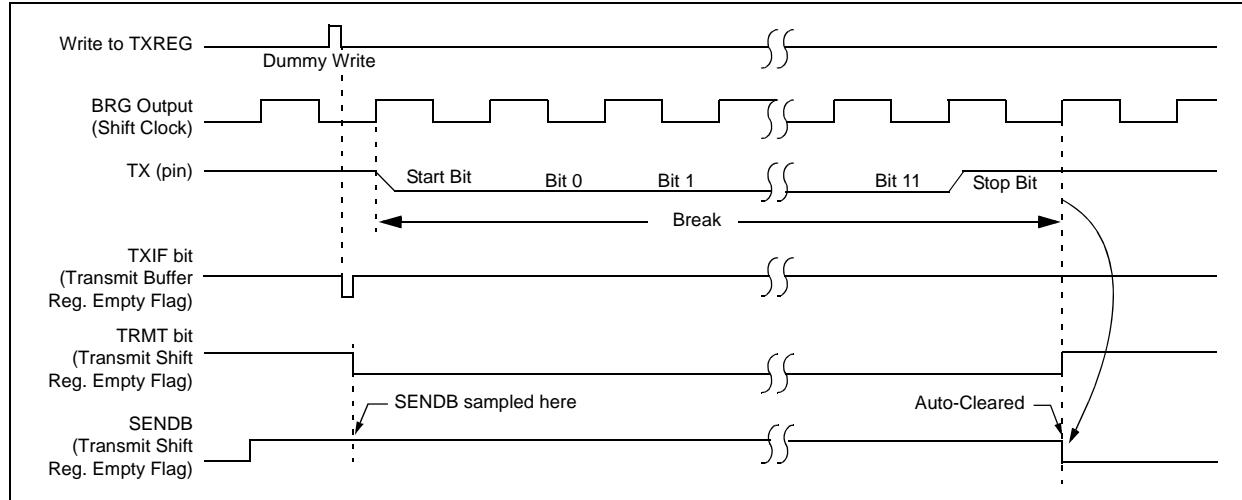
The Enhanced USART module can receive a Break character in two ways.

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 RX/DT, cause an RCIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABD bit once the TXIF interrupt is observed.

**FIGURE 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 (RCSTA<7>), is set in order to configure the TX and RX pins to CK (clock) and DT (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CK line. Clock polarity is selected with the TXCKP bit (BAUDCON<4>); setting TXCKP sets the Idle state on CK as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

### 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, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available).

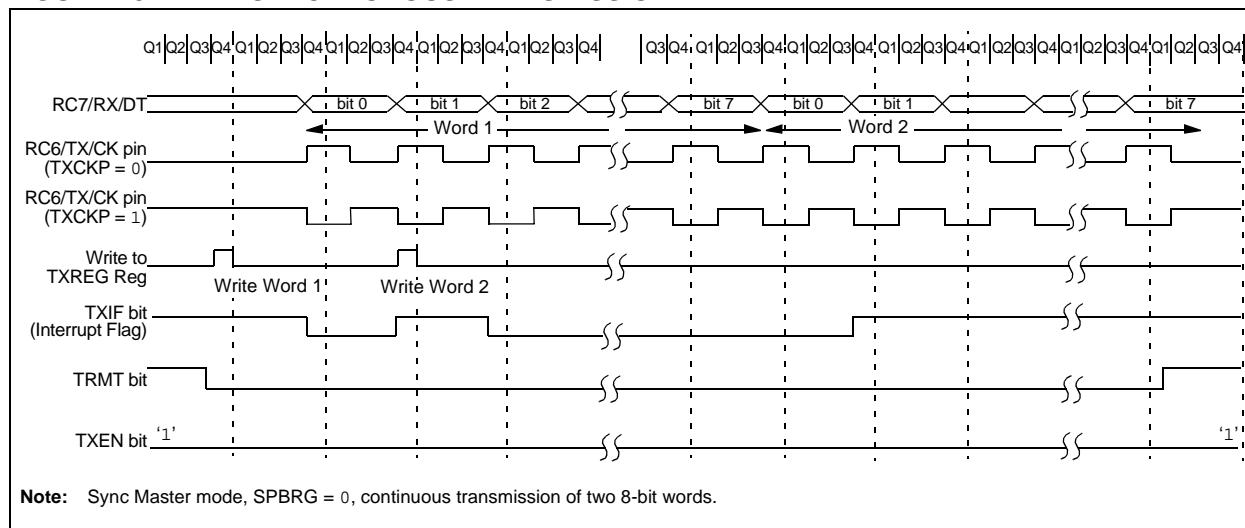
Once the TXREG register transfers the data to the TSR register (occurs in one Tcy), the TXREG is empty and the TXIF flag bit (PIR1<4>) is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXIE (PIE1<4>). TXIF is set regardless of the state of enable bit, TXIE;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

While flag bit, TXIF, indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

To set up a Synchronous Master Transmission:

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

**FIGURE 18-11: SYNCHRONOUS TRANSMISSION**



# PIC18F2525/2620/4525/4620

FIGURE 18-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

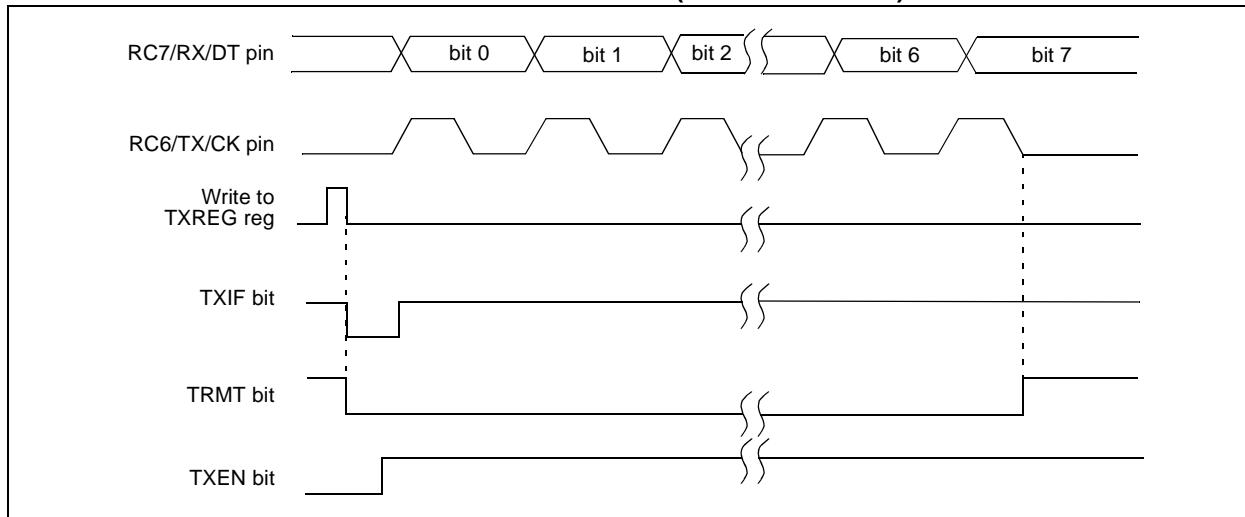


TABLE 18-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

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	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
TXREG	EUSART Transmit Register								51
TXSTA	CSRC	TX9	TXEN	SYNC	SEND <sub>B</sub>	BRGH	TRMT	TX9D	51
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

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

**Note 1:** These bits are unimplemented on 28-pin devices and read as '0'.

### 18.3.2 EUSART SYNCHRONOUS MASTER RECEPTION

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

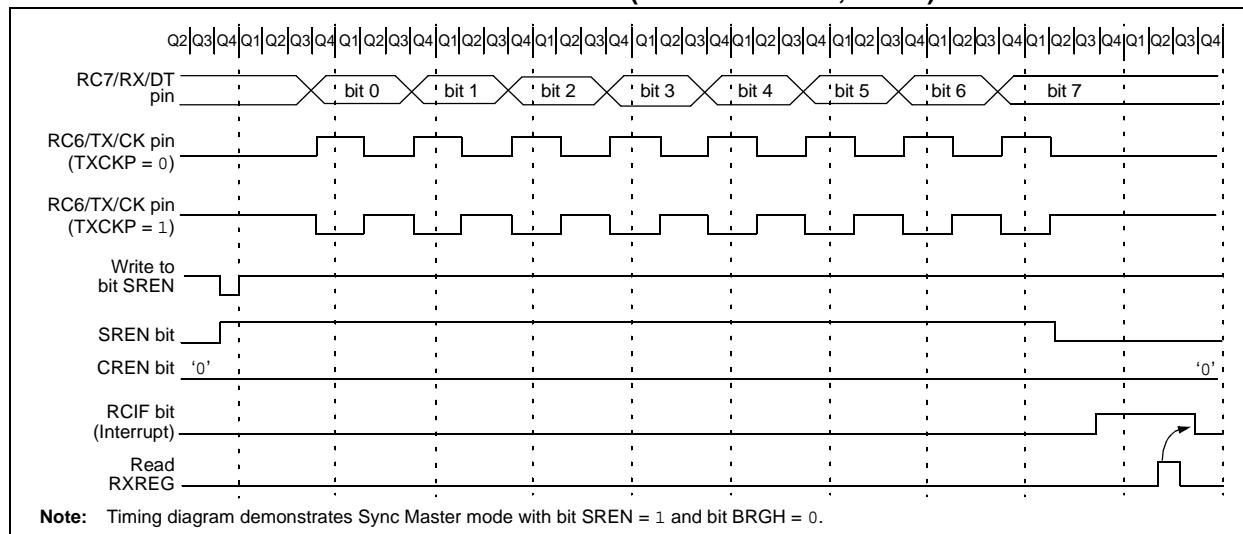
If enable bit, SREN, is set, only a single word is received. If enable bit, CREN, is set, the reception is continuous until CREN is cleared. If both bits are set, then CREN takes precedence.

To set up a Synchronous Master Reception:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.

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

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



**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	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
RCREG	EUSART Receive Register								51
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	51
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

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

**Note 1:** These bits are unimplemented on 28-pin devices and read as '0'.

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

### 18.4.1 EUSART SYNCHRONOUS SLAVE TRANSMISSION

The operation of the Synchronous Master and Slave modes is identical, except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

- The first word will immediately transfer to the TSR register and transmit.
- The second word will remain in the TXREG register.
- Flag bit, TXIF, will not be set.
- When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and flag bit, TXIF, will now be set.
- If enable bit, TXIE, is set, the interrupt will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Transmission:

- 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 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	TMROIE	INT0IE	RBIE	TMROIF	INT0IF	RBIF	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
TXREG	EUSART Transmit Register								51
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	51
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

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

**Note 1:** These bits are unimplemented on 28-pin devices and read as '0'.

## 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 RCREG register. If the RCIE enable bit is set, the interrupt generated will wake the chip from the low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. If interrupts are desired, set enable bit, RCIE.
3. If 9-bit reception is desired, set bit, RX9.
4. To enable reception, set enable bit, CREN.
5. Flag bit, RCIF, will be set when reception is complete. An interrupt will be generated if enable bit, RCIE, was set.
6. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit, CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 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	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	51
RCREG	EUSART Receive Register								51
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	51
BAUDCON	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	51
SPBRGH	EUSART Baud Rate Generator Register High Byte								51
SPBRG	EUSART Baud Rate Generator Register Low Byte								51

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

**Note 1:** These bits are unimplemented on 28-pin devices and read as '0'.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**



# PIC18F2525/2620/4525/4620

## REGISTER 19-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-q <sup>(1)</sup>	R/W-q <sup>(1)</sup>	R/W-q <sup>(1)</sup>
—	—	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 = VSS

bit 4

**VCFG0:** Voltage Reference Configuration bit (VREF+ source)

1 = VREF+ (AN3)

0 = VDD

bit 3-0

**PCFG3:PCFG0:** A/D Port Configuration Control bits:

PCFG3: PCFG0	AN12	AN11	AN10	AN9	AN8	AN7 <sup>(2)</sup>	AN6 <sup>(2)</sup>	AN5 <sup>(2)</sup>	AN4	AN3	AN2	AN1	AN0
0000 <sup>(1)</sup>	A	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	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111 <sup>(1)</sup>	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	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	A
1010	D	D	D	D	D	D	D	A	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	A	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	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	D

A = Analog input

D = Digital I/O

Port + AD Enable

**Note 1:** The POR value of the PCFG bits depends on the value of the PBADEN Configuration bit. When PBADEN = 1, PCFG<2:0> = 000; when PBADEN = 0, PCFG<2:0> = 111.

**2:** AN5 through AN7 are available only on 40/44 pin devices.

Power  
or resets

PCFG0:3  
= 1011  
لعاو  
5 only  
pins

1110  
0x0E

#define

ADC - AN0 - Analog - Functionality

Adc value 10 bits =  $ADCL + (ADCH * 256)$   $\rightarrow$  justified (Right)

**PIC18F2525/2620/4525/4620**

**REGISTER 19-3: ADCON2: A/D CONTROL REGISTER 2**

A/D conversion clock

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

+ AD Format select

bit 6

**Unimplemented:** Read as '0'

bit 5-3

**ACQT2:ACQT0:** 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

**ADCS2:ADCS0:** 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

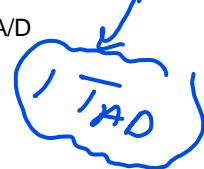
000 = Fosc/2

Setting T<sub>A/D</sub>?

- ex: 8MHz  $\rightarrow$  8,000,000

- CLK some Fosc/16

$$\frac{8\text{MHz}}{16} = 500,000 \rightarrow \frac{1}{500,000} = 2.4\text{s}$$



**Note 1:** If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

\* Acquisition time  $\rightarrow$  time to charge and discharge of capacitors  
 $\rightarrow$  auto (2,4/ $\mu$ sec)  $\rightarrow$  manual (ACQT0:2)

\* 10 bit  $\rightarrow$  11 T<sub>A/D</sub>  $\rightarrow$  time to complete one bit conversion

T<sub>CY</sub>  $\rightarrow$  T<sub>A/D</sub> (11 bit)  $\rightarrow$  (discharge)  $\rightarrow$   
 set Go/Done  $\rightarrow$  set ADRESH/L  $\rightarrow$  ADIF (interrupt)

# PIC18F2525/2620/4525/4620

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (VDD and Vss), or the voltage level on the RA3/AN3/VREF+ and RA2/AN2/VREF-/CVREF pins.

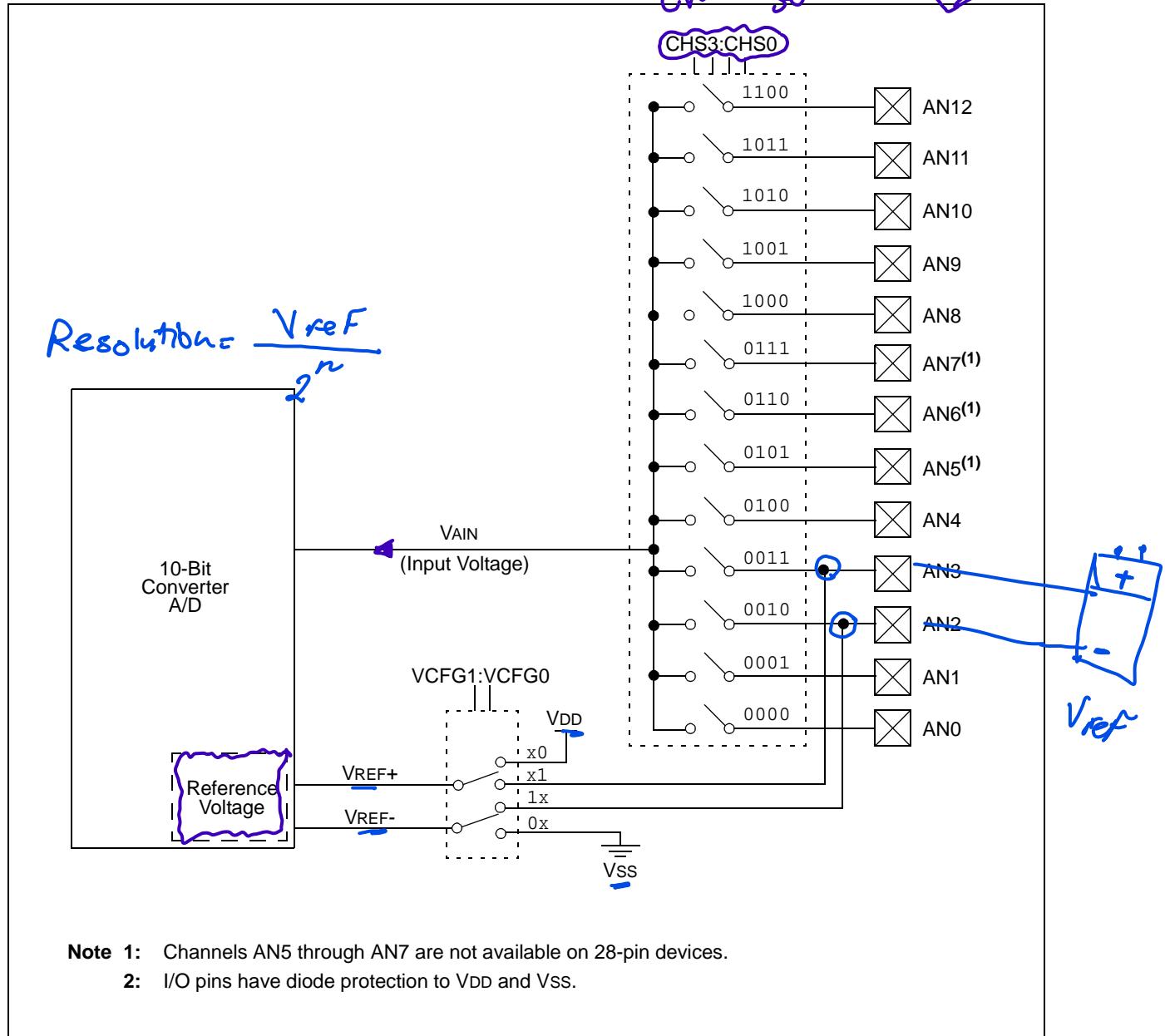
The A/D converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

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 register pair, the GO/DONE bit (ADCON0 register) is cleared and A/D Interrupt Flag bit, ADIF, is set. The block diagram of the A/D module is shown in Figure 19-1.

FIGURE 19-1: A/D BLOCK DIAGRAM



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.

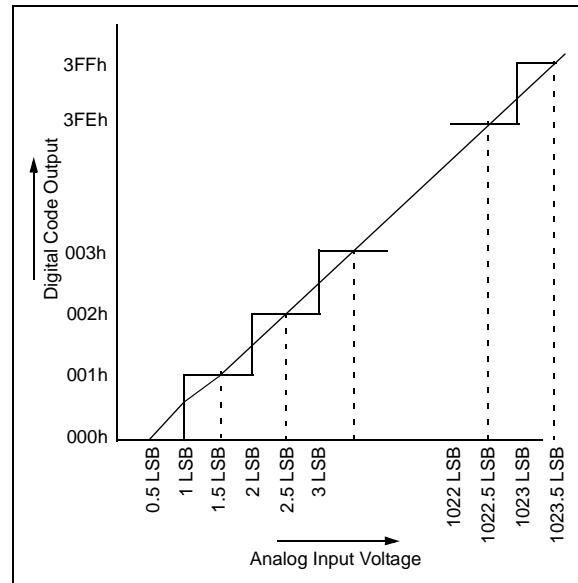
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see **Section 19.1 “A/D Acquisition Requirements”**. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/DONE bit and the actual start of the conversion.

The following steps should be followed to perform an A/D conversion:

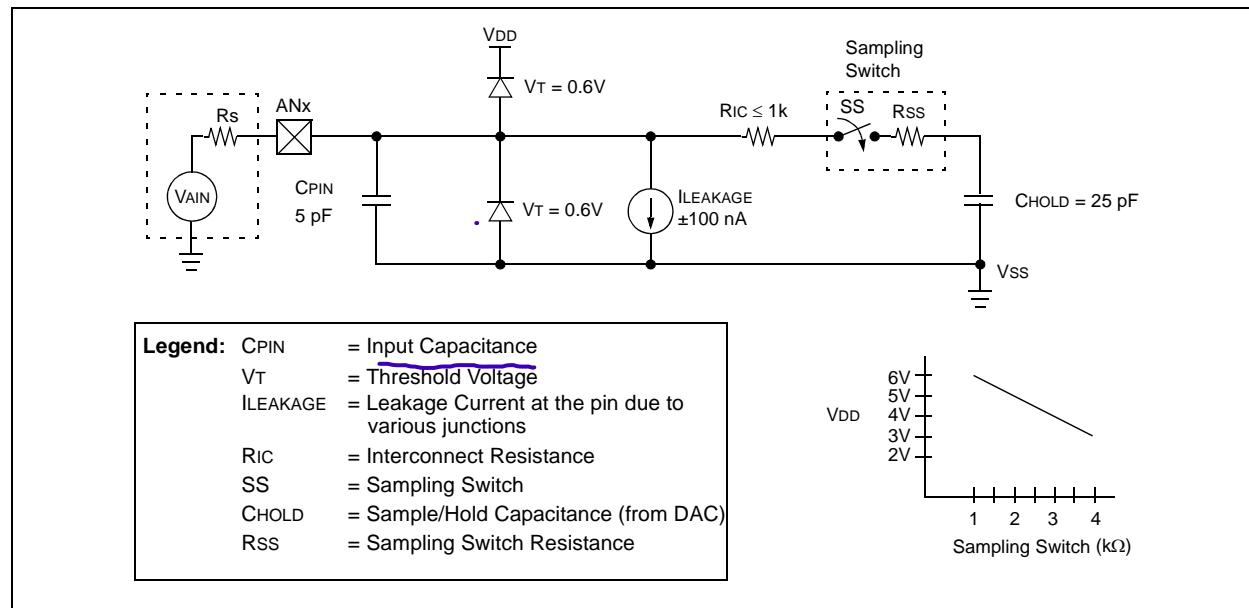
1. Configure the A/D module:
  - Configure analog pins, voltage reference and digital I/O (ADCON1)
  - Select A/D input channel (ADCON0)
  - Select A/D acquisition time (ADCON2)
  - Select A/D conversion clock (ADCON2)
  - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
  - Clear ADIF bit
  - Set ADIE bit
  - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
  - Set GO/DONE bit (ADCON0 register)

5. Wait for A/D conversion to complete, by either:
  - Polling for the GO/DONE bit to be cleared
  - OR
  - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF, if required.
7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2 TAD is required before the next acquisition starts.

**FIGURE 19-2: A/D TRANSFER FUNCTION**



**FIGURE 19-3: ANALOG INPUT MODEL**



# PIC18F2525/2620/4525/4620

## 19.1 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 19-3. 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 19-1 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 19-3 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 19-1: ACQUISITION TIME

$$\begin{aligned} \text{TACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= \underline{\text{TAMP}} + \underline{\text{TC}} + \underline{\text{TCOFF}} \end{aligned}$$

## EQUATION 19-2: 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 19-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$$\begin{aligned} \text{TACQ} &= \text{TAMP} + \text{TC} + \text{TCOFF} \\ \text{TAMP} &= 0.2 \mu\text{s} \\ \text{TCOFF} &= (\text{Temp} - 25^\circ\text{C})(0.02 \mu\text{s}/^\circ\text{C}) \\ &\quad (85^\circ\text{C} - 25^\circ\text{C})(0.02 \mu\text{s}/^\circ\text{C}) \\ &\quad 1.2 \mu\text{s} \\ \text{Temperature coefficient is only required for temperatures} &> 25^\circ\text{C. Below } 25^\circ\text{C, TCOFF} = 0 \text{ ms.} \\ \text{TC} &= -(\text{CHOLD})(\text{RIC} + \text{RSS} + \text{RS}) \ln(1/2047) \\ &\quad -(25 \text{ pF})(1 \text{ k}\Omega + 2 \text{ k}\Omega + 2.5 \text{ k}\Omega) \ln(0.0004883) \\ &\quad 1.05 \mu\text{s} \\ \text{TACQ} &= 0.2 \mu\text{s} + 1 \mu\text{s} + 1.2 \mu\text{s} \\ &\quad \underline{\underline{2.4 \mu\text{s if } \text{TC} \neq 0}} \end{aligned}$$

2,415

James TACQ ↑ sampling freq ↓ Analog Value Could be not accurate

## 19.2 Selecting and Configuring Acquisition Time

The ADCON2 register allows the user to select an acquisition time that occurs each time the 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 ACQT2:ACQT0 bits (ADCON2<5:3>), which provides a range of 2 to 20 TAD. When the GO/DONE bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Since the acquisition time is programmed, there may be no need to wait for an acquisition time between selecting a channel and setting the GO/DONE bit.

Manual acquisition is selected when ACQT2:ACQT0 = 000. When the GO/DONE bit is set, sampling is stopped and a conversion begins. The user is responsible for ensuring the required acquisition time has passed between selecting the desired input channel and setting the GO/DONE bit. This option is also the default Reset state of the ACQT2:ACQT0 bits and is compatible with devices that do not offer programmable acquisition times.

In either case, when the conversion is completed, the GO/DONE bit is cleared, the ADIF flag is set and the A/D begins sampling the currently selected channel again. If an acquisition time is programmed, there is nothing to indicate if the acquisition time has ended or if the conversion has begun.

**TABLE 19-1: TAD vs. DEVICE OPERATING FREQUENCIES**

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	ADCS2:ADCS0	PIC18F2X20/4X20	PIC18LF2X20/4X20 <sup>(4)</sup>
2 Tosc	000	2.86 MHz	1.43 kHz
4 Tosc	100	5.71 MHz	2.86 MHz
8 Tosc	001	11.43 MHz	5.72 MHz
16 Tosc	101	22.86 MHz	11.43 MHz
32 Tosc	010	40.0 MHz	22.86 MHz
64 Tosc	110	40.0 MHz	22.86 MHz
RC <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 1.2  $\mu$ s.

**2:** The RC source has a typical TAD time of 2.5  $\mu$ s.

**3:** For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or the A/D accuracy may be out of specification.

**4:** Low-power (PIC18LFXXXX) devices only.

## 19.3 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 (see parameter 130 for more information).

Table 19-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

## 19.4 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 ACQT2:ACQT0 and ADCS2:ADCS0 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 ACQT2:ACQT0 are set to '000' and a conversion is started, the conversion will be delayed one instruction cycle to allow execution of the SLEEP instruction and entry to Sleep mode. The IDLEN bit (OSCCON<7>) must have already been cleared prior to starting the conversion.

## 19.5 Configuring Analog Port Pins

The ADCON1, TRISA, TRISB and TRISE registers all configure the A/D port pins. The port pins needed as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (V<sub>OH</sub> or V<sub>OL</sub>) will be converted.

The A/D operation is independent of the state of the CHS3:CHS0 bits and the TRIS bits.

- Note 1:** When reading the PORT register, all pins configured as analog input channels will read as cleared (a low level). Pins configured as digital inputs will convert as analog inputs. Analog levels on a digitally configured input will be accurately converted.
- 2:** Analog levels on any pin defined as a digital input may cause the digital input buffer to consume current out of the device's specification limits.
- 3:** The PBADEN bit in Configuration Register 3H configures PORTB pins to reset as analog or digital pins by controlling how the PCFG<3:0> bits in ADCON1 are reset.

## 19.6 A/D Conversions

Figure 19-4 shows the operation of the A/D converter after the GO/DONE bit has been set and the ACQT2:ACQT0 bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 19-5 shows the operation of the A/D converter after the GO/DONE bit has been set and the ACQT2:ACQT0 bits are set to '010' and selecting a 4 TAD acquisition time before the conversion starts.

Clearing the GO/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).

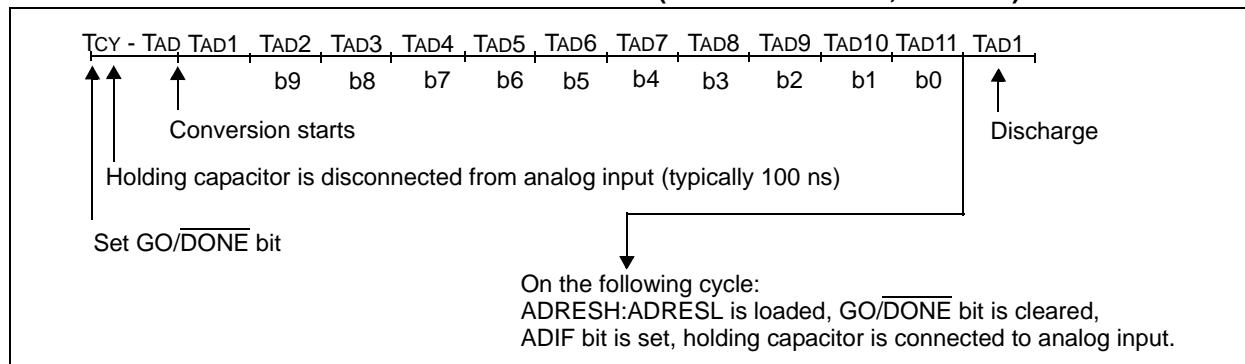
After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

**Note:** The GO/DONE bit should **NOT** be set in the same instruction that turns on the A/D.

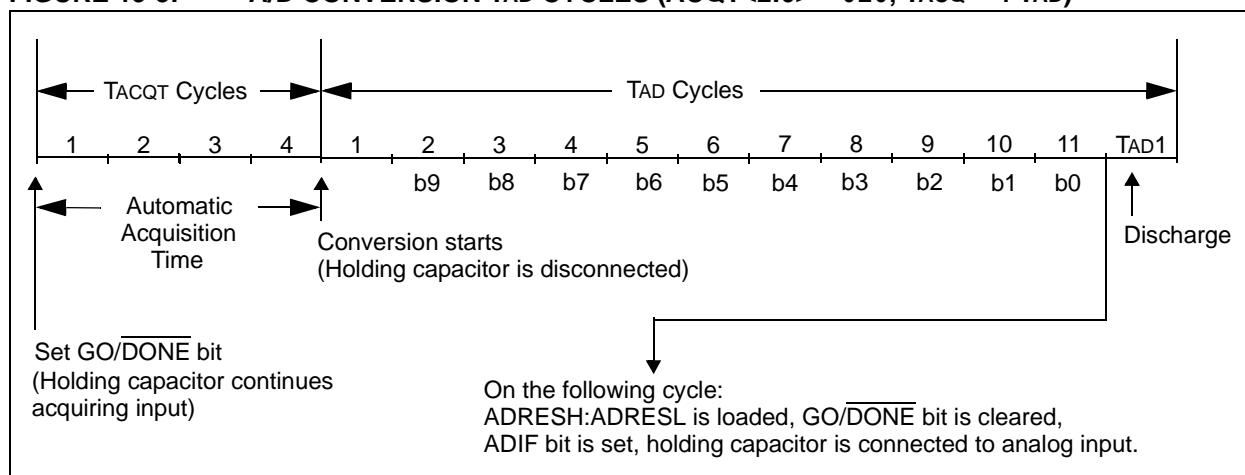
## 19.7 Discharge

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 19-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**



**FIGURE 19-5: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**



# PIC18F2525/2620/4525/4620

## 19.8 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 CCP2M3:CCP2M0 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 19-2: 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	49
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	52
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	52
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	52
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52
ADRESH	A/D Result Register High Byte								51
ADRESL	A/D Result Register Low Byte								51
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	51
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	51
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	51
PORTA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5	RA4	RA3	RA2	RA1	RA0	52
TRISA	TRISA7 <sup>(2)</sup>	TRISA6 <sup>(2)</sup>	PORTA Data Direction Control Register						52
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	52
TRISB	PORTB Data Direction Control Register								52
LATB	PORTB Data Latch Register (Read and Write to Data Latch)								52
PORTE <sup>(4)</sup>	—	—	—	—	RE3 <sup>(3)</sup>	RE2	RE1	RE0	52
TRISE <sup>(4)</sup>	IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0	52
LATE <sup>(4)</sup>	—	—	—	—	—	PORTE Data Latch Register			52

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

**Note 1:** These bits are unimplemented on 28-pin devices; always maintain these bits clear.

**2:** PORTA<7:6> and their direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

**3:** RE3 port bit is available only as an input pin when the MCLRE Configuration bit is '0'.

**4:** These registers are not implemented on 28-pin devices.

## 20.0 COMPARATOR MODULE

The analog comparator module contains two comparators that can be configured in a variety of ways. The inputs can be selected from the analog inputs multiplexed with pins RA0 through RA5, as well as the on-chip voltage reference (see **Section 21.0 “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 20-1) selects the comparator input and output configuration. Block diagrams of the various comparator configurations are shown in Figure 20-1.

### REGISTER 20-1: 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

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7      **C2OUT:** Comparator 2 Output bit

When C2INV = 0:

1 = C2 VIN+ > C2 VIN-  
0 = C2 VIN+ < C2 VIN-

When C2INV = 1:

1 = C2 VIN+ < C2 VIN-  
0 = C2 VIN+ > C2 VIN-

bit 6      **C1OUT:** Comparator 1 Output bit

When C1INV = 0:

1 = C1 VIN+ > C1 VIN-  
0 = C1 VIN+ < C1 VIN-

When C1INV = 1:

1 = C1 VIN+ < C1 VIN-  
0 = C1 VIN+ > C1 VIN-

bit 5      **C2INV:** Comparator 2 Output Inversion bit

1 = C2 output inverted  
0 = C2 output not inverted

bit 4      **C1INV:** Comparator 1 Output Inversion bit

1 = C1 output inverted  
0 = C1 output not inverted

bit 3      **CIS:** Comparator Input Switch bit

When CM2:CM0 = 110:

1 = C1 VIN- connects to RA3/AN3/VREF+  
    C2 VIN- connects to RA2/AN2/VREF-/CVREF  
0 = C1 VIN- connects to RA0/AN0  
    C2 VIN- connects to RA1/AN1

bit 2-0      **CM2:CM0:** Comparator Mode bits

Figure 20-1 shows the Comparator modes and the CM2:CM0 bit settings.

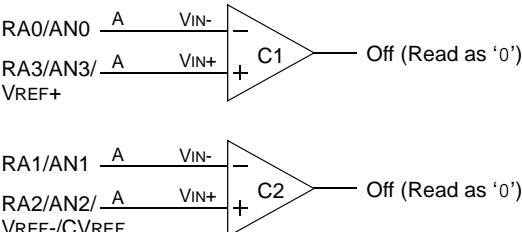
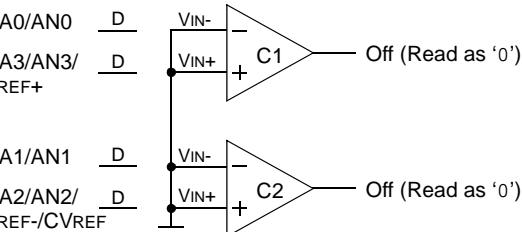
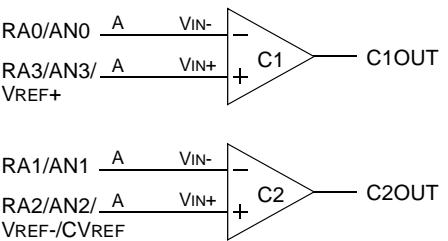
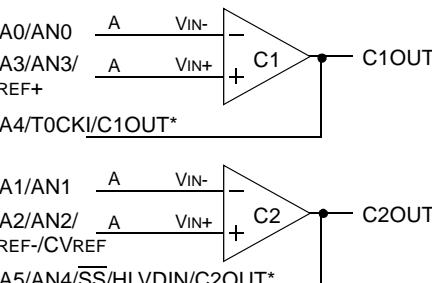
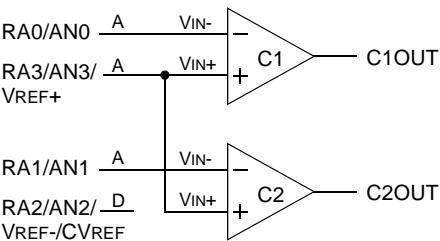
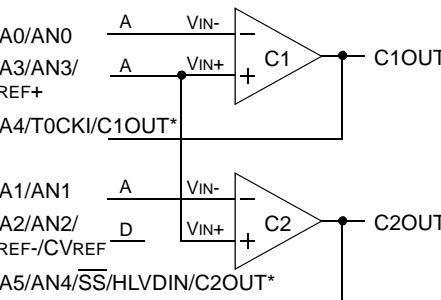
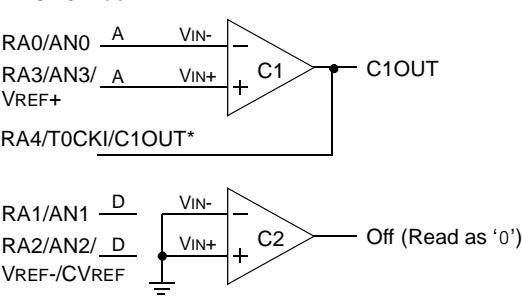
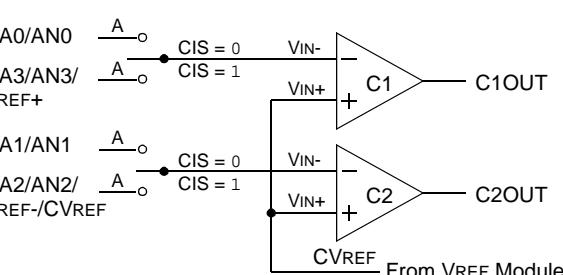
## 20.1 Comparator Configuration

There are eight modes of operation for the comparators, shown in Figure 20-1. Bits, CM2:CM0 of the CMCON register, are used to select these modes. The TRISA register controls the data direction of the comparator pins for each mode. If the Comparator mode is

changed, the comparator output level may not be valid for the specified mode change delay shown in **Section 26.0 “Electrical Characteristics”**.

**Note:** Comparator interrupts should be disabled during a Comparator mode change; otherwise, a false interrupt may occur.

**FIGURE 20-1: COMPARATOR I/O OPERATING MODES**

<p><b>Comparators Reset</b> <b>CM2:CM0 = 000</b></p> 	<p><b>Comparators Off (POR Default Value)</b> <b>CM2:CM0 = 111</b></p> 
<p><b>Two Independent Comparators</b> <b>CM2:CM0 = 010</b></p> 	<p><b>Two Independent Comparators with Outputs</b> <b>CM2:CM0 = 011</b></p> 
<p><b>Two Common Reference Comparators</b> <b>CM2:CM0 = 100</b></p> 	<p><b>Two Common Reference Comparators with Outputs</b> <b>CM2:CM0 = 101</b></p> 
<p><b>One Independent Comparator with Output</b> <b>CM2:CM0 = 001</b></p> 	<p><b>Four Inputs Multiplexed to Two Comparators</b> <b>CM2:CM0 = 110</b></p> 

A = Analog Input, port reads zeros always

D = Digital Input

CIS (CMCON<3>) is the Comparator Input Switch

\* Setting the TRISA<5:4> bits will disable the comparator outputs by configuring the pins as inputs.

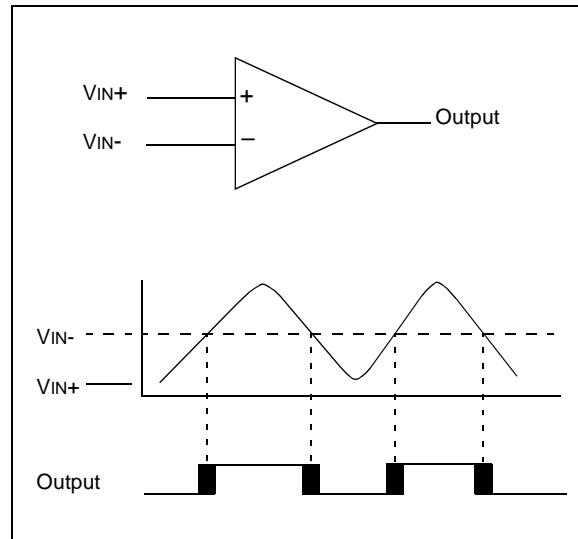
## 20.2 Comparator Operation

A single comparator is shown in Figure 20-2, 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 20-2 represent the uncertainty, due to input offsets and response time.

## 20.3 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 20-2).

**FIGURE 20-2: SINGLE COMPARATOR**



### 20.3.1 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).

### 20.3.2 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 21.0 “Comparator Voltage Reference Module”**.

The internal reference is only available in the mode where four inputs are multiplexed to two comparators ( $CM2:CM0 = 110$ ). In this mode, the internal voltage reference is applied to the  $V_{IN+}$  pin of both comparators.

## 20.4 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 26.0 “Electrical Characteristics”**).

## 20.5 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 RA4 and RA5 I/O pins. When enabled, multiplexors in the output path of the RA4 and RA5 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 20-3 shows the comparator output block diagram.

The TRISA bits will still function as an output enable/disable for the RA4 and RA5 pins while in this mode.

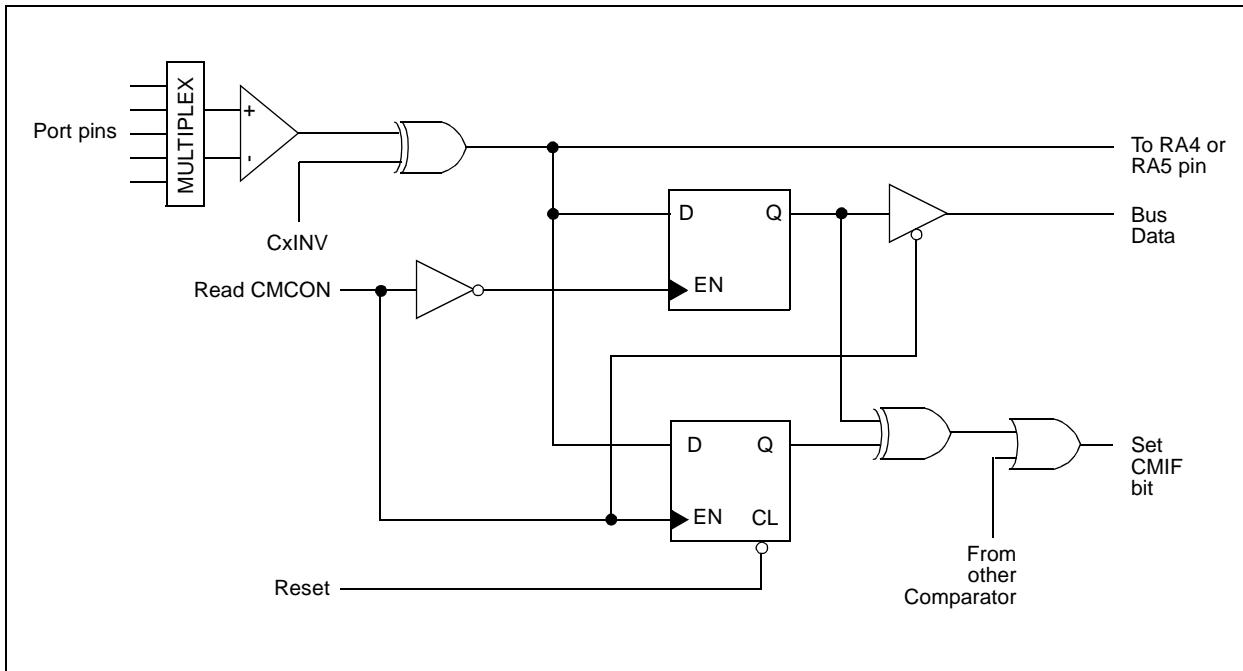
The polarity of the comparator outputs can be changed using the C2INV and C1INV bits (CMCON<5:4>).

**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.

# PIC18F2525/2620/4525/4620

FIGURE 20-3: COMPARATOR OUTPUT BLOCK DIAGRAM



## 20.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that occurred. The CMIF bit (PIR2<6>) is the Comparator Interrupt Flag. The CMIF bit must be reset by clearing it. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

Both the CMIE bit (PIE2<6>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit (INTCON<7>) must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.

**Note:** If a change in the CMCON register (C1OUT or C2OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMIF (PIR registers) interrupt flag may not get set.

The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

- 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.

## 20.7 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. 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 (CM2:CM0 = 111) before entering Sleep. If the device wakes up from Sleep, the contents of the CMCON register are not affected.

## 20.8 Effects of a Reset

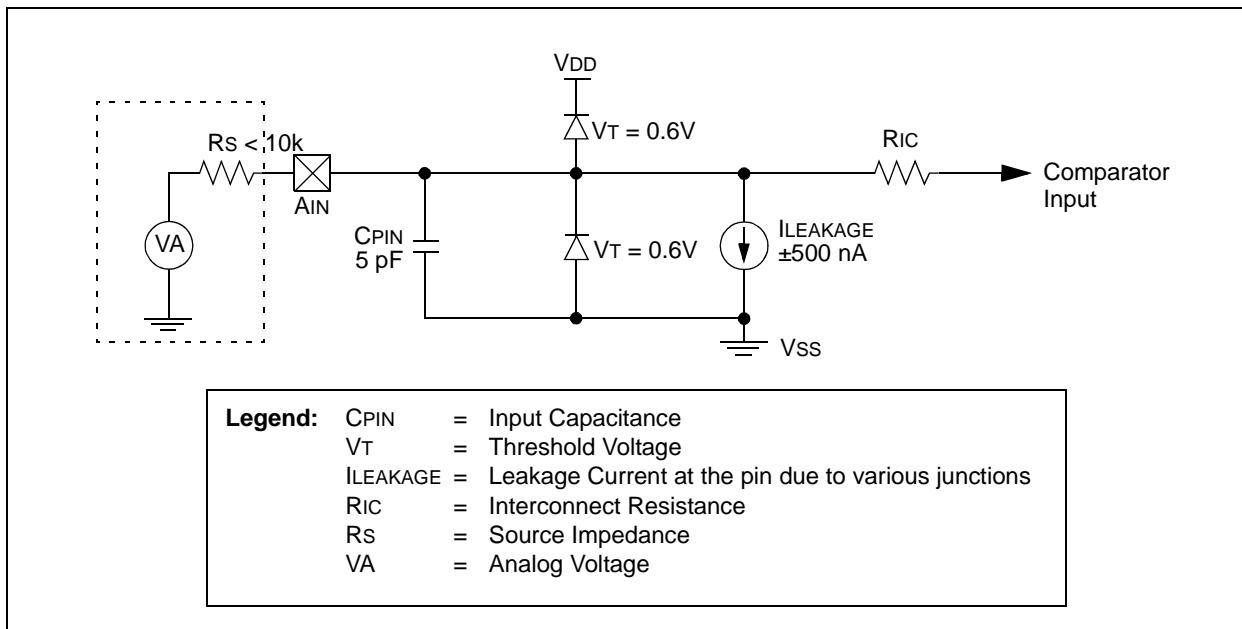
A device Reset forces the CMCON register to its Reset state, causing the comparator modules to be turned off (CM2:CM0 = 111). However, the input pins (RA0 through RA3) are configured as analog inputs by default on device Reset. The I/O configuration for these pins is determined by the setting of the PCFG3:PCFG0 bits (ADCON1<3:0>). Therefore, device current is minimized when analog inputs are present at Reset time.

## 20.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 20-4. 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 k $\Omega$  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 20-4: COMPARATOR ANALOG INPUT MODEL**



**TABLE 20-1: 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	51
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	51
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	52
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52
PORTA	RA7 <sup>(1)</sup>	RA6 <sup>(1)</sup>	RA5	RA4	RA3	RA2	RA1	RA0	52
LATA	LATA7 <sup>(1)</sup>	LATA6 <sup>(1)</sup>	PORTA Data Latch Register (Read and Write to Data Latch)						52
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	PORTA Data Direction Control Register						52

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

**Note 1:** PORTA<7:6> and their direction and latch bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 21.0 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 21-1. 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.

### 21.1 Configuring the Comparator Voltage Reference

The voltage reference module is controlled through the CVRCON register (Register 21-1). 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 (CVR3:CVR0), with one range offering finer resolution. The equations used to calculate the output of the comparator voltage reference are as follows:

If CVRR = 1:

$$CVREF = ((CVR3:CVR0)/24) \times CVRSRC$$

If CVRR = 0:

$$CVREF = (CVRSRC \times 1/4) + (((CVR3:CVR0)/32) \times CVRSRC)$$

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 26-3 in **Section 26.0 “Electrical Characteristics”**).

#### REGISTER 21-1: 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

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

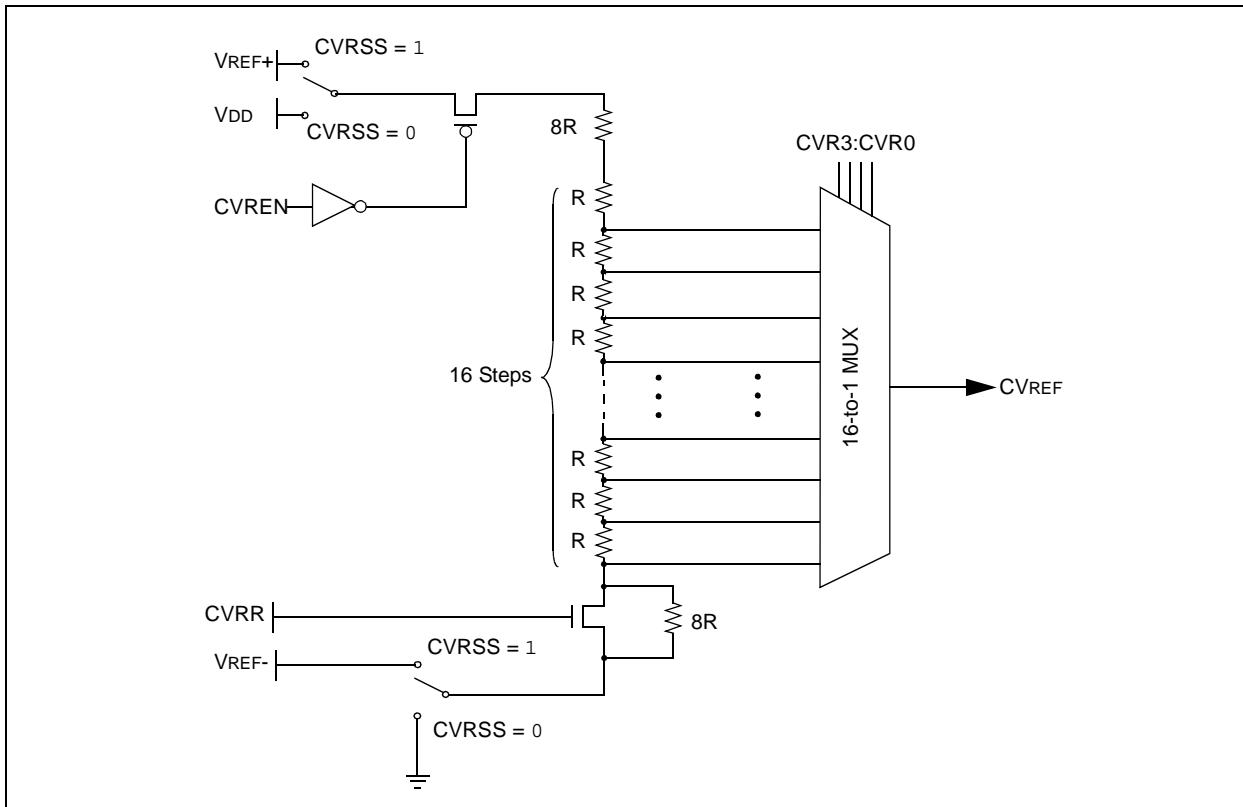
x = Bit is unknown

bit 7	<b>CVREN:</b> Comparator Voltage Reference Enable bit 1 = CVREF circuit powered on 0 = CVREF circuit powered down
bit 6	<b>CVROE:</b> Comparator VREF Output Enable bit <sup>(1)</sup> 1 = CVREF voltage level is also output on the RA2/AN2/VREF-/CVREF pin 0 = CVREF voltage is disconnected from the RA2/AN2/VREF-/CVREF pin
bit 5	<b>CVRR:</b> Comparator VREF Range Selection bit 1 = 0 to 0.667 CVRSRC, with CVRSRC/24 step size (low range) 0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size (high range)
bit 4	<b>CVRSS:</b> Comparator VREF Source Selection bit 1 = Comparator reference source, CVRSRC = (VREF+) – (VREF-) 0 = Comparator reference source, CVRSRC = VDD – VSS
bit 3-0	<b>CVR3:CVR0:</b> Comparator VREF Value Selection bits (0 ≤ (CVR3:CVR0) ≤ 15) <u>When CVRR = 1:</u> CVREF = ((CVR3:CVR0)/24) • (CVRSRC) <u>When CVRR = 0:</u> CVREF = (CVRSRC/4) + (((CVR3:CVR0)/32) • (CVRSRC))

**Note 1:** CVROE overrides the TRISA<2> bit setting.

# PIC18F2525/2620/4525/4620

FIGURE 21-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM



## 21.2 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 21-1) 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 26.0 “Electrical Characteristics”**.

## 21.3 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.

## 21.4 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.

## 21.5 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RA2 pin if the CVROE bit is set. Enabling the voltage reference output onto RA2 when it is configured as a digital input will increase current consumption. Connecting RA2 as a digital output with CVRSS enabled will also increase current consumption.

The RA2 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 21-2 shows an example buffering technique.

# PIC18F2525/2620/4525/4620

FIGURE 21-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE

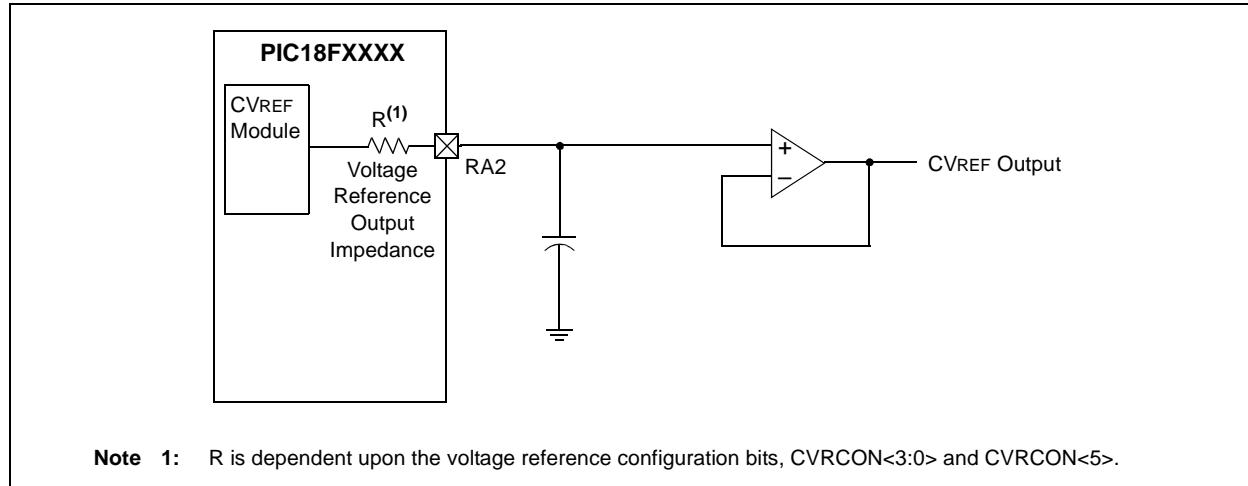


TABLE 21-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	51
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	51
TRISA	TRISA7 <sup>(1)</sup>	TRISA6 <sup>(1)</sup>	PORTA Data Direction Control Register						52

**Legend:** Shaded cells are not used with the comparator voltage reference.

**Note 1:** PORTA pins are enabled based on oscillator configuration.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 22.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

PIC18F2525/2620/4525/4620 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 22-1) 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 22-1.

### REGISTER 22-1: 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	<b>VDIRMAG:</b> Voltage Direction Magnitude Select bit 1 = Event occurs when voltage equals or exceeds trip point (HLVDL3:HLVDL0) 0 = Event occurs when voltage equals or falls below trip point (HLVDL3:HLVDL0)
bit 6	<b>Unimplemented:</b> Read as ‘0’
bit 5	<b>IRVST:</b> 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	<b>HLVDEN:</b> High/Low-Voltage Detect Power Enable bit 1 = HLVD enabled 0 = HLVD disabled
bit 3-0	<b>HLVDL3:HLVDL0:</b> Voltage Detection Limit bits <sup>(1)</sup> 1111 = External analog input is used (input comes from the HLVDIN pin) 1110 = Maximum setting • • • 0000 = Minimum setting

**Note 1:** See Table 26-4 in **Section 26.0 “Electrical Characteristics”** for the specifications.

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.

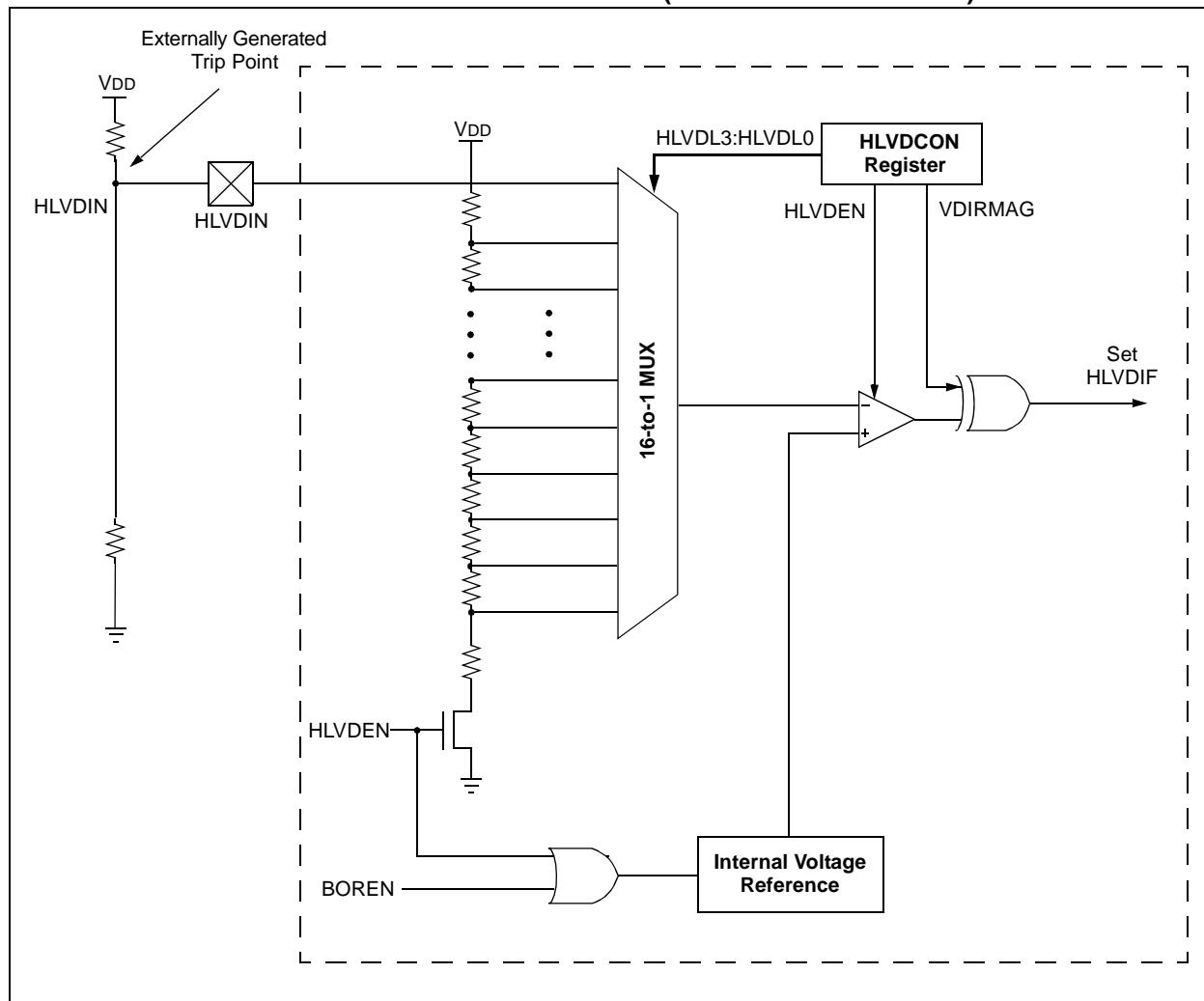
## 22.1 Operation

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 HLVDL3:HLVDL0 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 HLVDL3:HLVDL0 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 22-1: HLVD MODULE BLOCK DIAGRAM (WITH EXTERNAL INPUT)**



## 22.2 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 HLVDL3:HLVDL0 bits that selects 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.

## 22.3 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.

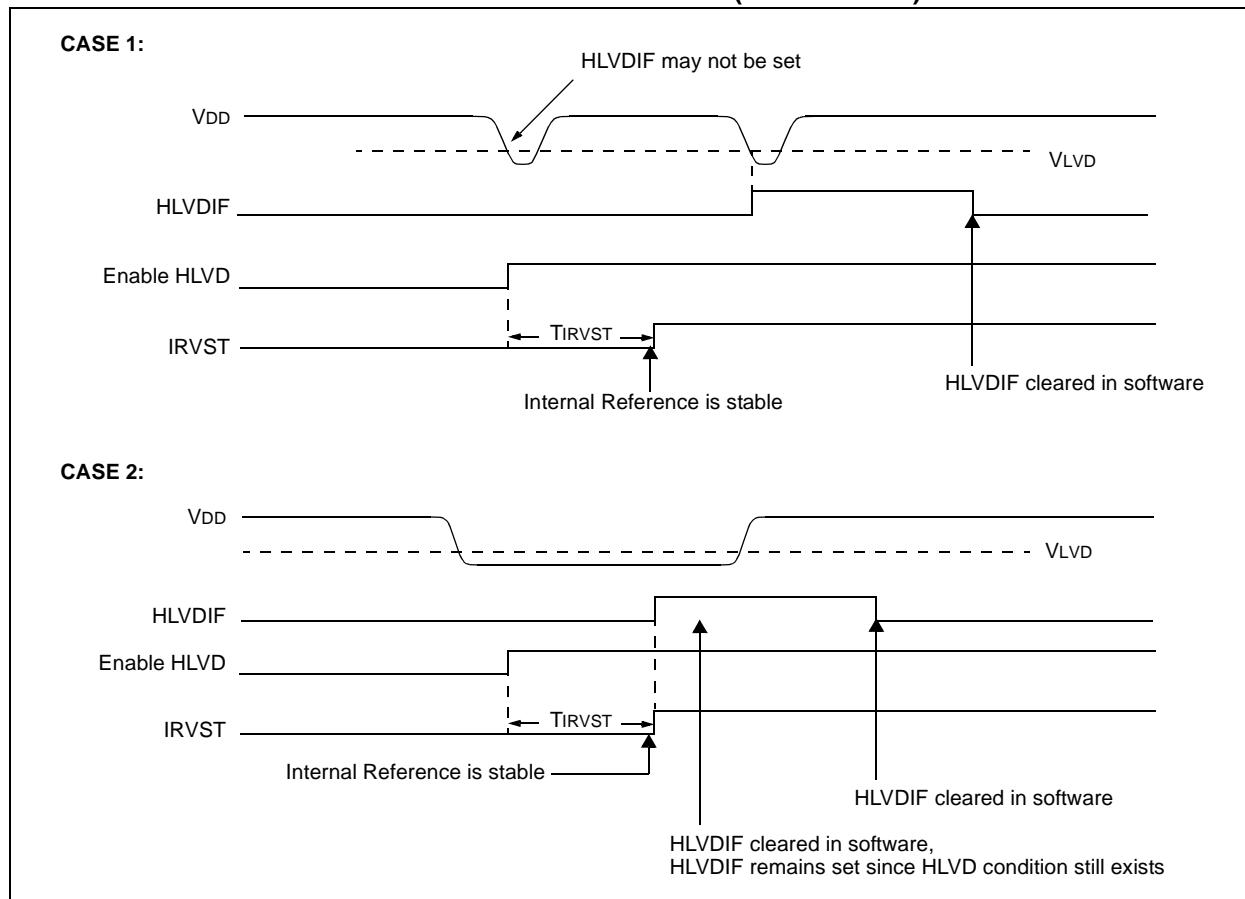
Depending on the application, the HLVD module does not need to be operating constantly. To decrease the current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After doing the check, the HLVD module may be disabled.

## 22.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification parameter D420, may be used by other internal circuitry, such as the Programmable Brown-out Reset. If the HLVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time, TIRVST, is an interval that is independent of device clock speed. It is specified in electrical specification parameter 36.

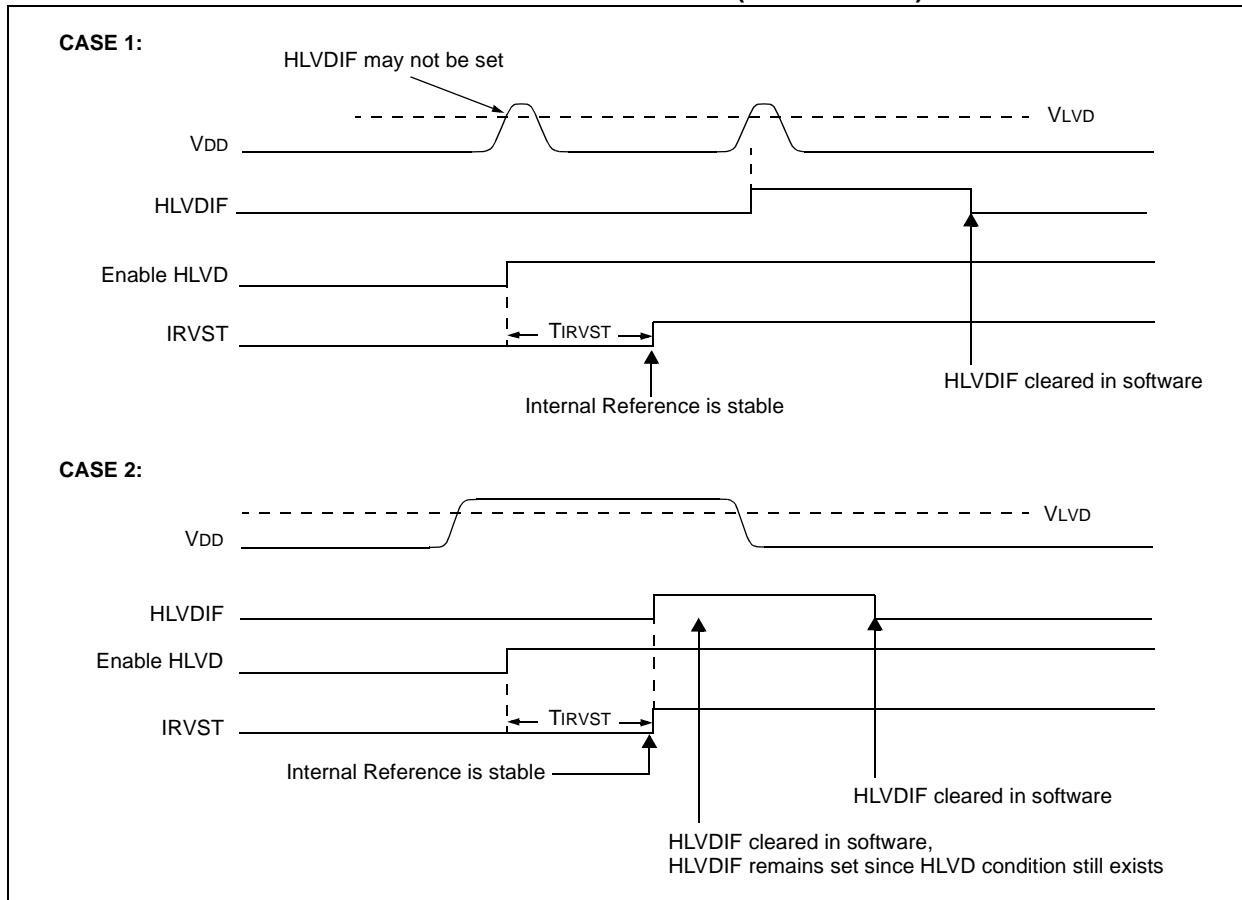
The HLVD interrupt flag is not enabled until TIRVST has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval. Refer to Figure 22-2 or Figure 22-3.

**FIGURE 22-2: LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 0)**



# PIC18F2525/2620/4525/4620

FIGURE 22-3: HIGH-VOLTAGE DETECT OPERATION (VDIRMAG = 1)

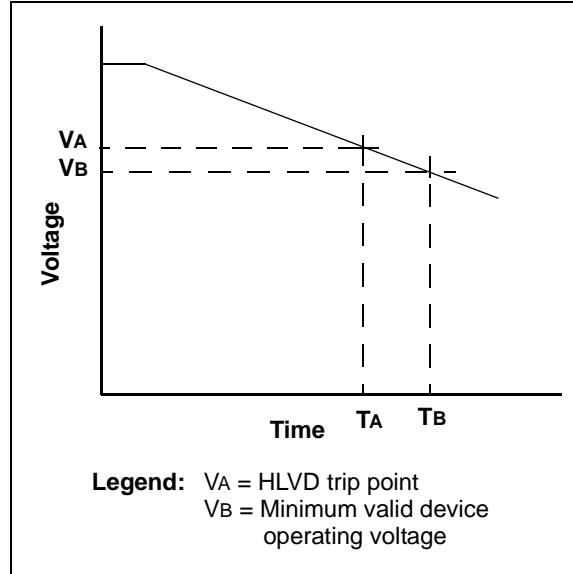


## 22.5 Applications

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 a Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a high-voltage detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 22-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage  $V_A$ , the HLVD logic generates an interrupt at time  $T_A$ . 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  $T_B$ . The HLVD, thus, would give the application a time window, represented by the difference between  $T_A$  and  $T_B$ , to safely exit.

FIGURE 22-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION



## 22.6 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.

## 22.7 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

**TABLE 22-1: 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	50
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	49
PIR2	OSCFIF	CMIF	—	EEIF	BCLIF	HLVDIF	TMR3IF	CCP2IF	52
PIE2	OSCFIE	CMIE	—	EEIE	BCLIE	HLVDIE	TMR3IE	CCP2IE	52
IPR2	OSCFIP	CMIP	—	EEIP	BCLIP	HLVDIP	TMR3IP	CCP2IP	52

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the HLVD module.

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 23.0 SPECIAL FEATURES OF THE CPU

PIC18F2525/2620/4525/4620 devices include several features intended to maximize reliability and minimize cost through elimination of external components. These are:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Fail-Safe Clock Monitor
- Two-Speed Start-up
- Code Protection
- ID Locations
- In-Circuit Serial Programming

The oscillator can be configured for the application depending on frequency, power, accuracy and cost. All of the options are discussed in detail in **Section 2.0 “Oscillator Configurations”**.

A complete discussion of device Resets and interrupts is available in previous sections of this data sheet.

In addition to their Power-up and Oscillator Start-up Timers provided for Resets, PIC18F2525/2620/4525/4620 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.

### 23.1 Configuration Bits

The Configuration bits can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh), which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The WR bit in the EECON1 register starts a self-timed write to the Configuration register. In normal operation mode, a TBLWT instruction, with the TBLPTR pointing to the Configuration register, sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a TBLWT instruction can write a '1' or a '0' into the cell. For additional details on Flash programming, refer to **Section 7.5 “Writing to Flash Program Memory”**.

**TABLE 23-1: 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	PWRTE	---1 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTE	---1 1111
300005h	CONFIG3H	MCLRE	—	—	—	—	LPT1OSC	PBADEN	CCP2MX	1--- -011
300006h	CONFIG4L	DEBUG	XINST	—	—	—	LVP	—	STVREN	10-- -1-1
300008h	CONFIG5L	—	—	—	—	CP3 <sup>(1)</sup>	CP2	CP1	CP0	---- 1111
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	WRT3 <sup>(1)</sup>	WRT2	WRT1	WRT0	---- 1111
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	EBTR3 <sup>(1)</sup>	EBTR2	EBTR1	EBTR0	---- 1111
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	xxxx xxxx <sup>(2)</sup>
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 1100 <sup>(2)</sup>

**Legend:** x = unknown, u = unchanged, - = unimplemented. Shaded cells are unimplemented, read as '0'.

**Note 1:** Unimplemented in PIC18FX525 devices; maintain this bit set.

**2:** See Register 23-12 and Register 23-13 for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

# PIC18F2525/2620/4525/4620

---

---

## REGISTER 23-1: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

R/P-0	R/P-0	U-0	U-0	R/P-0	R/P-1	R/P-1	R/P-1
IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0
bit 7							bit 0

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7 **IESO:** Internal/External Oscillator Switchover bit

1 = Oscillator Switchover mode enabled

0 = Oscillator Switchover mode disabled

bit 6 **FCMEN:** Fail-Safe Clock Monitor Enable bit

1 = Fail-Safe Clock Monitor enabled

0 = Fail-Safe Clock Monitor disabled

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **FOSC3:FOSC0:** Oscillator Selection bits

11xx = External RC oscillator, CLKO function on RA6

101x = External RC oscillator, CLKO function on RA6

1001 = Internal oscillator block, CLKO function on RA6, port function on RA7

1000 = Internal oscillator block, port function on RA6 and RA7

0111 = External RC oscillator, port function on RA6

0110 = HS oscillator, PLL enabled (Clock Frequency = 4 x FOSC1)

0101 = EC oscillator, port function on RA6

0100 = EC oscillator, CLKO function on RA6

0011 = External RC oscillator, CLKO function on RA6

0010 = HS oscillator

0001 = XT oscillator

0000 = LP oscillator

# PIC18F2525/2620/4525/4620

## REGISTER 23-2: 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 <sup>(1)</sup>	BORV0 <sup>(1)</sup>	BOREN1 <sup>(2)</sup>	BOREN0 <sup>(2)</sup>	<u>PWRTE</u> N <sup>(2)</sup>
bit 7							

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7-5	<b>Unimplemented:</b> Read as '0'
bit 4-3	<b>BORV1:BORV0:</b> Brown-out Reset Voltage bits <sup>(1)</sup> 11 = Minimum setting • • • 00 = Maximum setting
bit 2-1	<b>BOREN1:BOREN0:</b> Brown-out Reset Enable bits <sup>(2)</sup> 11 = Brown-out Reset enabled in hardware only (SBOREN is disabled) 10 = Brown-out Reset enabled in hardware only and disabled in Sleep mode (SBOREN is disabled) 01 = Brown-out Reset enabled and controlled by software (SBOREN is enabled) 00 = Brown-out Reset disabled in hardware and software
bit 0	<b>PWRTE</b> N: Power-up Timer Enable bit <sup>(2)</sup> 1 = PWRT disabled 0 = PWRT enabled

**Note 1:** See **Section 26.1 “DC Characteristics: Supply Voltage”** for specifications.

**2:** The Power-up Timer is decoupled from Brown-out Reset, allowing these features to be independently controlled.

# PIC18F2525/2620/4525/4620

## REGISTER 23-3: 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							

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

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

bit 4-1      **WDTPS3:WDTPS0:** Watchdog Timer Postscale Select bits

1111 = 1:32,768

1110 = 1:16,384

1101 = 1:8,192

1100 = 1:4,096

1011 = 1:2,048

1010 = 1:1,024

1001 = 1:512

1000 = 1:256

0111 = 1:128

0110 = 1:64

0101 = 1:32

0100 = 1:16

0011 = 1:8

0010 = 1:4

0001 = 1:2

0000 = 1:1

bit 0      **WDTEN:** Watchdog Timer Enable bit

1 = WDT enabled

0 = WDT disabled (control is placed on the SWDTEN bit)

# PIC18F2525/2620/4525/4620

## REGISTER 23-4: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

R/P-1	U-0	U-0	U-0	U-0	R/P-0	R/P-1	R/P-1
MCLRE	—	—	—	—	LPT1OSC	PBADEN	CCP2MX
bit 7	bit 0						

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7	<b>MCLRE:</b> <u>MCLR</u> Pin Enable bit 1 = <u>MCLR</u> pin enabled; RE3 input pin disabled 0 = RE3 input pin enabled; <u>MCLR</u> disabled
bit 6-3	<b>Unimplemented:</b> Read as '0'
bit 2	<b>LPT1OSC:</b> Low-Power Timer1 Oscillator Enable bit 1 = Timer1 configured for low-power operation 0 = Timer1 configured for higher power operation
bit 1	<b>PBADEN:</b> PORTB A/D Enable bit (Affects ADCON1 Reset state. ADCON1 controls PORTB<4:0> pin configuration.) 1 = PORTB<4:0> pins are configured as analog input channels on Reset 0 = PORTB<4:0> pins are configured as digital I/O on Reset
bit 0	<b>CCP2MX:</b> CCP2 MUX bit 1 = CCP2 input/output is multiplexed with RC1 0 = CCP2 input/output is multiplexed with RB3

## REGISTER 23-5: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/P-1	R/P-0	U-0	U-0	U-0	R/P-1	U-0	R/P-1
DEBUG	XINST	—	—	—	LVP	—	STVREN
bit 7	bit 0						

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7	<b>DEBUG:</b> Background Debugger Enable bit 1 = Background debugger disabled, RB6 and RB7 configured as general purpose I/O pins 0 = Background debugger enabled, RB6 and RB7 are dedicated to In-Circuit Debug
bit 6	<b>XINST:</b> Extended Instruction Set Enable bit 1 = Instruction set extension and Indexed Addressing mode enabled 0 = Instruction set extension and Indexed Addressing mode disabled (Legacy mode)
bit 5-3	<b>Unimplemented:</b> Read as '0'
bit 2	<b>LVP:</b> Single-Supply ICSP™ Enable bit 1 = Single-Supply ICSP enabled 0 = Single-Supply ICSP disabled
bit 1	<b>Unimplemented:</b> Read as '0'
bit 0	<b>STVREN:</b> Stack Full/Underflow Reset Enable bit 1 = Stack full/underflow will cause Reset 0 = Stack full/underflow will not cause Reset

# PIC18F2525/2620/4525/4620

## REGISTER 23-6: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	CP3 <sup>(1)</sup>	CP2	CP1	CP0
bit 7				bit 0			

### Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7-4      **Unimplemented:** Read as '0'

bit 3      **CP3:** Code Protection bit<sup>(1)</sup>

1 = Block 3 (006000-007FFFh) not code-protected

0 = Block 3 (006000-007FFFh) code-protected

bit 2      **CP2:** Code Protection bit

1 = Block 2 (004000-005FFFh) not code-protected

0 = Block 2 (004000-005FFFh) code-protected

bit 1      **CP1:** Code Protection bit

1 = Block 1 (002000-003FFFh) not code-protected

0 = Block 1 (002000-003FFFh) code-protected

bit 0      **CP0:** Code Protection bit

1 = Block 0 (000800-001FFFh) not code-protected

0 = Block 0 (000800-001FFFh) code-protected

**Note 1:** Unimplemented in PIC18FX525 devices; maintain this bit set.

## REGISTER 23-7: CONFIG5H: CONFIGURATION REGISTER 5 HIGH (BYTE ADDRESS 300009h)

R/C-1	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
CPD	CPB	—	—	—	—	—	—
bit 7				bit 0			

### Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7      **CPD:** Data EEPROM Code Protection bit

1 = Data EEPROM not code-protected

0 = Data EEPROM code-protected

bit 6      **CPB:** Boot Block Code Protection bit

1 = Boot block (000000-0007FFFh) not code-protected

0 = Boot block (000000-0007FFFh) code-protected

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

# PIC18F2525/2620/4525/4620

## REGISTER 23-8: CONFIG6L: CONFIGURATION REGISTER 6 LOW (BYTE ADDRESS 30000Ah)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	WRT3 <sup>(1)</sup>	WRT2	WRT1	WRT0
bit 7							bit 0

### Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7-4	<b>Unimplemented:</b> Read as '0'
bit 3	<b>WRT3:</b> Write Protection bit <sup>(1)</sup> 1 = Block 3 (006000-007FFFFh) not write-protected 0 = Block 3 (006000-007FFFFh) write-protected
bit 2	<b>WRT2:</b> Write Protection bit 1 = Block 2 (004000-005FFFFh) not write-protected 0 = Block 2 (004000-005FFFFh) write-protected
bit 1	<b>WRT1:</b> Write Protection bit 1 = Block 1 (002000-003FFFFh) not write-protected 0 = Block 1 (002000-003FFFFh) write-protected
bit 0	<b>WRT0:</b> Write Protection bit 1 = Block 0 (000800-001FFFFh) not write-protected 0 = Block 0 (000800-001FFFFh) write-protected

**Note 1:** Unimplemented in PIC18FX525 devices; maintain this bit set.

## REGISTER 23-9: CONFIG6H: CONFIGURATION REGISTER 6 HIGH (BYTE ADDRESS 30000Bh)

R/C-1	R/C-1	R/C-1	U-0	U-0	U-0	U-0	U-0
WRTD	WRTB	WRTC <sup>(1)</sup>	—	—	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7	<b>WRTD:</b> Data EEPROM Write Protection bit 1 = Data EEPROM not write-protected 0 = Data EEPROM write-protected
bit 6	<b>WRTB:</b> Boot Block Write Protection bit 1 = Boot block (000000-0007FFh) not write-protected 0 = Boot block (000000-0007FFh) write-protected
bit 5	<b>WRTC:</b> Configuration Register Write Protection bit <sup>(1)</sup> 1 = Configuration registers (300000-3000FFh) not write-protected 0 = Configuration registers (300000-3000FFh) write-protected
bit 4-0	<b>Unimplemented:</b> Read as '0'

**Note 1:** This bit is read-only in normal execution mode; it can be written only in Program mode.

# PIC18F2525/2620/4525/4620

## REGISTER 23-10: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 30000Ch)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	EBTR3 <sup>(1)</sup>	EBTR2	EBTR1	EBTR0
bit 7	bit 0						

### Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7-4      **Unimplemented:** Read as '0'

bit 3      **EBTR3:** Table Read Protection bit<sup>(1)</sup>

1 = Block 3 (006000-007FFFh) not protected from table reads executed in other blocks

0 = Block 3 (006000-007FFFh) protected from table reads executed in other blocks

bit 2      **EBTR2:** Table Read Protection bit

1 = Block 2 (004000-005FFFh) not protected from table reads executed in other blocks

0 = Block 2 (004000-005FFFh) protected from table reads executed in other blocks

bit 1      **EBTR1:** Table Read Protection bit

1 = Block 1 (002000-003FFFh) not protected from table reads executed in other blocks

0 = Block 1 (002000-003FFFh) protected from table reads executed in other blocks

bit 0      **EBTR0:** Table Read Protection bit

1 = Block 0 (000800-001FFFh) not protected from table reads executed in other blocks

0 = Block 0 (000800-001FFFh) protected from table reads executed in other blocks

**Note 1:** Unimplemented in PIC18FX525 devices; maintain this bit set.

## REGISTER 23-11: CONFIG7H: CONFIGURATION REGISTER 7 HIGH (BYTE ADDRESS 30000Dh)

U-0	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
—	EBTRB	—	—	—	—	—	—
bit 7	bit 0						

### Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

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

bit 6      **EBTRB:** Boot Block Table Read Protection bit

1 = Boot block (000000-0007FFh) not protected from table reads executed in other blocks

0 = Boot block (000000-0007FFh) protected from table reads executed in other blocks

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

# PIC18F2525/2620/4525/4620

## REGISTER 23-12: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F2525/2620/4525/4620

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
bit 7							

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7-5      **DEV2:DEV0:** Device ID bits

000 = PIC18F4620

010 = PIC18F4525

100 = PIC18F2620

110 = PIC18F2525

bit 4-0      **REV4:REV0:** Revision ID bits

These bits are used to indicate the device revision.

## REGISTER 23-13: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F2525/2620/4525/4620

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							

### Legend:

R = Read-only bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

u = Unchanged from programmed state

bit 7-0      **DEV10:DEV3:** Device ID bits<sup>(1)</sup>

These bits are used with the DEV2:DEV0 bits in Device ID Register 1 to identify the part number.

0000 1100 = PIC18F2525/2620/4525/4620 devices

**Note 1:** These values for DEV10:DEV3 may be shared with other devices. The specific device is always identified by using the entire DEV10:DEV0 bit sequence.

# **PIC18F2525/2620/4525/4620**

## 23.2 Watchdog Timer (WDT)

For PIC18F2525/2620/4525/4620 devices, the WDT is driven by the INTRC source. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by bits in Configuration Register 2H. Available periods range from 4 ms to 131.072 seconds (2.18 minutes). The WDT and postscaler are cleared when any of the following events occur: a SLEEP or CLRWD<sub>T</sub> instruction is executed, the IRCF bits (OSCCON<6:4>) are changed or a clock failure has occurred.

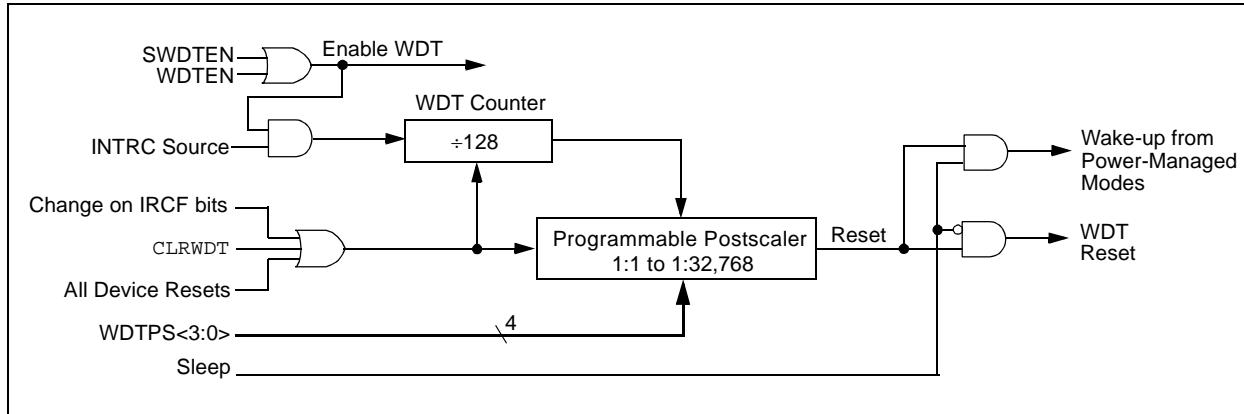
**Note 1:** The CLRWD<sub>T</sub> and SLEEP instructions clear the WDT and postscaler counts when executed.

- 2: Changing the setting of the IRCF bits (OSCCON<6:4>) clears the WDT and postscaler counts.
  - 3: When a CLRWD<sub>T</sub> instruction is executed, the postscaler count will be cleared.

### 23.2.1 CONTROL REGISTER

Register 23-14 shows the WDTCON register. This is a readable and writable register which contains a control bit that allows software to override the WDT enable Configuration bit, but only if the Configuration bit has disabled the WDT.

## FIGURE 23-1: WDT BLOCK DIAGRAM



# PIC18F2525/2620/4525/4620

## REGISTER 23-14: 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

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-1      **Unimplemented:** Read as '0'

bit 0      **SWDTEN:** Software Controlled Watchdog Timer Enable bit<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 23-2: 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 <sup>(1)</sup>	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	50
WDTCON	—	—	—	—	—	—	—	SWDTEN	50

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

**Note 1:** The SBOREN bit is only available when the BOREN1:BOREN0 Configuration bits = 01; otherwise, it is disabled and reads as '0'. See **Section 4.4 "Brown-out Reset (BOR)"**.

## 23.3 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 INTOSC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO Configuration bit.

Two-Speed Start-up should be enabled only if the primary oscillator mode is LP, XT, HS or HSPLL (Crystal-Based modes). Other sources do not require an 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, IRCF2:IRCF0, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF2:IRCF0 bits prior to entering Sleep mode.

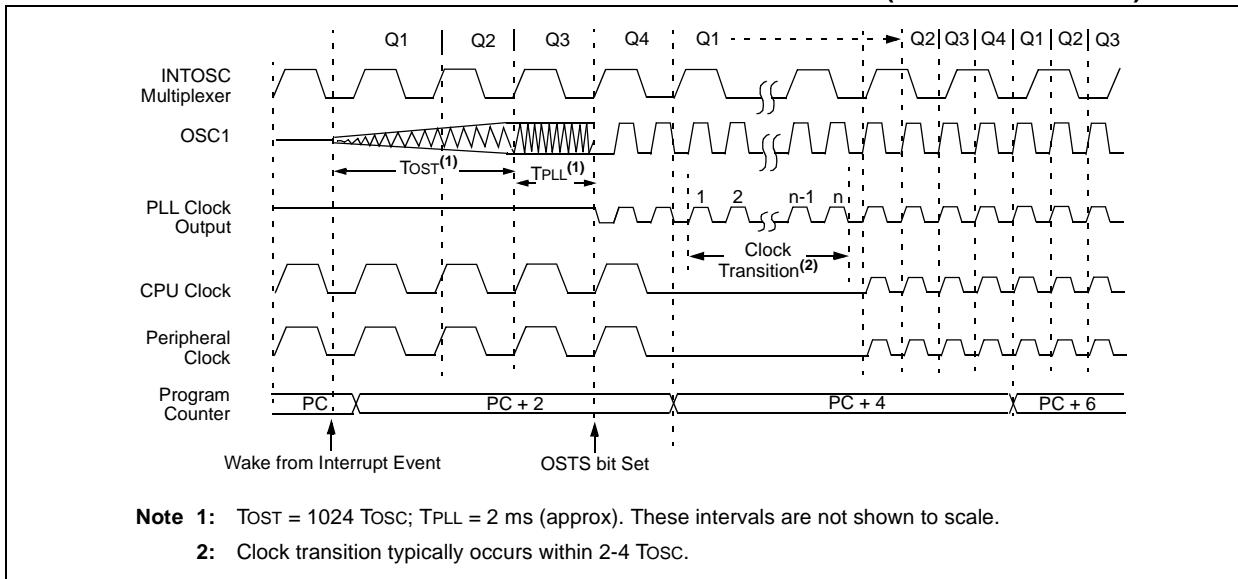
In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

### 23.3.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTOSC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power-managed modes, including multiple SLEEP instructions (refer to **Section 3.1.4 “Multiple Sleep Commands”**). In practice, this means that user code can change the SCS1:SCS0 bit settings or issue SLEEP instructions before the OST times out. This would allow an application to briefly wake-up, perform routine “housekeeping” tasks and return to Sleep before the device starts to operate from the primary oscillator.

User code can also check if the primary clock source is currently providing the device clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

**FIGURE 23-2: TIMING TRANSITION FOR TWO-SPEED START-UP (INTOSC TO HSPLL)**

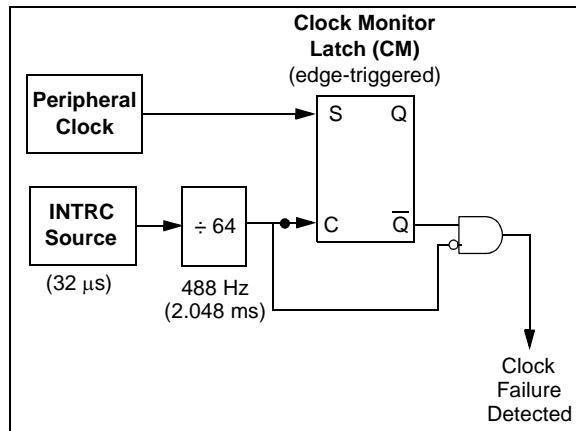


## 23.4 Fail-Safe Clock Monitor

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 23-3) 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 23-3: 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 23-4). This causes the following:

- the FSCM generates an oscillator fail interrupt by setting bit, OSCFIF (PIR2<7>);
- the device clock source is switched to the internal oscillator block (OSCCON is not updated to show the current clock source – this is the fail-safe condition); and
- the WDT is reset.

During switchover, the postscaler frequency from the internal oscillator block may not be sufficiently stable for timing sensitive applications. In these cases, it may be desirable to select another clock configuration and enter an alternate power-managed mode. This can be done to attempt a partial recovery or execute a controlled shutdown. See **Section 3.1.4 “Multiple Sleep Commands”** and **Section 23.3.1 “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, IRCF2:IRCF0, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF2:IRCF0 bits prior to entering Sleep mode.

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.

### 23.4.1 FSCM AND THE WATCHDOG TIMER

Both the FSCM and the WDT are clocked by the INTRC oscillator. Since the WDT operates with a separate divider and counter, disabling the WDT has no effect on the operation of the INTRC oscillator when the FSCM is enabled.

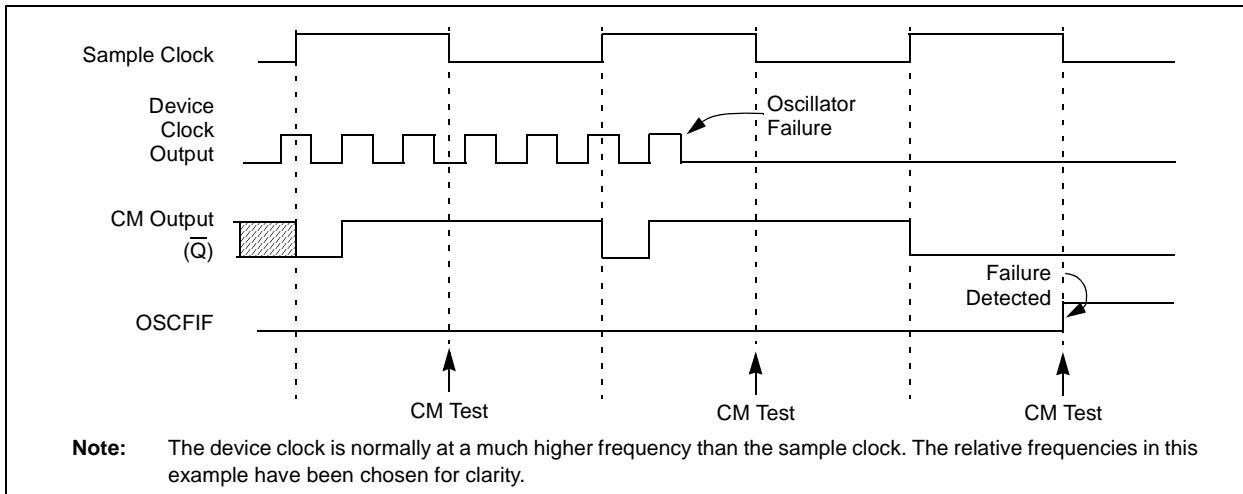
As already noted, the clock source is switched to the INTOSC clock when a clock failure is detected. Depending on the frequency selected by the IRCF2:IRCF0 bits, this may mean a substantial change in the speed of code execution. If the WDT is enabled with a small prescale value, a decrease in clock speed allows a WDT time-out to occur and a subsequent device Reset. For this reason, fail-safe clock events also reset the WDT and postscaler, allowing it to start timing from when execution speed was changed and decreasing the likelihood of an erroneous time-out.

### 23.4.2 EXITING FAIL-SAFE OPERATION

The fail-safe condition is terminated by either a device Reset or by entering a power-managed mode. On Reset, the controller starts the primary clock source specified in Configuration Register 1H (with any required start-up delays that are required for the oscillator mode, such as OST or PLL timer). The INTOSC multiplexer provides the device clock until the primary clock source becomes ready (similar to a Two-Speed Start-up). The clock source is then switched to the primary clock (indicated by the OSTs bit in the OSCCON register becoming set). The Fail-Safe Clock Monitor then resumes monitoring the peripheral clock.

The primary clock source may never become ready during start-up. In this case, operation is clocked by the INTOSC multiplexer. The OSCCON register will remain in its Reset state until a power-managed mode is entered.

**FIGURE 23-4: FSCM TIMING DIAGRAM**



### 23.4.3 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 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 ( $\text{OSCFIF} = 1$ ), code execution will be clocked by the INTOSC multiplexer. An automatic transition back to the failed clock source will not occur.

If the interrupt is disabled, subsequent interrupts while in Idle mode will cause the CPU to begin executing instructions while being clocked by the INTOSC source.

### 23.4.4 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 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 23.3.1 “Special Considerations for Using Two-Speed Start-up”**, it is also possible to select another clock configuration and enter an alternate power-managed mode while waiting for the primary clock to become stable. When the new power-managed mode is selected, the primary clock is disabled.

## 23.5 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 Flash devices differs significantly from other PIC® devices.

The user program memory is divided into five blocks. One of these is a boot block of 2 Kbytes. The remainder of the memory is divided into four blocks on binary boundaries.

Each of the five blocks has three code protection bits associated with them. They are:

- Code-Protect bit (CPn)
- Write-Protect bit (WRTn)
- External Block Table Read bit (EBTRn)

Figure 23-5 shows the program memory organization for 48 and 64-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 23-3.

**FIGURE 23-5: CODE-PROTECTED PROGRAM MEMORY FOR PIC18F2525/2620/4525/4620**

MEMORY SIZE/DEVICE		Address Range	Block Code Protection Controlled By:
48 Kbytes (PIC18F2525/4525)	64 Kbytes (PIC18F2620/4620)		
Boot Block	Boot Block	000000h 0007FFh	CPB, WRTB, EBTRB
Block 0	Block 0	000800h 003FFFh	CP0, WRT0, EBTR0
Block 1	Block 1	004000h 007FFFh	CP1, WRT1, EBTR1
Block 2	Block 2	008000h 00B7FFh	CP2, WRT2, EBTR2
Unimplemented Read '0's	Block 3	00C000h 00FFFFh	CP3, WRT3, EBTR3
Unimplemented Read '0's	Unimplemented Read '0's	010000h 1FFFFFFh	(Unimplemented Memory Space)

**TABLE 23-3: SUMMARY OF CODE PROTECTION REGISTERS**

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h	CONFIG5L	—	—	—	CP3 <sup>(1)</sup>	CP2	CP1	CP0
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—
30000Ah	CONFIG6L	—	—	—	WRT3 <sup>(1)</sup>	WRT2	WRT1	WRT0
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—
30000Ch	CONFIG7L	—	—	—	EBTR3 <sup>(1)</sup>	EBTR2	EBTR1	EBTR0
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—

**Legend:** Shaded cells are unimplemented.

**Note 1:** These bits are unimplemented in PIC18FX525 devices; maintain this bit set.

# PIC18F2525/2620/4525/4620

## 23.5.1 PROGRAM MEMORY CODE PROTECTION

The program memory may be read to or written from any location using the table read and table write instructions. The device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In normal execution mode, the CPx bits have no direct effect. CPx bits inhibit external reads and writes. A block of user memory may be protected from table writes if the WRTx Configuration bit is '0'. The EBTRx bits control table reads. For a block of user memory with the EBTRx bit set to '0', a table read instruction that executes from within that block is allowed to read.

A table read instruction that executes from a location outside of that block is not allowed to read and will result in reading '0's. Figures 23-6 through 23-8 illustrate table write and table read protection.

**Note:** Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full chip erase or block erase function. The full chip erase and block erase functions can only be initiated via ICSP operation or an external programmer.

**FIGURE 23-6: TABLE WRITE (WRTx) DISALLOWED**

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 0008FFh	000000h 000800h	WRTB, EBTRB = 11
PC = 003FFEh	003FFFh 004000h	WRT0, EBTR0 = 01
PC = 00BFFEh	007FFFh 008000h 00BFFFh 00C000h	WRT1, EBTR1 = 11 WRT2, EBTR2 = 11 WRT3, EBTR3 = 11
	00FFFFh	

**Results:** All table writes disabled to Blockn whenever WRTx = 0.

# PIC18F2525/2620/4525/4620

**FIGURE 23-7: EXTERNAL BLOCK TABLE READ (EBTRx) DISALLOWED**

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 0008FFh	000000h	WRTB, EBTRB = 11
PC = 007FFEh	0007FFh 000800h	WRT0, EBTR0 = 10
	003FFFh 004000h	WRT1, EBTR1 = 11
	007FFFh 008000h	WRT2, EBTR2 = 11
	00BFFFh 00C000h	WRT3, EBTR3 = 11
	00FFFFh	

**Results:** All table reads from external blocks to Blockn are disabled whenever EBTRx = 0.  
TABLAT register returns a value of '0'.

**FIGURE 23-8: EXTERNAL BLOCK TABLE READ (EBTRx) ALLOWED**

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 0008FFh	000000h	WRTB, EBTRB = 11
PC = 003FFEh	0007FFh 000800h	WRT0, EBTR0 = 10
	003FFFh 004000h	WRT1, EBTR1 = 11
	007FFFh 008000h	WRT2, EBTR2 = 11
	00BFFFh 00C000h	WRT3, EBTR3 = 11
	00FFFFh	

**Results:** Table reads permitted within Blockn, even when EBTRx = 0.  
TABLAT register returns the value of the data at the location TBLPTR.

## 23.5.2 DATA EEPROM CODE PROTECTION

The entire data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of data EEPROM. WRTD inhibits internal and external writes to data EEPROM. The CPU can always read data EEPROM under normal operation, regardless of the protection bit settings.

## 23.5.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In normal execution mode, the WRTC bit is readable only. WRTC can only be written via ICSP operation or an external programmer.

## 23.6 ID Locations

Eight memory locations (200000h-200007h) are designated as ID locations, where the user can store checksum or other code identification numbers. These locations are both readable and writable during normal execution through the TBLRD and TBLWT instructions or during program/verify. The ID locations can be read when the device is code-protected.

## 23.7 In-Circuit Serial Programming

PIC18F2525/2620/4525/4620 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.

## 23.8 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 23-4 shows which resources are required by the background debugger.

TABLE 23-4: DEBUGGER RESOURCES

I/O pins:	RB6, RB7
Stack:	2 levels
Program Memory:	512 bytes
Data Memory:	10 bytes

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to MCLR/VPP/RE3, VDD, Vss, RB7 and RB6. This will interface to the In-Circuit Debugger module available from Microchip or one of the third party development tool companies.

## 23.9 Single-Supply ICSP Programming

The LVP Configuration bit enables Single-Supply ICSP Programming (formerly known as *Low-Voltage ICSP Programming* or *LVP*). When Single-Supply Programming is enabled, the microcontroller can be programmed without requiring high voltage being applied to the MCLR/VPP/RE3 pin, but the RB5/KBI1/PGM pin is then dedicated to controlling Program mode entry and is not available as a general purpose I/O pin.

While programming, using Single-Supply Programming, VDD is applied to the MCLR/VPP/RE3 pin as in normal execution mode. To enter Programming mode, VDD is applied to the PGM pin.

**Note 1:** High-voltage programming is always available, regardless of the state of the LVP bit or the PGM pin, by applying VIHH to the MCLR pin.

- 2:** By default, Single-Supply ICSP Programming is enabled in unprogrammed devices (as supplied from Microchip) and erased devices.
- 3:** When Single-Supply Programming is enabled, the RB5 pin can no longer be used as a general purpose I/O pin.
- 4:** When LVP is enabled, externally pull the PGM pin to Vss to allow normal program execution.

If Single-Supply ICSP Programming mode will not be used, the LVP bit can be cleared. RB5/KBI1/PGM then becomes available as the digital I/O pin, RB5. The LVP bit may be set or cleared only when using standard high-voltage programming (VIHH applied to the MCLR/VPP/RE3 pin). Once LVP has been disabled, only the standard high-voltage programming is available and must be used to program the device.

Memory that is not code-protected can be erased using either a block erase, or erased row by row, then written at any specified VDD. If code-protected memory is to be erased, a block erase is required. If a block erase is to be performed when using Low-Voltage Programming, the device must be supplied with VDD of 4.5V to 5.5V.

## 24.0 INSTRUCTION SET SUMMARY

PIC18F2525/2620/4525/4620 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.

### 24.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are four instructions that require two program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

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

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18 instruction set summary in Table 24-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 24-1 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 24-1 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 24-2, lists the standard instructions recognized by the Microchip MPASM™ Assembler.

**Section 24.1.1 “Standard Instruction Set”** provides a description of each instruction.

# PIC18F2525/2620/4525/4620

---

TABLE 24-1: 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.
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.
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.
ε	In the set of.
italics	User-defined term (font is Courier New).

# PIC18F2525/2620/4525/4620

**FIGURE 24-1: GENERAL FORMAT FOR INSTRUCTIONS**

Byte-oriented file register operations	Example Instruction											
<p>15 10 9 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td>d</td> <td>a</td> <td>f (FILE #)</td> <td>0</td> </tr> </table> <p>d = 0 for result destination to be WREG register  d = 1 for result destination to be file register (f)  a = 0 to force Access Bank  a = 1 for BSR to select bank  f = 8-bit file register address</p>	OPCODE	d	a	f (FILE #)	0	ADDWF MYREG, W, B						
OPCODE	d	a	f (FILE #)	0								
<b>Byte to Byte move operations (2-word)</b>												
<p>15 12 11 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">f (Source FILE #)</td> <td>0</td> </tr> </table> <p>15 12 11 0</p> <table border="1"> <tr> <td>1111</td> <td colspan="3">f (Destination FILE #)</td> <td>0</td> </tr> </table> <p>f = 12-bit file register address</p>	OPCODE	f (Source FILE #)			0	1111	f (Destination FILE #)			0	MOVFF MYREG1, MYREG2	
OPCODE	f (Source FILE #)			0								
1111	f (Destination FILE #)			0								
<b>Bit-oriented file register operations</b>												
<p>15 12 11 9 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td>b (BIT #)</td> <td>a</td> <td>f (FILE #)</td> <td>0</td> </tr> </table> <p>b = 3-bit position of bit in file register (f)  a = 0 to force Access Bank  a = 1 for BSR to select bank  f = 8-bit file register address</p>	OPCODE	b (BIT #)	a	f (FILE #)	0	BSF MYREG, bit, B						
OPCODE	b (BIT #)	a	f (FILE #)	0								
<b>Literal operations</b>												
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">k (literal)</td> <td>0</td> </tr> </table> <p>k = 8-bit immediate value</p>	OPCODE	k (literal)			0	MOVLW 7Fh						
OPCODE	k (literal)			0								
<b>Control operations</b>												
<b>CALL, GOTO and Branch operations</b>												
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">n&lt;7:0&gt; (literal)</td> <td>0</td> </tr> </table> <p>15 12 11 0</p> <table border="1"> <tr> <td>1111</td> <td colspan="3">n&lt;19:8&gt; (literal)</td> <td>0</td> </tr> </table> <p>n = 20-bit immediate value</p>	OPCODE	n<7:0> (literal)			0	1111	n<19:8> (literal)			0	GOTO Label	
OPCODE	n<7:0> (literal)			0								
1111	n<19:8> (literal)			0								
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td>S</td> <td colspan="3">n&lt;7:0&gt; (literal)</td> <td>0</td> </tr> </table> <p>15 12 11 0</p> <table border="1"> <tr> <td>1111</td> <td colspan="3">n&lt;19:8&gt; (literal)</td> <td>0</td> </tr> </table> <p>S = Fast bit</p>	OPCODE	S	n<7:0> (literal)			0	1111	n<19:8> (literal)			0	CALL MYFUNC
OPCODE	S	n<7:0> (literal)			0							
1111	n<19:8> (literal)			0								
<p>15 11 10 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="4">n&lt;10:0&gt; (literal)</td> <td>0</td> </tr> </table>	OPCODE	n<10:0> (literal)				0	BRA MYFUNC					
OPCODE	n<10:0> (literal)				0							
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="4">n&lt;7:0&gt; (literal)</td> <td>0</td> </tr> </table>	OPCODE	n<7:0> (literal)				0	BC MYFUNC					
OPCODE	n<7:0> (literal)				0							

# PIC18F2525/2620/4525/4620

TABLE 24-2: 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 1st word f <sub>d</sub> (destination) 2nd word	2	1100	ffff	ffff	ffff	None		
			1111	ffff	ffff	ffff			
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.

TABLE 24-2: PIC18FXXXX 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, b, 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		
BN n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None		
BNC n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None		
BNN n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None		
BNOV n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None		
BNZ n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None		
BOV n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None		
BRA n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None		
BZ n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None		
CALL n, s	Call Subroutine 1st word	2	1110	110s	kkkk	kkkk	None		
	2nd word		1111	kkkk	kkkk	kkkk			
CLRWDT —	Clear Watchdog Timer	1	0000	0000	0000	0100	TO, PD		
DAW —	Decimal Adjust WREG	1	0000	0000	0000	0111	C		
GOTO n	Go to Address 1st word	2	1110	1111	kkkk	kkkk	None		
	2nd word		1111	kkkk	kkkk	kkkk			
NOP —	No Operation	1	0000	0000	0000	0000	None		
NOP —	No Operation	1	1111	xxxx	xxxx	xxxx	None	4	
POP —	Pop Top of Return Stack (TOS)	1	0000	0000	0000	0110	None		
PUSH —	Push Top of Return Stack (TOS)	1	0000	0000	0000	0101	None		
RCALL n	Relative Call	2	1101	1nnn	nnnn	nnnn	None		
RESET	Software Device Reset	1	0000	0000	1111	1111	All		
RETFIE s	Return from Interrupt Enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL		
RETLW k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None		
RETURN s	Return from Subroutine	2	0000	0000	0001	001s	None		
SLEEP —	Go into Standby mode	1	0000	0000	0000	0011	TO, PD		

- Note 1:** When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** If the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

# PIC18F2525/2620/4525/4620

TABLE 24-2: 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	0fff	kkkk	None		
			1111	0000	kkkk	kkkk			
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		
TBLWT*+	Table Write with Post-Increment		0000	0000	0000	1101	None		
TBLWT*-	Table Write with Post-Decrement		0000	0000	0000	1110	None		
TBLWT+*	Table Write with Pre-Increment		0000	0000	0000	1111	None		

- Note 1:** When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** If 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.

## 24.1.1 STANDARD INSTRUCTION SET

ADDLW	ADD Literal to W				ADDWF	ADD W to f										
Syntax:	ADDLW	k			Syntax:	ADDWF	f	{,d {,a}}								
Operands:	0 ≤ k ≤ 255				Operands:	0 ≤ f ≤ 255										
Operation:	(W) + k → W				d ∈ [0,1]											
Status Affected:	N, OV, C, DC, Z				a ∈ [0,1]											
Encoding:	0000	1111	kkkk	kkkk	Operation:	(W) + (f) → dest										
Description:	The contents of W are added to the 8-bit literal 'k' and the result is placed in W.				Status Affected:	N, OV, C, DC, Z										
Words:	1				Encoding:	0010	01da	ffff	ffff							
Cycles:	1				Description:	Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).										
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'k'</td> <td>Process Data</td> <td>Write to W</td> </tr> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to W	Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).			
Q1	Q2	Q3	Q4													
Decode	Read literal 'k'	Process Data	Write to W													

Example: ADDLW 15h

Before Instruction

W = 10h

After Instruction

W = 25h

Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </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: ADDWF REG, 0, 0

Before Instruction

W = 17h  
REG = 0C2h

After Instruction

W = 0D9h  
REG = 0C2h

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

# PIC18F2525/2620/4525/4620

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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0010</td> <td>00da</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0010	00da	ffff	ffff				
0010	00da	ffff	ffff						
Description:	<p>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'.</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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to W</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to W						

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:	$(W) .AND. k \rightarrow W$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>1011</td> <td>kkkk</td> <td>kkkk</td> </tr> </table>	0000	1011	kkkk	kkkk				
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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'k'</td> <td>Process Data</td> <td>Write to W</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read literal 'k'	Process Data	Write to W						

Example: ANDLW 05Fh

Before Instruction

W = A3h

After Instruction

W = 03h

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:	$(W) .AND. (f) \rightarrow \text{dest}$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <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 ANDed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank 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 <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </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" style="display: inline-table; vertical-align: middle;"> <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. 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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <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> </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	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>No operation</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	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)

# PIC18F2525/2620/4525/4620

---



---

BCF		Bit Clear f											
Syntax:		BCF f, b {,a}											
Operands:		0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]											
Operation:		0 → f<b>											
Status Affected:		None											
Encoding:		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1001</td><td>bbba</td><td>ffff</td><td>ffff</td></tr> </table>				1001	bbba	ffff	ffff				
1001	bbba	ffff	ffff										
Description:		<p>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 ≤ 95 (5Fh). See <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>											
Words:		1											
Cycles:		1											
Q Cycle Activity:		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write register 'f'</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
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 ≤ n ≤ 127															
Operation:		if Negative bit is '1', (PC) + 2 + 2n → PC															
Status Affected:		None															
Encoding:		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0110</td><td>nnnn</td><td>nnnn</td></tr> </table>				1110	0110	nnnn	nnnn								
1110	0110	nnnn	nnnn														
Description:		<p>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.</p>															
Words:		1															
Cycles:		1(2)															
Q Cycle Activity:		If Jump:															
		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <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> </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	Write to PC														
No operation	No operation	No operation	No operation														
If No Jump:		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
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 PC = 1;  
 If Negative PC = address (Jump)  
 If Negative PC = 0;  
 If Negative PC = address (HERE + 2)

BNC	Branch if Not Carry															
Syntax:	BNC n															
Operands:	-128 ≤ n ≤ 127															
Operation:	if Carry bit is '0', (PC) + 2 + 2n → PC															
Status Affected:	None															
Encoding:	1110	0011	nnnn	nnnn												
Description:	<p>If the Carry bit is '0', 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 PC + 2 + 2n. This instruction is then a two-cycle instruction.</p>															
Words:	1															
Cycles:	1(2)															
Q Cycle Activity:																
If Jump:	<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>				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	Write to PC													
No operation	No operation	No operation	No operation													
If No Jump:	<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	No operation				
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	No operation													

Example:	HERE	BNC	Jump
Before Instruction			
PC	=	address (HERE)	
After Instruction			
If Carry	=	0;	
PC	=	address (Jump)	
If Carry	=	1;	
PC	=	address (HERE + 2)	

BNN	Branch if Not Negative															
Syntax:	BNN n															
Operands:	-128 ≤ n ≤ 127															
Operation:	if Negative bit is '0', (PC) + 2 + 2n → PC															
Status Affected:	None															
Encoding:	1110	0111	nnnn	nnnn												
Description:	<p>If the Negative bit is '0', 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 PC + 2 + 2n. This instruction is then a two-cycle instruction.</p>															
Words:	1															
Cycles:	1(2)															
Q Cycle Activity:																
If Jump:	<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>				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	Write to PC													
No operation	No operation	No operation	No operation													
If No Jump:	<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	No operation				
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)	

# PIC18F2525/2620/4525/4620

---



---

<b>BNOV</b> <b>Branch if Not Overflow</b>													
Syntax:	BNOV n												
Operands:	$-128 \leq n \leq 127$												
Operation:	if Overflow bit is '0', (PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>0101</td><td>nnnn</td><td>nnnn</td></tr></table>	1110	0101	nnnn	nnnn								
1110	0101	nnnn	nnnn										
Description:	<p>If the Overflow bit is '0', 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 PC + 2 + 2n. This instruction is then a two-cycle instruction.</p>												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:													
If Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><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></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	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
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)

<b>BNZ</b> <b>Branch if Not Zero</b>													
Syntax:	BNZ n												
Operands:	$-128 \leq n \leq 127$												
Operation:	if Zero bit is '0', (PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>0001</td><td>nnnn</td><td>nnnn</td></tr></table>	1110	0001	nnnn	nnnn								
1110	0001	nnnn	nnnn										
Description:	<p>If the Zero bit is '0', 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 PC + 2 + 2n. This instruction is then a two-cycle instruction.</p>												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:													
If Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><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></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	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	No operation										

Example:      HERE      BNZ      Jump

Before Instruction  
 PC = address (HERE)

After Instruction  
 If Zero = 0;  
 PC = address (Jump)  
 If Zero = 1;  
 PC = address (HERE + 2)

BRA	Unconditional Branch			
Syntax:	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 <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> 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

# PIC18F2525/2620/4525/4620

BTFS <sub>C</sub>		Bit Test File, Skip if Clear	
Syntax:	BTFS <sub>C</sub> f, b {,a}		
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]		
Operation:	skip if (f<b>) = 0		
Status Affected:	None		
Encoding:	1011 bbba ffff ffff		
Description:	If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.		
Words:	1		
Cycles:	1(2)		
<b>Note:</b>	3 cycles if skip and followed by a 2-word instruction.		

## Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE BTFSC FLAG, 1, 0  
FALSE :  
TRUE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;  
PC = address (TRUE)  
If FLAG<1> = 1;  
PC = address (FALSE)

BTFS <sub>S</sub>		Bit Test File, Skip if Set	
Syntax:	BTFS <sub>S</sub> f, b {,a}		
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]		
Operation:	skip if (f<b>) = 1		
Status Affected:	None		
Encoding:	1010 bbba ffff ffff		
Description:	If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.		
Words:	1		
Cycles:	1(2)		
<b>Note:</b>	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 BTFS<sub>S</sub> 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)

BTG	Bit Toggle f								
Syntax:	BTG f, b {,a}								
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]								
Operation:	$(\overline{f<b>}) \rightarrow f<b>$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0111</td> <td>bbba</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0111	bbba	ffff	ffff				
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 ≤ 95 (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write register 'f'</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
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 ≤ n ≤ 127												
Operation:	if Overflow bit is '1', (PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1110</td> <td>0100</td> <td>nnnn</td> <td>nnnn</td> </tr> </table>	1110	0100	nnnn	nnnn								
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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <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> </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	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>No operation</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
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)

# PIC18F2525/2620/4525/4620

---

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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0000</td><td>nnnn</td><td>nnnn</td></tr> </table>	1110	0000	nnnn	nnnn								
1110	0000	nnnn	nnnn										
Description:	<p>If the Zero 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 PC + 2 + 2n. This instruction is then a two-cycle instruction.</p>												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:													
If Jump:													
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <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> </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	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:													
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
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 PC = 1;  
 PC = address (Jump)  
 If Zero PC = 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) $\rightarrow$ STATUS, (BSR) $\rightarrow$ BSRS												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>110s</td><td>k<sub>7</sub>kkk</td><td>kkkk<sub>0</sub></td></tr> <tr><td>1111</td><td>k<sub>19</sub>kkk</td><td>kkkk</td><td>kkkk<sub>8</sub></td></tr> </table>	1110	110s	k <sub>7</sub> kkk	kkkk <sub>0</sub>	1111	k <sub>19</sub> kkk	kkkk	kkkk <sub>8</sub>				
1110	110s	k <sub>7</sub> kkk	kkkk <sub>0</sub>										
1111	k <sub>19</sub> kkk	kkkk	kkkk <sub>8</sub>										
Description:	<p>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, STATUS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC&lt;20:1&gt;. CALL is a two-cycle instruction.</p>												
Words:	2												
Cycles:	2												
Q Cycle Activity:													
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read literal 'k'&lt;7:0&gt;;</td><td>PUSH PC to stack</td><td>Read literal 'k'&lt;19:8&gt;; Write to PC</td></tr> <tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'<7:0>;	PUSH PC to stack	Read literal 'k'<19:8>; Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'k'<7:0>;	PUSH PC to stack	Read literal 'k'<19:8>; Write to PC										
No operation	No operation	No operation	No operation										

Example: HERE CALL THERE, 1

Before Instruction  
 PC = address (HERE)

After Instruction  
 PC = address (THERE)  
 TOS = address (HERE + 4)  
 WS = W  
 BSRS = BSR  
 STATUS = STATUS

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:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0110</td><td>101a</td><td>ffff</td><td>ffff</td></tr></table>	0110	101a	ffff	ffff				
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 <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write register 'f'</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
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:	$000h \rightarrow \text{WDT}$ , $000h \rightarrow \text{WDT postscaler}$ , $1 \rightarrow \overline{\text{TO}}$ , $1 \rightarrow \overline{\text{PD}}$								
Status Affected:	$\overline{\text{TO}}, \overline{\text{PD}}$								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>0000</td><td>0000</td><td>0100</td></tr></table>	0000	0000	0000	0100				
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:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>No operation</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	No operation	Process Data	No operation
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

# PIC18F2525/2620/4525/4620

COMF	Complement f								
Syntax:	COMF f {,d {,a}}								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(\bar{f}) \rightarrow \text{dest}$								
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' (default). 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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </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: 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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.

Words: 1  
Cycles: 1(2)  
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process Data	No operation
If skip:				
Q1	Q2	Q3	Q4	
No operation	No operation	No operation	No operation	
If skip and followed by 2-word instruction:				
Q1	Q2	Q3	Q4	
No operation	No operation	No operation	No operation	
No operation	No operation	No operation	No operation	

Example: HERE CPFSEQ REG, 0  
NEQUAL :  
EQUAL :

Before Instruction  
PC Address = HERE  
W = ?  
REG = ?  
After Instruction  
If REG = W;  
PC = Address (EQUAL)  
If REG ≠ W;  
PC = Address (NEQUAL)

CPFSGT	Compare f with W, Skip if f > W	CPFSLT	Compare f with W, Skip if f < W																		
Syntax:	CPFSGT f {,a}	Syntax:	CPFSLT f {,a}																		
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$	Operands:	$0 \leq f \leq 255$ $a \in [0,1]$																		
Operation:	$(f) - (W)$ , skip if $(f) > (W)$ (unsigned comparison)	Operation:	$(f) - (W)$ , skip if $(f) < (W)$ (unsigned comparison)																		
Status Affected:	None	Status Affected:	None																		
Encoding:	0110 010a ffff ffff	Encoding:	0110 000a 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 <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.	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	Words:	1																		
Cycles:	1(2)	Cycles:	1(2)																		
<b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.		<b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.																			
Q Cycle Activity:		Q Cycle Activity:																			
<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>No operation</td></tr> </table>		Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	No operation	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>No operation</td></tr> </table>			Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	No operation	
Q1	Q2	Q3	Q4																		
Decode	Read register 'f'	Process Data	No operation																		
Q1	Q2	Q3	Q4																		
Decode	Read register 'f'	Process Data	No operation																		
If skip:		<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>		Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>		Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4																		
No operation	No operation	No operation	No operation																		
Q1	Q2	Q3	Q4																		
No operation	No operation	No operation	No operation																		
If skip and followed by 2-word instruction:		<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>		Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>		Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4																		
No operation	No operation	No operation	No operation																		
Q1	Q2	Q3	Q4																		
No operation	No operation	No operation	No operation																		
<u>Example:</u> HERE CPFSGT REG, 0 NGREATER : GREATER :		<u>Example:</u> HERE CPFSLT REG, 1 NLESS : LESS :		<u>Before Instruction</u> $PC = \text{Address (HERE)}$ $W = ?$																	
<u>Before Instruction</u> $PC = \text{Address (HERE)}$ $W = ?$		<u>After Instruction</u> $PC < W;$ $PC = \text{Address (LESS)}$ $PC \geq W;$ $PC = \text{Address (NLESS)}$																			
<u>After Instruction</u> $\text{If REG} > W;$ $PC = \text{Address (GREATER)}$ $\text{If REG} \leq W;$ $PC = \text{Address (NGREATER)}$																					

# PIC18F2525/2620/4525/4620

DAW	Decimal Adjust W Register	DECF	Decrement f																
Syntax:	DAW	Syntax:	DECF f {,d {,a}}																
Operands:	None	Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$																
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 $[W<7:4> + DC > 9]$ or $[C = 1]$ then, $(W<7:4>) + 6 + DC \rightarrow W<7:4>;$ else, $(W<7:4>) + DC \rightarrow W<7:4>;$	Operation:	$(f) - 1 \rightarrow \text{dest}$																
Status Affected:	C	Status Affected:	C, DC, N, OV, Z																
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0111</td></tr> </table>	0000	0000	0000	0111	Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>01da</td><td>ffff</td><td>ffff</td></tr> </table>	0000	01da	ffff	ffff								
0000	0000	0000	0111																
0000	01da	ffff	ffff																
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.	Description:	<p>Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).</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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>																
Words:	1	Words:	1																
Cycles:	1	Cycles:	1																
Q Cycle Activity:		Q Cycle Activity:																	
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read register W</td><td>Process Data</td><td>Write W</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register W	Process Data	Write W		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4																
Decode	Read register W	Process Data	Write W																
Q1	Q2	Q3	Q4																
Decode	Read register 'f'	Process Data	Write to destination																

## 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

## Example:

DECF CNT, 1, 0

### Before Instruction

CNT	=	01h
Z	=	0

### After Instruction

CNT	=	00h
Z	=	1

DECFSZ	Decrement f, Skip if 0															
Syntax:	DECFSZ f {,d {,a}}															
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]															
Operation:	(f) – 1 → dest, skip if result = 0															
Status Affected:	None															
Encoding:	0010	11da	ffff	ffff												
Description:	The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.															
Words:	1															
Cycles:	1(2) <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.															
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </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													
If skip:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation				
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
If skip and followed by 2-word instruction:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
No operation	No operation	No operation	No operation													
<b>Example:</b>	HERE	DECFSZ	CNT, 1, 1													
		GOTO	LOOP													
	CONTINUE															

Before Instruction  
 PC = Address (HERE)  
 After Instruction  
 CNT = CNT – 1  
 If CNT = 0;  
 PC = Address (CONTINUE)  
 If CNT ≠ 0;  
 PC = Address (HERE + 2)

DCFSNZ	Decrement f, Skip if Not 0															
Syntax:	DCFSNZ f {,d {,a}}															
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]															
Operation:	(f) – 1 → dest, skip if result ≠ 0															
Status Affected:	None															
Encoding:	0100	11da	ffff	ffff												
Description:	The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.															
Words:	1															
Cycles:	1(2) <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.															
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </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													
If skip:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation				
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
If skip and followed by 2-word instruction:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
No operation	No operation	No operation	No operation													

**Example:** HERE DCFNZ TEMP, 1, 0  
 ZERO :  
 NZERO :  
 Before Instruction  
 TEMP = ?  
 After Instruction  
 TEMP = TEMP – 1,  
 If TEMP = 0;  
 PC = Address (ZERO)  
 If TEMP ≠ 0;  
 PC = Address (NZERO)

# PIC18F2525/2620/4525/4620

---



---

<b>GOTO</b>		<b>Unconditional Branch</b>																					
Syntax:		GOTO k																					
Operands:		0 ≤ k ≤ 1048575																					
Operation:		k → PC<20:1>																					
Status Affected:		None																					
Encoding:		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1110</td> <td>1111</td> <td><math>k_7</math>kkk</td> <td><math>kkk_0</math></td> </tr> <tr> <td>1111</td> <td><math>k_{19}</math>kkk</td> <td>kkkk</td> <td><math>kkk_8</math></td> </tr> </table>				1110	1111	$k_7$ kkk	$kkk_0$	1111	$k_{19}$ kkk	kkkk	$kkk_8$										
1110	1111	$k_7$ kkk	$kkk_0$																				
1111	$k_{19}$ kkk	kkkk	$kkk_8$																				
Description:		<p>GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC&lt;20:1&gt;. GOTO is always a two-cycle instruction.</p>																					
Words:		2																					
Cycles:		2																					
Q Cycle Activity:																							
<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th><th></th><th></th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'k'&lt;7:0&gt;,</td><td>No operation</td><td>Read literal 'k'&lt;19:8&gt;, Write to PC</td><td></td><td></td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td><td></td><td></td></tr> </tbody> </table>						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		
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)

<b>INCF</b>		<b>Increment f</b>															
Syntax:		INCF f {,d {,a}}															
Operands:		0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]															
Operation:		$(f) + 1 \rightarrow \text{dest}$															
Status Affected:		C, DC, N, OV, Z															
Encoding:		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0010</td> <td>10da</td> <td>ffff</td> <td>ffff</td> </tr> </table>		0010	10da	ffff	ffff										
0010	10da	ffff	ffff														
Description:		<p>The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank 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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>															
Words:		1															
Cycles:		1															
Q Cycle Activity:																	
<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th><th></th><th></th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td><td></td><td></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: INCF CNT, 1, 0

Before Instruction

CNT	=	FFh
Z	=	0
C	=	?
DC	=	?

After Instruction

CNT	=	00h
Z	=	1
C	=	1
DC	=	1

INCFSZ	Increment f, Skip if 0															
Syntax:	INCFSZ f {,d {,a}}															
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]															
Operation:	(f) + 1 → dest, skip if result = 0															
Status Affected:	None															
Encoding:	0011	11da	ffff	ffff												
Description:	<p>The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>															
Words:	1															
Cycles:	1(2) <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.															
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </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													
If skip:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation				
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
If skip and followed by 2-word instruction:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
No operation	No operation	No operation	No operation													

Example: HERE INCFSZ CNT, 1, 0  
NZERO :  
ZERO :

Before Instruction  
PC = Address (HERE)  
After Instruction  
CNT = CNT + 1  
If CNT = 0;  
PC = Address (ZERO)  
If CNT ≠ 0;  
PC = Address (NZERO)

INFSNZ	Increment f, Skip if Not 0															
Syntax:	INFSNZ f {,d {,a}}															
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]															
Operation:	(f) + 1 → dest, skip if result ≠ 0															
Status Affected:	None															
Encoding:	0100	10da	ffff	ffff												
Description:	<p>The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>															
Words:	1															
Cycles:	1(2) <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.															
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </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													
If skip:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation				
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
If skip and followed by 2-word instruction:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>				Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
No operation	No operation	No operation	No operation													
No operation	No operation	No operation	No operation													

Example: HERE INFSNZ REG, 1, 0  
ZERO :  
NZERO :

Before Instruction  
PC = Address (HERE)  
After Instruction  
REG = REG + 1  
If REG ≠ 0;  
PC = Address (NZERO)  
If REG = 0;  
PC = Address (ZERO)

# PIC18F2525/2620/4525/4620

---



---

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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td><td>1001</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1001	kkkk	kkkk				
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:	<table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to W</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to W
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:	$(W) .OR. (f) \rightarrow \text{dest}$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0001</td><td>00da</td><td>ffff</td><td>ffff</td></tr> </table>	0001	00da	ffff	ffff				
0001	00da	ffff	ffff						
Description:	Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank 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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </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:      IORWF      RESULT, 0, 1

Before Instruction

RESULT = 13h

W = 91h

After Instruction

RESULT = 13h

W = 93h

LFSR	Load FSR															
Syntax:	LFSR f, k															
Operands:	0 ≤ f ≤ 2 0 ≤ k ≤ 4095															
Operation:	k → FSRf															
Status Affected:	None															
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>1110</td><td>00ff</td><td><math>k_{11}k_{10}k_9k_8</math></td></tr> <tr><td>1111</td><td>0000</td><td></td><td></td></tr> </table>				1110	1110	00ff	$k_{11}k_{10}k_9k_8$	1111	0000						
1110	1110	00ff	$k_{11}k_{10}k_9k_8$													
1111	0000															
Description:	The 12-bit literal 'k' is loaded into the File Select Register pointed to by 'f'.															
Words:	2															
Cycles:	2															
Q Cycle Activity:	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'k' MSB</td><td>Process Data</td><td>Write literal 'k' MSB to FSRfH</td></tr> <tr> <td>Decode</td><td>Read literal 'k' LSB</td><td>Process Data</td><td>Write literal 'k' to FSRfL</td></tr> </tbody> </table>				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
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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	f → dest								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0101</td><td>00da</td><td>ffff</td><td>ffff</td></tr> </table>	0101	00da	ffff	ffff				
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' (default). 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 ≤ 95 (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; text-align: center;"> <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 W</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write W
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

# PIC18F2525/2620/4525/4620

MOVFF	Move f to f
Syntax:	MOVFF f <sub>s</sub> ,f <sub>d</sub>
Operands:	0 ≤ f <sub>s</sub> ≤ 4095 0 ≤ f <sub>d</sub> ≤ 4095
Operation:	(f <sub>s</sub> ) → f <sub>d</sub>
Status Affected:	None
Encoding:	
1st word (source)	1100 ffff ffff ffff <sub>s</sub>
2nd word (destin.)	1111 ffff ffff ffff <sub>d</sub>
Description:	<p>The contents of source register 'f<sub>s</sub>' are moved to destination register 'f<sub>d</sub>'. Location of source 'f<sub>s</sub>' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f<sub>d</sub>' can also be anywhere from 000h to FFFh.</p> <p>Either source or destination can be W (a useful special situation).</p> <p>MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).</p> <p>The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.</p>
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 ≤ k ≤ 255		
Operation:	k → BSR		
Status Affected:	None		
Encoding:	0000 0001 kkkk kkkk		
Description:	<p>The eight-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR&lt;7:4&gt; always remains '0', regardless of the value of k<sub>7:k<sub>4</sub></sub>.</p>		
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

## MOVLW

### Move Literal to W

Syntax:	MOVLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	k → 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 ≤ f ≤ 255			
Operation:	a ∈ [0,1]			
Status Affected:	(W) → f			
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 ≤ 95 (5Fh). See **Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1  
Cycles: 1

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: MOVWF REG, 0

#### Before Instruction

W = 4Fh  
REG = FFh

#### After Instruction

W = 4Fh  
REG = 4Fh

# PIC18F2525/2620/4525/4620

MULLW	Multiply Literal with W								
Syntax:	MULLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	$(W) \times k \rightarrow \text{PRODH:PRODL}$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>1101</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1101	kkkk	kkkk				
0000	1101	kkkk	kkkk						
Description:	<p>An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.</p> <p>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.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write registers PRODH: PRODL</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL
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:	$(W) \times (f) \rightarrow \text{PRODH:PRODL}$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>001a</td><td>ffff</td><td>ffff</td></tr> </table>	0000	001a	ffff	ffff				
0000	001a	ffff	ffff						
Description:	<p>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.</p> <p>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.</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 <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write registers PRODH: PRODL</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL
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

NEGF	Negate f											
Syntax:	NEGF f {,a}											
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]											
Operation:	$(\bar{f}) + 1 \rightarrow f$											
Status Affected:	N, OV, C, DC, Z											
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0110</td> <td>110a</td> <td>ffff</td> <td>ffff</td> </tr> </table>				0110	110a	ffff	ffff				
0110	110a	ffff	ffff									
Description:	<p>Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write register 'f'</td> </tr> </table>				Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>0000</td> </tr> <tr> <td>1111</td> <td>xxxx</td> <td>xxxx</td> <td>xxxx</td> </tr> </table>				0000	0000	0000	0000	1111	xxxx	xxxx	xxxx
0000	0000	0000	0000									
1111	xxxx	xxxx	xxxx									
Description:	No operation.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </table>				Q1	Q2	Q3	Q4	Decode	No operation	No operation	No operation
Q1	Q2	Q3	Q4									
Decode	No operation	No operation	No operation									

Example:

None.

# PIC18F2525/2620/4525/4620

---

POP	Pop Top of Return Stack								
Syntax:	POP								
Operands:	None								
Operation:	(TOS) → bit bucket								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0110</td></tr> </table>	0000	0000	0000	0110				
0000	0000	0000	0110						
Description:	<p>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.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>POP TOS value</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	POP TOS value	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	POP TOS value	No operation						

<u>Example:</u>	POP		
	GOTO	NEW	
<b>Before Instruction</b>			
TOS	=	0031A2h	
Stack (1 level down)	=	014332h	
<b>After Instruction</b>			
TOS	=	014332h	
PC	=	NEW	

PUSH	Push Top of Return Stack								
Syntax:	PUSH								
Operands:	None								
Operation:	(PC + 2) → TOS								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0101</td></tr> </table>	0000	0000	0000	0101				
0000	0000	0000	0101						
Description:	<p>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.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>PUSH PC + 2 onto return stack</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	PUSH PC + 2 onto return stack	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	PUSH PC + 2 onto return stack	No operation	No operation						

<u>Example:</u>	PUSH		
<b>Before Instruction</b>			
TOS	=	345Ah	
PC	=	0124h	
<b>After Instruction</b>			
PC	=	0126h	
TOS	=	0126h	
Stack (1 level down)	=	345Ah	

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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1101</td> <td>1nnn</td> <td>nnnn</td> <td>nnnn</td> </tr> </table>	1101	1nnn	nnnn	nnnn								
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:													
	<table border="1" style="display: inline-table; vertical-align: middle;"> <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' PUSH PC to stack</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>	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
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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>1111</td> <td>1111</td> </tr> </table>	0000	0000	1111	1111				
0000	0000	1111	1111						
Description:	This instruction provides a way to execute a MCLR Reset in software.								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Start Reset</td> <td>No operation</td> <td>No operation</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Start Reset	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	Start Reset	No operation	No operation						

Example:      RESET

After Instruction

Registers = Reset Value  
Flags\* = Reset Value

# PIC18F2525/2620/4525/4620

RETFIE	Return from Interrupt												
Syntax:	RETFIE {s}												
Operands:	$s \in [0,1]$												
Operation:	$(TOS) \rightarrow PC$ , $1 \rightarrow GIE/GIEH$ or $PEIE/GIEL$ ; if $s = 1$ , $(WS) \rightarrow W$ , $(STATUS) \rightarrow STATUS$ , $(BSRS) \rightarrow BSR$ , $PCLATU$ , $PCLATH$ are unchanged												
Status Affected:	GIE/GIEH, PEIE/GIEL.												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0001</td><td>000s</td></tr> </table>	0000	0000	0001	000s								
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 BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers occurs (default).												
Words:	1												
Cycles:	2												
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>No operation</td><td>POP PC from stack, Set GIEH or GIEL</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	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
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	=	BSRS
STATUS	=	STATUS
GIE/GIEH, PEIE/GIEL	=	1

RETLW	Return Literal to W												
Syntax:	RETLW k												
Operands:	$0 \leq k \leq 255$												
Operation:	$k \rightarrow W$ , $(TOS) \rightarrow PC$ , $PCLATU$ , $PCLATH$ are unchanged												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>1100</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1100	kkkk	kkkk								
0000	1100	kkkk	kkkk										
Description:	<p>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.</p> <p>Words: 1</p> <p>Cycles: 2</p> <p>Q Cycle Activity:</p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>POP PC from stack, Write to W</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	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
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

```

# PIC18F2525/2620/4525/4620

RETURN	Return from Subroutine	RLCF	Rotate Left f through Carry								
Syntax:	RETURN {s}	Syntax:	RLCF f {,d {,a}}								
Operands:	$s \in [0,1]$	Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(TOS) \rightarrow PC;$ if $s = 1$ , $(WS) \rightarrow W$ , $(STATUS) \rightarrow STATUS$ , $(BSRS) \rightarrow BSR$ , PCLATU, PCLATH are unchanged	Operation:	$(f<n>) \rightarrow dest<n+1>$ , $(f<7>) \rightarrow C$ , $(C) \rightarrow dest<0>$								
Status Affected:	None	Status Affected:	C, N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>0000</td><td>0001</td><td>001s</td></tr></table>	0000	0000	0001	001s	Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0011</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>	0011	01da	ffff	ffff
0000	0000	0001	001s								
0011	01da	ffff	ffff								
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 (default).	Description:	The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank 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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode</b> ” for details.								
Words:	1	Words:	1								
Cycles:	2	Cycles:	1								
Q Cycle Activity:		Q Cycle Activity:									
	Q1      Q2      Q3      Q4		Q1      Q2      Q3      Q4								
	Decode      No operation      Process Data      POP PC from stack		C      register f								
	No operation      No operation      No operation      No operation										

Example: RETURN

After Instruction:  
PC = TOS

Example:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

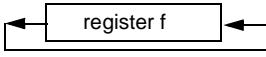
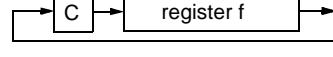
Example: RLCF REG, 0, 0

Before Instruction  
REG = 1110 0110  
C = 0

After Instruction

REG = 1110 0110  
W = 1100 1100  
C = 1

# PIC18F2525/2620/4525/4620

RLNCF	Rotate Left f (No Carry)	RRCF	Rotate Right f through Carry																
Syntax:	RLNCF f {,d {,a}}	Syntax:	RRCF f {,d {,a}}																
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$	Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$																
Operation:	$(f < n >) \rightarrow \text{dest} < n + 1 >$ , $(f < 7 >) \rightarrow \text{dest} < 0 >$	Operation:	$(f < n >) \rightarrow \text{dest} < n - 1 >$ , $(f < 0 >) \rightarrow C$ , $(C) \rightarrow \text{dest} < 7 >$																
Status Affected:	N, Z	Status Affected:	C, N, Z																
Encoding:	0100 01da ffff ffff	Encoding:	0011 00da ffff ffff																
Description:	<p>The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank 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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p> 	<p>The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).</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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p> 																	
Words:	1	Words:	1																
Cycles:	1	Cycles:	1																
Q Cycle Activity:		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		<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																
Q1	Q2	Q3	Q4																
Decode	Read register 'f'	Process Data	Write to destination																
Example:	RLNCF REG, 1, 0	Example:	RRCF REG, 0, 0																
Before Instruction		Before Instruction																	
REG = 1010 1011		REG = 1110 0110																	
After Instruction		C = 0																	
REG = 0101 0111		After Instruction																	
		REG = 1110 0110																	
		W = 0111 0011																	
		C = 0																	

RRNCF	Rotate Right f (No Carry)	SETF	Set f	
Syntax:	RRNCF f {,d {,a}}	Syntax:	SETF f {,a}	
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$	Operands:	$0 \leq f \leq 255$ $a \in [0,1]$	
Operation:	$(f < n >) \rightarrow \text{dest} < n - 1 >$ , $(f < 0 >) \rightarrow \text{dest} < 7 >$	Operation:	$\text{FFh} \rightarrow f$	
Status Affected:	N, Z	Status Affected:	None	
Encoding:	0100 00da ffff ffff	Encoding:	0110 100a ffff ffff	
Description:	<p>The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default). 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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p> 	Description:	<p>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 <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>	
Words:	1	Words:	1	
Cycles:	1	Cycles:	1	
Q Cycle Activity:		Q Cycle Activity:		
	Q1      Q2      Q3      Q4		Q1      Q2      Q3      Q4	
	Decode	Read register 'f'	Process Data	Write register 'f'

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

Example:      SETF      REG, 1

Before Instruction  
REG = 5Ah  
After Instruction  
REG = FFh

# PIC18F2525/2620/4525/4620

SLEEP	Enter Sleep mode								
Syntax:	SLEEP								
Operands:	None								
Operation:	00h → WDT, 0 → WDT postscaler, 1 → $\overline{TO}$ , 0 → PD								
Status Affected:	$\overline{TO}$ , PD								
Encoding:	0000 0000 0000 0011								
Description:	The Power-Down status bit (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:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>Process Data</td><td>Go to Sleep</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	Process Data	Go to Sleep
Q1	Q2	Q3	Q4						
Decode	No operation	Process Data	Go to Sleep						

Example: SLEEP

Before Instruction

$\overline{TO}$  = ?  
PD = ?

After Instruction

$\overline{TO}$  = 1†  
PD = 0

† If WDT causes wake-up, this bit is cleared.

SUBFWB	Subtract f from W with Borrow
Syntax:	SUBFWB f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ {0,1} a ∈ {0,1}
Operation:	(W) – (f) – ( $\overline{C}$ ) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	0101 01da ffff ffff
Description:	<p>Subtract register 'f' and Carry flag (borrow) from W (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored in register 'f' (default). If 'a' is '0', the Access Bank 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 <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>

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

<b>SUBLW</b>		<b>Subtract W from Literal</b>	
Syntax:		SUBLW k	
Operands:		0 ≤ k ≤ 255	
Operation:		k - (W) → 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
Decode	Read literal 'k'	Process Data	Write to W
Q4			

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

<b>SUBWF</b>		<b>Subtract W from f</b>	
Syntax:		SUBWF f {,d {,a}}	
Operands:		0 ≤ f ≤ 255	
d ∈ [0,1]		d ∈ [0,1]	
a ∈ [0,1]		a ∈ [0,1]	
Operation:		(f) - (W) → dest	
Status Affected:		N, OV, C, DC, Z	
Encoding:		0101 11da ffff ffff	
Description:		Subtract W from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank 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 <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> 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

# PIC18F2525/2620/4525/4620

SUBWFB	Subtract W from f with Borrow	SWAPF	Swap f																
Syntax:	SUBWFB f {,d {,a}}	Syntax:	SWAPF f {,d {,a}}																
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$	Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$																
Operation:	$(f) - (W) - (\bar{C}) \rightarrow \text{dest}$	Operation:	$(f<3:0>) \rightarrow \text{dest}<7:4>,$ $(f<7:4>) \rightarrow \text{dest}<3:0>$																
Status Affected:	N, OV, C, DC, Z	Status Affected:	None																
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0101</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0101	10da	ffff	ffff	Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0011</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0011	10da	ffff	ffff								
0101	10da	ffff	ffff																
0011	10da	ffff	ffff																
Description:	<p>Subtract W and the Carry flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).</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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>	Description:	<p>The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default).</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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>																
Words:	1	Words:	1																
Cycles:	1	Cycles:	1																
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination	Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></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																
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)
W	=	0Eh	; [2's comp] (0000 1101)
C	=	0	
Z	=	0	
N	=	1	; result is negative

Example: SWAPF REG, 1, 0

Before Instruction

REG	=	53h	
-----	---	-----	--

After Instruction

REG	=	35h	
-----	---	-----	--

TBLRD	Table Read								
Syntax:	TBLRD ( *; *+; *-; +* )								
Operands:	None								
Operation:	<p>if TBLRD *,          (Prog Mem (TBLPTR)) → TABLAT,          TBLPTR – No Change;</p> <p>if TBLRD *+,          (Prog Mem (TBLPTR)) → TABLAT,          (TBLPTR) + 1 → TBLPTR;</p> <p>if TBLRD *-,          (Prog Mem (TBLPTR)) → TABLAT,          (TBLPTR) – 1 → TBLPTR;</p> <p>if TBLRD +*,          (TBLPTR) + 1 → TBLPTR,          (Prog Mem (TBLPTR)) → TABLAT</p>								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>10nn</td></tr> <tr><td></td><td></td><td></td><td>nn=0 * =1 *+ =2 *- =3 +*</td></tr> </table>	0000	0000	0000	10nn				nn=0 * =1 *+ =2 *- =3 +*
0000	0000	0000	10nn						
			nn=0 * =1 *+ =2 *- =3 +*						
Description:	<p>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.</p> <p>The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.</p> <p>TBLPTR[0] = 0: Least Significant Byte of Program Memory Word</p> <p>TBLPTR[0] = 1: Most Significant Byte of Program Memory Word</p> <p>The TBLRD instruction can modify the value of TBLPTR as follows:</p> <ul style="list-style-type: none"> <li>• no change</li> <li>• post-increment</li> <li>• post-decrement</li> <li>• pre-increment</li> </ul>								
Words:	1								
Cycles:	2								

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)

TBLRD	Table Read (Continued)
Example 1:	TBLRD *+ ;
Before Instruction	
TABLAT	= 55h
TBLPTR	= 00A356h
MEMORY (00A356h)	= 34h
After Instruction	
TABLAT	= 34h
TBLPTR	= 00A357h
Example 2:	TBLRD +* ;
Before Instruction	
TABLAT	= AAh
TBLPTR	= 01A357h
MEMORY (01A357h)	= 12h
MEMORY (01A358h)	= 34h
After Instruction	
TABLAT	= 34h
TBLPTR	= 01A358h

# PIC18F2525/2620/4525/4620

TBLWT	Table Write												
Syntax:	TBLWT ( *, *+; *-; +*)												
Operands:	None												
Operation:	<p>if TBLWT*,          (TABLAT) → Holding Register,          TBLPTR – No Change;</p> <p>if TBLWT*+,          (TABLAT) → Holding Register,          (TBLPTR) + 1 → TBLPTR;</p> <p>if TBLWT*-,          (TABLAT) → Holding Register,          (TBLPTR) – 1 → TBLPTR;</p> <p>if TBLWT+*,          (TBLPTR) + 1 → TBLPTR,          (TABLAT) → Holding Register</p>												
Status Affected:	None												
Encoding:	<table border="1"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>11nn nn=0 * =1 *+ =2 *- =3 +*</td> </tr> </table>	0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*								
0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*										
Description:	<p>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 <b>Section 7.0 “Flash Program Memory”</b> 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.</p> <p>TBLPTR[0] = 0: Least Significant Byte of Program Memory Word          TBLPTR[0] = 1: Most Significant Byte of Program Memory Word</p> <p>The TBLWT instruction can modify the value of TBLPTR as follows:</p> <ul style="list-style-type: none"> <li>• no change</li> <li>• post-increment</li> <li>• post-decrement</li> <li>• pre-increment</li> </ul>												
Words:	1												
Cycles:	2												
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> <tr> <td>No operation</td> <td>No operation (Read TABLAT)</td> <td>No operation</td> <td>No operation (Write to Holding Register)</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)				
Q1	Q2	Q3	Q4										
Decode	No operation	No operation	No operation										
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)										

TBLWT	Table Write (Continued)
Example 1:	TBLWT * + ;
Before Instruction	
TABLAT	= 55h
TBLPTR	= 00A356h
HOLDING REGISTER (00A356h)	= FFh
After Instructions (table write completion)	
TABLAT	= 55h
TBLPTR	= 00A357h
HOLDING REGISTER (00A356h)	= 55h
Example 2:	TBLWT + * ;
Before Instruction	
TABLAT	= 34h
TBLPTR	= 01389Ah
HOLDING REGISTER (01389Ah)	= FFh
HOLDING REGISTER (01389Bh)	= FFh
After Instruction (table write completion)	
TABLAT	= 34h
TBLPTR	= 01389Bh
HOLDING REGISTER (01389Ah)	= FFh
HOLDING REGISTER (01389Bh)	= 34h

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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0110</td><td>011a</td><td>ffff</td><td>ffff</td></tr> </table>	0110	011a	ffff	ffff
0110	011a	ffff	ffff		
Description:	<p>If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a <b>NOP</b> is executed, making this a two-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>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 <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>				
Words:	1				
Cycles:	1(2)				
	<b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.				

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:      HERE      TSTFSZ CNT, 1  
                   NZERO      :  
                   ZERO      :

Before Instruction  
     PC      =      Address (HERE)  
 After Instruction  
     If CNT      =      00h,  
     PC      =      Address (ZERO)  
     If CNT      ≠      00h,  
     PC      =      Address (NZERO)

XORLW	Exclusive OR Literal with W								
Syntax:	XORLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	$(W) .XOR. k \rightarrow W$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>1010</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1010	kkkk	kkkk				
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:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'k'</td> <td>Process Data</td> <td>Write to W</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read literal 'k'	Process Data	Write to W						

Example:      XORLW      0AFh

Before Instruction  
     W      =      B5h  
 After Instruction  
     W      =      1Ah

# PIC18F2525/2620/4525/4620

---

---

## 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:	$(W) .XOR. (f) \rightarrow \text{dest}$			
Status Affected:	N, Z			
Encoding:	0001	10da	ffff	ffff
Description:	Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default). If 'a' is '0', the Access Bank 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 <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> 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

## 24.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18F2525/2620/4525/4620 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features 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 (with the exception of CALLW, MOVSF and MOVSS) 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 deallocation 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 24-3. Detailed descriptions are provided in **Section 24.2.2 “Extended Instruction Set”**. The opcode field descriptions in Table 24-1 (page 268) 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.

### 24.2.1 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 24.2.3.1 “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 24-3: 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	1111	ffff	ffff	ffff	None
PUSHL k	Store Literal at FSR2, Decrement FSR2	1	1110	1011	1zzz	zzzz	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

# PIC18F2525/2620/4525/4620

## 24.2.2 EXTENDED INSTRUCTION SET

ADDFSR	Add Literal to FSR	ADDULNK	Add Literal to FSR2 and Return							
Syntax:	ADDFSR f, k	Syntax:	ADDULNK k							
Operands:	$0 \leq k \leq 63$ $f \in [0, 1, 2]$	Operands:	$0 \leq k \leq 63$							
Operation:	$FSR(f) + k \rightarrow FSR(f)$	Operation:	$FSR2 + k \rightarrow FSR2$ , (TOS) $\rightarrow$ PC							
Status Affected:	None	Status Affected:	None							
Encoding:	1110 1000 ffk <sub>k</sub> kkkk	Encoding:	1110 1000 11k <sub>k</sub> kkkk							
Description:	The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.	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.							
Words:	1	Words:	1							
Cycles:	1	Cycles:	2							
Q Cycle Activity:		Q Cycle Activity:	The instruction takes two cycles to execute; a NOP is performed during the second cycle.							
	<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 'k'</td><td>Process Data</td><td>Write to FSR</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to FSR	<p>This may be thought of as a special case of the ADDFSR instruction, where <math>f = 3</math> (binary '11'); it operates only on FSR2.</p>
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

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)

**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).

CALLW	Subroutine Call Using WREG												
Syntax:	CALLW												
Operands:	None												
Operation:	$(PC + 2) \rightarrow TOS,$ $(W) \rightarrow PCL,$ $(PCLATH) \rightarrow PCH,$ $(PCLATU) \rightarrow PCU$												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>0000</td><td>0001</td><td>0100</td></tr></table>	0000	0000	0001	0100								
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:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read WREG</td><td>PUSH PC to stack</td><td>No operation</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read WREG	PUSH PC to stack	No operation				
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:	$0 \leq z_s \leq 127$ $0 \leq f_d \leq 4095$												
Operation:	$((FSR2) + z_s) \rightarrow f_d$												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>1011</td><td>0zzz</td><td>zzzz<sub>s</sub></td></tr><tr><td>1111</td><td>ffff</td><td>ffff</td><td>ffff<sub>d</sub></td></tr></table>	1110	1011	0zzz	zzzz <sub>s</sub>	1111	ffff	ffff	ffff <sub>d</sub>				
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:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Determine source addr</td><td>Determine source addr</td><td>Read source reg</td></tr><tr><td>Decode</td><td>No operation No dummy read</td><td>No operation</td><td>Write register 'f' (dest)</td></tr></table>	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)
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

# PIC18F2525/2620/4525/4620

MOVSS	Move Indexed to Indexed	PUSHL	Store Literal at FSR2, Decrement FSR2																				
Syntax:	MOVSS [z <sub>s</sub> ], [z <sub>d</sub> ]	Syntax:	PUSHL k																				
Operands:	0 ≤ z <sub>s</sub> ≤ 127 0 ≤ z <sub>d</sub> ≤ 127	Operands:	0 ≤ k ≤ 255																				
Operation:	((FSR2) + z <sub>s</sub> ) → ((FSR2) + z <sub>d</sub> )	Operation:	k → (FSR2), FSR2 – 1 → FSR2																				
Status Affected:	None	Status Affected:	None																				
Encoding:		Encoding:																					
1st word (source)	1110 1011 1zzz zzzz <sub>s</sub>	1111 1010 kkkk kkkk																					
2nd word (dest.)	1111 xxxx xzzz zzzz <sub>d</sub>																						
Description	<p>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).</p> <p>The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.</p> <p>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.</p>	<p>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.</p>																					
Words:	2	Words:	1																				
Cycles:	2	Cycles:	1																				
Q Cycle Activity:		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>Determine source addr</td><td>Determine source addr</td><td>Read source reg</td></tr> <tr> <td>Decode</td><td>Determine dest addr</td><td>Determine dest addr</td><td>Write to dest reg</td></tr> </tbody> </table>	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	<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 'k'</td><td>Process data</td><td>Write to destination</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read 'k'	Process data	Write to destination	
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																				
Q1	Q2	Q3	Q4																				
Decode	Read 'k'	Process data	Write to destination																				

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

Example: PUSHL 08h

Before Instruction

FSR2H:FSR2L	=	01ECh
Memory (01ECh)	=	00h

After Instruction

FSR2H:FSR2L	=	01EBh
Memory (01ECh)	=	08h

<b>SUBFSR</b>	<b>Subtract Literal from FSR</b>								
Syntax:	SUBFSR f, k								
Operands:	$0 \leq k \leq 63$ $f \in [0, 1, 2]$								
Operation:	$FSR(f - k) \rightarrow FSR(f)$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>1001</td><td>ffkk</td><td>kkkk</td></tr></table>	1110	1001	ffkk	kkkk				
1110	1001	ffkk	kkkk						
Description:	The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </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: SUBFSR 2, 23h

Before Instruction

FSR2 = 03FFh

After Instruction

FSR2 = 03DCh

<b>SUBULNK</b>	<b>Subtract Literal from FSR2 and Return</b>												
Syntax:	SUBULNK k												
Operands:	$0 \leq k \leq 63$												
Operation:	$FSR2 - k \rightarrow FSR2$ , (TOS) $\rightarrow$ PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>1001</td><td>11kk</td><td>kkkk</td></tr></table>	1110	1001	11kk	kkkk								
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:													
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> <tr> <td>No Operation</td><td>No Operation</td><td>No Operation</td><td>No Operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination	No Operation	No Operation	No Operation	No Operation
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)

## 24.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

**Note:** Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing mode (**Section 5.5.1 “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 24.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”**).

Although the Indexed Literal Offset Addressing mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing mode.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset Addressing mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

## 24.2.3.1 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 mode, 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.

## 24.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18F2525/2620/4525/4620, 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.

ADDWF	ADD W to Indexed (Indexed Literal Offset mode)											
Syntax:	ADDWF [k] {,d}											
Operands:	0 ≤ k ≤ 95 d ∈ {0,1}											
Operation:	(W) + ((FSR2) + k) → dest											
Status Affected:	N, OV, C, DC, Z											
Encoding:	0010	01d0	kkkk	kkkk								
Description:	The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read 'k'</td><td>Process Data</td><td>Write to destination</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read 'k'	Process Data	Write to destination
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 ≤ f ≤ 95 0 ≤ b ≤ 7											
Operation:	1 → ((FSR2) + k)<b>											
Status Affected:	None											
Encoding:	1000	bbb0	kkkk	kkkk								
Description:	Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </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: 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 ≤ k ≤ 95											
Operation:	FFh → ((FSR2) + k)											
Status Affected:	None											
Encoding:	0110	1000	kkkk	kkkk								
Description:	The contents of the register indicated by FSR2, offset by 'k', are set to FFh.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read 'k'</td><td>Process Data</td><td>Write register</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read 'k'	Process Data	Write register
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

# PIC18F2525/2620/4525/4620

---

## 24.2.5 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 PIC18F2525/2620/4525/4620 family of devices. This includes the MPLAB C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

## 25.0 DEVELOPMENT SUPPORT

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

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C18 and MPLAB C30 C Compilers
  - MPLINK™ Object Linker/  
MPLIB™ Object Librarian
  - MPLAB ASM30 Assembler/Linker/Library
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD 2
- Device Programmers
  - PICSTART® Plus Development Programmer
  - MPLAB PM3 Device Programmer
  - PICkit™ 2 Development Programmer
- Low-Cost Demonstration and Development Boards and Evaluation Kits

## 25.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16-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)
  - 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
- Visual device initializer for easy register initialization
- 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 HI-TECH Software C Compilers and IAR C Compilers

The MPLAB IDE allows you to:

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

## 25.2 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for all PIC 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

## 25.3 MPLAB C18 and MPLAB C30 C Compilers

The MPLAB C18 and MPLAB C30 Code Development Systems are complete ANSI C compilers for Microchip's PIC18 and PIC24 families of microcontrollers and the dsPIC30 and dsPIC33 family of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

## 25.4 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

## 25.5 MPLAB ASM30 Assembler, Linker and Librarian

MPLAB ASM30 Assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 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 dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

## 25.6 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C18 and MPLAB C30 C Compilers, and the MPASM and MPLAB ASM30 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.

## 25.7 MPLAB ICE 2000 High-Performance In-Circuit Emulator

The MPLAB ICE 2000 In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers. Software control of the MPLAB ICE 2000 In-Circuit Emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The architecture of the MPLAB ICE 2000 In-Circuit Emulator allows expansion to support new PIC microcontrollers.

The MPLAB ICE 2000 In-Circuit Emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft® Windows® 32-bit operating system were chosen to best make these features available in a simple, unified application.

## 25.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs PIC® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The MPLAB REAL ICE probe 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 the popular MPLAB ICD 2 system (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

MPLAB REAL ICE is field upgradeable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added, such as software breakpoints and assembly code trace. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, real-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 25.9 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low-cost, run-time development tool, connecting to the host PC via an RS-232 or high-speed USB interface. This tool is based on the Flash PIC MCUs and can be used to develop for these and other PIC MCUs and dsPIC DSCs. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the Flash devices. This feature, along with Microchip's In-Circuit Serial Programming™ (ICSP™) protocol, offers cost-effective, in-circuit Flash debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single stepping and watching variables, and CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real time. MPLAB ICD 2 also serves as a development programmer for selected PIC devices.

## 25.10 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 SD/MMC card for file storage and secure data applications.

# PIC18F2525/2620/4525/4620

---

## 25.11 PICSTART Plus Development Programmer

The PICSTART Plus Development Programmer is an easy-to-use, low-cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. The PICSTART Plus Development Programmer supports most PIC devices in DIP packages up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus Development Programmer is CE compliant.

## 25.12 PICkit 2 Development Programmer

The PICkit™ 2 Development Programmer is a low-cost programmer and selected Flash device debugger with an easy-to-use interface for programming many of Microchip's baseline, mid-range and PIC18F families of Flash memory microcontrollers. The PICkit 2 Starter Kit includes a prototyping development board, twelve sequential lessons, software and HI-TECH's PICC™ Lite C compiler, and is designed to help get up to speed quickly using PIC® microcontrollers. The kit provides everything needed to program, evaluate and develop applications using Microchip's powerful, mid-range Flash memory family of microcontrollers.

## 25.13 Demonstration, Development and Evaluation Boards

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.

Check the Microchip web page ([www.microchip.com](http://www.microchip.com)) for the complete list of demonstration, development and evaluation kits.

## 26.0 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 Vss (except VDD and <u>MCLR</u> ) .....	-0.3V to (VDD + 0.3V)
Voltage on VDD with respect to Vss .....	-0.3V to +7.5V
Voltage on <u>MCLR</u> with respect to Vss ( <b>Note 2</b> ) .....	0V to +13.25V
Total power dissipation ( <b>Note 1</b> ) .....	1.0W
Maximum current out of Vss pin .....	300 mA
Maximum current into VDD pin .....	250 mA
Input clamp current, <u>I<sub>IK</sub></u> ( <u>V<sub>I</sub></u> < 0 or <u>V<sub>I</sub></u> > VDD).....	±20 mA
Output clamp current, <u>I<sub>OK</sub></u> ( <u>V<sub>O</sub></u> < 0 or <u>V<sub>O</sub></u> > VDD) .....	±20 mA
Maximum output current sunk by any I/O pin.....	25 mA
Maximum output current sourced by any I/O pin .....	25 mA
Maximum current sunk by all ports .....	200 mA
Maximum current sourced by all ports .....	200 mA

**Note 1:** Power dissipation is calculated as follows:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$$

**2:** Voltage spikes below Vss at the MCLR/VPP/RE3 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 MCLR/VPP/RE3 pin, rather than pulling this pin directly to Vss.

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

# PIC18F2525/2620/4525/4620

FIGURE 26-1: PIC18F2525/2620/4525/4620 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

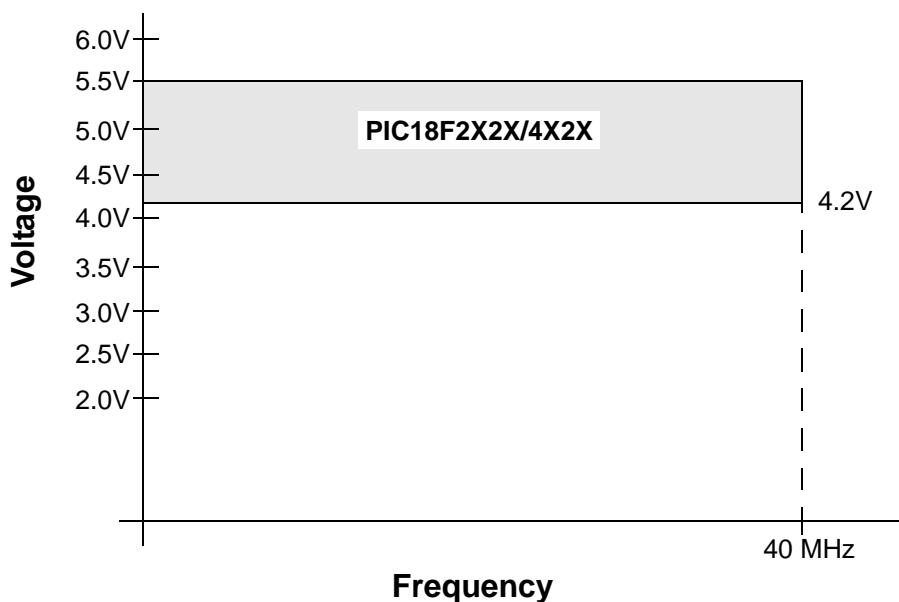
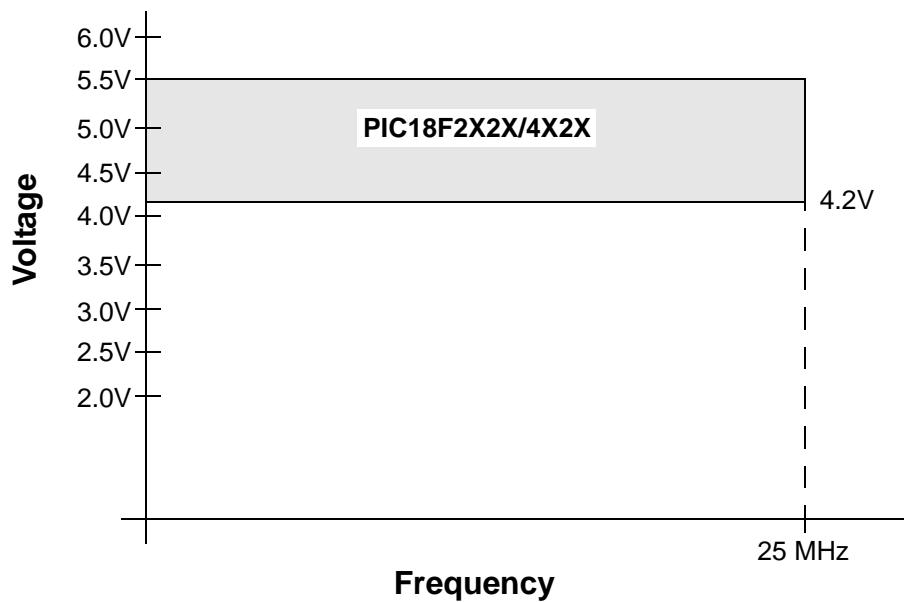
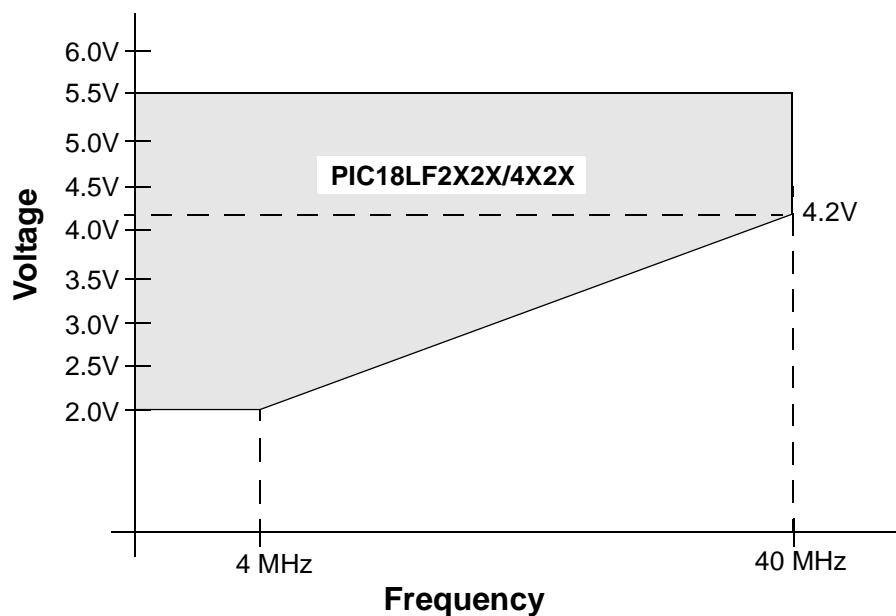


FIGURE 26-2: PIC18F2525/2620/4525/4620 VOLTAGE-FREQUENCY GRAPH (EXTENDED)



# PIC18F2525/2620/4525/4620

FIGURE 26-3: PIC18LF2525/2620/4525/4620 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



$$F_{MAX} = (16.36 \text{ MHz/V}) (V_{DDAPPMIN} - 2.0V) + 4 \text{ MHz}$$

**Note:**  $V_{DDAPPMIN}$  is the minimum voltage of the PIC<sup>®</sup> device in the application.

# PIC18F2525/2620/4525/4620

## 26.1 DC Characteristics: Supply Voltage

**PIC18F2525/2620/4525/4620 (Industrial)**

**PIC18LF2525/2620/4525/4620 (Industrial)**

<b>PIC18LF2525/2620/4525/4620</b> (Industrial)		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial					
<b>PIC18F2525/2620/4525/4620</b> (Industrial, Extended)		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended					
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
D001	VDD	<b>Supply Voltage</b>					
		PIC18LFX525/X620	2.0	—	5.5	V	HS, XT, RC and LP Oscillator mode
		PIC18FX525/X620	4.2	—	5.5	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 section on Power-on Reset for details
D004	SVDD	<b>VDD Rise Rate</b> to ensure internal Power-on Reset signal	0.05	—	—	V/ms	See section on Power-on Reset for details
D005	VBOR	<b>Brown-out Reset Voltage</b>					
		PIC18LFX525/X620					
		BORV1:BORV0 = 11	2.00	2.11	2.22	V	
		BORV1:BORV0 = 10	2.65	2.79	2.93	V	
D005		All Devices					
		BORV1:BORV0 = 01 <sup>(2)</sup>	4.11	4.33	4.55	V	
		BORV1:BORV0 = 00	4.36	4.59	4.82	V	

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

**2:** With BOR enabled, full-speed operation (Fosc = 40 MHz) is supported until a BOR occurs. This is valid although VDD may be below the minimum voltage for this frequency.

## 26.2 DC Characteristics: Power-Down and Supply Current

**PIC18F2525/2620/4525/4620 (Industrial)**

**PIC18LF2525/2620/4525/4620 (Industrial)**

<b>PIC18LF2525/2620/4525/4620 (Industrial)</b>		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial						
<b>PIC18F2525/2620/4525/4620 (Industrial, Extended)</b>		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended						
Param No.	Device	Typ	Max	Units	Conditions			
<b>Power-Down Current (IPD)<sup>(1)</sup></b>								
PIC18LFX525/X620								
0.1 0.5 $\mu\text{A}$					-40°C	$\text{VDD} = 2.0\text{V}$ <b>(Sleep mode)</b>		
0.1 0.5 $\mu\text{A}$					+25°C			
0.2 2.5 $\mu\text{A}$					+85°C			
PIC18LFX525/X620					-40°C	$\text{VDD} = 3.0\text{V}$ <b>(Sleep mode)</b>		
0.1 0.7 $\mu\text{A}$					+25°C			
0.3 3.5 $\mu\text{A}$					+85°C			
All devices					-40°C	$\text{VDD} = 5.0\text{V}$ <b>(Sleep mode)</b>		
0.2 1.0 $\mu\text{A}$					+25°C			
0.7 10 $\mu\text{A}$					+85°C			
Extended devices only					+125°C			

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD or VSS;  
MCLR = VDD; WDT enabled/disabled as specified.

**3:** When operation below  $-10^{\circ}\text{C}$  is expected, use 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 HILVD enable internal band gap reference. With both modules enabled, current consumption will be less than the sum of both specifications.

# PIC18F2525/2620/4525/4620

## 26.2 DC Characteristics: Power-Down and Supply Current

**PIC18F2525/2620/4525/4620 (Industrial)**  
**PIC18LF2525/2620/4525/4620 (Industrial) (Continued)**

<b>PIC18LF2525/2620/4525/4620</b> (Industrial)		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial																																																																																																																																					
<b>PIC18F2525/2620/4525/4620</b> (Industrial, Extended)		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended																																																																																																																																					
Param No.	Device	Typ	Max	Units	Conditions																																																																																																																																		
<b>Supply Current (IDD)<sup>(2)</sup></b>																																																																																																																																							
<table border="1"> <tbody> <tr> <td>PIC18LFX525/X620</td><td>13</td><td>25</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 2.0V</td><td rowspan="12" style="vertical-align: middle; text-align: center;"> <math>\text{FOSC} = 31 \text{ kHz}</math>  <b>(RC_RUN mode,</b>  <b>INTRC source)</b> </td></tr> <tr> <td></td><td>13</td><td>22</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td><td></td></tr> <tr> <td></td><td>14</td><td>25</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td><td></td></tr> <tr> <td>PIC18LFX525/X620</td><td>42</td><td>61</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 3.0V</td><td></td></tr> <tr> <td></td><td>34</td><td>46</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td><td></td></tr> <tr> <td></td><td>28</td><td>45</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td><td></td></tr> <tr> <td>All devices</td><td>103</td><td>160</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 5.0V</td><td></td></tr> <tr> <td></td><td>82</td><td>130</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td><td></td></tr> <tr> <td></td><td>67</td><td>120</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td><td></td></tr> <tr> <td>Extended devices only</td><td>71</td><td>230</td><td><math>\mu\text{A}</math></td><td><math>+125^{\circ}\text{C}</math></td><td rowspan="3">VDD = 2.0V</td><td rowspan="12" style="vertical-align: middle; text-align: center;"> <math>\text{FOSC} = 1 \text{ MHz}</math>  <b>(RC_RUN mode,</b>  <b>INTOSC source)</b> </td></tr> <tr> <td>PIC18LFX525/X620</td><td>320</td><td>440</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td></td></tr> <tr> <td></td><td>330</td><td>440</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td><td></td></tr> <tr> <td>PIC18LFX525/X620</td><td>330</td><td>440</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td><td rowspan="3">VDD = 3.0V</td><td></td></tr> <tr> <td></td><td>630</td><td>800</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td></td></tr> <tr> <td></td><td>590</td><td>720</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td><td></td></tr> <tr> <td>All devices</td><td>570</td><td>700</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td><td rowspan="3">VDD = 5.0V</td><td></td></tr> <tr> <td></td><td>1.2</td><td>1.6</td><td><math>\text{mA}</math></td><td><math>-40^{\circ}\text{C}</math></td><td></td></tr> <tr> <td></td><td>1.0</td><td>1.5</td><td><math>\text{mA}</math></td><td><math>+25^{\circ}\text{C}</math></td><td></td></tr> <tr> <td>Extended devices only</td><td>1.0</td><td>1.5</td><td><math>\text{mA}</math></td><td><math>+85^{\circ}\text{C}</math></td><td></td><td></td></tr> <tr> <td></td><td>1.0</td><td>1.5</td><td><math>\text{mA}</math></td><td><math>+125^{\circ}\text{C}</math></td><td></td><td></td></tr> </tbody> </table>								PIC18LFX525/X620	13	25	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 2.0V	$\text{FOSC} = 31 \text{ kHz}$ <b>(RC_RUN mode,</b> <b>INTRC source)</b>		13	22	$\mu\text{A}$	$+25^{\circ}\text{C}$			14	25	$\mu\text{A}$	$+85^{\circ}\text{C}$		PIC18LFX525/X620	42	61	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.0V			34	46	$\mu\text{A}$	$+25^{\circ}\text{C}$			28	45	$\mu\text{A}$	$+85^{\circ}\text{C}$		All devices	103	160	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 5.0V			82	130	$\mu\text{A}$	$+25^{\circ}\text{C}$			67	120	$\mu\text{A}$	$+85^{\circ}\text{C}$		Extended devices only	71	230	$\mu\text{A}$	$+125^{\circ}\text{C}$	VDD = 2.0V	$\text{FOSC} = 1 \text{ MHz}$ <b>(RC_RUN mode,</b> <b>INTOSC source)</b>	PIC18LFX525/X620	320	440	$\mu\text{A}$	$-40^{\circ}\text{C}$			330	440	$\mu\text{A}$	$+25^{\circ}\text{C}$		PIC18LFX525/X620	330	440	$\mu\text{A}$	$+85^{\circ}\text{C}$	VDD = 3.0V			630	800	$\mu\text{A}$	$-40^{\circ}\text{C}$			590	720	$\mu\text{A}$	$+25^{\circ}\text{C}$		All devices	570	700	$\mu\text{A}$	$+85^{\circ}\text{C}$	VDD = 5.0V			1.2	1.6	$\text{mA}$	$-40^{\circ}\text{C}$			1.0	1.5	$\text{mA}$	$+25^{\circ}\text{C}$		Extended devices only	1.0	1.5	$\text{mA}$	$+85^{\circ}\text{C}$				1.0	1.5	$\text{mA}$	$+125^{\circ}\text{C}$		
PIC18LFX525/X620	13	25	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 2.0V	$\text{FOSC} = 31 \text{ kHz}$ <b>(RC_RUN mode,</b> <b>INTRC source)</b>																																																																																																																																	
	13	22	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																			
	14	25	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																																																			
PIC18LFX525/X620	42	61	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.0V																																																																																																																																		
	34	46	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																			
	28	45	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																																																			
All devices	103	160	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 5.0V																																																																																																																																		
	82	130	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																			
	67	120	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																																																			
Extended devices only	71	230	$\mu\text{A}$	$+125^{\circ}\text{C}$	VDD = 2.0V		$\text{FOSC} = 1 \text{ MHz}$ <b>(RC_RUN mode,</b> <b>INTOSC source)</b>																																																																																																																																
PIC18LFX525/X620	320	440	$\mu\text{A}$	$-40^{\circ}\text{C}$																																																																																																																																			
	330	440	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																			
PIC18LFX525/X620	330	440	$\mu\text{A}$	$+85^{\circ}\text{C}$	VDD = 3.0V																																																																																																																																		
	630	800	$\mu\text{A}$	$-40^{\circ}\text{C}$																																																																																																																																			
	590	720	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																			
All devices	570	700	$\mu\text{A}$	$+85^{\circ}\text{C}$	VDD = 5.0V																																																																																																																																		
	1.2	1.6	$\text{mA}$	$-40^{\circ}\text{C}$																																																																																																																																			
	1.0	1.5	$\text{mA}$	$+25^{\circ}\text{C}$																																																																																																																																			
Extended devices only	1.0	1.5	$\text{mA}$	$+85^{\circ}\text{C}$																																																																																																																																			
	1.0	1.5	$\text{mA}$	$+125^{\circ}\text{C}$																																																																																																																																			

**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 VDD or Vss and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD or Vss;

MCLR = VDD; WDT enabled/disabled as specified.

**3:** When operation below  $-10^{\circ}\text{C}$  is expected, use 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.

# PIC18F2525/2620/4525/4620

## 26.2 DC Characteristics: Power-Down and Supply Current

**PIC18F2525/2620/4525/4620 (Industrial)**  
**PIC18LF2525/2620/4525/4620 (Industrial) (Continued)**

<b>PIC18LF2525/2620/4525/4620</b> (Industrial)	<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial						
<b>PIC18F2525/2620/4525/4620</b> (Industrial, Extended)	<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended						
Param No.	Device	Typ	Max	Units	Conditions		
<b>Supply Current (I<sub>DD</sub>)<sup>(2)</sup></b>							
PIC18LFX525/X620	0.8	1.1	mA	-40°C	V <sub>DD</sub> = 2.0V	FOSC = 4 MHz ( <b>RC_RUN</b> mode, INTOSC source)	
	0.8	1.1	mA	+25°C			
	0.8	1.1	mA	+85°C			
PIC18LFX525/X620	1.3	1.7	mA	-40°C	V <sub>DD</sub> = 3.0V		
	1.3	1.7	mA	+25°C			
	1.3	1.7	mA	+85°C			
All devices	2.5	3.5	mA	-40°C	V <sub>DD</sub> = 5.0V		
	2.5	3.5	mA	+25°C			
	2.5	3.5	mA	+85°C			
Extended devices only	2.5	3.5	mA	+125°C			
PIC18LFX525/X620	2.9	5	μA	-40°C	V <sub>DD</sub> = 2.0V	FOSC = 31 kHz ( <b>RC_IDLE</b> mode, INTRC source)	
	3.1	5	μA	+25°C			
	3.6	9.5	μA	+85°C			
PIC18LFX525/X620	4.5	8	μA	-40°C	V <sub>DD</sub> = 3.0V		
	4.8	8	μA	+25°C			
	5.8	15	μA	+85°C			
All devices	9.2	16	μA	-40°C	V <sub>DD</sub> = 5.0V		
	9.8	16	μA	+25°C			
	11.0	35	μA	+85°C			
Extended devices only	21	160	μA	+125°C			

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to 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 tri-stated, pulled to V<sub>DD</sub> or V<sub>SS</sub>;  
MCLR = V<sub>DD</sub>; WDT enabled/disabled as specified.

**3:** When operation below -10°C is expected, use 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.

# PIC18F2525/2620/4525/4620

## 26.2 DC Characteristics: Power-Down and Supply Current

**PIC18F2525/2620/4525/4620 (Industrial)**  
**PIC18LF2525/2620/4525/4620 (Industrial) (Continued)**

<b>PIC18LF2525/2620/4525/4620 (Industrial)</b>		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial																																																																																																							
<b>PIC18F2525/2620/4525/4620 (Industrial, Extended)</b>		<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended																																																																																																							
Param No.	Device	Typ	Max	Units	Conditions																																																																																																				
<b>Supply Current (IDD)<sup>(2)</sup></b>																																																																																																									
<table border="1"> <tbody> <tr> <td rowspan="3">PIC18LFX525/X620</td><td>165</td><td>250</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 2.0V</td><td rowspan="12" style="vertical-align: middle; text-align: center;"> <math>\text{FOSC} = 1 \text{ MHz}</math>  <math>(\text{RC\_IDLE} \text{ mode, INTOSC source})</math> </td></tr> <tr><td>175</td><td>250</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td>190</td><td>270</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr> <td rowspan="3">PIC18LFX525/X620</td><td>250</td><td>360</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 3.0V</td></tr> <tr><td>270</td><td>360</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td>290</td><td>380</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr> <td rowspan="3">All devices</td><td>500</td><td>700</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 5.0V</td></tr> <tr><td>520</td><td>700</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td>550</td><td>700</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr> <td>Extended devices only</td><td>0.6</td><td>1</td><td><math>\text{mA}</math></td><td><math>+125^{\circ}\text{C}</math></td><td></td><td></td></tr> <tr> <td rowspan="3">PIC18LFX525/X620</td><td>340</td><td>500</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 2.0V</td></tr> <tr><td>350</td><td>500</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td>360</td><td>500</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr> <td rowspan="3">PIC18LFX525/X620</td><td>520</td><td>800</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 3.0V</td></tr> <tr><td>540</td><td>800</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td>580</td><td>850</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr> <td rowspan="3">All devices</td><td>1.0</td><td>1.6</td><td><math>\text{mA}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 5.0V</td></tr> <tr><td>1.1</td><td>1.4</td><td><math>\text{mA}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td>1.1</td><td>1.4</td><td><math>\text{mA}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr> <td>Extended devices only</td><td>1.1</td><td>2.0</td><td><math>\text{mA}</math></td><td><math>+125^{\circ}\text{C}</math></td><td></td><td></td></tr> </tbody> </table>							PIC18LFX525/X620	165	250	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 2.0V	$\text{FOSC} = 1 \text{ MHz}$ $(\text{RC\_IDLE} \text{ mode, INTOSC source})$	175	250	$\mu\text{A}$	$+25^{\circ}\text{C}$	190	270	$\mu\text{A}$	$+85^{\circ}\text{C}$	PIC18LFX525/X620	250	360	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.0V	270	360	$\mu\text{A}$	$+25^{\circ}\text{C}$	290	380	$\mu\text{A}$	$+85^{\circ}\text{C}$	All devices	500	700	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 5.0V	520	700	$\mu\text{A}$	$+25^{\circ}\text{C}$	550	700	$\mu\text{A}$	$+85^{\circ}\text{C}$	Extended devices only	0.6	1	$\text{mA}$	$+125^{\circ}\text{C}$			PIC18LFX525/X620	340	500	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 2.0V	350	500	$\mu\text{A}$	$+25^{\circ}\text{C}$	360	500	$\mu\text{A}$	$+85^{\circ}\text{C}$	PIC18LFX525/X620	520	800	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.0V	540	800	$\mu\text{A}$	$+25^{\circ}\text{C}$	580	850	$\mu\text{A}$	$+85^{\circ}\text{C}$	All devices	1.0	1.6	$\text{mA}$	$-40^{\circ}\text{C}$	VDD = 5.0V	1.1	1.4	$\text{mA}$	$+25^{\circ}\text{C}$	1.1	1.4	$\text{mA}$	$+85^{\circ}\text{C}$	Extended devices only	1.1	2.0	$\text{mA}$	$+125^{\circ}\text{C}$		
PIC18LFX525/X620	165	250	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 2.0V	$\text{FOSC} = 1 \text{ MHz}$ $(\text{RC\_IDLE} \text{ mode, INTOSC source})$																																																																																																			
	175	250	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																					
	190	270	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																					
PIC18LFX525/X620	250	360	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.0V																																																																																																				
	270	360	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																					
	290	380	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																					
All devices	500	700	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 5.0V																																																																																																				
	520	700	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																					
	550	700	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																					
Extended devices only	0.6	1	$\text{mA}$	$+125^{\circ}\text{C}$																																																																																																					
PIC18LFX525/X620	340	500	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 2.0V																																																																																																				
	350	500	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																					
	360	500	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																					
PIC18LFX525/X620	520	800	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.0V																																																																																																				
	540	800	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																					
	580	850	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																					
All devices	1.0	1.6	$\text{mA}$	$-40^{\circ}\text{C}$	VDD = 5.0V																																																																																																				
	1.1	1.4	$\text{mA}$	$+25^{\circ}\text{C}$																																																																																																					
	1.1	1.4	$\text{mA}$	$+85^{\circ}\text{C}$																																																																																																					
Extended devices only	1.1	2.0	$\text{mA}$	$+125^{\circ}\text{C}$																																																																																																					

**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 VDD or Vss and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD or Vss;  
MCLR = VDD; WDT enabled/disabled as specified.

- 3: When operation below  $-10^{\circ}\text{C}$  is expected, use 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.

## 26.2 DC Characteristics: Power-Down and Supply Current

**PIC18F2525/2620/4525/4620 (Industrial)**  
**PIC18LF2525/2620/4525/4620 (Industrial) (Continued)**

<b>PIC18LF2525/2620/4525/4620 (Industrial)</b>		<b>Standard Operating Conditions (unless otherwise stated)</b>						
		Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial						
<b>PIC18F2525/2620/4525/4620 (Industrial, Extended)</b>		<b>Standard Operating Conditions (unless otherwise stated)</b>						
		Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended						
Param No.	Device	Typ	Max	Units	Conditions			
<b>Supply Current (IDD)<sup>(2)</sup></b>								
PIC18LFX525/X620		250	350	$\mu\text{A}$	-40°C	VDD = 2.0V	FOSC = 1 MHz ( <b>PRI_RUN</b> , EC oscillator)	
		260	350	$\mu\text{A}$	+25°C			
		250	350	$\mu\text{A}$	+85°C			
PIC18LFX525/X620		550	650	$\mu\text{A}$	-40°C	VDD = 3.0V		
		480	640	$\mu\text{A}$	+25°C			
		460	600	$\mu\text{A}$	+85°C			
All devices		1.2	1.5	mA	-40°C	VDD = 5.0V		
		1.1	1.4	mA	+25°C			
		1.0	1.3	mA	+85°C			
Extended devices only		1.0	3.0	mA	+125°C			
PIC18LFX525/X620		0.72	1.0	mA	-40°C	VDD = 2.0V	FOSC = 4 MHz ( <b>PRI_RUN</b> , EC oscillator)	
		0.74	1.0	mA	+25°C			
		0.74	1.0	mA	+85°C			
PIC18LFX525/X620		1.3	1.8	mA	-40°C	VDD = 3.0V		
		1.3	1.8	mA	+25°C			
		1.3	1.8	mA	+85°C			
All devices		2.7	4.0	mA	-40°C	VDD = 5.0V		
		2.6	4.0	mA	+25°C			
		2.5	4.0	mA	+85°C			
Extended devices only		2.6	5.0	mA	+125°C			
Extended devices only		8.4	13	mA	+125°C	VDD = 4.2V	Fosc = 25 MHz ( <b>PRI_RUN</b> , EC oscillator)	
		11	16	mA	+125°C	VDD = 5.0V		
All devices		15	20	mA	-40°C	VDD = 4.2V	Fosc = 40 MHz ( <b>PRI_RUN</b> , EC oscillator)	
		15	20	mA	+25°C			
		15	20	mA	+85°C			
All devices		20	25	mA	-40°C	VDD = 5.0V		
		20	25	mA	+25°C			
		20	25	mA	+85°C			

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

Osc1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD or VSS;  
MCLR = VDD; WDT enabled/disabled as specified.

**3:** When operation below  $-10^{\circ}\text{C}$  is expected, use 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.

# PIC18F2525/2620/4525/4620

## 26.2 DC Characteristics: Power-Down and Supply Current

**PIC18F2525/2620/4525/4620 (Industrial)**  
**PIC18LF2525/2620/4525/4620 (Industrial) (Continued)**

<b>PIC18LF2525/2620/4525/4620</b> (Industrial)	<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial					
<b>PIC18F2525/2620/4525/4620</b> (Industrial, Extended)	<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended					
Param No.	Device	Typ	Max	Units	Conditions	
<b>Supply Current (IDD)<sup>(2)</sup></b>						
All devices	7.5	10	mA	$-40^{\circ}\text{C}$	VDD = 4.2V	Fosc = 4 MHz, 16 MHz internal (PRI_RUN HS+PLL)
	7.4	10	mA	$+25^{\circ}\text{C}$		
	7.3	10	mA	$+85^{\circ}\text{C}$		
Extended devices only	8.0	12	mA	$+125^{\circ}\text{C}$	VDD = 5.0V	Fosc = 4 MHz, 16 MHz internal (PRI_RUN HS+PLL)
All devices	10	12	mA	$-40^{\circ}\text{C}$		
	10	12	mA	$+25^{\circ}\text{C}$		
	9.7	12	mA	$+85^{\circ}\text{C}$		
Extended devices only	10	14	mA	$+125^{\circ}\text{C}$	VDD = 4.2V	Fosc = 10 MHz, 40 MHz internal (PRI_RUN HS+PLL)
All devices	15	20	mA	$-40^{\circ}\text{C}$		
	15	20	mA	$+25^{\circ}\text{C}$		
	15	20	mA	$+85^{\circ}\text{C}$		
All devices	20	25	mA	$-40^{\circ}\text{C}$	VDD = 5.0V	Fosc = 10 MHz, 40 MHz internal (PRI_RUN HS+PLL)
	20	25	mA	$+25^{\circ}\text{C}$		
	20	25	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 tri-stated, pulled to VDD or VSS;

MCLR = VDD; WDT enabled/disabled as specified.

**3:** When operation below  $-10^{\circ}\text{C}$  is expected, use 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.

# PIC18F2525/2620/4525/4620

## 26.2 DC Characteristics: Power-Down and Supply Current

**PIC18F2525/2620/4525/4620 (Industrial)**  
**PIC18LF2525/2620/4525/4620 (Industrial) (Continued)**

<b>PIC18LF2525/2620/4525/4620 (Industrial)</b>	<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial																																																																																																																																																																																			
<b>PIC18F2525/2620/4525/4620 (Industrial, Extended)</b>	<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended																																																																																																																																																																																			
Param No.	Device	Typ	Max	Units	Conditions																																																																																																																																																																															
<b>Supply Current (IDD)<sup>(2)</sup></b>																																																																																																																																																																																				
<table border="1"> <tr><td>PIC18LFX525/X620</td><td>65</td><td>100</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 2.0V</td><td rowspan="12" style="vertical-align: middle; text-align: center;"><math>\text{Fosc} = 1 \text{ MHz}</math> <b>(PRI_IDLE mode, EC oscillator)</b></td></tr> <tr><td></td><td>65</td><td>100</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td><td></td></tr> <tr><td></td><td>70</td><td>110</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td><td></td></tr> <tr><td colspan="2" style="text-align: center;">PIC18LFX525/X620</td><td>120</td><td>140</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 3.0V</td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>120</td><td>140</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>130</td><td>160</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;">All devices</td><td>230</td><td>300</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 5.0V</td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>235</td><td>300</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>240</td><td>300</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;">Extended devices only</td><td>260</td><td>500</td><td><math>\mu\text{A}</math></td><td><math>+125^{\circ}\text{C}</math></td><td rowspan="6" style="vertical-align: middle; text-align: center;"><math>\text{Fosc} = 4 \text{ MHz}</math> <b>(PRI_IDLE mode, EC oscillator)</b></td></tr> <tr><td colspan="2" style="text-align: center;">PIC18LFX525/X620</td><td>260</td><td>360</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>255</td><td>360</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>270</td><td>360</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;">PIC18LFX525/X620</td><td>420</td><td>620</td><td><math>\mu\text{A}</math></td><td><math>-40^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>430</td><td>620</td><td><math>\mu\text{A}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>450</td><td>650</td><td><math>\mu\text{A}</math></td><td><math>+85^{\circ}\text{C}</math></td><td rowspan="5" style="vertical-align: middle; text-align: center;"><math>\text{Fosc} = 25 \text{ MHz}</math> <b>(PRI_IDLE mode, EC oscillator)</b></td></tr> <tr><td colspan="2" style="text-align: center;">All devices</td><td>0.9</td><td>1.2</td><td><math>\text{mA}</math></td><td><math>-40^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>0.9</td><td>1.2</td><td><math>\text{mA}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>0.9</td><td>1.2</td><td><math>\text{mA}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;">Extended devices only</td><td>1</td><td>1.3</td><td><math>\text{mA}</math></td><td><math>+125^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;">Extended devices only</td><td>2.8</td><td>6.0</td><td><math>\text{mA}</math></td><td><math>+125^{\circ}\text{C}</math></td><td>VDD = 4.2V</td><td rowspan="5" style="vertical-align: middle; text-align: center;"><math>\text{Fosc} = 40 \text{ MHz}</math> <b>(PRI_IDLE mode, EC oscillator)</b></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>4.3</td><td>8.0</td><td><math>\text{mA}</math></td><td><math>+125^{\circ}\text{C}</math></td><td>VDD = 5.0V</td></tr> <tr><td colspan="2" style="text-align: center;">All devices</td><td>6.0</td><td>10</td><td><math>\text{mA}</math></td><td><math>-40^{\circ}\text{C}</math></td><td colspan="2" rowspan="3">VDD = 4.2V</td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>6.2</td><td>10</td><td><math>\text{mA}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>6.6</td><td>10</td><td><math>\text{mA}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;">All devices</td><td>8.1</td><td>13</td><td><math>\text{mA}</math></td><td><math>-40^{\circ}\text{C}</math></td><td rowspan="3">VDD = 5.0V</td><td rowspan="3" style="vertical-align: middle; text-align: center;"><math>\text{Fosc} = 25 \text{ MHz}</math> <b>(PRI_IDLE mode, EC oscillator)</b></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>9.1</td><td>12</td><td><math>\text{mA}</math></td><td><math>+25^{\circ}\text{C}</math></td></tr> <tr><td colspan="2" style="text-align: center;"></td><td>8.3</td><td>12</td><td><math>\text{mA}</math></td><td><math>+85^{\circ}\text{C}</math></td></tr> </table>	PIC18LFX525/X620	65	100	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 2.0V	$\text{Fosc} = 1 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>		65	100	$\mu\text{A}$	$+25^{\circ}\text{C}$			70	110	$\mu\text{A}$	$+85^{\circ}\text{C}$		PIC18LFX525/X620		120	140	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.0V			120	140	$\mu\text{A}$	$+25^{\circ}\text{C}$			130	160	$\mu\text{A}$	$+85^{\circ}\text{C}$	All devices		230	300	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 5.0V			235	300	$\mu\text{A}$	$+25^{\circ}\text{C}$			240	300	$\mu\text{A}$	$+85^{\circ}\text{C}$	Extended devices only		260	500	$\mu\text{A}$	$+125^{\circ}\text{C}$	$\text{Fosc} = 4 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>	PIC18LFX525/X620		260	360	$\mu\text{A}$	$-40^{\circ}\text{C}$			255	360	$\mu\text{A}$	$+25^{\circ}\text{C}$			270	360	$\mu\text{A}$	$+85^{\circ}\text{C}$	PIC18LFX525/X620		420	620	$\mu\text{A}$	$-40^{\circ}\text{C}$			430	620	$\mu\text{A}$	$+25^{\circ}\text{C}$			450	650	$\mu\text{A}$	$+85^{\circ}\text{C}$	$\text{Fosc} = 25 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>	All devices		0.9	1.2	$\text{mA}$	$-40^{\circ}\text{C}$			0.9	1.2	$\text{mA}$	$+25^{\circ}\text{C}$			0.9	1.2	$\text{mA}$	$+85^{\circ}\text{C}$	Extended devices only		1	1.3	$\text{mA}$	$+125^{\circ}\text{C}$	Extended devices only		2.8	6.0	$\text{mA}$	$+125^{\circ}\text{C}$	VDD = 4.2V	$\text{Fosc} = 40 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>			4.3	8.0	$\text{mA}$	$+125^{\circ}\text{C}$	VDD = 5.0V	All devices		6.0	10	$\text{mA}$	$-40^{\circ}\text{C}$	VDD = 4.2V				6.2	10	$\text{mA}$	$+25^{\circ}\text{C}$			6.6	10	$\text{mA}$	$+85^{\circ}\text{C}$	All devices		8.1	13	$\text{mA}$	$-40^{\circ}\text{C}$	VDD = 5.0V	$\text{Fosc} = 25 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>			9.1	12	$\text{mA}$	$+25^{\circ}\text{C}$			8.3	12	$\text{mA}$	$+85^{\circ}\text{C}$
PIC18LFX525/X620	65	100	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 2.0V			$\text{Fosc} = 1 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>																																																																																																																																																																												
	65	100	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																																																																
	70	110	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																																																																																																
PIC18LFX525/X620		120	140	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 3.0V																																																																																																																																																																														
		120	140	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																																																															
		130	160	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																																																																																															
All devices		230	300	$\mu\text{A}$	$-40^{\circ}\text{C}$	VDD = 5.0V																																																																																																																																																																														
		235	300	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																																																															
		240	300	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																																																																																															
Extended devices only		260	500	$\mu\text{A}$	$+125^{\circ}\text{C}$	$\text{Fosc} = 4 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>																																																																																																																																																																														
PIC18LFX525/X620		260	360	$\mu\text{A}$	$-40^{\circ}\text{C}$																																																																																																																																																																															
		255	360	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																																																															
		270	360	$\mu\text{A}$	$+85^{\circ}\text{C}$																																																																																																																																																																															
PIC18LFX525/X620		420	620	$\mu\text{A}$	$-40^{\circ}\text{C}$																																																																																																																																																																															
		430	620	$\mu\text{A}$	$+25^{\circ}\text{C}$																																																																																																																																																																															
		450	650	$\mu\text{A}$	$+85^{\circ}\text{C}$	$\text{Fosc} = 25 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>																																																																																																																																																																														
All devices		0.9	1.2	$\text{mA}$	$-40^{\circ}\text{C}$																																																																																																																																																																															
		0.9	1.2	$\text{mA}$	$+25^{\circ}\text{C}$																																																																																																																																																																															
		0.9	1.2	$\text{mA}$	$+85^{\circ}\text{C}$																																																																																																																																																																															
Extended devices only		1	1.3	$\text{mA}$	$+125^{\circ}\text{C}$																																																																																																																																																																															
Extended devices only		2.8	6.0	$\text{mA}$	$+125^{\circ}\text{C}$	VDD = 4.2V	$\text{Fosc} = 40 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>																																																																																																																																																																													
		4.3	8.0	$\text{mA}$	$+125^{\circ}\text{C}$	VDD = 5.0V																																																																																																																																																																														
All devices		6.0	10	$\text{mA}$	$-40^{\circ}\text{C}$	VDD = 4.2V																																																																																																																																																																														
		6.2	10	$\text{mA}$	$+25^{\circ}\text{C}$																																																																																																																																																																															
		6.6	10	$\text{mA}$	$+85^{\circ}\text{C}$																																																																																																																																																																															
All devices		8.1	13	$\text{mA}$	$-40^{\circ}\text{C}$	VDD = 5.0V	$\text{Fosc} = 25 \text{ MHz}$ <b>(PRI_IDLE mode, EC oscillator)</b>																																																																																																																																																																													
		9.1	12	$\text{mA}$	$+25^{\circ}\text{C}$																																																																																																																																																																															
		8.3	12	$\text{mA}$	$+85^{\circ}\text{C}$																																																																																																																																																																															

  |  |  |  |  |  |  |

**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 tri-stated, pulled to VDD or VSS;

MCLR = VDD; WDT enabled/disabled as specified.

**3:** When operation below  $-10^{\circ}\text{C}$  is expected, use 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.

# PIC18F2525/2620/4525/4620

## 26.2 DC Characteristics: Power-Down and Supply Current

PIC18F2525/2620/4525/4620 (Industrial)

PIC18LF2525/2620/4525/4620 (Industrial) (Continued)

PIC18LF2525/2620/4525/4620 (Industrial)		Standard Operating Conditions (unless otherwise stated)				
PIC18F2525/2620/4525/4620 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated)				
Param No.	Device	Typ	Max	Units	Conditions	
<b>Supply Current (IDD)<sup>(2)</sup></b>						FOSC = 32 kHz <sup>(3)</sup> <b>(SEC_RUN mode, Timer1 as clock)</b>
PIC18LFX525/X620		10	25	µA	-40°C <sup>(3)</sup>	
		11	21	µA	+25°C	
		12	25	µA	+85°C	
PIC18LFX525/X620		42	57	µA	-40°C <sup>(3)</sup>	
		33	45	µA	+25°C	
		29	45	µA	+85°C	
All devices		105	150	µA	-40°C <sup>(3)</sup>	VDD = 5.0V
		81	130	µA	+25°C	
		67	130	µA	+85°C	
PIC18LFX525/X620		3.0	12	µA	-40°C <sup>(3)</sup>	VDD = 2.0V
		3.0	6	µA	+25°C	
		3.7	10	µA	+85°C	
PIC18LFX525/X620		5.0	15	µA	-40°C <sup>(3)</sup>	VDD = 3.0V
		5.4	10	µA	+25°C	
		6.3	15	µA	+85°C	
All devices		8.5	25	µA	-40°C <sup>(3)</sup>	VDD = 5.0V
		9.0	20	µA	+25°C	
		10.5	30	µA	+85°C	

**Legend:** Shading of rows is to assist in readability of the table.

- Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or Vss and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).
- 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD or Vss;  
 MCLR = VDD; WDT enabled/disabled as specified.

- 3:** When operation below -10°C is expected, use 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.

## 26.2 DC Characteristics: Power-Down and Supply Current

**PIC18F2525/2620/4525/4620 (Industrial)**  
**PIC18LF2525/2620/4525/4620 (Industrial) (Continued)**

PIC18LF2525/2620/4525/4620 (Industrial)		Standard Operating Conditions (unless otherwise stated)							
PIC18F2525/2620/4525/4620 (Industrial, Extended)		Standard Operating Conditions (unless otherwise stated)							
Param No.	Device	Typ	Max	Units	Conditions				
D026 ( $\Delta IAD$ )	<b>Module Differential Currents (<math>\Delta I_{WDT}</math>, <math>\Delta I_{BOR}</math>, <math>\Delta I_{LVD}</math>, <math>\Delta I_{OSCB}</math>, <math>\Delta I_{AD}</math>)</b>		A/D Converter	0.2	1.0	$\mu A$	-40°C to +85°C	VDD = 2.0V	A/D on, not converting
				0.2	1.0	$\mu A$	-40°C to +85°C	VDD = 3.0V	
				0.2	1.0	$\mu A$	-40°C to +85°C	VDD = 5.0V	
				0.5	4.0	$\mu A$	-40°C to +125°C		
D022 ( $\Delta I_{WDT}$ )	<b>Watchdog Timer</b>		Watchdog Timer	1.3	2.2	$\mu A$	-40°C	VDD = 2.0V	
				1.4	2.2	$\mu A$	+25°C		
				1.6	2.3	$\mu A$	+85°C		
				1.9	3.5	$\mu A$	-40°C		
				2.0	3.5	$\mu A$	+25°C		
				2.2	3.5	$\mu A$	+85°C		
				3.0	7.5	$\mu A$	-40°C		
				3.5	7.5	$\mu A$	+25°C		
				3.5	7.8	$\mu A$	+85°C		
				4.0	10	$\mu A$	+125°C		
D022A ( $\Delta I_{BOR}$ )	<b>Brown-out Reset<sup>(4)</sup></b>		Brown-out Reset <sup>(4)</sup>	35	50	$\mu A$	-40°C to +85°C	VDD = 3.0V	Sleep mode, BOREN1:BOREN0 = 10
				40	55	$\mu A$	-40°C to +85°C		
				55	65	$\mu A$	-40°C to +125°C		
				0	2	$\mu A$	-40°C to +85°C		
				0	5	$\mu A$	-40°C to +125°C		
D022B ( $\Delta I_{LVD}$ )	<b>High/Low-Voltage Detect<sup>(4)</sup></b>		High/Low-Voltage Detect <sup>(4)</sup>	22	38	$\mu A$	-40°C to +85°C	VDD = 2.0V	
				25	40	$\mu A$	-40°C to +85°C	VDD = 3.0V	
				29	45	$\mu A$	-40°C to +85°C		
				30	45	$\mu A$	-40°C to +125°C	VDD = 5.0V	

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or Vss and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD or Vss;  
MCLR = VDD; WDT enabled/disabled as specified.

**3:** When operation below -10°C is expected, use 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.

# PIC18F2525/2620/4525/4620

## 26.2 DC Characteristics: Power-Down and Supply Current

**PIC18F2525/2620/4525/4620 (Industrial)**  
**PIC18LF2525/2620/4525/4620 (Industrial) (Continued)**

<b>PIC18LF2525/2620/4525/4620</b> (Industrial)	<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial						
<b>PIC18F2525/2620/4525/4620</b> (Industrial, Extended)	<b>Standard Operating Conditions (unless otherwise stated)</b> Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended						
Param No.	Device	Typ	Max	Units	Conditions		
D025L ( $\Delta$ loscb)	Timer1 Oscillator	4.5	9.0	$\mu\text{A}$	$-40^{\circ}\text{C}$ <sup>(3)</sup>	VDD = 2.0V	32 kHz on Timer1
		0.9	1.6	$\mu\text{A}$	$-10^{\circ}\text{C}$		
		0.9	1.6	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		0.9	1.8	$\mu\text{A}$	$+85^{\circ}\text{C}$		
		4.8	10	$\mu\text{A}$	$-40^{\circ}\text{C}$ <sup>(3)</sup>		
		1.0	2.0	$\mu\text{A}$	$-10^{\circ}\text{C}$	VDD = 3.0V	32 kHz on Timer1
		1.0	2.0	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		1.0	2.6	$\mu\text{A}$	$+85^{\circ}\text{C}$		
		6.0	11	$\mu\text{A}$	$-40^{\circ}\text{C}$ <sup>(3)</sup>	VDD = 5.0V	32 kHz on Timer1
		1.6	4.0	$\mu\text{A}$	$-10^{\circ}\text{C}$		
		1.6	4.0	$\mu\text{A}$	$+25^{\circ}\text{C}$		
		1.6	4.0	$\mu\text{A}$	$+85^{\circ}\text{C}$		

**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 tri-stated, pulled to VDD or VSS;  
 MCLR = VDD; WDT enabled/disabled as specified.

- 3:** When operation below  $-10^{\circ}\text{C}$  is expected, use 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.

## 26.3 DC Characteristics: PIC18F2525/2620/4525/4620 (Industrial) PIC18LF2525/2620/4525/4620 (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D030 D030A D031 D031A D031B D032 D033 D033A D033B D034	VIL	<b>Input Low Voltage</b> I/O Ports: with TTL Buffer	Vss	0.15 VDD	V	VDD < 4.5V
			—	0.8	V	$4.5V \leq \text{VDD} \leq 5.5V$
		with Schmitt Trigger Buffer	Vss	0.2 VDD	V	
		RC3 and RC4	Vss	0.3 VDD	V	$\text{I}^2\text{C}^{\text{TM}}$ enabled
			Vss	0.8	V	SMBus enabled
		<u>MCLR</u>	Vss	0.2 VDD	V	
		OSC1	Vss	0.3 VDD	V	HS, HSPLL modes
		OSC1	Vss	0.2 VDD	V	RC, EC modes <sup>(1)</sup>
		OSC1	Vss	0.3	V	XT, LP modes
		T13CKI	Vss	0.3	V	
D040 D040A D041 D041A D041B D042 D043 D043A D043B D043C D044	VIH	<b>Input High Voltage</b> I/O Ports: with TTL Buffer	0.25 VDD + 0.8V	VDD	V	VDD < 4.5V
			2.0	VDD	V	$4.5V \leq \text{VDD} \leq 5.5V$
		with Schmitt Trigger Buffer	0.8 VDD	VDD	V	
		RC3 and RC4	0.7 VDD	VDD	V	$\text{I}^2\text{C}$ enabled
			2.1	VDD	V	SMBus enabled
		<u>MCLR</u>	0.8 VDD	VDD	V	
		OSC1	0.7 VDD	VDD	V	HS, HSPLL modes
		OSC1	0.8 VDD	VDD	V	EC mode
		OSC1	0.9 VDD	VDD	V	RC mode <sup>(1)</sup>
		OSC1	1.6	VDD	V	XT, LP modes
		T13CKI	1.6	VDD	V	
D060	IIL	<b>Input Leakage Current<sup>(2,3)</sup></b> I/O Ports	—	$\pm 200$	nA	VDD < 5.5V $\text{Vss} \leq \text{VPIN} \leq \text{VDD}$ , Pin at high-impedance
				$\pm 50$	nA	$\text{VDD} < 3V$ $\text{Vss} \leq \text{VPIN} \leq \text{VDD}$ , Pin at high-impedance
		<u>MCLR</u>	—	$\pm 1$	$\mu\text{A}$	$\text{Vss} \leq \text{VPIN} \leq \text{VDD}$
D061		OSC1	—	$\pm 1$	$\mu\text{A}$	$\text{Vss} \leq \text{VPIN} \leq \text{VDD}$
D063						
D070	IPU	<b>Weak Pull-up Current</b>	50	400	$\mu\text{A}$	$\text{VDD} = 5V$ , $\text{VPIN} = \text{Vss}$
D070	IPURB	PORTB Weak Pull-up Current				

**Note 1:** In RC oscillator configuration, the OSC1/CLK1 pin is a Schmitt Trigger input. It is not recommended that the PIC<sup>®</sup> device be driven with an external clock while in RC mode.

- 2: The leakage current on the 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.

# PIC18F2525/2620/4525/4620

## 26.3 DC Characteristics: PIC18F2525/2620/4525/4620 (Industrial) PIC18LF2525/2620/4525/4620 (Industrial) (Continued)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D080	VOL	Output Low Voltage I/O Ports	—	0.6	V	$I_{OL} = 8.5 \text{ mA}$ , $V_{DD} = 4.5\text{V}$ , $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
		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	Output High Voltage <sup>(3)</sup> 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 $+85^{\circ}\text{C}$
		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 $+85^{\circ}\text{C}$
Capacitive Loading Specs on Output Pins			—	—	—	—
D100	COSC2	OSC2 pin	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
D101	CIO	All I/O pins and OSC2 (in RC mode)	—	50	pF	To meet the AC Timing Specifications
D102	CB	SCL, SDA	—	400	pF	I <sup>2</sup> C <sup>TM</sup> Specification

- Note 1:** In RC oscillator configuration, the OSC1/CLK1 pin is a Schmitt Trigger input. It is not recommended that the PIC<sup>®</sup> device be driven with an external clock while in RC mode.
- 2:** The leakage current on the 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.

TABLE 26-1: MEMORY PROGRAMMING REQUIREMENTS

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial				
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D120	ED	<b>Data EEPROM Memory</b> Byte Endurance	100K	1M	—	E/W	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D121	VDRW	VDD for Read/Write	V <sub>MIN</sub>	—	5.5	V	Using EECON to read/write V <sub>MIN</sub> = Minimum operating voltage
D122	TDEW	Erase/Write Cycle Time	—	4	—	ms	
D123	TRETD	Characteristic Retention	40	—	—	Year	Provided no other specifications are violated
D124	TREF	Number of Total Erase/Write Cycles before Refresh <sup>(1)</sup>	1M	10M	—	E/W	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D125	IDD <sub>P</sub>	Supply Current during Programming	—	10	—	mA	
<b>Program Flash Memory</b>							
D130	EP	Cell Endurance	10K	100K	—	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	3.0	—	5.5	V	Using ICSP™ port, $+25^{\circ}\text{C}$
D132A	VIW	VDD for Externally Timed Erase or Write	4.5	—	5.5	V	Using ICSP™ port, $+25^{\circ}\text{C}$
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	$\text{VDD} \geq 4.5\text{V}$
D133A	TIW	ICSP Erase or Write Cycle Time (externally timed)	1	—	—	ms	$\text{VDD} \geq 4.5\text{V}, +25^{\circ}\text{C}$
D133A	TIW	Self-Timed Write Cycle Time	—	2	—	ms	
D134	TRETD	Characteristic Retention	40	100	—	Year	Provided no other specifications are violated
D135	IDD <sub>P</sub>	Supply Current during Programming	—	10	—	mA	

† Data in "Typ" column is at 5.0V,  $25^{\circ}\text{C}$  unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Refer to **Section 6.8 “Using the Data EEPROM”** for a more detailed discussion on data EEPROM endurance.

# PIC18F2525/2620/4525/4620

---



---

TABLE 26-2: COMPARATOR SPECIFICATIONS

Operating Conditions: $3.0V < VDD < 5.5V$ , $-40^{\circ}C < TA < +85^{\circ}C$ (unless otherwise stated).							
Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D300	VIOFF	Input Offset Voltage	—	$\pm 5.0$	$\pm 10$	mV	
D301	VICM	Input Common Mode Voltage	0	—	$VDD - 1.5$	V	
D302	CMRR	Common Mode Rejection Ratio	55	—	—	dB	
300	TRESP	Response Time <sup>(1)</sup>	—	150	400	ns	PIC18FXXXX
300A			—	150	600	ns	PIC18LFXXXX, $VDD = 2.0V$
301	TMC2OV	Comparator Mode Change to Output Valid	—	—	10	$\mu s$	

**Note 1:** Response time measured with one comparator input at  $(VDD - 1.5)/2$ , while the other input transitions from VSS to VDD.

TABLE 26-3: VOLTAGE REFERENCE SPECIFICATIONS

Operating Conditions: $3.0V < VDD < 5.5V$ , $-40^{\circ}C < TA < +85^{\circ}C$ (unless otherwise stated).							
Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D310	VRES	Resolution	$VDD/24$	—	$VDD/32$	LSb	
D311	VRAA	Absolute Accuracy	—	—	1/2	LSb	
D312	VRUR	Unit Resistor Value (R)	—	2k	—	$\Omega$	
310	TSET	Settling Time <sup>(1)</sup>	—	—	10	$\mu s$	

**Note 1:** Settling time measured while CVRR = 1 and CVR3:CVR0 transitions from '0000' to '1111'.

# PIC18F2525/2620/4525/4620

FIGURE 26-4: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

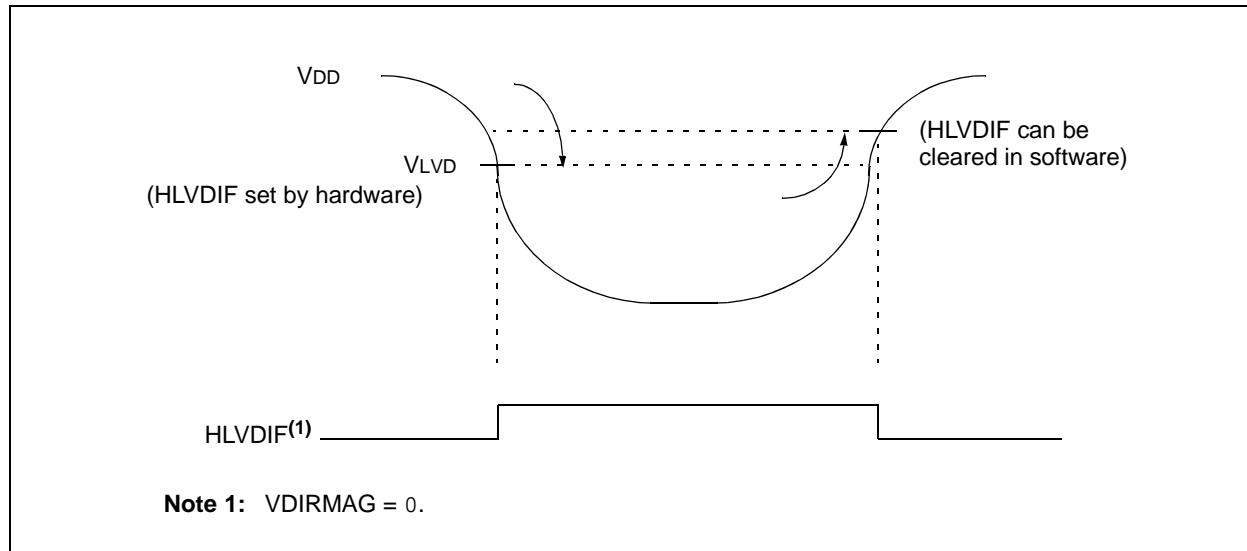


TABLE 26-4: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

Standard Operating Conditions (unless otherwise stated)							
Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial							
Param No.	Sym	Characteristic	Min	Typ	Max	Units	Conditions
D420		HLVD Voltage on VDD Transition High-to-Low	HLVDL = 0000	2.06	2.17	2.28	V
			HLVDL = 0001	2.12	2.23	2.34	V
			HLVDL = 0010	2.24	2.36	2.48	V
			HLVDL = 0011	2.32	2.44	2.56	V
			HLVDL = 0100	2.47	2.60	2.73	V
			HLVDL = 0101	2.65	2.79	2.93	V
			HLVDL = 0110	2.74	2.89	3.04	V
			HLVDL = 0111	2.96	3.12	3.28	V
			HLVDL = 1000	3.22	3.39	3.56	V
			HLVDL = 1001	3.37	3.55	3.73	V
			HLVDL = 1010	3.52	3.71	3.90	V
			HLVDL = 1011	3.70	3.90	4.10	V
			HLVDL = 1100	3.90	4.11	4.32	V
			HLVDL = 1101	4.11	4.33	4.55	V
			HLVDL = 1110	4.36	4.59	4.82	V

# PIC18F2525/2620/4525/4620

---

## 26.4 AC (Timing) Characteristics

### 26.4.1 TIMING PARAMETER SYMOLOGY

The timing parameter symbols have been created using one of the following formats:

1. TppS2ppS

3. Tcc:ST (I<sup>2</sup>C specifications only)

2. TppS

4. Ts (I<sup>2</sup>C specifications only)

T		T	Time
F	Frequency		

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T13CKI
mc	MCLR	wr	$\overline{WR}$

Uppercase letters and their meanings:

S		P	Period
F	Fall	R	Rise
H	High	V	Valid
I	Invalid (High-impedance)	Z	High-impedance
L	Low	High	High
I <sup>2</sup> C only		Low	Low
AA	output access		
BUF	Bus free		

Tcc:ST (I<sup>2</sup>C specifications only)

CC		SU	Setup
HD	Hold		
ST		STO	Stop condition
DAT	DATA input hold		
STA	Start condition		

## 26.4.2 TIMING CONDITIONS

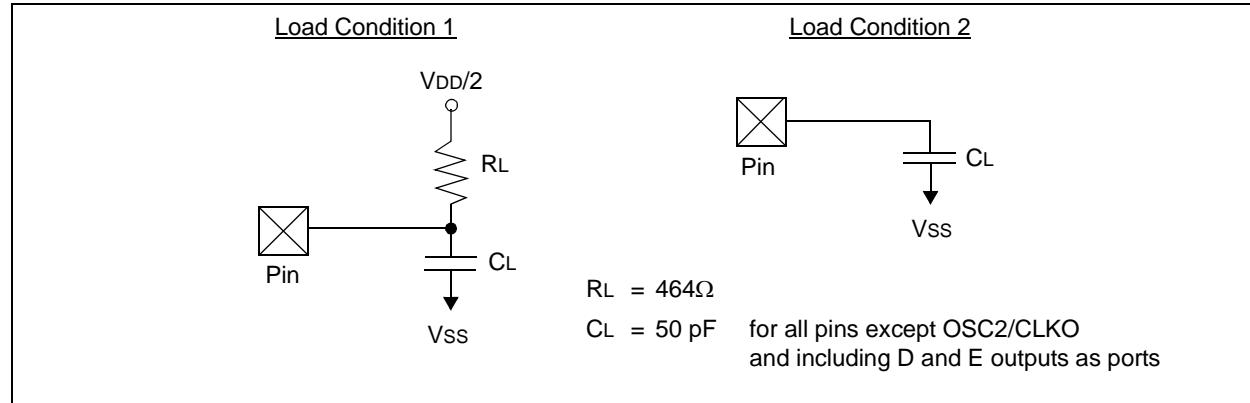
The temperature and voltages specified in Table 26-5 apply to all timing specifications unless otherwise noted. Figure 26-5 specifies the load conditions for the timing specifications.

**Note:** Because of space limitations, the generic terms "PIC18FXXXX" and "PIC18LFXXXX" are used throughout this section to refer to the PIC18F2525/2620/4525/4620 and PIC18LF2525/2620/4525/4620 families of devices specifically and only those devices.

**TABLE 26-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

<b>AC CHARACTERISTICS</b>	<b>Standard Operating Conditions (unless otherwise stated)</b>
	Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial Operating voltage VDD range as described in DC spec <b>Section 26.1</b> and <b>Section 26.3</b> . LF parts operate for industrial temperatures only.

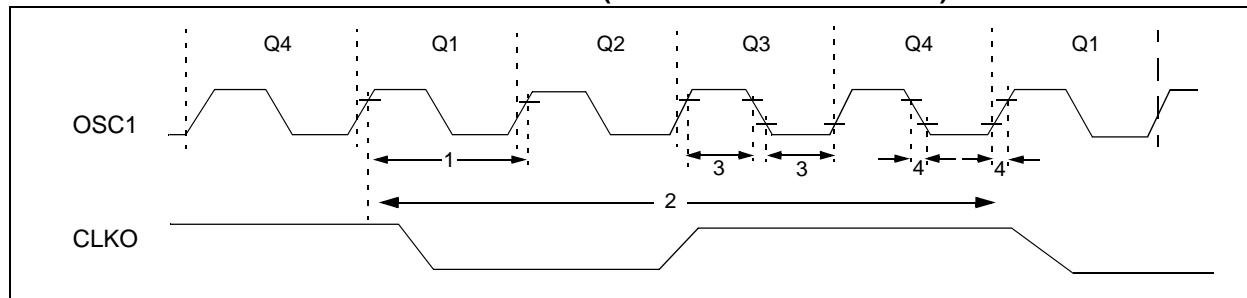
**FIGURE 26-5: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**



# PIC18F2525/2620/4525/4620

## 26.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 26-6: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)**



**TABLE 26-6: EXTERNAL CLOCK TIMING REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	FOSC	External CLK1 Frequency <sup>(1)</sup> Oscillator Frequency <sup>(1)</sup>	DC	1	MHz	XT, RC Oscillator mode
			DC	25	MHz	HS Oscillator mode
			DC	31.25	kHz	LP Oscillator mode
			DC	40	MHz	EC Oscillator mode
			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 CLK1 Period <sup>(1)</sup> Oscillator Period <sup>(1)</sup>	1000	—	ns	XT, RC Oscillator mode
			40	—	ns	HS Oscillator mode
			32	—	μs	LP Oscillator mode
			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/CLK1 pin. When an external clock input is used, the "max." cycle time limit is "DC" (no clock) for all devices.

# PIC18F2525/2620/4525/4620

TABLE 26-7: PLL CLOCK TIMING SPECIFICATIONS (VDD = 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 26-8: AC CHARACTERISTICS: INTERNAL RC ACCURACY

PIC18F2525/2620/4525/4620 (INDUSTRIAL)

PIC18LF2525/2620/4525/4620 (INDUSTRIAL)

PIC18LF2525/2620/4525/4620 (Industrial)		Standard Operating Conditions (unless otherwise stated)						
		Operating temperature -40°C ≤ TA ≤ +85°C for industrial						
PIC18F2525/2620/4525/4620 (Industrial)		Standard Operating Conditions (unless otherwise stated)						
		Operating temperature -40°C ≤ TA ≤ +85°C for industrial						
Param No.	Device	Min	Typ	Max	Units	Conditions		
	INTOSC Accuracy @ Freq = 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz, 125 kHz, 31 kHz <sup>(1)</sup>							
	PIC18LF2525/2620/4525/4620	-2	+/-1	2	%	+25°C	VDD = 2.7-3.3V	
		-5	+/-1	5	%	-40°C to +85°C	VDD = 2.7-3.3V	
	PIC18F2525/2620/4525/4620	-2	+/-1	2	%	+25°C	VDD = 4.5-5.5V	
		-5	+/-1	5	%	-40°C to +85°C	VDD = 4.5-5.5V	
	INTRC Accuracy @ Freq = 31 kHz							
	PIC18LF2525/2620/4525/4620	26.562	—	35.938	kHz	-40°C to +85°C	VDD = 2.7-3.3V	
	PIC18F2525/2620/4525/4620	26.562	—	35.938	kHz	-40°C to +85°C	VDD = 4.5-5.5V	

**Legend:** Shading of rows is to assist in readability of the table.

**Note 1:** Frequency calibrated at 25°C. OSCTUNE register can be used to compensate for temperature drift.

# PIC18F2525/2620/4525/4620

FIGURE 26-7: CLKO AND I/O TIMING

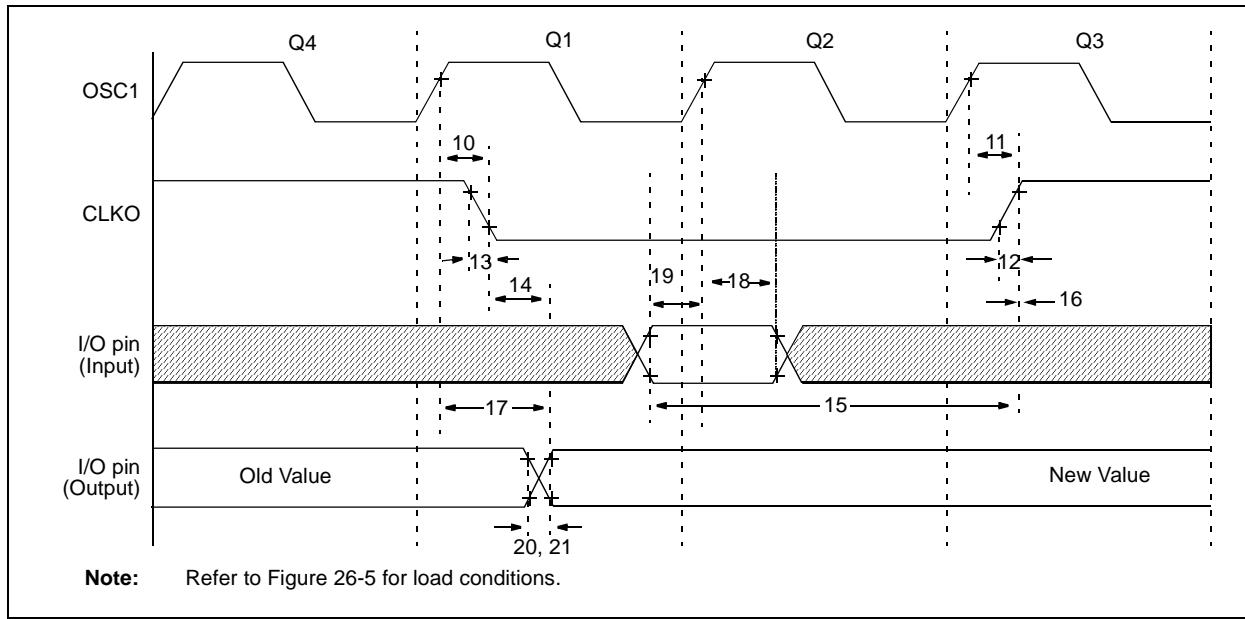


TABLE 26-9: CLKO AND I/O TIMING REQUIREMENTS

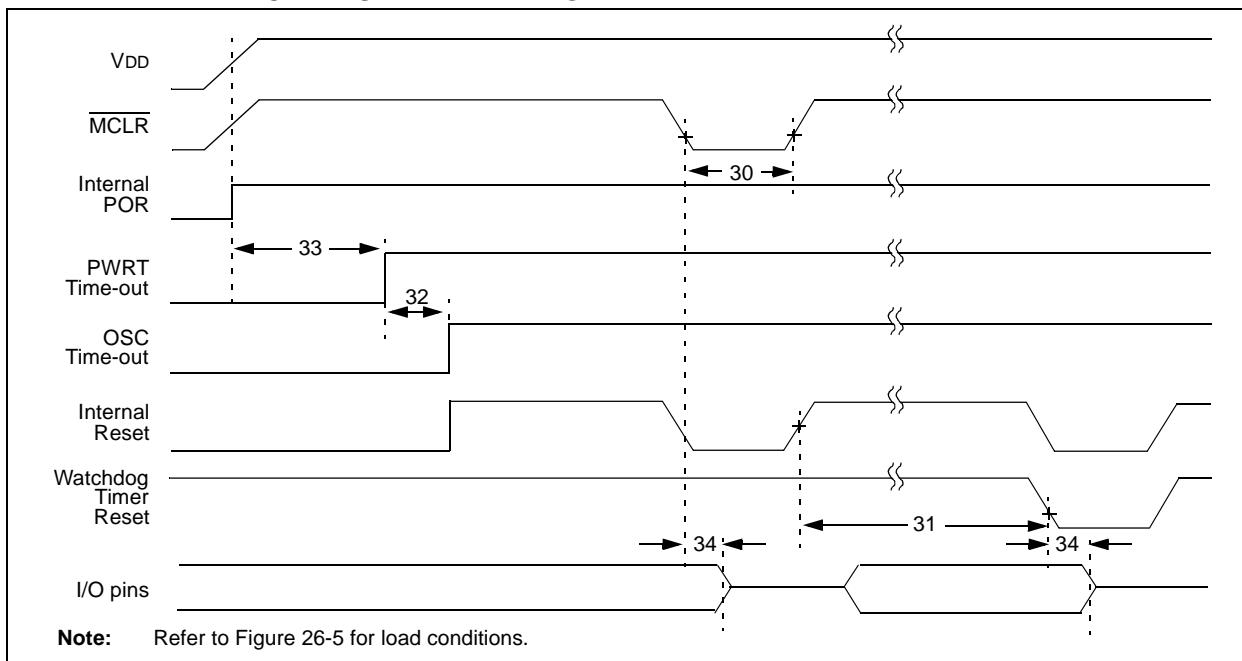
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
10	TosH2ckL	OSC1 $\uparrow$ to CLKO $\downarrow$	—	75	200	ns	(Note 1)
11	TosH2ckH	OSC1 $\uparrow$ to CLKO $\uparrow$	—	75	200	ns	(Note 1)
12	TckR	CLKO Rise Time	—	35	100	ns	(Note 1)
13	TckF	CLKO Fall Time	—	35	100	ns	(Note 1)
14	TckL2ioV	CLKO $\downarrow$ to Port Out Valid	—	—	0.5 TCY + 20	ns	(Note 1)
15	TioV2ckH	Port In Valid before CLKO $\uparrow$	0.25 TCY + 25	—	—	ns	(Note 1)
16	TckH2iol	Port In Hold after CLKO $\uparrow$	0	—	—	ns	(Note 1)
17	TosH2ioV	OSC1 $\uparrow$ (Q1 cycle) to Port Out Valid	—	50	150	ns	
18	TosH2iol	OSC1 $\uparrow$ (Q2 cycle) to Port Input Invalid (I/O in hold time)	100	—	—	ns	
18A			200	—	—	ns	VDD = 2.0V
19	TioV2osH	Port Input Valid to OSC1 $\uparrow$ (I/O in setup time)	0	—	—	ns	
20	TioR	Port Output Rise Time	PIC18FXXXX	—	10	25	ns
20A			PIC18LFXXXX	—	—	60	ns VDD = 2.0V
21	TioF	Port Output Fall Time	PIC18FXXXX	—	10	25	ns
21A			PIC18LFXXXX	—	—	60	ns VDD = 2.0V
22†	TINP	INTx pin High or Low Time	TCY	—	—	ns	
23†	TRBP	RB7:RB4 Change INTx High or Low Time	TCY	—	—	ns	

† These parameters are asynchronous events not related to any internal clock edges.

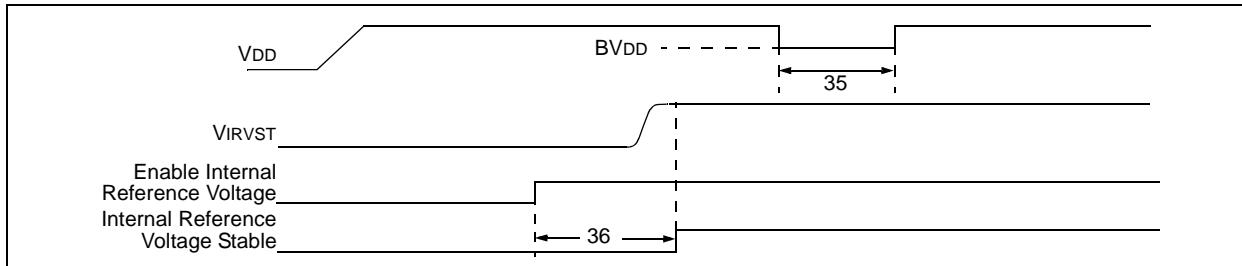
**Note 1:** Measurements are taken in RC mode, where CLKO output is 4 x Tosc.

# PIC18F2525/2620/4525/4620

**FIGURE 26-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**



**FIGURE 26-9: BROWN-OUT RESET TIMING**



**TABLE 26-10: 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>mCL</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>	Oscillation 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	55.6	65.5	75.4	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	VDD ≤ BVDD (see D005)
36	T <sub>IVRST</sub>	Time for Internal Reference Voltage to become Stable	—	20	50	μs	
37	T <sub>LVD</sub>	High/Low-Voltage Detect Pulse Width	200	—	—	μs	VDD ≤ VLVD
38	T <sub>CSD</sub>	CPU Start-up Time	—	10	—	μs	
39	T <sub>IOBST</sub>	Time for INTOSC to Stabilize	—	1	—	μs	

# PIC18F2525/2620/4525/4620

FIGURE 26-10: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

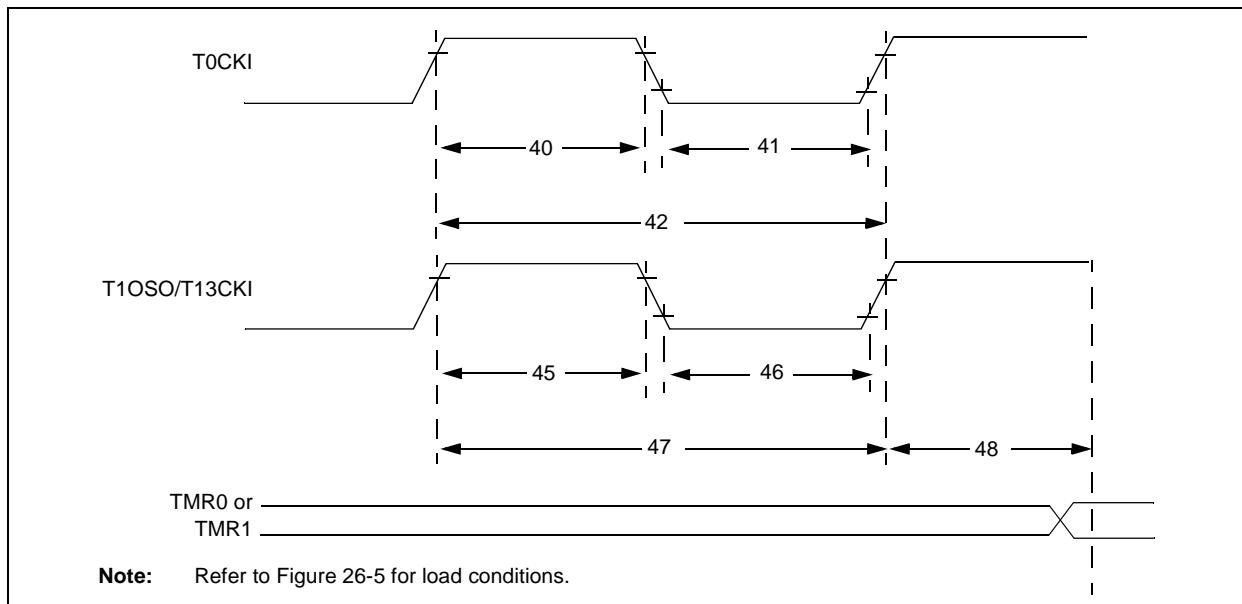
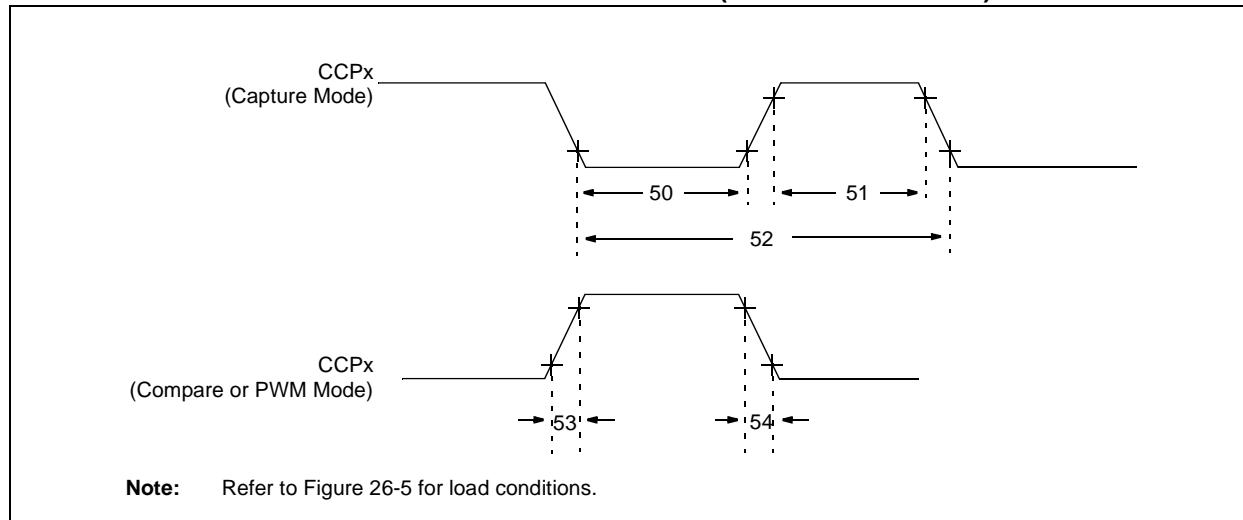


TABLE 26-11: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No prescaler	0.5 TCY + 20	—	ns	
			With prescaler	10	—	ns	
41	Tt0L	T0CKI Low Pulse Width	No prescaler	0.5 TCY + 20	—	ns	
			With prescaler	10	—	ns	
42	Tt0P	T0CKI Period	No prescaler	TCY + 10	—	ns	N = prescale value (1, 2, 4, ..., 256)
			With prescaler	Greater of: 20 ns or (TCY + 40)/N	—	ns	
45	Tt1H	T13CKI High Time	Synchronous, no prescaler	0.5 TCY + 20	—	ns	
			Synchronous, with prescaler	PIC18FXXXX	10	—	
			PIC18LFXXXX	25	—	ns	
			Asynchronous	PIC18FXXXX	30	—	
46	Tt1L	T13CKI Low Time	PIC18LFXXXX	50	—	ns	VDD = 2.0V
			Synchronous, no prescaler	0.5 TCY + 5	—	ns	
			Synchronous, with prescaler	PIC18FXXXX	10	—	
			PIC18LFXXXX	25	—	ns	
47	Tt1P	T13CKI Input Period	Asynchronous	PIC18FXXXX	30	—	VDD = 2.0V
			PIC18LFXXXX	50	—	ns	
48	Ft1	T13CKI Oscillator Input Frequency Range	Synchronous	Greater of: 20 ns or (TCY + 40)/N	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous	60	—	ns	
48	Tcke2tmrl	Delay from External T13CKI Clock Edge to Timer Increment	DC	50	kHz		
			2 Tosc	7 Tosc	—		

# PIC18F2525/2620/4525/4620

**FIGURE 26-11: CAPTURE/COMPARE/PWM TIMINGS (ALL CCP MODULES)**



**TABLE 26-12: CAPTURE/COMPARE/PWM REQUIREMENTS (ALL CCP MODULES)**

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
50	TccL	CCPx Input Low Time	No prescaler	0.5 TCY + 20	—	ns	VDD = 2.0V
			With prescaler	PIC18FXXXX	10	—	
			With prescaler	PIC18LFXXXX	20	—	
51	TccH	CCPx Input High Time	No prescaler	0.5 TCY + 20	—	ns	VDD = 2.0V
			With prescaler	PIC18FXXXX	10	—	
			With prescaler	PIC18LFXXXX	20	—	
52	TccP	CCPx Input Period		<u>3 TCY + 40</u> N	—	ns	N = prescale value (1, 4 or 16)
53	TccR	CCPx Output Fall Time		PIC18FXXXX	—	25	ns
				PIC18LFXXXX	—	45	ns
54	TccF	CCPx Output Fall Time		PIC18FXXXX	—	25	ns
				PIC18LFXXXX	—	45	ns

# PIC18F2525/2620/4525/4620

FIGURE 26-12: PARALLEL SLAVE PORT TIMING (PIC18F4525/4620)

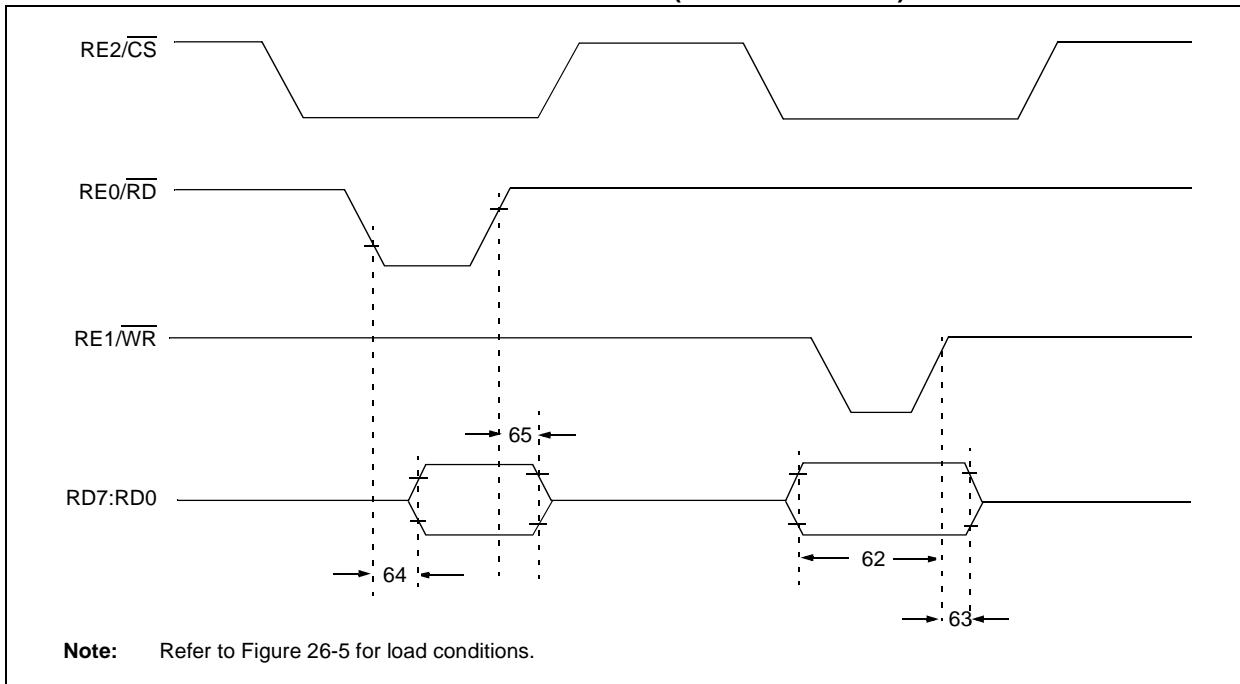


TABLE 26-13: PARALLEL SLAVE PORT REQUIREMENTS (PIC18F4525/4620)

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
62	TdtV2wrH	Data In Valid before WR $\uparrow$ or CS $\uparrow$ (setup time)	20	—	ns	
63	TwrH2dtl	WR $\uparrow$ or CS $\uparrow$ to Data-In Invalid (hold time)	20	—	ns	
		PIC18FXXXX	35	—	ns	VDD = 2.0V
64	TrdL2dtV	RD $\downarrow$ and CS $\downarrow$ to Data-Out Valid	—	80	ns	
65	TrdH2dtl	RD $\uparrow$ or CS $\downarrow$ to Data-Out Invalid	10	30	ns	
66	TibfINH	Inhibit of the IBF Flag bit being Cleared from WR $\uparrow$ or CS $\uparrow$	—	3 TCY		

FIGURE 26-13: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)

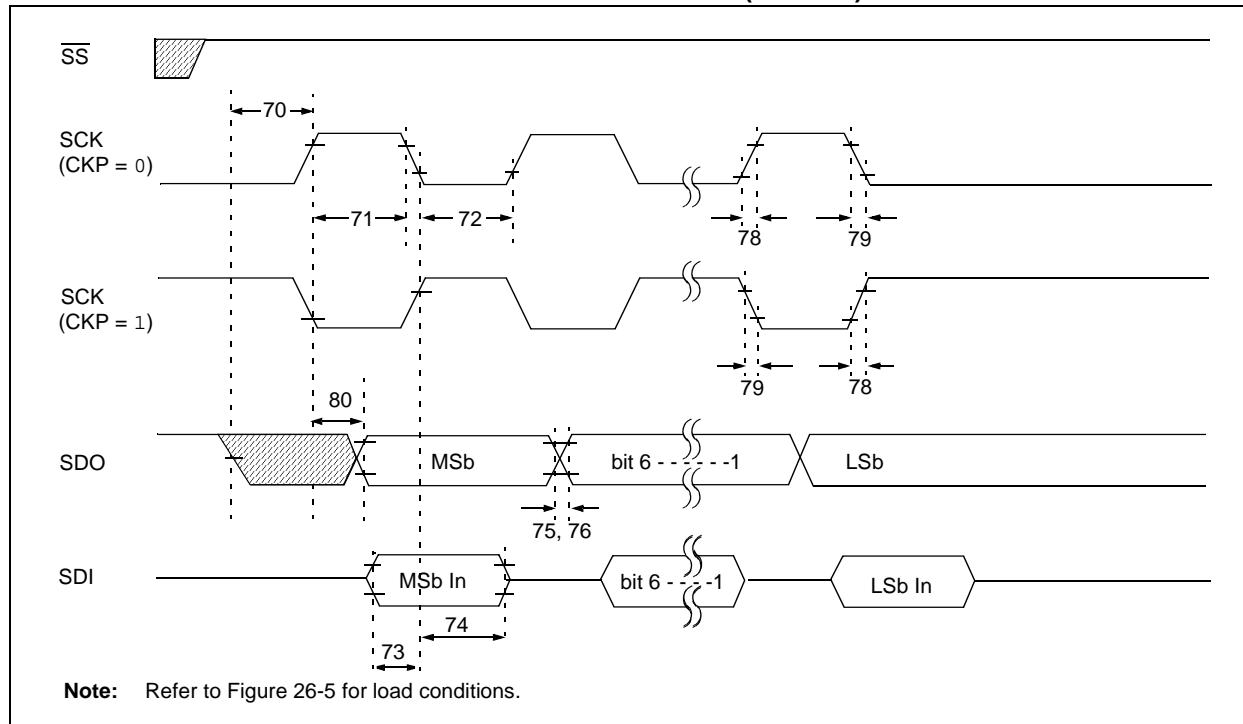


TABLE 26-14: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	SS ↓ to SCK ↓ or SCK ↑ Input		TCY	—	ns	
71	Tsch	SCK Input High Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
		Single Byte	40	—	ns		(Note 1)
72	TscL	SCK Input Low Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
		Single Byte	40	—	ns		(Note 1)
73	TdiV2scH, TdiV2scL	Setup Time of SDI Data Input to SCK Edge		20	—	ns	
73A	Tb2b	Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2		1.5 TCY + 40	—	ns	(Note 2)
74	Tsch2diL, TscL2diL	Hold Time of SDI Data Input to SCK Edge		40	—	ns	
75	TdoR	SDO Data Output Rise Time	PIC18FXXXX	—	25	ns	
			PIC18LFXXXX	—	45	ns	VDD = 2.0V
76	TdoF	SDO Data Output Fall Time		—	25	ns	
78	TscR	SCK Output Rise Time (Master mode)	PIC18FXXXX	—	25	ns	
			PIC18LFXXXX	—	45	ns	VDD = 2.0V
79	TscF	SCK Output Fall Time (Master mode)		—	25	ns	
80	TscH2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXXXX	—	50	ns	
			PIC18LFXXXX	—	100	ns	VDD = 2.0V

Note 1: Requires the use of Parameter #73A.

2: Only if Parameter #71A and #72A are used.

# PIC18F2525/2620/4525/4620

FIGURE 26-14: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)

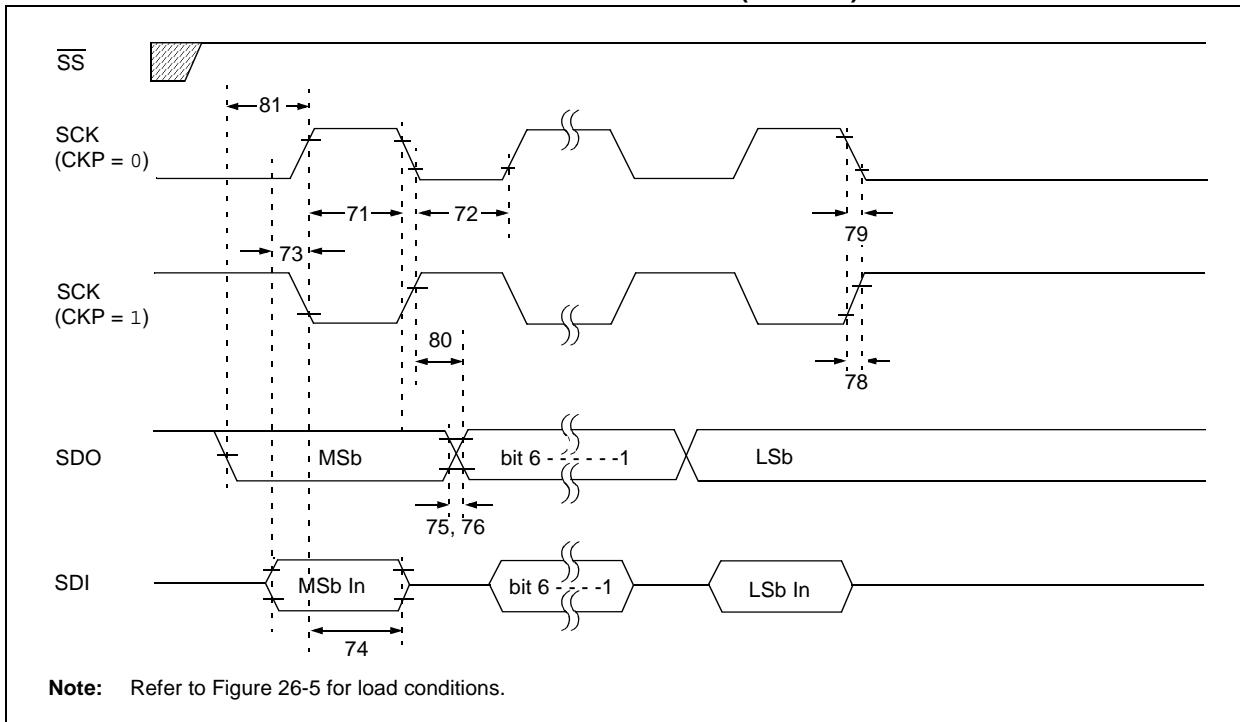


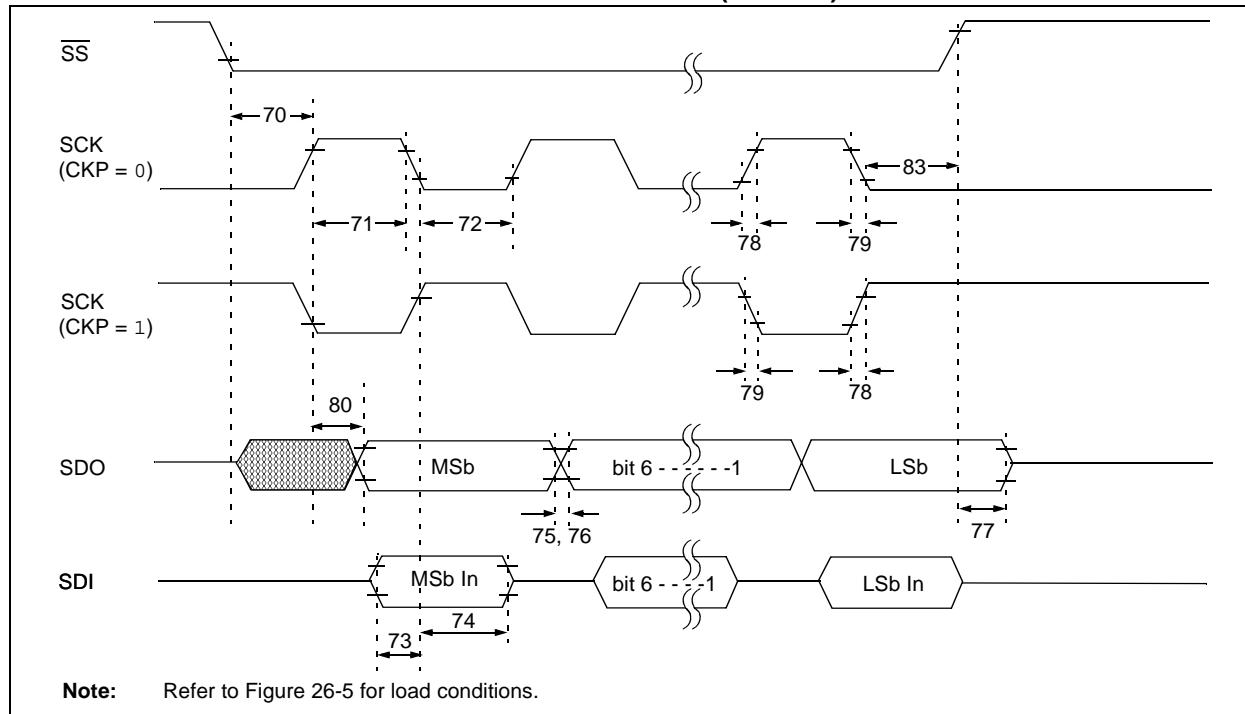
TABLE 26-15: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
71 71A	TscH	SCK Input High Time (Slave mode)	Continuous	1.25 T <sub>CY</sub> + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
72 72A	TscL	SCK Input Low Time (Slave mode)	Continuous	1.25 T <sub>CY</sub> + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
73	TdiV2scH, TdiV2scL	Setup Time of SDI Data Input to SCK Edge		20	—	ns	
73A	Tb2b	Last Clock Edge of Byte 1 to the 1st 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	
			PIC18LFXXXX		45	ns	V <sub>DD</sub> = 2.0V
76	TdoF	SDO Data Output Fall Time		—	25	ns	
78	TscR	SCK Output Rise Time (Master mode)	PIC18FXXXX	—	25	ns	
			PIC18LFXXXX		45	ns	V <sub>DD</sub> = 2.0V
79	TscF	SCK Output Fall Time (Master mode)		—	25	ns	
80	TscH2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXXXX	—	50	ns	
			PIC18LFXXXX		100	ns	V <sub>DD</sub> = 2.0V
81	TdoV2scH, TdoV2scL	SDO Data Output Setup to SCK Edge		T <sub>CY</sub>	—	ns	

**Note 1:** Requires the use of Parameter #73A.

**2:** Only if Parameter #71A and #72A are used.

**FIGURE 26-15: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)**



# PIC18F2525/2620/4525/4620

TABLE 26-16: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	SS ↓ to SCK ↓ or SCK ↑ Input	3 TCY	—	ns	
71 71A	TscH	SCK Input High Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns
			Single Byte	40	—	ns <b>(Note 1)</b>
72 72A	TscL	SCK Input Low Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns
			Single Byte	40	—	ns <b>(Note 1)</b>
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 TCY + 40	—	ns	<b>(Note 2)</b>
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 $V_{DD} = 2.0V$
76	TdoF	SDO Data Output Fall Time	—	25	ns	
77	TssH2doZ	SS ↑ to SDO Output High-Impedance	10	50	ns	
78	TscR	SCK Output Rise Time (Master mode)	PIC18FXXXX	—	25	ns
			PIC18LFXXXX		45	ns $V_{DD} = 2.0V$
79	TscF	SCK Output Fall Time (Master mode)	—	25	ns	
80	Tsch2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXXXX	—	50	ns
			PIC18LFXXXX		100	ns $V_{DD} = 2.0V$
83	Tsch2ssH, TscL2ssH	SS ↑ after SCK edge	1.5 TCY + 40	—	ns	

**Note 1:** Requires the use of Parameter #73A.

**2:** Only if Parameter #71A and #72A are used.

FIGURE 26-16: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)

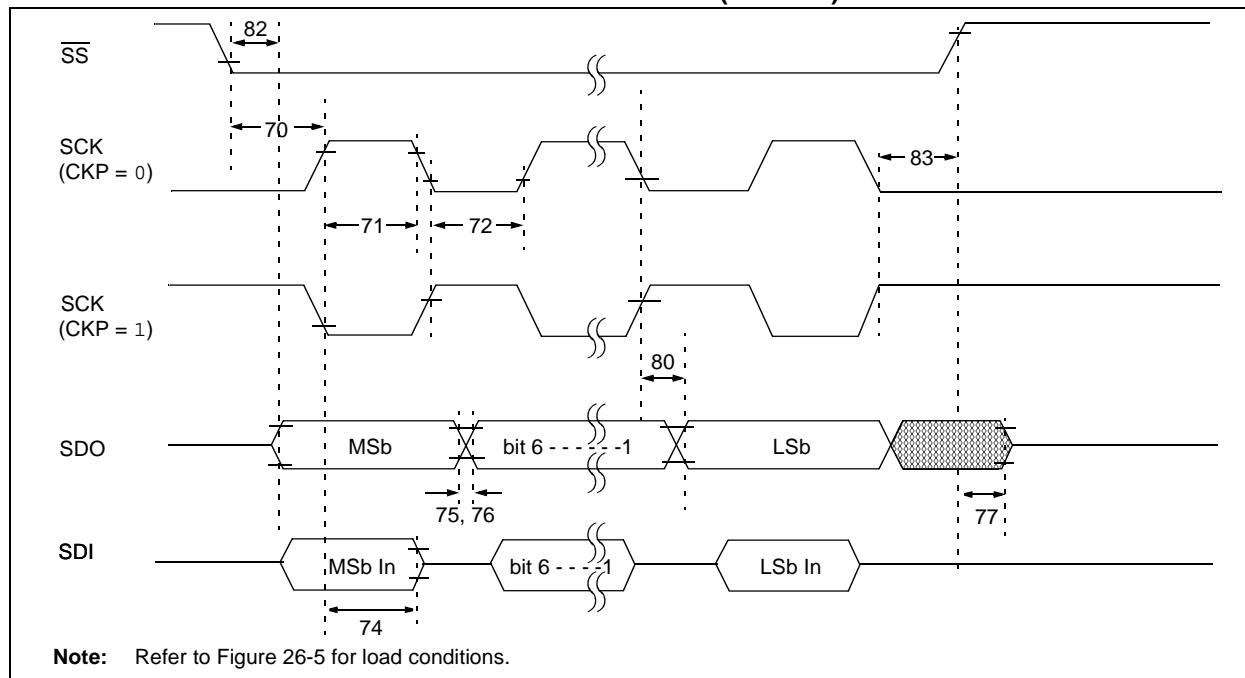


TABLE 26-17: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)

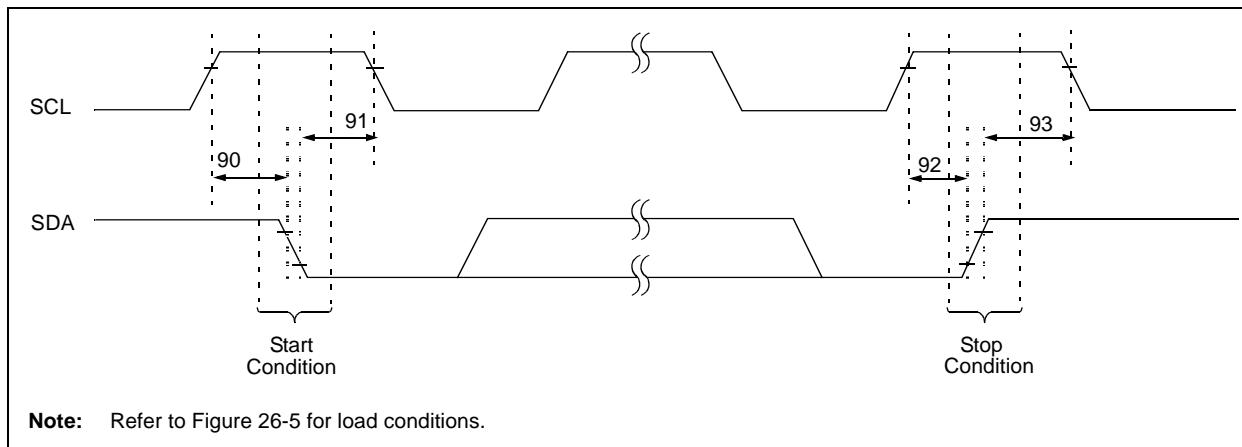
Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	SS ↓ to SCK ↓ or SCK ↑ Input		3 TCY	—	ns	
71	TscH	SCK Input High Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
72	TscL	SCK Input Low Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
73A	Tb2b	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2		1.5 TCY + 40	—	ns	(Note 2)
74	Tsch2diL, TscL2diL	Hold Time of SDI Data Input to SCK Edge		40	—	ns	
75	TdoR	SDO Data Output Rise Time	PIC18FXXXX	—	25	ns	
			PIC18LFXXXX	—	45	ns	VDD = 2.0V
76	TdoF	SDO Data Output Fall Time		—	25	ns	
77	TssH2doZ	SS ↑ to SDO Output High-Impedance		10	50	ns	
78	TscR	SCK Output Rise Time (Master mode)	PIC18FXXXX	—	25	ns	
			PIC18LFXXXX	—	45	ns	VDD = 2.0V
79	TscF	SCK Output Fall Time (Master mode)		—	25	ns	
80	Tsch2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXXXX	—	50	ns	
			PIC18LFXXXX	—	100	ns	VDD = 2.0V
82	TssL2doV	SDO Data Output Valid after SS ↓ Edge	PIC18FXXXX	—	50	ns	
			PIC18LFXXXX	—	100	ns	VDD = 2.0V
83	Tsch2ssH, TscL2ssH	SS ↑ after SCK Edge		1.5 TCY + 40	—	ns	

Note 1: Requires the use of Parameter #73A.

2: Only if Parameter #71A and #72A are used.

# PIC18F2525/2620/4525/4620

**FIGURE 26-17: I<sup>2</sup>C<sup>TM</sup> BUS START/STOP BITS TIMING**



**TABLE 26-18: I<sup>2</sup>C<sup>TM</sup> BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
90	TSU:STA	Start Condition	100 kHz mode	4700	—	ns Only relevant for Repeated Start condition
		Setup Time	400 kHz mode	600	—	
91	THD:STA	Start Condition	100 kHz mode	4000	—	ns After this period, the first clock pulse is generated
		Hold Time	400 kHz mode	600	—	
92	TSU:STO	Stop Condition	100 kHz mode	4700	—	ns
		Setup Time	400 kHz mode	600	—	
93	THD:STO	Stop Condition	100 kHz mode	4000	—	ns
		Hold Time	400 kHz mode	600	—	

**FIGURE 26-18: I<sup>2</sup>C<sup>TM</sup> BUS DATA TIMING**

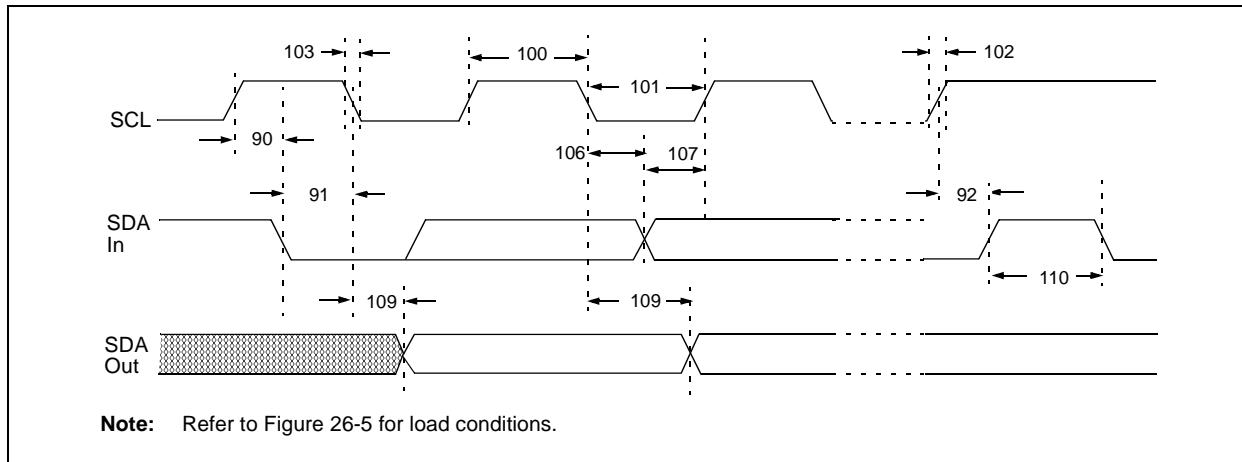


TABLE 26-19: I<sup>2</sup>C™ BUS DATA REQUIREMENTS (SLAVE MODE)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	4.0	—	μs	
			400 kHz mode	0.6	—	μs	
			MSSP module	1.5 TCY	—		
101	TLOW	Clock Low Time	100 kHz mode	4.7	—	μs	
			400 kHz mode	1.3	—	μs	
			MSSP module	1.5 TCY	—		
102	TR	SDA and SCL Rise Time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	CB is specified to be from 10 to 400 pF
103	TF	SDA and SCL Fall Time	100 kHz mode	—	300	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	CB is specified to be from 10 to 400 pF
90	TSU:STA	Start Condition Setup Time	100 kHz mode	4.7	—	μs	Only relevant for Repeated Start condition
			400 kHz mode	0.6	—	μs	
91	THD:STA	Start Condition Hold Time	100 kHz mode	4.0	—	μs	After this period, the first clock pulse is generated
			400 kHz mode	0.6	—	μs	
106	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	μs	
107	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	(Note 2)
			400 kHz mode	100	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	4.7	—	μs	
			400 kHz mode	0.6	—	μs	
109	TAA	Output Valid from Clock	100 kHz mode	—	3500	ns	(Note 1)
			400 kHz mode	—	—	ns	
110	TBUF	Bus Free Time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	μs	
D102	CB	Bus Capacitive Loading		—	400	pF	

**Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

**2:** A Fast mode I<sup>2</sup>C bus device can be used in a Standard mode I<sup>2</sup>C bus system, but the requirement  $TSU:DAT \geq 250$  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.

# PIC18F2525/2620/4525/4620

FIGURE 26-19: MASTER SSP I<sup>2</sup>C<sup>TM</sup> BUS START/STOP BITS TIMING WAVEFORMS

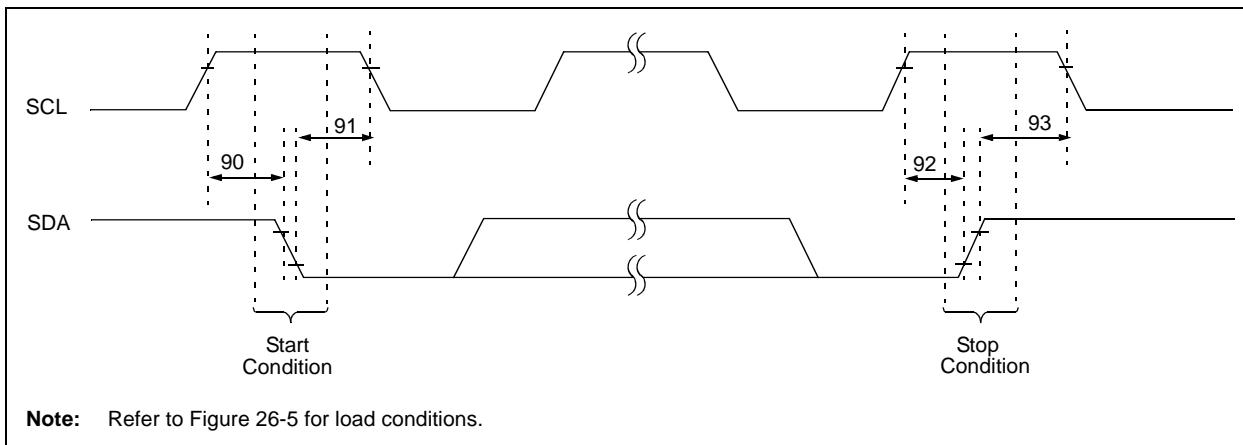


TABLE 26-20: MASTER SSP I<sup>2</sup>C<sup>TM</sup> BUS START/STOP BITS REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns Only relevant for Repeated Start condition
			400 kHz mode	2(Tosc)(BRG + 1)	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns After this period, the first clock pulse is generated
			400 kHz mode	2(Tosc)(BRG + 1)	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns
			400 kHz mode	2(Tosc)(BRG + 1)	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	
93	THD:STO	Stop Condition Hold Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ns
			400 kHz mode	2(Tosc)(BRG + 1)	—	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	

**Note 1:** Maximum pin capacitance = 10 pF for all I<sup>2</sup>C pins.

FIGURE 26-20: MASTER SSP I<sup>2</sup>C<sup>TM</sup> BUS DATA TIMING

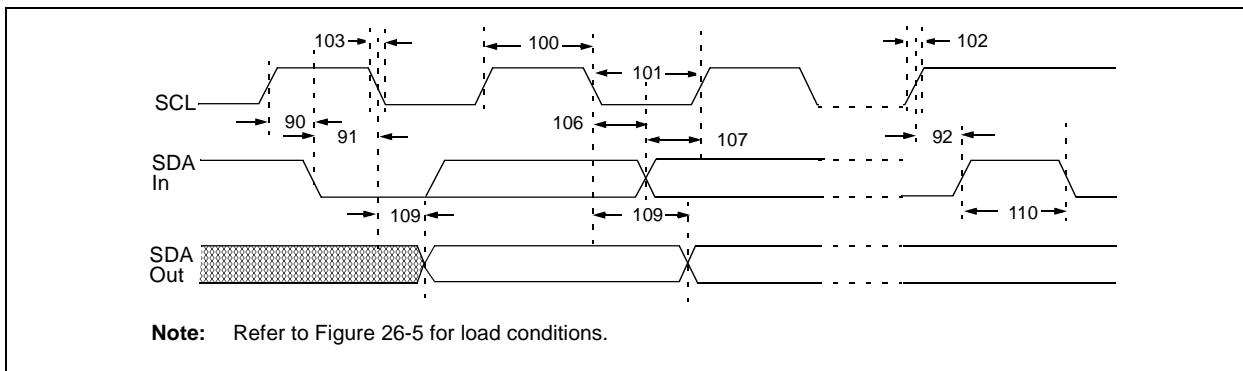


TABLE 26-21: MASTER SSP I<sup>2</sup>C™ BUS DATA REQUIREMENTS

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	CB is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 CB	300	ns	
			1 MHz mode <sup>(1)</sup>	—	300	ns	
103	TF	SDA and SCL Fall Time	100 kHz mode	—	300	ns	CB is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 CB	300	ns	
			1 MHz mode <sup>(1)</sup>	—	100	ns	
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	Only relevant for Repeated Start condition
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	ms	
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	After this period, the first clock pulse is generated
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			1 MHz mode <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	(Note 2)
			400 kHz mode	100	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			1 MHz mode <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	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	ms	
D102	CB	Bus Capacitive Loading		—	400	pF	

**Note 1:** Maximum pin capacitance = 10 pF for all 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  $\geq$  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.

# PIC18F2525/2620/4525/4620

FIGURE 26-21: EUSART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

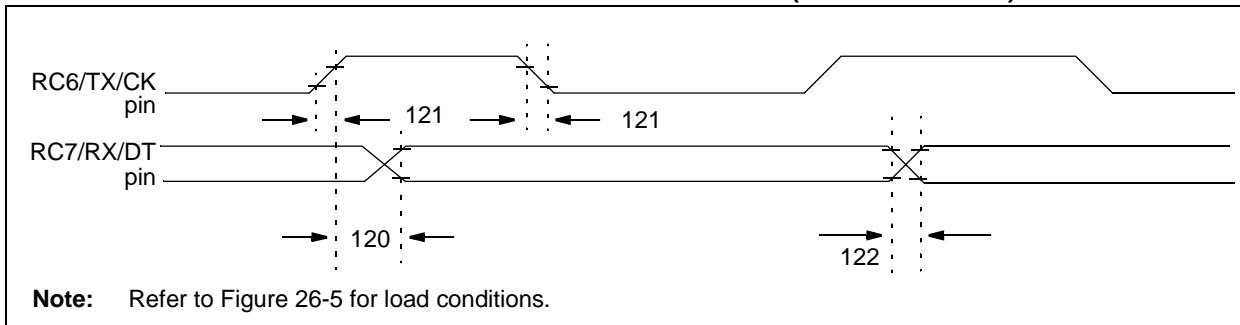


TABLE 26-22: EUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
120	TckH2dtV	SYNC XMIT (MASTER & SLAVE) Clock High to Data Out Valid	PIC18FXXXX	—	40	ns
			PIC18LFXXXX	—	100	ns VDD = 2.0V
121	Tckrf	Clock Out Rise Time and Fall Time (Master mode)	PIC18FXXXX	—	20	ns
			PIC18LFXXXX	—	50	ns VDD = 2.0V
122	Tdtrf	Data Out Rise Time and Fall Time	PIC18FXXXX	—	20	ns
			PIC18LFXXXX	—	50	ns VDD = 2.0V

# PIC18F2525/2620/4525/4620

FIGURE 26-22: EUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

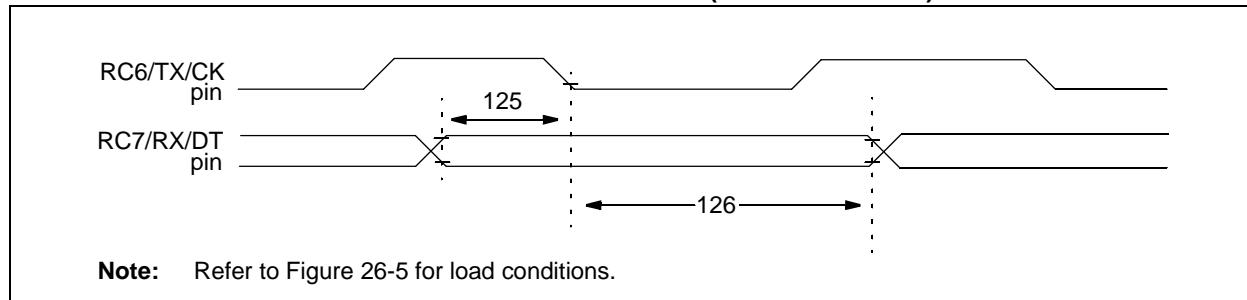


TABLE 26-23: EUSART SYNCHRONOUS RECEIVE REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TdtV2ckl	SYNC RCV (MASTER & SLAVE) Data Hold before CK ↓ (DT hold time)	10	—	ns	
126	TckL2dtl	Data Hold after CK ↓ (DT hold time)	15	—	ns	

TABLE 26-24: A/D CONVERTER CHARACTERISTICS: PIC18F2525/2620/4525/4620 (INDUSTRIAL)  
PIC18LF2525/2620/4525/4620 (INDUSTRIAL)

Param. No.	Symbol	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	E0FF	Offset Error	—	—	$<\pm 2.0$	LSb	$\Delta V_{REF} \geq 3.0V$
A07	EGN	Gain Error	—	—	$<\pm 1$	LSb	$\Delta V_{REF} \geq 3.0V$
A10	—	Monotonicity	Guaranteed <sup>(1)</sup>			—	$V_{SS} \leq V_{AIN} \leq V_{REF}$
A20	$\Delta V_{REF}$	Reference Voltage Range ( $V_{REFH} - V_{REFL}$ )	1.8 3	— —	— —	V V	$V_{DD} < 3.0V$ $V_{DD} \geq 3.0V$
A21	VREFH	Reference Voltage High	V <sub>SS</sub>	—	V <sub>REFH</sub>	V	
A22	VREFL	Reference Voltage Low	V <sub>SS</sub> – 0.3V	—	V <sub>DD</sub> – 3.0V	V	
A25	V <sub>AIN</sub>	Analog Input Voltage	V <sub>REFL</sub>	—	V <sub>REFH</sub>	V	
A30	Z <sub>AIN</sub>	Recommended Impedance of Analog Voltage Source	—	—	2.5	k $\Omega$	
A40	I <sub>A</sub> D	A/D Current from V <sub>DD</sub>	PIC18FXXXX PIC18LFXX20	— —	180 90	$\mu A$ $\mu A$	Average current during conversion
A50	I <sub>REF</sub>	V <sub>REF</sub> Input Current <sup>(2)</sup>	— —	— —	5 150	$\mu A$ $\mu A$	During V <sub>AIN</sub> acquisition. During A/D conversion cycle.

**Note 1:** The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

**2:** V<sub>REFH</sub> current is from RA3/AN3/V<sub>REF+</sub> pin or V<sub>DD</sub>, whichever is selected as the V<sub>REFH</sub> source.  
V<sub>REFL</sub> current is from RA2/AN2/V<sub>REF-</sub>/CVREF pin or V<sub>SS</sub>, whichever is selected as the V<sub>REFL</sub> source.

# PIC18F2525/2620/4525/4620

FIGURE 26-23: A/D CONVERSION TIMING

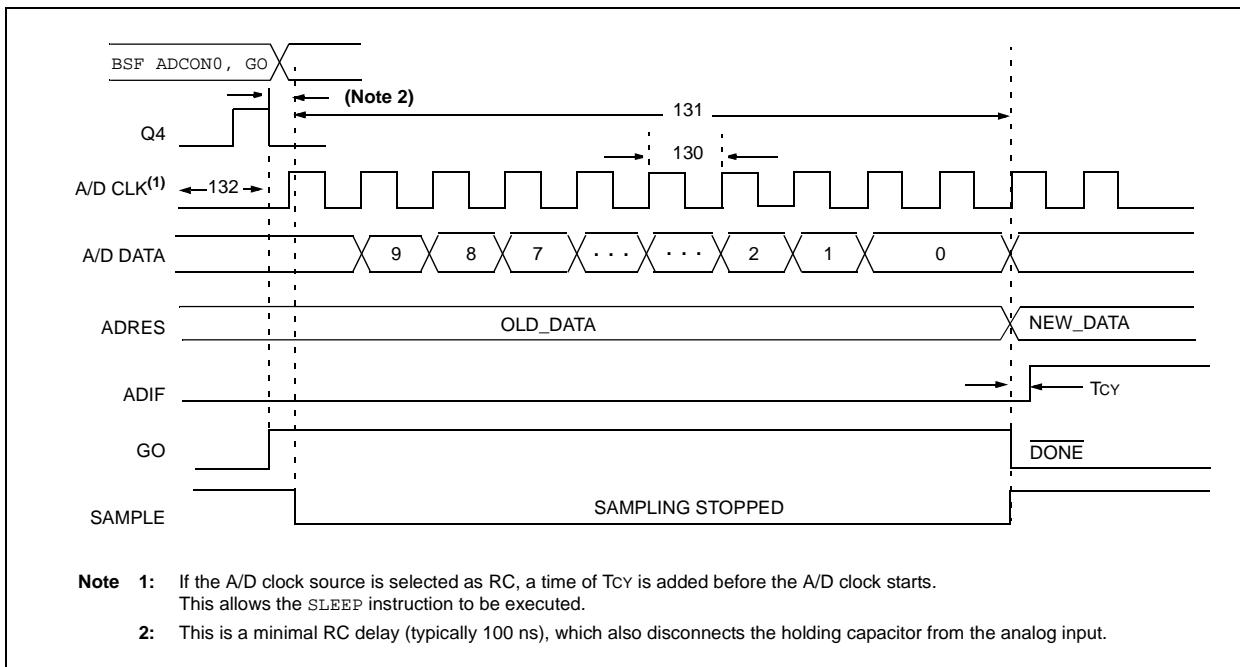


TABLE 26-25: A/D CONVERSION REQUIREMENTS

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, V <sub>REF</sub> ≥ 3.0V
			PIC18LFXXXX	1.4	25.0 <sup>(1)</sup>	μs	V <sub>DD</sub> = 2.0V; Tosc based, V <sub>REF</sub> full range
			PIC18FXXXX	—	1	μs	A/D RC mode
			PIC18LFXXXX	—	3	μs	V <sub>DD</sub> = 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	TSCW	Switching Time from Convert → Sample		—	(Note 4)		
TBD	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.

**2:** ADRES register may be read on the following T<sub>CY</sub> cycle.

**3:** The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (V<sub>DD</sub> to V<sub>SS</sub> or V<sub>SS</sub> to V<sub>DD</sub>). The source impedance (R<sub>S</sub>) on the input channels is 50Ω.

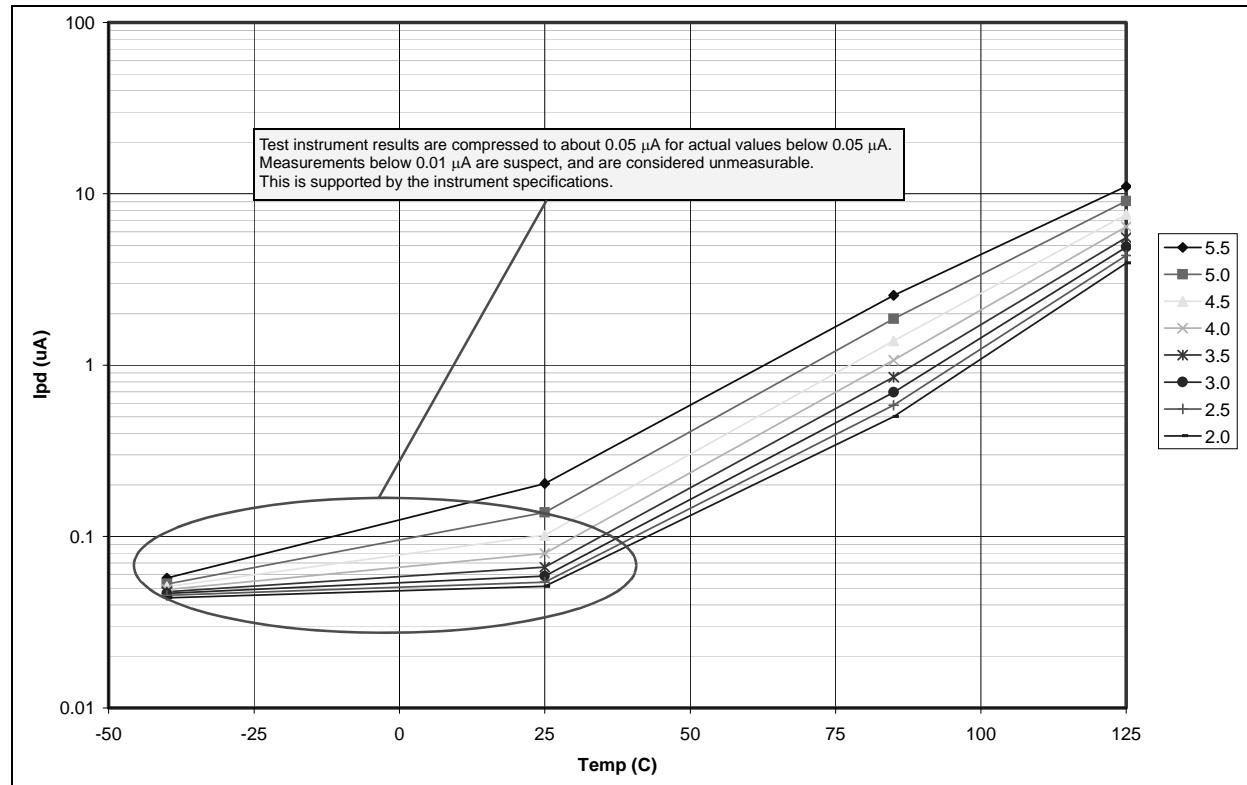
**4:** On the following cycle of the device clock.

## 27.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

**Note:** The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "Maximum" or "minimum" represents (mean + 3 $\sigma$ ) or (mean - 3 $\sigma$ ) respectively, where  $\sigma$  is a standard deviation, over the whole temperature range.

**FIGURE 27-1: SLEEP MODE**



# PIC18F2525/2620/4525/4620

FIGURE 27-2: TYPICAL IPD VS. VDD ACROSS TEMPERATURE (SLEEP MODE)

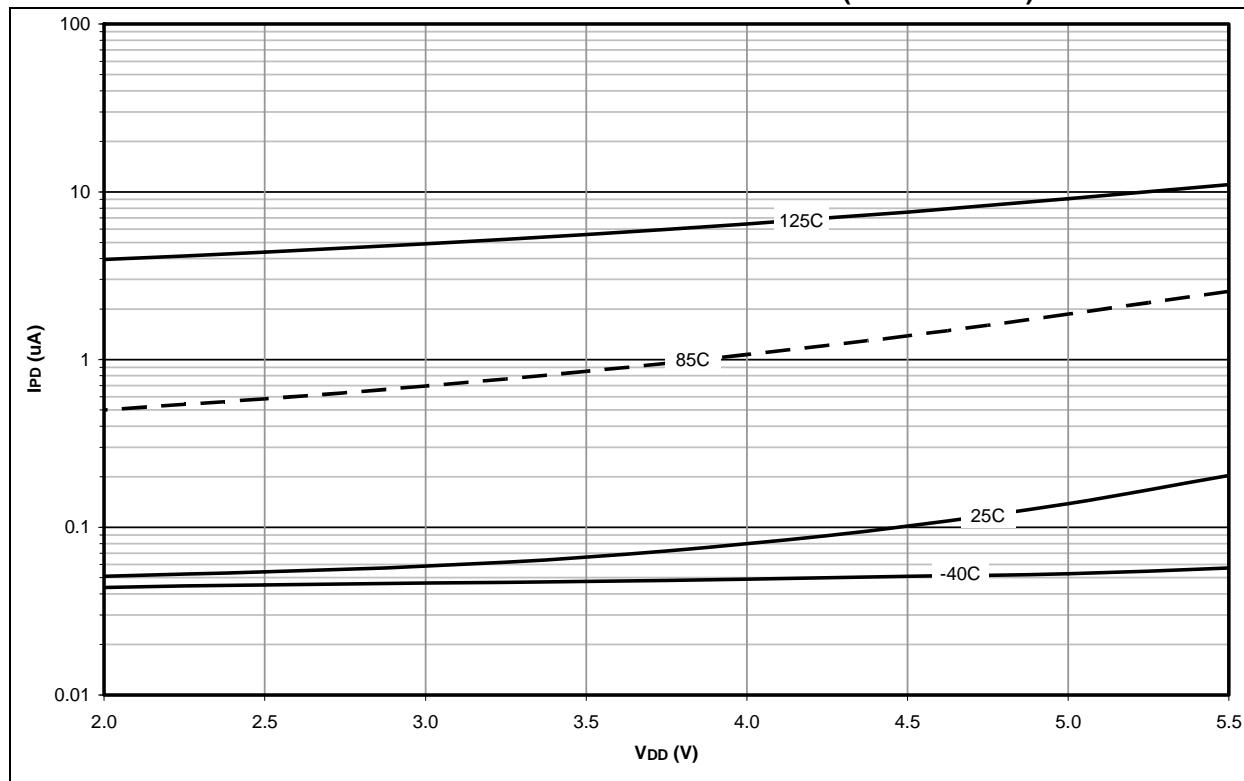
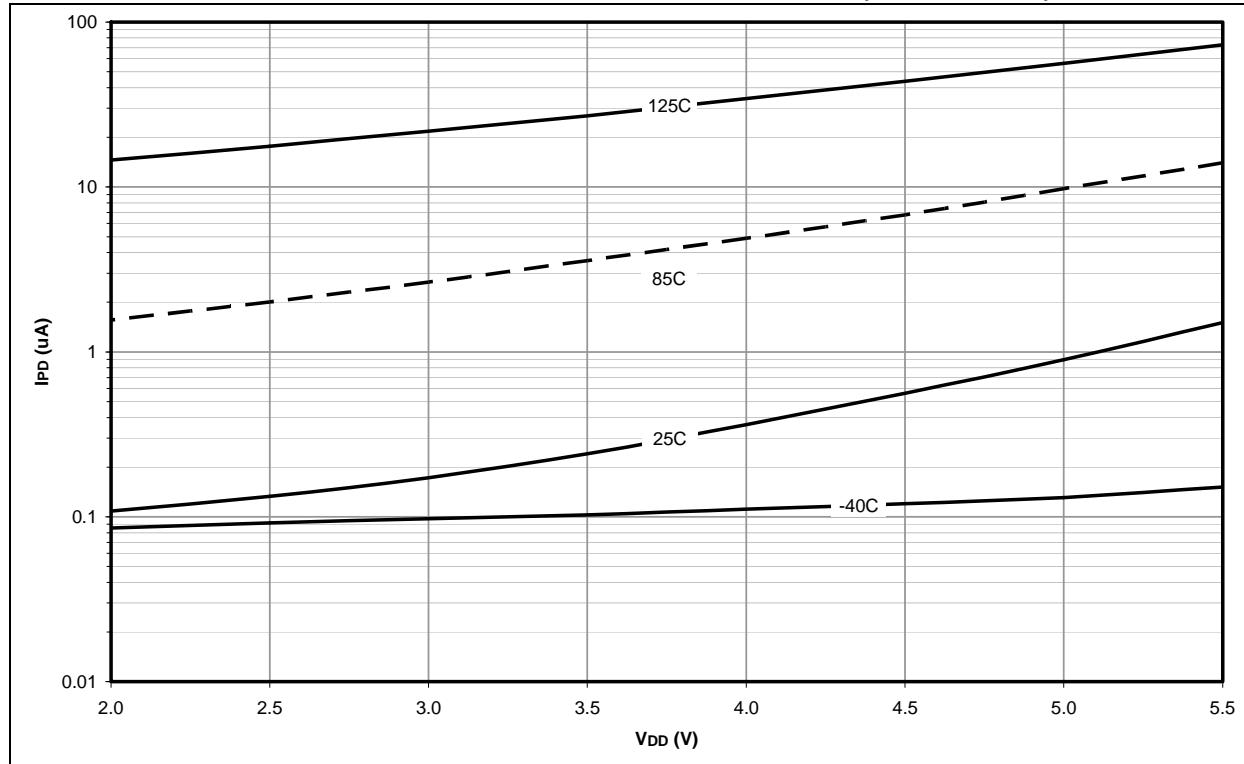
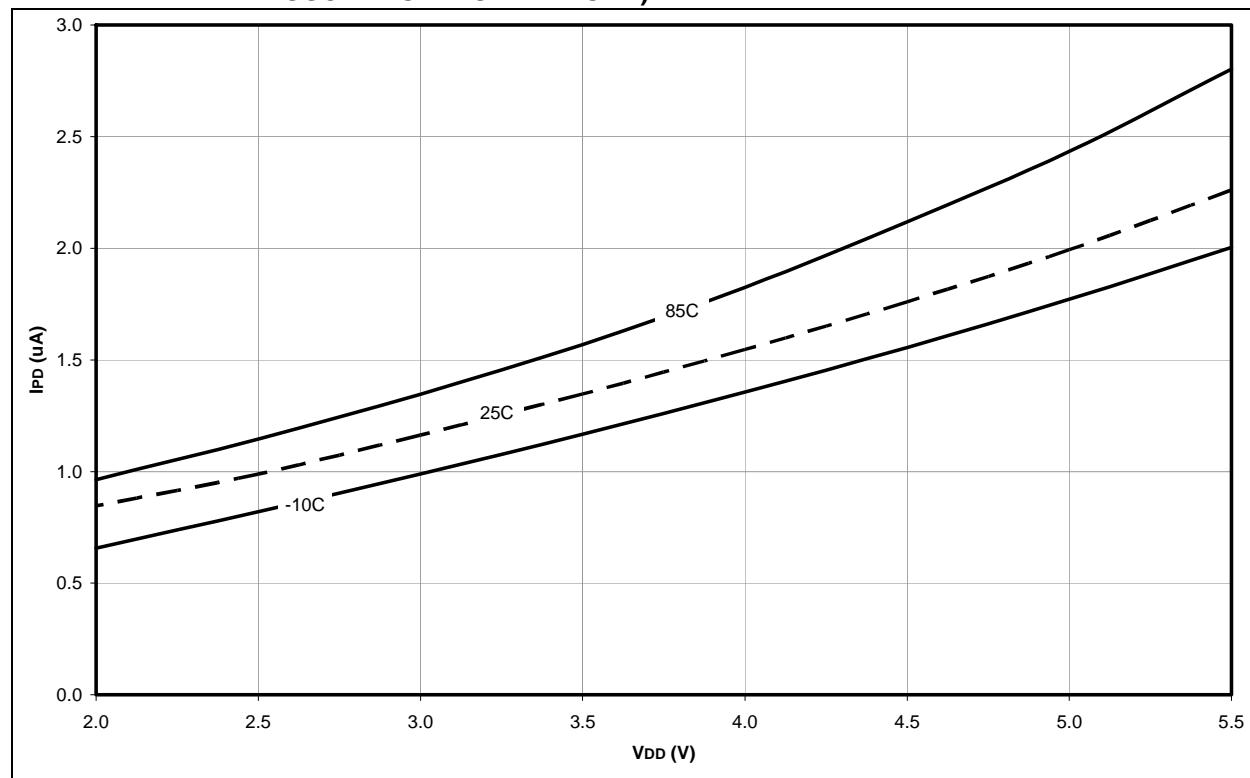


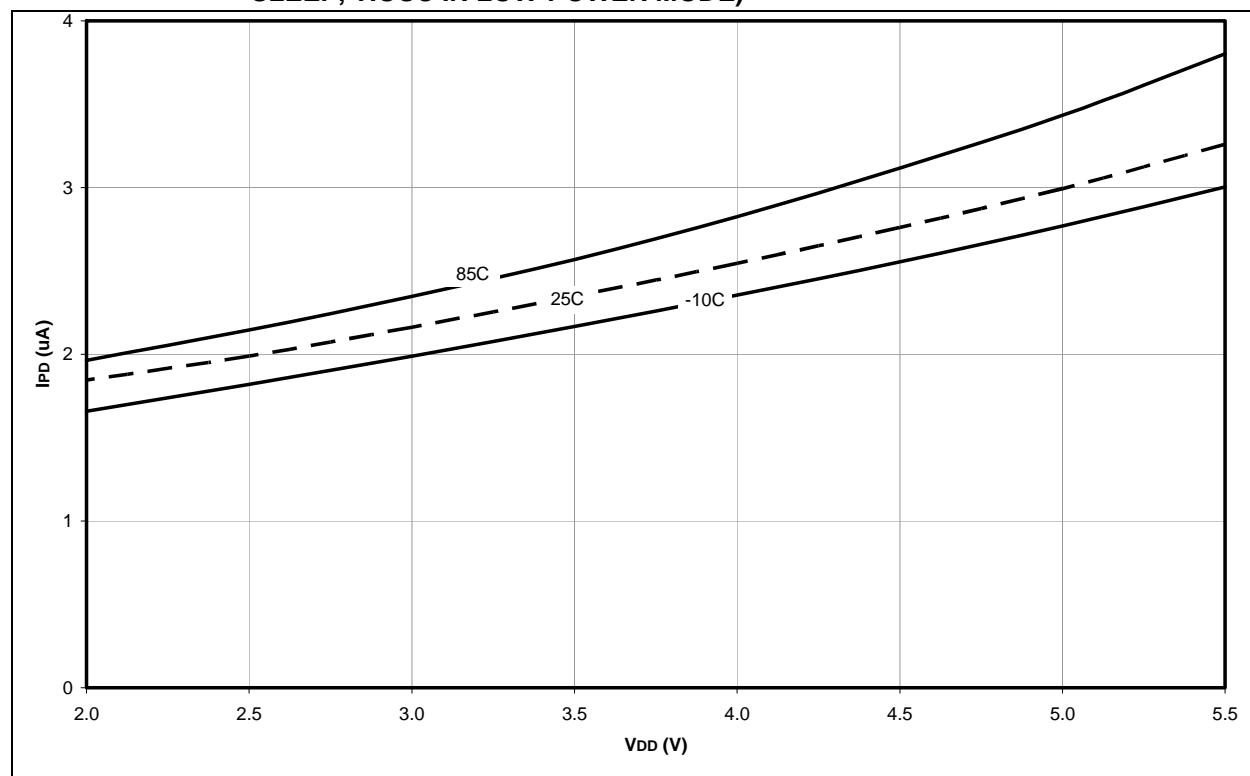
FIGURE 27-3: MAXIMUM IPD VS. VDD ACROSS TEMPERATURE (SLEEP MODE)



**FIGURE 27-4: TYPICAL T1OSC DELTA CURRENT vs. V<sub>DD</sub> ACROSS TEMP. (DEVICE IN SLEEP, T1OSC IN LOW-POWER MODE)**

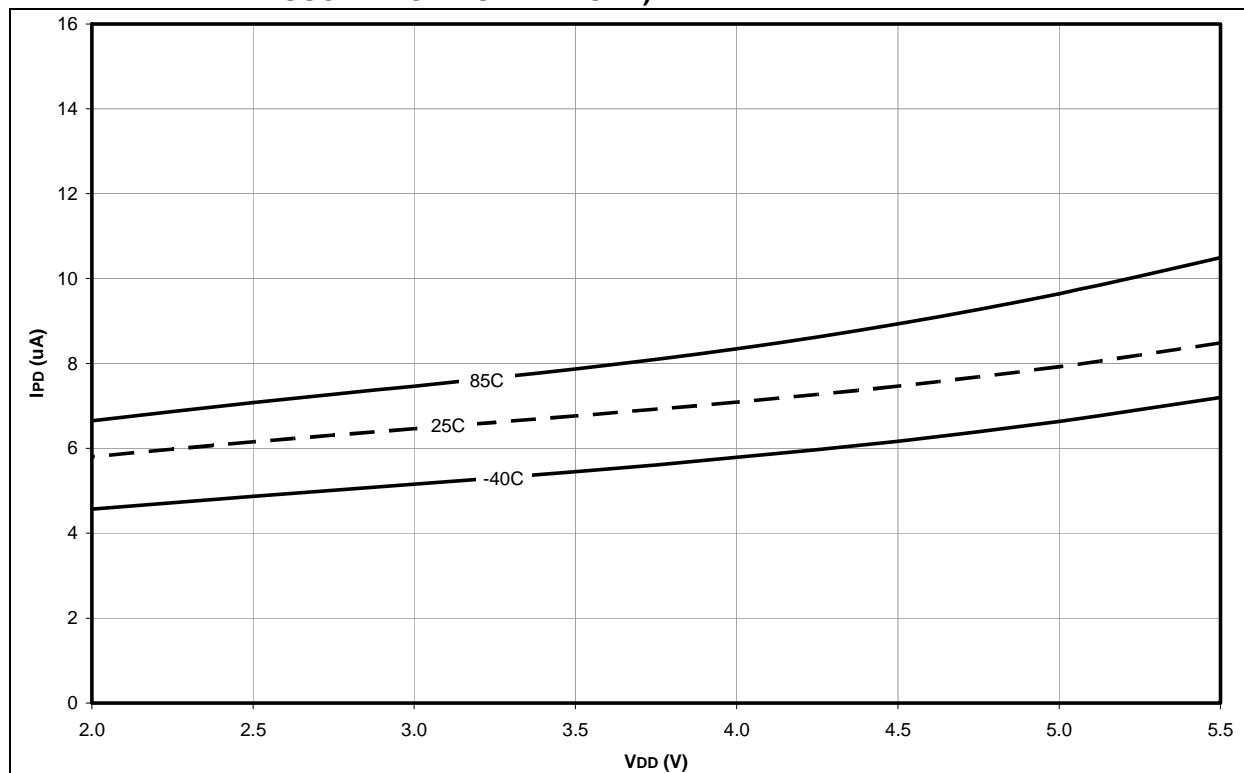


**FIGURE 27-5: MAXIMUM T1OSC DELTA CURRENT vs. V<sub>DD</sub> ACROSS TEMP. (DEVICE IN SLEEP, TIOSC IN LOW-POWER MODE)**

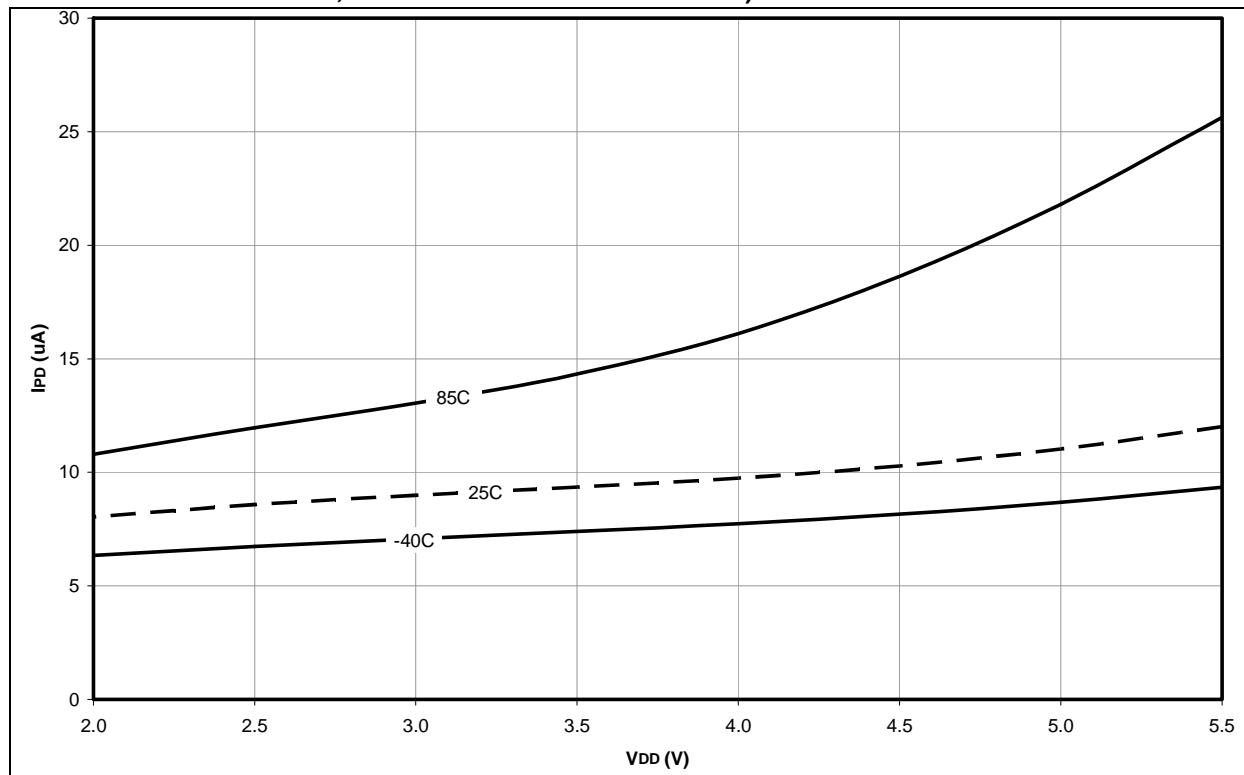


# PIC18F2525/2620/4525/4620

**FIGURE 27-6: TYPICAL T1OSC DELTA CURRENT vs. V<sub>DD</sub> ACROSS TEMP. (DEVICE IN SLEEP, T1OSC IN HIGH-POWER MODE)**

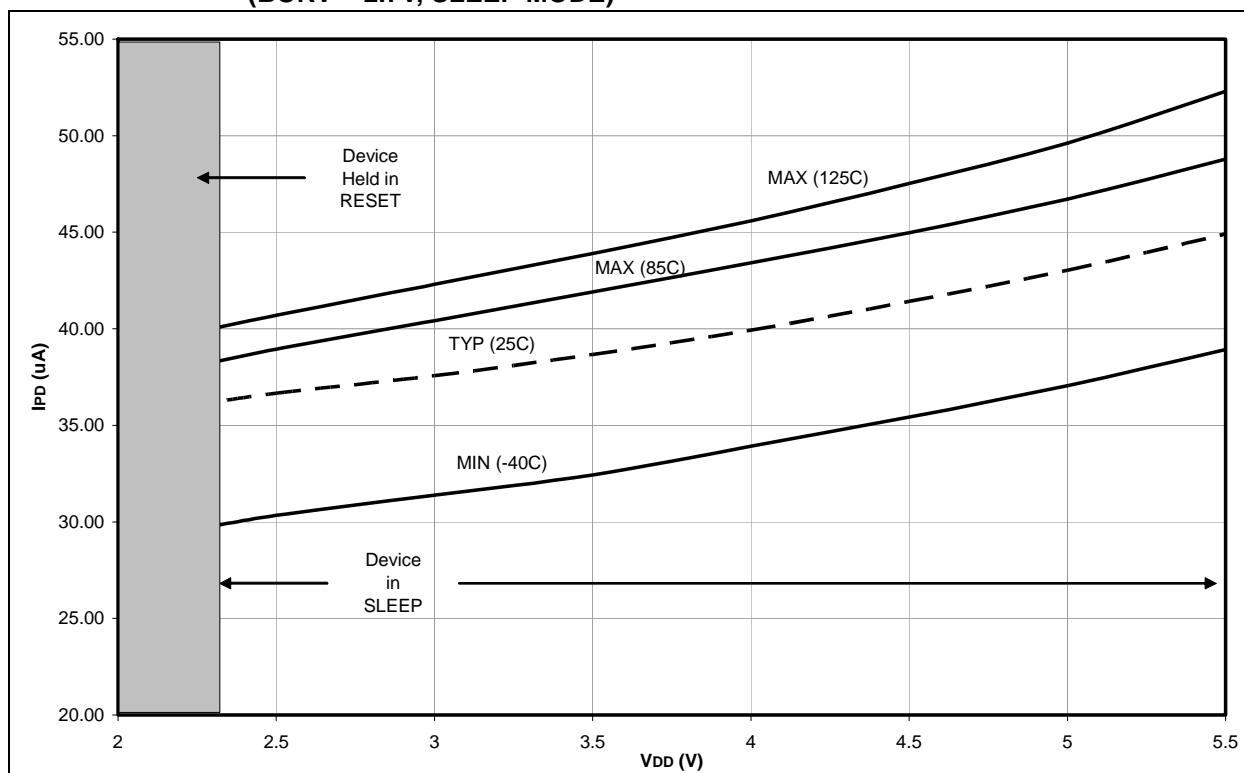


**FIGURE 27-7: MAXIMUM T1OSC DELTA CURRENT vs. V<sub>DD</sub> ACROSS TEMP. (DEVICE IN SLEEP, T1OSC IN HIGH-POWER MODE)**



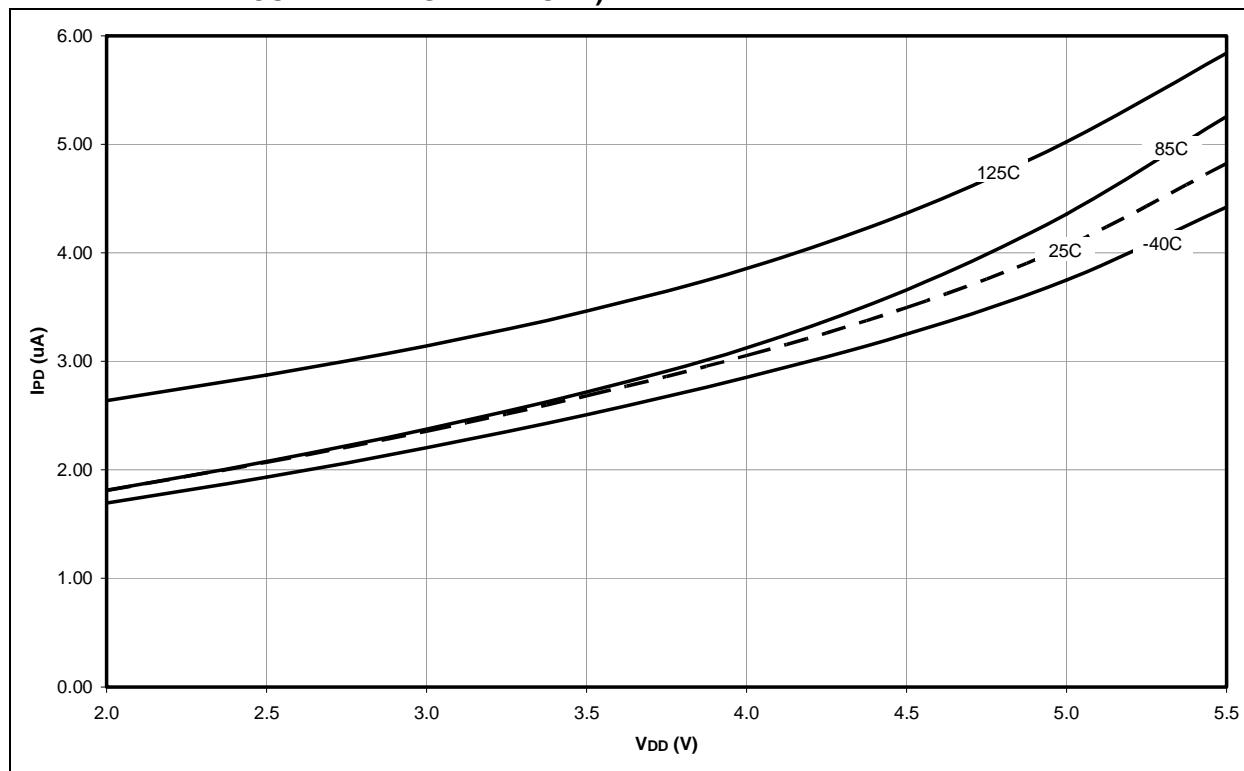
# PIC18F2525/2620/4525/4620

**FIGURE 27-8: TYPICAL BOR DELTA CURRENT vs. V<sub>DD</sub> ACROSS TEMP.  
(BORV = 2.7V, SLEEP MODE)**

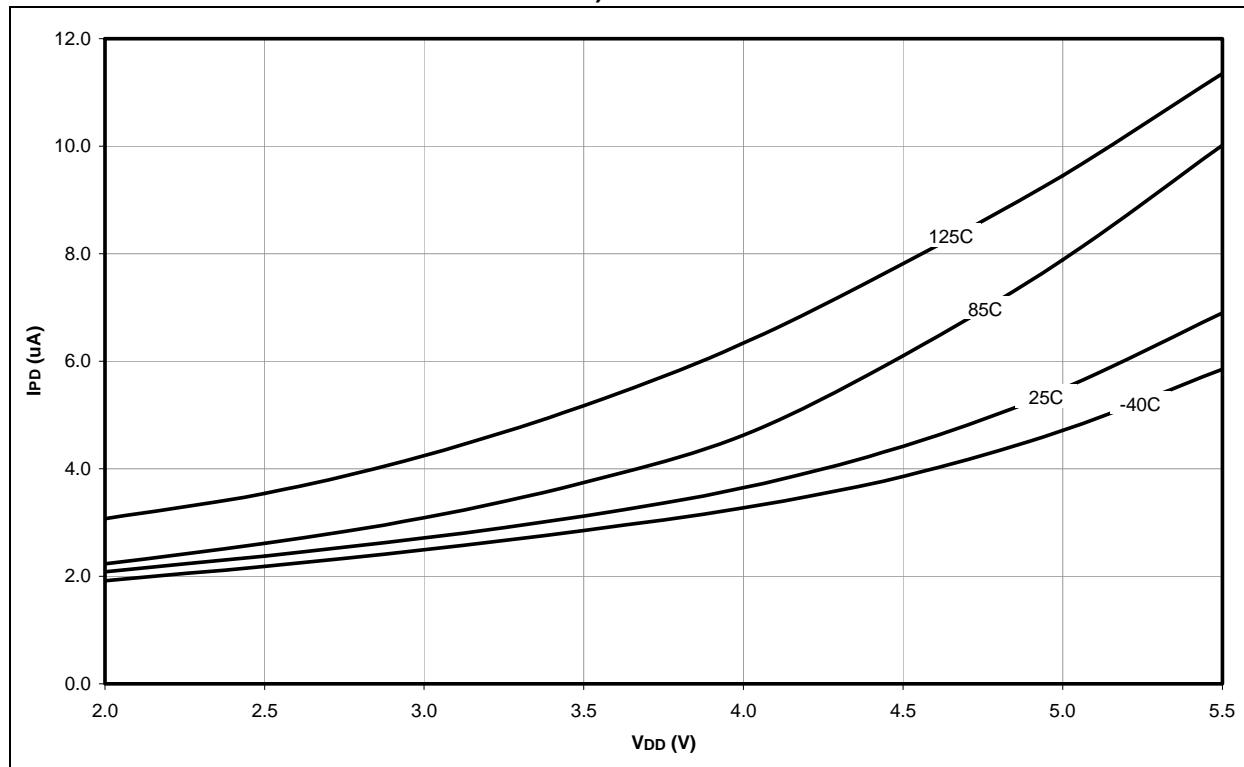


# PIC18F2525/2620/4525/4620

**FIGURE 27-9: TYPICAL WDT CURRENT vs. V<sub>DD</sub> ACROSS TEMPERATURE (WDT DELTA CURRENT IN SLEEP MODE)**



**FIGURE 27-10: MAXIMUM WDT CURRENT vs. V<sub>DD</sub> ACROSS TEMPERATURE (WDT DELTA CURRENT IN SLEEP MODE)**



# PIC18F2525/2620/4525/4620

FIGURE 27-11: TYPICAL IDD ACROSS V<sub>DD</sub> (RC\_RUN MODE, 25°C)

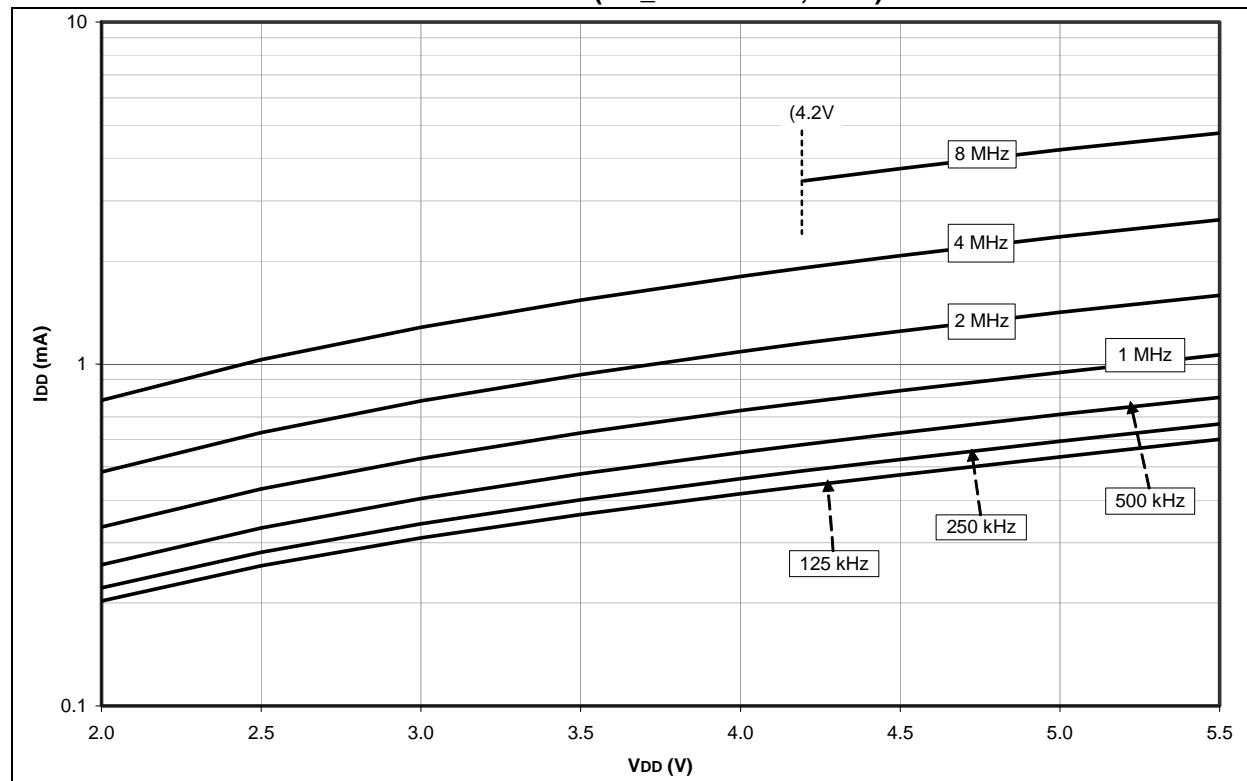
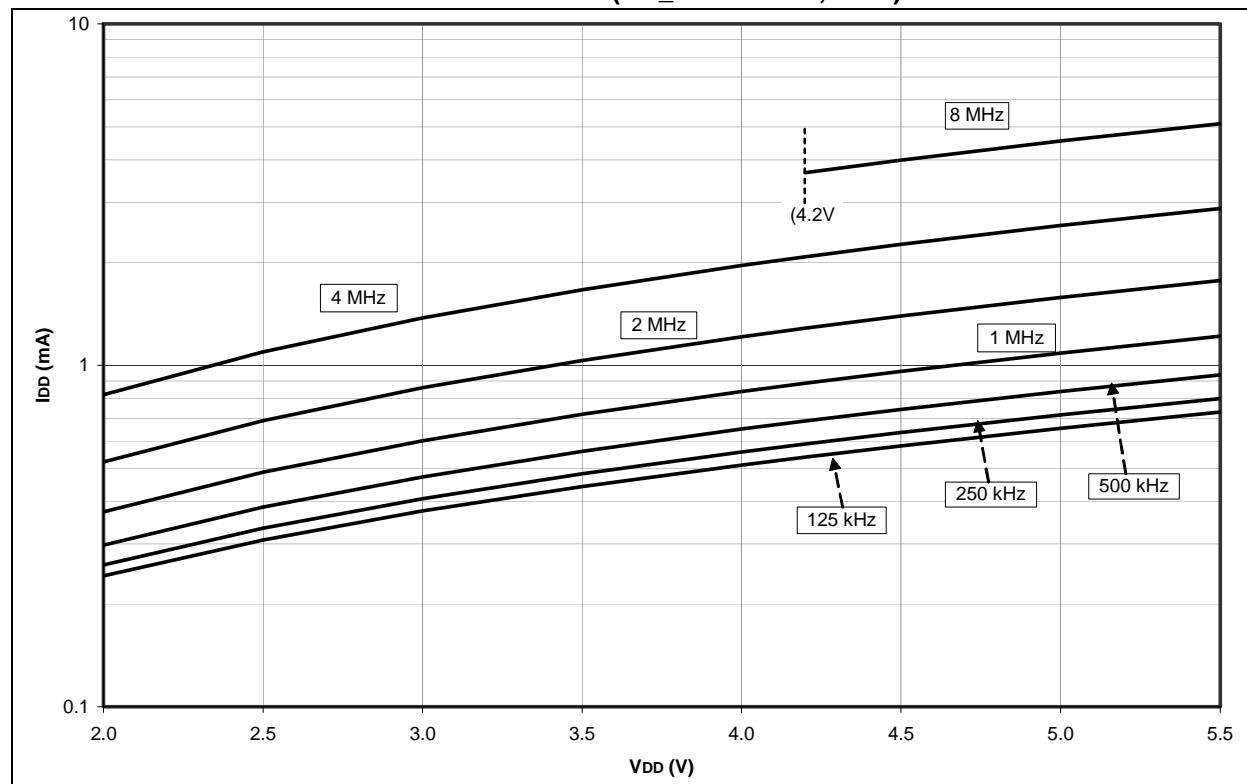


FIGURE 27-12: MAXIMUM IDD ACROSS V<sub>DD</sub> (RC\_RUN MODE, 85°C)



# PIC18F2525/2620/4525/4620

FIGURE 27-13: TYPICAL AND MAXIMUM IDD ACROSS V<sub>DD</sub> (RC\_RUN MODE, 31 kHz)

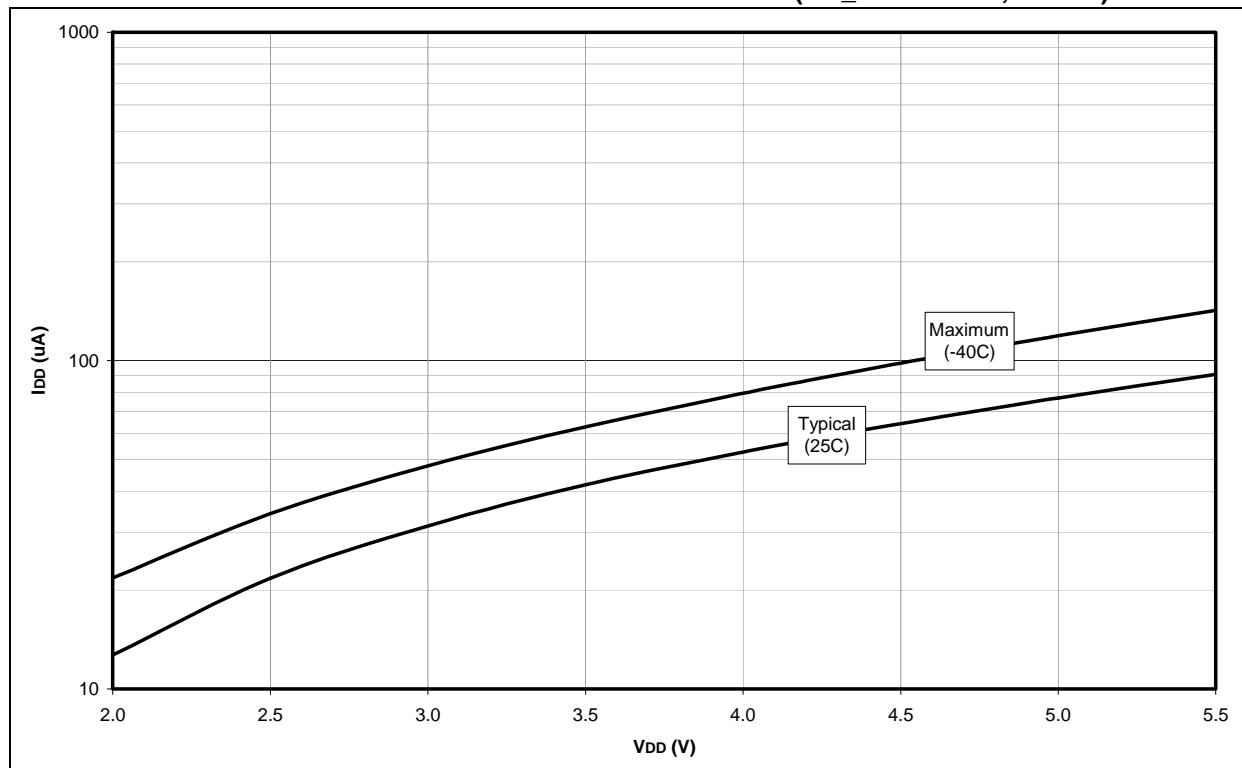
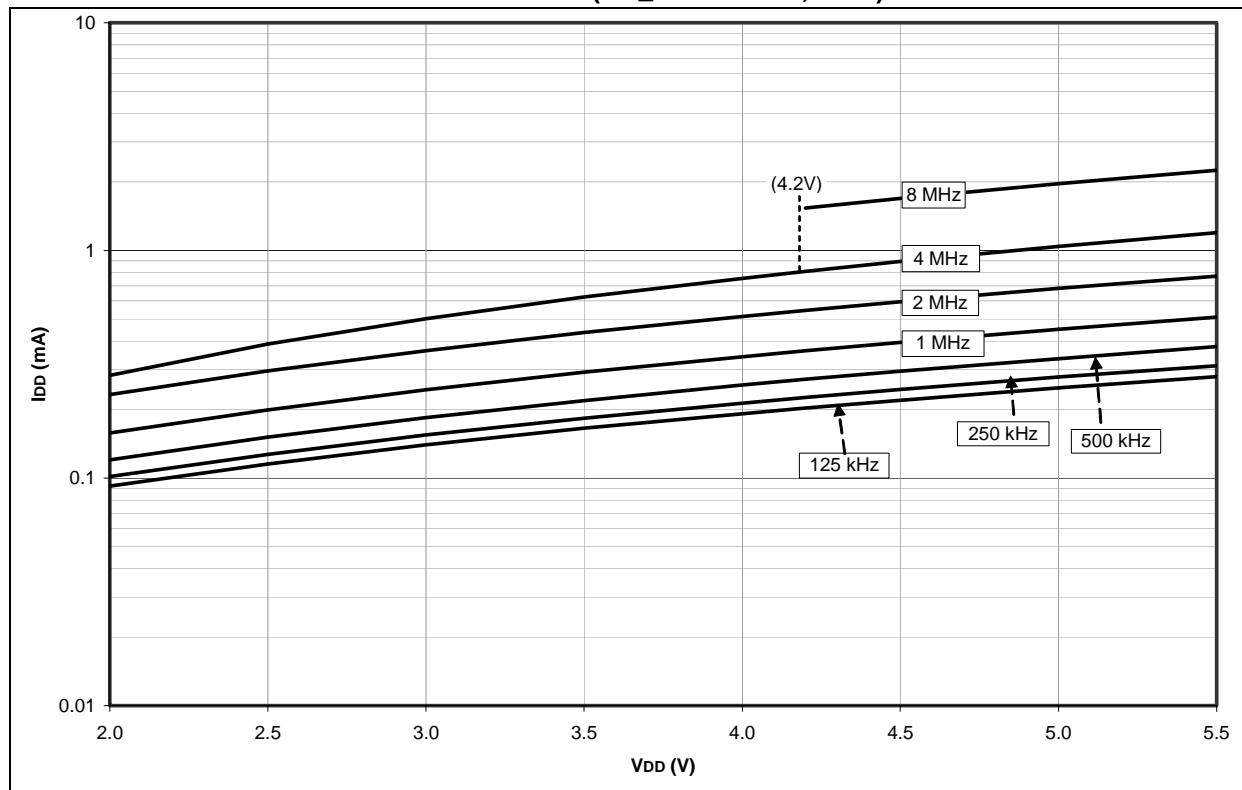


FIGURE 27-14: TYPICAL IDD ACROSS V<sub>DD</sub> (RC\_IDLE MODE, 25°C)



# PIC18F2525/2620/4525/4620

FIGURE 27-15: MAXIMUM IDD ACROSS V<sub>DD</sub> (RC\_IDLE MODE, -40°C-85°C)

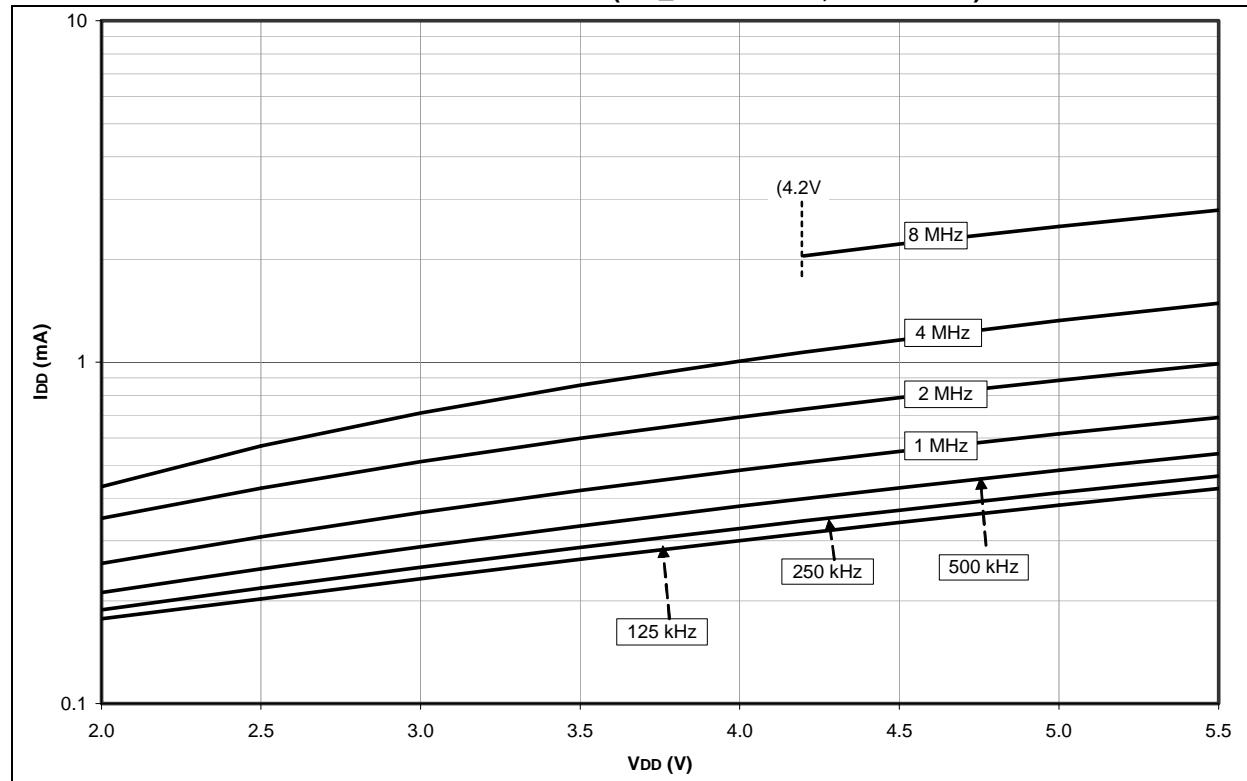
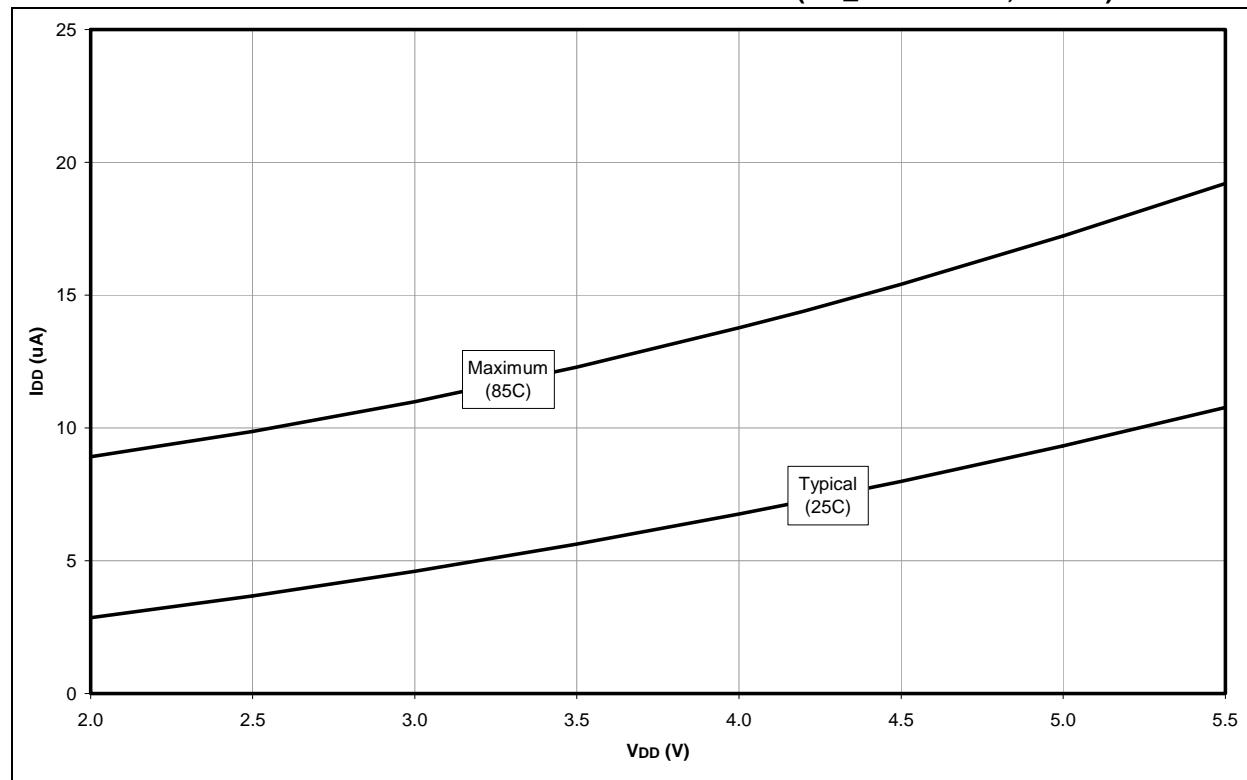
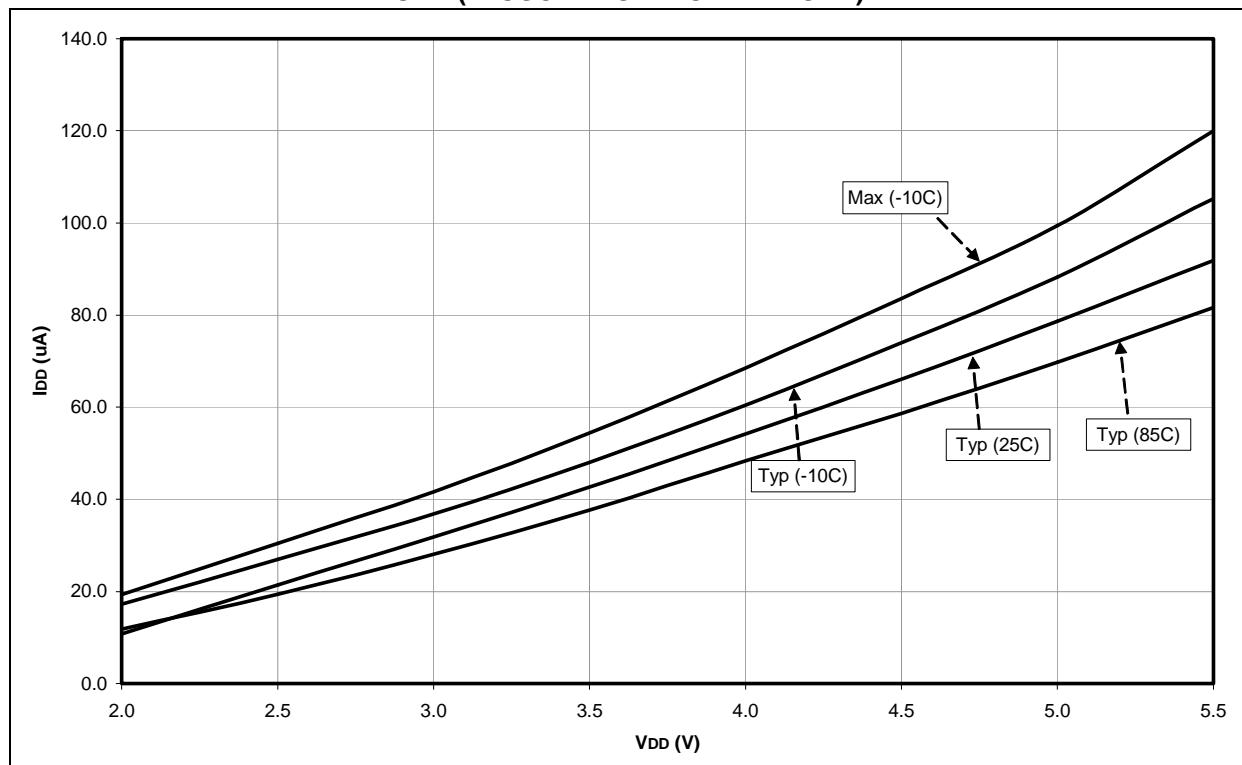


FIGURE 27-16: TYPICAL AND MAXIMUM IDD ACROSS V<sub>DD</sub> (RC\_IDLE MODE, 31 kHz)

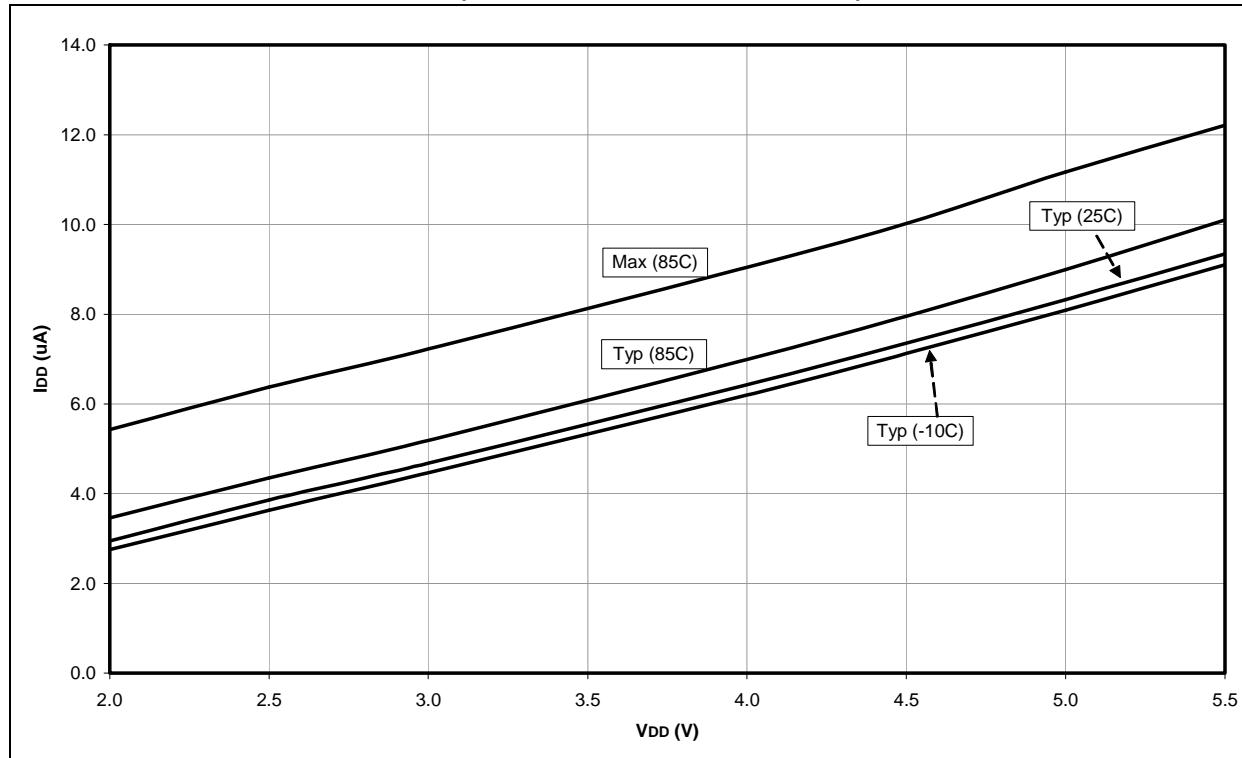


# PIC18F2525/2620/4525/4620

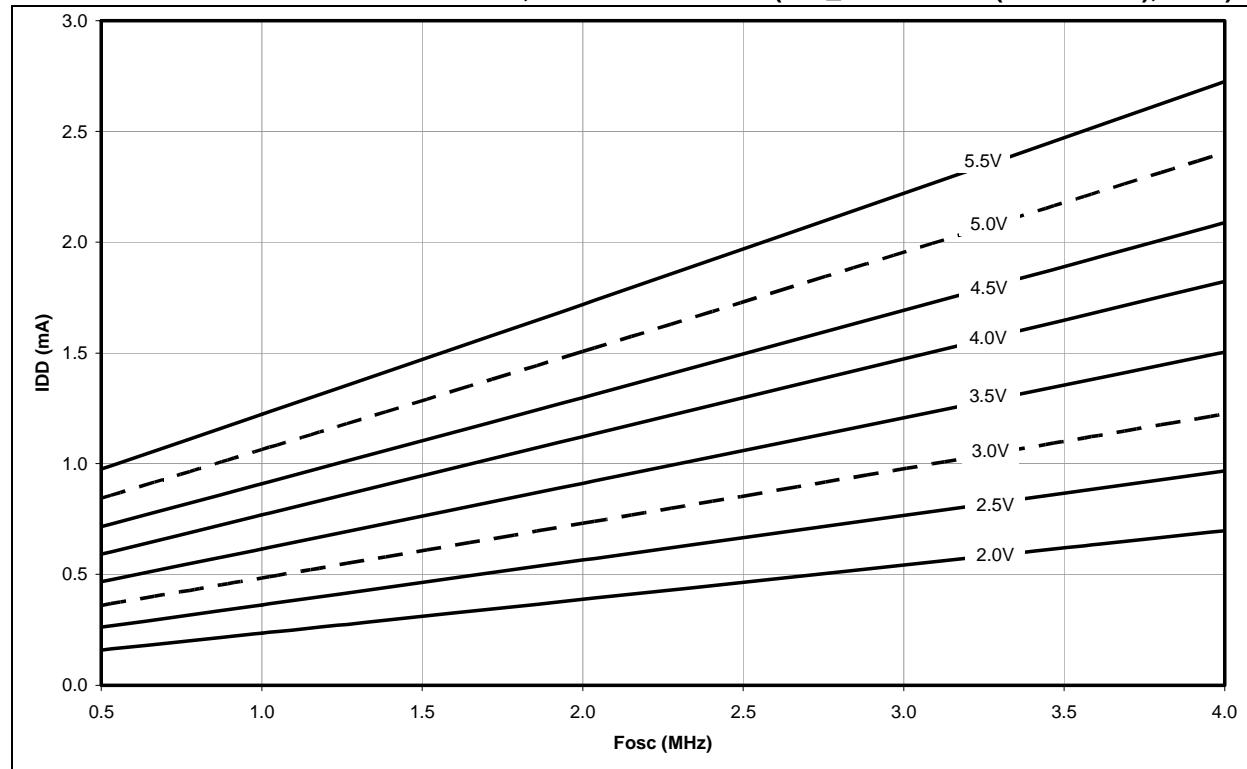
**FIGURE 27-17: TYPICAL AND MAXIMUM SEC\_RUN CURRENT vs. V<sub>DD</sub> ACROSS TEMPERATURE (T1OSC IN LOW-POWER MODE)**



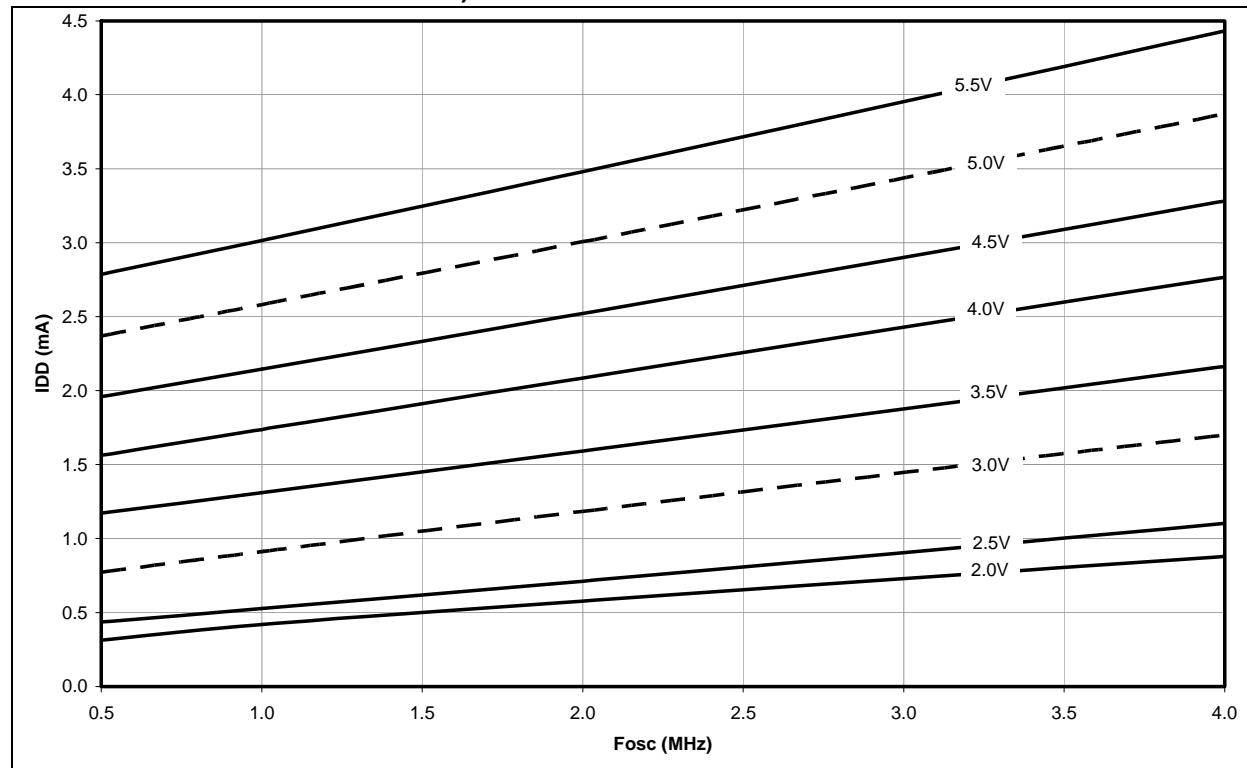
**FIGURE 27-18: TYPICAL AND MAXIMUM SEC\_IDLE CURRENT vs. V<sub>DD</sub> ACROSS TEMPERATURE (T1OSC IN LOW-POWER MODE)**



**FIGURE 27-19: TYPICAL IDD vs. Fosc, 500 kHz TO 4 MHz (PRI\_RUN MODE (EC CLOCK), 25°C)**



**FIGURE 27-20: MAXIMUM IDD vs. Fosc, 500 kHz TO 4 MHz (PRI\_RUN MODE (EC CLOCK), -40°C TO +125°C)**



# PIC18F2525/2620/4525/4620

FIGURE 27-21: TYPICAL  $I_{DD}$  vs.  $F_{osc}$ , 4 MHz TO 40 MHz (PRI\_RUN MODE (EC CLOCK), 25°C)

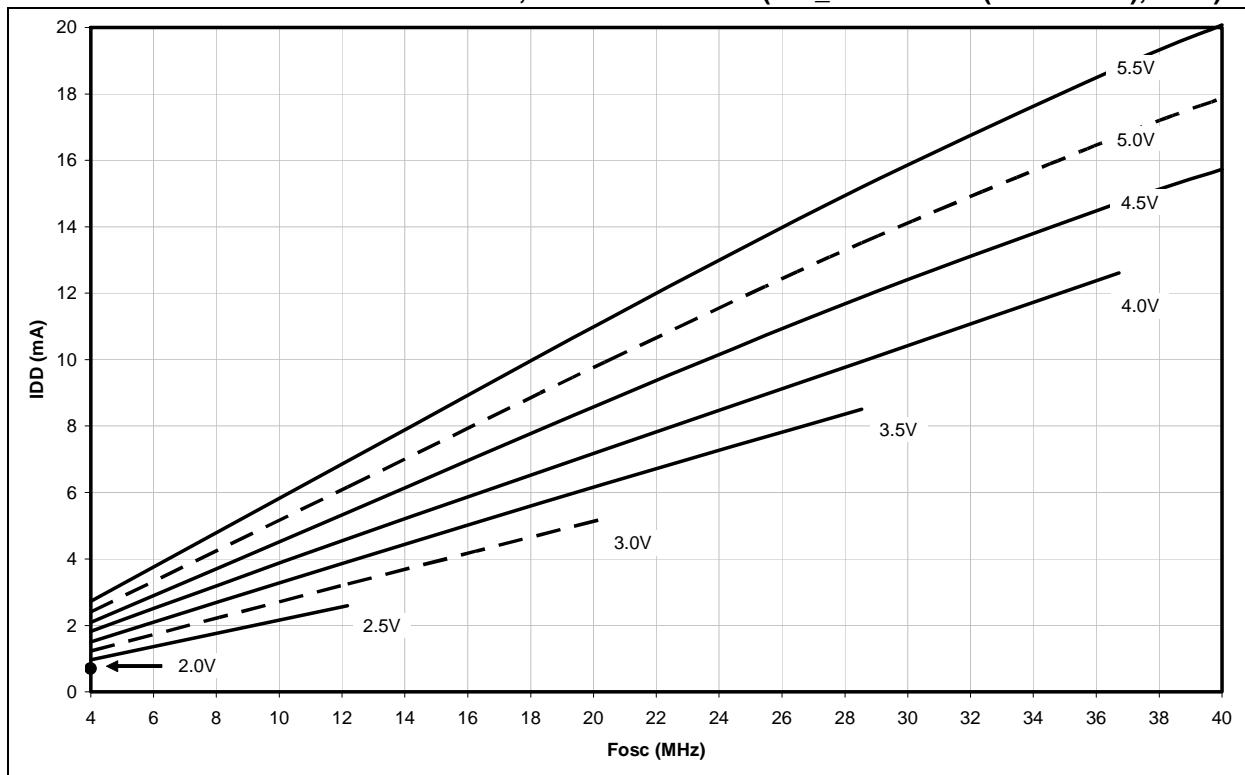


FIGURE 27-22: MAXIMUM  $I_{DD}$  vs.  $F_{osc}$ , 4 MHz TO 40 MHz (PRI\_RUN MODE (EC CLOCK), -40°C TO +125°C)

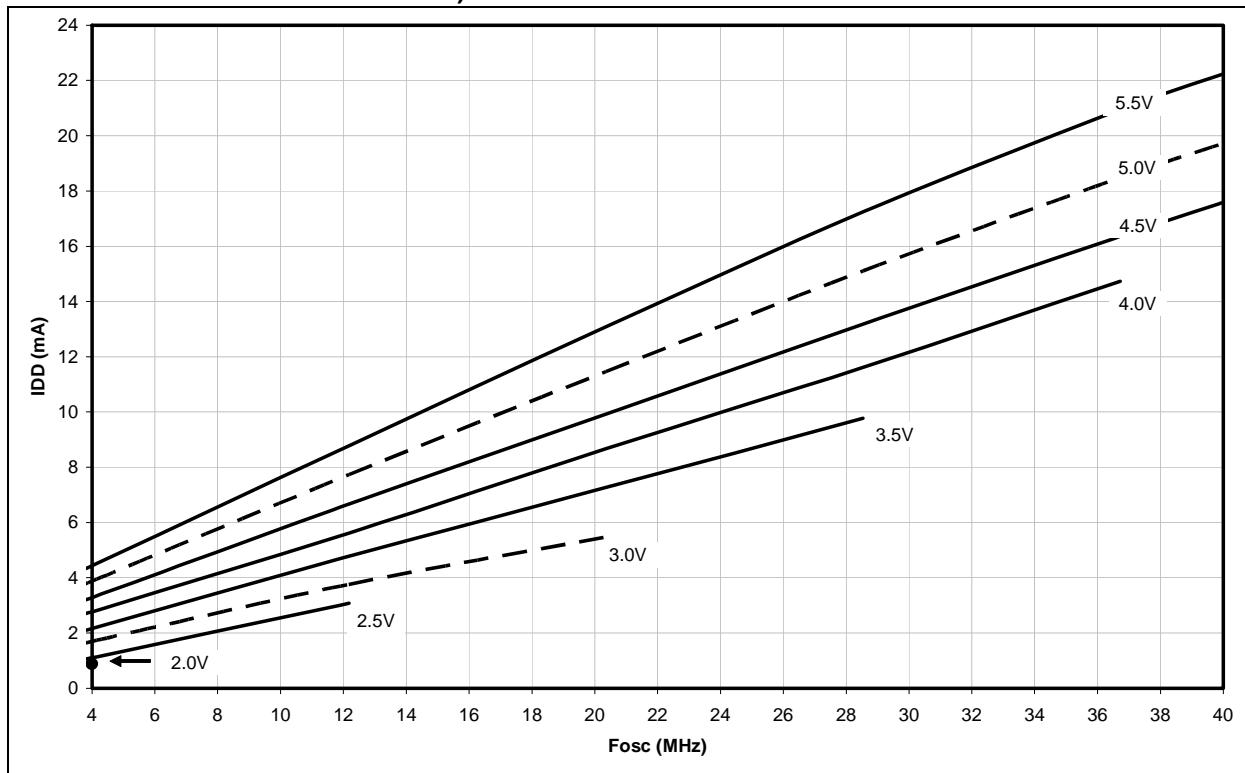


FIGURE 27-23: TYPICAL  $I_{DD}$  vs.  $F_{osc}$ , HS/PLL (PRI\_RUN MODE, 25°C)

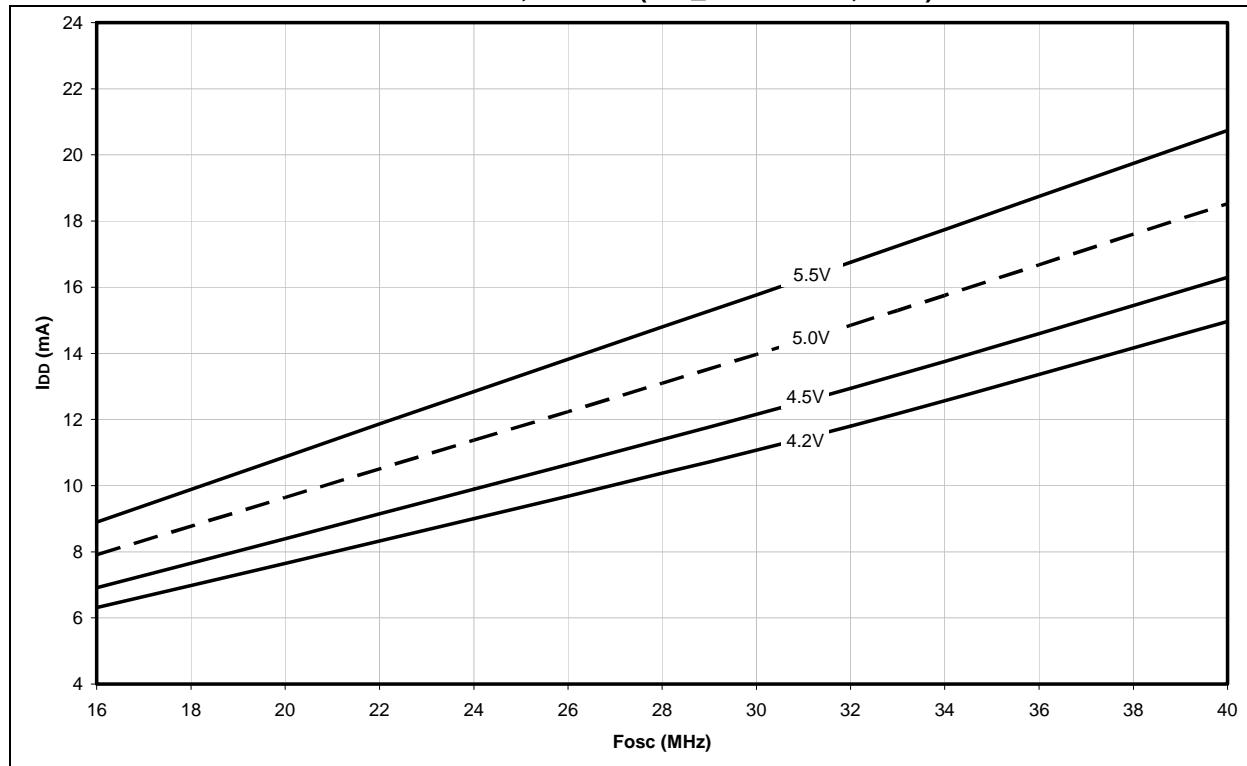
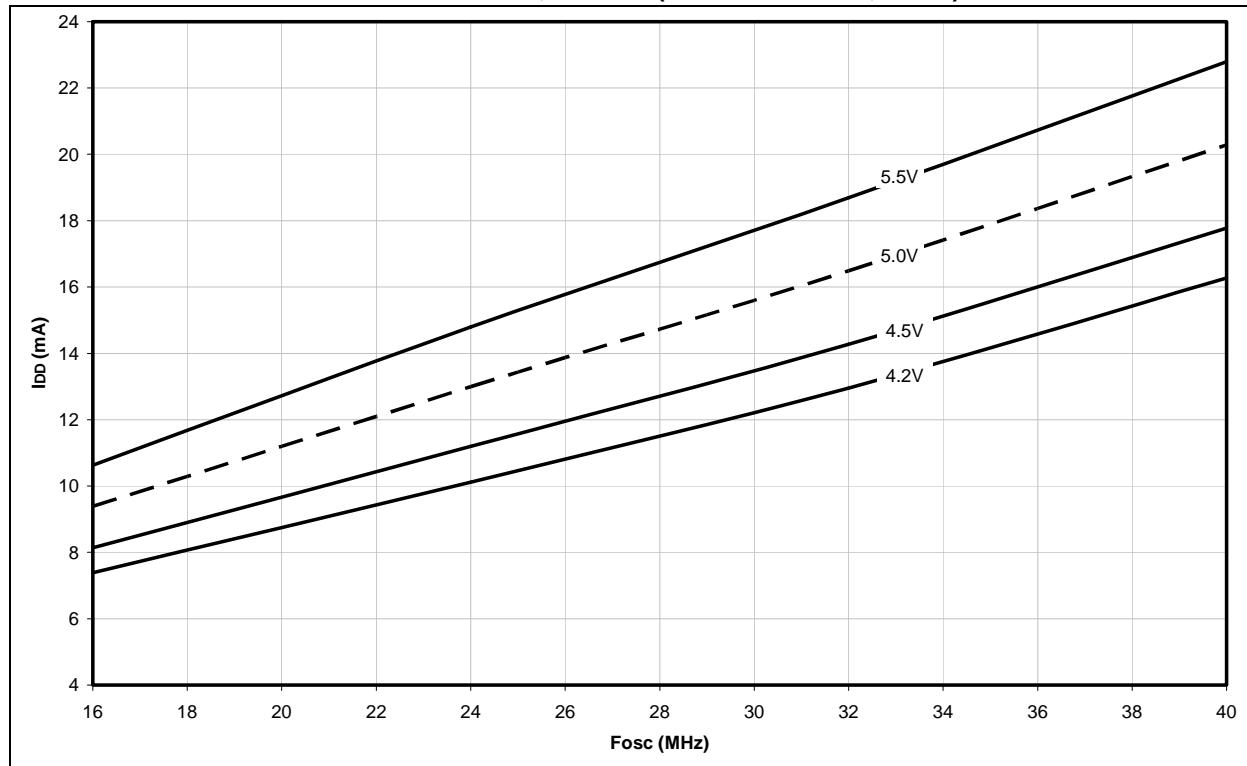


FIGURE 27-24: MAXIMUM  $I_{DD}$  vs.  $F_{osc}$ , HS/PLL (PRI\_RUN MODE, -40°C)



# PIC18F2525/2620/4525/4620

FIGURE 27-25: TYPICAL  $I_{DD}$  vs.  $F_{osc}$ , 500 kHz TO 4 MHz (PRI\_IDLE MODE, 25°C)

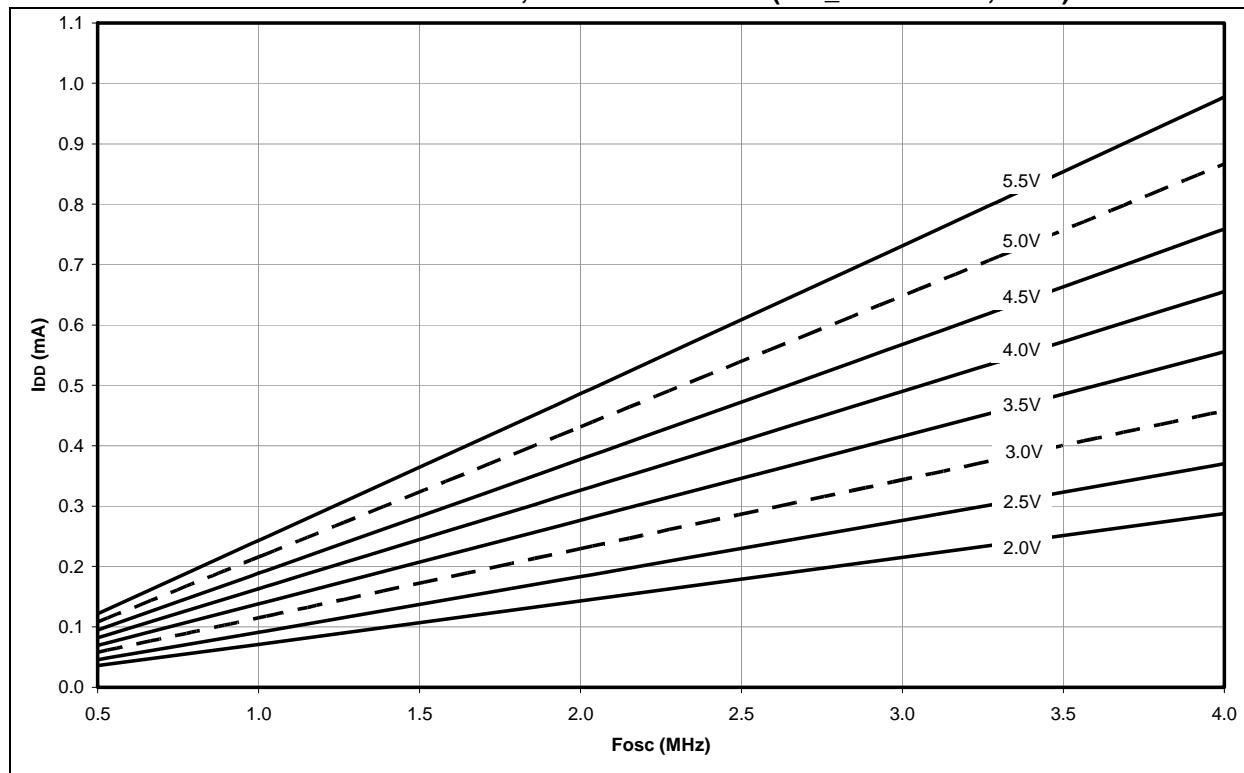
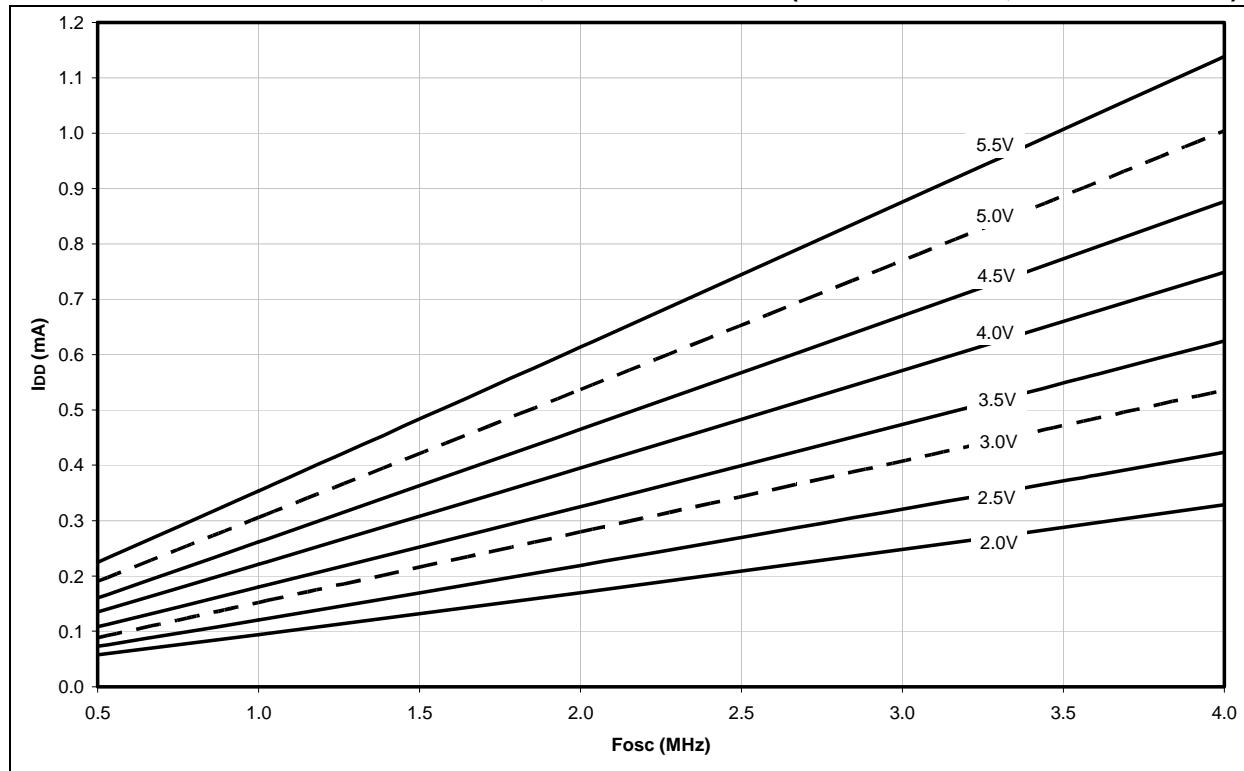


FIGURE 27-26: MAXIMUM  $I_{DD}$  vs.  $F_{osc}$ , 500 kHz TO 4 MHz (PRI\_IDLE MODE, -40°C TO +125°C)



# PIC18F2525/2620/4525/4620

FIGURE 27-27: TYPICAL IDD vs. Fosc, 4 MHz TO 40 MHz (PRI\_IDLE MODE, 25°C)

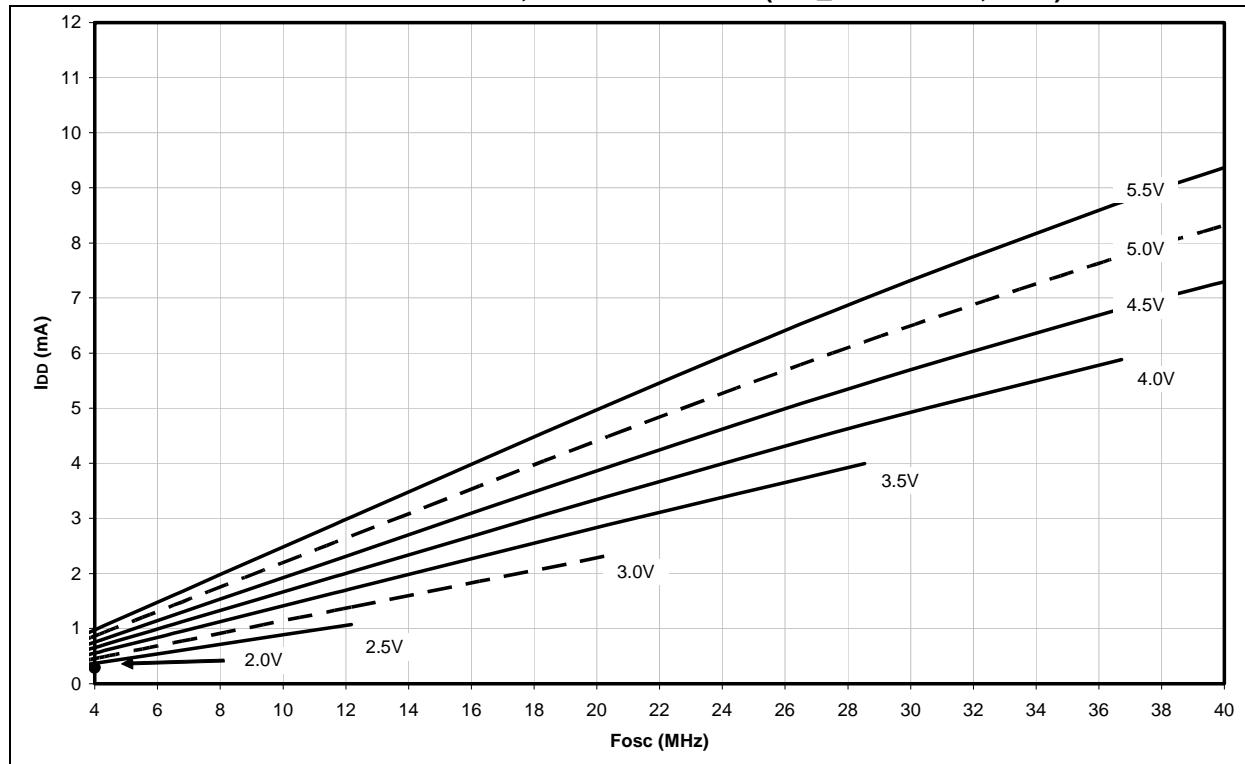
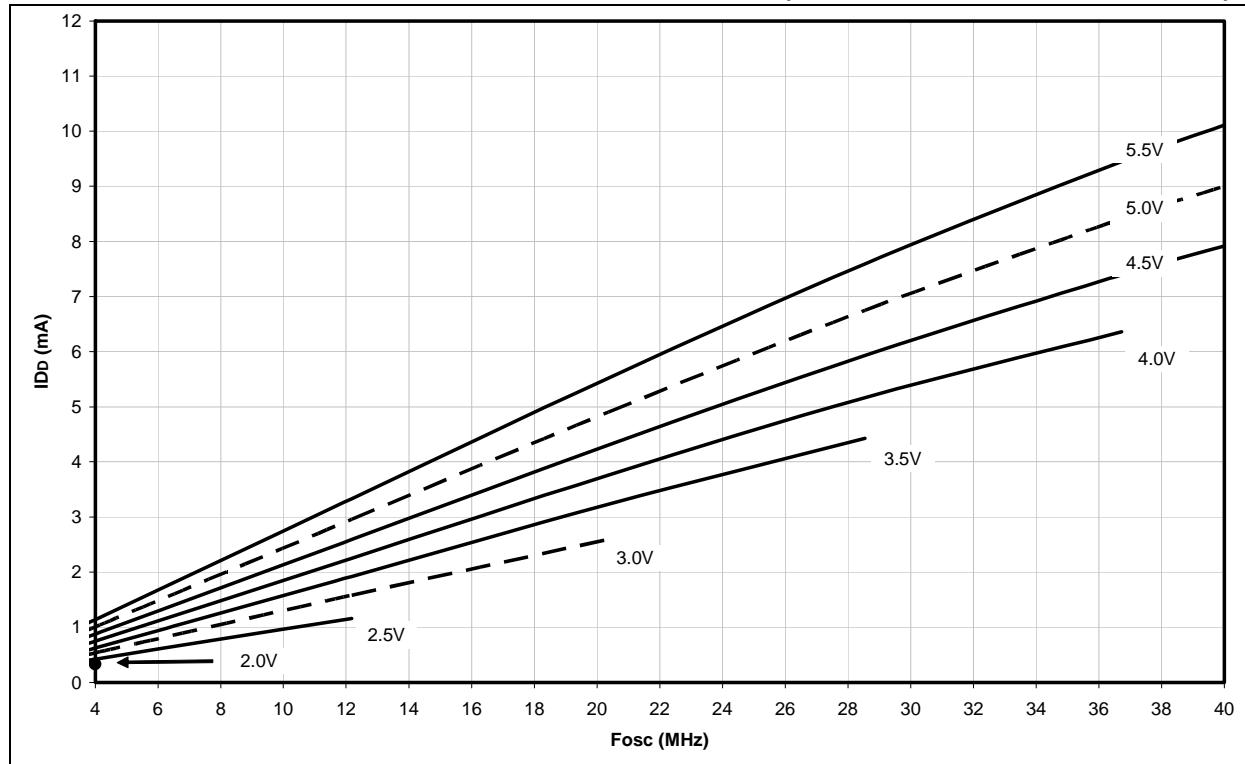


FIGURE 27-28: MAXIMUM IDD vs. Fosc, 4 MHz TO 40 MHz (PRI\_IDLE MODE, -40°C TO +125°C)



# PIC18F2525/2620/4525/4620

FIGURE 27-29: TYPICAL IDD vs. Fosc, HS/PLL (PRI\_IDLE MODE, 25°C)

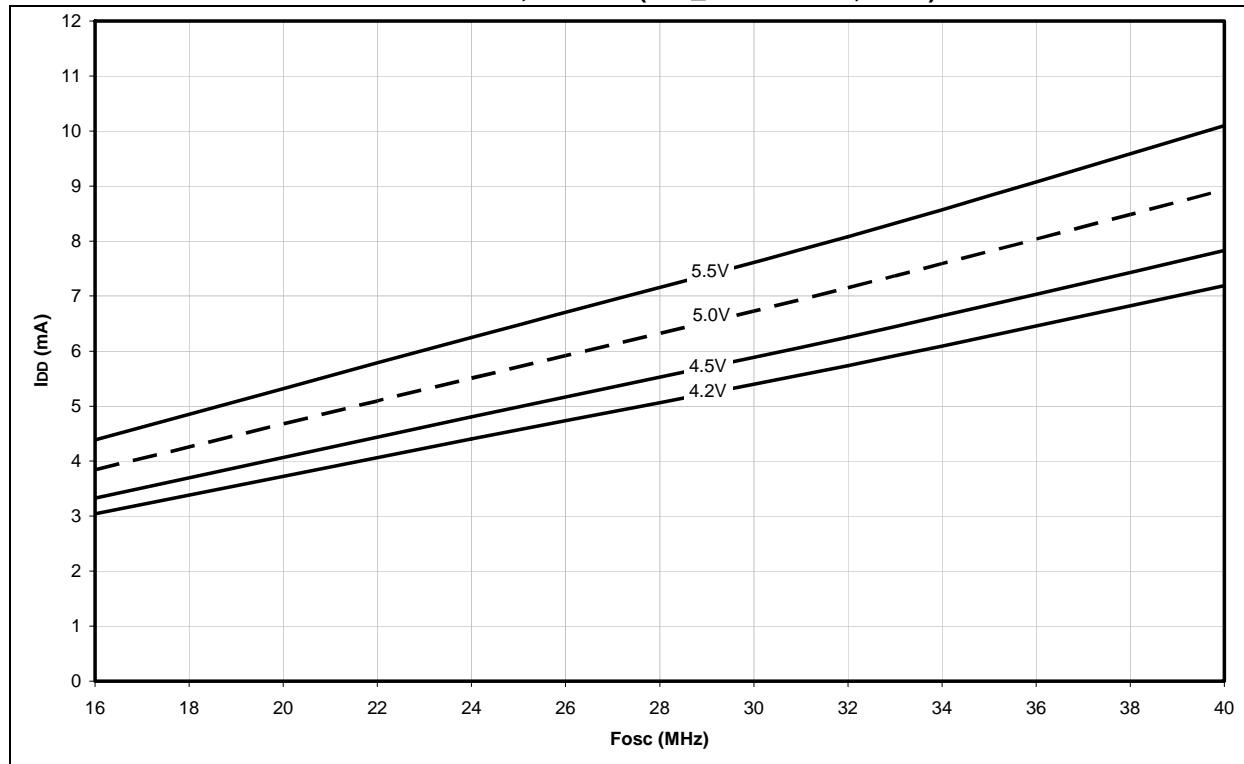


FIGURE 27-30: MAXIMUM IDD vs. Fosc, HS/PLL (PRI\_IDLE MODE, -40°C)

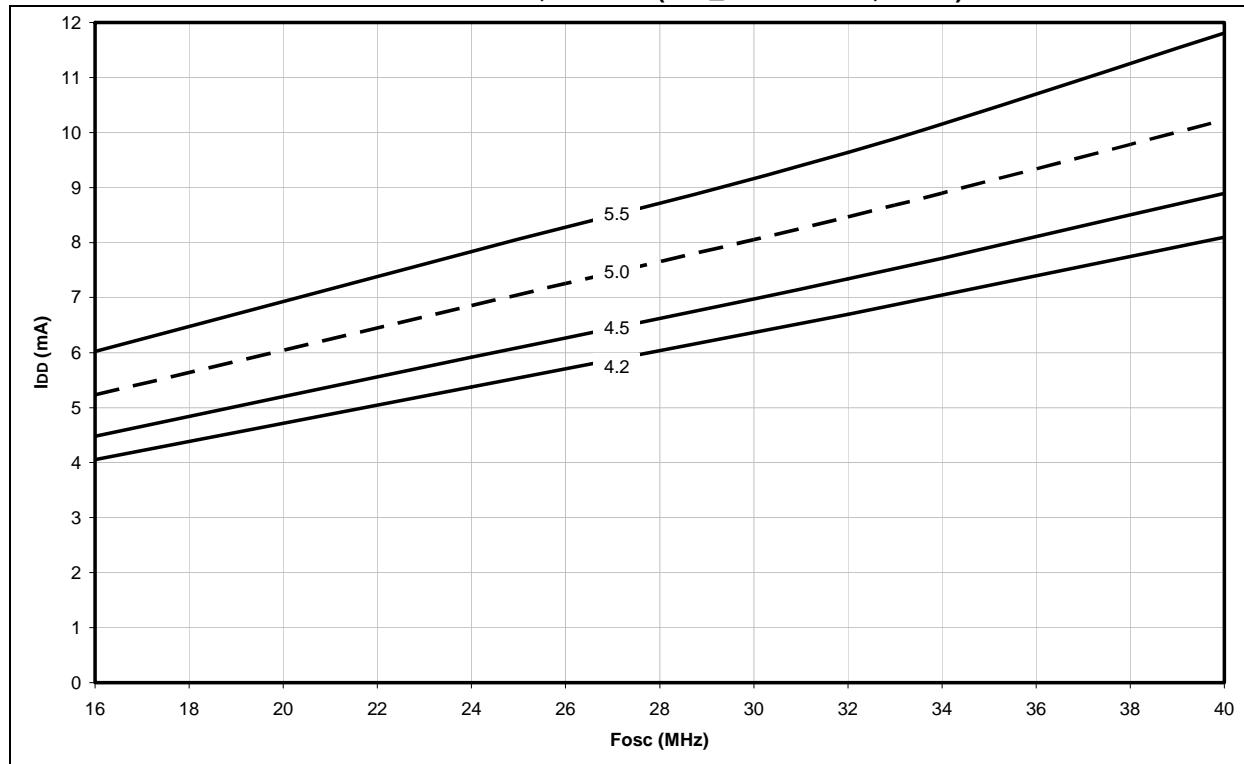


FIGURE 27-31:  $V_{IN}$  (ST) vs.  $V_{DD}$ , 25°C (-40°C TO +125°C)

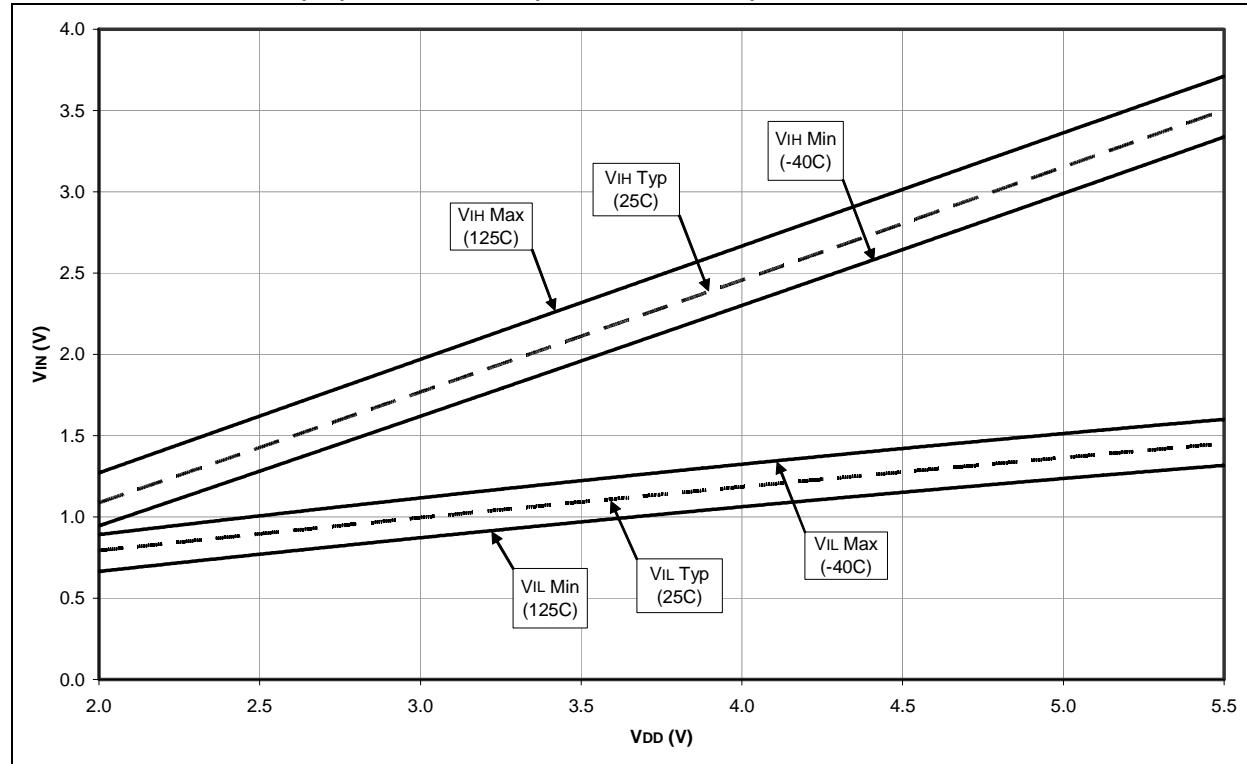
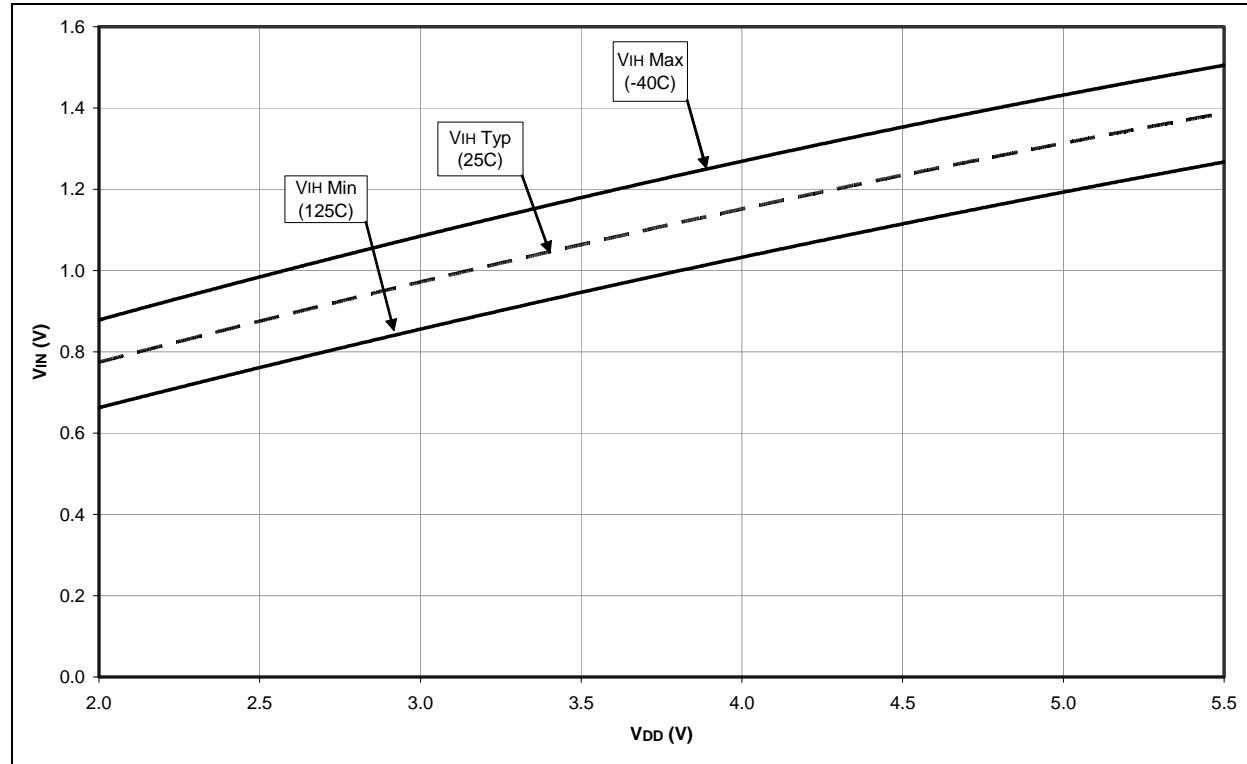


FIGURE 27-32:  $V_{IN}$  (TTL) vs.  $V_{DD}$ , 25°C (-40°C TO +125°C)



# PIC18F2525/2620/4525/4620

FIGURE 27-33: VOL vs. IOL (VDD = 3.0V, -40°C TO +85°C)

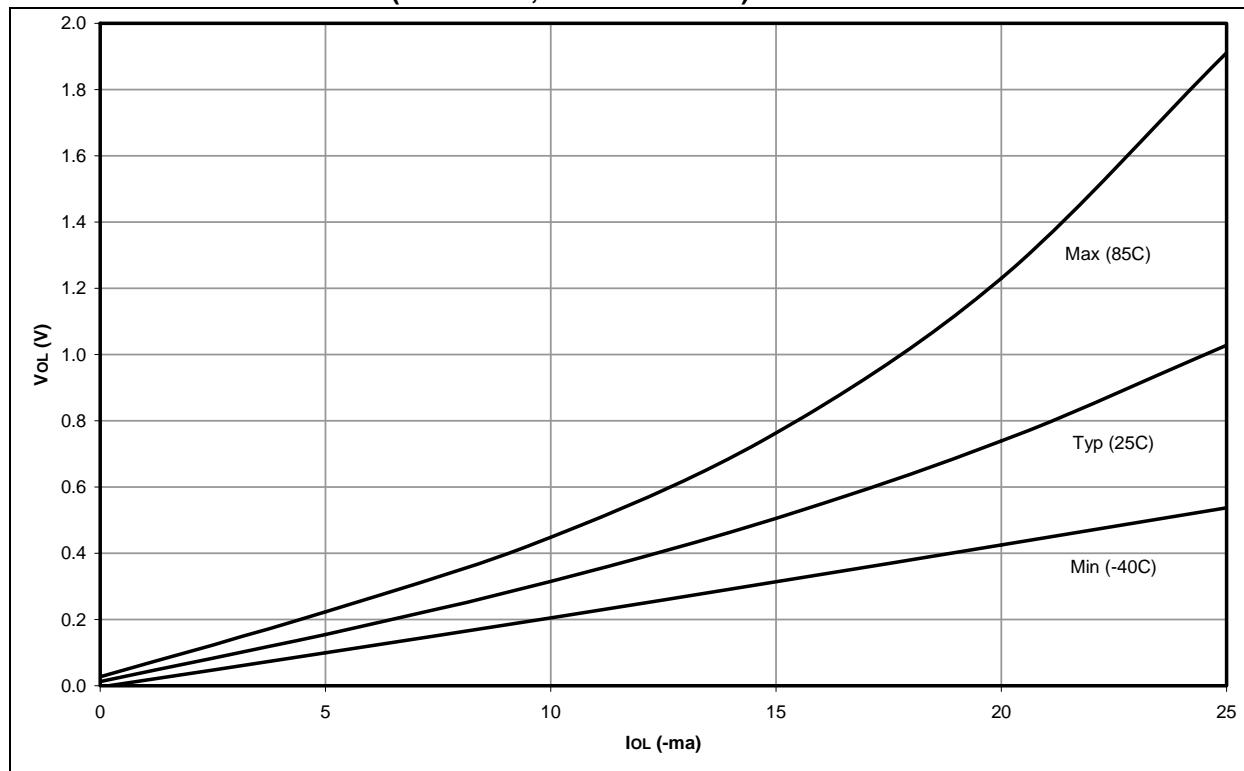
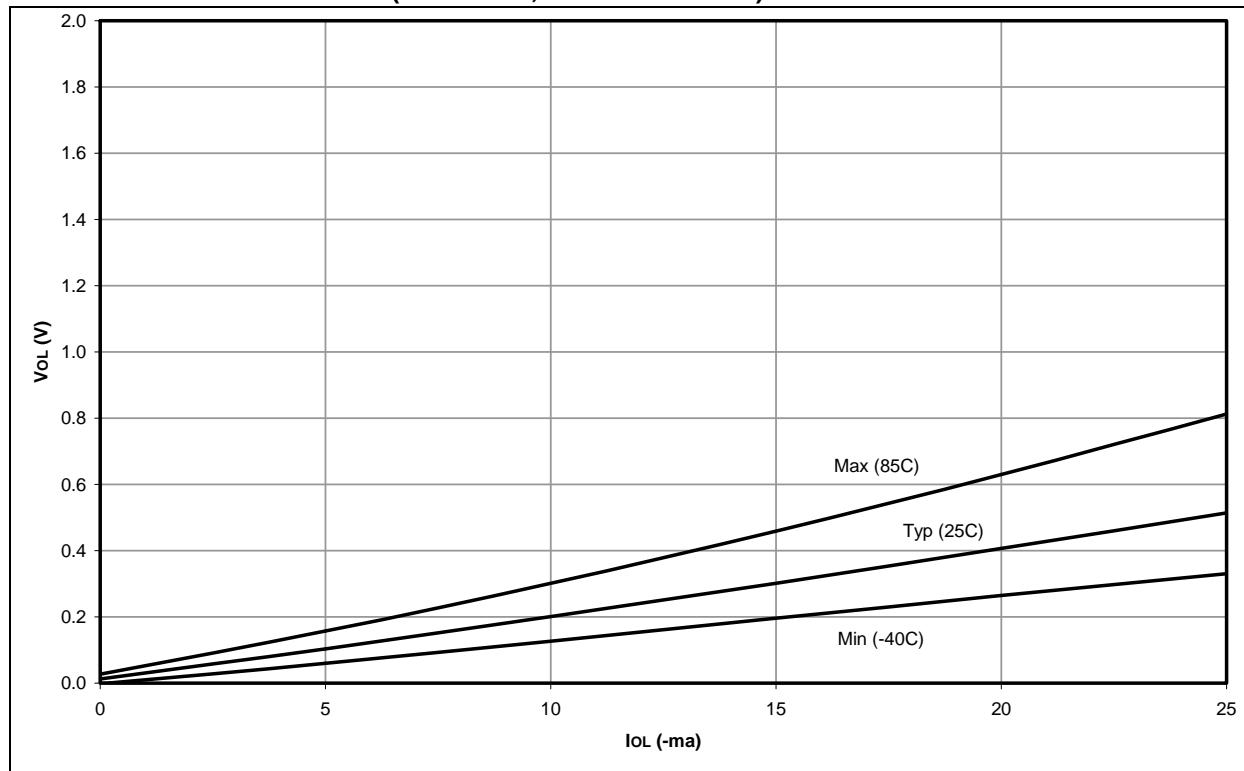
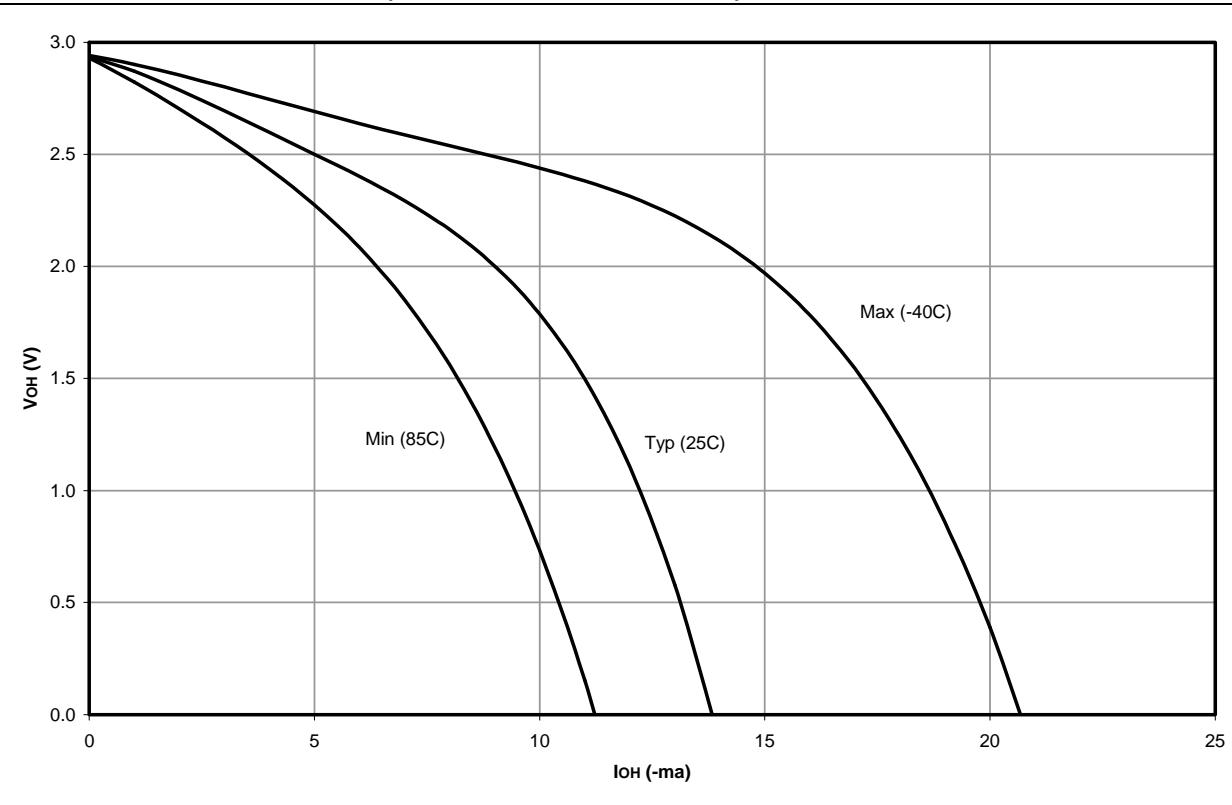


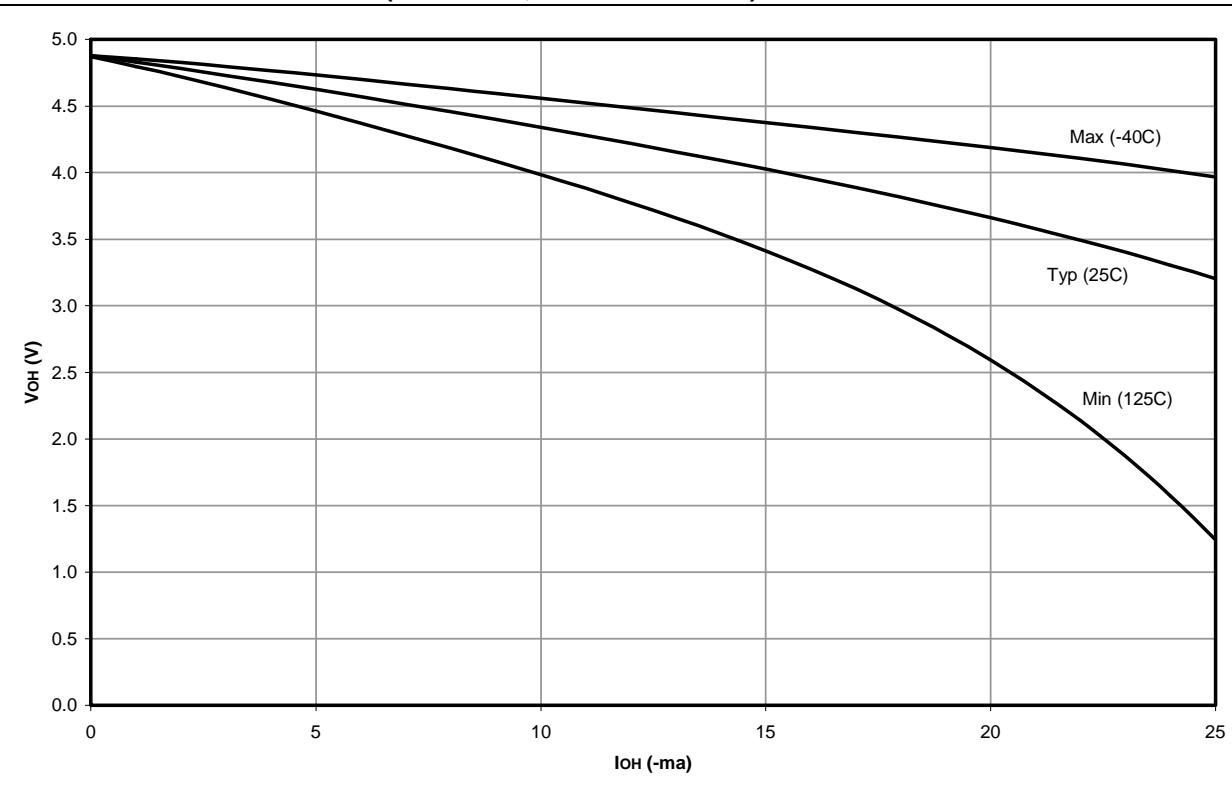
FIGURE 27-34: VOL vs. IOL (VDD = 5.0V, -40°C TO +125°C)



**FIGURE 27-35:  $V_{OH}$  vs.  $I_{OH}$  ( $V_{DD} = 3.0V$ ,  $-40^{\circ}C$  TO  $+85^{\circ}C$ )**



**FIGURE 27-36:  $V_{OH}$  vs.  $I_{OH}$  ( $V_{DD} = 5.0V$ ,  $-40^{\circ}C$  TO  $+125^{\circ}C$ )**



# PIC18F2525/2620/4525/4620

FIGURE 27-37: INTOSC FREQUENCY vs. V<sub>DD</sub>, TEMPERATURE (-40°C, +25°C, +85°C, +125°C)

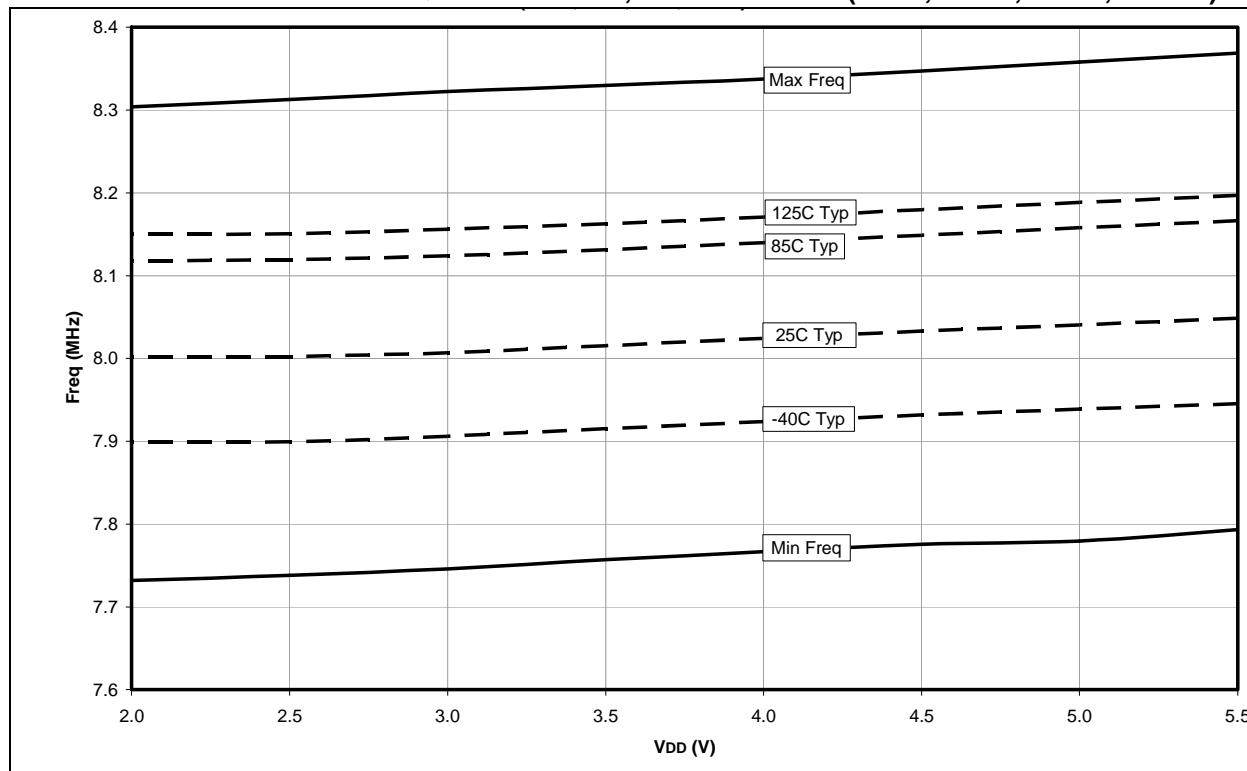
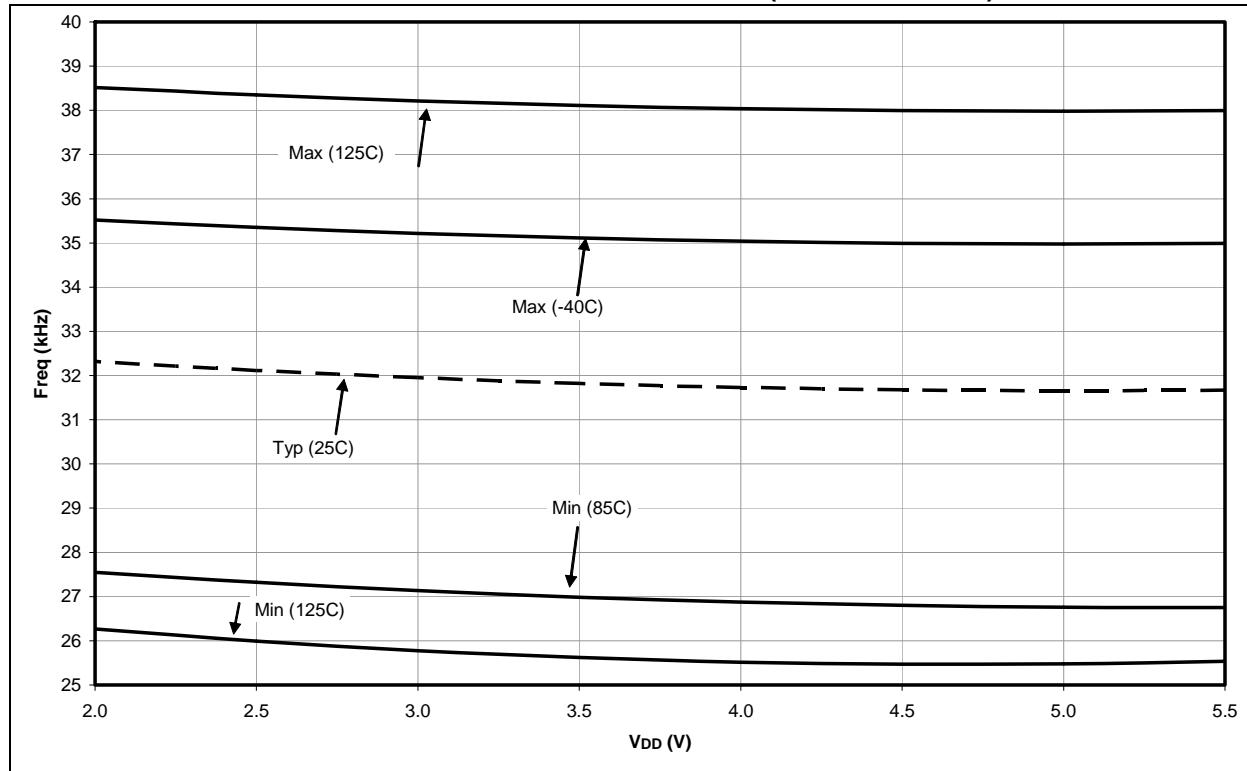
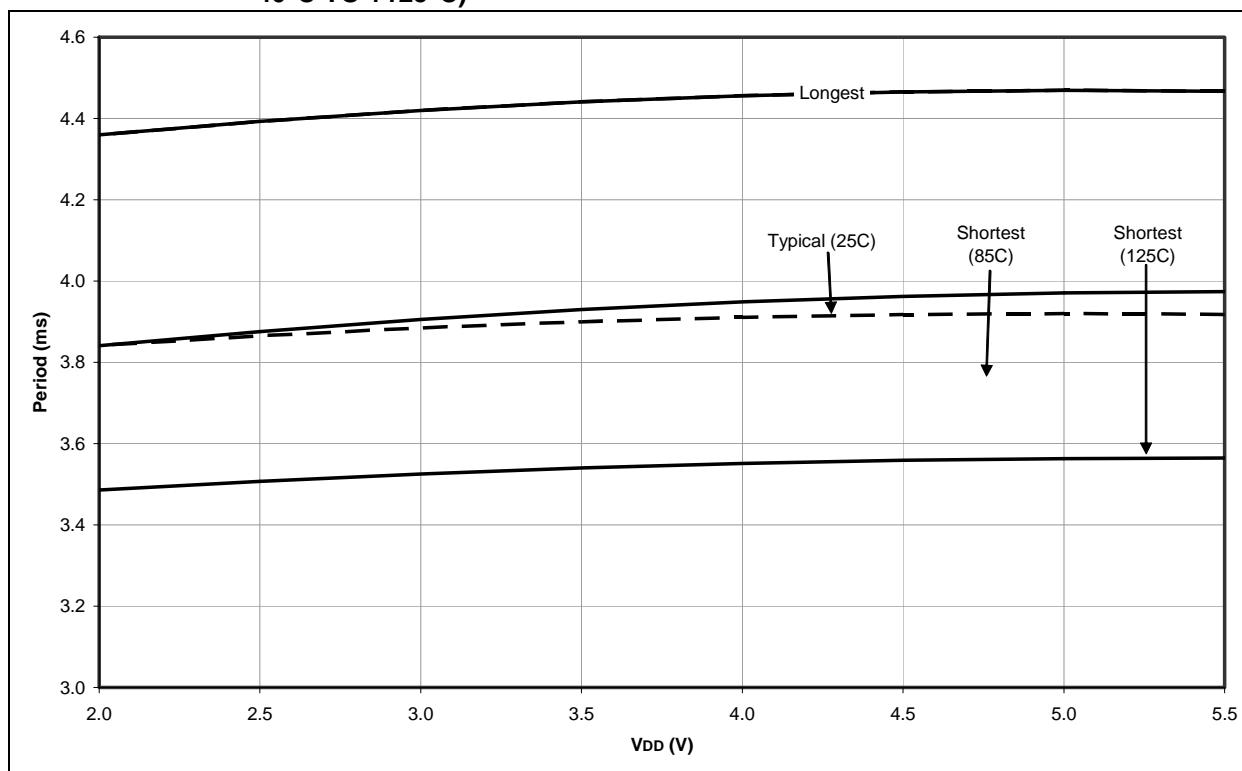


FIGURE 27-38: INTRC vs. V<sub>DD</sub> ACROSS TEMPERATURE (-40°C TO +125°C)



# PIC18F2525/2620/4525/4620

**FIGURE 27-39: WDT PERIOD vs. V<sub>DD</sub> ACROSS TEMPERATURE (1:1 POSTSCALER, -40°C TO +125°C)**



# **PIC18F2525/2620/4525/4620**

---

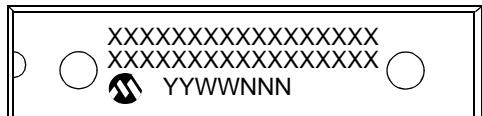
---

## **NOTES:**

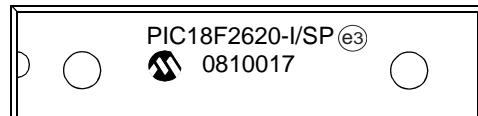
## 28.0 PACKAGING INFORMATION

### 28.1 Package Marking Information

28-Lead SPDIP



Example



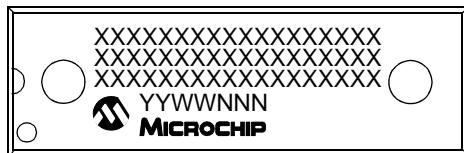
28-Lead SOIC



Example



40-Lead PDIP



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
	(e3)	Pb-free JEDEC designator for Matte Tin (Sn)
*		This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

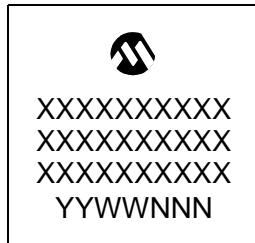
**Note:** In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

# PIC18F2525/2620/4525/4620

---

## 28.1 Package Marking Information (Continued)

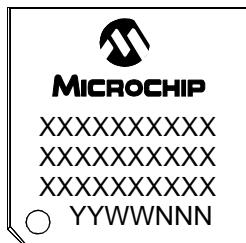
44-Lead QFN



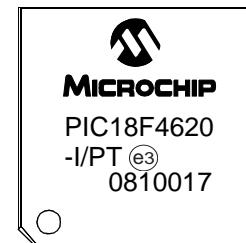
Example



44-Lead TQFP



Example

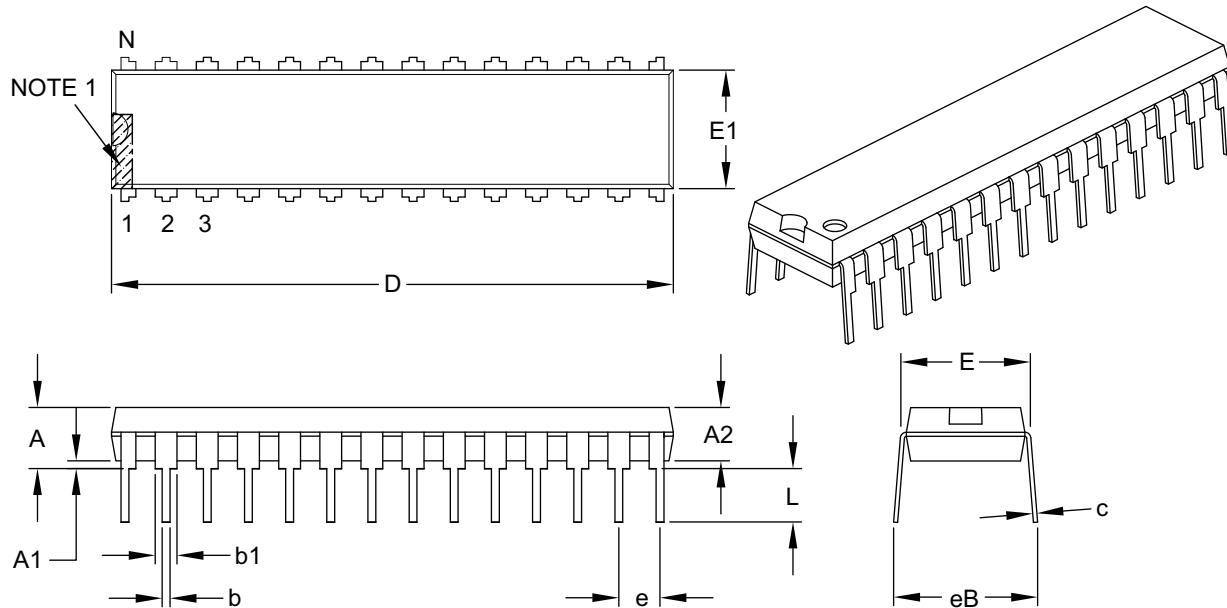


## 28.2 Package Details

The following sections give the technical details of the packages.

### 28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		.100 BSC	
Top to Seating Plane	A	—	—	.200
Molded Package Thickness	A2	.120	.135	.150
Base to Seating Plane	A1	.015	—	—
Shoulder to Shoulder Width	E	.290	.310	.335
Molded Package Width	E1	.240	.285	.295
Overall Length	D	1.345	1.365	1.400
Tip to Seating Plane	L	.110	.130	.150
Lead Thickness	c	.008	.010	.015
Upper Lead Width	b1	.040	.050	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	—	—	.430

#### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

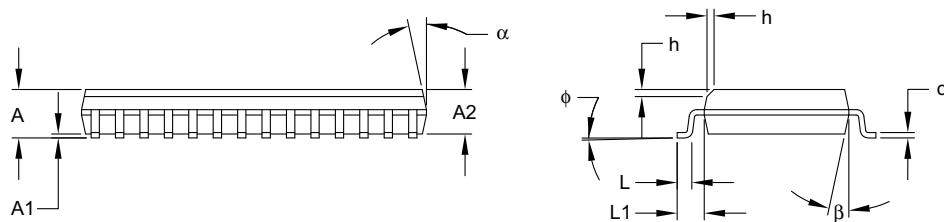
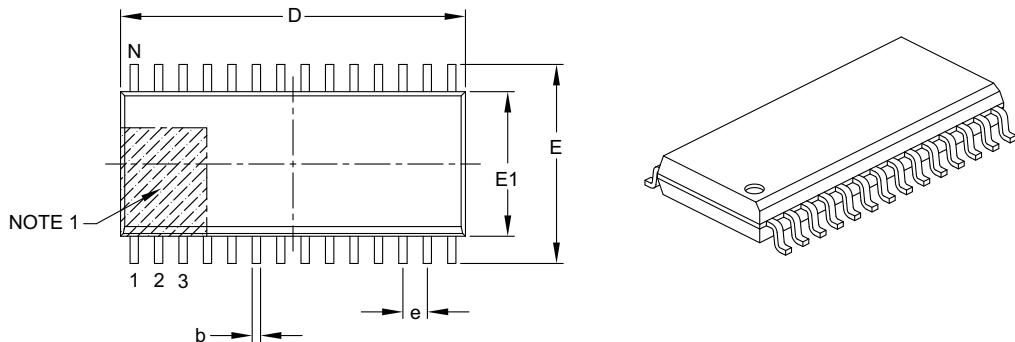
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

# PIC18F2525/2620/4525/4620

## 28-Lead Plastic Small Outline (SO) – Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Dimension Limits		MILLIMETERS		
		MIN	NOM	MAX
Number of Pins	N	28		
Pitch	e	1.27 BSC		
Overall Height	A	–	–	2.65
Molded Package Thickness	A2	2.05	–	–
Standoff §	A1	0.10	–	0.30
Overall Width	E	10.30 BSC		
Molded Package Width	E1	7.50 BSC		
Overall Length	D	17.90 BSC		
Chamfer (optional)	h	0.25	–	0.75
Foot Length	L	0.40	–	1.27
Footprint	L1	1.40 REF		
Foot Angle Top	ϕ	0°	–	8°
Lead Thickness	c	0.18	–	0.33
Lead Width	b	0.31	–	0.51
Mold Draft Angle Top	α	5°	–	15°
Mold Draft Angle Bottom	β	5°	–	15°

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

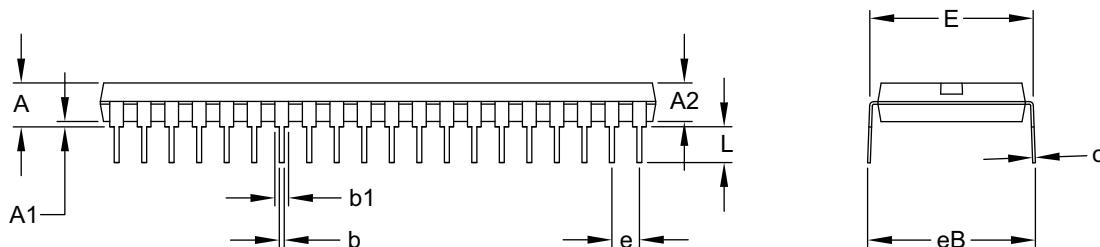
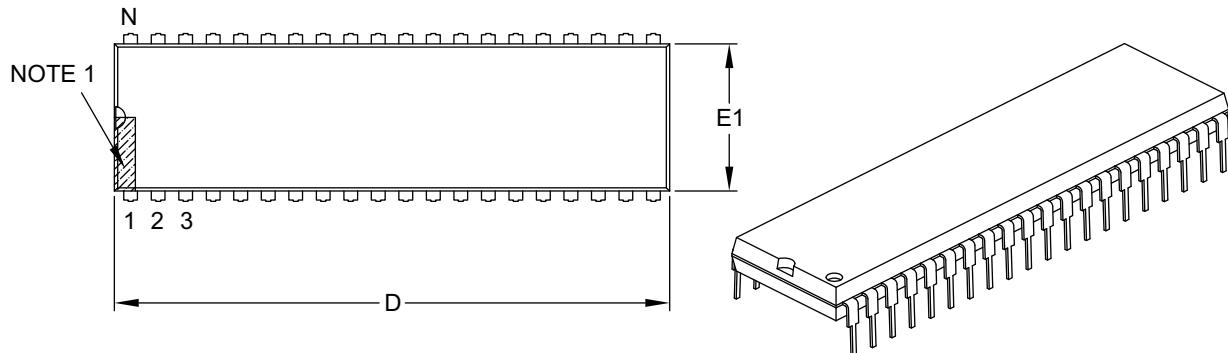
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-052B

# PIC18F2525/2620/4525/4620

## 40-Lead Plastic Dual In-Line (P) – 600 mil Body [PDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		40	
Pitch	e		.100 BSC	
Top to Seating Plane	A	—	—	.250
Molded Package Thickness	A2	.125	—	.195
Base to Seating Plane	A1	.015	—	—
Shoulder to Shoulder Width	E	.590	—	.625
Molded Package Width	E1	.485	—	.580
Overall Length	D	1.980	—	2.095
Tip to Seating Plane	L	.115	—	.200
Lead Thickness	c	.008	—	.015
Upper Lead Width	b1	.030	—	.070
Lower Lead Width	b	.014	—	.023
Overall Row Spacing §	eB	—	—	.700

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

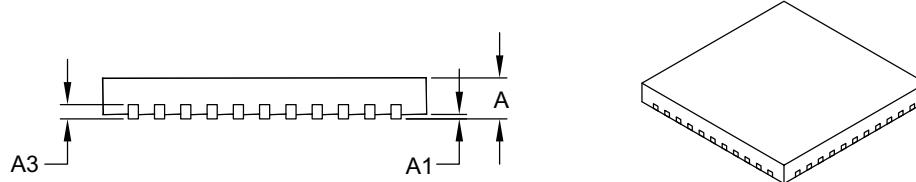
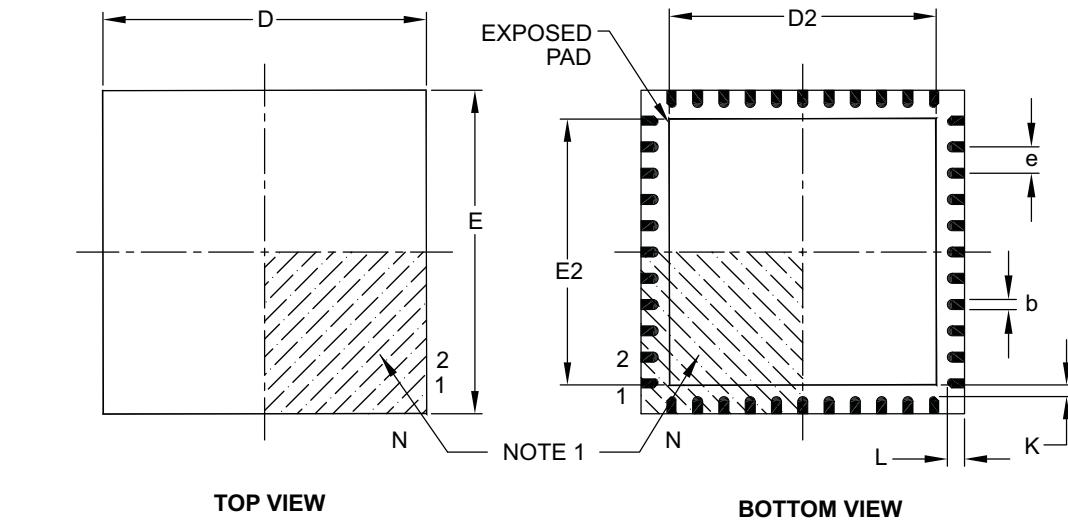
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-016B

# PIC18F2525/2620/4525/4620

## 44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	44		
Pitch	e	0.65	BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Contact Thickness	A3	0.20 REF		
Overall Width	E	8.00 BSC		
Exposed Pad Width	E2	6.30	6.45	6.80
Overall Length	D	8.00 BSC		
Exposed Pad Length	D2	6.30	6.45	6.80
Contact Width	b	0.25	0.30	0.38
Contact Length	L	0.30	0.40	0.50
Contact-to-Exposed Pad	K	0.20	–	–

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.

2. Package is saw singulated.

3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

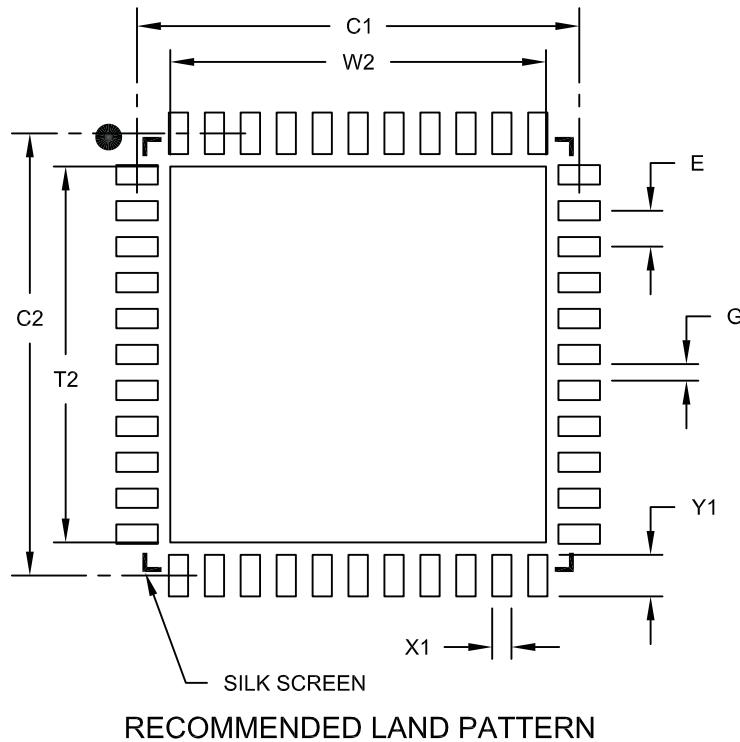
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-103B

# PIC18F2525/2620/4525/4620

## 44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.65	BSC	
Optional Center Pad Width	W2			6.80
Optional Center Pad Length	T2			6.80
Contact Pad Spacing	C1	8.00		
Contact Pad Spacing	C2	8.00		
Contact Pad Width (X44)	X1			0.35
Contact Pad Length (X44)	Y1			0.80
Distance Between Pads	G	0.25		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

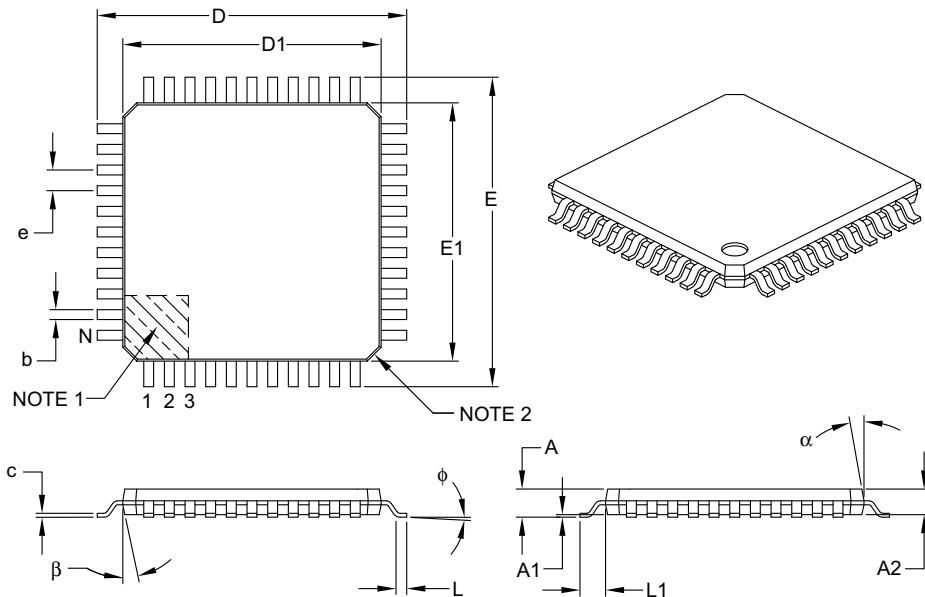
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2103A

# PIC18F2525/2620/4525/4620

## 44-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Leads	N	44		
Lead Pitch	e	0.80 BSC		
Overall Height	A	–	–	1.20
Molded Package Thickness	A2	0.95	1.00	1.05
Standoff	A1	0.05	–	0.15
Foot Length	L	0.45	0.60	0.75
Footprint	L1	1.00 REF		
Foot Angle	ϕ	0°	3.5°	7°
Overall Width	E	12.00 BSC		
Overall Length	D	12.00 BSC		
Molded Package Width	E1	10.00 BSC		
Molded Package Length	D1	10.00 BSC		
Lead Thickness	c	0.09	–	0.20
Lead Width	b	0.30	0.37	0.45
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

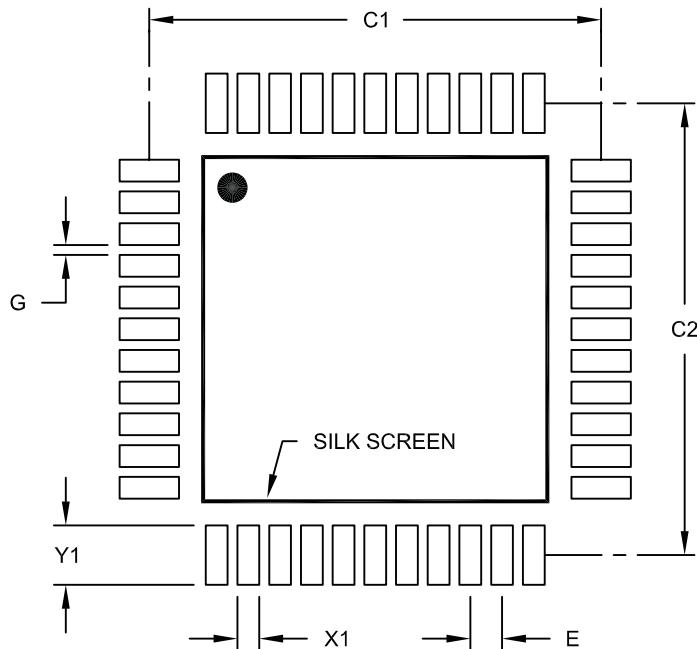
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-076B

# PIC18F2525/2620/4525/4620

## 44-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

Dimension	Limits	UNITS MILLIMETERS		
		MIN	NOM	MAX
Contact Pitch	E		0.80	BSC
Contact Pad Spacing	C1		11.40	
Contact Pad Spacing	C2		11.40	
Contact Pad Width (X44)	X1			0.55
Contact Pad Length (X44)	Y1			1.50
Distance Between Pads	G	0.25		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2076A

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## APPENDIX A: REVISION HISTORY

### Revision A (April 2004)

Original data sheet for PIC18F2525/2620/4525/4620 devices.

### Revision B (June 2004)

This revision introduces High/Low-Voltage Detect updates to Section 22.0 and includes minor corrections to the data sheet text related to the High/Low-Voltage Detect update.

### Revision C (January 2007)

This update includes revisions to the packaging diagrams.

### Revision D (November 2007)

Updated values in "Power-Down and Supply Current" tables in **Section 26.2, "DC Characteristics"** and modified Figure 17-9, "I<sup>2</sup>C Slave Mode Timing (Transmission, 7-Bit Address)". In **Section 28.2, "Package Details"**, added land-pattern drawings for both 44-lead packages.

### Revision E (May 2008)

Updated **Section 26.0 "Electrical Characteristics"**. Added characteristics data tables in **Section 27.0 "DC and AC Characteristics Graphs and Tables"**. Minor text edits throughout the rest of the document.

# PIC18F2525/2620/4525/4620

---

---

## 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	PIC18F2525	PIC18F2620	PIC18F4525	PIC18F4620
Program Memory (Bytes)	49152	65536	49152	65536
Program Memory (Instructions)	24576	32768	24576	32768
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/Compare/PWM Modules	0	0	1	1
Parallel Communications (PSP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Packages	28-Pin SPDIP 28-Pin SOIC	28-Pin SPDIP 28-Pin SOIC	40-Pin PDIP 44-Pin TQFP 44-Pin QFN	40-Pin PDIP 44-Pin TQFP 44-Pin QFN

## APPENDIX C: CONVERSION CONSIDERATIONS

This appendix discusses the considerations for converting from previous versions of a device to the ones listed in this data sheet. Typically, these changes are due to the differences in the process technology used. An example of this type of conversion is from a PIC16C74A to a PIC16C74B.

**Not Applicable**

## APPENDIX D: MIGRATION FROM BASELINE TO ENHANCED DEVICES

This section discusses how to migrate from a Baseline device (i.e., PIC16C5X) to an Enhanced MCU device (i.e., PIC18FXXX).

The following are the list of modifications over the PIC16C5X microcontroller family:

**Not Currently Available**

# PIC18F2525/2620/4525/4620

---

---

## 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.

## INDEX

### A

A/D .....	223
A/D Converter Interrupt, Configuring .....	227
Acquisition Requirements .....	228
ADCON0 Register .....	223
ADCON1 Register .....	223
ADCON2 Register .....	223
ADRESH Register .....	223, 226
ADRESL Register .....	223
Analog Port Pins, Configuring .....	230
Associated Registers .....	232
Configuring the Module .....	227
Conversion Clock (TAD) .....	229
Conversion Status (GO/DONE Bit) .....	226
Conversions .....	231
Converter Characteristics .....	359
Discharge .....	231
Operation in Power-Managed Modes .....	230
Selecting and Configuring Acquisition Time .....	229
Special Event Trigger (CCP) .....	232
Special Event Trigger (ECCP) .....	148
Use of the CCP2 Trigger .....	232
Absolute Maximum Ratings .....	321
AC (Timing) Characteristics .....	340
Load Conditions for Device	
Timing Specifications .....	341
Parameter Symbology .....	340
Temperature and Voltage Specifications .....	341
Timing Conditions .....	341
AC Characteristics	
Internal RC Accuracy .....	343
Access Bank	
Mapping with Indexed Literal Offset	
Addressing Mode .....	71
Remapping with Indexed Literal Offset	
Addressing Mode .....	71
ACKSTAT .....	191
ACKSTAT Status Flag .....	191
ADCON0 Register .....	223
GO/DONE Bit .....	226
ADCON1 Register .....	223
ADCON2 Register .....	223
ADDFSR .....	310
ADDLW .....	273
ADDULNK .....	310
ADDWF .....	273
ADDWFC .....	274
ADRESH Register .....	223
ADRESL Register .....	223, 226
Analog-to-Digital Converter. See A/D.	
ANDLW .....	274
ANDWF .....	275
Assembler	
MPASM Assembler .....	318
Auto-Wake-up on Sync Break Character .....	214
<b>B</b>	
Bank Select Register (BSR) .....	59
Baud Rate Generator .....	187
BC .....	275
BCF .....	276
BF .....	191
BF Status Flag .....	191

### Block Diagrams

A/D .....	226
Analog Input Model .....	227
Baud Rate Generator .....	187
Capture Mode Operation .....	141
Comparator Analog Input Model .....	237
Comparator I/O Operating Modes .....	234
Comparator Output .....	236
Comparator Voltage Reference .....	240
Compare Mode Operation .....	142
Device Clock .....	28
Enhanced PWM .....	149
EUSART Receive .....	213
EUSART Transmit .....	211
External Power-on Reset Circuit	
(Slow VDD Power-up) .....	43
Fail-Safe Clock Monitor .....	261
Generic I/O Port .....	91
High/Low-Voltage Detect with External Input .....	244
Interrupt Logic .....	110
MSSP (I <sup>2</sup> C Master Mode) .....	185
MSSP (I <sup>2</sup> C Mode) .....	170
MSSP (SPI Mode) .....	161
On-Chip Reset Circuit .....	41
PIC18F2525/2620 .....	10
PIC18F4525/4620 .....	11
PLL (HS Mode) .....	25
PORTD and PORTE (Parallel Slave Port) .....	106
PWM Operation (Simplified) .....	144
Reads from Flash Program Memory .....	83
Single Comparator .....	235
Table Read Operation .....	79
Table Write Operation .....	80
Table Writes to Flash Program Memory .....	85
Timer0 in 16-Bit Mode .....	124
Timer0 in 8-Bit Mode .....	124
Timer1 .....	128
Timer1 (16-Bit Read/Write Mode) .....	128
Timer2 .....	134
Timer3 .....	136
Timer3 (16-Bit Read/Write Mode) .....	136
Voltage Reference Output Buffer Example .....	241
Watchdog Timer .....	258
BN .....	276
BNC .....	277
BNN .....	277
BNOV .....	278
BNZ .....	278
BOR. See Brown-out Reset.	
BOV .....	281
BRA .....	279
Break Character (12-Bit) Transmit and Receive .....	216
BRG. See Baud Rate Generator.	
Brown-out Reset (BOR) .....	44
Detecting .....	44
Disabling in Sleep Mode .....	44
Software Enabled .....	44
BSF .....	279
BTFS .....	280
BTFS .....	280
BTG .....	281
BZ .....	282

# PIC18F2525/2620/4525/4620

---

## C

C Compilers	
MPLAB C18	318
MPLAB C30	318
CALL	282
CALLW	311
Capture (CCP Module)	141
Associated Registers	143
CCP Pin Configuration	141
CCPRxH:CCPRxL Registers	141
Prescaler	141
Software Interrupt	141
Timer1/Timer3 Mode Selection	141
Capture (ECCP Module)	148
Capture/Compare/PWM (CCP)	139
Capture Mode. See Capture.	
CCPRxH Register	140
CCPRxL Register	140
Compare Mode. See Compare.	
Interaction of Two CCP Modules	140
Module Configuration	140
Pin Assignment	140
Timer Resources	140
Clock Sources	28
Selecting the 31 kHz Source	29
Selection Using OSCCON Register	29
CLRF	283
CLRWDT	283
Code Examples	
16 x 16 Signed Multiply Routine	90
16 x 16 Unsigned Multiply Routine	90
8 x 8 Signed Multiply Routine	89
8 x 8 Unsigned Multiply Routine	89
Changing Between Capture Prescalers	141
Computed GOTO Using an Offset Value	56
Data EEPROM Read	75
Data EEPROM Refresh Routine	76
Data EEPROM Write	75
Erasing a Flash Program Memory Row	84
Fast Register Stack	56
How to Clear RAM (Bank 1) Using Indirect Addressing	67
Implementing a Real-Time Clock Using a Timer1 Interrupt Service	131
Initializing PORTA	91
Initializing PORTB	94
Initializing PORTC	97
Initializing PORTD	100
Initializing PORTE	103
Loading the SSPBUF (SSPSR) Register	164
Reading a Flash Program Memory Word	83
Saving STATUS, WREG and BSR Registers in RAM	121
Writing to Flash Program Memory	86-87
Code Protection	249, 263
Associated Registers	263
Configuration Register Protection	266
Data EEPROM	266
Program Memory	264
COMF	284
Comparator	233
Analog Input Connection Considerations	237
Associated Registers	237
Configuration	234
Effects of a Reset	236

Interrupts	236
Operation	235
Operation During Sleep	236
Outputs	235
Reference	235
External Signal	235
Internal Signal	235
Response Time	235
Comparator Specifications	338
Comparator Voltage Reference	239
Accuracy and Error	240
Associated Registers	241
Configuring	239
Connection Considerations	240
Effects of a Reset	240
Operation During Sleep	240
Compare (CCP Module)	142
Associated Registers	143
CCPx Register	142
Pin Configuration	142
Software Interrupt	142
Special Event Trigger	137, 142, 232
Timer1/Timer3 Mode Selection	142
Compare (ECCP Module)	148
Special Event Trigger	148
Computed GOTO	56
Configuration Bits	249
Context Saving During Interrupts	121
Conversion Considerations	395
CPFSEQ	284
CPFSGT	285
CPFSLT	285
Crystal Oscillator/Ceramic Resonator	23
Customer Change Notification Service	407
Customer Notification Service	407
Customer Support	407
<b>D</b>	
Data Addressing Modes	67
Comparing Options with the Extended Instruction Set Enabled	70
Direct	67
Indexed Literal Offset	69
Instructions Affected	69
Indirect	67
Inherent and Literal	67
Data EEPROM Memory	73
Associated Registers	77
EEADR and EEADRH Registers	73
EECON1 and EECON2 Registers	73
Operation During Code-Protect	76
Protection Against Spurious Write	76
Reading	75
Using	76
Write Verify	75
Writing	75
Data Memory	59
Access Bank	61
and the Extended Instruction Set	69
Bank Select Register (BSR)	59
General Purpose Registers	61
Map for PIC18FX525/X620	60
Special Function Registers	62
DAW	286
DC and AC Characteristics	
Graphs and Tables	361

DC Characteristics .....	335	Synchronous Master Mode .....	217
Power-Down and Supply Current .....	325	Associated Registers, Receive .....	219
Supply Voltage .....	324	Associated Registers, Transmit .....	218
DCFSNZ .....	287	Reception .....	219
DECFSZ .....	286	Transmission .....	217
DECF .....	287	Synchronous Slave Mode .....	220
Development Support .....	317	Associated Registers, Receive .....	221
Device Differences .....	394	Associated Registers, Transmit .....	220
Device Overview .....	7	Reception .....	221
Details on Individual Family Members .....	8	Transmission .....	220
Features (table) .....	9	Extended Instruction Set .....	
New Core Features .....	7	ADDFSR .....	310
Other Special Features .....	8	ADDULNK .....	310
Device Reset Timers .....	45	and Using MPLAB IDE Tools .....	316
Oscillator Start-up Timer (OST) .....	45	CALLW .....	311
PLL Lock Time-out .....	45	Considerations for Use .....	314
Power-up Timer (PWRT) .....	45	MOVSF .....	311
Time-out Sequence .....	45	MOVSS .....	312
Direct Addressing .....	68	PUSHL .....	312
<b>E</b> .....		SUBFSR .....	313
Effect on Standard PIC MCU Instructions .....	314	SUBULNK .....	313
Effects of Power-Managed Modes on .....		Syntax .....	309
Various Clock Sources .....	31	External Clock Input .....	24
Electrical Characteristics .....	321	<b>F</b> .....	
Enhanced Capture/Compare/PWM (ECCP) .....	147	Fail-Safe Clock Monitor .....	249, 261
Associated Registers .....	160	Exiting Operation .....	261
Capture and Compare Modes .....	148	Interrupts in Power-Managed Modes .....	262
Capture Mode. See Capture (ECCP Module). .....		POR or Wake from Sleep .....	262
Outputs and Configuration .....	148	WDT During Oscillator Failure .....	261
Pin Configurations for ECCP1 .....	148	Fast Register Stack .....	56
PWM Mode. See PWM (ECCP Module). .....		Firmware Instructions .....	267
Standard PWM Mode .....	148	Flash Program Memory .....	79
Timer Resources .....	148	Associated Registers .....	87
Enhanced PWM Mode. See PWM (ECCP Module). .....	149	Control Registers .....	80
Enhanced Universal Synchronous Asynchronous .....		EECON1 and EECON2 .....	80
Receiver Transmitter (EUSART). See EUSART. .....		TABLAT (Table Latch) Register .....	82
Equations .....		TBLPTR (Table Pointer) Register .....	82
A/D Acquisition Time .....	228	Erase Sequence .....	84
A/D Minimum Charging Time .....	228	Erasing .....	84
Calculating the Minimum Required .....		Operation During Code-Protect .....	87
Acquisition Time .....	228	Reading .....	83
Errata .....	5	Table Pointer .....	
EUSART .....		Boundaries Based on Operation .....	82
Asynchronous Mode .....	211	Operations with TBLRD and	
12-Bit Break Transmit and Receive .....	216	TBLWT (table) .....	82
Associated Registers, Receive .....	214	Table Pointer Boundaries .....	82
Associated Registers, Transmit .....	212	Table Reads and Table Writes .....	79
Auto-Wake-up on Sync Break .....	214	Write Sequence .....	85
Receiver .....	213	Writing .....	85
Setting up 9-Bit Mode with .....		Protection Against Spurious Writes .....	87
Address Detect .....	213	Unexpected Termination .....	87
Transmitter .....	211	Write Verify .....	87
Baud Rate Generator .....		FSCM. See Fail-Safe Clock Monitor. .....	
Operation in Power-Managed Mode .....	205	<b>G</b> .....	
Baud Rate Generator (BRG) .....	205	GOTO .....	288
Associated Registers .....	206	<b>H</b> .....	
Auto-Baud Rate Detect .....	209	Hardware Multiplier .....	89
Baud Rate Error, Calculating .....	206	Introduction .....	89
Baud Rates, Asynchronous Modes .....	207	Operation .....	89
High Baud Rate Select (BRGH Bit) .....	205	Performance Comparison .....	89
Sampling .....	205		

# PIC18F2525/2620/4525/4620

---

High/Low-Voltage Detect .....	243	Instruction Set .....	267
Applications .....	246	ADDLW .....	273
Associated Registers .....	247	ADDWF .....	273
Characteristics .....	339	ADDWF (Indexed Literal Offset Mode) .....	315
Current Consumption .....	245	ADDWFC .....	274
Effects of a Reset .....	247	ANDLW .....	274
Operation .....	244	ANDWF .....	275
During Sleep .....	247	BC .....	275
Setup .....	245	BCF .....	276
Start-up Time .....	245	BN .....	276
Typical Application .....	246	BNC .....	277
HLVD. See High/Low-Voltage Detect. ....	243	BNN .....	277
<b>I</b>		BNOV .....	278
I/O Ports .....	91	BNZ .....	278
I <sup>2</sup> C Mode (MSSP)		BOV .....	281
Acknowledge Sequence Timing .....	194	BRA .....	279
Baud Rate Generator .....	187	BSF .....	279
Bus Collision		BSF (Indexed Literal Offset Mode) .....	315
During a Repeated Start Condition .....	198	BTFSC .....	280
During a Start Condition .....	196	BTFSS .....	280
During a Stop Condition .....	199	BTG .....	281
Clock Arbitration .....	188	BZ .....	282
Clock Stretching .....	180	CALL .....	282
10-Bit Slave Receive Mode (SEN = 1) .....	180	CLRF .....	283
10-Bit Slave Transmit Mode .....	180	CLRWDT .....	283
7-Bit Slave Receive Mode (SEN = 1) .....	180	COMF .....	284
7-Bit Slave Transmit Mode .....	180	CPFSEQ .....	284
Clock Synchronization and the CKP bit		CPFSGT .....	285
(SEN = 1) .....	181	CPFSLT .....	285
Effects of a Reset .....	195	DAW .....	286
General Call Address Support .....	184	DCFSNZ .....	287
I <sup>2</sup> C Clock Rate w/BRG .....	187	DECFSZ .....	287
Master Mode .....	185	Extended Instruction Set .....	309
Operation .....	186	General Format .....	269
Reception .....	191	GOTO .....	288
Repeated Start Condition Timing .....	190	INCFSZ .....	288
Start Condition Timing .....	189	INFSNZ .....	289
Transmission .....	191	IORLW .....	290
Multi-Master Communication, Bus Collision		IORWF .....	290
and Arbitration .....	195	LFSR .....	291
Multi-Master Mode .....	195	MOVF .....	291
Operation .....	174	MOVFF .....	292
Read/Write Bit Information (R/W Bit) .....	174, 175	MOVLB .....	292
Registers .....	170	MOV LW .....	293
Serial Clock (RC3/SCK/SCL) .....	175	MOVWF .....	293
Slave Mode .....	174	MULLW .....	294
Addressing .....	174	MULWF .....	294
Reception .....	175	NEGF .....	295
Transmission .....	175	NOP .....	295
Sleep Operation .....	195	Opcode Field Descriptions .....	268
Stop Condition Timing .....	194	POP .....	296
ID Locations .....	249, 266	PUSH .....	296
INCFSZ .....	289	RCALL .....	297
In-Circuit Debugger .....	266	RESET .....	297
In-Circuit Serial Programming (ICSP) .....	249, 266	RETFIE .....	298
Indexed Literal Offset Addressing		RETLW .....	298
and Standard PIC18 Instructions .....	314	RETURN .....	299
Indexed Literal Offset Mode .....	314	RLCF .....	299
Indirect Addressing .....	68	RLNCF .....	300
INFSNZ .....	289	RRCF .....	300
Initialization Conditions for all Registers .....	49–52	RRNCF .....	301
Instruction Cycle .....	57	SETF .....	301
Clocking Scheme .....	57	SETF (Indexed Literal Offset Mode) .....	315
Instruction Flow/Pipelining .....	57		

SLEEP .....	302	MOVWF .....	293
SUBFWB .....	302	MPLAB ASM30 Assembler, Linker, Librarian .....	318
SUBLW .....	303	MPLAB ICD 2 In-Circuit Debugger .....	319
SUBWF .....	303	MPLAB ICE 2000 High-Performance Universal In-Circuit Emulator .....	319
SUBWFB .....	304	MPLAB Integrated Development Environment Software .....	317
SWAPF .....	304	MPLAB PM3 Device Programmer .....	319
TBLRD .....	305	MPLAB REAL ICE In-Circuit Emulator System .....	319
TBLWT .....	306	MPLINK Object Linker/MPLIB Object Librarian .....	318
TSTFSZ .....	307	<b>MSSP</b>	
XORLW .....	307	ACK Pulse .....	174, 175
XORWF .....	308	Control Registers (general) .....	161
<b>INTCON Registers</b> .....	111–113	I <sup>2</sup> C Mode. See I <sup>2</sup> C Mode.	
Inter-Integrated Circuit. See I <sup>2</sup> C.		Module Overview .....	161
Internal Oscillator Block .....	26	SPI Master/Slave Connection .....	165
Adjustment .....	26	SPI Mode. See SPI Mode.	
INTIO Modes .....	26	SSPBUF Register .....	166
INTOSC Frequency Drift .....	26	SSPSR Register .....	166
INTOSC Output Frequency .....	26	MULLW .....	294
OSCTUNE Register .....	26	MULWF .....	294
PLL in INTOSC Modes .....	26		
Internal RC Oscillator .....		<b>N</b>	
Use with WDT .....	258	NEGF .....	295
Internet Address .....	407	NOP .....	295
Interrupt Sources .....	249		
A/D Conversion Complete .....	227	<b>O</b>	
Capture Complete (CCP) .....	141	Oscillator Configuration .....	23
Compare Complete (CCP) .....	142	EC .....	23
Interrupt-on-Change (RB7:RB4) .....	94	ECIO .....	23
INTx Pin .....	121	HS .....	23
PORTB, Interrupt-on-Change .....	121	HSPLL .....	23
TMR0 .....	121	Internal Oscillator Block .....	26
TMR0 Overflow .....	125	INTIO1 .....	23
TMR1 Overflow .....	127	INTIO2 .....	23
TMR2 to PR2 Match (PWM) .....	144, 149	LP .....	23
TMR3 Overflow .....	135, 137	RC .....	23
Interrupts .....	109	RCIO .....	23
Interrupts, Flag Bits .....		XT .....	23
Interrupt-on-Change (RB7:RB4) .....		Oscillator Selection .....	249
Flag (RBIF Bit) .....	94	Oscillator Start-up Timer (OST) .....	31, 45
INTOSC, INTRC. See Internal Oscillator Block.		Oscillator Switching .....	28
IORLW .....	290	Oscillator Transitions .....	29
IORWF .....	290	Oscillator, Timer1 .....	127, 137
IPR Registers .....	118	Oscillator, Timer3 .....	135
<b>L</b>			
LFSR .....	291	<b>P</b>	
Low-Voltage ICSP Programming. See Single-Supply ICSP Programming .....		Packaging Information .....	383
<b>M</b>		Details .....	385
Master Clear (MCLR) .....	43	Marking .....	383
Master Synchronous Serial Port (MSSP). See MSSP.		Parallel Slave Port (PSP) .....	100, 106
Memory Organization .....	53	Associated Registers .....	107
Data Memory .....	59	CS (Chip Select) .....	106
Program Memory .....	53	PORTD .....	106
Memory Programming Requirements .....	337	RD (Read Input) .....	106
Microchip Internet Web Site .....	407	Select (PSPMODE Bit) .....	100, 106
Migration from Baseline to Enhanced Devices .....	395	WR (Write Input) .....	106
Migration from High-End to Enhanced Devices .....	396	PICSTART Plus Development Programmer .....	320
Migration from Mid-Range to Enhanced Devices .....	396	PIE Registers .....	116
MOVF .....	291	Pin Functions .....	
MOVFF .....	292	MCLR/VPP/RE3 .....	12, 16
MOVLB .....	292	OSC1/CLK1/RA7 .....	12, 16
MOVLW .....	293	OSC2/CLK0/RA6 .....	12, 16
MOVSF .....	311	RA0/AN0 .....	13, 17
MOVSS .....	312	RA1/AN1 .....	13, 17
		RA2/AN2/VREF-/CVREF .....	13, 17

# PIC18F2525/2620/4525/4620

---

RA3/AN3/VREF+	13, 17	TRISD Register	100
RA4/T0CKI/C1OUT	13, 17	PORTE	
RA5/AN4/SS/HLDIN/C2OUT	13, 17	Associated Registers	105
RB0/INT0/FLT0/AN12	14, 18	LATE Register	103
RB1/INT1/AN10	14, 18	PORTE Register	103
RB2/INT2/AN8	14, 18	PSP Mode Select (PSPMODE Bit)	100
RB3/AN9/CCP2	14, 18	TRISE Register	103
RB4/KBI0/AN11	14, 18	Power-Managed Modes	33
RB5/KBI1/PGM	14, 18	and A/D Operation	230
RB6/KBI2/PGC	14, 18	and EUSART Operation	205
RB7/KBI3/PGD	14, 18	and PWM Operation	159
RC0/T1OSO/T13CKI	15, 19	and SPI Operation	169
RC1/T1OSI/CCP2	15, 19	Clock Sources	33
RC2/CCP1	15	Clock Transitions and Status Indicators	34
RC2/CCP1/P1A	19	Effects on Clock Sources	31
RC3/SCK/SCL	15, 19	Entering	33
RC4/SDI/SDA	15, 19	Exiting Idle and Sleep Modes	39
RC5/SDO	15, 19	By Interrupt	39
RC6/TX/CK	15, 19	By Reset	39
RC7/RX/DT	15, 19	By WDT Time-out	39
RD0/PSP0	20	Without an Oscillator Start-up Delay	40
RD1/PSP1	20	Idle Modes	37
RD2/PSP2	20	PRI_IDLE	38
RD3/PSP3	20	RC_IDLE	39
RD4/PSP4	20	SEC_IDLE	38
RD5/PSP5/P1B	20	Multiple Sleep Commands	34
RD6/PSP6/P1C	20	Run Modes	34
RD7/PSP7/P1D	20	PRI_RUN	34
RE0/RD/AN5	21	RC_RUN	35
RE1/WR/AN6	21	SEC_RUN	34
RE2/CS/AN7	21	Selecting	33
VDD	15, 21	Sleep Mode	37
Vss	15, 21	Summary (table)	33
Pinout I/O Descriptions		Power-on Reset (POR)	43
PIC18F2525/2620	12	Power-up Timer (PWRT)	45
PIC18F4525/4620	16	Time-out Sequence	45
PIR Registers	114	Power-up Delays	31
PLL Frequency Multiplier	25	Power-up Timer (PWRT)	31
HSPLL Oscillator Mode	25	Prescaler	
Use with INTOSC	25	Timer2	150
POP	296	Prescaler, Timer0	125
POR. See Power-on Reset.		Prescaler, Timer2	145
PORTA		PRI_IDLE Mode	38
Associated Registers	93	PRI_RUN Mode	34
LATA Register	91	Program Counter	54
PORTA Register	91	PCL, PCH and PCU Registers	54
TRISA Register	91	PCLATH and PCLATU Registers	54
PORTB		Program Memory	
Associated Registers	96	And Extended Instruction Set	71
LATB Register	94	Instructions	58
PORTB Register	94	Two-Word	58
RB7:RB4 Interrupt-on-Change Flag		Interrupt Vector	53
(RBIF Bit)	94	Look-up Tables	56
TRISB Register	94	Map and Stack (diagram)	53
PORTC		Reset Vector	53
Associated Registers	99	Program Verification	263
LATC Register	97	Programming, Device Instructions	267
PORTC Register	97	PSP. See Parallel Slave Port.	
RC3/SCK/SCL Pin	175	Pulse-Width Modulation. See PWM (CCP Module)	
TRISC Register	97	and PWM (ECCP Module).	
PORTD		PUSH	296
Associated Registers	102	PUSH and POP Instructions	55
LATD Register	100	PUSHL	312
Parallel Slave Port (PSP) Function	100		
PORTD Register	100		

PWM (CCP Module)	
Associated Registers	146
Auto-Shutdown (CCP1 Only)	145
Duty Cycle	144
Example Frequencies/Resolutions	145
Operation Setup	145
Period	144
TMR2 to PR2 Match	144, 149
PWM (ECCP Module)	149
CCPR1H:CCPR1L Registers	149
Duty Cycle	150
Effects of a Reset	159
Enhanced PWM Auto-Shutdown	156
Example Frequencies/Resolutions	150
Full-Bridge Application Example	154
Full-Bridge Mode	153
Direction Change	154
Half-Bridge Mode	152
Half-Bridge Output Mode Applications Example	152
Operation in Power-Managed Modes	159
Operation with Fail-Safe Clock Monitor	159
Output Configurations	150
Output Relationships (Active-High)	151
Output Relationships (Active-Low)	151
Period	149
Programmable Dead-Band Delay	156
Setup for PWM Operation	159
Start-up Considerations	158
<b>Q</b>	
Q Clock	145, 150
<b>R</b>	
RAM. See Data Memory.	
RBIF Bit	94
RC Oscillator	25
RCIO Oscillator Mode	25
RC_IDLE Mode	39
RC_RUN Mode	35
RCALL	297
RCON Register Bit Status During Initialization	48
Reader Response	408
Register File	61
Register File Summary	63–65
Registers	
ADCON0 (A/D Control 0)	223
ADCON1 (A/D Control 1)	224
ADCON2 (A/D Control 2)	225
BAUDCON (Baud Rate Control)	204
CCP1CON (ECCP Control, 40/44-Pin Devices)	147
CCPxCON (CCPx Control, 28-Pin Devices)	139
CMCON (Comparator Control)	233
CONFIG1H (Configuration 1 High)	250
CONFIG2H (Configuration 2 High)	252
CONFIG2L (Configuration 2 Low)	251
CONFIG3H (Configuration 3 High)	253
CONFIG4L (Configuration 4 Low)	253
CONFIG5H (Configuration 5 High)	254
CONFIG5L (Configuration 5 Low)	254
CONFIG6H (Configuration 6 High)	255
CONFIG6L (Configuration 6 Low)	255
CONFIG7H (Configuration 7 High)	256
CONFIG7L (Configuration 7 Low)	256
CVRCON (Comparator Voltage Reference Control)	239
DEVID1 (Device ID 1)	257
DEVID2 (Device ID 2)	257
ECCP1AS (ECCP Auto-Shutdown Control)	157
EECON1 (EEPROM Control 1)	74, 81
HLVDCON (High/Low-Voltage Detect Control)	243
INTCON (Interrupt Control)	111
INTCON2 (Interrupt Control 2)	112
INTCON3 (Interrupt Control 3)	113
IPR1 (Peripheral Interrupt Priority 1)	118
IPR2 (Peripheral Interrupt Priority 2)	119
OSCCON (Oscillator Control)	30
OSCTUNE (Oscillator Tuning)	27
PIE1 (Peripheral Interrupt Enable 1)	116
PIE2 (Peripheral Interrupt Enable 2)	117
PIR1 (Peripheral Interrupt Request (Flag) 1)	114
PIR2 (Peripheral Interrupt Request (Flag) 2)	115
PWM1CON (PWM Configuration)	156
RCON (Reset Control)	42, 120
RCSTA (Receive Status and Control)	203
SSPCON1 (MSSP Control 1, I <sup>2</sup> C Mode)	172
SSPCON1 (MSSP Control 1, SPI Mode)	163
SSPCON2 (MSSP Control 2, I <sup>2</sup> C Mode)	173
SSPSTAT (MSSP Status, I <sup>2</sup> C Mode)	171
SSPSTAT (MSSP Status, SPI Mode)	162
STATUS	66
STKPTR (Stack Pointer)	55
T0CON (Timer0 Control)	123
T1CON (Timer1 Control)	127
T2CON (Timer2 Control)	133
T3CON (Timer3 Control)	135
TRISE (PORTE/PSP Control)	104
TXSTA (Transmit Status and Control)	202
WDTCON (Watchdog Timer Control)	259
<b>RESET</b>	297
Reset State of Registers	48
Resets	41, 249
Brown-out Reset (BOR)	249
Oscillator Start-up Timer (OST)	249
Power-on Reset (POR)	249
Power-up Timer (PWRT)	249
RETFIE	298
RETLW	298
RETURN	299
Return Address Stack	54
Associated Registers	54
Return Stack Pointer (STKPTR)	55
Revision History	393
RLCF	299
RLNCF	300
RRCF	300
RRNCF	301
<b>S</b>	
SCK	161
SDI	161
SDO	161
SEC_IDLE Mode	38
SEC_RUN Mode	34
Serial Clock, SCK	161
Serial Data In (SDI)	161
Serial Data Out (SDO)	161
Serial Peripheral Interface. See SPI Mode.	
SETF	301
Single-Supply ICSP Programming.	
Slave Select (SS)	161
SLEEP	302

# PIC18F2525/2620/4525/4620

---

Sleep	
OSC1 and OSC2 Pin States	31
Software Simulator (MPLAB SIM)	318
Special Event Trigger. See Compare (ECCP Mode).	
Special Event Trigger. See Compare (ECCP Module).	
Special Features of the CPU	249
Special Function Registers	62
Map	62
SPI Mode (MSSP)	
Associated Registers	169
Bus Mode Compatibility	169
Effects of a Reset	169
Enabling SPI I/O	165
Master Mode	166
Master/Slave Connection	165
Operation	164
Operation in Power-Managed Modes	169
Serial Clock	161
Serial Data In	161
Serial Data Out	161
Slave Mode	167
Slave Select	161
Slave Select Synchronization	167
SPI Clock	166
Typical Connection	165
SS	161
SSPOV	191
SSPOV Status Flag	191
SSPSTAT Register	
R/W Bit	174, 175
Stack Full/Underflow Resets	56
Standard Instructions	267
STATUS Register	66
SUBFSR	313
SUBFWB	302
SUBLW	303
SUBLNK	313
SUBWF	303
SUBWFB	304
SWAPP	304
<b>T</b>	
Table Reads/Table Writes	56
TBLRD	305
TBLWT	306
Time-out in Various Situations (table)	45
Timer0	123
Associated Registers	125
Operation	124
Overflow Interrupt	125
Prescaler	125
Prescaler Assignment (PSA Bit)	125
Prescaler Select (T0PS2:T0PS0 Bits)	125
Prescaler. See Prescaler, Timer0.	
Reads and Writes in 16-Bit Mode	124
Source Edge Select (T0SE Bit)	124
Source Select (T0CS Bit)	124
Switching Prescaler Assignment	125
Timer1	127
16-Bit Read/Write Mode	129
Associated Registers	131
Interrupt	130
Operation	128
Oscillator	127, 129
Layout Considerations	130
Low-Power Option	129
Overflow Interrupt	127
Resetting, Using the CCP Special Event Trigger	130
Special Event Trigger (ECCP)	148
TMR1H Register	127
TMR1L Register	127
Use as a Real-Time Clock	130
Timer2	133
Associated Registers	134
Interrupt	134
Operation	133
Output	134
PR2 Register	144, 149
TMR2 to PR2 Match Interrupt	144, 149
Timer3	135
16-Bit Read/Write Mode	137
Associated Registers	137
Operation	136
Oscillator	135, 137
Overflow Interrupt	135, 137
Special Event Trigger (CCP)	137
TMR3H Register	135
TMR3L Register	135
Timing Diagrams	
A/D Conversion	360
Acknowledge Sequence	194
Asynchronous Reception	214
Asynchronous Transmission	212
Asynchronous Transmission (Back to Back)	212
Automatic Baud Rate Calculation	210
Auto-Wake-up Bit (WUE) During Normal Operation	215
Auto-Wake-up Bit (WUE) During Sleep	215
Baud Rate Generator with Clock Arbitration	188
BRG Overflow Sequence	210
BRG Reset Due to SDA Arbitration During Start Condition	197
Brown-out Reset (BOR)	345
Bus Collision During a Repeated Start Condition (Case 1)	198
Bus Collision During a Repeated Start Condition (Case 2)	198
Bus Collision During a Start Condition (SCL = 0)	197
Bus Collision During a Stop Condition (Case 1)	199
Bus Collision During a Stop Condition (Case 2)	199
Bus Collision During Start Condition (SDA Only)	196
Bus Collision for Transmit and Acknowledge	195
Capture/Compare/PWM (All CCP Modules)	347
CLKO and I/O	344
Clock Synchronization	181
Clock/Instruction Cycle	57
EUSART Synchronous Receive (Master/Slave)	359
EUSART Synchronous Transmission (Master/Slave)	358
Example SPI Master Mode (CKE = 0)	349
Example SPI Master Mode (CKE = 1)	350
Example SPI Slave Mode (CKE = 0)	351
Example SPI Slave Mode (CKE = 1)	353
External Clock (All Modes Except PLL)	342
Fail-Safe Clock Monitor	262

First Start Bit Timing	189	Transition to RC_RUN Mode	36
Full-Bridge PWM Output	153	Timing Diagrams and Specifications	342
Half-Bridge PWM Output	152	A/D Conversion Requirements	360
High/Low-Voltage Detect Characteristics	339	Capture/Compare/PWM (CCP) Requirements	347
High-Voltage Detect Operation (VDIRMAG = 1)	246	CLKO and I/O Requirements	344
I <sup>2</sup> C Bus Data	354	EUSART Synchronous Receive Requirements	359
I <sup>2</sup> C Bus Start/Stop Bits	354	EUSART Synchronous Transmission	
I <sup>2</sup> C Master Mode (7 or 10-Bit Transmission)	192	Requirements	358
I <sup>2</sup> C Master Mode (7-Bit Reception)	193	Example SPI Mode Requirements	
I <sup>2</sup> C Slave Mode (10-Bit Reception, SEN = 0)	178	(Master Mode, CKE = 0)	349
I <sup>2</sup> C Slave Mode (10-Bit Reception, SEN = 1)	183	Example SPI Mode Requirements	
I <sup>2</sup> C Slave Mode (10-Bit Transmission)	179	(Master Mode, CKE = 1)	350
I <sup>2</sup> C Slave Mode (7-Bit Reception, SEN = 0)	176	Example SPI Mode Requirements	
I <sup>2</sup> C Slave Mode (7-Bit Reception, SEN = 1)	182	(Slave Mode, CKE = 0)	352
I <sup>2</sup> C Slave Mode (7-Bit Transmission)	177	Example SPI Mode Requirements	
I <sup>2</sup> C Slave Mode General Call Address		(Slave Mode, CKE = 1)	353
Sequence (7 or 10-Bit Address Mode)	184	External Clock Requirements	342
I <sup>2</sup> C Stop Condition Receive or Transmit Mode	194	I <sup>2</sup> C Bus Data Requirements (Slave Mode)	355
Low-Voltage Detect Operation (VDIRMAG = 0)	245	Master SSP I <sup>2</sup> C Bus Data Requirements	357
Master SSP I <sup>2</sup> C Bus Data	356	Master SSP I <sup>2</sup> C Bus Start/Stop Bits	
Master SSP I <sup>2</sup> C Bus Start/Stop Bits	356	Requirements	356
Parallel Slave Port (PIC18F4525/4620)	348	Parallel Slave Port Requirements	
Parallel Slave Port (PSP) Read	107	(PIC18F4525/4620)	348
Parallel Slave Port (PSP) Write	107	PLL Clock	343
PWM Auto-Shutdown (PRSEN = 0,		Reset, Watchdog Timer, Oscillator Start-up	
Auto-Restart Disabled)	158	Timer, Power-up Timer and Brown-out	
PWM Auto-Shutdown (PRSEN = 1,		Reset Requirements	345
Auto-Restart Enabled)	158	Timer0 and Timer1 External Clock	
PWM Direction Change	155	Requirements	346
PWM Direction Change at Near		Top-of-Stack Access	54
100% Duty Cycle	155	TRISE Register	
PWM Output	144	PSPMODE Bit	100
Repeat Start Condition	190	TSTFSZ	307
Reset, Watchdog Timer (WDT), Oscillator Start-up		Two-Speed Start-up	249, 260
Timer (OST), Power-up Timer (PWRT)	345	Two-Word Instructions	
Send Break Character Sequence	216	Example Cases	58
Slave Synchronization	167	TXSTA Register	
Slow Rise Time (MCLR Tied to VDD,		BRGH Bit	205
VDD Rise > TPWRT)	47	<b>V</b>	
SPI Mode (Master Mode)	166	Voltage Reference Specifications	338
SPI Mode (Slave Mode, CKE = 0)	168	<b>W</b>	
SPI Mode (Slave Mode, CKE = 1)	168	Watchdog Timer (WDT)	249, 258
Synchronous Reception (Master Mode, SREN)	219	Associated Registers	259
Synchronous Transmission	217	Control Register	258
Synchronous Transmission (Through TXEN)	218	During Oscillator Failure	261
Time-out Sequence on POR w/PLL Enabled		Programming Considerations	258
(MCLR Tied to VDD)	47	WCOL	189, 190, 191, 194
Time-out Sequence on Power-up		WCOL Status Flag	189, 190, 191, 194
(MCLR Not Tied to VDD, Case 1)	46	WWW Address	407
Time-out Sequence on Power-up		WWW, On-Line Support	5
(MCLR Not Tied to VDD, Case 2)	46	<b>X</b>	
Time-out Sequence on Power-up		XORLW	307
(MCLR Tied to VDD, VDD Rise < TPWRT)	46	XORWF	308
Timer0 and Timer1 External Clock	346		
Transition for Entry to Idle Mode	38		
Transition for Entry to SEC_RUN Mode	35		
Transition for Entry to Sleep Mode	37		
Transition for Two-Speed Start-up			
(INTOSC to HSPLL)	260		
Transition for Wake from Idle to Run Mode	38		
Transition for Wake from Sleep (HSPLL)	37		
Transition from RC_RUN Mode to			
PRI_RUN Mode	36		
Transition from SEC_RUN Mode to			
PRI_RUN Mode (HSPLL)	35		

# **PIC18F2525/2620/4525/4620**

---

---

## **NOTES:**

## 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), 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>**

# PIC18F2525/2620/4525/4620

---

---

## 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: PIC18F2525/2620/4525/4620

Literature Number: DS39626E

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?

---

# PIC18F2525/2620/4525/4620

## PIC18F2525/2620/4525/4620 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>			
Device	X	XX	XXX
	Temperature Range	Package	Pattern
Device			
		PIC18F2525/2620 <sup>(1)</sup> , PIC18F4525/4620 <sup>(1)</sup> , PIC18F2525/2620T <sup>(2)</sup> , PIC18F4525/4620T <sup>(2)</sup> , VDD range 4.2V to 5.5V PIC18LF2525/2620 <sup>(1)</sup> , PIC18LF4525/4620 <sup>(1)</sup> , PIC18LF2525/2620T <sup>(2)</sup> , PIC18LF4525/4620T <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)	
	SO	= SOIC	
	SP	= Skinny Plastic DIP	
	P	= PDIP	
	ML	= QFN	
Pattern	QTP, SQTP, Code or Special Requirements (blank otherwise)		

### Examples:

- a) PIC18LF4620-I/P 301 = Industrial temp., PDIP package, Extended VDD limits, QTP pattern #301.
- b) PIC18LF2620-I/SO = Industrial temp., SOIC package, Extended VDD limits.
- c) PIC18F4620-I/P = Industrial temp., PDIP package, normal VDD limits.

**Note 1:** F = Standard Voltage Range  
LF = Wide Voltage Range  
**2:** T = in tape and reel TQFP packages only.



# 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

**Dallas**

Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**

Farmington Hills, MI  
Tel: 248-538-2250  
Fax: 248-538-2260

**Kokomo**

Kokomo, IN  
Tel: 765-864-8360  
Fax: 765-864-8387

**Los Angeles**

Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**Santa Clara**

Santa Clara, CA  
Tel: 408-961-6444  
Fax: 408-961-6445

**Toronto**

Mississauga, Ontario,  
Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8528-2100  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Hong Kong SAR**  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Nanjing**  
Tel: 86-25-8473-2460  
Fax: 86-25-8473-2470

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8203-2660  
Fax: 86-755-8203-1760

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-4182-8400  
Fax: 91-80-4182-8422

**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: 886-3-572-9526  
Fax: 886-3-572-6459

**Taiwan - Kaohsiung**  
Tel: 886-7-536-4818  
Fax: 886-7-536-4803

**Taiwan - Taipei**  
Tel: 886-2-2500-6610  
Fax: 886-2-2508-0102

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820