



---

# **PIC18FXX2**

## **Data Sheet**

**High Performance, Enhanced FLASH  
Microcontrollers with 10-Bit A/D**

---

---

**Note the following details of the code protection feature on PICmicro® MCUs.**

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products 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 PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be 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 product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

---

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip's products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

#### Trademarks

The Microchip name and logo, the Microchip logo, KEELOQ, MPLAB, PIC, PICmicro, PICSTART and PRO MATE are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, microID, MXDEV, MXLAB, PICMASTER, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

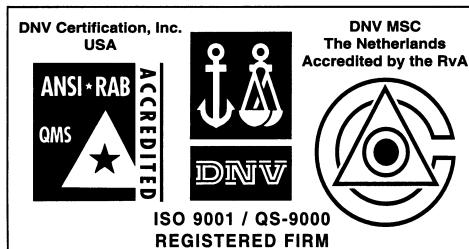
dsPIC, dsPICDEM.net, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICC, PICDEM, PICDEM.net, rfPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (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.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMS, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

## 28/40-pin High Performance, Enhanced FLASH Microcontrollers with 10-Bit A/D

### High Performance RISC CPU:

- C compiler optimized architecture/instruction set
  - Source code compatible with the PIC16 and PIC17 instruction sets
- Linear program memory addressing to 32 Kbytes
- Linear data memory addressing to 1.5 Kbytes

Device	On-Chip Program Memory		On-Chip RAM (bytes)	Data EEPROM (bytes)
	FLASH (bytes)	# Single Word Instructions		
PIC18F242	16K	8192	768	256
PIC18F252	32K	16384	1536	256
PIC18F442	16K	8192	768	256
PIC18F452	32K	16384	1536	256

- Up to 10 MIPS operation:
  - DC - 40 MHz osc./clock input
  - 4 MHz - 10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single Cycle Hardware Multiplier

### Peripheral Features:

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option - Timer1/Timer3
- Two Capture/Compare/PWM (CCP) modules. CCP pins that can be configured as:
  - Capture input: capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1- to 10-bit, max. PWM freq. @: 8-bit resolution = 156 kHz  
10-bit resolution = 39 kHz
- Master Synchronous Serial Port (MSSP) module, Two modes of operation:
  - 3-wire SPI™ (supports all 4 SPI modes)
  - I<sup>2</sup>C™ Master and Slave mode

### Peripheral Features (Continued):

- Addressable USART module:
  - Supports RS-485 and RS-232
- Parallel Slave Port (PSP) module

### Analog Features:

- Compatible 10-bit Analog-to-Digital Converter module (A/D) with:
  - Fast sampling rate
  - Conversion available during SLEEP
  - Linearity  $\leq 1$  LSb
- Programmable Low Voltage Detection (PLVD)
  - Supports interrupt on-Low Voltage Detection
- Programmable Brown-out Reset (BOR)

### Special Microcontroller Features:

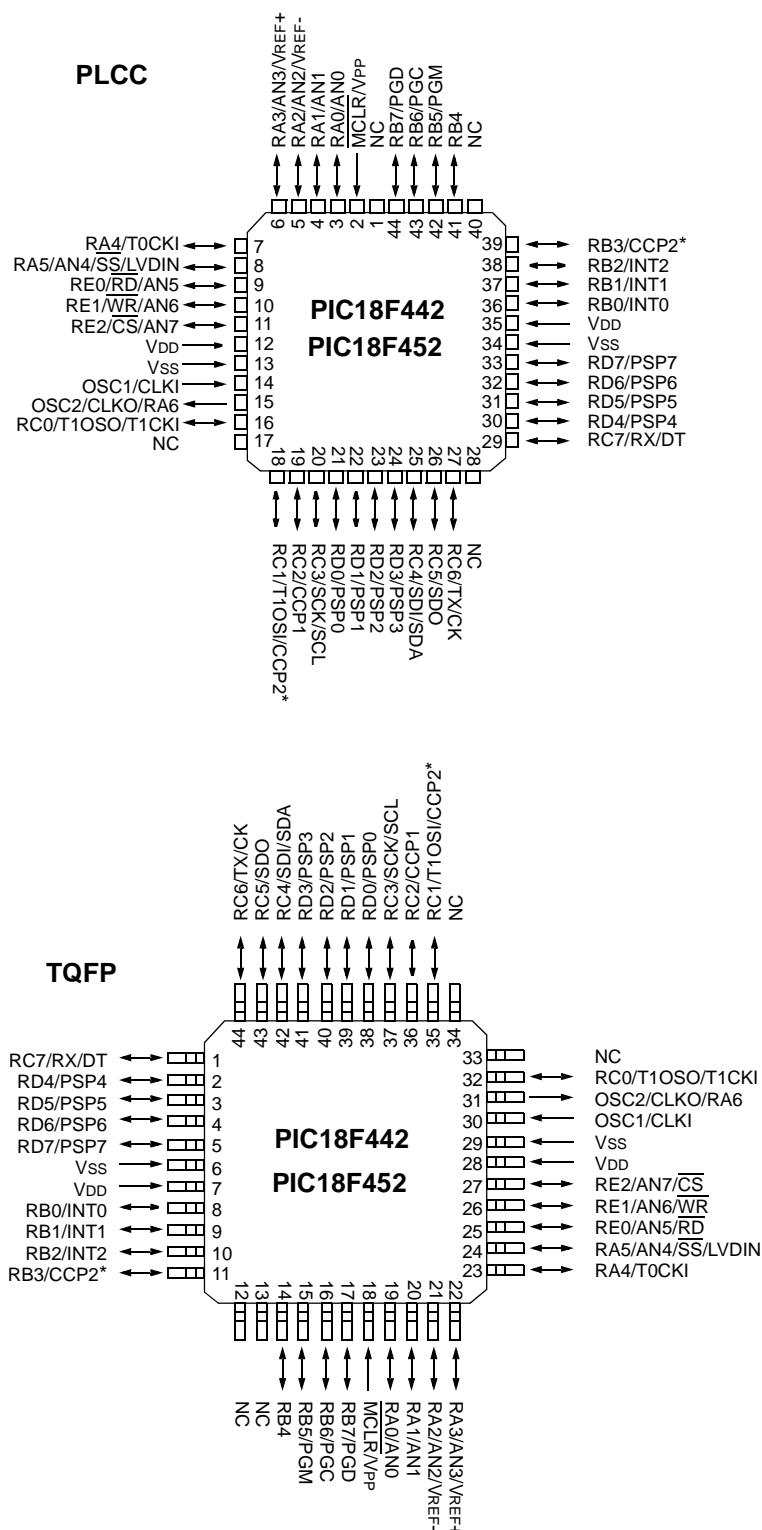
- 100,000 erase/write cycle Enhanced FLASH program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory
- FLASH/Data EEPROM Retention: > 40 years
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options including:
  - 4X Phase Lock Loop (of primary oscillator)
  - Secondary Oscillator (32 kHz) clock input
- Single supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins

### CMOS Technology:

- Low power, high speed FLASH/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges
- Low power consumption:
  - < 1.6 mA typical @ 5V, 4 MHz
  - 25 µA typical @ 3V, 32 kHz
  - < 0.2 µA typical standby current

# PIC18FXX2

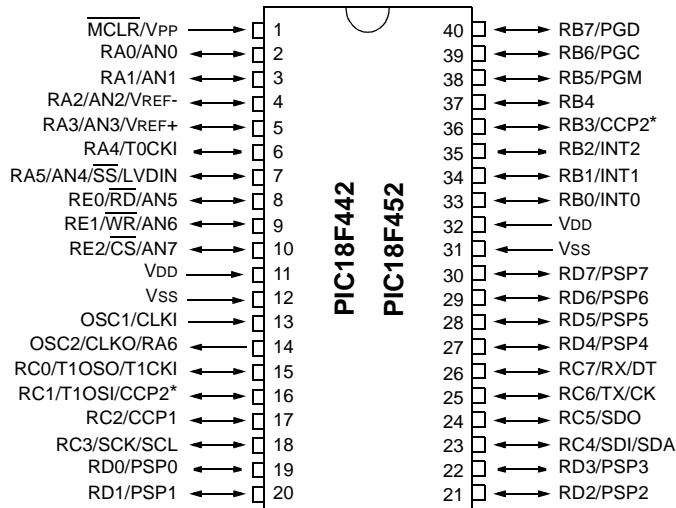
## Pin Diagrams



\* RB3 is the alternate pin for the CCP2 pin multiplexing.

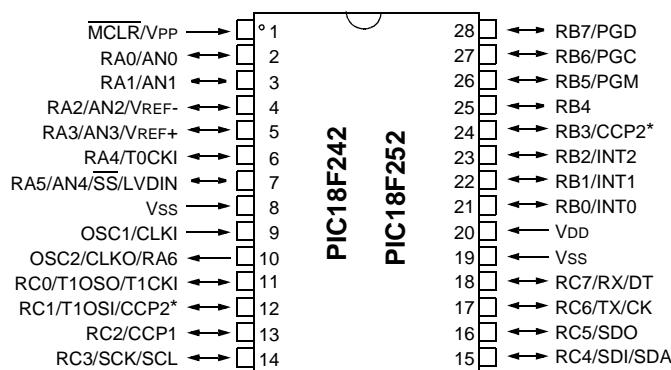
## Pin Diagrams (Cont.'d)

### DIP



**Note:** Pin compatible with 40-pin PIC16C7X devices.

### DIP, SOIC



\* RB3 is the alternate pin for the CCP2 pin multiplexing.

# PIC18FXX2

---

## Table of Contents

1.0	Device Overview .....	7
2.0	Oscillator Configurations .....	17
3.0	Reset .....	25
4.0	Memory Organization .....	35
5.0	FLASH Program Memory .....	55
6.0	Data EEPROM Memory .....	65
7.0	8 X 8 Hardware Multiplier .....	71
8.0	Interrupts .....	73
9.0	I/O Ports .....	87
10.0	Timer0 Module .....	103
11.0	Timer1 Module .....	107
12.0	Timer2 Module .....	111
13.0	Timer3 Module .....	113
14.0	Capture/Compare/PWM (CCP) Modules .....	117
15.0	Master Synchronous Serial Port (MSSP) Module .....	125
16.0	Addressable Universal Synchronous Asynchronous Receiver Transmitter (USART) .....	165
17.0	Compatible 10-bit Analog-to-Digital Converter (A/D) Module .....	181
18.0	Low Voltage Detect .....	189
19.0	Special Features of the CPU .....	195
20.0	Instruction Set Summary .....	211
21.0	Development Support .....	253
22.0	Electrical Characteristics .....	259
23.0	DC and AC Characteristics Graphs and Tables .....	289
24.0	Packaging Information .....	305
	Appendix A: Revision History .....	313
	Appendix B: Device Differences .....	313
	Appendix C: Conversion Considerations .....	314
	Appendix D: Migration from Baseline to Enhanced Devices .....	314
	Appendix E: Migration from Mid-range to Enhanced Devices .....	315
	Appendix F: Migration from High-end to Enhanced Devices .....	315
	Index .....	317
	On-Line Support .....	327
	Reader Response .....	328
	PIC18FXX2 Product Identification System .....	329

## TO OUR VALUED CUSTOMERS

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

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at [docerrors@mail.microchip.com](mailto:docerrors@mail.microchip.com) or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

### Most Current Data Sheet

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

<http://www.microchip.com>

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

### Errata

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

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

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

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

### Customer Notification System

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

# **PIC18FXX2**

---

---

## **NOTES:**

## 1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F242
- PIC18F442
- PIC18F252
- PIC18F452

These devices come in 28-pin and 40/44-pin packages. The 28-pin devices do not have a Parallel Slave Port (PSP) implemented and the number of Analog-to-Digital (A/D) converter input channels is reduced to 5. An overview of features is shown in Table 1-1.

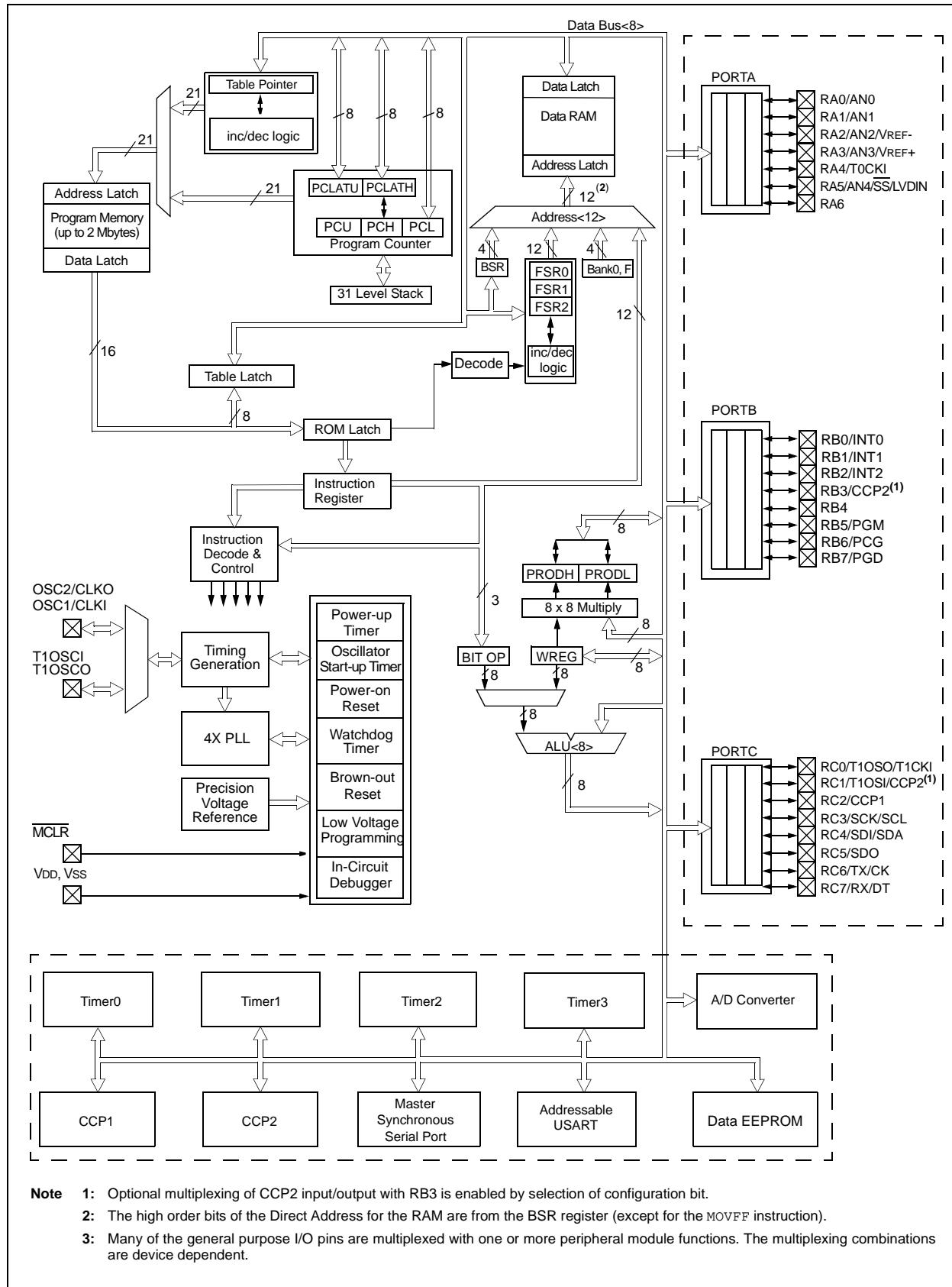
The following two figures are device block diagrams sorted by pin count: 28-pin for Figure 1-1 and 40/44-pin for Figure 1-2. The 28-pin and 40/44-pin pinouts are listed in Table 1-2 and Table 1-3, respectively.

**TABLE 1-1: DEVICE FEATURES**

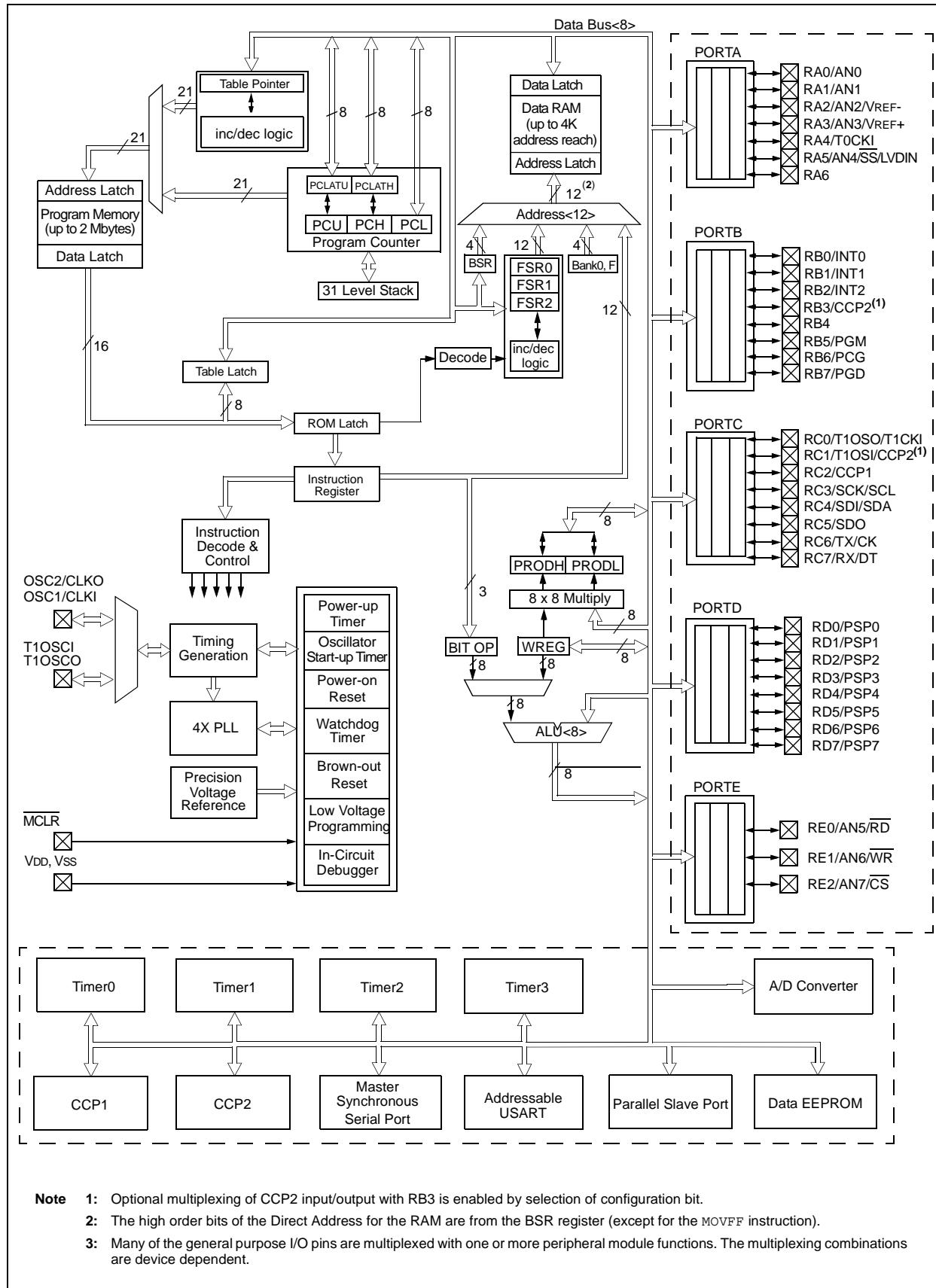
Features	PIC18F242	PIC18F252	PIC18F442	PIC18F452
Operating Frequency	DC - 40 MHz			
Program Memory (Bytes)	16K	32K	16K	32K
Program Memory (Instructions)	8192	16384	8192	16384
Data Memory (Bytes)	768	1536	768	1536
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	17	17	18	18
I/O Ports	Ports A, B, C	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, Addressable USART	MSSP, Addressable USART	MSSP, Addressable USART	MSSP, Addressable USART
Parallel Communications	—	—	PSP	PSP
10-bit Analog-to-Digital Module	5 input channels	5 input channels	8 input channels	8 input channels
RESETS (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)			
Programmable Low Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions	75 Instructions	75 Instructions	75 Instructions
Packages	28-pin DIP 28-pin SOIC	28-pin DIP 28-pin SOIC	40-pin DIP 44-pin PLCC 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin TQFP

# PIC18FXX2

**FIGURE 1-1: PIC18F2X2 BLOCK DIAGRAM**



**FIGURE 1-2: PIC18F4X2 BLOCK DIAGRAM**



# PIC18FXX2

---

**TABLE 1-2: PIC18F2X2 PINOUT I/O DESCRIPTIONS**

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	DIP	SOIC			
MCLR/VPP  MCLR  VPP	1	1	I	ST	Master Clear (input) or high voltage ICSP programming enable pin.  Master Clear (Reset) input. This pin is an active low RESET to the device.  High voltage ICSP programming enable pin.
NC	—	—	—	—	These pins should be left unconnected.
OSC1/CLKI OSC1  CLKI	9	9	I	ST CMOS	Oscillator crystal or external clock input.  Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise. External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
OSC2/CLKO/RA6 OSC2  CLKO  RA6	10	10	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 CLK0 which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.  General Purpose I/O pin.
RA0/AN0 RA0 AN0  RA1/AN1 RA1 AN1  RA2/AN2/VREF- RA2 AN2 VREF-  RA3/AN3/VREF+ RA3 AN3 VREF+  RA4/T0CKI RA4 T0CKI  RA5/AN4/SS/LVDIN RA5 AN4 SS LVDIN  RA6	2  3  4  5  6  7  	2  3  4  5  6  7	I/O I I/O I I/O I I/O I I/O I I/O I	TTL Analog  TTL Analog  TTL Analog Analog  TTL Analog Analog  ST/OD ST  TTL Analog ST Analog	PORTA is a bi-directional I/O port.  Digital I/O. Analog input 0.  Digital I/O. Analog input 1.  Digital I/O. Analog input 2. A/D Reference Voltage (Low) input.  Digital I/O. Analog input 3. A/D Reference Voltage (High) input.  Digital I/O. Open drain when configured as output. Timer0 external clock input.  Digital I/O. Analog input 4. SPI Slave Select input. Low Voltage Detect Input.  See the OSC2/CLKO/RA6 pin.

Legend: TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

OD = Open Drain (no P diode to VDD)

CMOS = CMOS compatible input or output

I = Input

P = Power

**TABLE 1-2: PIC18F2X2 PINOUT I/O DESCRIPTIONS (CONTINUED)**

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	DIP	SOIC			
RB0/INT0 RB0 INT0	21	21	I/O I	TTL ST	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.  Digital I/O. External Interrupt 0.
RB1/INT1 RB1 INT1	22	22	I/O I	TTL ST	External Interrupt 1.
RB2/INT2 RB2 INT2	23	23	I/O I	TTL ST	Digital I/O. External Interrupt 2.
RB3/CCP2 RB3 CCP2	24	24	I/O I/O	TTL ST	Digital I/O. Capture2 input, Compare2 output, PWM2 output.
RB4	25	25	I/O	TTL	Digital I/O. Interrupt-on-change pin.
RB5/PGM RB5 PGM	26	26	I/O I/O	TTL ST	Digital I/O. Interrupt-on-change pin. Low Voltage ICSP programming enable pin.
RB6/PGC RB6 PGC	27	27	I/O I/O	TTL ST	Digital I/O. Interrupt-on-change pin. In-Circuit Debugger and ICSP programming clock pin.
RB7/PGD RB7 PGD	28	28	I/O I/O	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  
 OD = Open Drain (no P diode to VDD)

CMOS = CMOS compatible input or output  
 I = Input  
 P = Power

# PIC18FXX2

---

**TABLE 1-2: PIC18F2X2 PINOUT I/O DESCRIPTIONS (CONTINUED)**

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	DIP	SOIC			
RC0/T1OSO/T1CKI RC0 T1OSO T1CKI	11	11	I/O O I	ST — ST	PORTC is a bi-directional I/O port.  Digital I/O. Timer1 oscillator output. Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2 RC1 T1OSI CCP2	12	12	I/O I I/O	ST CMOS ST	Digital I/O. Timer1 oscillator input. Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1 RC2 CCP1	13	13	I/O I/O	ST ST	Digital I/O. Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL RC3 SCK SCL	14	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 mode
RC4/SDI/SDA RC4 SDI SDA	15	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	16	I/O O	ST —	Digital I/O. SPI Data Out.
RC6/TX/CK RC6 TX CK	17	17	I/O O I/O	ST — ST	Digital I/O. USART Asynchronous Transmit. USART Synchronous Clock (see related RX/DT).
RC7/RX/DT RC7 RX DT	18	18	I/O I I/O	ST ST ST	Digital I/O. USART Asynchronous Receive. USART Synchronous Data (see related TX/CK).
Vss	8, 19	8, 19	P	—	Ground reference for logic and I/O pins.
Vdd	20	20	P	—	Positive supply for logic and I/O pins.

Legend: TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

OD = Open Drain (no P diode to VDD)

CMOS = CMOS compatible input or output

I = Input

P = Power

**TABLE 1-3: PIC18F4X2 PINOUT I/O DESCRIPTIONS**

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	PLCC	TQFP			
MCLR/VPP  MCLR  VPP	1	2	18	I	ST	Master Clear (input) or high voltage ICSP programming enable pin.  Master Clear (Reset) input. This pin is an active low RESET to the device.  High voltage ICSP programming enable pin.
NC	—	—	—	—	—	These pins should be left unconnected.
OSC1/CLKI OSC1  CLKI	13	14	30	I	ST	Oscillator crystal or external clock input.  Oscillator crystal input or external clock source input. ST buffer when configured in RC mode, CMOS otherwise.
				I	CMOS	External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
OSC2/CLKO/RA6 OSC2  CLKO  RA6	14	15	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 CLK0, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.  General Purpose I/O pin.
RA0/AN0 RA0 AN0  RA1/AN1 RA1 AN1  RA2/AN2/VREF- RA2 AN2 VREF-  RA3/AN3/VREF+ RA3 AN3 VREF+  RA4/T0CKI RA4 T0CKI  RA5/AN4/SS/LVDIN RA5 AN4 SS LVDIN  RA6	2  3  4  5  6  7  8	3  4  5  6  7  8	19  20  21  22  23  24	I/O  I  I/O  I  I/O  I  I  I/O  I  I/O  I  I	TTL  Analog  TTL  Analog  TTL  Analog  TTL  Analog  ST/OD  TTL  Analog  TTL  Analog  TTL  Analog	PORTA is a bi-directional I/O port.  Digital I/O. Analog input 0.  Digital I/O. Analog input 1.  Digital I/O. Analog input 2. A/D Reference Voltage (Low) input.  Digital I/O. Analog input 3. A/D Reference Voltage (High) input.  Digital I/O. Open drain when configured as output. Timer0 external clock input.  Digital I/O. Analog input 4. SPI Slave Select input. Low Voltage Detect Input. (See the OSC2/CLKO/RA6 pin.)

Legend: TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

OD = Open Drain (no P diode to VDD)

CMOS = CMOS compatible input or output

I = Input

P = Power

# PIC18FXX2

---

TABLE 1-3: PIC18F4X2 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	PLCC	TQFP			
RB0/INT0 RB0 INT0	33	36	8	I/O I	TTL ST	PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.  Digital I/O. External Interrupt 0.
RB1/INT1 RB1 INT1	34	37	9	I/O I	TTL ST	External Interrupt 1.
RB2/INT2 RB2 INT2	35	38	10	I/O I	TTL ST	Digital I/O. External Interrupt 2.
RB3/CCP2 RB3 CCP2	36	39	11	I/O I/O	TTL ST	Digital I/O. Capture2 input, Compare2 output, PWM2 output.
RB4	37	41	14	I/O	TTL	Digital I/O. Interrupt-on-change pin.
RB5/PGM RB5 PGM	38	42	15	I/O I/O	TTL ST	Digital I/O. Interrupt-on-change pin. Low Voltage ICSP programming enable pin.
RB6/PGC RB6 PGC	39	43	16	I/O I/O	TTL ST	Digital I/O. Interrupt-on-change pin. In-Circuit Debugger and ICSP programming clock pin.
RB7/PGD RB7 PGD	40	44	17	I/O I/O	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

OD = Open Drain (no P diode to VDD)

CMOS = CMOS compatible input or output

I = Input

P = Power

TABLE 1-3: PIC18F4X2 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	PLCC	TQFP			
RC0/T1OSO/T1CKI RC0 T1OSO T1CKI	15	16	32	I/O O I	ST — ST	PORTC is a bi-directional I/O port.  Digital I/O. Timer1 oscillator output. Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2 RC1 T1OSI CCP2	16	18	35	I/O I I/O	ST CMOS ST	Digital I/O. Timer1 oscillator input. Capture2 input, Compare2 output, PWM2 output.
RC2/CCP1 RC2 CCP1	17	19	36	I/O I/O	ST ST	Digital I/O. Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL RC3 SCK  SCL	18	20	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 mode.
RC4/SDI/SDA RC4 SDI SDA	23	25	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	26	43	I/O O	ST —	Digital I/O. SPI Data Out.
RC6/TX/CK RC6 TX CK	25	27	44	I/O O I/O	ST — ST	Digital I/O. USART Asynchronous Transmit. USART Synchronous Clock (see related RX/DT).
RC7/RX/DT RC7 RX DT	26	29	1	I/O I I/O	ST ST ST	Digital I/O. USART Asynchronous Receive. USART Synchronous Data (see related TX/CK).

Legend: TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

OD = Open Drain (no P diode to VDD)

CMOS = CMOS compatible input or output

I = Input

P = Power

# PIC18FXX2

---

TABLE 1-3: PIC18F4X2 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	DIP	PLCC	TQFP			
RD0/PSP0	19	21	38	I/O	ST TTL	PORTD is a bi-directional I/O port, or a Parallel Slave Port (PSP) for interfacing to a microprocessor port. These pins have TTL input buffers when PSP module is enabled. Digital I/O. Parallel Slave Port Data.
RD1/PSP1	20	22	39	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD2/PSP2	21	23	40	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD3/PSP3	22	24	41	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD4/PSP4	27	30	2	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD5/PSP5	28	31	3	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD6/PSP6	29	32	4	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RD7/PSP7	30	33	5	I/O	ST TTL	Digital I/O. Parallel Slave Port Data.
RE0/RD/AN5  RE0 RD	8	9	25	I/O	ST TTL	PORTE is a bi-directional I/O port.  Digital I/O. Read control for parallel slave port (see also WR and CS pins).
AN5  RE1/WR/AN6  RE1 WR	9	10	26	I/O	Analog	Analog input 5.  Digital I/O. Write control for parallel slave port (see CS and RD pins).
AN6  RE2/CS/AN7  RE2 CS	10	11	27	I/O	ST TTL	Analog input 6.  Digital I/O. Chip Select control for parallel slave port (see related RD and WR).
AN7					Analog	Analog input 7.
VSS	12, 31	13, 34	6, 29	P	—	Ground reference for logic and I/O pins.
VDD	11, 32	12, 35	7, 28	P	—	Positive supply for logic and I/O pins.

Legend: TTL = TTL compatible input

ST = Schmitt Trigger input with CMOS levels

O = Output

OD = Open Drain (no P diode to VDD)

CMOS = CMOS compatible input or output

I = Input

P = Power

## 2.0 OSCILLATOR CONFIGURATIONS

### 2.1 Oscillator Types

The PIC18FXX2 can be operated in eight different Oscillator modes. The user can program three configuration bits (FOSC2, FOSC1, and FOSC0) to select one of these eight modes:

1. LP Low Power Crystal
2. XT Crystal/Resonator
3. HS High Speed Crystal/Resonator
4. HS + PLL High Speed Crystal/Resonator with PLL enabled
5. RC External Resistor/Capacitor
6. RCIO External Resistor/Capacitor with I/O pin enabled
7. EC External Clock
8. ECIO External Clock with I/O pin enabled

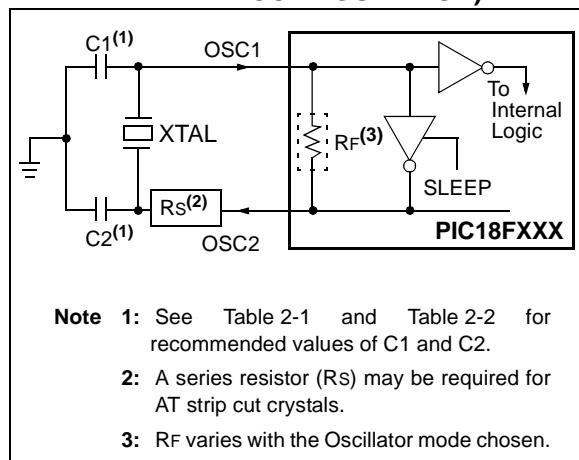
### 2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HS+PLL 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 PIC18FXX2 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 manufacturers specifications.

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



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

Ranges Tested:			
Mode	Freq	C1	C2
XT	455 kHz	68 - 100 pF	68 - 100 pF
	2.0 MHz	15 - 68 pF	15 - 68 pF
	4.0 MHz	15 - 68 pF	15 - 68 pF
HS	8.0 MHz	10 - 68 pF	10 - 68 pF
	16.0 MHz	10 - 22 pF	10 - 22 pF

**These values are for design guidance only.**  
See notes following this table.

Resonators Used:		
455 kHz	Panasonic EFO-A455K04B	$\pm 0.3\%$
2.0 MHz	Murata Erie CSA2.00MG	$\pm 0.5\%$
4.0 MHz	Murata Erie CSA4.00MG	$\pm 0.5\%$
8.0 MHz	Murata Erie CSA8.00MT	$\pm 0.5\%$
16.0 MHz	Murata Erie CSA16.00MX	$\pm 0.5\%$

All resonators used did not have built-in capacitors.

**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 high-gain HS mode, try a lower frequency resonator, 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, or verify oscillator performance.

# PIC18FXX2

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

Ranges Tested:			
Mode	Freq	C1	C2
LP	32.0 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	22-68 pF	22-68 pF
	1.0 MHz	15 pF	15 pF
	4.0 MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33 pF	15-33 pF
	20.0 MHz	15-33 pF	15-33 pF
	25.0 MHz	15-33 pF	15-33 pF

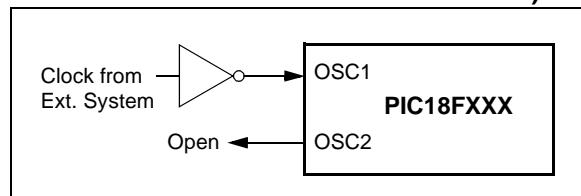
**These values are for design guidance only.**  
See notes following this table.

Crystals Used		
32.0 kHz	Epson C-001R32.768K-A	$\pm 20$ PPM
200 kHz	STD XTL 200.000KHz	$\pm 20$ PPM
1.0 MHz	ECS ECS-10-13-1	$\pm 50$ PPM
4.0 MHz	ECS ECS-40-20-1	$\pm 50$ PPM
8.0 MHz	Epson CA-301 8.000M-C	$\pm 30$ PPM
20.0 MHz	Epson CA-301 20.000M-C	$\pm 30$ PPM

- Note 1:** Higher capacitance increases the stability of the oscillator, but also increases the start-up time.
- 2:** Rs may be required in HS mode, as well as XT mode, to avoid overdriving crystals with low drive level specification.
- 3:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components., or verify oscillator performance.

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

**FIGURE 2-2: EXTERNAL CLOCK INPUT OPERATION (HS, XT OR LP OSC CONFIGURATION)**



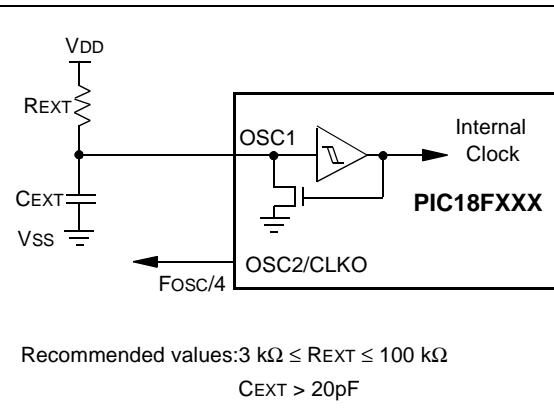
## 2.3 RC Oscillator

For timing-insensitive applications, the "RC" and "RCIO" device options offer additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor ( $R_{EXT}$ ) and capacitor ( $C_{EXT}$ ) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low  $C_{EXT}$  values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 2-3 shows how the R/C combination is connected.

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

**Note:** If the oscillator frequency divided by 4 signal is not required in the application, it is recommended to use RCIO mode to save current.

**FIGURE 2-3: RC OSCILLATOR MODE**



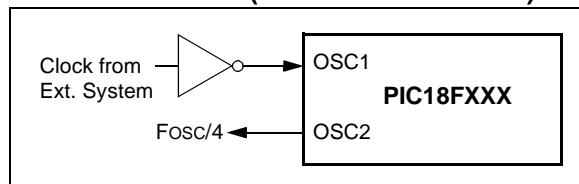
The RCIO Oscillator mode functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

## 2.4 External Clock Input

The EC and ECIO Oscillator modes require an external clock source to be connected to the OSC1 pin. The feedback device between OSC1 and OSC2 is turned off in these modes to save current. There is no oscillator start-up time required after a Power-on Reset or after a recovery from SLEEP mode.

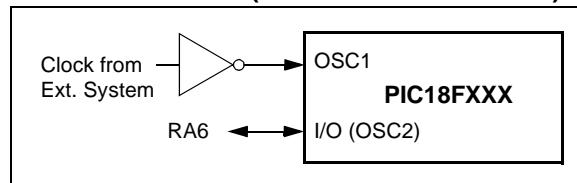
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-4 shows the pin connections for the EC Oscillator mode.

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



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

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



## 2.5 HS/PLL

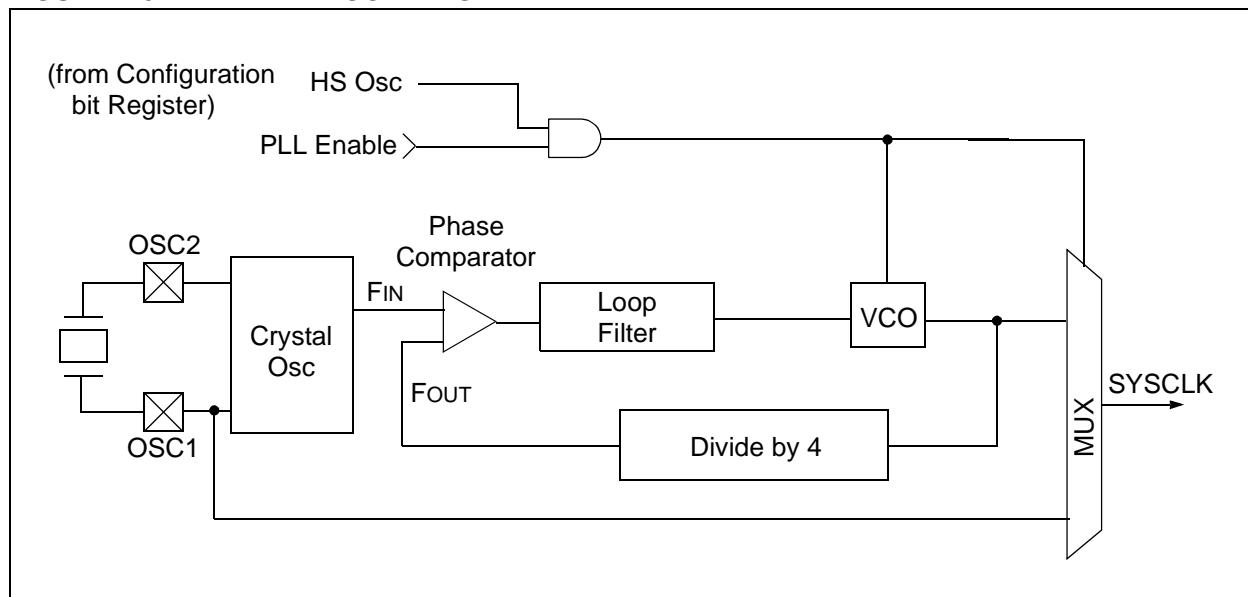
A Phase Locked Loop circuit is provided as a programmable option for users that want to multiply the frequency of the incoming crystal oscillator signal by 4. For an input clock frequency of 10 MHz, the internal clock frequency will be multiplied to 40 MHz. This is useful for customers who are concerned with EMI due to high frequency crystals.

The PLL can only be enabled when the oscillator configuration bits are programmed for HS mode. If they are programmed for any other mode, the PLL is not enabled and the system clock will come directly from OSC1.

The PLL is one of the modes of the FOSC<2:0> configuration bits. The Oscillator mode is specified during device programming.

A PLL lock timer is used to ensure that the PLL has locked before device execution starts. The PLL lock timer has a time-out that is called TPLL.

**FIGURE 2-6:** PLL BLOCK DIAGRAM

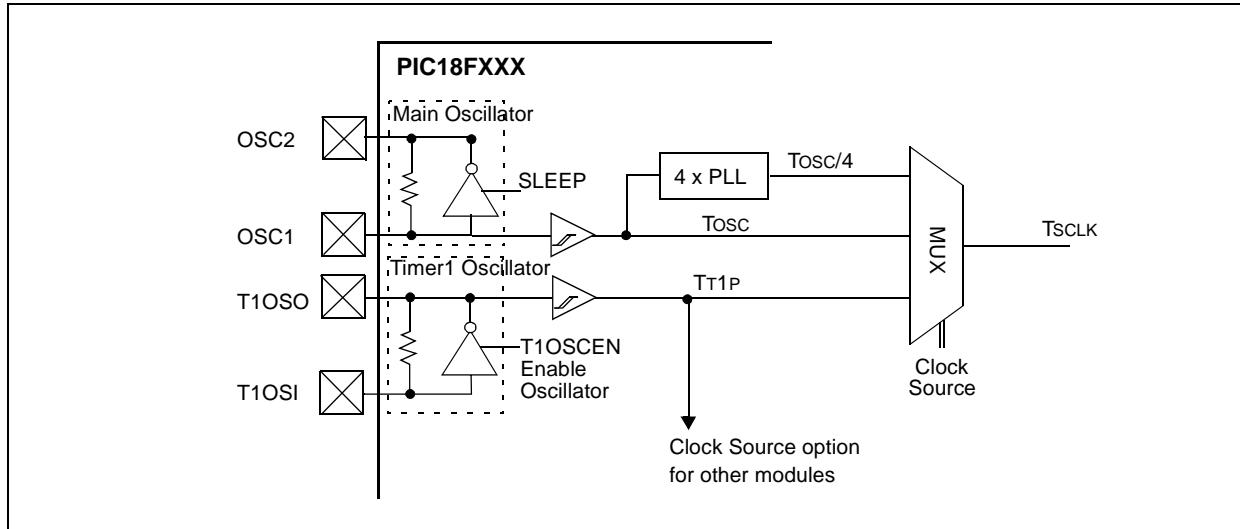


## 2.6 Oscillator Switching Feature

The PIC18FXX2 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low frequency clock source. For the PIC18FXX2 devices, this alternate clock source is the Timer1 oscillator. If a low frequency crystal (32 kHz, for example) has been attached to the Timer1 oscillator pins and the Timer1 oscillator has been enabled, the device can switch to a Low Power Execu-

tion mode. Figure 2-7 shows a block diagram of the system clock sources. The clock switching feature is enabled by programming the Oscillator Switching Enable (OSCSEN) bit in Configuration Register1H to a '0'. Clock switching is disabled in an erased device. See Section 11.0 for further details of the Timer1 oscillator. See Section 19.0 for Configuration Register details.

**FIGURE 2-7: DEVICE CLOCK SOURCES**



## 2.6.1 SYSTEM CLOCK SWITCH BIT

The system clock source switching is performed under software control. The system clock switch bit, SCS (OSCCON<0>) controls the clock switching. When the SCS bit is '0', the system clock source comes from the main oscillator that is selected by the FOSC configuration bits in Configuration Register1H. When the SCS bit is set, the system clock source will come from the Timer1 oscillator. The SCS bit is cleared on all forms of RESET.

**Note:** The Timer1 oscillator must be enabled and operating to switch the system clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 control register (T1CON). If the Timer1 oscillator is not enabled, then any write to the SCS bit will be ignored (SCS bit forced cleared) and the main oscillator will continue to be the system clock source.

## REGISTER 2-1: OSCCON REGISTER

U-0	R/W-1						
—	—	—	—	—	—	—	SCS bit 0

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **SCS:** System Clock Switch bit

When OSCSEN configuration bit = '0' and T1OSCEN bit is set:

1 = Switch to Timer1 oscillator/clock pin

0 = Use primary oscillator/clock input pin

When OSCSEN and T1OSCEN are in other states:

bit is forced clear

### 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

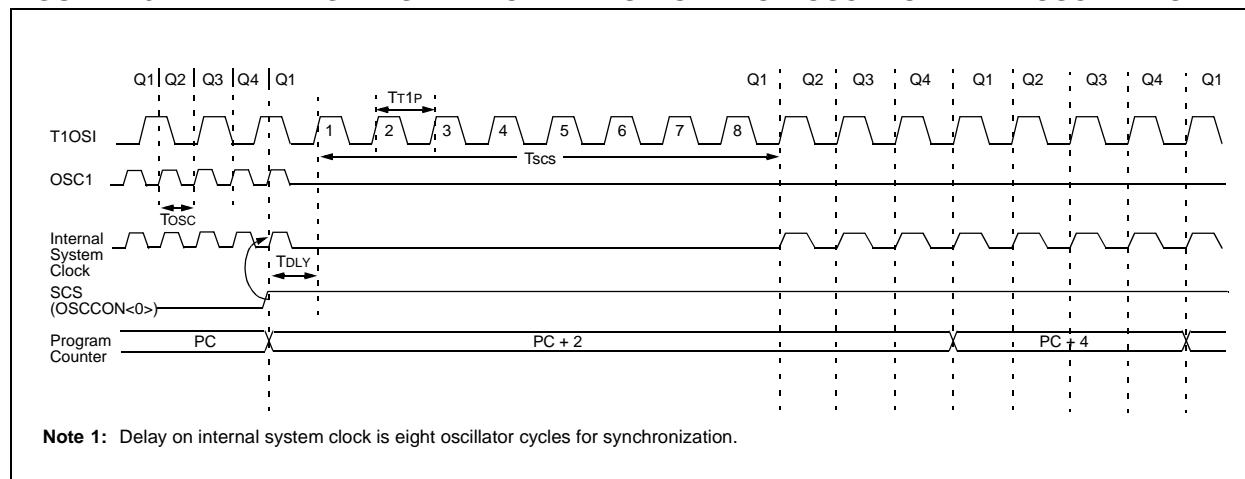
x = Bit is unknown

## 2.6.2 OSCILLATOR TRANSITIONS

The PIC18FXX2 devices contain circuitry to prevent "glitches" when switching between oscillator sources. Essentially, the circuitry waits for eight rising edges of the clock source that the processor is switching to. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

A timing diagram indicating the transition from the main oscillator to the Timer1 oscillator is shown in Figure 2-8. The Timer1 oscillator is assumed to be running all the time. After the SCS bit is set, the processor is frozen at the next occurring Q1 cycle. After eight synchronization cycles are counted from the Timer1 oscillator, operation resumes. No additional delays are required after the synchronization cycles.

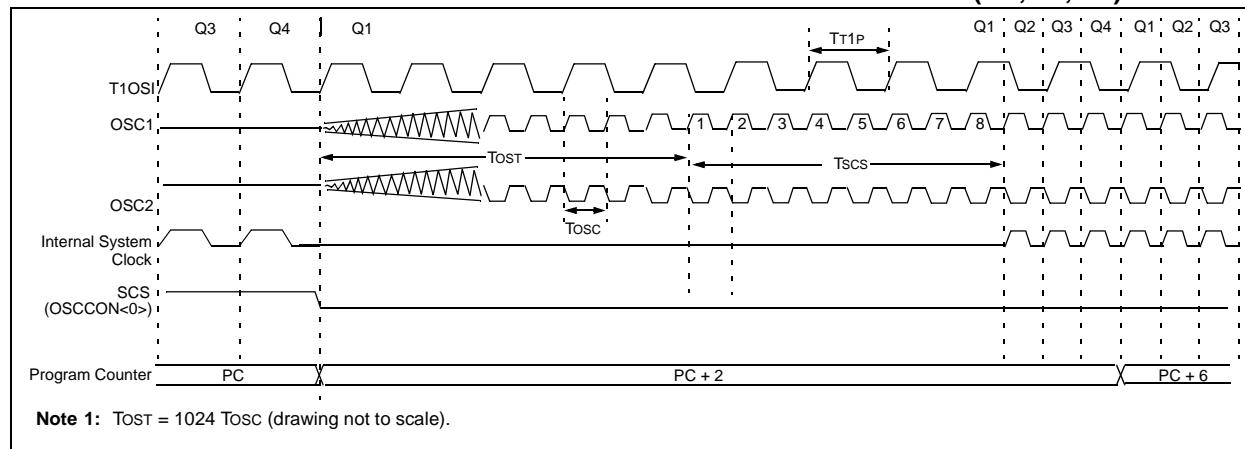
**FIGURE 2-8: TIMING DIAGRAM FOR TRANSITION FROM OSC1 TO TIMER1 OSCILLATOR**



The sequence of events that takes place when switching from the Timer1 oscillator to the main oscillator will depend on the mode of the main oscillator. In addition to eight clock cycles of the main oscillator, additional delays may take place.

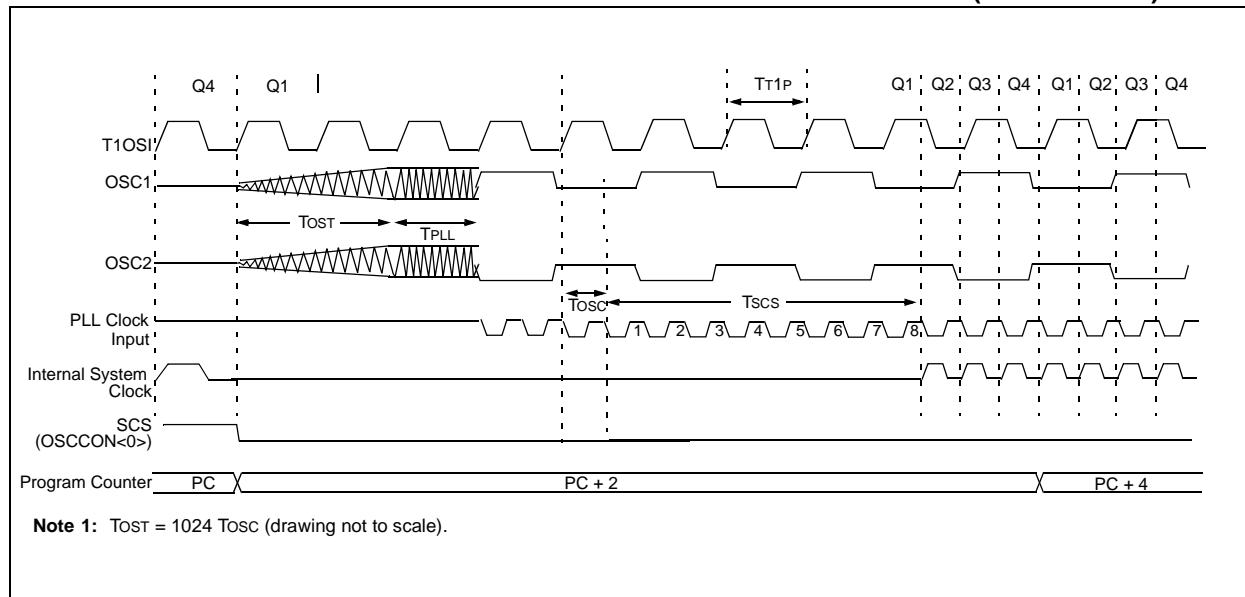
If the main oscillator is configured for an external crystal (HS, XT, LP), then the transition will take place after an oscillator start-up time ( $T_{OST}$ ) has occurred. A timing diagram, indicating the transition from the Timer1 oscillator to the main oscillator for HS, XT and LP modes, is shown in Figure 2-9.

**FIGURE 2-9: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS, XT, LP)**



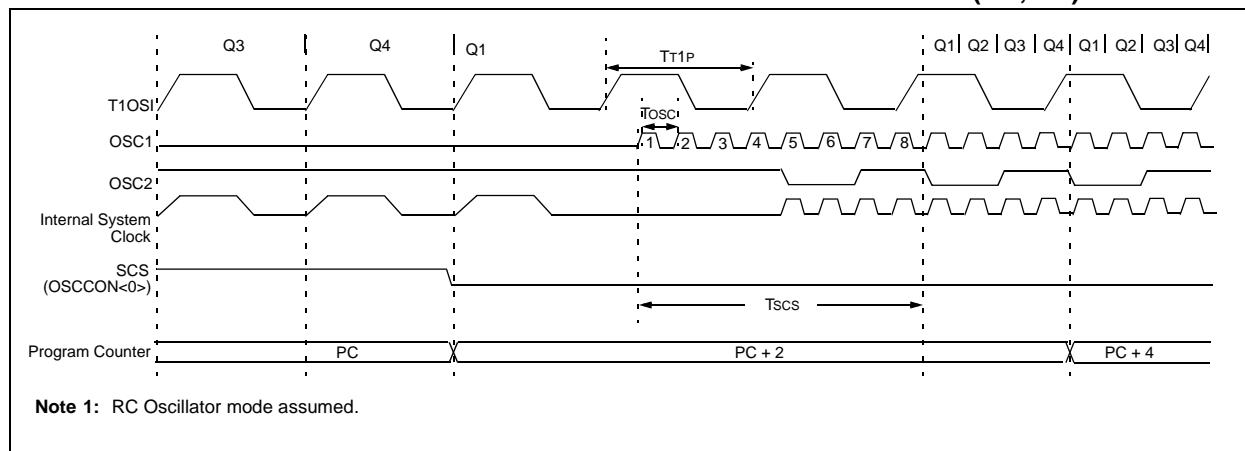
If the main oscillator is configured for HS-PLL mode, an oscillator start-up time (TOST) plus an additional PLL time-out (TPLL) will occur. The PLL time-out is typically 2 ms and allows the PLL to lock to the main oscillator frequency. A timing diagram indicating the transition from the Timer1 oscillator to the main oscillator for HS-PLL mode is shown in Figure 2-10.

**FIGURE 2-10: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS WITH PLL)**



If the main oscillator is configured in the RC, RCIO, EC or ECIO modes, there is no oscillator start-up time-out. Operation will resume after eight cycles of the main oscillator have been counted. A timing diagram, indicating the transition from the Timer1 oscillator to the main oscillator for RC, RCIO, EC and ECIO modes, is shown in Figure 2-11.

**FIGURE 2-11: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (RC, EC)**



## 2.7 Effects of SLEEP Mode on the On-Chip Oscillator

When the device executes a SLEEP instruction, the on-chip clocks and oscillator are turned off and the device is held at the beginning of an instruction cycle (Q1 state). With the oscillator off, the OSC1 and OSC2 signals will stop oscillating. Since all the transistor

switching currents have been removed, 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 user can wake from SLEEP through external RESET, Watchdog Timer Reset, or through an interrupt.

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

OSC Mode	OSC1 Pin	OSC2 Pin
RC	Floating, external resistor should pull high	At logic low
RCIO	Floating, external resistor should pull high	Configured as PORTA, bit 6
ECIO	Floating	Configured as PORTA, bit 6
EC	Floating	At logic low
LP, XT, and HS	Feedback inverter disabled, at quiescent voltage level	Feedback inverter disabled, at quiescent voltage level

**Note:** See Table 3-1, in the “Reset” section, for time-outs due to SLEEP and MCLR Reset.

## 2.8 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 and clock are stable. For additional information on RESET operation, see Section 3.0.

The first timer is the Power-up Timer (PWRT), which optionally provides a fixed delay of 72 ms (nominal) on power-up only (POR and BOR). The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable.

With the PLL enabled (HS/PLL Oscillator mode), the time-out sequence following a Power-on Reset is different from other Oscillator modes. The time-out sequence is as follows: First, the PWRT time-out is invoked after a POR time delay has expired. Then, the Oscillator Start-up Timer (OST) is invoked. However, this is still not a sufficient amount of time to allow the PLL to lock at high frequencies. The PWRT timer is used to provide an additional fixed 2 ms (nominal) time-out to allow the PLL ample time to lock to the incoming clock frequency.

### 3.0 RESET

The PIC18FXXX differentiates between various kinds of RESET:

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

Most registers are unaffected by a RESET. Their status is unknown on POR and unchanged by all other RESETS. The other registers are forced to a "RESET state" on Power-on Reset, MCLR, WDT Reset, Brown-out Reset, MCLR Reset during SLEEP and by the RESET instruction.

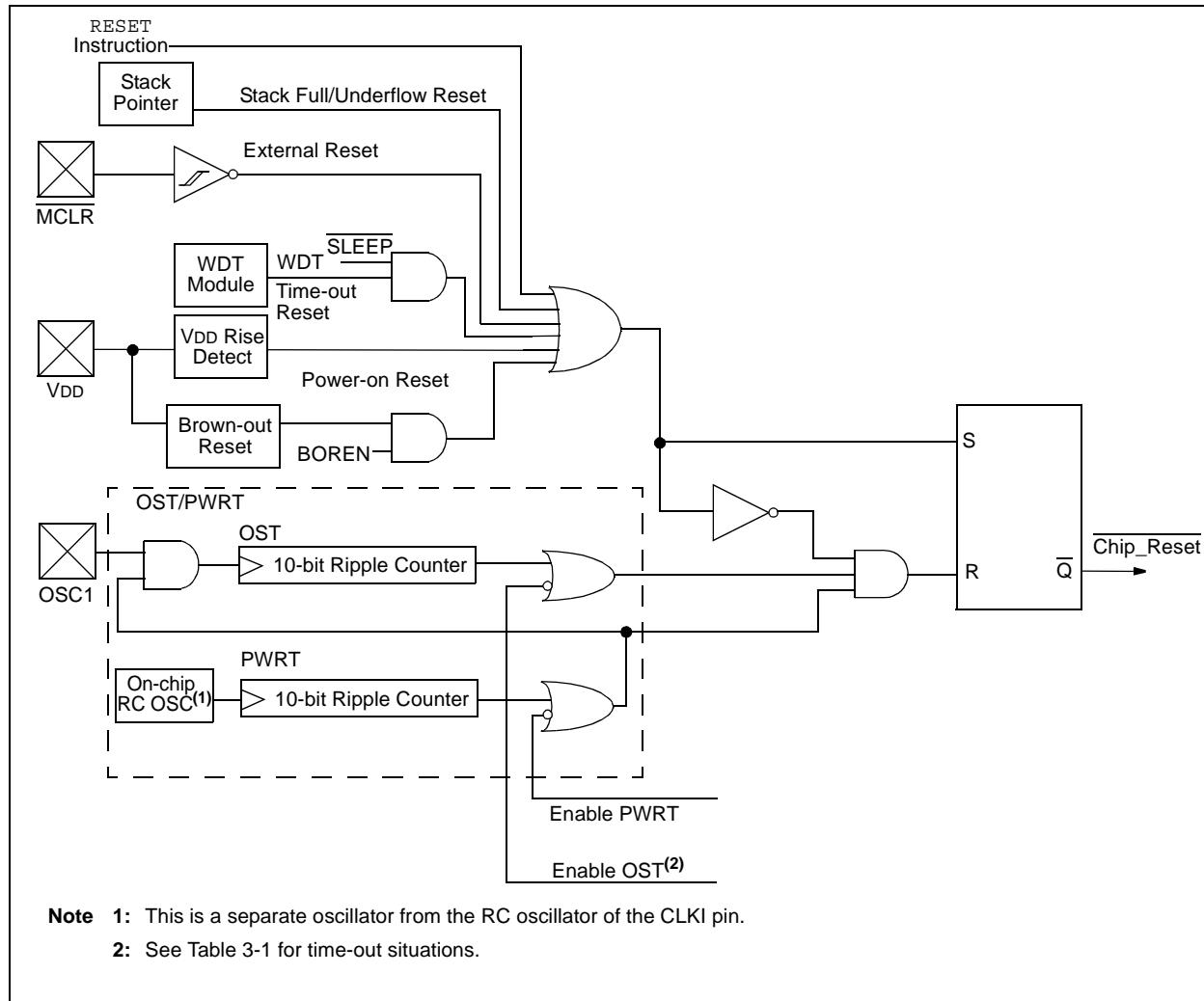
Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register, RI, TO, PD, POR and BOR, are set or cleared differently in different RESET situations, as indicated in Table 3-2. These bits are used in software to determine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

A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 3-1.

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

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

**FIGURE 3-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**



**Note 1:** This is a separate oscillator from the RC oscillator of the CLKI pin.

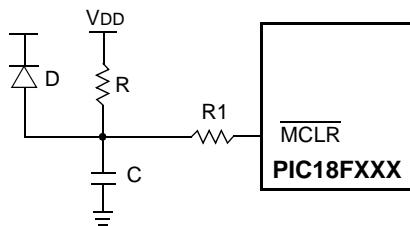
**2:** See Table 3-1 for time-out situations.

### 3.1 Power-On Reset (POR)

A Power-on Reset pulse is generated on-chip when VDD rise is detected. To take advantage of the POR circuitry, just tie the MCLR pin directly (or through a resistor) 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 3-2.

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

**FIGURE 3-2: 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:**  $R1 = 100\Omega$  to  $1\text{ k}\Omega$  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).

### 3.2 Power-up Timer (PWRT)

The Power-up Timer provides a fixed nominal time-out (parameter 33) only on power-up from the POR. The Power-up Timer operates on an internal RC oscillator. The chip is kept in RESET as long as the PWRT is active. The PWRT's time delay allows VDD to rise to an acceptable level. A configuration bit is provided to enable/disable the PWRT.

The power-up time delay will vary from chip-to-chip due to VDD, temperature and process variation. See DC parameter D033 for details.

### 3.3 Oscillator Start-up Timer (OST)

The Oscillator Start-up Timer (OST) provides a 1024 oscillator cycle (from OSC1 input) delay after the PWRT delay is over (parameter 32). This ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP and HS modes and only on Power-on Reset or wake-up from SLEEP.

### 3.4 PLL Lock Time-out

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

### 3.5 Brown-out Reset (BOR)

A configuration bit, BOREN, can disable (if clear/programmed), or enable (if set) the Brown-out Reset circuitry. If VDD falls below parameter D005 for greater than parameter 35, the brown-out situation will reset the chip. A RESET may not occur if VDD falls below parameter D005 for less than parameter 35. The chip will remain in Brown-out Reset until VDD rises above BVDD. If the Power-up Timer is enabled, it will be invoked after VDD rises above BVDD; it then will keep the chip in RESET for an additional time delay (parameter 33). If VDD drops below BVDD 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 BVDD, the Power-up Timer will execute the additional time delay.

### 3.6 Time-out Sequence

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

Since the time-outs occur from the POR pulse, if MCLR is kept low long enough, the time-outs will expire. Bringing MCLR high will begin execution immediately (Figure 3-5). This is useful for testing purposes or to synchronize more than one PIC18FXXX device operating in parallel.

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

**TABLE 3-1: TIME-OUT IN VARIOUS SITUATIONS**

Oscillator Configuration	Power-up <sup>(2)</sup>		Brown-out	Wake-up from SLEEP or Oscillator Switch
	PWRTE = 0	PWRTE = 1		
HS with PLL enabled <sup>(1)</sup>	72 ms + 1024 Tosc + 2ms	1024 Tosc + 2 ms	72 ms <sup>(2)</sup> + 1024 Tosc + 2 ms	1024 Tosc + 2 ms
HS, XT, LP	72 ms + 1024 Tosc	1024 Tosc	72 ms <sup>(2)</sup> + 1024 Tosc	1024 Tosc
EC	72 ms	—	72 ms <sup>(2)</sup>	—
External RC	72 ms	—	72 ms <sup>(2)</sup>	—

**Note 1:** 2 ms is the nominal time required for the 4x PLL to lock.

**2:** 72 ms is the nominal power-up timer delay, if implemented.

**REGISTER 3-1: RCON REGISTER BITS AND POSITIONS**

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	RI	TO	PD	POR	BOR
bit 7							bit 0

**Note 1:** Refer to Section 4.14 (page 53) for bit definitions.

**TABLE 3-2: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER**

Condition	Program Counter	RCON Register	RI	TO	PD	POR	BOR	STKFUL	STKUNF
Power-on Reset	0000h	0--1 1100	1	1	1	0	0	u	u
MCLR Reset during normal operation	0000h	0--u uuuu	u	u	u	u	u	u	u
Software Reset during normal operation	0000h	0--0 uuuu	0	u	u	u	u	u	u
Stack Full Reset during normal operation	0000h	0--u uu11	u	u	u	u	u	u	1
Stack Underflow Reset during normal operation	0000h	0--u uu11	u	u	u	u	u	1	u
MCLR Reset during SLEEP	0000h	0--u 10uu	u	1	0	u	u	u	u
WDT Reset	0000h	0--u 01uu	1	0	1	u	u	u	u
WDT Wake-up	PC + 2	u--u 00uu	u	0	0	u	u	u	u
Brown-out Reset	0000h	0--1 11u0	1	1	1	1	0	u	u
Interrupt wake-up from SLEEP	PC + 2 <sup>(1)</sup>	u--u 00uu	u	1	0	u	u	u	u

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

**Note 1:** When the wake-up is due to an interrupt and the GIEH or GIEL bits are set, the PC is loaded with the interrupt vector (0x000008h or 0x0000018h).

# PIC18FXX2

TABLE 3-3: 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	242	442	252	452	---0 0000	---0 0000	---0 uuuu <sup>(3)</sup>
TOSH	242	442	252	452	0000 0000	0000 0000	uuuu uuuu <sup>(3)</sup>
TOSL	242	442	252	452	0000 0000	0000 0000	uuuu uuuu <sup>(3)</sup>
STKPTR	242	442	252	452	00-0 0000	uu-0 0000	uu-u uuuu <sup>(3)</sup>
PCLATU	242	442	252	452	---0 0000	---0 0000	---u uuuu
PCLATH	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
PCL	242	442	252	452	0000 0000	0000 0000	PC + 2 <sup>(2)</sup>
TBLPTRU	242	442	252	452	--00 0000	--00 0000	--uu uuuu
TBLPTRH	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
TABLAT	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
PRODH	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	242	442	252	452	0000 000x	0000 000u	uuuu uuuu <sup>(1)</sup>
INTCON2	242	442	252	452	1111 -1-1	1111 -1-1	uuuu -u-u <sup>(1)</sup>
INTCON3	242	442	252	452	11-0 0-00	11-0 0-00	uu-u u-u <sup>(1)</sup>
INDF0	242	442	252	452	N/A	N/A	N/A
POSTINC0	242	442	252	452	N/A	N/A	N/A
POSTDEC0	242	442	252	452	N/A	N/A	N/A
PREINC0	242	442	252	452	N/A	N/A	N/A
PLUSW0	242	442	252	452	N/A	N/A	N/A
FSR0H	242	442	252	452	---- xxxx	---- uuuu	---- uuuu
FSR0L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	242	442	252	452	N/A	N/A	N/A
POSTINC1	242	442	252	452	N/A	N/A	N/A
POSTDEC1	242	442	252	452	N/A	N/A	N/A
PREINC1	242	442	252	452	N/A	N/A	N/A
PLUSW1	242	442	252	452	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 3-2 for RESET value for specific condition.
- 5:** Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.
- 6:** Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they are read '0'.

TABLE 3-3: 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	242	442	252	452	---- xxxx	---- uuuu	---- uuuu
FSR1L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	242	442	252	452	---- 0000	---- 0000	---- uuuu
INDF2	242	442	252	452	N/A	N/A	N/A
POSTINC2	242	442	252	452	N/A	N/A	N/A
POSTDEC2	242	442	252	452	N/A	N/A	N/A
PREINC2	242	442	252	452	N/A	N/A	N/A
PLUSW2	242	442	252	452	N/A	N/A	N/A
FSR2H	242	442	252	452	---- xxxx	---- uuuu	---- uuuu
FSR2L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	242	442	252	452	---x xxxx	---u uuuu	---u uuuu
TMR0H	242	442	252	452	0000 0000	uuuu uuuu	uuuu uuuu
TMR0L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
OSCCON	242	442	252	452	---- ---0	---- ---0	---- ---u
LVDCON	242	442	252	452	--00 0101	--00 0101	--uu uuuu
WDTCON	242	442	252	452	---- ---0	---- ---0	---- ---u
RCON <sup>(4)</sup>	242	442	252	452	0--q 11qq	0--q qquu	u--u qquu
TMR1H	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	242	442	252	452	0-00 0000	u-uu uuuu	u-uu uuuu
TMR2	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
PR2	242	442	252	452	1111 1111	1111 1111	1111 1111
T2CON	242	442	252	452	-000 0000	-000 0000	-uuu uuuu
SSPBUF	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
SSPCON1	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
SSPCON2	242	442	252	452	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 3-2 for RESET value for specific condition.
- 5:** Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.
- 6:** Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they are read '0'.

# PIC18FXX2

TABLE 3-3: 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	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADC0N0	242	442	252	452	0000 00-0	0000 00-0	uuuu uu-u
ADC0N1	242	442	252	452	00-- 0000	00-- 0000	uu-- uuuu
CCPR1H	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	242	442	252	452	--00 0000	--00 0000	--uu uuuu
CCP2R2H	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2R2L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	242	442	252	452	--00 0000	--00 0000	--uu uuuu
TMR3H	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	242	442	252	452	0000 0000	uuuu uuuu	uuuu uuuu
SPBRG	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
RCREG	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
TXREG	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
TXSTA	242	442	252	452	0000 -010	0000 -010	uuuu -uuu
RCSTA	242	442	252	452	0000 000x	0000 000x	uuuu uuuu
EEADR	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
EEDATA	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
EECON1	242	442	252	452	xx-0 x000	uu-0 u000	uu-0 u000
EECON2	242	442	252	452	-----	-----	-----

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 3-2 for RESET value for specific condition.
- 5:** Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.
- 6:** Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they are read '0'.

TABLE 3-3: 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	242	442	252	452	---1 1111	---1 1111	---u uuuu
PIR2	242	442	252	452	---0 0000	---0 0000	---u uuuu <sup>(1)</sup>
PIE2	242	442	252	452	---0 0000	---0 0000	---u uuuu
IPR1	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
	242	442	252	452	-111 1111	-111 1111	-uuu uuuu
PIR1	242	442	252	452	0000 0000	0000 0000	uuuu uuuu <sup>(1)</sup>
	242	442	252	452	-000 0000	-000 0000	-uuu uuuu <sup>(1)</sup>
PIE1	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
	242	442	252	452	-000 0000	-000 0000	-uuu uuuu
TRISE	242	442	252	452	0000 -111	0000 -111	uuuu -uuu
TRISD	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
TRISC	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
TRISB	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
TRISA <sup>(5,6)</sup>	242	442	252	452	-111 1111 <sup>(5)</sup>	-111 1111 <sup>(5)</sup>	-uuu uuuu <sup>(5)</sup>
LATE	242	442	252	452	---- -xxx	---- -uuu	---- -uuu
LATD	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA <sup>(5,6)</sup>	242	442	252	452	-xxx xxxx <sup>(5)</sup>	-uuu uuuu <sup>(5)</sup>	-uuu uuuu <sup>(5)</sup>
PORTE	242	442	252	452	---- -000	---- -000	---- -uuu
PORTD	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA <sup>(5,6)</sup>	242	442	252	452	-x0x 0000 <sup>(5)</sup>	-u0u 0000 <sup>(5)</sup>	-uuu 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 3-2 for RESET value for specific condition.
- 5:** Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other Oscillator modes, they are disabled and read '0'.
- 6:** Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they are read '0'.

# PIC18FXX2

FIGURE 3-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD)

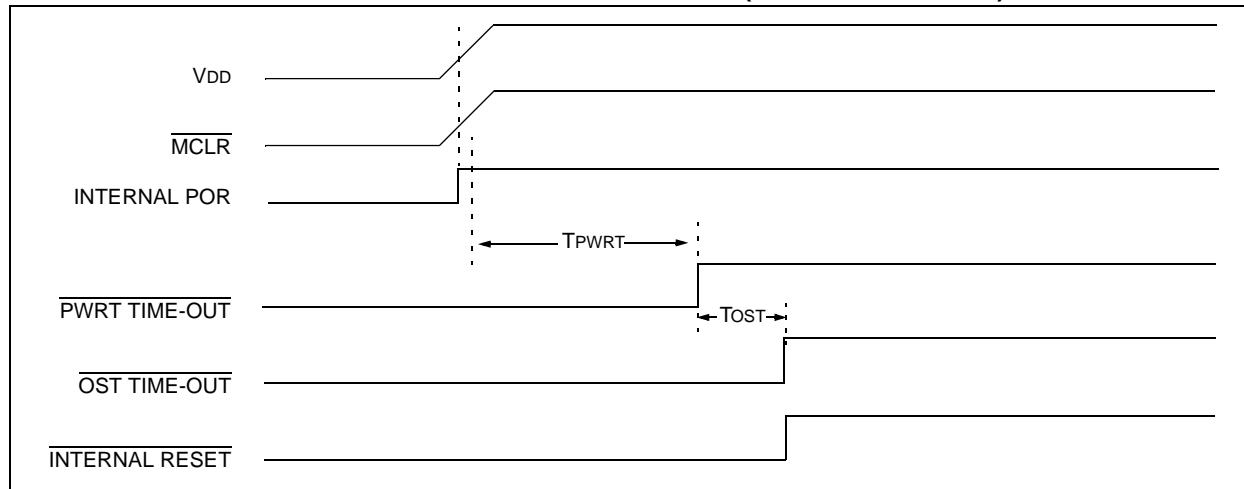


FIGURE 3-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

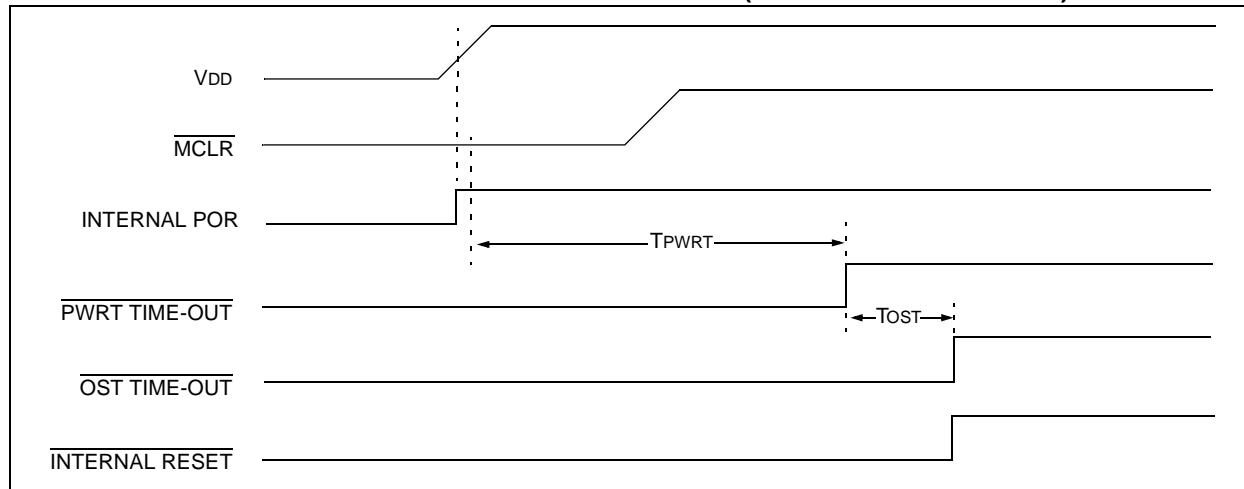
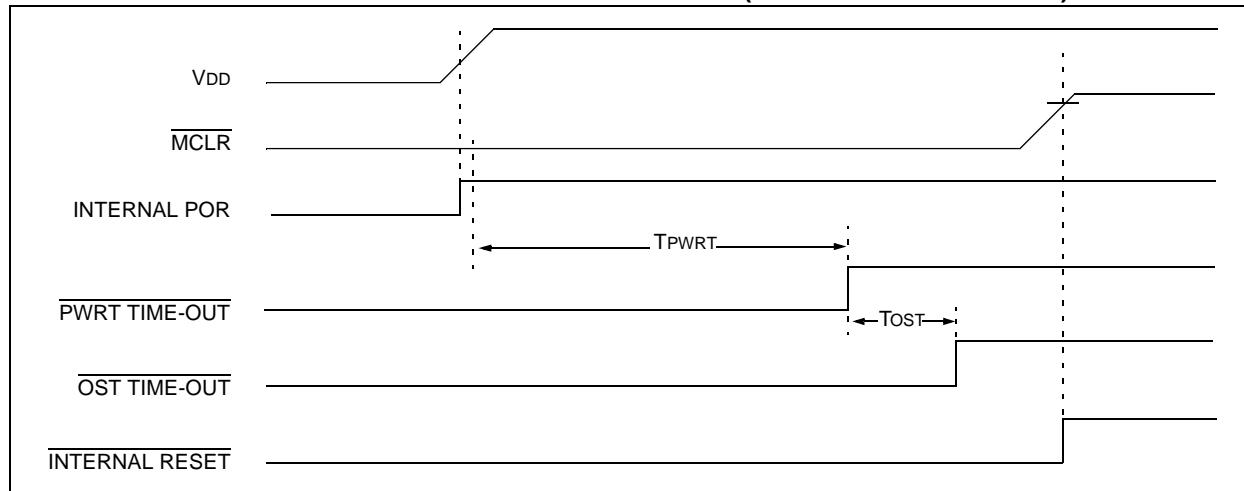
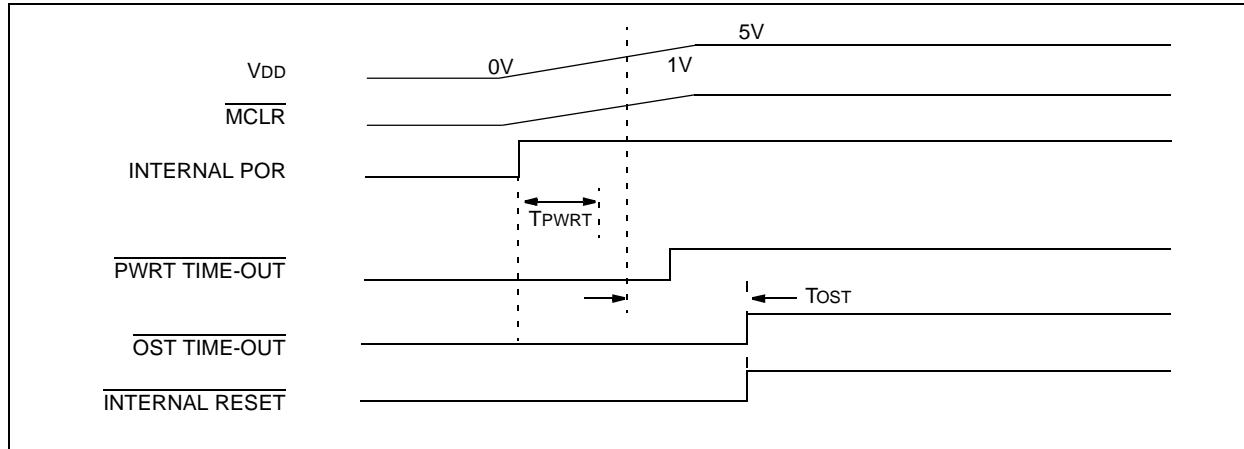


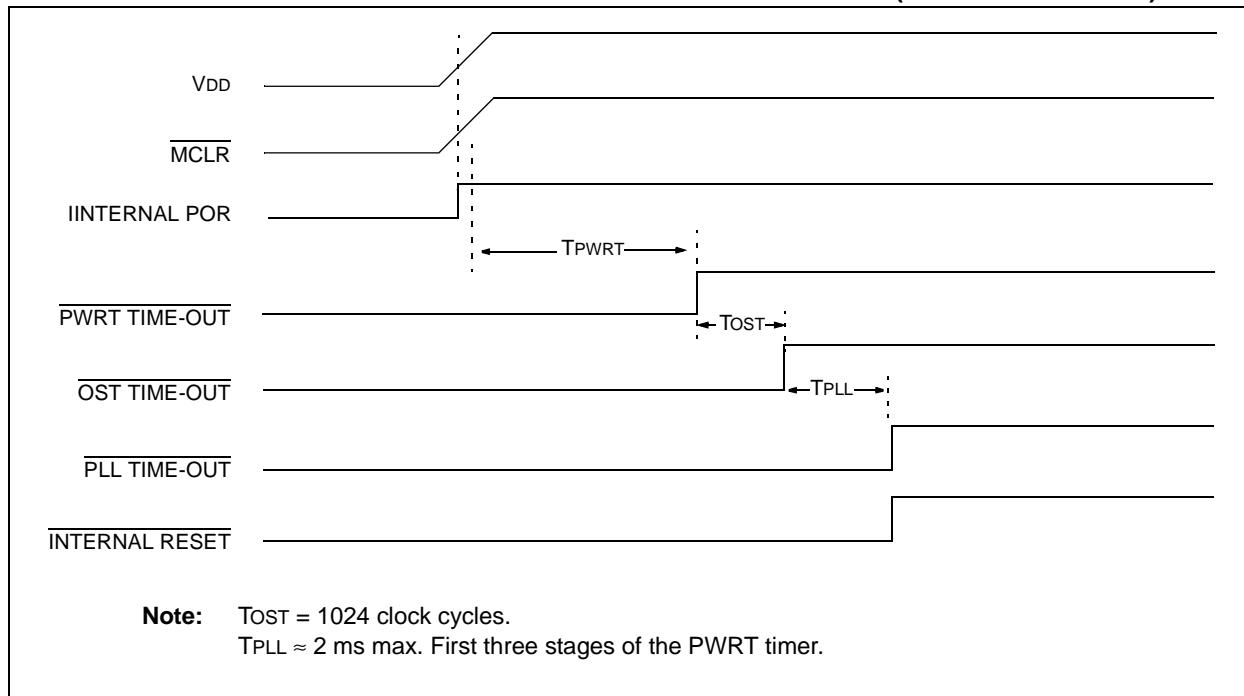
FIGURE 3-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2



**FIGURE 3-6: SLOW RISE TIME (MCLR TIED TO V<sub>DD</sub>)**



**FIGURE 3-7: TIME-OUT SEQUENCE ON POR W/ PLL ENABLED (MCLR TIED TO V<sub>DD</sub>)**



# **PIC18FXX2**

---

---

## **NOTES:**

## 4.0 MEMORY ORGANIZATION

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

- Program Memory
- Data RAM
- Data EEPROM

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

Additional detailed information for FLASH program memory and Data EEPROM is provided in Section 5.0 and Section 6.0, respectively.

### 4.1 Program Memory Organization

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

The PIC18F252 and PIC18F452 each have 32 Kbytes of FLASH memory, while the PIC18F242 and PIC18F442 have 16 Kbytes of FLASH. This means that PIC18FX52 devices can store up to 16K of single word instructions, and PIC18FX42 devices can store up to 8K of single word instructions.

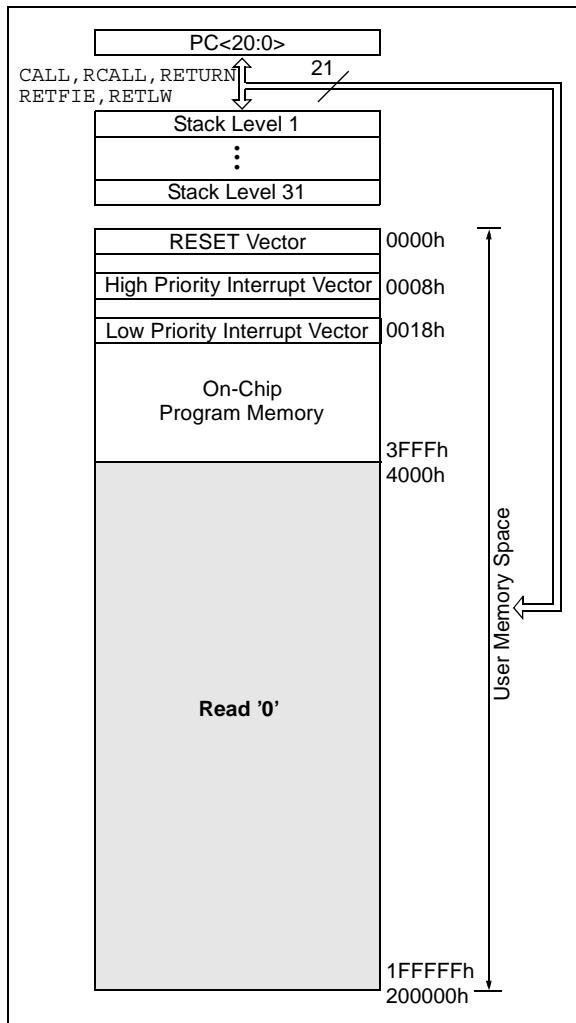
The RESET vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

Figure 4-1 shows the Program Memory Map for PIC18F242/442 devices and Figure 4-2 shows the Program Memory Map for PIC18F252/452 devices.

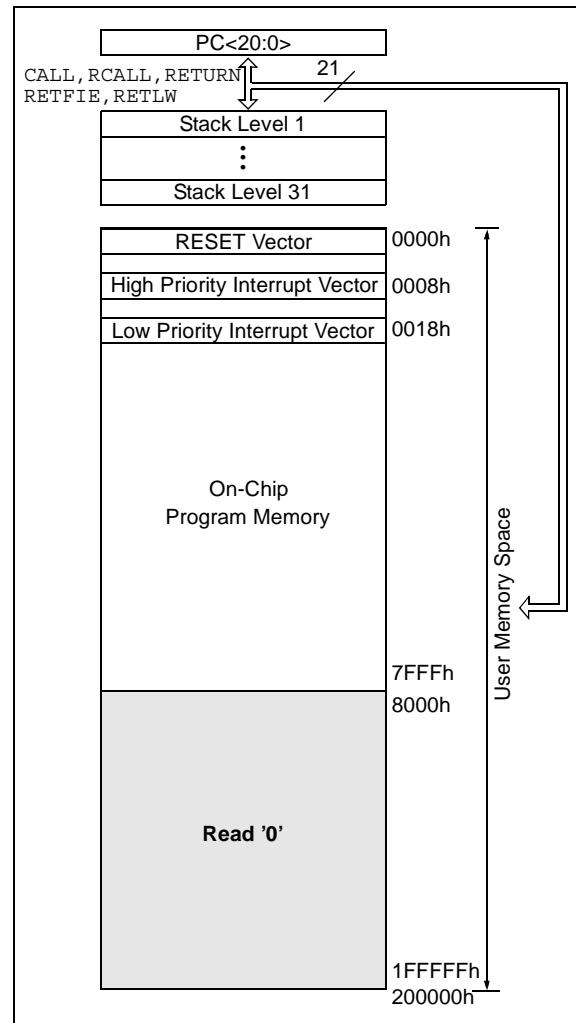
# PIC18FXX2

---

**FIGURE 4-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F442/242**



**FIGURE 4-2: PROGRAM MEMORY MAP AND STACK FOR PIC18F452/252**



## 4.2 Return Address Stack

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

The stack operates as a 31-word by 21-bit RAM and a 5-bit stack pointer, with the stack pointer initialized to 00000b after all RESETS. There is no RAM associated with stack pointer 00000b. This is only a RESET value. During a CALL type instruction, causing a push onto the stack, the stack pointer is first incremented and the RAM location pointed to by the stack pointer is written with the contents of the PC. During a RETURN type instruction, causing a pop from the stack, the contents of the RAM location pointed to by the STKPTR are transferred to the PC and then the stack pointer is decremented.

The stack space is not part of either program or data space. The stack pointer is readable and writable, and the address on the top of the stack is readable and writable through SFR registers. Data can also be pushed to, or popped from, the stack using the top-of-stack SFRs. Status bits indicate if the stack pointer is at, or beyond the 31 levels provided.

### 4.2.1 TOP-OF-STACK ACCESS

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

The user must disable the global interrupt enable bits during this time to prevent inadvertent stack operations.

### 4.2.2 RETURN STACK POINTER (STKPTR)

The STKPTR register contains the stack pointer value, the STKFUL (stack full) status bit, and the STKUNF (stack underflow) status bits. Register 4-1 shows the STKPTR register. The value of the stack pointer can be 0 through 31. The stack pointer increments when values are pushed onto the stack and decrements when values are popped off the stack. At RESET, the stack pointer value will be 0. The user may read and write the stack pointer value. This feature can be used by a Real Time Operating System for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit can only be cleared in 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 20.0 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 '0'.

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 0. The STKUNF bit will remain set until cleared in software or a POR occurs.

**Note:** Returning a value of zero to the PC on an underflow has the effect of vecting the program to the RESET vector, where the stack conditions can be verified and appropriate actions can be taken.

# PIC18FXX2

## REGISTER 4-1: STKPTR REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKOVF	STKUNF	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

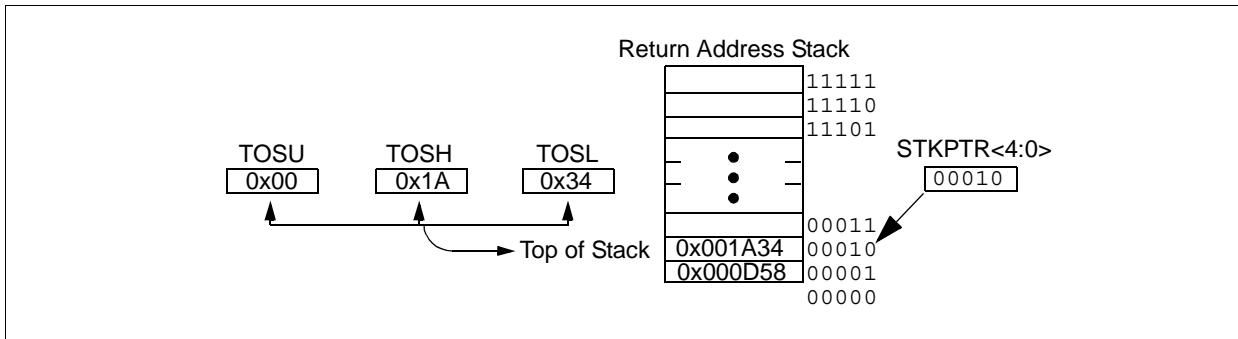
- bit 7<sup>(1)</sup> **STKOVF:** Stack Full Flag bit  
1 = Stack became full or overflowed  
0 = Stack has not become full or overflowed
- bit 6<sup>(1)</sup> **STKUNF:** Stack Underflow Flag bit  
1 = Stack underflow occurred  
0 = Stack underflow did not occur
- bit 5 **Unimplemented:** Read as '0'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

**Note 1:** Bit 7 and bit 6 can only be cleared in user software or by a POR.

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

## FIGURE 4-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



### 4.2.3 PUSH AND POP INSTRUCTIONS

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

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

### 4.2.4 STACK FULL/UNDERFLOW RESETS

These resets are enabled by programming the STVREN configuration bit. When the STVREN bit is disabled, a full or underflow condition will set the appropriate STKFUL or STKUNF bit, but not cause a device RESET. When the STVREN bit is enabled, a full or underflow will set the appropriate STKFUL or STKUNF bit and then cause a device RESET. The STKFUL or STKUNF bits are only cleared by the user software or a POR Reset.

### 4.3 Fast Register Stack

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

A low or high priority interrupt source will push values into the stack registers. If both low and high priority interrupts are enabled, the stack registers cannot be used reliably for 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.

If high priority interrupts are not disabled during low priority interrupts, users must save the key registers in software during a low priority 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 **FAST CALL** instruction must be executed.

Example 4-1 shows a source code example that uses the fast register stack.

#### EXAMPLE 4-1: 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

```

### 4.4 PCL, PCLATH and PCLATU

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21-bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the  $\text{PC}_{<15:8>}$  bits and is not directly readable or writable. Updates to the PCH register may be performed through the PCLATH register. The upper byte is called PCU. This register contains the  $\text{PC}_{<20:16>}$  bits and is not directly readable or writable. Updates to the PCU register may be performed through the PCLATU register.

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

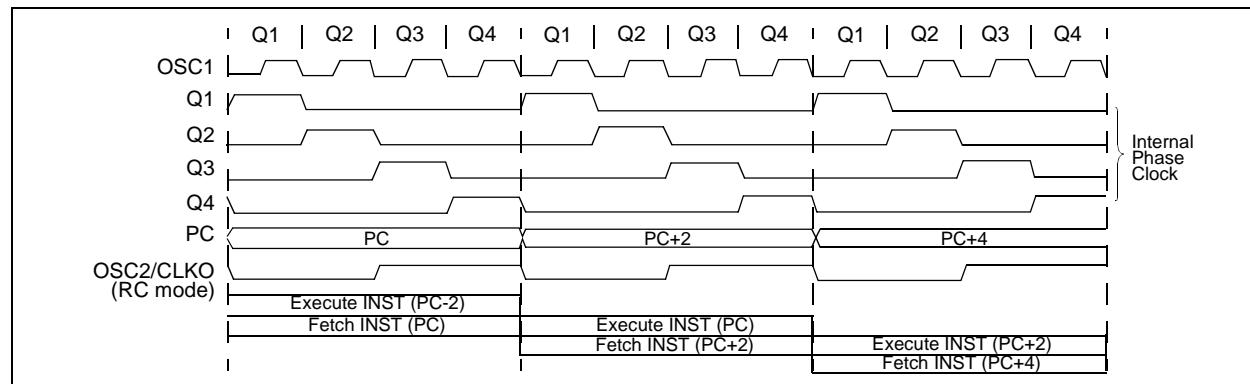
The **CALL**, **RCALL**, **GOTO** and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

The contents of PCLATH and PCLATU will be transferred to the program counter by an operation that writes PCL. Similarly, the upper two bytes of the program counter will be transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see Section 4.8.1).

### 4.5 Clocking Scheme/Instruction Cycle

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

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



## 4.6 Instruction Flow/Pipelining

An “Instruction Cycle” consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle, while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO) then two cycles are required to complete the instruction (Example 4-2).

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

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

### EXAMPLE 4-2: 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.

## 4.7 Instructions in Program Memory

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSB = '0'). Figure 4-5 shows an example of how instruction words are stored in the program memory. To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see Section 4.4).

The CALL and GOTO instructions have an 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 4-5 shows how the instruction “GOTO 000006h” is encoded in the program memory. Program branch instructions which encode a relative address offset operate in the same manner. The offset value stored in a branch instruction represents the number of single word instructions that the PC will be offset by. Section 20.0 provides further details of the instruction set.

FIGURE 4-5: INSTRUCTIONS IN PROGRAM MEMORY

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

#### 4.7.1 TWO-WORD INSTRUCTIONS

The PIC18FXX2 devices have four two-word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the 4 MSBs set to 1's and is a special kind of NOP instruction. The lower 12 bits of the second word contain data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the

second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two-word instruction is preceded by a conditional instruction that changes the PC. A program example that demonstrates this concept is shown in Example 4-3. Refer to Section 20.0 for further details of the instruction set.

#### EXAMPLE 4-3: TWO-WORD INSTRUCTIONS

<b>CASE 1:</b>	
<b>Object Code</b>	<b>Source Code</b>
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; No, execute 2-word instruction
1111 0100 0101 0110	; 2nd operand holds address of REG2
0010 0100 0000 0000	ADDWF REG3 ; continue code

<b>CASE 2:</b>	
<b>Object Code</b>	<b>Source Code</b>
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; Yes
1111 0100 0101 0110	; 2nd operand becomes NOP
0010 0100 0000 0000	ADDWF REG3 ; continue code

## 4.8 Lookup Tables

Lookup tables are implemented two ways. These are:

- Computed GOTO
- Table Reads

### 4.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

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

The offset value (value in WREG) specifies the number of bytes that the program counter should advance.

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

**Note:** The ADDWF PCL instruction does not update PCLATH and PCLATU. A read operation on PCL must be performed to update PCLATH and PCLATU.

### 4.8.2 TABLE READS/TABLE WRITES

A better method of storing data in program memory allows 2 bytes of data to be stored in each instruction location.

Lookup table data may be stored 2 bytes per program word by using table reads and writes. The table pointer (TBLPTR) specifies the byte address and the table latch (TABLAT) contains the data that is read from, or written to program memory. Data is transferred to/from program memory, one byte at a time.

A description of the Table Read/Table Write operation is shown in Section 3.0.

## 4.9 Data Memory Organization

The data memory is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. Figure 4-6 and Figure 4-7 show the data memory organization for the PIC18FXX2 devices.

The data memory map is divided into as many as 16 banks that contain 256 bytes each. The lower 4 bits of the Bank Select Register (BSR<3:0>) select which bank will be accessed. The upper 4 bits for the BSR are not implemented.

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

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

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

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM. Section 4.10 provides a detailed description of the Access RAM.

### 4.9.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly or indirectly. Indirect addressing operates using a File Select Register and corresponding Indirect File Operand. The operation of indirect addressing is shown in Section 4.12.

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

Data RAM is available for use as GPR registers by all instructions. The top half of Bank 15 (0xF80 to 0xFFFF) contains SFRs. All other banks of data memory contain GPR registers, starting with Bank 0.

### 4.9.2 SPECIAL FUNCTION REGISTERS

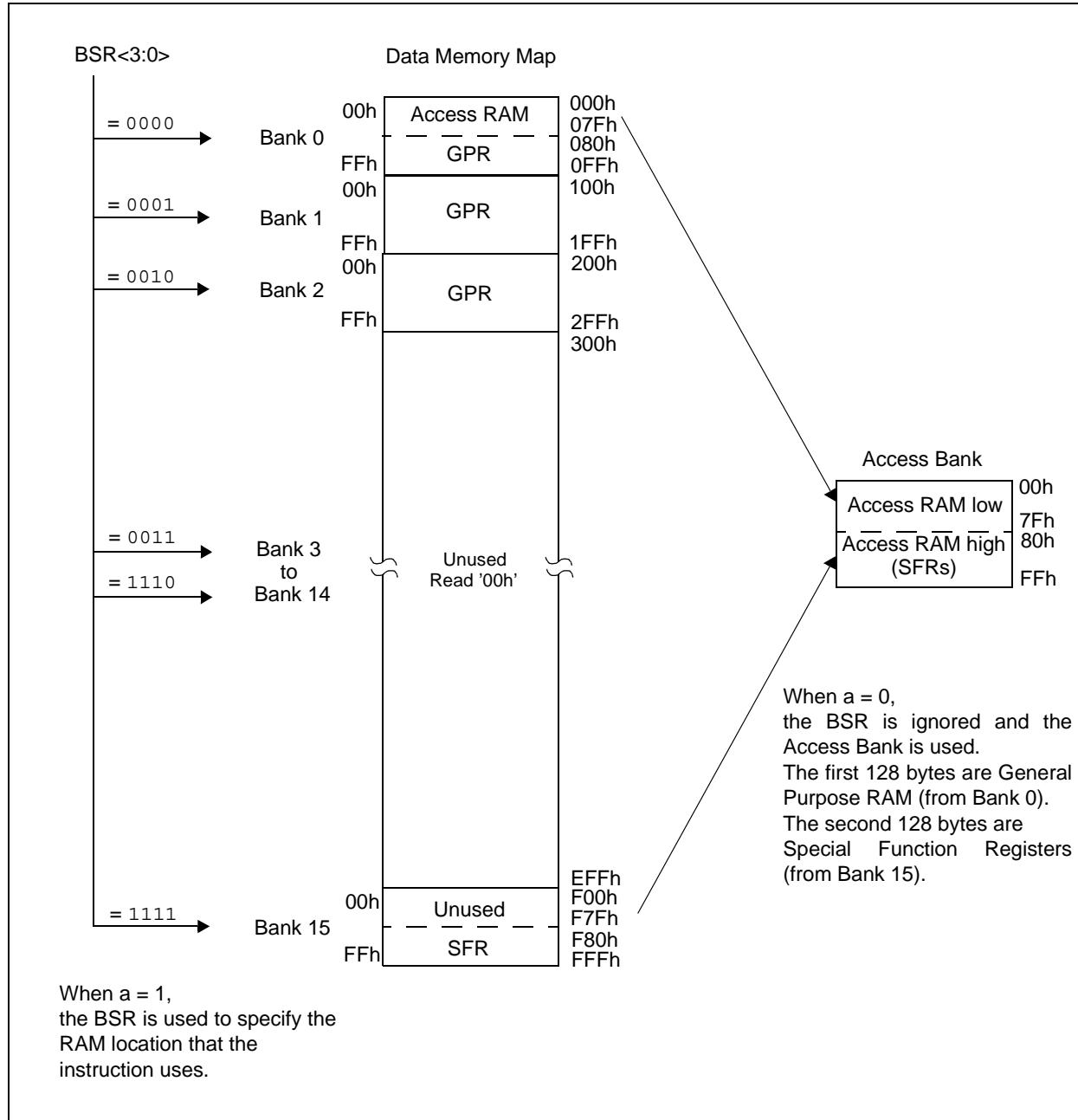
The Special Function Registers (SFRs) are registers used by the CPU and Peripheral Modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 4-1 and Table 4-2.

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

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

The unused SFR locations will be unimplemented and read as '0's. See Table 4-1 for addresses for the SFRs.

**FIGURE 4-6: DATA MEMORY MAP FOR PIC18F242/442**



# PIC18FXX2

**FIGURE 4-7: DATA MEMORY MAP FOR PIC18F252/452**

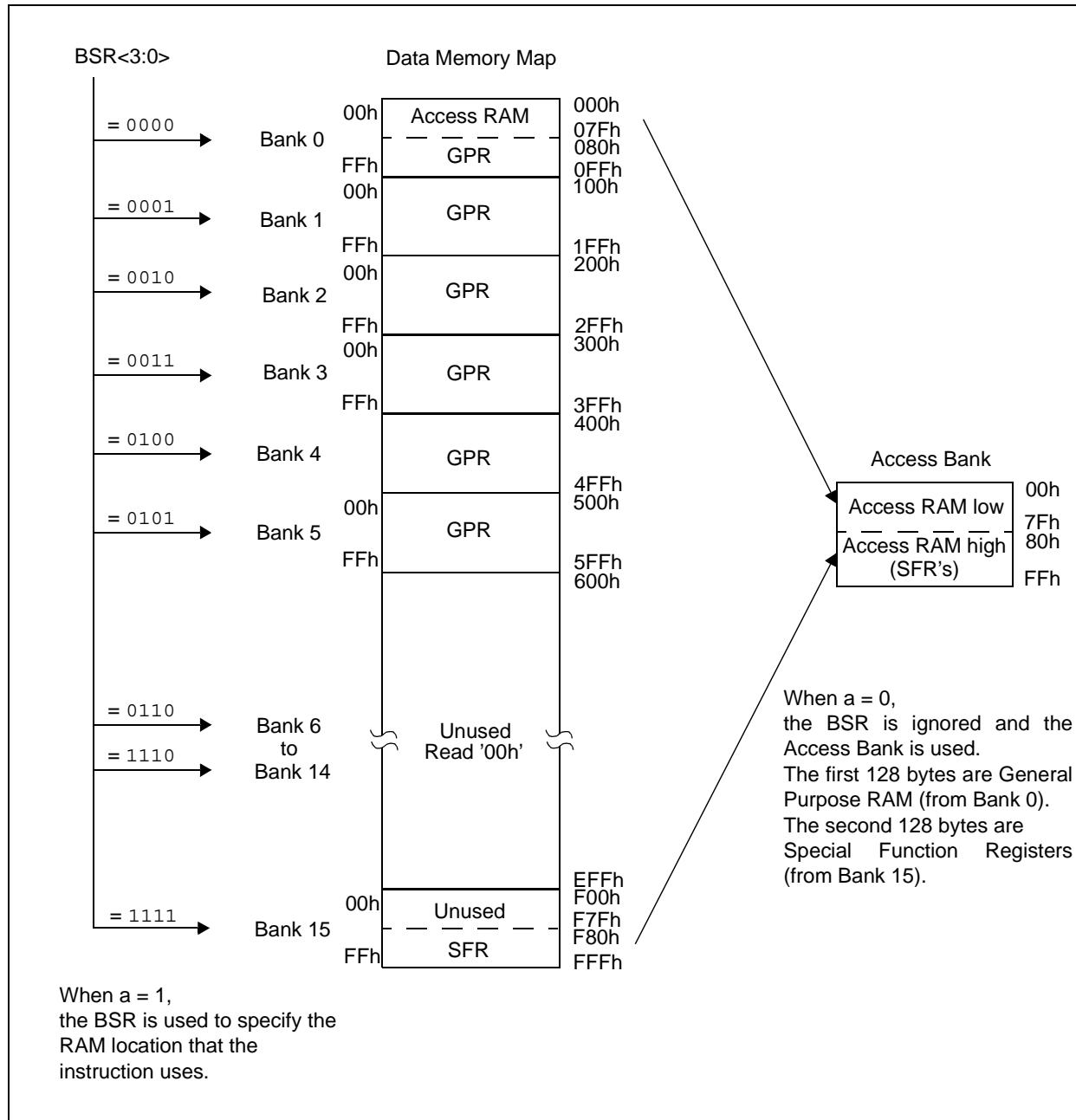


TABLE 4-1: SPECIAL FUNCTION REGISTER MAP

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

Note 1: Unimplemented registers are read as '0'.

2: This register is not available on PIC18F2X2 devices.

3: This is not a physical register.

# PIC18FXX2

---

**TABLE 4-2: REGISTER FILE SUMMARY**

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 37		
TOSH	Top-of-Stack High Byte (TOS<15:8>)						0000 0000 37					
TOSL	Top-of-Stack Low Byte (TOS<7:0>)						0000 0000 37					
STKPTR	STKFUL	STKUNF	—	Return Stack Pointer						00-0 0000 38		
PCLATU	—	—	—	Holding Register for PC<20:16>						--- 0 0000 39		
PCLATH	Holding Register for PC<15:8>						0000 0000 39					
PCL	PC Low Byte (PC<7:0>)						0000 0000 39					
TBLPTRU	—	—	bit21 <sup>(2)</sup>	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)						--00 0000 58		
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)						0000 0000 58					
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)						0000 0000 58					
TABLAT	Program Memory Table Latch						0000 0000 58					
PRODH	Product Register High Byte						xxxx xxxx 71					
PRODL	Product Register Low Byte						xxxx xxxx 71					
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x 75			
INTCON2	RBU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1 76			
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00 77			
INDF0	Uses contents of FSR0 to address data memory - value of FSR0 not changed (not a physical register)								n/a	50		
POSTINC0	Uses contents of FSR0 to address data memory - value of FSR0 post-incremented (not a physical register)								n/a	50		
POSTDEC0	Uses contents of FSR0 to address data memory - value of FSR0 post-decremented (not a physical register)								n/a	50		
PREINC0	Uses contents of FSR0 to address data memory - value of FSR0 pre-incremented (not a physical register)								n/a	50		
PLUSW0	Uses contents of FSR0 to address data memory - value of FSR0 (not a physical register). Offset by value in WREG.								n/a	50		
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte							
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx 50			
WREG	Working Register								xxxx xxxx n/a			
INDF1	Uses contents of FSR1 to address data memory - value of FSR1 not changed (not a physical register)								n/a	50		
POSTINC1	Uses contents of FSR1 to address data memory - value of FSR1 post-incremented (not a physical register)								n/a	50		
POSTDEC1	Uses contents of FSR1 to address data memory - value of FSR1 post-decremented (not a physical register)								n/a	50		
PREINC1	Uses contents of FSR1 to address data memory - value of FSR1 pre-incremented (not a physical register)								n/a	50		
PLUSW1	Uses contents of FSR1 to address data memory - value of FSR1 (not a physical register). Offset by value in WREG.								n/a	50		
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte							
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx 50			
BSR	—	—	—	—	Bank Select Register							
INDF2	Uses contents of FSR2 to address data memory - value of FSR2 not changed (not a physical register)								n/a	50		
POSTINC2	Uses contents of FSR2 to address data memory - value of FSR2 post-incremented (not a physical register)								n/a	50		
POSTDEC2	Uses contents of FSR2 to address data memory - value of FSR2 post-decremented (not a physical register)								n/a	50		
PREINC2	Uses contents of FSR2 to address data memory - value of FSR2 pre-incremented (not a physical register)								n/a	50		
PLUSW2	Uses contents of FSR2 to address data memory - value of FSR2 (not a physical register). Offset by value in WREG.								n/a	50		
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte							
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx 50			
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx 52			
TMR0H	Timer0 Register High Byte								0000 0000 105			
TMR0L	Timer0 Register Low Byte								xxxx xxxx 105			
T0CON	TMROON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111 103			

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

**Note 1:** RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other Oscillator modes.

**2:** Bit 21 of the TBLPTRU allows access to the device configuration bits.

**3:** These registers and bits are reserved on the PIC18F2X2 devices; always maintain these clear.

TABLE 4-2: REGISTER FILE SUMMARY (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:
OSCCON	—	—	—	—	—	—	—	SCS	-----0	21
LVDCON	—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	191
WDTCON	—	—	—	—	—	—	—	SWDTE	-----0	203
RCON	IPEN	—	—	RI	TO	PD	POR	BOR	0--1 11qq	53, 28, 84
TMR1H	Timer1 Register High Byte								xxxx xxxx	107
TMR1L	Timer1 Register Low Byte								xxxx xxxx	107
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0-00 0000	107
TMR2	Timer2 Register								0000 0000	111
PR2	Timer2 Period Register								1111 1111	112
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	111
SSPBUF	SSP Receive Buffer/Transmit Register								xxxx xxxx	125
SSPADD	SSP Address Register in I <sup>2</sup> C Slave mode. SSP Baud Rate Reload Register in I <sup>2</sup> C Master mode.								0000 0000	134
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000	126
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	127
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	137
ADRESH	A/D Result Register High Byte								xxxx xxxx	187,188
ADRESL	A/D Result Register Low Byte								xxxx xxxx	187,188
ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON	0000 00-0	181
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	182
CCPR1H	Capture/Compare/PWM Register1 High Byte								xxxx xxxx	121, 123
CCPR1L	Capture/Compare/PWM Register1 Low Byte								xxxx xxxx	121, 123
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	-00 0000	117
CCPR2H	Capture/Compare/PWM Register2 High Byte								xxxx xxxx	121, 123
CCPR2L	Capture/Compare/PWM Register2 Low Byte								xxxx xxxx	121, 123
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	-00 0000	117
TMR3H	Timer3 Register High Byte								xxxx xxxx	113
TMR3L	Timer3 Register Low Byte								xxxx xxxx	113
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	0000 0000	113
SPBRG	USART1 Baud Rate Generator								0000 0000	168
RCREG	USART1 Receive Register								0000 0000	175, 178, 180
TXREG	USART1 Transmit Register								0000 0000	173, 176, 179
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	166
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	167
EEADR	Data EEPROM Address Register								0000 0000	65, 69
EEDATA	Data EEPROM Data Register								0000 0000	69
EECON2	Data EEPROM Control Register 2 (not a physical register)								---- ----	65, 69
EECON1	EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	66

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

**Note 1:** RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other Oscillator modes.

**2:** Bit 21 of the TBLPTRU allows access to the device configuration bits.

**3:** These registers and bits are reserved on the PIC18F2X2 devices; always maintain these clear.

# PIC18FXX2

---

**TABLE 4-2: REGISTER FILE SUMMARY (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:					
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	83					
PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	79					
PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	81					
IPR1	PSPIP <sup>(3)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	82					
PIR1	PSPIF <sup>(3)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	78					
PIE1	PSPIE <sup>(3)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	80					
TRISE <sup>(3)</sup>	IBF	OBF	IBOV	PSPMODE	—	Data Direction bits for PORTE			0000 -111	98					
TRISD <sup>(3)</sup>	Data Direction Control Register for PORTD							1111 1111	96						
TRISC	Data Direction Control Register for PORTC							1111 1111	93						
TRISB	Data Direction Control Register for PORTB							1111 1111	90						
TRISA	—	TRISA6 <sup>(1)</sup>	Data Direction Control Register for PORTA					-111 1111	87						
LATE <sup>(3)</sup>	—	—	—	—	—	Read PORTE Data Latch, Write PORTE Data Latch			---- -xxx	99					
LATD <sup>(3)</sup>	Read PORTD Data Latch, Write PORTD Data Latch							xxxx xxxx	95						
LATC	Read PORTC Data Latch, Write PORTC Data Latch							xxxx xxxx	93						
LATB	Read PORTB Data Latch, Write PORTB Data Latch							xxxx xxxx	90						
LATA	—	LATA6 <sup>(1)</sup>	Read PORTA Data Latch, Write PORTA Data Latch <sup>(1)</sup>					-xxx xxxx	87						
PORTE <sup>(3)</sup>	Read PORTE pins, Write PORTE Data Latch							---- -000	99						
PORTD <sup>(3)</sup>	Read PORTD pins, Write PORTD Data Latch							xxxx xxxx	95						
PORTC	Read PORTC pins, Write PORTC Data Latch							xxxx xxxx	93						
PORTB	Read PORTB pins, Write PORTB Data Latch							xxxx xxxx	90						
PORTA	—	RA6 <sup>(1)</sup>	Read PORTA pins, Write PORTA Data Latch <sup>(1)</sup>					-x0x 0000	87						

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

**Note 1:** RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other Oscillator modes.

**2:** Bit 21 of the TBLPTRU allows access to the device configuration bits.

**3:** These registers and bits are reserved on the PIC18F2X2 devices; always maintain these clear.

## 4.10 Access Bank

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

This data memory region can be used for:

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

The Access Bank is comprised of the upper 128 bytes in Bank 15 (SFRs) and the lower 128 bytes in Bank 0. These two sections will be referred to as Access RAM High and Access RAM Low, respectively. Figure 4-6 and Figure 4-7 indicate the Access RAM areas.

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

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

## 4.11 Bank Select Register (BSR)

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

BSR<3:0> holds the upper 4 bits of the 12-bit RAM address. The BSR<7:4> bits will always read '0's, and writes will have no effect.

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

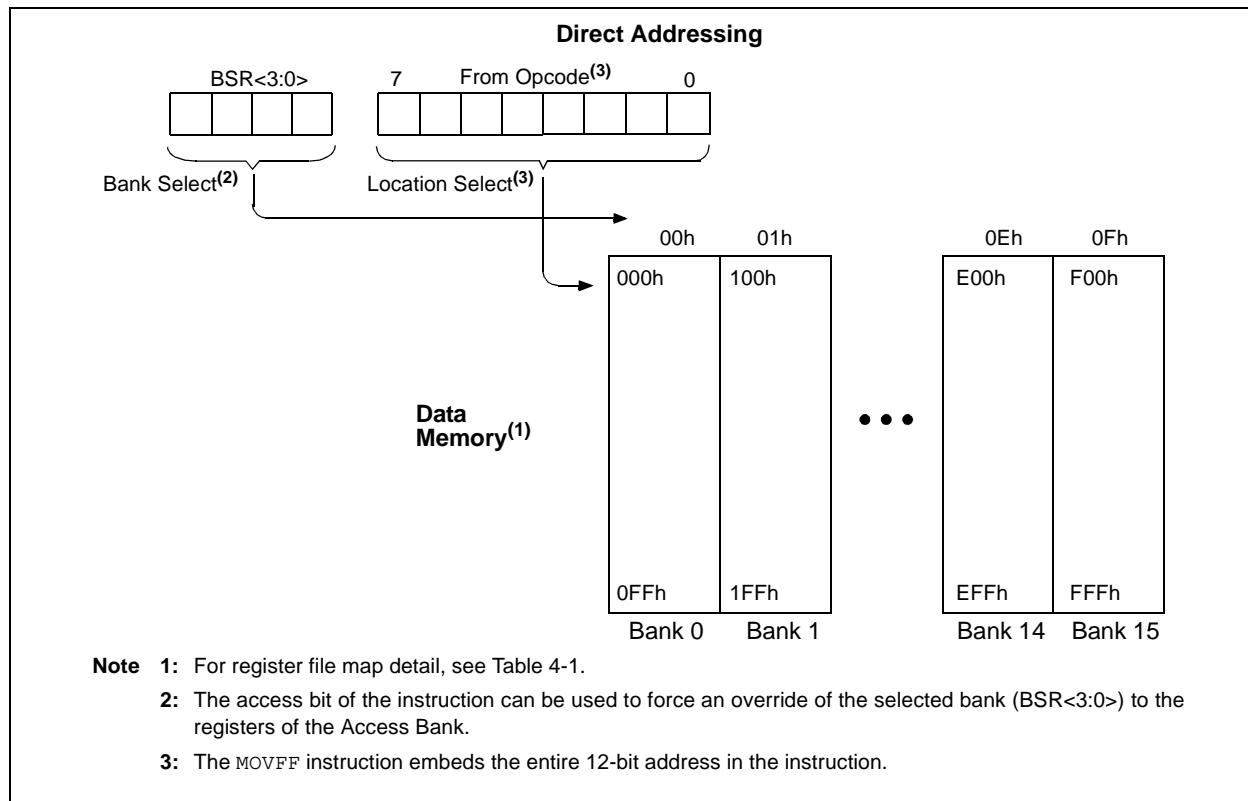
If the currently selected bank is not implemented, any read will return all '0's and all writes are ignored. The STATUS register bits will be set/cleared as appropriate for the instruction performed.

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

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

Section 4.12 provides a description of indirect addressing, which allows linear addressing of the entire RAM space.

**FIGURE 4-8: DIRECT ADDRESSING**



## 4.12 Indirect Addressing, INDF and FSR Registers

Indirect addressing is a mode of addressing data memory, where the data memory address in the instruction is not fixed. An FSR register is used as a pointer to the data memory location that is to be read or written. Since this pointer is in RAM, the contents can be modified by the program. This can be useful for data tables in the data memory and for software stacks. Figure 4-9 shows the operation of indirect addressing. This shows the moving of the value of the FSR register.

Indirect addressing is possible by using one of the INDF registers. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = 0), will read 00h. Writing to the INDF register indirectly, results in a no operation. The FSR register contains a 12-bit address, which is shown in Figure 4-10.

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

Example 4-4 shows a simple use of indirect addressing to clear the RAM in Bank1 (locations 100h-1FFh) in a minimum number of instructions.

### EXAMPLE 4-4: HOW TO CLEAR RAM (BANK1) USING INDIRECT ADDRESSING

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

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

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

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

the data from the address pointed to by FSR1H:FSR1L. INDFn can be used in code anywhere an operand can be used.

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

#### 4.12.1 INDIRECT ADDRESSING OPERATION

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

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

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

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

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

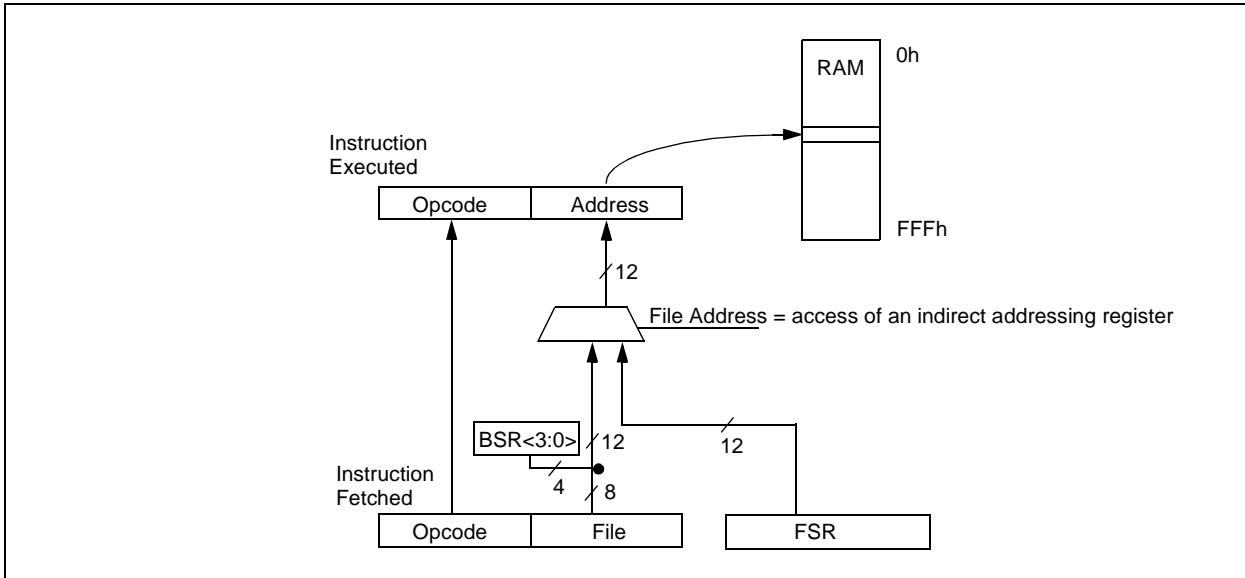
Adding these features allows the FSRn to be used as a stack pointer, in addition to its uses for table operations in data memory.

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

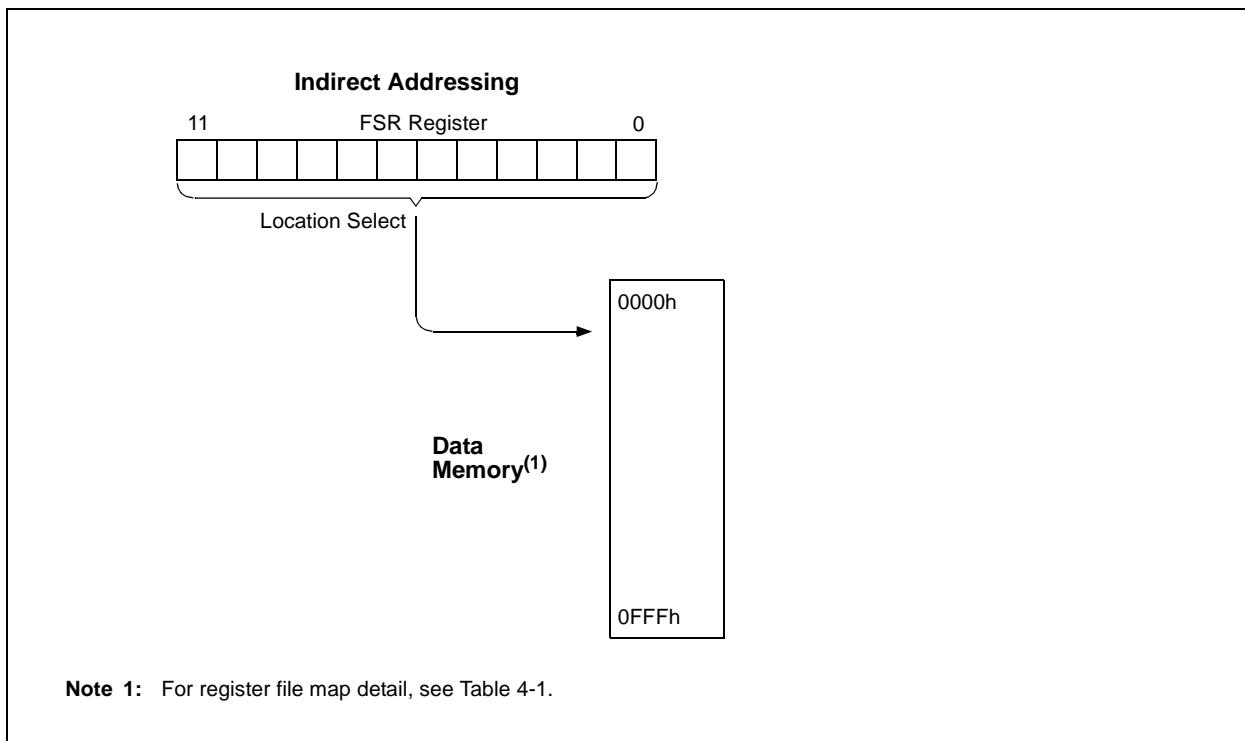
If an FSR register contains a value that points to one of the INDFn, an indirect read will read 00h (zero bit is set), while an indirect write will be equivalent to a NOP (STATUS bits are not affected).

If an indirect addressing operation is done where the target address is an FSRnH or FSRnL register, the write operation will dominate over the pre- or post-increment/decrement functions.

**FIGURE 4-9: INDIRECT ADDRESSING OPERATION**



**FIGURE 4-10: INDIRECT ADDRESSING**





## 4.14 RCON Register

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

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

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

## **REGISTER 4-3: RCON REGISTER**

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	RI	TO	PD	POR	BOR
bit 7						bit 0	

- |         |  |
|---------|--|
| bit 7   | <b>IPEN:</b> Interrupt Priority Enable bit<br>1 = Enable priority levels on interrupts<br>0 = Disable priority levels on interrupts (16CXXX Compatibility mode)  |
| bit 6-5 | <b>Unimplemented:</b> Read as '0'  |
| bit 4   | <b>RI:</b> RESET Instruction Flag bit<br>1 = The RESET instruction was not executed<br>0 = The RESET instruction was executed causing a device RESET<br>(must be set in software after a Brown-out Reset occurs) |
| bit 3   | <b>TO:</b> Watchdog Time-out Flag bit<br>1 = After power-up, CLRWDT instruction, or SLEEP instruction<br>0 = A WDT time-out occurred   |
| bit 2   | <b>PD:</b> Power-down Detection Flag bit<br>1 = After power-up or by the CLRWDT instruction<br>0 = By execution of the SLEEP instruction   |
| bit 1   | <b>POR:</b> Power-on Reset Status bit<br>1 = A Power-on Reset has not occurred<br>0 = A Power-on Reset occurred<br>(must be set in software after a Power-on Reset occurs)                                       |
| bit 0   | <b>BOR:</b> Brown-out Reset Status bit<br>1 = A Brown-out Reset has not occurred<br>0 = A Brown-out Reset occurred<br>(must be set in software after a Brown-out Reset occurs)                                   |

---

**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

# **PIC18FXX2**

---

---

## **NOTES:**

## 5.0 FLASH PROGRAM MEMORY

The FLASH Program Memory is readable, writable, and erasable during normal operation over the entire VDD range.

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

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

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.

## 5.1 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)

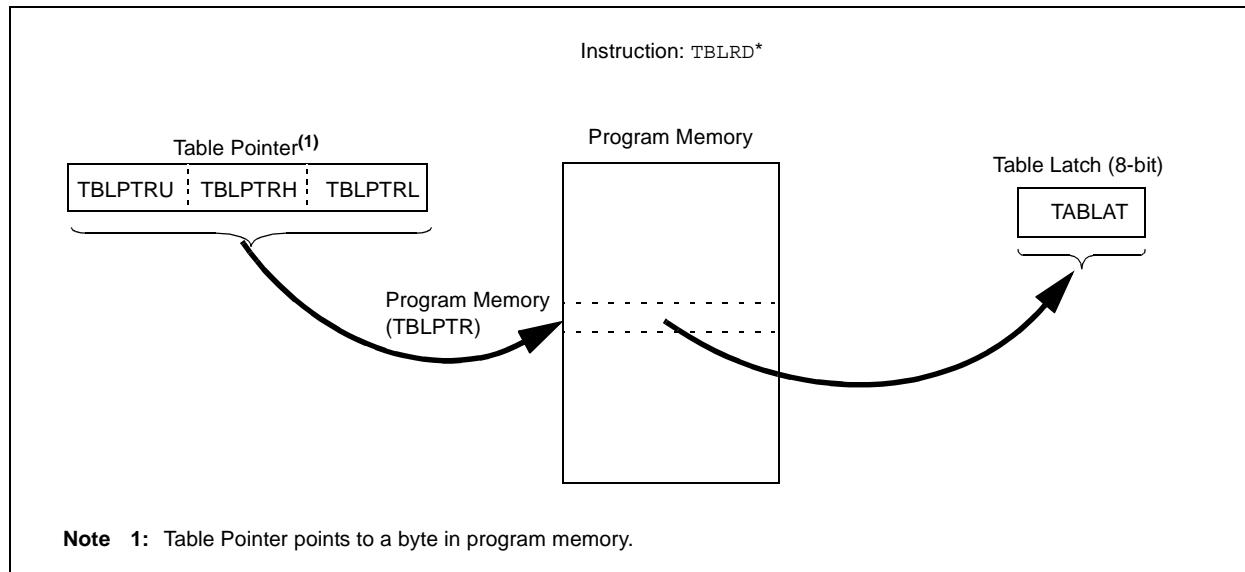
The program memory space is 16-bits wide, while the data RAM space is 8-bits wide. Table Reads and Table Writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table Read operations retrieve data from program memory and places it into the data RAM space. Figure 5-1 shows the operation of a Table Read with program memory and data RAM.

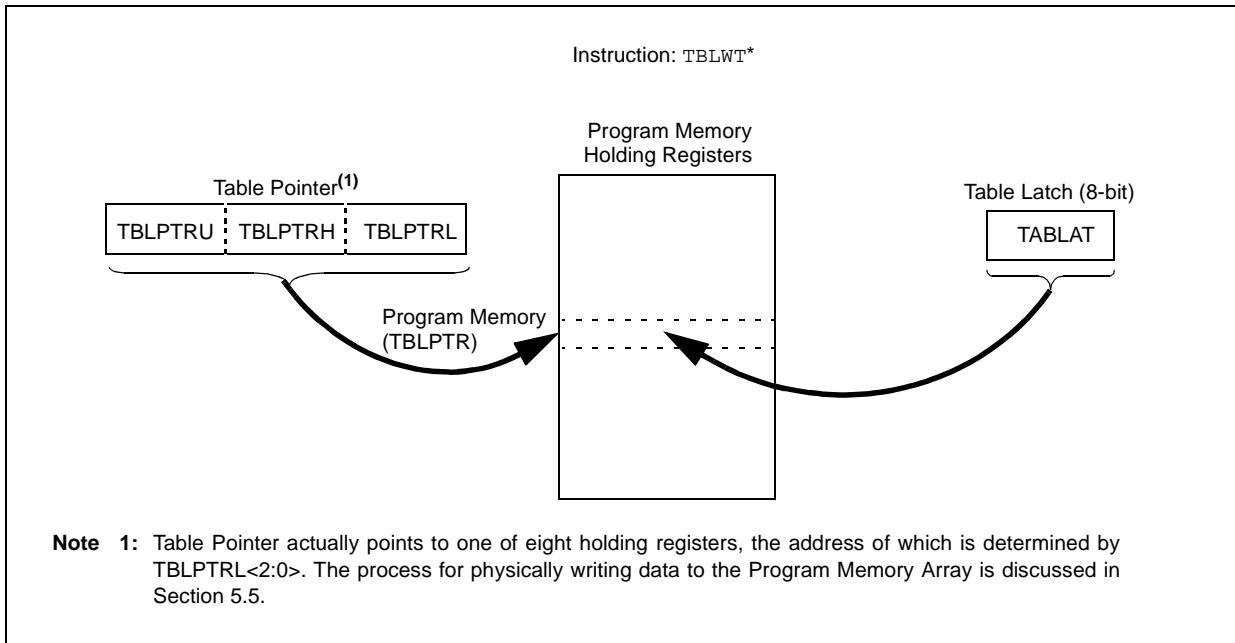
Table Write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in Section 5.5, "Writing to FLASH Program Memory". Figure 5-2 shows the operation of a Table Write with program 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 5-1: TABLE READ OPERATION**



**FIGURE 5-2: TABLE WRITE OPERATION**



## 5.2 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

### 5.2.1 EECON1 AND EECON2 REGISTERS

EECON1 is the control register for memory accesses.

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

Control bit EEPGD determines if the access will be 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.

Control bit CFGS determines if the access will be to the configuration registers or to program memory/data EEPROM memory. When set, subsequent operations will operate on configuration registers, regardless of EEPGD (see "Special Features of the CPU", Section 19.0). When clear, memory selection access is determined by EEPGD.

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

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a MCLR Reset or a WDT Time-out Reset during normal operation. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR), due to RESET values of zero.

Control bit WR initiates write operations. This bit cannot be cleared, only set, in software. It is cleared in hardware at the completion of the write operation. The inability to clear the WR bit in software prevents the accidental or premature termination of a write operation.

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

## REGISTER 5-1: EECON1 REGISTER (ADDRESS FA6h)

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	WREN	WR	RD
bit 7							bit 0

- 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 EE 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 EE Error Flag bit  
           1 = A write operation is prematurely terminated  
                       (any RESET during self-timed programming in normal operation)  
           0 = The write operation completed  
**Note:** When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.
- bit 2      **WREN:** FLASH Program/Data EE Write Enable bit  
           1 = Allows write cycles  
           0 = Inhibits write to the 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.)  
           0 = Does not initiate an EEPROM read

## 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

## 5.2.2 TABLAT - TABLE LATCH REGISTER

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

## 5.2.3 TBLPTR - TABLE POINTER REGISTER

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

The table pointer, TBLPTR, is used by the TBLRD and TBLWT instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations are shown in Table 5-1. These operations on the TBLPTR only affect the low order 21 bits.

## 5.2.4 TABLE POINTER BOUNDARIES

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

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

When a TBLWT is executed, the three LSbs of the Table Pointer (TBLPTR<2:0>) determine which of the eight program memory holding registers is written to. When the timed write to program memory (long write) begins, the 19 MSbs of the Table Pointer, TBLPTR (TBLPTR<21:3>), will determine which program memory block of 8 bytes is written to. For more detail, see Section 5.5 ("Writing to FLASH Program Memory").

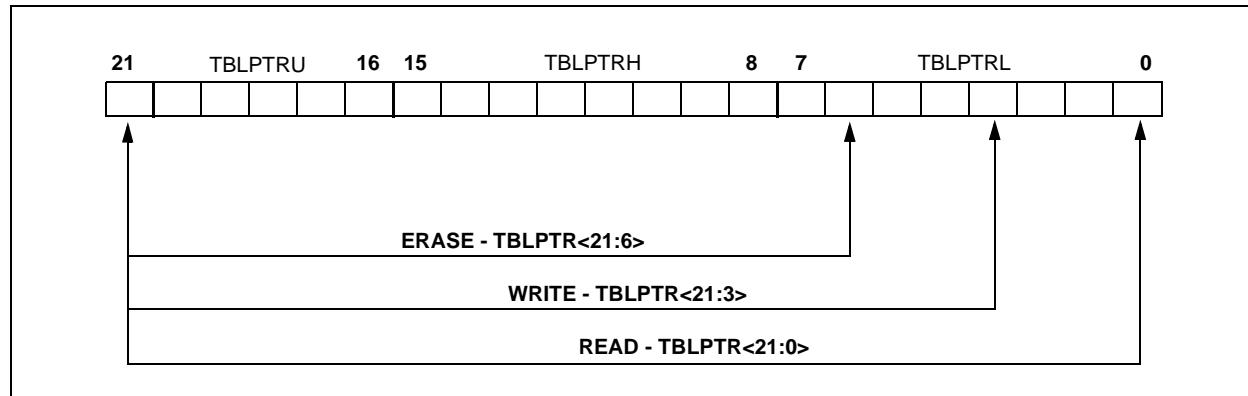
When an erase of program memory is executed, the 16 MSbs of the Table Pointer (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

Figure 5-3 describes the relevant boundaries of TBLPTR based on FLASH program memory operations.

**TABLE 5-1: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS**

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

**FIGURE 5-3: TABLE POINTER BOUNDARIES BASED ON OPERATION**



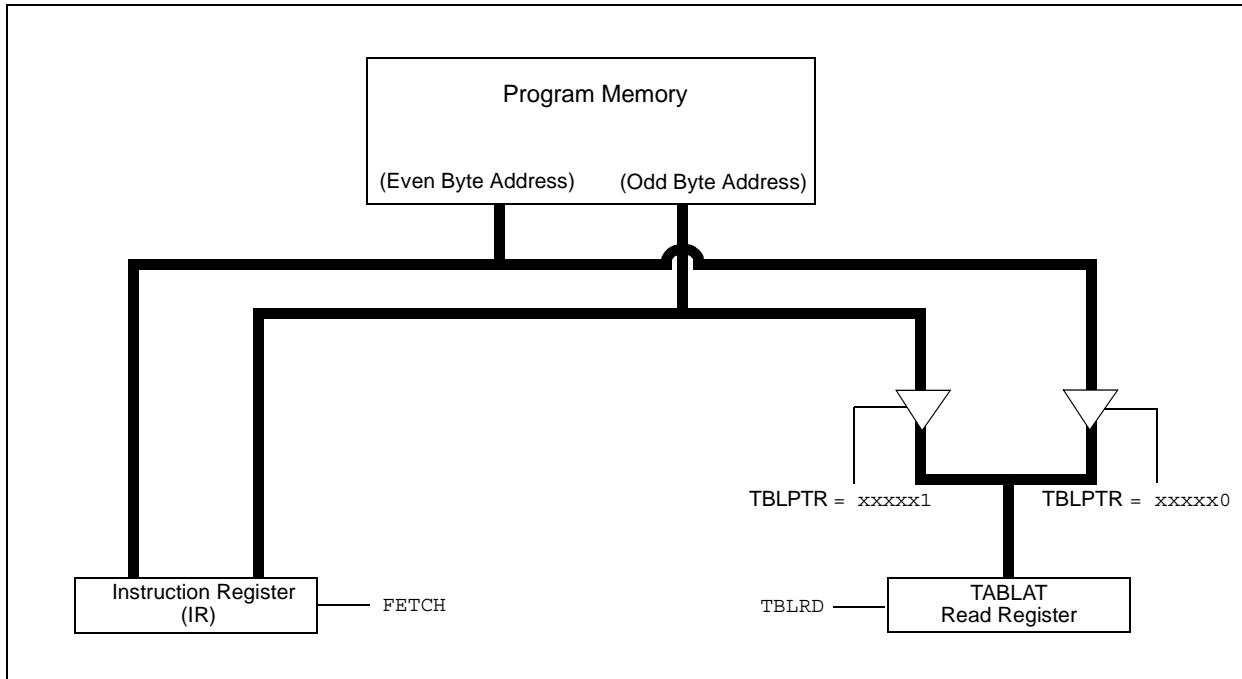
### 5.3 Reading the FLASH Program Memory

The TBLRD instruction is used to retrieve data from program memory and place 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 5-4 shows the interface between the internal program memory and the TABLAT.

**FIGURE 5-4: READS FROM FLASH PROGRAM MEMORY**



### EXAMPLE 5-1: READING A FLASH PROGRAM MEMORY WORD

```

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

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

```

## 5.4 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.

### 5.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

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

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

### EXAMPLE 5-2: ERASING A FLASH PROGRAM MEMORY ROW

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

## 5.5 Writing to FLASH Program Memory

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

Table Writes are used internally to load the holding registers needed to program the FLASH memory. There are 8 holding registers used by the Table Writes for programming.

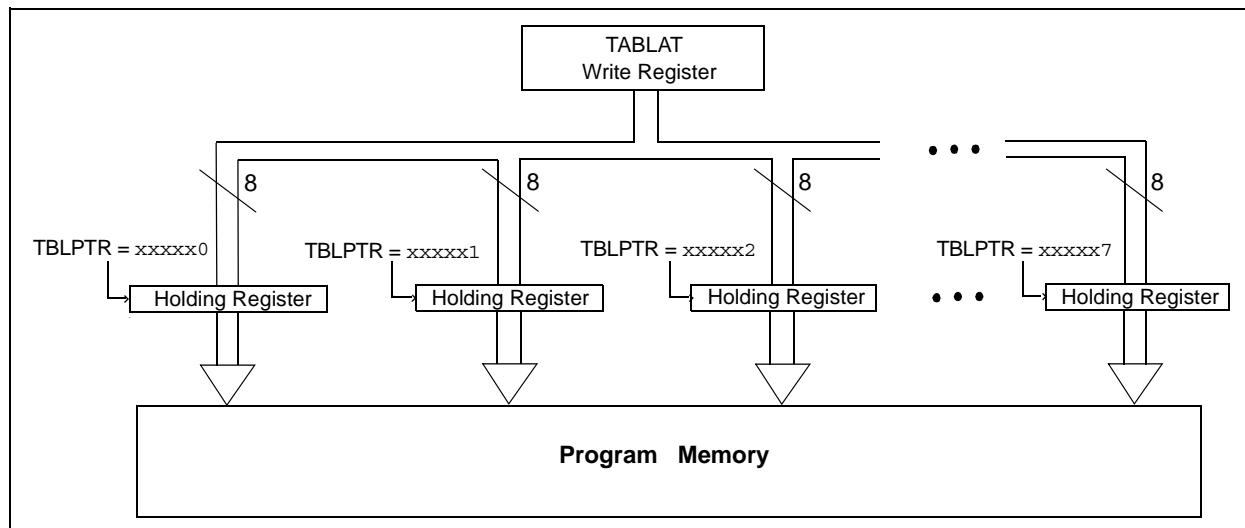
Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction has to be executed 8 times for each programming operation. All of the Table Write

operations will essentially be short writes, because only the holding registers are written. At the end of updating 8 registers, the EECON1 register must be written to, to start the programming operation with a long write.

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

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 for byte or word operations.

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



### 5.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

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

1. Read 64 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load Table Pointer with address being erased.
4. Do the row erase procedure.
5. Load Table Pointer with address of first byte being written.
6. Write the first 8 bytes into the holding registers with auto-increment (TBLWT\*+ or TBLWT+\*).
7. Set EEPGD bit to point to program memory, clear the CFGS bit to access program memory, and set WREN to enable byte writes.
8. Disable interrupts.
9. Write 55h to EECON2.

10. Write AAh to EECON2.
11. Set the WR bit. This will begin the write cycle.
12. The CPU will stall for duration of the write (about 2 ms using internal timer).
13. Re-enable interrupts.
14. Repeat steps 6-14 seven times, to write 64 bytes.
15. Verify the memory (Table Read).

This procedure will require about 18 ms to update one row of 64 bytes of memory. An example of the required code is given in Example 5-3.

**Note:** Before setting the WR bit, the table pointer address needs to be within the intended address range of the 8 bytes in the holding registers.

# PIC18FXX2

## EXAMPLE 5-3: WRITING TO FLASH PROGRAM MEMORY

```
        MOVLW  D'64                      ; number of bytes in erase block
        MOVWF  COUNTER
        MOVLW  BUFFER_ADDR_HIGH          ; point to buffer
        MOVWF  FSROH
        MOVLW  BUFFER_ADDR_LOW
        MOVWF  FSROL
        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
        DECFSZ COUNTER                 ; done?
        BRA    READ_BLOCK               ; repeat
MODIFY_WORD
        MOVLW  DATA_ADDR_HIGH          ; point to buffer
        MOVWF  FSROH
        MOVLW  DATA_ADDR_LOW
        MOVWF  FSROL
        MOVLW  NEW_DATA_LOW            ; update buffer word
        MOVWF  POSTINCO
        MOVLW  NEW_DATA_HIGH
        MOVWF  INDF0
ERASE_BLOCK
        MOVLW  CODE_ADDR_UPPER          ; load TBLPTR with the base
        MOVWF  TBLPTRU                 ; address of the memory block
        MOVLW  CODE_ADDR_HIGH
        MOVWF  TBLPTRH
        MOVLW  CODE_ADDR_LOW
        MOVWF  TBLPTRL
        BSF    EECON1, EEPGD             ; point to FLASH program memory
        BCF    EECON1, CFGS
        BSF    EECON1, WREN
        BSF    EECON1, FREE
        BCF    INTCON, GIE
        MOVLW  55h
        MOVWF  EECON2
        MOVLW  AAh
        MOVWF  EECON2
        BSF    EECON1, WR
        BSF    INTCON, GIE
        TBLRD*-
WRITE_BUFFER_BACK
        MOVLW  8                        ; number of write buffer groups of 8 bytes
        MOVWF  COUNTER_HI
        MOVLW  BUFFER_ADDR_HIGH          ; point to buffer
        MOVWF  FSROH
        MOVLW  BUFFER_ADDR_LOW
        MOVWF  FSROL
PROGRAM_LOOP
        MOVLW  8                        ; number of bytes in holding register
        MOVWF  COUNTER
WRITE_WORD_TO_HREGS
        MOVF   POSTINCO, W              ; get low byte of buffer data
        MOVWF  TABLAT                  ; present data to table latch
        TBLWT+*                         ; write data, perform a short write
                                         ; to internal TBLWT holding register.
                                         ; loop until buffers are full
        DECFSZ COUNTER
        BRA    WRITE_WORD_TO_HREGS
```

**EXAMPLE 5-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)**

PROGRAM_MEMORY			
	BSF	EECON1, EEPGD	; point to FLASH program memory
	BCF	EECON1, CFGS	; access FLASH program memory
	BSF	EECON1, WREN	; enable write to memory
	BCF	INTCON, GIE	; disable interrupts
	MOVLW	55h	
<b>Required Sequence</b>	MOVWF	EECON2	; write 55h
	MOVLW	AAh	
	MOVWF	EECON2	; write AAh
	BSF	EECON1, WR	; start program (CPU stall)
	BSF	INTCON, GIE	; re-enable interrupts
	DECFSZ	COUNTER_HI	; loop until done
	BRA	PROGRAM_LOOP	
	BCF	EECON1, WREN	; disable write to memory

**5.5.2 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.

**5.5.3 UNEXPECTED TERMINATION OF WRITE OPERATION**

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

**5.5.4 PROTECTION AGAINST SPURIOUS WRITES**

To protect against spurious writes to FLASH program memory, the write initiate sequence must also be followed. See "Special Features of the CPU" (Section 19.0) for more detail.

**5.6 FLASH Program Operation During Code Protection**

See "Special Features of the CPU" (Section 19.0) for details on code protection of FLASH program memory.

**TABLE 5-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on All Other RESETS
FF8h	TBLPTRU	—	—	bit21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					-- 00 0000	-- 00 0000
FF7h	TBPLTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	0000 0000
FF6h	TBLPTRL	Program Memory Table Pointer High Byte (TBLPTR<7:0>)								0000 0000	0000 0000
FF5h	TABLAT	Program Memory Table Latch								0000 0000	0000 0000
FF2h	INTCON	GIE/ GIEH	PEIE/ GIEL	TMROIE	INTE	RBIE	TMROIF	INTF	RBIF	0000 000x	0000 000u
FA7h	EECON2	EEPROM Control Register2 (not a physical register)								—	—
FA6h	EECON1	EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	uu-0 u000
FA2h	IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 1111
FA1h	PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
FA0h	PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000

Legend: x = unknown, u = unchanged, r = reserved, - = unimplemented read as '0'.

Shaded cells are not used during FLASH/EEPROM access.

# **PIC18FXX2**

---

---

## **NOTES:**

## 6.0 DATA EEPROM MEMORY

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

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

- EECON1
- EECON2
- EEDATA
- EEADR

The EEPROM data memory allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and EEADR holds the address of the EEPROM location being accessed. These devices have 256 bytes of data EEPROM with an address range from 0h to FFh.

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

### 6.1 EEADR

The address register can address up to a maximum of 256 bytes of data EEPROM.

### 6.2 EECON1 and EECON2 Registers

EECON1 is the control register for EEPROM memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the EEPROM write sequence.

Control bits RD and WR initiate read and write operations, respectively. These bits cannot be cleared, only set, in software. They are cleared in hardware at the completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental or premature termination of a write operation.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a MCLR Reset, or a WDT Time-out Reset during normal operation. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR), due to the RESET condition forcing the contents of the registers to zero.

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

# PIC18FXX2

---

## REGISTER 6-1: EECON1 REGISTER (ADDRESS FA6h)

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	WREN	WR	RD

bit 7

bit 0

- 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 EE or Configuration Select bit  
1 = Access Configuration or Calibration 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 EE Error Flag bit  
1 = A write operation is prematurely terminated  
      (any MCLR or any WDT Reset during self-timed programming in normal operation)  
0 = The write operation completed  
**Note:** When a WRERR occurs, the EEPGD or FREE bits are not cleared. This allows tracing of the error condition.
- bit 2      **WREN:** FLASH Program/Data EE Write Enable bit  
1 = Allows write cycles  
0 = Inhibits write to the 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.)  
0 = Does not initiate an EEPROM read

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

### 6.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADR register, clear the EEPGD control bit (EECON1<7>), clear the CFGS control bit

(EECON1<6>), and then set control bit RD (EECON1<0>). The data is available for the very next instruction cycle; therefore, the EEDATA register can be read by the next instruction. EEDATA will hold this value until another read operation, or until it is written to by the user (during a write operation).

#### EXAMPLE 6-1: DATA EEPROM READ

```

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

```

### 6.4 Writing to the Data EEPROM Memory

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

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

Additionally, the WREN bit in EECON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code exe-

cution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, EECON1, EEADR and EDATA 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 Write Complete Interrupt Flag bit (EEIF) is set. The user may either enable this interrupt, or poll this bit. EEIF must be cleared by software.

#### EXAMPLE 6-2: DATA EEPROM WRITE

	MOVLW  DATA_EE_ADDR      ;	
	MOVWF  EEADR              ; Data Memory Address to read	
	MOVLW  DATA_EE_DATA       ;	
	MOVWF  EEDATA              ; Data Memory Value to write	
	BCF    EECON1, EEPGD       ; Point to DATA memory	
	BCF    EECON1, CFGS        ; Access program FLASH or Data EEPROM memory	
	BSF    EECON1, WREN         ; Enable writes	
<b>Required Sequence</b>	BCF    INTCON, GIE          ; Disable interrupts	
	MOVLW  55h                  ;	
	MOVWF  EECON2              ; Write 55h	
	MOVLW  AAh                  ;	
	MOVWF  EECON2              ; Write AAh	
	BSF    EECON1, WR            ; Set WR bit to begin write	
	BSF    INTCON, GIE          ; Enable interrupts	
	.	;
	.	;
	.	;
	BCF    EECON1, WREN         ; Disable writes on write complete (EEIF set)	

## 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 Protection Against Spurious Write

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

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

## 6.7 Operation During Code Protect

Data EEPROM memory has its own code protect mechanism. External Read and Write operations are disabled if either of these mechanisms are enabled.

The microcontroller itself can both read and write to the internal Data EEPROM, regardless of the state of the code protect configuration bit. Refer to "Special Features of the CPU" (Section 19.0) for additional information.

## 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 rarely, an array refresh is likely not required. See specification D124.

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

```
        clrf    EEADR           ; Start at address 0
        bcf    EECON1,CFG5       ; 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             ;
        movwf   EECON2          ; Read current address
        movlw   AAh              ;
        movwf   EECON2          ; Write 55h
        movwf   EECON2          ; Write AAh
        bsf    EECON1,WR        ; Set WR bit to begin write
        btfsc  EECON1,WR        ; Wait for write to complete
        bra    $-2              ;
        incfsz EEADR,F         ; Increment address
        bra    Loop             ; Not zero, do it again
        bcf    EECON1,WREN       ; Disable writes
        bsf    INTCON,GIE        ; Enable interrupts
```

**TABLE 6-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY**

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on All Other RESETS
FF2h	INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
FA9h	EEADR	EEPROM Address Register							0000 0000	0000 0000	
FA8h	EEDATA	EEPROM Data Register							0000 0000	0000 0000	
FA7h	EECON2	EEPROM Control Register2 (not a physical register)							—	—	
FA6h	EECON1	EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	uu-0 u000
FA2h	IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 1111
FA1h	PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
FA0h	PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000

Legend: x = unknown, u = unchanged, r = reserved, - = unimplemented, read as '0'.

Shaded cells are not used during FLASH/EEPROM access.

# **PIC18FXX2**

---

---

## **NOTES:**

## 7.0 8 X 8 HARDWARE MULTIPLIER

### 7.1 Introduction

An 8 x 8 hardware multiplier is included in the ALU of the PIC18FXX2 devices. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored into the 16-bit product register pair (PRODH:PRODL). The multiplier does not affect any flags in the ALUSTA register.

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

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

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

Table 7-1 shows a performance comparison between enhanced devices using the single cycle hardware multiply, and performing the same function without the hardware multiply.

**TABLE 7-1: PERFORMANCE COMPARISON**

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 40 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	6.9 $\mu$ 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	24	24	2.4 $\mu$ s	9.6 $\mu$ s	24 $\mu$ s
16 x 16 signed	Without hardware multiply	52	254	25.4 $\mu$ s	102.6 $\mu$ s	254 $\mu$ s
	Hardware multiply	36	36	3.6 $\mu$ s	14.4 $\mu$ s	36 $\mu$ s

### 7.2 Operation

Example 7-1 shows the sequence to do an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 7-2 shows the sequence to do an 8 x 8 signed multiply. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

#### EXAMPLE 7-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

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

#### EXAMPLE 7-2: 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
```

Example 7-3 shows the sequence to do a 16 x 16 unsigned multiply. Equation 7-1 shows the algorithm that is used. The 32-bit result is stored in four registers, RES3:RES0.

#### EQUATION 7-1: 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 7-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE**

```

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

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

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

; MOVF ARG1H, W    ;
MULWF ARG2L      ; ARG1H * ARG2L ->
                  ; PRODH:PRODL
MOVF PRODL, W    ;
ADDWF RES1, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;
;
```

Example 7-4 shows the sequence to do a 16 x 16 signed multiply. Equation 7-2 shows the algorithm used. The 32-bit result is stored in four registers, RES3:RES0. To account for the sign bits of the arguments, each argument pairs Most Significant bit (MSb) is tested and the appropriate subtractions are done.

**EQUATION 7-2: 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 7-4: 16 x 16 SIGNED MULTIPLY ROUTINE**

```

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

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

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

; MOVF ARG1H, W    ;
MULWF ARG2L      ; ARG1H * ARG2L ->
                  ; PRODH:PRODL
MOVF PRODL, W    ;
ADDWF RES1, F    ; Add cross
MOVF PRODH, W    ; products
ADDWFC RES2, F   ;
CLRF WREG        ;
ADDWFC RES3, F   ;

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

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

; CONT_CODE
:
```

## 8.0 INTERRUPTS

The PIC18FXX2 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will override 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.

Each interrupt source, except INT0, has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
- Priority bit to select high priority or low priority

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

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

When an interrupt is responded to, the Global Interrupt Enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High priority interrupt sources can interrupt a low priority interrupt.

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

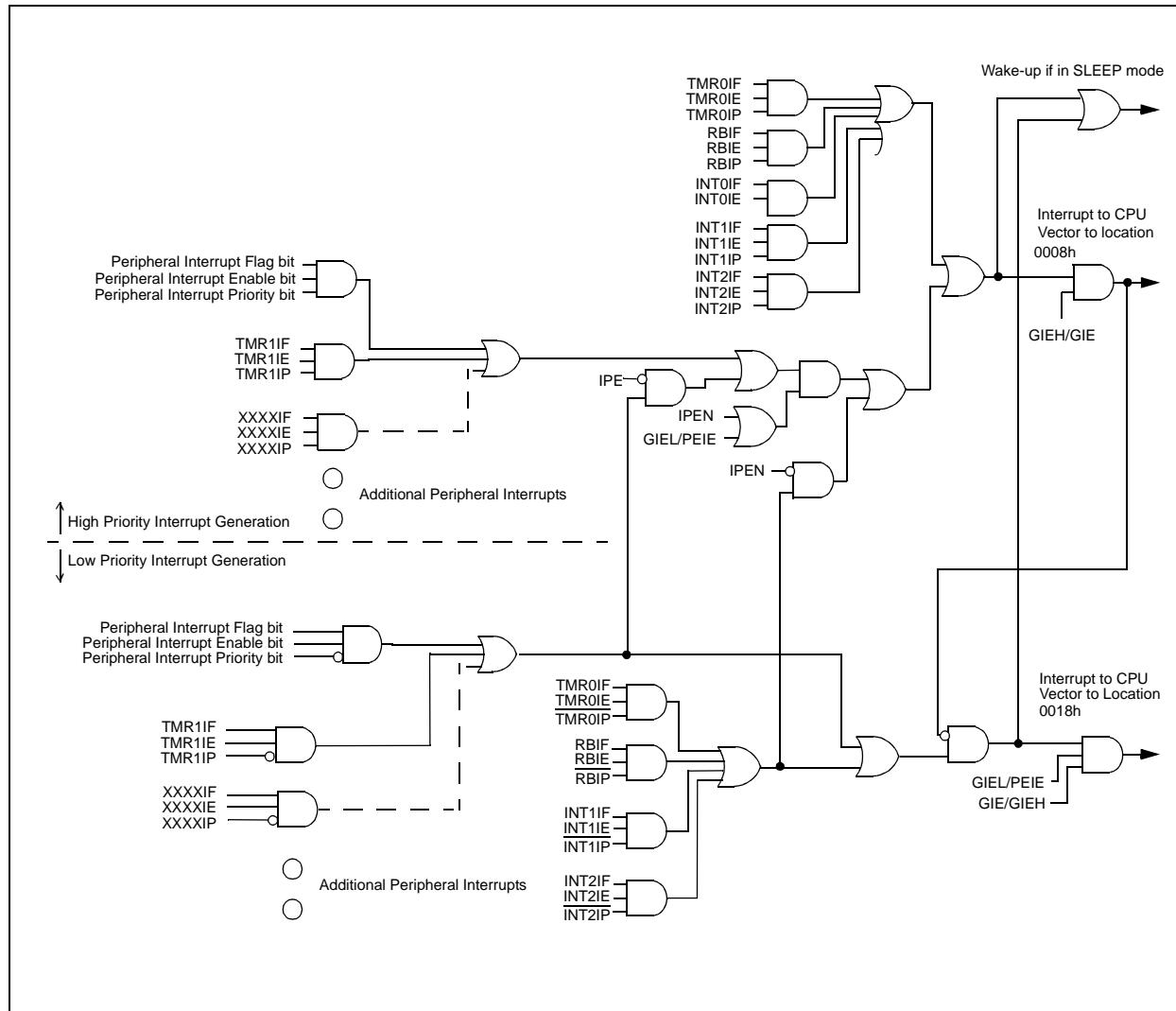
The “return from interrupt” instruction, RETFIE, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

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

<b>Note:</b>	Do not use the MOVFF instruction to modify any of the Interrupt control registers while <b>any</b> interrupt is enabled. Doing so may cause erratic microcontroller behavior.
--------------	---

# PIC18FXX2

**FIGURE 8-1: INTERRUPT LOGIC**



## 8.1 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 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 8-1: INTCON 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	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	
bit 7								bit 0

- |       |  |
|-------|--|
| bit 7 | <b>GIE/GIEH:</b> Global Interrupt Enable bit<br><u>When IPEN = 0:</u><br>1 = Enables all unmasked interrupts<br>0 = Disables all interrupts<br><u>When IPEN = 1:</u><br>1 = Enables all high priority interrupts<br>0 = Disables all interrupts  |
| bit 6 | <b>PEIE/GIEL:</b> Peripheral Interrupt Enable bit<br><u>When IPEN = 0:</u><br>1 = Enables all unmasked peripheral interrupts<br>0 = Disables all peripheral interrupts<br><u>When IPEN = 1:</u><br>1 = Enables all low priority peripheral interrupts<br>0 = Disables all low priority peripheral interrupts |
| bit 5 | <b>TMR0IE:</b> TMR0 Overflow Interrupt Enable bit<br>1 = Enables the TMR0 overflow interrupt<br>0 = Disables the TMR0 overflow interrupt   |
| bit 4 | <b>INT0IE:</b> INT0 External Interrupt Enable bit<br>1 = Enables the INT0 external interrupt<br>0 = Disables the INT0 external interrupt   |
| bit 3 | <b>RBIE:</b> RB Port Change Interrupt Enable bit<br>1 = Enables the RB port change interrupt<br>0 = Disables the RB port change interrupt  |
| bit 2 | <b>TMR0IF:</b> TMR0 Overflow Interrupt Flag bit<br>1 = TMR0 register has overflowed (must be cleared in software)<br>0 = TMR0 register did not overflow  |
| bit 1 | <b>INT0IF:</b> INT0 External Interrupt Flag bit<br>1 = The INT0 external interrupt occurred (must be cleared in software)<br>0 = The INT0 external interrupt did not occur   |
| bit 0 | <b>RBIF:</b> RB Port Change Interrupt Flag bit<br>1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)<br>0 = None of the RB7:RB4 pins have changed state  |

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

#### 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

# PIC18FXX2

## REGISTER 8-2: INTCON2 REGISTER

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

- 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 Interrupt0 Edge Select bit  
1 = Interrupt on rising edge  
0 = Interrupt on falling edge
- bit 5 **INTEDG1:** External Interrupt1 Edge Select bit  
1 = Interrupt on rising edge  
0 = Interrupt on falling edge
- bit 4 **INTEDG2:** External Interrupt2 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

### 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

**Note:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global 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 8-3: INTCON3 REGISTER

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

- 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

## 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

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

## 8.2 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 Flag Registers (PIR1, PIR2).

**Note 1:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global 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 8-4: 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

- bit 7      **PSPIF<sup>(1)</sup>:** Parallel Slave Port Read/Write Interrupt Flag bit  
1 = A read or a write operation has taken place (must be cleared in software)  
0 = No read or write has occurred
- bit 6      **ADIF:** A/D Converter Interrupt Flag bit  
1 = An A/D conversion completed (must be cleared in software)  
0 = The A/D conversion is not complete
- bit 5      **RCIF:** USART Receive Interrupt Flag bit  
1 = The USART receive buffer, RCREG, is full (cleared when RCREG is read)  
0 = The USART receive buffer is empty
- bit 4      **TXIF:** USART Transmit Interrupt Flag bit (see Section 16.0 for details on TXIF functionality)  
1 = The USART transmit buffer, TXREG, is empty (cleared when TXREG is written)  
0 = The USART transmit buffer is full
- bit 3      **SSPIF:** Master Synchronous Serial Port Interrupt Flag bit  
1 = The transmission/reception is complete (must be cleared in software)  
0 = Waiting to transmit/receive
- bit 2      **CCP1IF:** CCP1 Interrupt Flag bit  
Capture mode:  
1 = A TMR1 register capture occurred (must be cleared in software)  
0 = No TMR1 register capture occurred  
Compare mode:  
1 = A TMR1 register compare match occurred (must be cleared in software)  
0 = No TMR1 register compare match occurred  
PWM mode:  
Unused in this mode
- bit 1      **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit  
1 = TMR2 to PR2 match occurred (must be cleared in software)  
0 = No TMR2 to PR2 match occurred
- bit 0      **TMR1IF:** TMR1 Overflow Interrupt Flag bit  
1 = TMR1 register overflowed (must be cleared in software)  
0 = TMR1 register did not overflow

**Note 1:** This bit is reserved on PIC18F2X2 devices; always maintain this bit clear.

#### 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

## REGISTER 8-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
bit 7	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF

bit 7-5

**Unimplemented:** Read as '0'

bit 4

**EEIF:** Data EEPROM/FLASH Write Operation Interrupt Flag bit1 = 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 bit1 = A bus collision occurred (must be cleared in software)  
0 = No bus collision occurred

bit 2

**LVDIF:** Low Voltage Detect Interrupt Flag bit1 = A low voltage condition occurred (must be cleared in software)  
0 = The device voltage is above the Low Voltage Detect trip point

bit 1

**TMR3IF:** TMR3 Overflow Interrupt Flag bit1 = TMR3 register overflowed (must be cleared in software)  
0 = TMR3 register did not overflow

bit 0

**CCP2IF:** CCPx Interrupt Flag bitCapture mode:1 = A TMR1 register capture occurred (must be cleared in software)  
0 = No TMR1 register capture occurredCompare mode:1 = A TMR1 register compare match occurred (must be cleared in software)  
0 = No TMR1 register compare match occurredPWM mode:

Unused in this mode

## 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

# PIC18FXX2

---

## 8.3 PIE Registers

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

**REGISTER 8-6: 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	R/W-0
PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	

bit 7

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

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:** USART Receive Interrupt Enable bit

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

bit 4

**TXIE:** USART Transmit Interrupt Enable bit

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

bit 3

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

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

bit 2

**CCP1IE:** CCP1 Interrupt Enable bit

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

bit 1

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

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

bit 0

**TMR1IE:** TMR1 Overflow Interrupt Enable bit

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

**Note 1:** This bit is reserved on PIC18F2X2 devices; always maintain this bit clear.

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

## REGISTER 8-7: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
bit 7	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE

- bit 7-5      **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          **LVDIE:** Low Voltage Detect Interrupt Enable bit  
1 = Enabled  
0 = Disabled
- bit 1          **TMR3IE:** TMR3 Overflow Interrupt Enable bit  
1 = Enables the TMR3 overflow interrupt  
0 = Disables the TMR3 overflow interrupt
- bit 0          **CCP2IE:** CCP2 Interrupt Enable bit  
1 = Enables the CCP2 interrupt  
0 = Disables the CCP2 interrupt

## 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

# **PIC18FXX2**

## 8.4 IPR Registers

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

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

- |       |  |
|-------|--|
| bit 7 | <b>PSPIP<sup>(1)</sup>:</b> Parallel Slave Port Read/Write Interrupt Priority bit<br>1 = High priority<br>0 = Low priority |
| bit 6 | <b>ADIP:</b> A/D Converter Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                                 |
| bit 5 | <b>RCIP:</b> USART Receive Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                                 |
| bit 4 | <b>TXIP:</b> USART Transmit Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                                |
| bit 3 | <b>SSPIP:</b> Master Synchronous Serial Port Interrupt Priority bit<br>1 = High priority<br>0 = Low priority               |
| bit 2 | <b>CCP1IP:</b> CCP1 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority  |
| bit 1 | <b>TMR2IP:</b> TMR2 to PR2 Match Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                           |
| bit 0 | <b>TMR1IP:</b> TMR1 Overflow Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                               |

**Note 1:** This bit is reserved on PIC18F2X2 devices; always maintain this bit set.

**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

**REGISTER 8-9: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2**

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP

- |         |  |
|---------|--|
| bit 7-5 | <b>Unimplemented:</b> Read as '0'  |
| bit 4   | <b>EEIP:</b> Data EEPROM/FLASH Write Operation Interrupt Priority bit<br>1 = High priority<br>0 = Low priority |
| bit 3   | <b>BCLIP:</b> Bus Collision Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                    |
| bit 2   | <b>LVDIP:</b> Low Voltage Detect Interrupt Priority bit<br>1 = High priority<br>0 = Low priority               |
| bit 1   | <b>TMR3IP:</b> TMR3 Overflow Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                   |
| bit 0   | <b>CCP2IP:</b> CCP2 Interrupt Priority bit<br>1 = High priority<br>0 = Low priority                            |

**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

# **PIC18FXX2**

## 8.5 RCON Register

The RCON register contains the bit which is used to enable prioritized interrupts (IPEN).

## **REGISTER 8-10: RCON REGISTER**

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	RI	TO	PD	POR	BOR
bit 7							bit 0

- |         |   |
|---------|---|
| bit 7   | <b>IPEN:</b> Interrupt Priority Enable bit<br>1 = Enable priority levels on interrupts<br>0 = Disable priority levels on interrupts (16CXXX Compatibility mode) |
| bit 6-5 | <b>Unimplemented:</b> Read as '0'   |
| bit 4   | <b>RI:</b> RESET Instruction Flag bit<br>For details of bit operation, see Register 4-3   |
| bit 3   | <b>TO:</b> Watchdog Time-out Flag bit<br>For details of bit operation, see Register 4-3   |
| bit 2   | <b>PD:</b> Power-down Detection Flag bit<br>For details of bit operation, see Register 4-3  |
| bit 1   | <b>POR:</b> Power-on Reset Status bit<br>For details of bit operation, see Register 4-3   |
| bit 0   | <b>BOR:</b> Brown-out Reset Status bit<br>For details of bit operation, see Register 4-3  |

**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

## 8.6 INT0 Interrupt

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

## 8.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow (FFh → 00h) in the TMR0 register will set flag bit TMR0IF. In 16-bit mode, an overflow (FFFFh → 0000h) in the TMR0H:TMR0L registers will set flag bit TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit T0IE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit TMR0IP (INTCON2<2>). See Section 10.0 for further details on the Timer0 module.

## 8.8 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>).

## 8.9 Context Saving During Interrupts

During an interrupt, the return PC value 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 4.3), the user may need to save the WREG, STATUS and BSR registers in software. Depending on the user's application, other registers may also need to be saved. Equation 8-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

### EXAMPLE 8-1: 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 located anywhere
;
; USER ISR CODE
;
MOVFF  BSR_TEMP,  BSR    ; Restore BSR
MOVF   W_TEMP,      W     ; Restore WREG
MOVFF  STATUS_TEMP, STATUS ; Restore STATUS
```

# **PIC18FXX2**

---

---

## **NOTES:**

## 9.0 I/O PORTS

Depending on the device selected, there are either five ports or three 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.

### 9.1 PORTA, TRISA and LATA Registers

PORTA is a 7-bit wide, bi-directional 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 Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

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

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

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

The other PORTA pins are multiplexed with analog inputs and the analog VREF+ and VREF- inputs. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

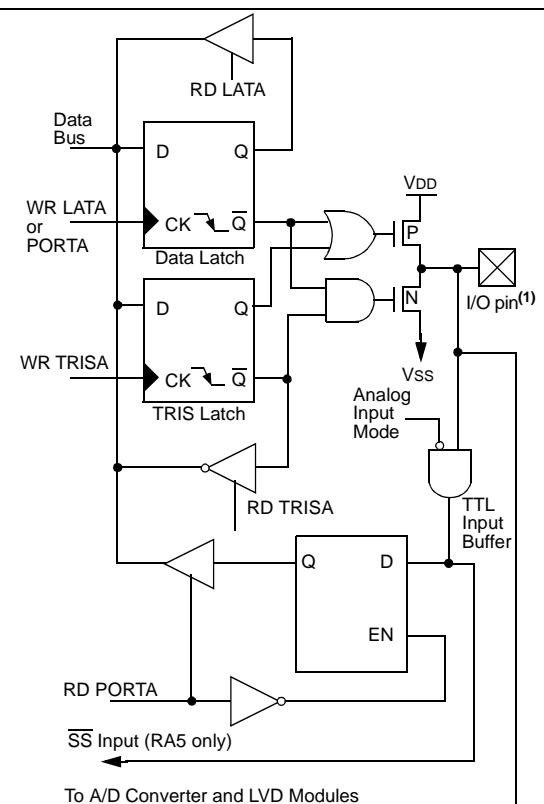
**Note:** On a Power-on Reset, RA5 and RA3:RA0 are configured as analog inputs and read as '0'. RA6 and RA4 are configured as digital inputs.

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

### EXAMPLE 9-1: INITIALIZING PORTA

```
CLRF PORTA      ; Initialize PORTA by
; clearing output
; data latches
CLRF LATA       ; Alternate method
; to clear output
; data latches
MOVLW 0x07      ; Configure A/D
MOVWF ADCON1    ; for digital inputs
MOVLW 0xcf      ; Value used to
; initialize data
; direction
MOVWF TRISA     ; Set RA<3:0> as inputs
; RA<5:4> as outputs
```

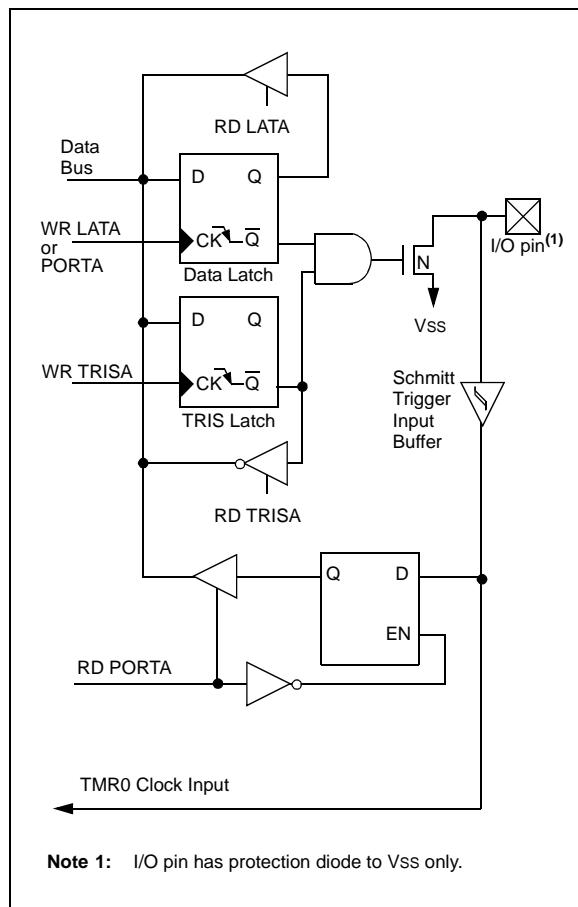
**FIGURE 9-1: BLOCK DIAGRAM OF RA3:RA0 AND RA5 PINS**



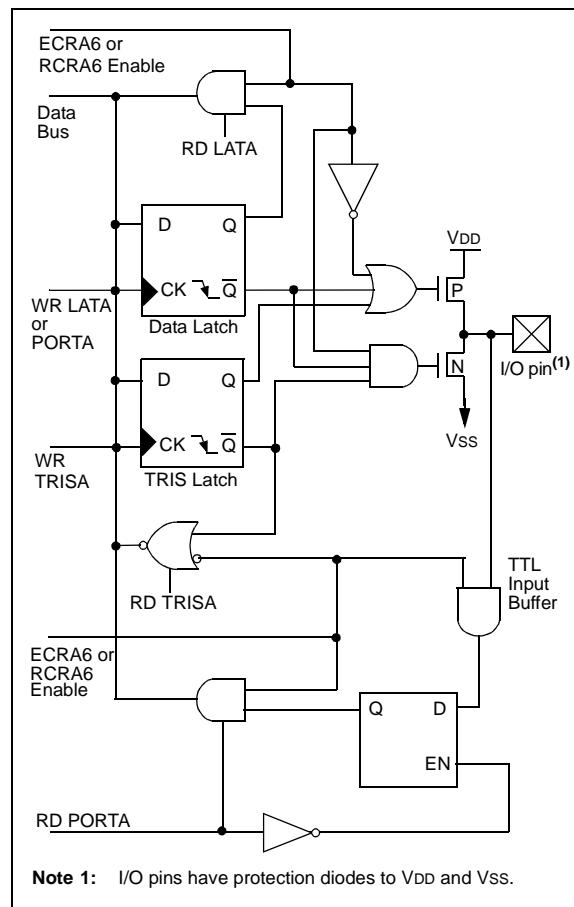
**Note 1:** I/O pins have protection diodes to VDD and Vss.

# PIC18FXX2

**FIGURE 9-2: BLOCK DIAGRAM OF RA4/T0CKI PIN**



**FIGURE 9-3: BLOCK DIAGRAM OF RA6 PIN**



**TABLE 9-1: PORTA FUNCTIONS**

Name	Bit#	Buffer	Function
RA0/AN0	bit0	TTL	Input/output or analog input.
RA1/AN1	bit1	TTL	Input/output or analog input.
RA2/AN2/VREF-	bit2	TTL	Input/output or analog input or VREF-.
RA3/AN3/VREF+	bit3	TTL	Input/output or analog input or VREF+.
RA4/T0CKI	bit4	ST	Input/output or external clock input for Timer0. Output is open drain type.
RA5/SS/AN4/LVDIN	bit5	TTL	Input/output or slave select input for synchronous serial port or analog input, or low voltage detect input.
OSC2/CLKO/RA6	bit6	TTL	OSC2 or clock output or I/O pin.

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

**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	Value on POR, BOR	Value on All Other RESETS
PORTA	—	RA6	RA5	RA4	RA3	RA2	RA1	RA0	-x0x 0000	-uuu 0000
LATA	—	LATA Data Output Register							-xxx xxxx	-uuu uuuu
TRISA	—	PORTA Data Direction Register							-111 1111	-111 1111
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

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

## 9.2 PORTB, TRISB and LATB Registers

PORTB is an 8-bit wide, bi-directional 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 Hi-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 reads and writes 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 0xCF      ; Value used to
                  ; initialize data
                  ; direction
MOVWF TRISB     ; Set RB<3:0> as inputs
                  ; RB<5:4> as outputs
                  ; RB<7:6> as inputs

```

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, these pins are configured as digital inputs.

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

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

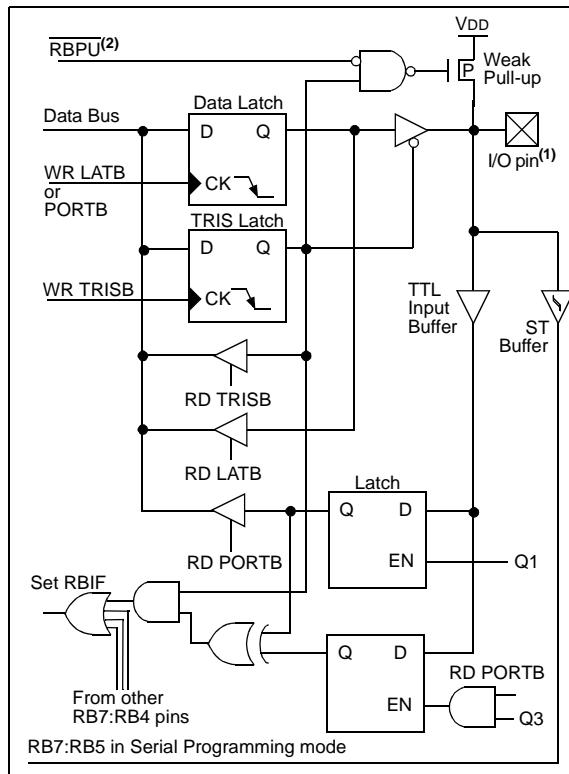
- Any read or write of PORTB (except with the MOVFF instruction). This will end the mismatch condition.
- Clear flag bit RBIF.

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

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

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

**FIGURE 9-4: BLOCK DIAGRAM OF RB7:RB4 PINS**



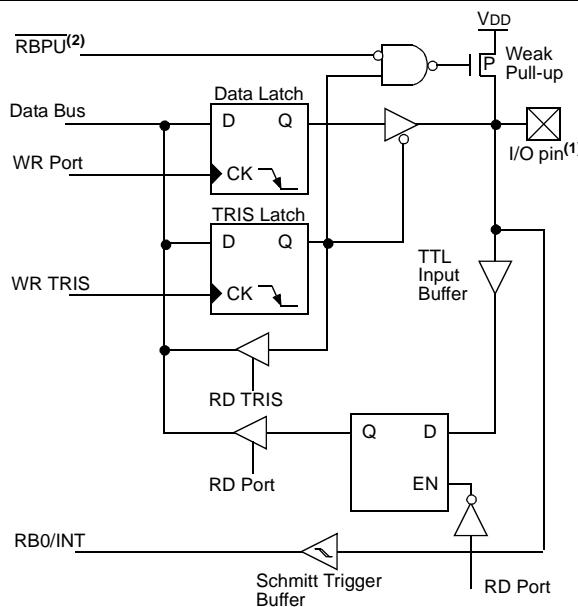
**Note 1:** I/O pins have diode protection to VDD and Vss.

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

**Note 1:** While in Low Voltage ICSP mode, the RB5 pin can no longer be used as a general purpose I/O pin, and should be held low during normal operation to protect against inadvertent ICSP mode entry.

**2:** When using Low Voltage ICSP programming (LVP), the pull-up on RB5 becomes disabled. If TRISB bit 5 is cleared, thereby setting RB5 as an output, LATB bit 5 must also be cleared for proper operation.

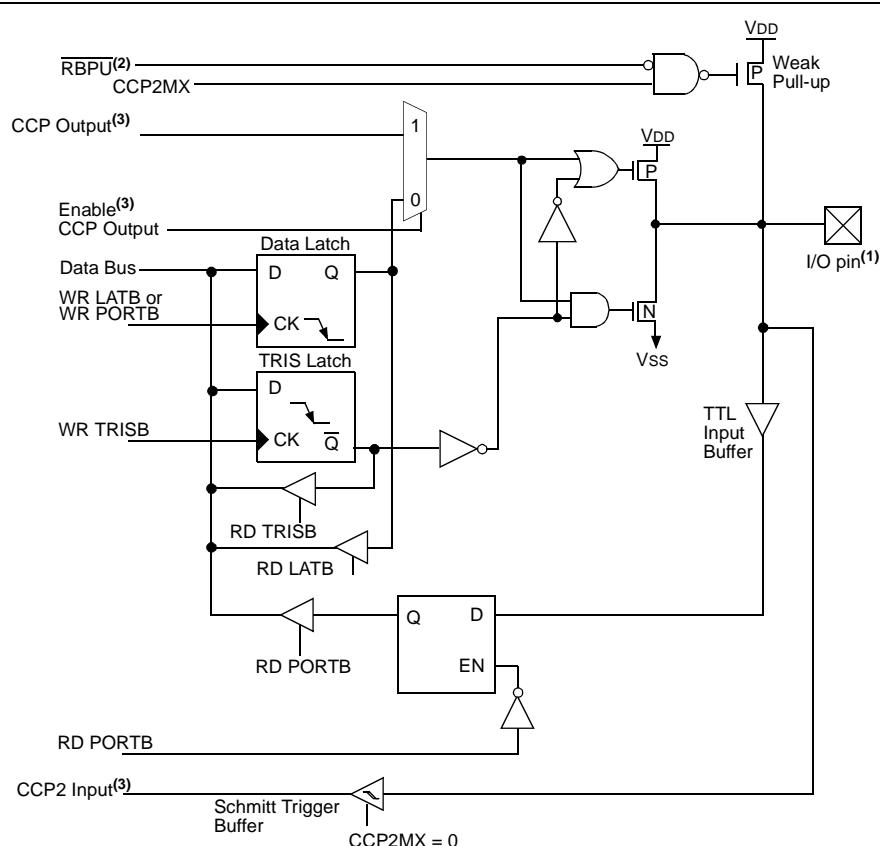
**FIGURE 9-5: BLOCK DIAGRAM OF RB2:RB0 PINS**



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

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

**FIGURE 9-6: BLOCK DIAGRAM OF RB3 PIN**



**Note 1:** I/O pin has diode protection to VDD and Vss.

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

**3:** The CCP2 input/output is multiplexed with RB3 if the CCP2MX bit is enabled (=0) in the configuration register.

# PIC18FXX2

---

**TABLE 9-3: PORTB FUNCTIONS**

Name	Bit#	Buffer	Function
RB0/INT0	bit0	TTL/ST <sup>(1)</sup>	Input/output pin or external interrupt input0. Internal software programmable weak pull-up.
RB1/INT1	bit1	TTL/ST <sup>(1)</sup>	Input/output pin or external interrupt input1. Internal software programmable weak pull-up.
RB2/INT2	bit2	TTL/ST <sup>(1)</sup>	Input/output pin or external interrupt input2. Internal software programmable weak pull-up.
RB3/CCP2 <sup>(3)</sup>	bit3	TTL/ST <sup>(4)</sup>	Input/output pin or Capture2 input/Compare2 output/PWM output when CCP2MX configuration bit is enabled. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5/PGM <sup>(5)</sup>	bit5	TTL/ST <sup>(2)</sup>	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Low voltage ICSP enable pin.
RB6/PGC	bit6	TTL/ST <sup>(2)</sup>	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit7	TTL/ST <sup>(2)</sup>	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming data.

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

- Note 1:** This buffer is a Schmitt Trigger input when configured as the external interrupt.
- 2:** This buffer is a Schmitt Trigger input when used in Serial Programming mode.
- 3:** A device configuration bit selects which I/O pin the CCP2 pin is multiplexed on.
- 4:** This buffer is a Schmitt Trigger input when configured as the CCP2 input.
- 5:** Low Voltage ICSP Programming (LVP) is enabled by default, which disables the RB5 I/O function. LVP must be disabled to enable RB5 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

**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	Value on POR, BOR	Value on All Other RESETS
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
LATB	LATB Data Output Register								xxxx xxxx	uuuu uuuu
TRISB	PORTB Data Direction Register								1111 1111	1111 1111
INTCON	GIE/ GIEH	PEIE/ GIEL	TMROIE	INT0IE	RBIE	TMROIF	INT0IF	RBIF	0000 000x	0000 000u
INTCON2	RBU	INTEDG0	INTEDG1	INTEDG2	—	TMROIP	—	RBIP	1111 -1-1	1111 -1-1
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	11-0 0-00

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

### 9.3 PORTC, TRISC and LATC Registers

PORTC is an 8-bit wide, bi-directional 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 Hi-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 reads and writes the latched output value for PORTC.

PORTC is multiplexed with several peripheral functions (Table 9-5). PORTC pins have Schmitt Trigger input buffers.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

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

The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register, without concern due to peripheral overrides.

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

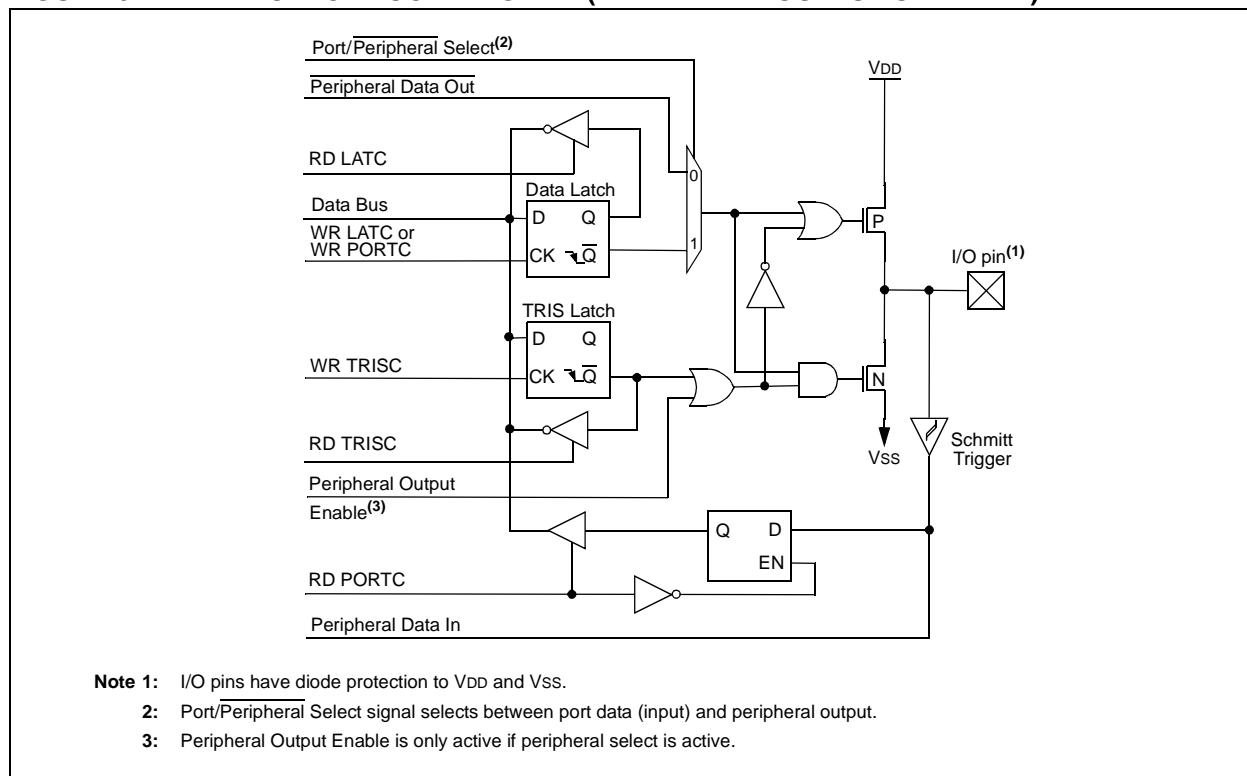
#### EXAMPLE 9-3: INITIALIZING PORTC

```

CLRF  PORTC    ; Initialize PORTC by
                 ; clearing output
                 ; data latches
CLRF  LATC     ; Alternate method
                 ; to clear output
                 ; data latches
MOVLW 0xCF    ; Value used to
                 ; initialize data
                 ; direction
MOVWF TRISC   ; Set RC<3:0> as inputs
                 ; RC<5:4> as outputs
                 ; RC<7:6> as inputs

```

**FIGURE 9-7: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE)**



# PIC18FXX2

---

**TABLE 9-5: PORTC FUNCTIONS**

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit1	ST	Input/output port pin, Timer1 oscillator input, or Capture2 input/Compare2 output/PWM output when CCP2MX configuration bit is set.
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/PWM output.
RC3/SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I <sup>2</sup> C modes.
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or Data I/O (I <sup>2</sup> C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit6	ST	Input/output port pin, Addressable USART Asynchronous Transmit, or Addressable USART Synchronous Clock.
RC7/RX/DT	bit7	ST	Input/output port pin, Addressable USART Asynchronous Receive, or Addressable USART Synchronous Data.

Legend: ST = Schmitt Trigger input

**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	Value on POR, BOR	Value on All Other RESETS
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
LATC	LATC Data Output Register								xxxx xxxx	uuuu uuuu
TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

## 9.4 PORTD, TRISD and LATD Registers

This section is applicable only to the PIC18F4X2 devices.

PORTD is an 8-bit wide, bi-directional 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 Hi-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 reads and writes the latched output value for PORTD.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

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

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

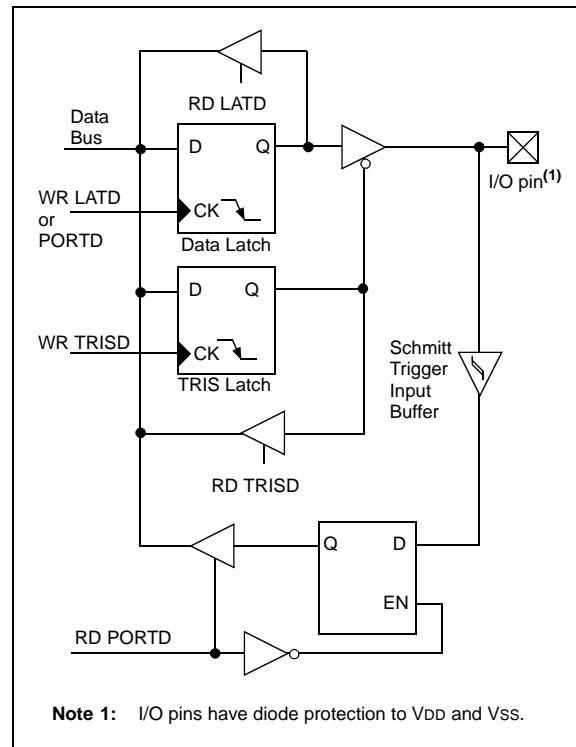
### EXAMPLE 9-4: INITIALIZING PORTD

```

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

```

**FIGURE 9-8: PORTD BLOCK DIAGRAM IN I/O PORT MODE**



# PIC18FXX2

---

**TABLE 9-7: PORTD FUNCTIONS**

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit0	ST/TTL <sup>(1)</sup>	Input/output port pin or parallel slave port bit0.
RD1/PSP1	bit1	ST/TTL <sup>(1)</sup>	Input/output port pin or parallel slave port bit1.
RD2/PSP2	bit2	ST/TTL <sup>(1)</sup>	Input/output port pin or parallel slave port bit2.
RD3/PSP3	bit3	ST/TTL <sup>(1)</sup>	Input/output port pin or parallel slave port bit3.
RD4/PSP4	bit4	ST/TTL <sup>(1)</sup>	Input/output port pin or parallel slave port bit4.
RD5/PSP5	bit5	ST/TTL <sup>(1)</sup>	Input/output port pin or parallel slave port bit5.
RD6/PSP6	bit6	ST/TTL <sup>(1)</sup>	Input/output port pin or parallel slave port bit6.
RD7/PSP7	bit7	ST/TTL <sup>(1)</sup>	Input/output port pin or parallel slave port bit7.

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

**Note 1:** Input buffers are Schmitt Triggers when in I/O mode and TTL buffer when in Parallel Slave Port mode.

**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	Value on POR, BOR	Value on All Other RESETS
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
LATD	LATD Data Output Register								xxxx xxxx	uuuu uuuu
TRISD	PORTD Data Direction Register								1111 1111	1111 1111
TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

## 9.5 PORTE, TRISE and LATE Registers

This section is only applicable to the PIC18F4X2 devices.

**PORTE** is a 3-bit wide, bi-directional port. The corresponding Data Direction register is **TRISE**. Setting a **TRISE** bit (= 1) will make the corresponding **PORTE** pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a **TRISE** bit (= 0) will make the corresponding **PORTE** pin an output (i.e., put the contents of the output latch on the selected pin).

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

PORTE has three pins (RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/AN7) which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

Register 9-1 shows the TRISE register, which also controls the parallel slave port operation.

PORTE pins are multiplexed with analog inputs. When selected as an analog input, these pins will read as '0's.

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, these pins are configured as analog inputs.

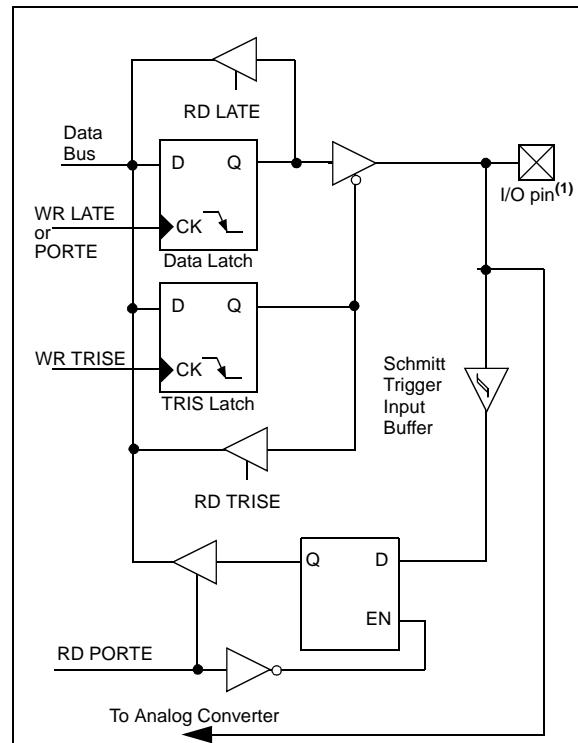
## **EXAMPLE 9-5: INITIALIZING PORTE**

```

CLRF  PORTE    ; Initialize PORTE by
              ; clearing output
              ; data latches
CLRF  LATE     ; Alternate method
              ; to clear output
              ; data latches
MOVLW 0x07    ; Configure A/D
MOVWF  ADCON1  ; for digital inputs
MOVLW 0x05    ; Value used to
                  ; initialize data
                  ; direction
MOVWF  TRISE   ; Set RE<0> as inputs
                  ; RE<1> as outputs
                  ; RE<2> as inputs

```

**FIGURE 9-9: PORTE BLOCK DIAGRAM IN I/O PORT MODE**



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

# **PIC18FXX2**

## **REGISTER 9-1: TRISE REGISTER**

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	

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

**Legend:**

R = Readable bit

- n = Value at POR

W = Writable bit

**U** = Unimplemented bit, read as '0'

'0' = Bit is cleared      x = Bit is unknown

**TABLE 9-9: PORTE FUNCTIONS**

Name	Bit#	Buffer Type	Function
RE0/RD/AN5	bit0	ST/TTL <sup>(1)</sup>	Input/output port pin or read control input in Parallel Slave Port mode or analog input: <b>RD</b> 1 = Not a read operation 0 = Read operation. Reads PORTD register (if chip selected).
RE1/WR/AN6	bit1	ST/TTL <sup>(1)</sup>	Input/output port pin or write control input in Parallel Slave Port mode or analog input: <b>WR</b> 1 = Not a write operation 0 = Write operation. Writes PORTD register (if chip selected).
RE2/CS/AN7	bit2	ST/TTL <sup>(1)</sup>	Input/output port pin or chip select control input in Parallel Slave Port mode or analog input: <b>CS</b> 1 = Device is not selected 0 = Device is selected

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

**Note 1:** Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

**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	Value on POR, BOR	Value on All Other RESETS
PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -000	---- -000
LATE	—	—	—	—	—	LATE Data Output Register			---- -xxx	---- -uuu
TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

# **PIC18FXX2**

## 9.6 Parallel Slave Port

The Parallel Slave Port is implemented on the 40-pin devices only (PIC18F4X2).

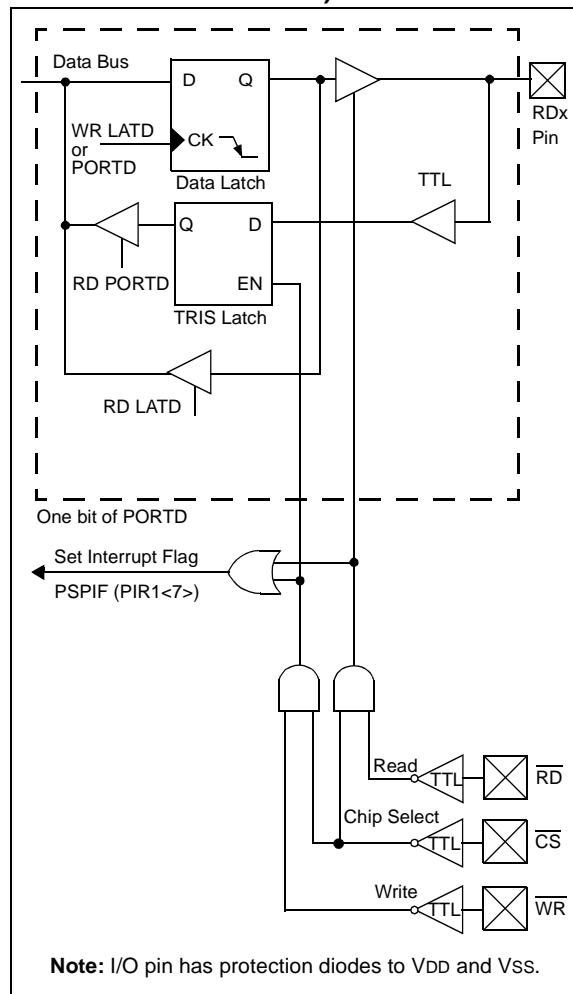
PORTD operates as an 8-bit wide Parallel Slave Port, or microprocessor port when control bit, PSPMODE (TRISE<4>) is set. It is asynchronously readable and writable by the external world through RD control input pin, RE0/RD and WR control input pin, RE1/WR.

It can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an 8-bit latch. Setting bit PSPMODE enables port pin RE0/RD to be the RD input, RE1/WR to be the WR input and RE2/CS to be 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 PCFG2:PCFG0 (ADCON1<2:0>) must be set, which will configure pins RE2:RE0 as digital I/O.

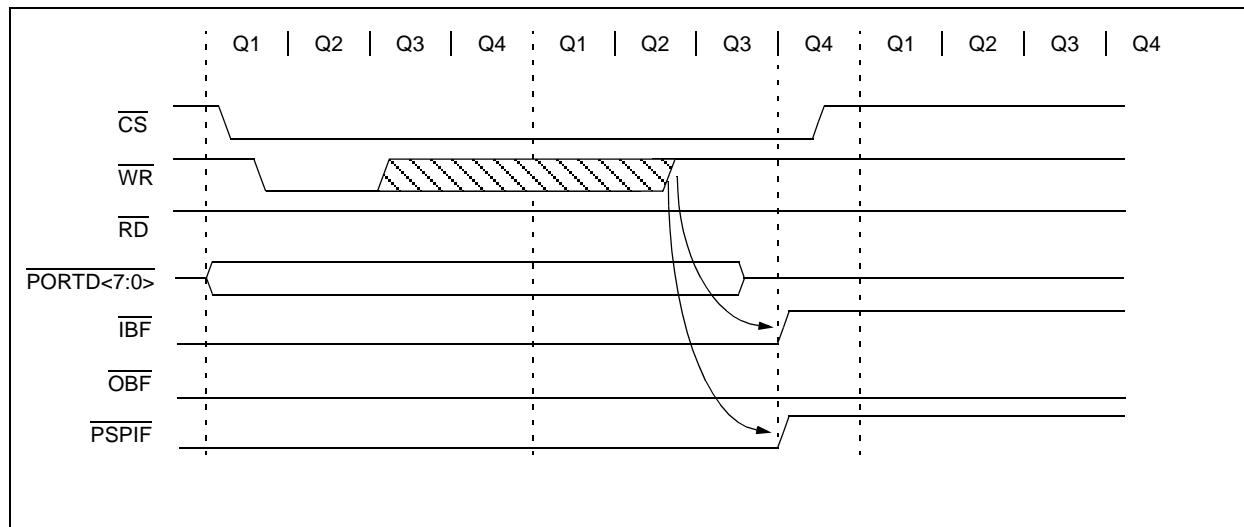
A write to the PSP occurs when both the CS and WR lines are first detected low. A read from the PSP occurs when both the CS and RD lines are first detected low.

The PORTE I/O pins become control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make sure that the TRISE<2:0> bits are set (pins are configured as digital inputs), and the ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

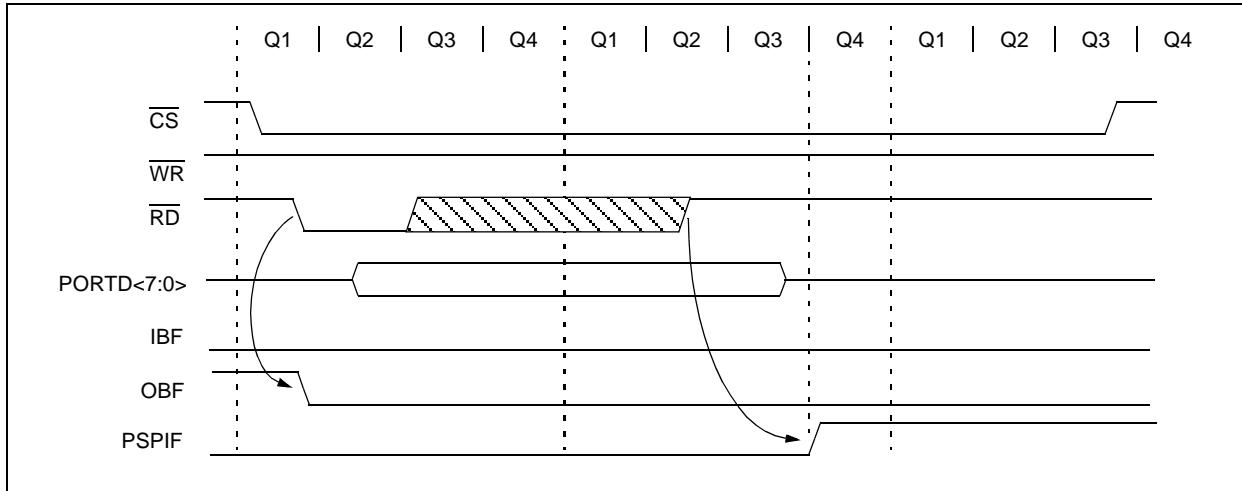
**FIGURE 9-10:** PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)



**FIGURE 9-11: PARALLEL SLAVE PORT WRITE WAVEFORMS**



**FIGURE 9-12: 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	Value on POR, BOR	Value on All Other RESETS
PORTD	Port Data Latch when written; Port pins when read								xxxx xxxx	uuuu uuuu
LATD	LATD Data Output bits								xxxx xxxx	uuuu uuuu
TRISD	PORTD Data Direction bits								1111 1111	1111 1111
PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -000	---- -000
LATE	—	—	—	—	—	LATE Data Output bits			---- -xxx	---- -uuu
TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IF	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Parallel Slave Port.

# **PIC18FXX2**

---

---

## **NOTES:**

## 10.0 TIMER0 MODULE

The Timer0 module has the following features:

- Software selectable as an 8-bit or 16-bit timer/counter
- Readable and writable
- Dedicated 8-bit software programmable prescaler
- Clock source selectable to be external or internal
- Interrupt-on-overflow from FFh to 00h in 8-bit mode and FFFFh to 0000h in 16-bit mode
- Edge select for external clock

Figure 10-1 shows a simplified block diagram of the Timer0 module in 8-bit mode and Figure 10-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

The T0CON register (Register 10-1) is a readable and writable register that controls all the aspects of Timer0, including the prescale selection.

### REGISTER 10-1: 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

- bit 7      **TMR0ON:** Timer0 On/Off Control bit  
               1 = Enables Timer0  
               0 = Stops Timer0
- bit 6      **T08BIT:** Timer0 8-bit/16-bit Control bit  
               1 = Timer0 is configured as an 8-bit timer/counter  
               0 = Timer0 is configured as a 16-bit timer/counter
- bit 5      **T0CS:** Timer0 Clock Source Select bit  
               1 = Transition on T0CKI pin  
               0 = Internal instruction cycle clock (CLKO)
- bit 4      **T0SE:** Timer0 Source Edge Select bit  
               1 = Increment on high-to-low transition on T0CKI pin  
               0 = Increment on low-to-high transition on T0CKI pin
- bit 3      **PSA:** Timer0 Prescaler Assignment bit  
               1 = Timer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.  
               0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.
- bit 2-0     **T0PS2:T0PS0:** 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

#### 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

# PIC18FXX2

FIGURE 10-1: TIMER0 BLOCK DIAGRAM IN 8-BIT MODE

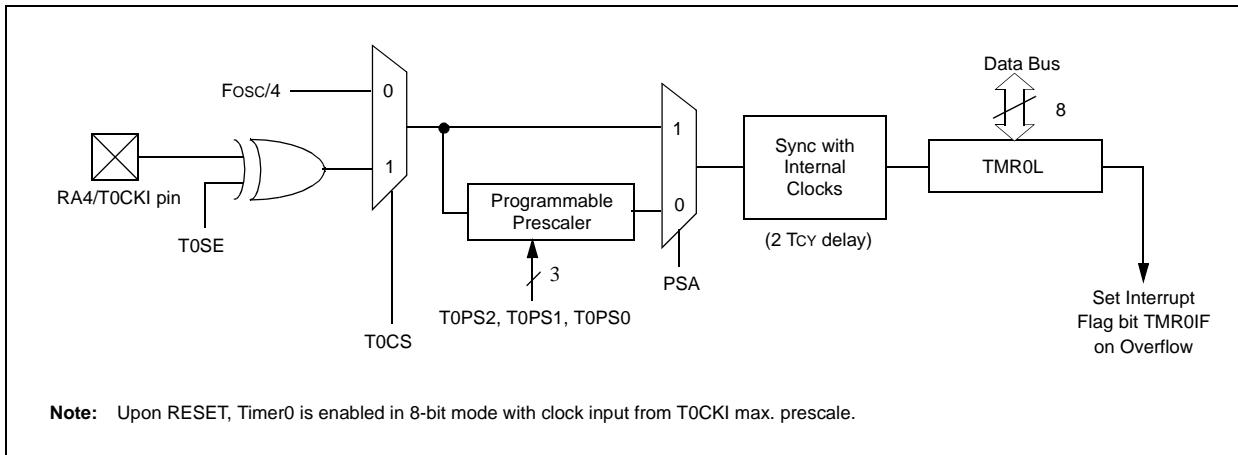
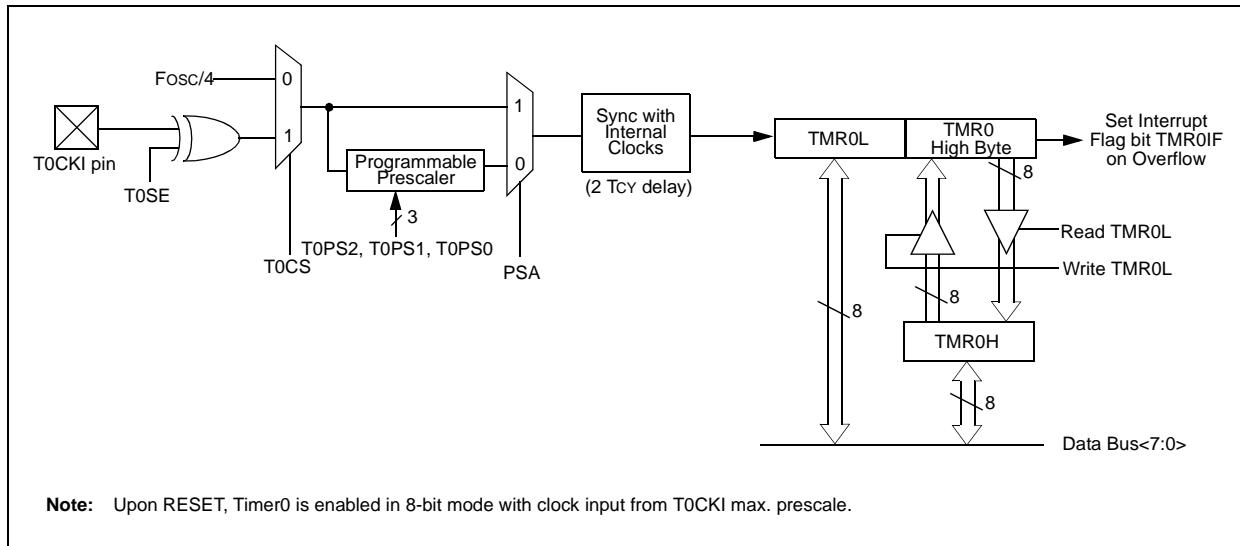


FIGURE 10-2: TIMER0 BLOCK DIAGRAM IN 16-BIT MODE



## 10.1 Timer0 Operation

Timer0 can operate as a timer or as a counter.

Timer mode is selected by clearing the T0CS bit. In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0L register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0L register.

Counter mode is selected by setting the T0CS bit. In Counter mode, Timer0 will increment, either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit (T0SE). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed below.

When an external clock input is used for Timer0, it must meet certain requirements. The requirements ensure the external clock can be synchronized with the internal phase clock (Tosc). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

## 10.2 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not readable or writable.

The PSA and T0PS2:T0PS0 bits determine the prescaler assignment and prescale ratio.

Clearing bit PSA will assign the prescaler to the Timer0 module. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4,..., 1:256 are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0L register (e.g., CLRF TMR0, MOVWF TMR0, BSF TMR0, x....etc.) will clear the prescaler count.

**Note:** Writing to TMR0L when the prescaler is assigned to Timer0 will clear the prescaler count, but will not change the prescaler assignment.

## 10.2.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control, (i.e., it can be changed "on-the-fly" during program execution).

## 10.3 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF bit. The interrupt can be masked by clearing the TMR0IE bit. The TMR0IE bit must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from SLEEP, since the timer is shut-off during SLEEP.

## 10.4 16-Bit Mode Timer Reads and Writes

TMR0H is not the high byte of the timer/counter in 16-bit mode, but is actually a buffered version of the high byte of Timer0 (refer to Figure 10-2). The high byte of the Timer0 counter/timer is not directly readable nor writable. TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16-bits of Timer0 without having to verify that the read of the high and low byte were valid due to a rollover between successive reads of the high and low byte.

A write to the high byte of Timer0 must also take place through the TMR0H buffer register. Timer0 high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16-bits of Timer0 to be updated at once.

TABLE 10-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
TMR0L	Timer0 Module Low Byte Register								xxxx xxxx	uuuu uuuu
TMR0H	Timer0 Module High Byte Register								0000 0000	0000 0000
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	1111 1111
TRISA	—	PORTA Data Direction Register								-111 1111 -111 1111

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by Timer0.

# **PIC18FXX2**

---

---

## **NOTES:**

## 11.0 TIMER1 MODULE

The Timer1 module timer/counter has the following features:

- 16-bit timer/counter  
(two 8-bit registers; TMR1H and TMR1L)
  - Readable and writable (both registers)
  - Internal or external clock select
  - Interrupt-on-overflow from FFFFh to 0000h
  - RESET from CCP module special event trigger

Figure 11-1 is a simplified block diagram of the Timer1 module.

Register 11-1 details the Timer1 control register. This register controls the Operating mode of the Timer1 module, and contains the Timer1 oscillator enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit TMR1ON (T1CON<0>).

## REGISTER 11-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	
bit 7					bit 0			

- |         |  |
|---------|--|
| bit 7   | <b>RD16:</b> 16-bit Read/Write Mode Enable bit<br>1 = Enables register Read/Write of Timer1 in one 16-bit operation<br>0 = Enables register Read/Write of Timer1 in two 8-bit operations   |
| bit 6   | <b>Unimplemented:</b> Read as '0'  |
| bit 5-4 | <b>T1CKPS1:T1CKPS0:</b> Timer1 Input Clock Prescale Select bits<br>11 = 1:8 Prescale value<br>10 = 1:4 Prescale value<br>01 = 1:2 Prescale value<br>00 = 1:1 Prescale value  |
| bit 3   | <b>T1OSCEN:</b> Timer1 Oscillator Enable bit<br>1 = Timer1 Oscillator is enabled<br>0 = Timer1 Oscillator is shut-off<br>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<br><u>When TMR1CS = 1:</u><br>1 = Do not synchronize external clock input<br>0 = Synchronize external clock input<br><u>When TMR1CS = 0:</u><br>This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0. |
| bit 1   | <b>TMR1CS:</b> Timer1 Clock Source Select bit<br>1 = External clock from pin RC0/T1OSO/T13CKI (on the rising edge)<br>0 = Internal clock (Fosc/4)  |
| bit 0   | <b>TMR1ON:</b> Timer1 On bit<br>1 = Enables Timer1<br>0 = Stops Timer1   |

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

## 11.1 Timer1 Operation

Timer1 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

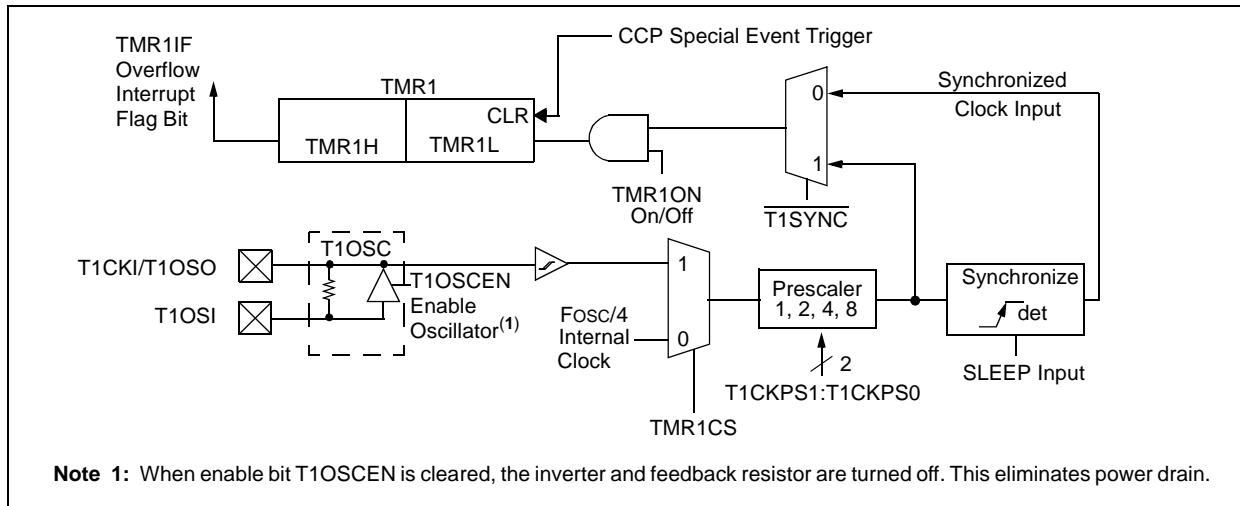
The Operating mode is determined by the clock select bit, TMR1CS (T1CON<1>).

When TMR1CS = 0, Timer1 increments every instruction cycle. When TMR1CS = 1, Timer1 increments on every rising edge of the external clock input or the Timer1 oscillator, if enabled.

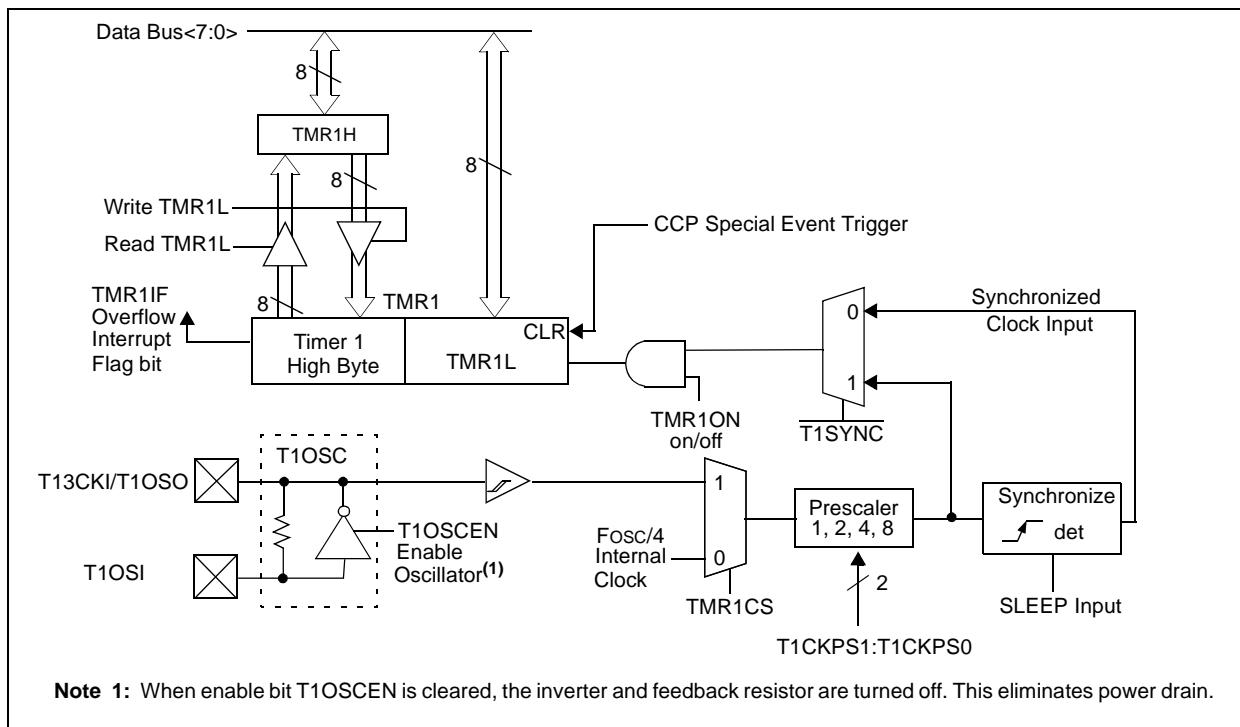
When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored, and the pins are read as '0'.

Timer1 also has an internal "RESET input". This RESET can be generated by the CCP module (Section 14.0).

**FIGURE 11-1: TIMER1 BLOCK DIAGRAM**



**FIGURE 11-2: TIMER1 BLOCK DIAGRAM: 16-BIT READ/WRITE MODE**



## 11.2 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit T1OSCEN (T1CON<3>). The oscillator is a low power oscillator rated up to 200 kHz. It will continue to run during SLEEP. It is primarily intended for a 32 kHz crystal. Table 11-1 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.

**TABLE 11-1: CAPACITOR SELECTION FOR THE ALTERNATE OSCILLATOR**

Osc Type	Freq	C1	C2
LP	32 kHz	TBD <sup>(1)</sup>	TBD <sup>(1)</sup>
<b>Crystal to be Tested:</b>			
32.768 kHz	Epson C-001R32.768K-A	± 20 PPM	

**Note 1:** Microchip suggests 33 pF 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.

## 11.3 Timer1 Interrupt

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

## 11.4 Resetting Timer1 using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a “special event trigger” (CCP1M3:CCP1M0 = 1011), this signal will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

**Note:** The special event triggers from the CCP1 module will not set interrupt flag bit TMR1IF (PIR1<0>).

Timer1 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer1 is running in Asynchronous Counter mode, this RESET operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1, the write will take precedence.

In this mode of operation, the CCPR1H:CCPR1L registers pair effectively becomes the period register for Timer1.

## 11.5 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 11-2). When the RD16 control bit (T1CON<7>) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 high byte buffer. This provides the user with the ability to accurately read all 16-bits of Timer1 without having to determine whether a read of the high byte followed by a read of the low byte is valid, due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H buffer register. Timer1 high byte is updated with the contents of TMR1H when a write occurs to TMR1L. This allows a user to write all 16 bits to both the high and low bytes of Timer1 at once.

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

# PIC18FXX2

---

**TABLE 11-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INTOIE	RBIE	TMROIF	INTOIF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

## 12.0 TIMER2 MODULE

The Timer2 module timer has the following features:

- 8-bit timer (TMR2 register)
- 8-bit period register (PR2)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMR2 match of PR2
- SSP module optional use of TMR2 output to generate clock shift

Timer2 has a control register shown in Register 12-1. Timer2 can be shut-off by clearing control bit TMR2ON (T2CON<2>) to minimize power consumption. Figure 12-1 is a simplified block diagram of the Timer2 module. Register 12-1 shows the Timer2 control register. The prescaler and postscaler selection of Timer2 are controlled by this register.

## 12.1 Timer2 Operation

Timer2 can be used as the PWM time-base for the PWM mode of the CCP module. The TMR2 register is readable and writable, and is cleared on any device RESET. The input clock (Fosc/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits T2CKPS1:T2CKPS0 (T2CON<1:0>). The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt (latched in flag bit TMR2IF, (PIR1<1>)).

The prescaler and postscaler counters are cleared when any of the following occurs:

- a write to the TMR2 register
- a write to the T2CON register
- any device RESET (Power-on Reset, MCLR Reset, Watchdog Timer Reset, or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

### REGISTER 12-1: 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
	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

bit 7

bit 0

- bit 7      **Unimplemented:** Read as '0'
- bit 6-3     **TOUTPS3:TOUTPS0:** 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

#### 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

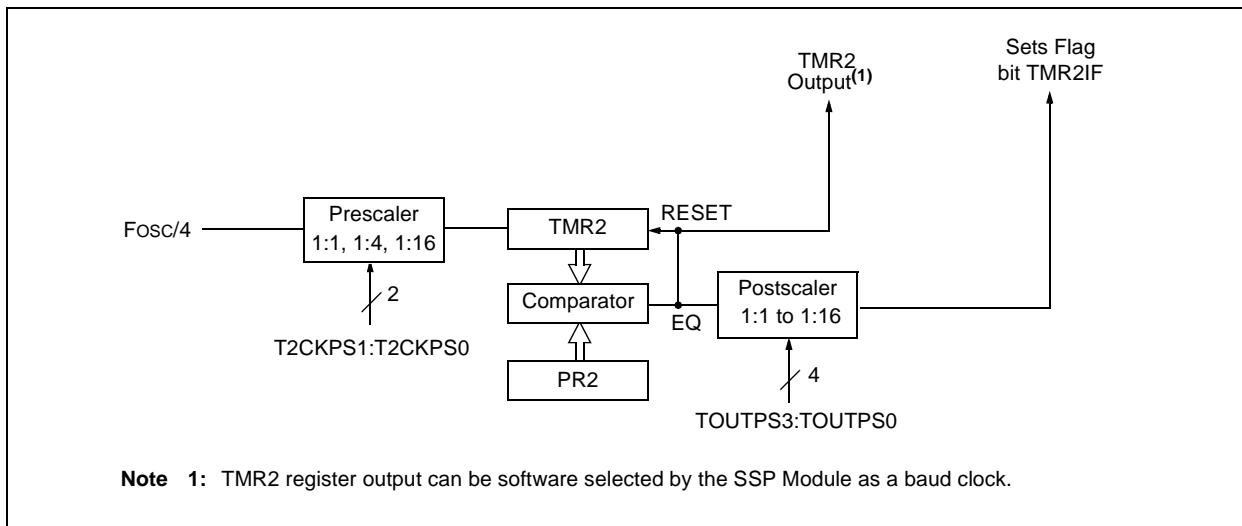
## 12.2 Timer2 Interrupt

The Timer2 module has an 8-bit period register, PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon RESET.

## 12.3 Output of TMR2

The output of TMR2 (before the postscaler) is fed to the Synchronous Serial Port module, which optionally uses it to generate the shift clock.

**FIGURE 12-1: TIMER2 BLOCK DIAGRAM**



**TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RBIE	TMROIF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TMR2	Timer2 Module Register								0000 0000	0000 0000
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by the Timer2 module.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

13.0 TIMER3 MODULE

The Timer3 module timer/counter has the following features:

- 16-bit timer/counter  
(two 8-bit registers; TMR3H and TMR3L)
  - Readable and writable (both registers)
  - Internal or external clock select
  - Interrupt-on-overflow from FFFFh to 0000h
  - RESET from CCP module trigger

Figure 13-1 is a simplified block diagram of the Timer3 module.

Register 13-1 shows the Timer3 control register. This register controls the Operating mode of the Timer3 module and sets the CCP clock source.

Register 11-1 shows the Timer1 control register. This register controls the Operating mode of the Timer1 module, as well as contains the Timer1 oscillator enable bit (T1OSCEN), which can be a clock source for Timer3.

## REGISTER 13-1: T3CON: TIMER3 CONTROL REGISTER

bit 7	<b>RD16:</b> 16-bit Read/Write Mode Enable bit 1 = Enables register Read/Write of Timer3 in one 16-bit operation 0 = Enables register Read/Write of Timer3 in two 8-bit operations
bit 6-3	<b>T3CCP2:T3CCP1:</b> Timer3 and Timer1 to CCPx Enable bits 1x = Timer3 is the clock source for compare/capture CCP modules 01 = Timer3 is the clock source for compare/capture of CCP2, Timer1 is the clock source for compare/capture of CCP1 00 = Timer1 is the clock source for compare/capture CCP modules
bit 5-4	<b>T3CKPS1:T3CKPS0:</b> Timer3 Input Clock Prescale Select bits 11 = 1:8 Prescale value 10 = 1:4 Prescale value 01 = 1:2 Prescale value 00 = 1:1 Prescale value
bit 2	<b>T3SYNC:</b> Timer3 External Clock Input Synchronization Control bit (Not usable if the system clock comes from Timer1/Timer3) <u>When TMR3CS = 1:</u> 1 = Do not synchronize external clock input 0 = Synchronize external clock input <u>When TMR3CS = 0:</u> This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0
bit 1	<b>TMR3CS:</b> Timer3 Clock Source Select bit 1 = External clock input from Timer1 oscillator or T1CKI (on the rising edge after the first falling edge) 0 = Internal clock (Fosc/4)
bit 0	<b>TMR3ON:</b> Timer3 On bit 1 = Enables Timer3 0 = Stops Timer3

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

## 13.1 Timer3 Operation

Timer3 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

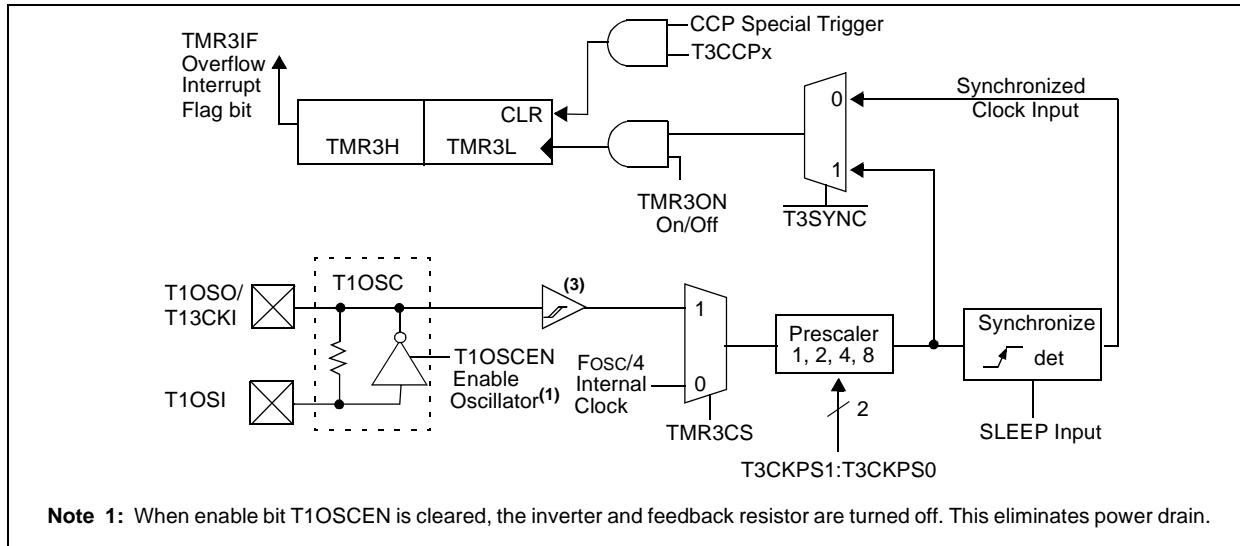
The Operating mode is determined by the clock select bit, TMR3CS (T3CON<1>).

When TMR3CS = 0, Timer3 increments every instruction cycle. When TMR3CS = 1, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

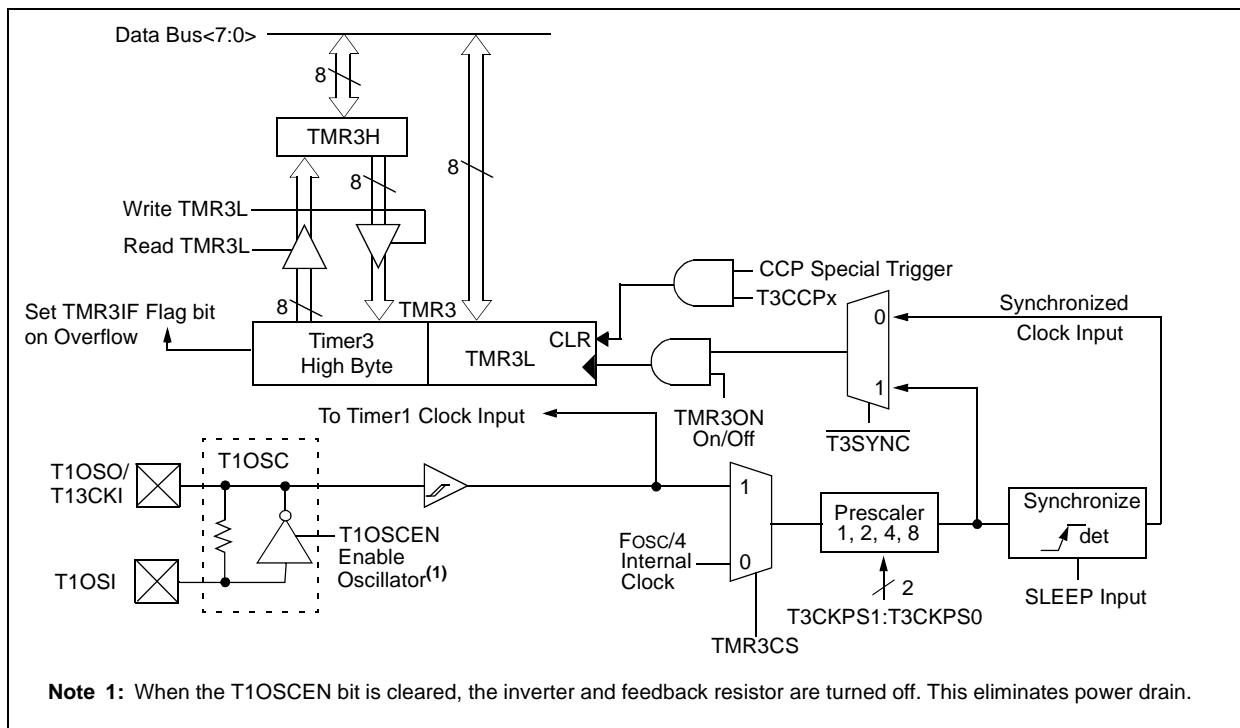
When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI and RC0/T1OSO/T1CKI pins become inputs. That is, the TRISC<1:0> value is ignored, and the pins are read as '0'.

Timer3 also has an internal "RESET input". This RESET can be generated by the CCP module (Section 14.0).

**FIGURE 13-1: TIMER3 BLOCK DIAGRAM**



**FIGURE 13-2: TIMER3 BLOCK DIAGRAM CONFIGURED IN 16-BIT READ/WRITE MODE**



### 13.2 Timer1 Oscillator

The Timer1 oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN (T1CON<3>) bit. The oscillator is a low power oscillator rated up to 200 KHz. See Section 11.0 for further details.

### 13.3 Timer3 Interrupt

The TMR3 Register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 Interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit, TMR3IF (PIR2<1>). This interrupt can be enabled/disabled by setting/clearing TMR3 interrupt enable bit, TMR3IE (PIE2<1>).

### 13.4 Resetting Timer3 Using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a “special event trigger” (CCP1M3:CCP1M0 = 1011), this signal will reset Timer3.

**Note:** The special event triggers from the CCP module will not set interrupt flag bit, TMR3IF (PIR1<0>).

Timer3 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer3 is running in Asynchronous Counter mode, this RESET operation may not work. In the event that a write to Timer3 coincides with a special event trigger from CCP1, the write will take precedence. In this mode of operation, the CCP1H:CCP1L registers pair effectively becomes the period register for Timer3.

TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 1111
TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	<u>T1SYNC</u>	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	<u>T3SYNC</u>	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

# **PIC18FXX2**

---

---

## **NOTES:**

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

Each CCP (Capture/Compare/PWM) module contains a 16-bit register which can operate as a 16-bit Capture register, as a 16-bit Compare register or as a PWM Master/Slave Duty Cycle register. Table 14-1 shows the timer resources of the CCP Module modes.

The operation of CCP1 is identical to that of CCP2, with the exception of the special event trigger. Therefore, operation of a CCP module in the following sections is described with respect to CCP1.

Table 14-2 shows the interaction of the CCP modules.

### REGISTER 14-1: CCP1CON REGISTER/CCP2CON REGISTER

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

bit 7

bit 0

- bit 7-6    **Unimplemented:** Read as '0'
- bit 5-4    **DCxB1:DCxB0:** PWM Duty Cycle bit1 and bit0  
Capture mode:  
 Unused  
Compare mode:  
 Unused  
PWM mode:  
 These bits are the two LSbs (bit1 and bit0) of the 10-bit PWM duty cycle. The upper eight bits (DCx9:DCx2) of the duty cycle are found in CCPRxL.
- bit 3-0    **CCPxM3:CCPxM0:** CCPx 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 CCP pin Low, on compare match force CCP pin High (CCPIF bit is set)  
 1001 = Compare mode,  
     Initialize CCP pin High, on compare match force CCP pin Low (CCPIF bit is set)  
 1010 = Compare mode,  
     Generate software interrupt on compare match (CCPIF bit is set, CCP pin is unaffected)  
 1011 = Compare mode,  
     Trigger special event (CCPIF bit is set)  
 11xx = PWM mode

#### 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

## 14.1 CCP1 Module

Capture/Compare/PWM Register 1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. All are readable and writable.

**TABLE 14-1: CCP MODE - TIMER RESOURCE**

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

## 14.2 CCP2 Module

Capture/Compare/PWM Register2 (CCPR2) is comprised of two 8-bit registers: CCPR2L (low byte) and CCPR2H (high byte). The CCP2CON register controls the operation of CCP2. All are readable and writable.

**TABLE 14-2: INTERACTION OF TWO CCP MODULES**

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	TMR1 or TMR3 time-base. Time-base can be different for each CCP.
Capture	Compare	The compare could be configured for the special event trigger, which clears either TMR1 or TMR3 depending upon which time-base is used.
Compare	Compare	The compare(s) could be configured for the special event trigger, which clears TMR1 or TMR3 depending upon which time-base is used.
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt).
PWM	Capture	None
PWM	Compare	None

### 14.3 Capture Mode

In Capture mode, CCP1H:CCPR1L captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on pin RC2/CCP1. An event is defined as one of the following:

- every falling edge
- every rising edge
- every 4th rising edge
- every 16th rising edge

The event is selected by control bits CCP1M3:CCP1M0 (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit CCP1IF (PIR1<2>) is set; it must be cleared in software. If another capture occurs before the value in register CCPR1 is read, the old captured value is overwritten by the new captured value.

#### 14.3.1 CCP PIN CONFIGURATION

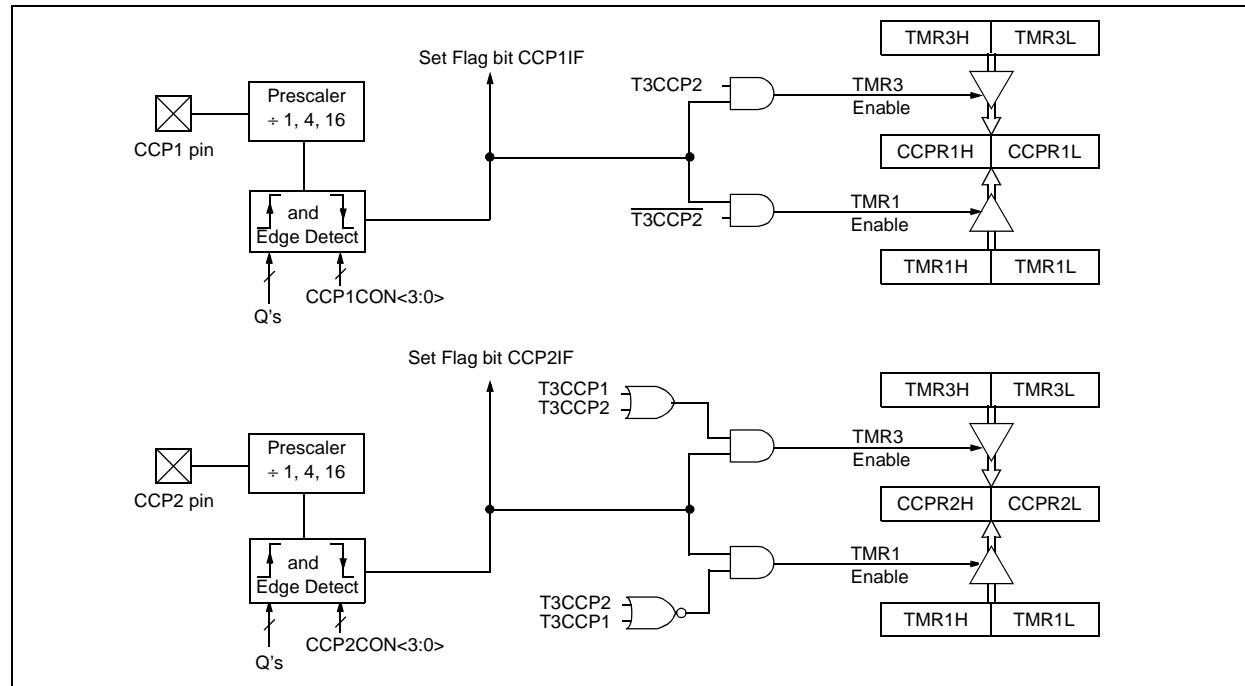
In Capture mode, the RC2/CCP1 pin should be configured as an input by setting the TRISC<2> bit.

**Note:** If the RC2/CCP1 is configured as an output, a write to the port can cause a capture condition.

#### 14.3.2 TIMER1/TIMER3 MODE SELECTION

The timers that are to be used with the capture feature (either Timer1 and/or Timer3) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work. The timer to be used with each CCP module is selected in the T3CON register.

**FIGURE 14-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**



### 14.3.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit CCP1IE (PIE1<2>) clear to avoid false interrupts and should clear the flag bit, CCP1IF, following any such change in Operating mode.

### 14.3.4 CCP PRESCALER

There are four prescaler settings, specified by bits CCP1M3:CCP1M0. Whenever the CCP module is turned off or the CCP module is not in Capture mode, the prescaler counter is cleared. This means that any RESET will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared, therefore, the first capture may be from a non-zero prescaler. Example 14-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the "false" interrupt.

#### EXAMPLE 14-1: CHANGING BETWEEN CAPTURE PRESCALERS

```

CLRF    CCP1CON, F ; Turn CCP module off
MOVLW  NEW_CAPT_PS ; Load WREG with the
                    ; new prescaler mode
                    ; value and CCP ON
MOVWF  CCP1CON      ; Load CCP1CON with
                    ; this value

```

## 14.4 Compare Mode

In Compare mode, the 16-bit CCPR1 (CCPR2) register value is constantly compared against either the TMR1 register pair value, or the TMR3 register pair value. When a match occurs, the RC2/CCP1 (RC1/CCP2) pin is:

- driven High
- driven Low
- toggle output (High to Low or Low to High)
- remains unchanged

The action on the pin is based on the value of control bits CCP1M3:CCP1M0 (CCP2M3:CCP2M0). At the same time, interrupt flag bit CCP1IF (CCP2IF) is set.

### 14.4.1 CCP PIN CONFIGURATION

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

**Note:** Clearing the CCP1CON register will force the RC2/CCP1 compare output latch to the default low level. This is not the PORTC I/O data latch.

### 14.4.2 TIMER1/TIMER3 MODE SELECTION

Timer1 and/or Timer3 must be running in Timer mode or Synchronized Counter mode if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

### 14.4.3 SOFTWARE INTERRUPT MODE

When generate software interrupt is chosen, the CCP1 pin is not affected. Only a CCP interrupt is generated (if enabled).

### 14.4.4 SPECIAL EVENT TRIGGER

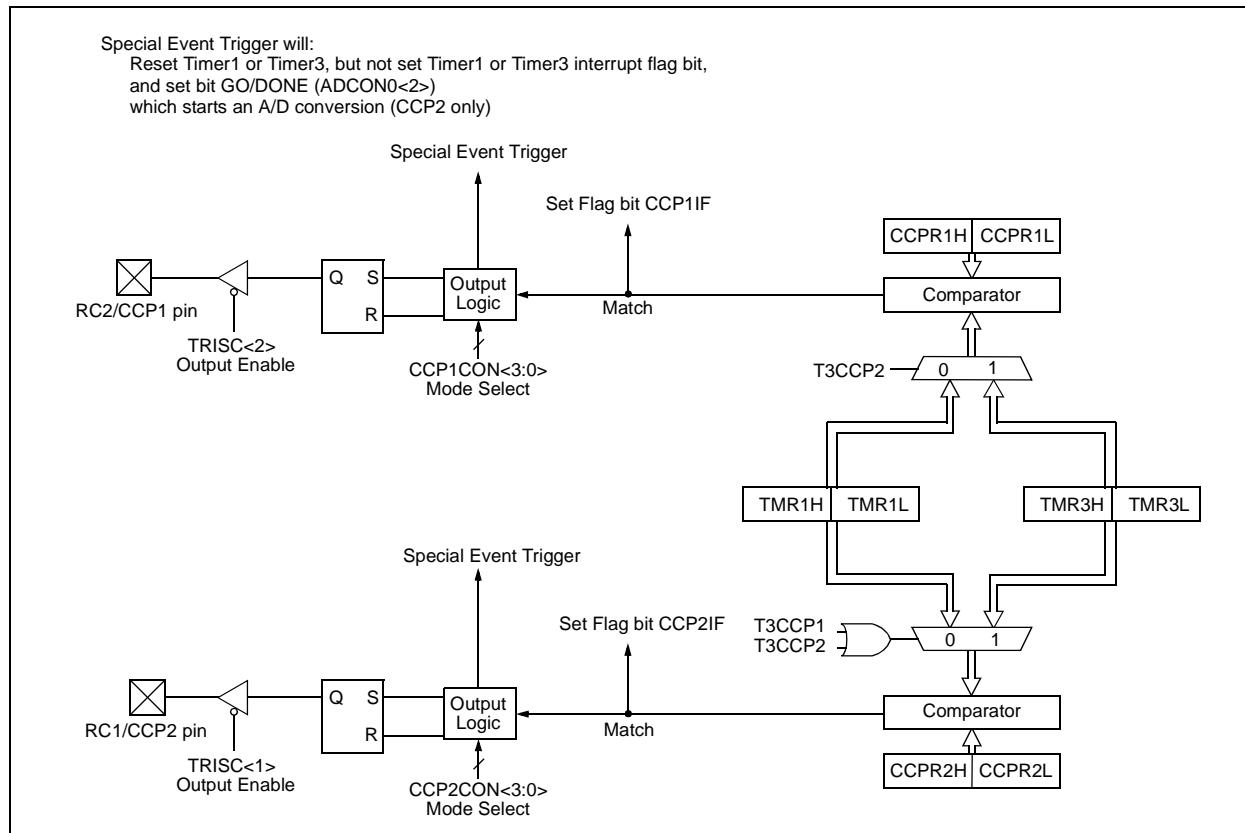
In this mode, an internal hardware trigger is generated, which may be used to initiate an action.

The special event trigger output of CCP1 resets the TMR1 register pair. This allows the CCPR1 register to effectively be a 16-bit programmable period register for Timer1.

The special trigger output of CCPx resets either the TMR1 or TMR3 register pair. Additionally, the CCP2 Special Event Trigger will start an A/D conversion if the A/D module is enabled.

**Note:** The special event trigger from the CCP2 module will not set the Timer1 or Timer3 interrupt flag bits.

**FIGURE 14-2: COMPARE MODE OPERATION BLOCK DIAGRAM**



**TABLE 14-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu
CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000
PIR2	—	—	—	EEIE	BCLIF	LVDIF	TMR3IF	CCP2IF	--0 0000	--0 0000
PIE2	—	—	—	EEIF	BCLIE	LVDIE	TMR3IE	CCP2IE	--0 0000	--0 0000
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	--1 1111	--1 1111
TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by Capture and Timer1.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2x2 devices; always maintain these bits clear.

## 14.5 PWM Mode

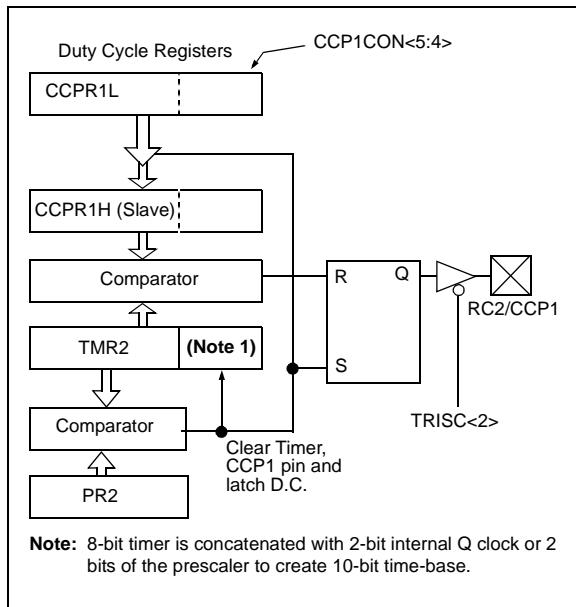
In Pulse Width Modulation (PWM) mode, the CCP1 pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

**Note:** Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 14-3 shows a simplified block diagram of the CCP module in PWM mode.

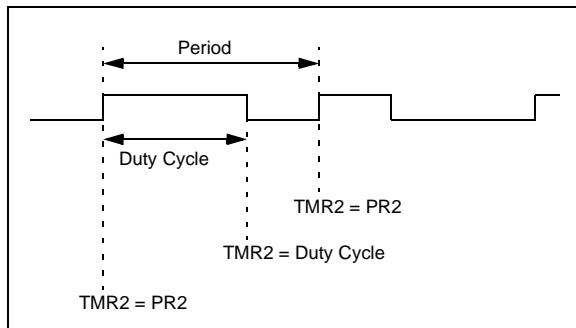
For a step-by-step procedure on how to set up the CCP module for PWM operation, see Section 14.5.3.

**FIGURE 14-3: SIMPLIFIED PWM BLOCK DIAGRAM**



A PWM output (Figure 14-4) 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 14-4: PWM OUTPUT**



### 14.5.1 PWM PERIOD

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

$$\text{PWM period} = (\text{PR2} + 1) \cdot 4 \cdot \text{Tosc} \cdot (\text{TMR2 prescale value})$$

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 (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

**Note:** The Timer2 postscaler (see Section 12.0) 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.

### 14.5.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM duty cycle} = (\text{CCPR1L:CCP1CON<5:4>} \cdot \text{Tosc} \cdot (\text{TMR2 prescale value}))$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read only register.

The CCPR1H register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

When the CCPR1H and 2-bit latch match TMR2 concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

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

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

**Note:** If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

## 14.5.3 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR1L register and CCP1CON<5:4> bits.

3. Make the CCP1 pin an output by clearing the TRISC<2> bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCP1 module for PWM operation.

TABLE 14-4: 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	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	14	12	10	8	7	6.58

TABLE 14-5: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TMR2	Timer2 Module Register								0000 0000	0000 0000
PR2	Timer2 Module Period Register								1111 1111	1111 1111
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu
CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PWM and Timer2.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

# **PIC18FXX2**

---

---

## **NOTES:**

## 15.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

### 15.1 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

### 15.2 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.

### 15.3 SPI Mode

The SPI mode allows 8-bits of data to be synchronously transmitted and received, simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

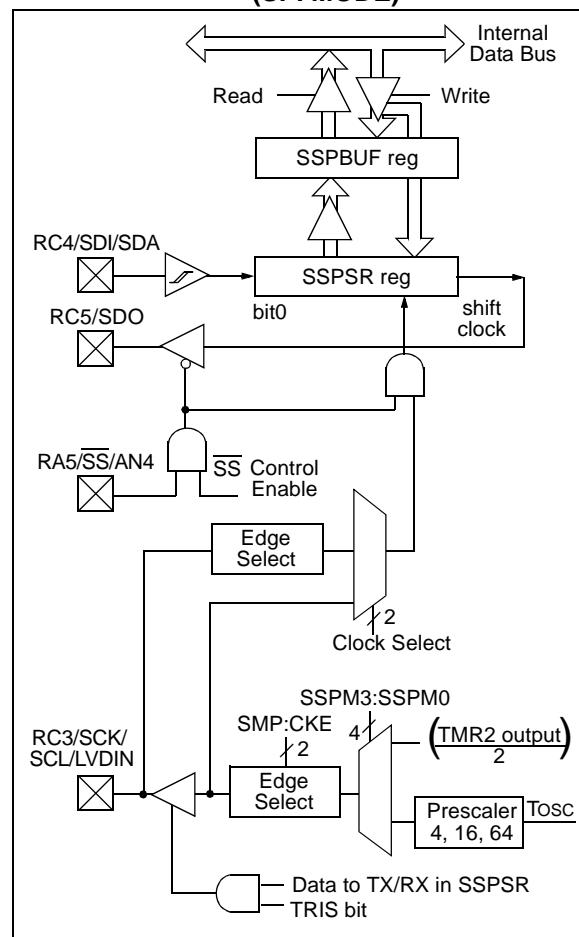
- Serial Data Out (SDO) - RC5/SDO
- Serial Data In (SDI) - RC4/SDI/SDA
- Serial Clock (SCK) - RC3/SCK/SCL/LVDIN

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

- Slave Select ( $\overline{SS}$ ) - RA5/ $\overline{SS}$ /AN4

Figure 15-1 shows the block diagram of the MSSP module when operating in SPI mode.

**FIGURE 15-1: MSSP BLOCK DIAGRAM (SPI MODE)**



# PIC18FXX2

## 15.3.1 REGISTERS

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register1 (SSPCON1)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (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 15-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/A	P	S	R/W	UA	BF

bit 7

bit 0

bit 7 **SMP:** Sample bit

SPI Master mode:

1 = Input data sampled at end of data output time  
0 = Input data sampled at middle of data output time

SPI Slave mode:

SMP must be cleared when SPI is used in Slave mode

bit 6 **CKE:** SPI Clock Edge Select

When CKP = 0:

1 = Data transmitted on rising edge of SCK  
0 = Data transmitted on falling edge of SCK

When CKP = 1:

1 = Data transmitted on falling edge of SCK  
0 = Data transmitted on rising edge of SCK

bit 5 **D/A:** Data/Address bit

Used in I<sup>2</sup>C mode only

bit 4 **P:** STOP bit

Used in I<sup>2</sup>C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** START bit

Used in I<sup>2</sup>C mode only

bit 2 **R/W:** Read/Write bit information

Used in I<sup>2</sup>C mode only

bit 1 **UA:** Update Address

Used in I<sup>2</sup>C mode only

bit 0 **BF:** Buffer Full Status bit (Receive mode only)

1 = Receive complete, SSPBUF is full  
0 = Receive not complete, SSPBUF is empty

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

## REGISTER 15-2: SSPCON1: MSSP CONTROL REGISTER1 (SPI MODE)

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| WCOL  | SSPOV | SSPEN | CKP   | SSPM3 | SSPM2 | SSPM1 | SSPM0 |
| bit 7 |       |       |       | bit 0 |       |       |       |

- bit 7     **WCOL:** Write Collision Detect bit (Transmit mode only)  
     1 = The SSPBUF register is written while it is still transmitting the previous word  
         (must be cleared in software)  
     0 = No collision
- bit 6     **SSPOV:** Receive Overflow Indicator bit  
SPI Slave mode:  
     1 = A new byte is received while the SSPBUF register is still holding the previous data. In case  
         of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user  
         must read the SSPBUF, even if only transmitting data, to avoid setting overflow  
         (must be cleared in software).  
     0 = No overflow  
**Note:** In Master mode, the overflow bit is not set since each new reception (and  
         transmission) is initiated by writing to the SSPBUF register.
- bit 5     **SSPEN:** Synchronous Serial Port Enable bit  
     1 = Enables serial port and configures SCK, SDO, SDI, and  $\overline{SS}$  as serial port pins  
     0 = Disables serial port and configures these pins as I/O port pins  
**Note:** When enabled, these pins must be properly configured as input or output.
- bit 4     **CKP:** Clock Polarity Select bit  
     1 = IDLE state for clock is a high level  
     0 = IDLE state for clock is a low level
- bit 3-0    **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits  
     0101 = SPI Slave mode, clock = SCK pin,  $\overline{SS}$  pin control disabled,  $\overline{SS}$  can be used as I/O pin  
     0100 = SPI Slave mode, clock = SCK pin, SS pin control enabled  
     0011 = SPI Master mode, clock = TMR2 output/2  
     0010 = SPI Master mode, clock = Fosc/64  
     0001 = SPI Master mode, clock = Fosc/16  
     0000 = SPI Master mode, clock = Fosc/4  
**Note:** Bit combinations not specifically listed here are either reserved, or implemented in  
         I<sup>2</sup>C mode only.

## 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

## 15.3.2 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. 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 15-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

The SSPSR is not directly readable or writable, and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP status register (SSPSTAT) indicates the various status conditions.

## EXAMPLE 15-1: LOADING THE SSPBUF (SSPSR) REGISTER

```
LOOP BTFS 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
```

### 15.3.3 ENABLING SPI I/O

To enable the serial port, SSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize 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. That is:

- 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 TRISC<4> bit set

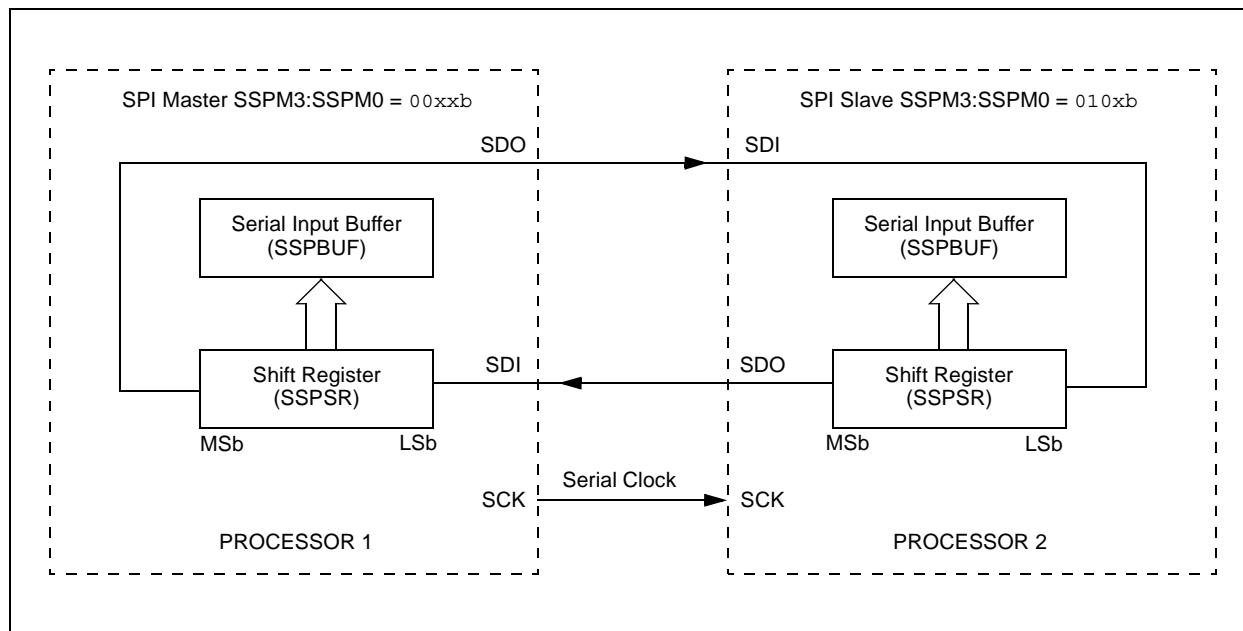
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

### 15.3.4 TYPICAL CONNECTION

Figure 15-2 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 15-2: SPI MASTER/SLAVE CONNECTION**



### 15.3.5 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 15-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode.

The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then, would give waveforms for SPI communication as shown in

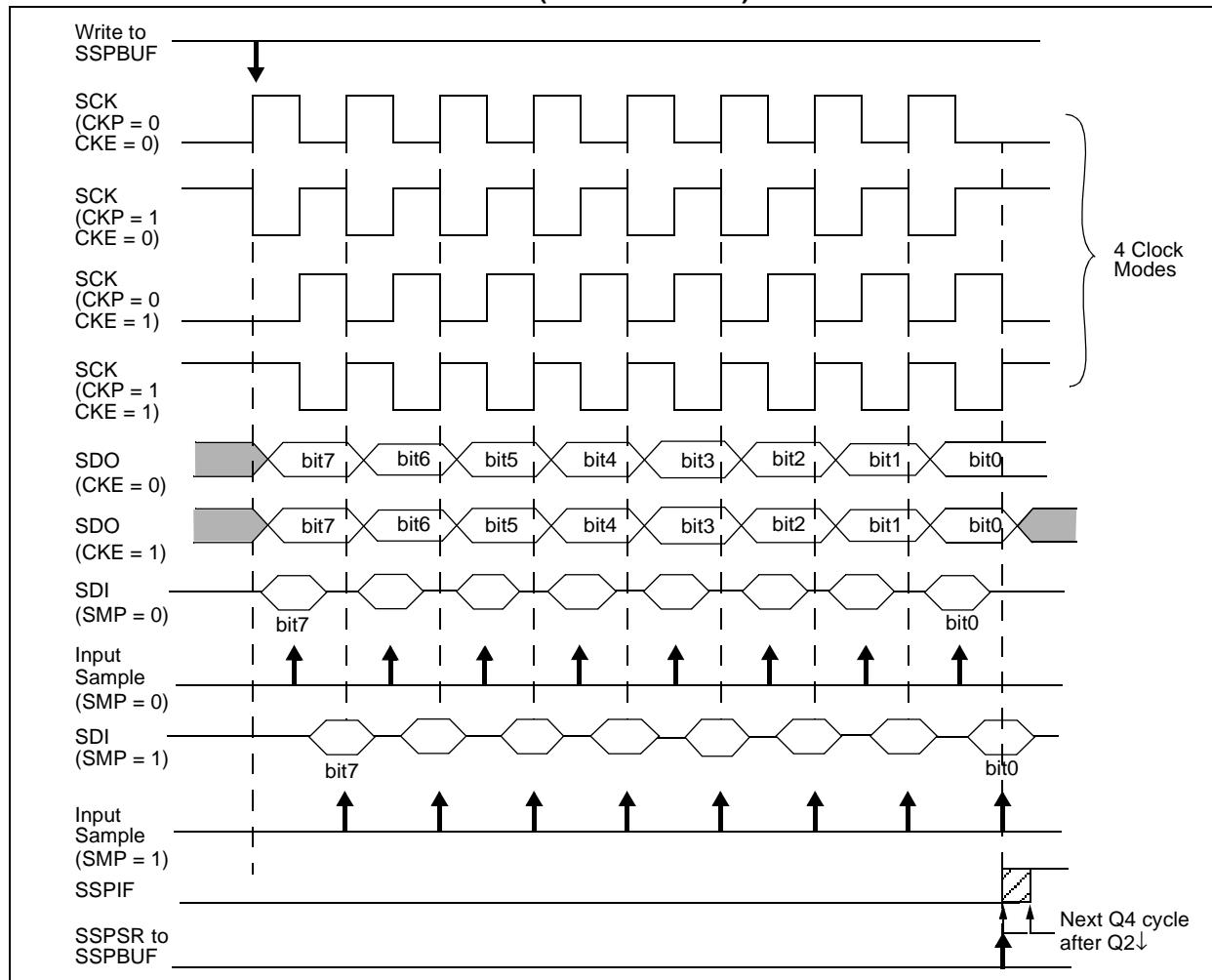
Figure 15-3, Figure 15-5, and Figure 15-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- Fosc/4 (or Tcy)
- Fosc/16 (or 4 • Tcy)
- Fosc/64 (or 16 • Tcy)
- Timer2 output/2

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 15-3 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 15-3: SPI MODE WAVEFORM (MASTER MODE)**



### 15.3.6 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag bit is set.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in SLEEP mode, the slave can transmit/receive data. When a byte is received, the device will wake-up from sleep.

### 15.3.7 SLAVE SELECT SYNCHRONIZATION

The SS pin allows a Synchronous Slave mode. The SPI must be in Slave mode with SS pin control enabled (SSPCON1<3:0> = 04h). The pin must not be driven low for the SS pin to function as an input. The Data Latch must be high. When the SS pin is low, transmission and reception are enabled and 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.

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

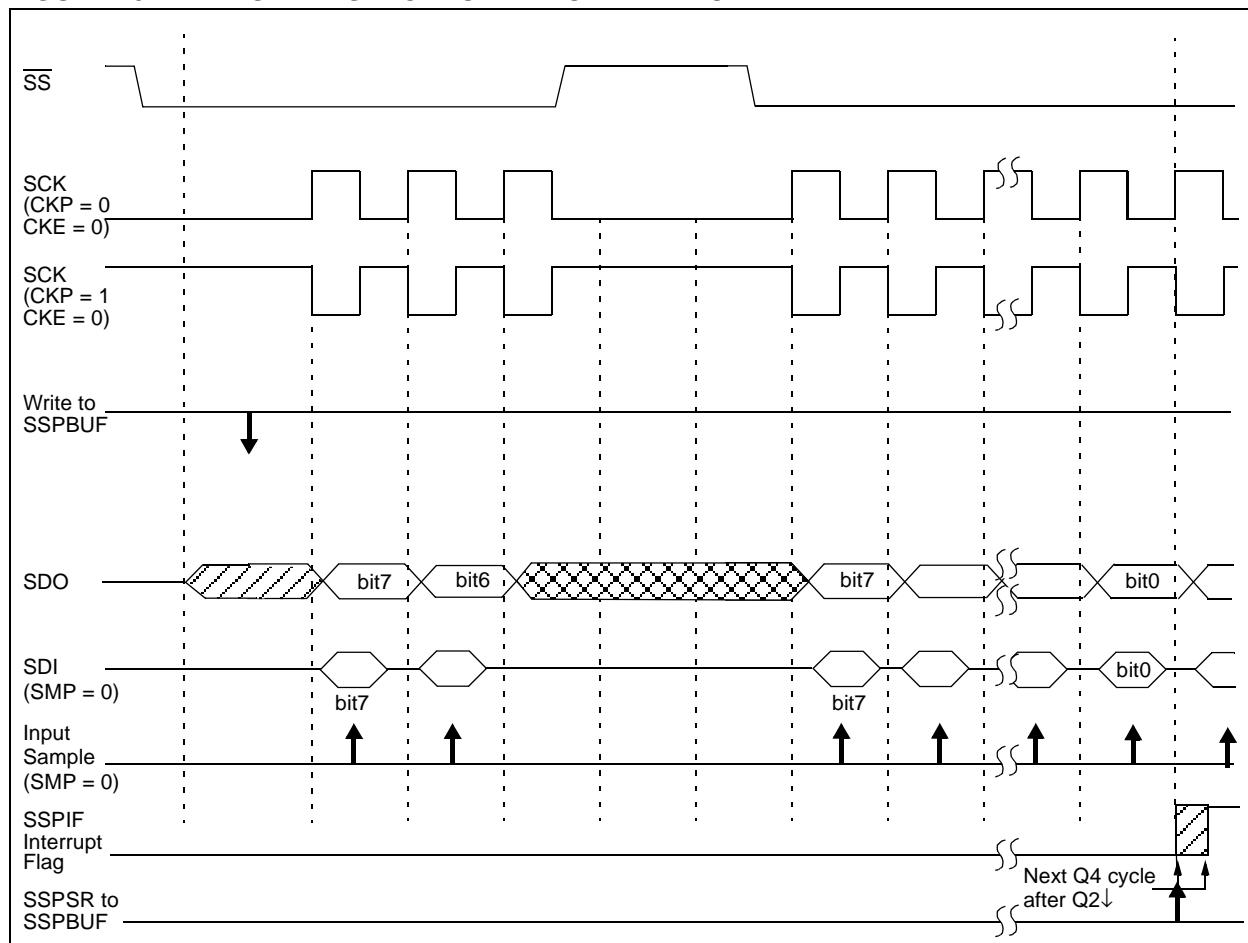
**Note 1:** When the SPI is in Slave mode with SS pin control enabled (SSPCON<3:0> = 0100), the SPI module will reset if the SS pin is set to VDD.

**2:** If the SPI 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.

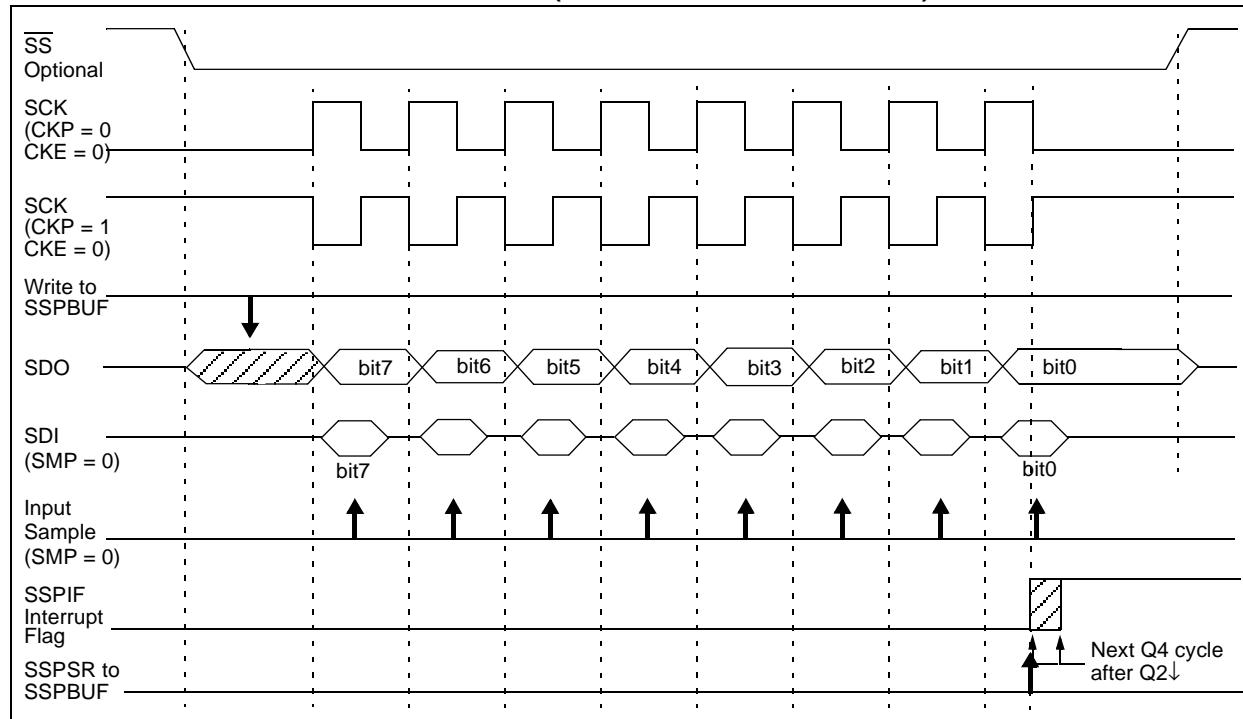
To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function), since it cannot create a bus conflict.

**FIGURE 15-4: SLAVE SYNCHRONIZATION WAVEFORM**

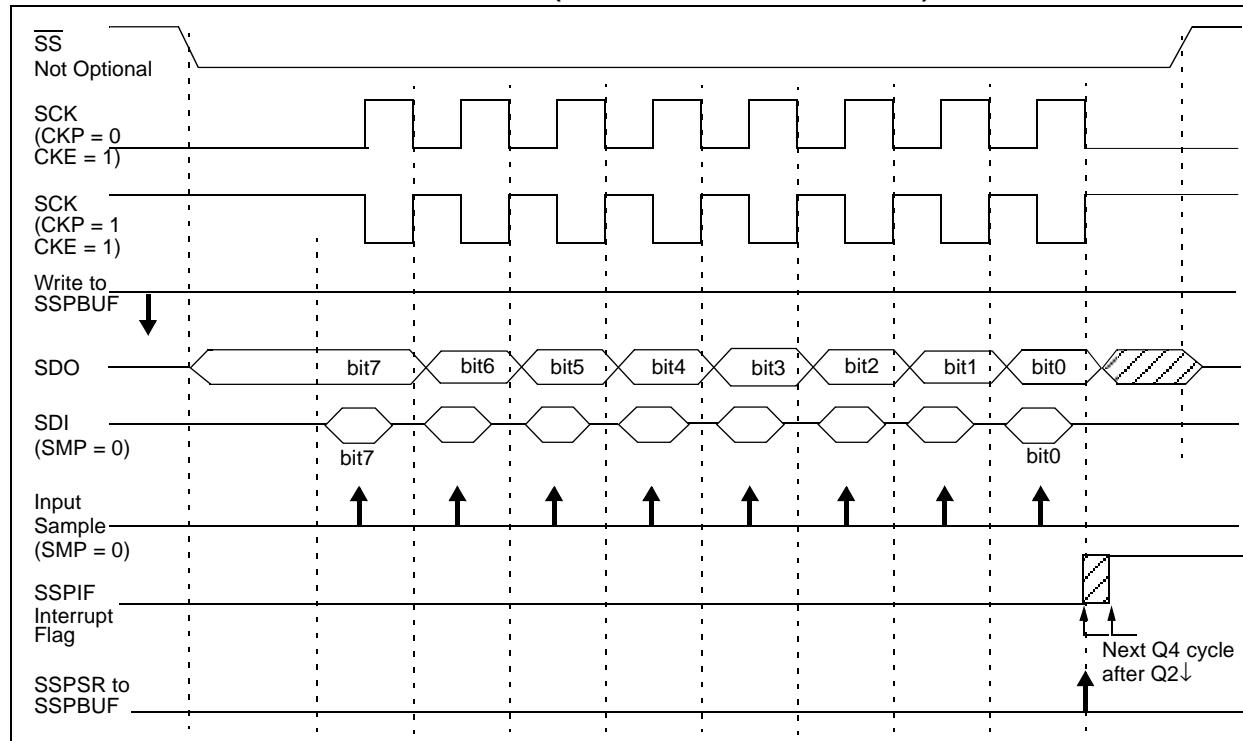


# PIC18FXX2

**FIGURE 15-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**



**FIGURE 15-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**



### 15.3.8 SLEEP OPERATION

In Master mode, all module clocks are halted and the transmission/reception will remain in that state until the device wakes from SLEEP. After the device returns to Normal mode, the module will continue to transmit/receive data.

In Slave mode, the SPI transmit/receive shift register operates asynchronously to the device. This allows the device to be placed in SLEEP 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 from SLEEP.

### 15.3.9 EFFECTS OF A RESET

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

### 15.3.10 BUS MODE COMPATIBILITY

Table 15-1 shows the compatibility between the standard SPI modes and the states the CKP and CKE control bits.

**TABLE 15-1: SPI BUS MODES**

Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

There is also a SMP bit which controls when the data is sampled.

**TABLE 15-2: REGISTERS ASSOCIATED WITH SPI OPERATION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
TRISA	—	PORTA Data Direction Register								-111 1111 -111 1111
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the MSSP in SPI mode.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18C2X2 devices; always maintain these bits clear.

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

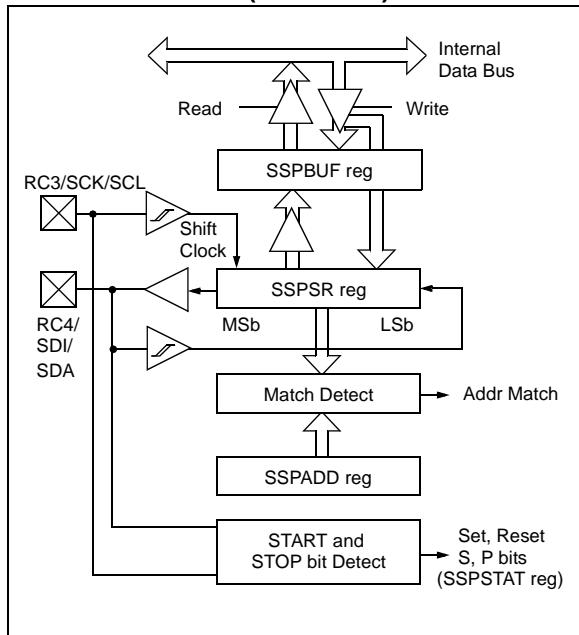
The MSSP module in I<sup>2</sup>C mode fully implements all master and slave functions (including general call support) and provides interrupts on START and STOP bits in hardware to determine a free bus (multi-master function). The MSSP module implements the Standard mode specifications, as well as 7-bit 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 15-7: MSSP BLOCK DIAGRAM (I<sup>2</sup>C MODE)**



### 15.4.1 REGISTERS

The MSSP module has six registers for I<sup>2</sup>C operation. These are:

- MSSP Control Register1 (SSPCON1)
- MSSP Control Register2 (SSPCON2)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- MSSP Shift Register (SSPSR) - Not directly accessible
- MSSP Address Register (SSPADD)

SSPCON, SSPCON2 and SSPSTAT are the control and status registers in I<sup>2</sup>C mode operation. The SSPCON 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 SSP is configured in I<sup>2</sup>C Slave mode. When the SSP 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.

## REGISTER 15-3: SSPSTAT: MSSP STATUS REGISTER (I<sup>2</sup>C MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/A	P	S	R/W	UA	BF
bit 7	bit 0						

- bit 7      **SMP:** Slew Rate Control bit  
In Master or Slave mode:  
 1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)  
 0 = Slew rate control enabled for High Speed mode (400 kHz)
- bit 6      **CKE:** SMBus Select bit  
In Master or Slave mode:  
 1 = Enable SMBus specific inputs  
 0 = Disable SMBus specific inputs
- bit 5      **D/A:** Data/Address bit  
In Master mode:  
 Reserved  
In Slave mode:  
 1 = Indicates that the last byte received or transmitted was data  
 0 = Indicates that the last byte received or transmitted was address
- bit 4      **P:** STOP bit  
 1 = Indicates that a STOP bit has been detected last  
 0 = STOP bit was not detected last  
**Note:** This bit is cleared on RESET and when SSPEN is cleared.
- bit 3      **S:** START bit  
 1 = Indicates that a start bit has been detected last  
 0 = START bit was not detected last  
**Note:** This bit is cleared on RESET and when SSPEN is cleared.
- bit 2      **R/W:** Read/Write bit Information (I<sup>2</sup>C mode only)  
In Slave mode:  
 1 = Read  
 0 = Write  
**Note:** 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.  
In Master mode:  
 1 = Transmit is in progress  
 0 = Transmit is not in progress  
**Note:** ORing this bit with SEN, RSEN, PEN, RCEN, or ACKEN will indicate if the MSSP is in IDLE mode.
- bit 1      **UA:** Update Address (10-bit Slave mode only)  
 1 = Indicates that the user needs to update the address in the SSPADD register  
 0 = Address does not need to be updated
- bit 0      **BF:** Buffer Full Status bit  
In Transmit mode:  
 1 = Receive complete, SSPBUF is full  
 0 = Receive not complete, SSPBUF is empty  
In Receive mode:  
 1 = Data transmit in progress (does not include the ACK and STOP bits), SSPBUF is full  
 0 = Data transmit complete (does not include the ACK and STOP bits), SSPBUF is empty

### 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

# PIC18FXX2

## REGISTER 15-4: SSPCON1: MSSP CONTROL REGISTER1 (I<sup>2</sup>C MODE)

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| WCOL  | SSPOV | SSPEN | CKP   | SSPM3 | SSPM2 | SSPM1 | SSPM0 |

bit 7

bit 0

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:** Synchronous Serial Port Enable bit

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

**Note:** When enabled, the SDA and SCL pins must be properly configured as input or output.

bit 4 **CKP:** SCK Release Control bit

In Slave mode:

1 = Release 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:** Synchronous Serial Port Mode Select bits

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

**Note:** Bit combinations not specifically listed here are either reserved, or implemented in SPI mode only.

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

REGISTER 15-5: SSPCON2: MSSP CONTROL REGISTER 2 (I<sup>2</sup>C MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

- bit 7      **GCEN:** General Call Enable bit (Slave mode only)  
               1 = Enable interrupt when a general call address (0000h) is received in the SSPSR  
               0 = General call address disabled
- bit 6      **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)  
               1 = Acknowledge was not received from slave  
               0 = Acknowledge was received from slave
- bit 5      **ACKDT:** Acknowledge Data bit (Master Receive mode only)  
               1 = Not Acknowledge  
               0 = Acknowledge  
               **Note:** Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
- bit 4      **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)  
               1 = Initiate Acknowledge sequence on SDA and SCL pins, and transmit ACKDT data bit.  
                     Automatically cleared by hardware.  
               0 = Acknowledge sequence IDLE
- bit 3      **RCEN:** Receive Enable bit (Master mode only)  
               1 = Enables Receive mode for I<sup>2</sup>C  
               0 = Receive IDLE
- bit 2      **PEN:** STOP Condition Enable bit (Master mode only)  
               1 = Initiate STOP condition on SDA and SCL pins. Automatically cleared by hardware.  
               0 = STOP condition IDLE
- bit 1      **RSEN:** Repeated START Condition Enabled bit (Master mode only)  
               1 = Initiate Repeated START condition on SDA and SCL pins.  
                     Automatically cleared by hardware.  
               0 = Repeated START condition IDLE
- bit 0      **SEN:** START Condition Enabled/Stretch Enabled bit  
In Master mode:  
               1 = Initiate START condition on SDA and SCL pins. Automatically cleared by hardware.  
               0 = START condition IDLE  
In Slave mode:  
               1 = Clock stretching is enabled for both Slave Transmit and Slave Receive (stretch enabled)  
               0 = Clock stretching is enabled for slave transmit only (Legacy mode)
- Note:** For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I<sup>2</sup>C module is not in the IDLE mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

## 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

## 15.4.2 OPERATION

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

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

- I<sup>2</sup>C Master mode, clock = OSC/4 (SSPADD +1)
- I<sup>2</sup>C Slave mode (7-bit address)
- I<sup>2</sup>C Slave mode (10-bit address)
- I<sup>2</sup>C Slave mode (7-bit address), with START and STOP bit interrupts enabled
- I<sup>2</sup>C Slave mode (10-bit address), with START and STOP bit interrupts enabled
- I<sup>2</sup>C Firmware controlled master operation, 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 guarantee proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

## 15.4.3 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 (SSPCON<6>) was set before the transfer was received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF (PIR1<3>) is set. The BF bit is cleared by reading the SSPBUF register, while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the I<sup>2</sup>C specification, as well as the requirement of the MSSP module, are shown in timing parameter 100 and parameter 101.

## 15.4.3.1 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 Address 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 address is as follows, with steps 7 through 9 for the slave-transmitter:

1. Receive first (high) byte of Address (bits SSPIF, BF and bit 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.

### 15.4.3.2 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 (SSPCON1<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 15.4.4 (“Clock Stretching”), for more detail.

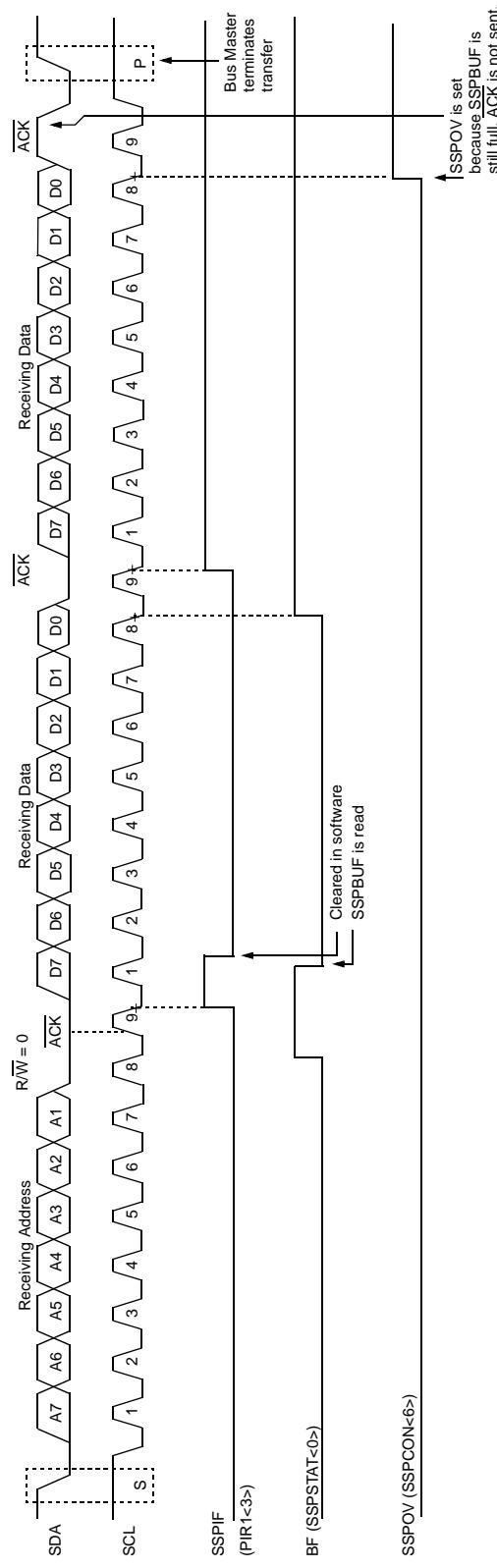
### 15.4.3.3 Transmission

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 pin RC3/SCK/SCL is held low, regardless of SEN (see “Clock Stretching”, Section 15.4.4, 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 pin RC3/SCK/SCL 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 15-9).

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 (resets SSPSTAT register) 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, pin RC3/SCK/SCL must be enabled by setting bit CKP.

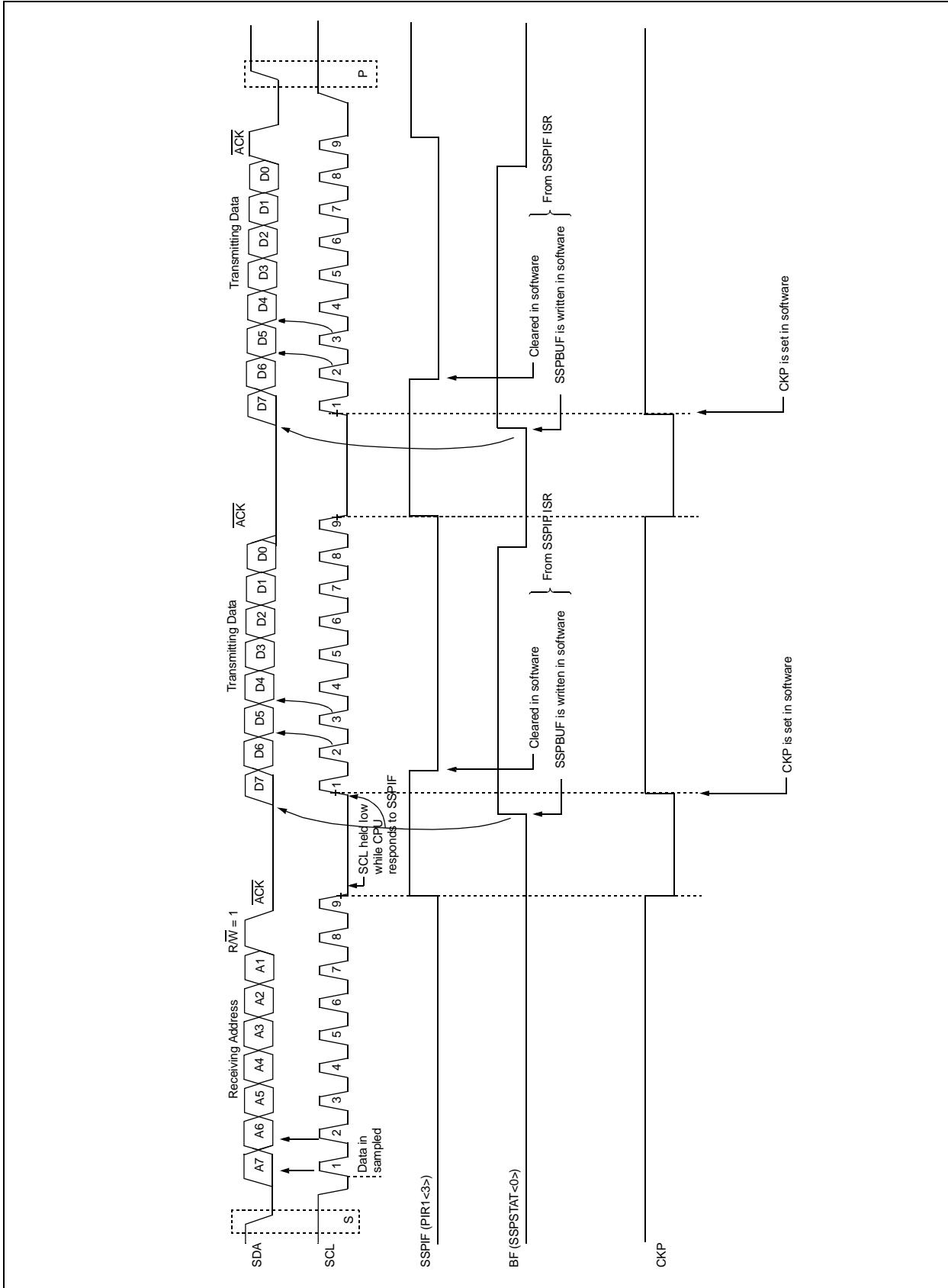
An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

**FIGURE 15-8: I<sup>2</sup>C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)**



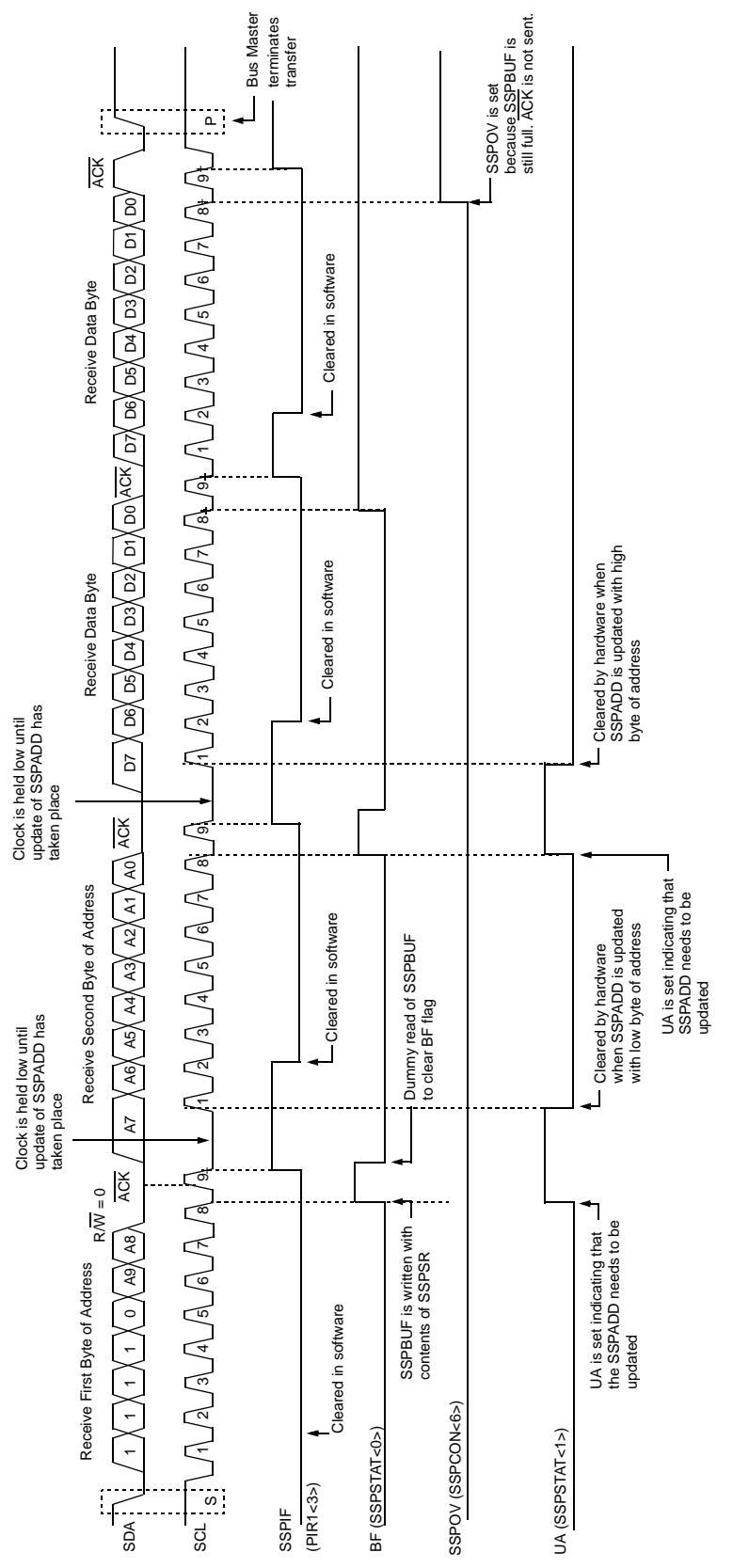
CKP (CKP does not reset to '0' when SEN = 0)

**FIGURE 15-9: I<sup>2</sup>C SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)**

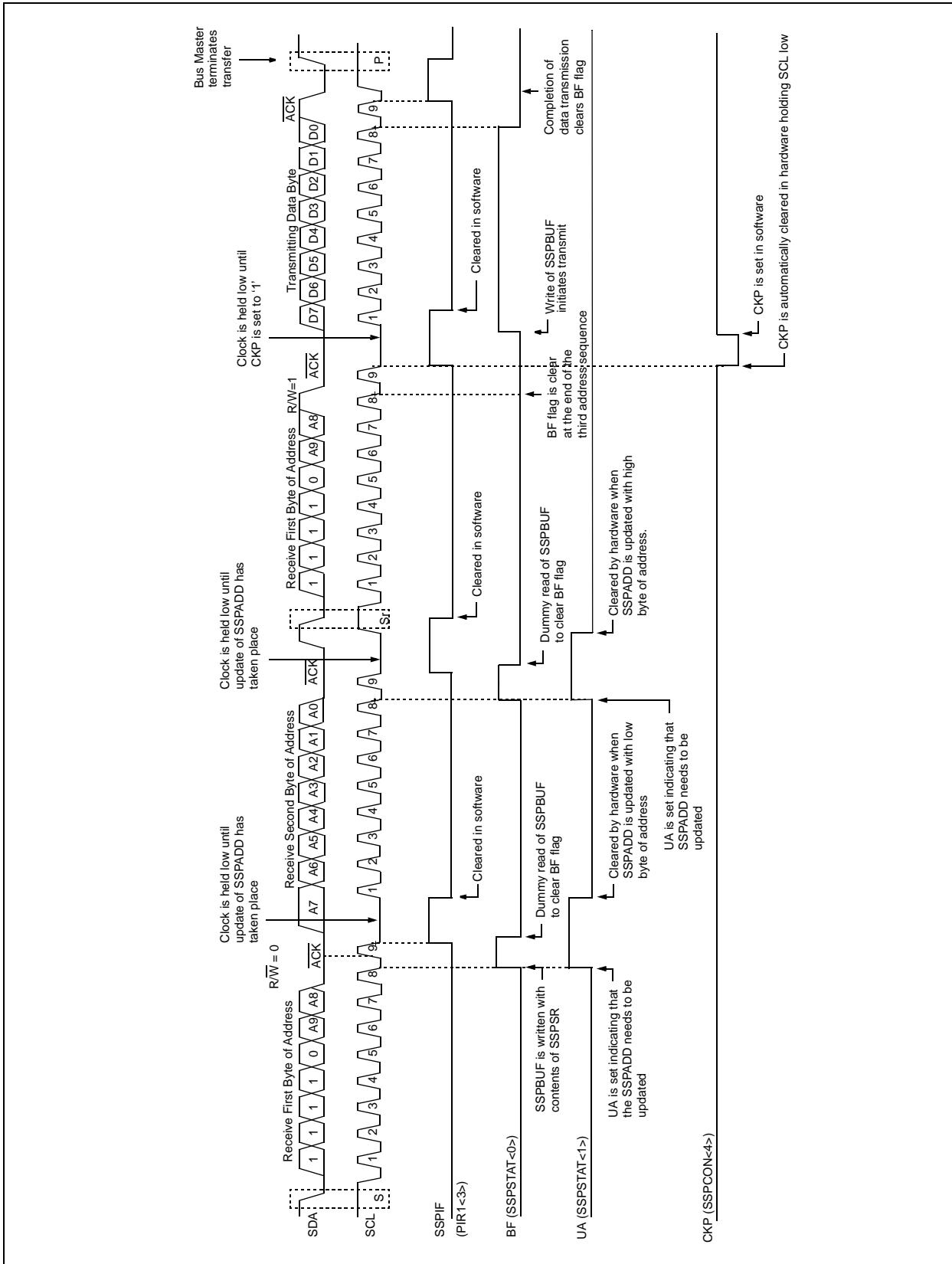


# **PIC18FXX2**

**FIGURE 15-10: I<sup>2</sup>C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)**



**FIGURE 15-11: I<sup>2</sup>C SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)**



## 15.4.4 CLOCK STRETCHING

Both 7- and 10-bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

### 15.4.4.1 Clock Stretching for 7-bit Slave Receive Mode (SEN = 1)

In 7-bit Slave Receive mode, on the falling edge of the ninth clock at the end of the  $\overline{\text{ACK}}$  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 15-13).

- 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.

### 15.4.4.2 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.

### 15.4.4.3 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 15-9).

- 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.

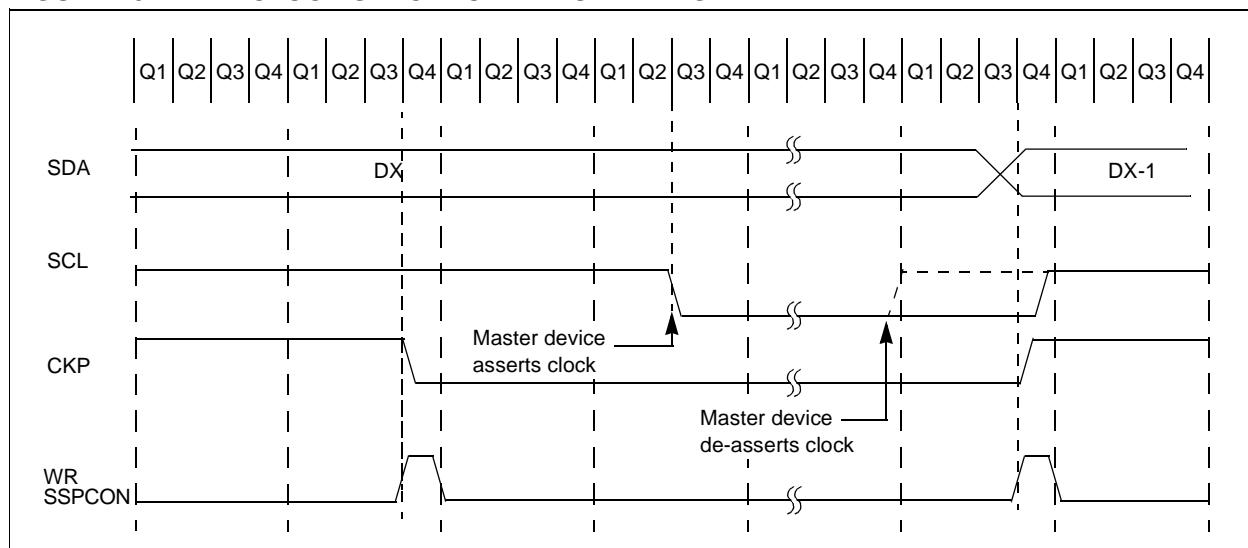
### 15.4.4.4 Clock Stretching for 10-bit Slave Transmit Mode

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 15-11).

#### 15.4.4.5 Clock Synchronization and the CKP bit

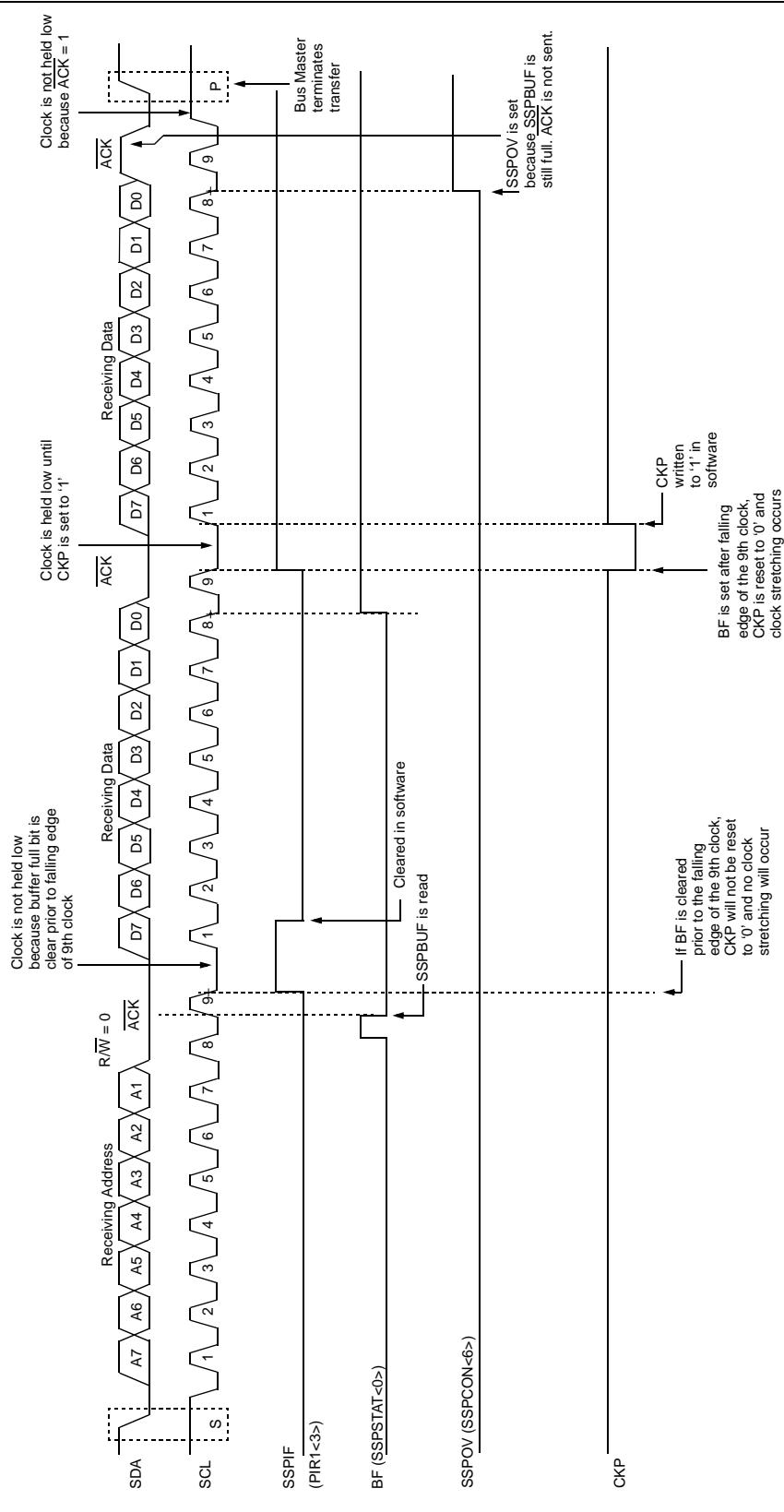
If a user clears the CKP bit, the SCL output is forced to '0'. Setting the CKP bit will not assert the SCL output low until the SCL output is already sampled low. If the user attempts to drive SCL low, the CKP bit will not assert the SCL line until an external I<sup>2</sup>C master device has already 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 de-asserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 15-12).

**FIGURE 15-12: CLOCK SYNCHRONIZATION TIMING**

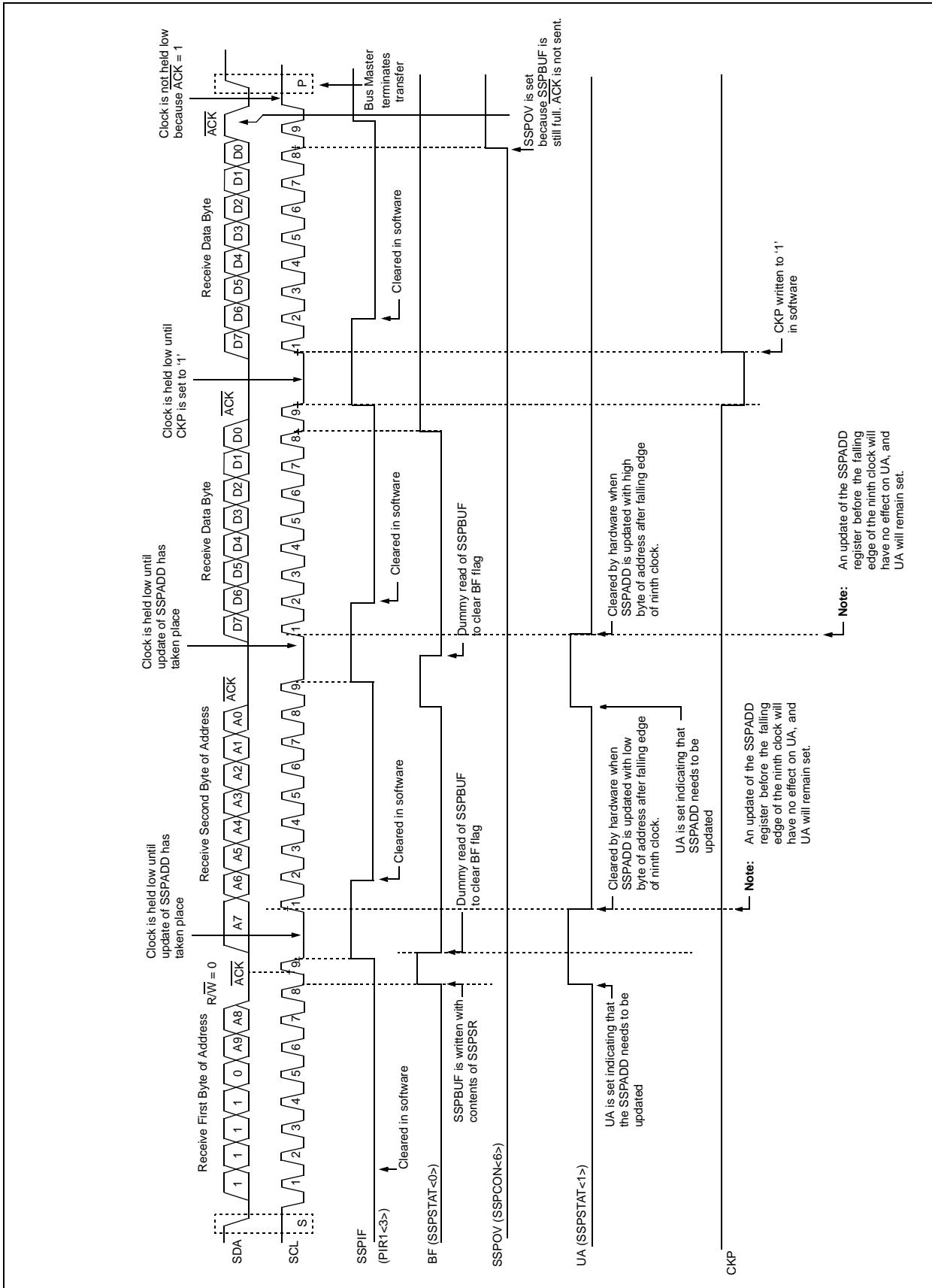


# PIC18FXX2

**FIGURE 15-13: I<sup>2</sup>C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)**



**FIGURE 15-14: I<sup>2</sup>C SLAVE MODE TIMING SEN = 1 (RECEPTION, 10-BIT ADDRESS)**



## 15.4.5 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I<sup>2</sup>C bus is such that the first byte after the START condition usually determines which device will be the slave addressed by the master. The exception is the general call address, which can address all devices. When this address is used, all devices should, in theory, respond with an Acknowledge.

The general call address is one of eight addresses reserved for specific purposes by the I<sup>2</sup>C protocol. It consists of all 0's with R/W = 0.

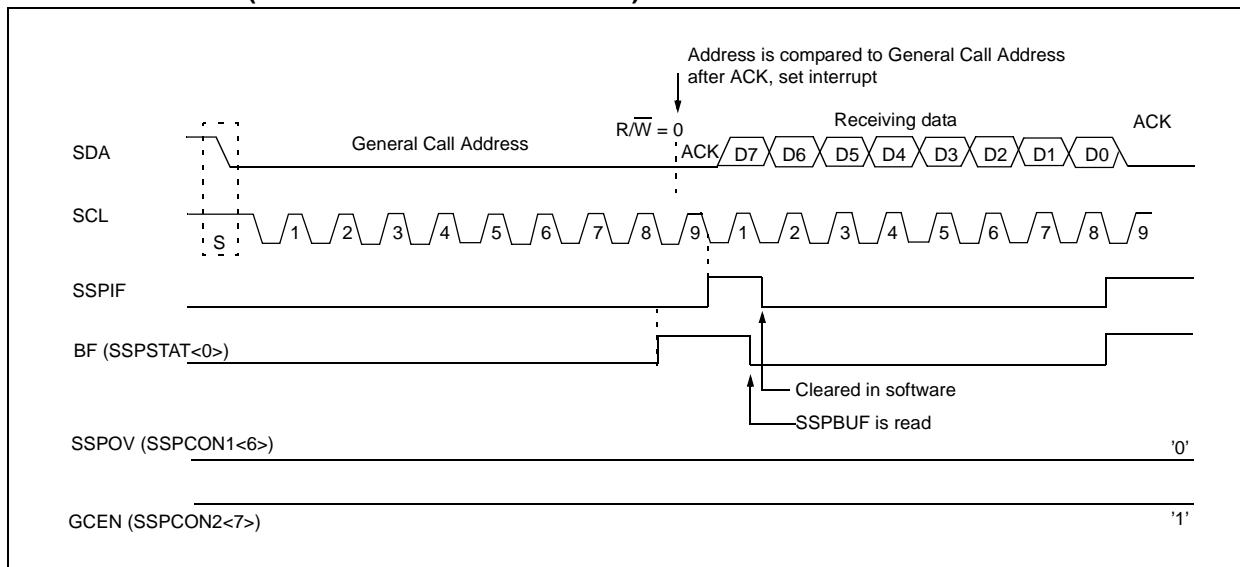
The general call address is recognized when the General Call Enable bit (GCEN) is enabled (SSPCON2<7> set). Following a START bit detect, 8-bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF flag bit is set (eighth bit), and on the falling edge of the ninth bit (ACK bit), the SSPIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPBUF. The value can be used to determine if the address was device specific or a general call address.

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match, and the UA bit is set (SSPSTAT<1>). If the general call address is sampled when the GCEN bit is set, while the slave is configured in 10-bit Address 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 15-15).

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



#### 15.4.6 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.

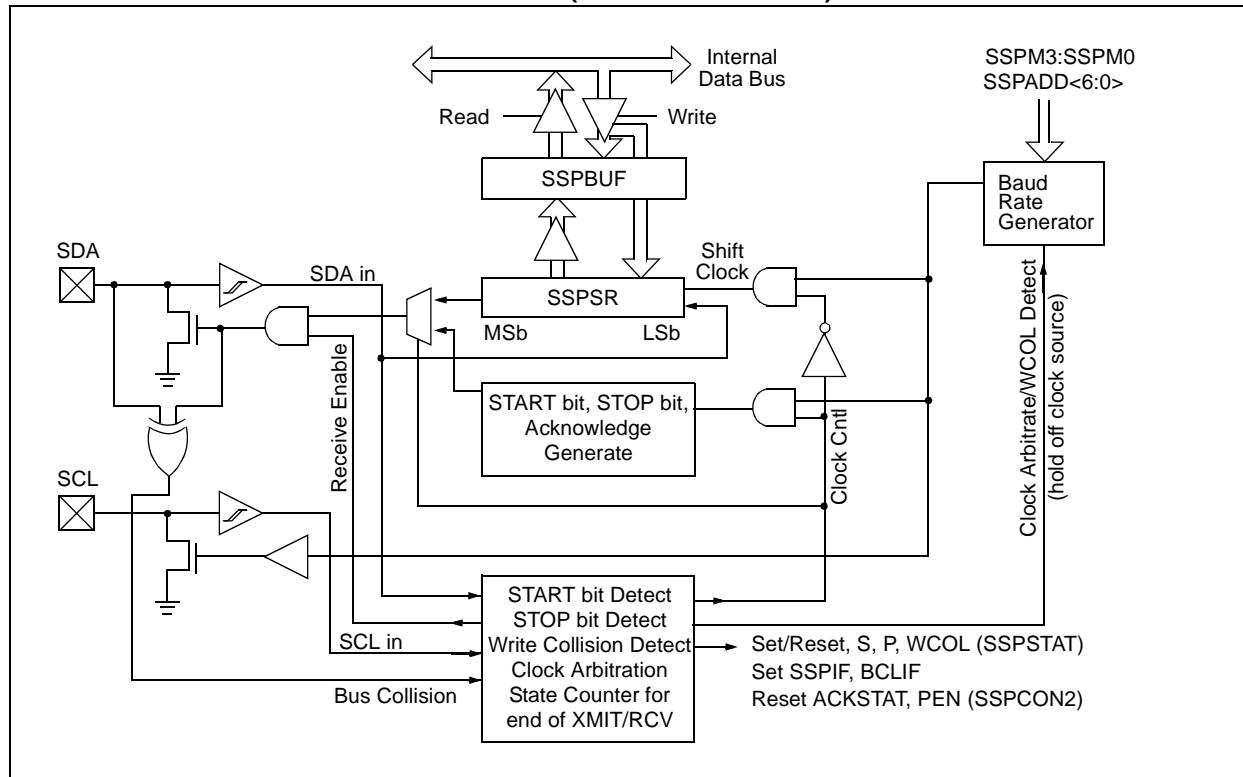
1. Assert a START condition on SDA and SCL.
2. Assert a Repeated START condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the I<sup>2</sup>C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. 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 SSP interrupt flag bit, SSPIF, to be set (SSP interrupt if enabled):

- START condition
- STOP condition
- Data transfer byte transmitted/received
- Acknowledge Transmit
- Repeated START

**FIGURE 15-16: MSSP BLOCK DIAGRAM (I<sup>2</sup>C MASTER MODE)**



## 15.4.6.1 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 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 15.4.7 ("Baud Rate Generator"), for more detail.

A typical transmit sequence would go as follows:

1. The user generates a START condition by setting the START enable bit, SEN (SSPCON2<0>).
2. SSPIF is set. The MSSP module will wait the required start time before any other operation takes place.
3. The user loads the SSPBUF with the slave address to transmit.
4. Address is shifted out the SDA pin until all 8 bits are transmitted.
5. The MSSP Module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
7. The user loads the SSPBUF with eight bits of data.
8. Data is shifted out the SDA pin until all 8 bits are transmitted.
9. The MSSP Module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
11. The user generates a STOP condition by setting the STOP enable bit PEN (SSPCON2<2>).
12. Interrupt is generated once the STOP condition is complete.

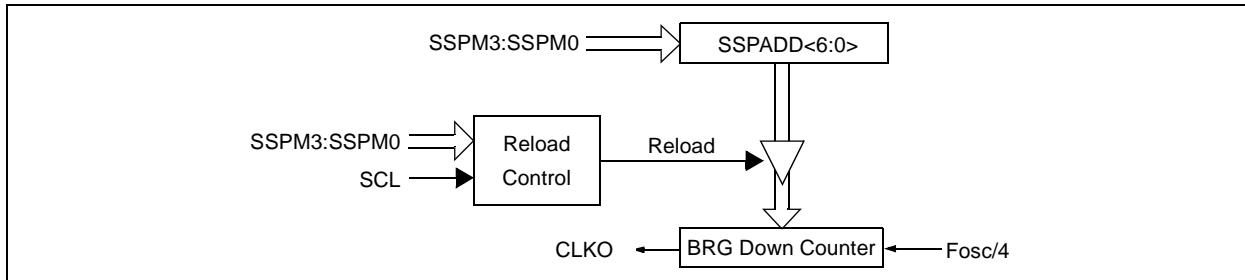
#### 15.4.7 BAUD RATE GENERATOR

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 15-17). 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 15-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

**FIGURE 15-17: BAUD RATE GENERATOR BLOCK DIAGRAM**



**TABLE 15-3: I<sup>2</sup>C CLOCK RATE W/BRG**

F <sub>CY</sub>	F <sub>CY</sub> *2	BRG Value	F <sub>SCL</sub> <sup>(2)</sup> (2 Rollovers of BRG)
10 MHz	20 MHz	19h	400 kHz <sup>(1)</sup>
10 MHz	20 MHz	20h	312.5 kHz
10 MHz	20 MHz	3Fh	100 kHz
4 MHz	8 MHz	0Ah	400 kHz <sup>(1)</sup>
4 MHz	8 MHz	0Dh	308 kHz
4 MHz	8 MHz	28h	100 kHz
1 MHz	2 MHz	03h	333 kHz <sup>(1)</sup>
1 MHz	2 MHz	0Ah	100 kHz
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.

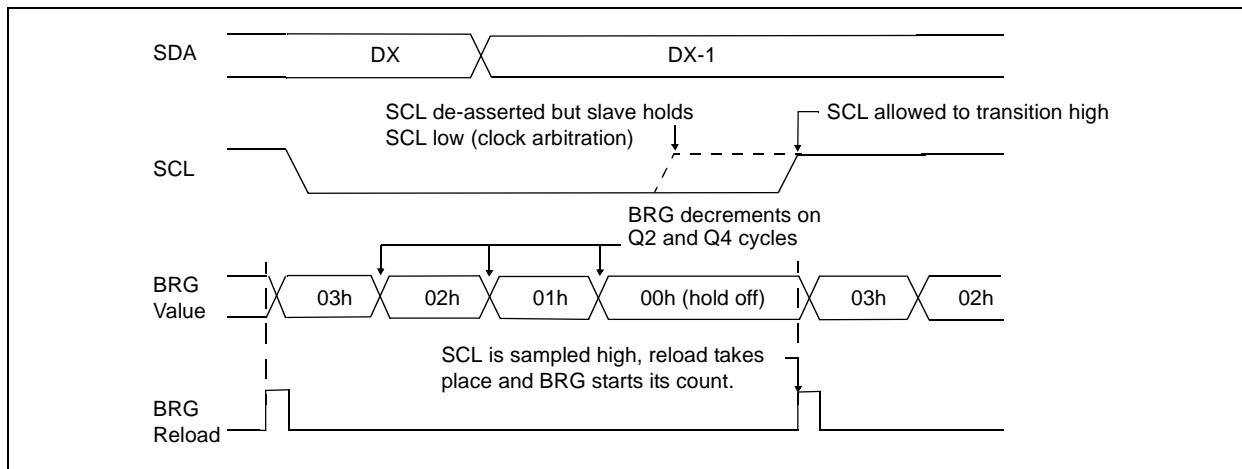
**2:** Actual frequency will depend on bus conditions. Theoretically, bus conditions will add rise time and extend low time of clock period, producing the effective frequency.

## 15.4.7.1 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated START/STOP condition, de-asserts 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 15-18).

**FIGURE 15-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**



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

To initiate a START condition, the user sets the START condition enable bit, SEN (SSPCON2<0>). If the SDA and SCL pins are sampled high, the baud rate generator is reloaded with the contents of SSPADD<6:0> and starts its count. If SCL and SDA are both sampled high when the baud rate generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low, while SCL is high, is the START condition and causes the S bit (SSPSTAT<3>) to be set. Following this, the baud rate generator is reloaded with the contents of SSPADD<6:0> and resumes its count. When the baud rate generator times out (TBRG), the SEN bit (SSPCON2<0>) will be automatically cleared by hardware, the baud rate generator is suspended, leaving the SDA line held low and the START condition is complete.

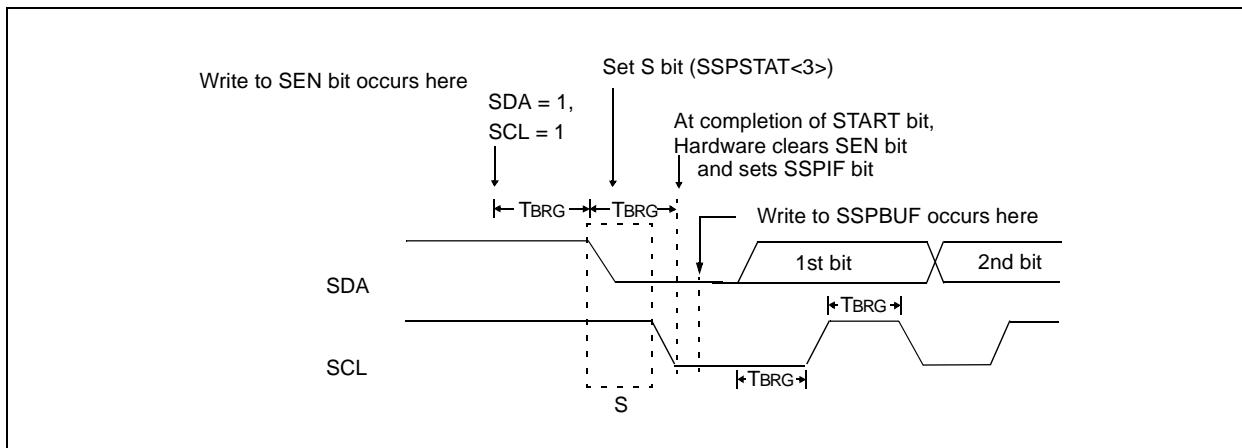
**Note:** If at the beginning of the START condition, the SDA and SCL pins are already sampled low, or if during the START condition the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF is set, the START condition is aborted, and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

#### 15.4.8.1 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 15-19: FIRST START BIT TIMING**



## 15.4.9 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 de-asserted (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".

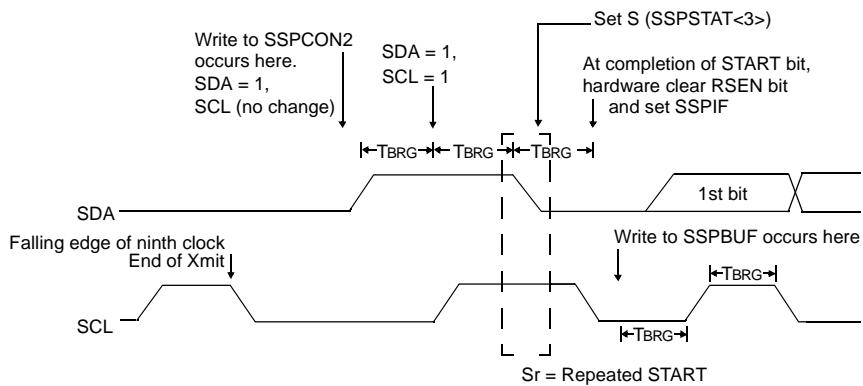
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode)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

### 15.4.9.1 WCOL Status Flag

If the user writes the SSPBUF when a Repeated START sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

**Note:** Because queuing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated START condition is complete.

**FIGURE 15-20: REPEAT START CONDITION WAVEFORM**



### 15.4.10 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 15-21).

After the write to the SSPBUF, each bit of address will be shifted out on the falling edge of SCL until all seven address bits and the R/W bit are completed. On the falling edge of the eighth clock, the master will de-assert 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.

#### 15.4.10.1 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.

#### 15.4.10.2 WCOL Status Flag

If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

WCOL must be cleared in software.

#### 15.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPCON2<6>) is cleared when the slave has sent an Acknowledge (ACK = 0), and is set when the slave does not Acknowledge (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.

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

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

**Note:** In the MSSP module, the RCEN bit must be set after the ACK sequence 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>).

#### 15.4.11.1 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.

#### 15.4.11.2 SSPOV Status Flag

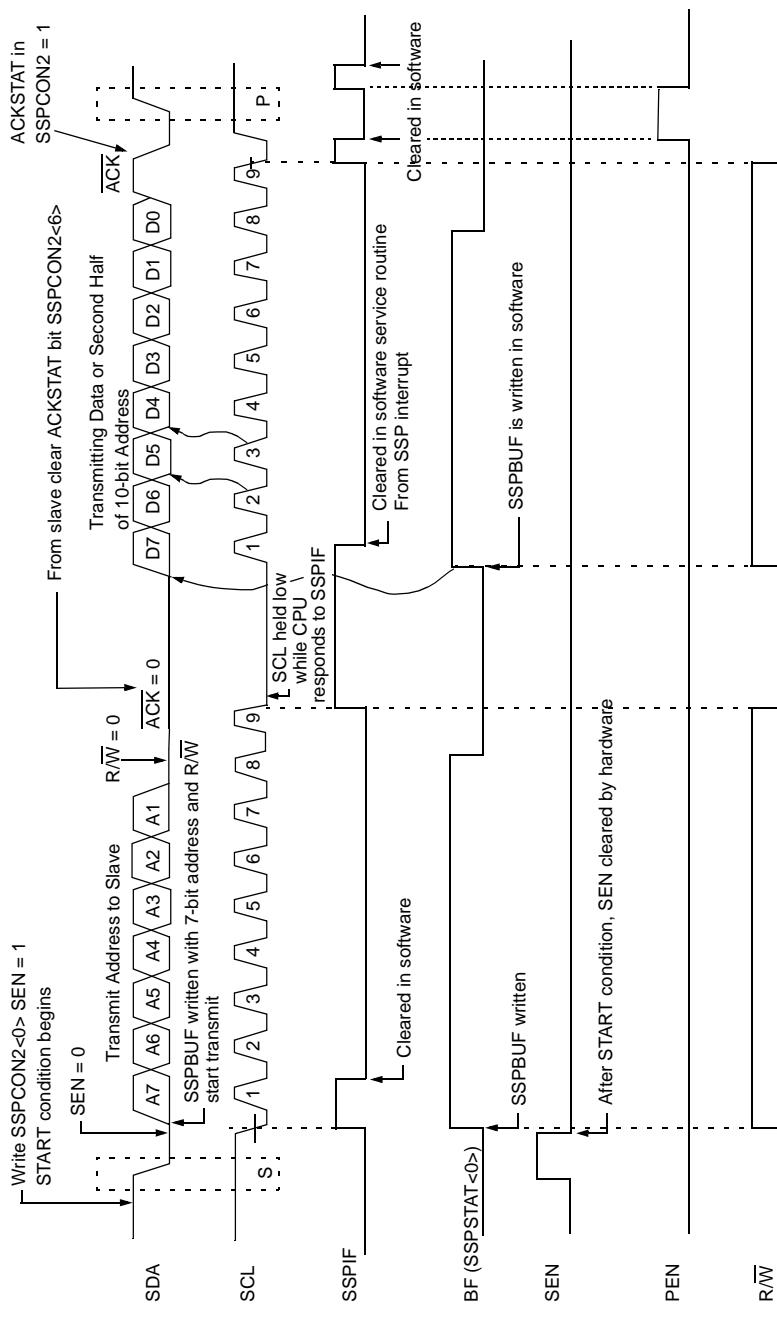
In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

#### 15.4.11.3 WCOL Status Flag

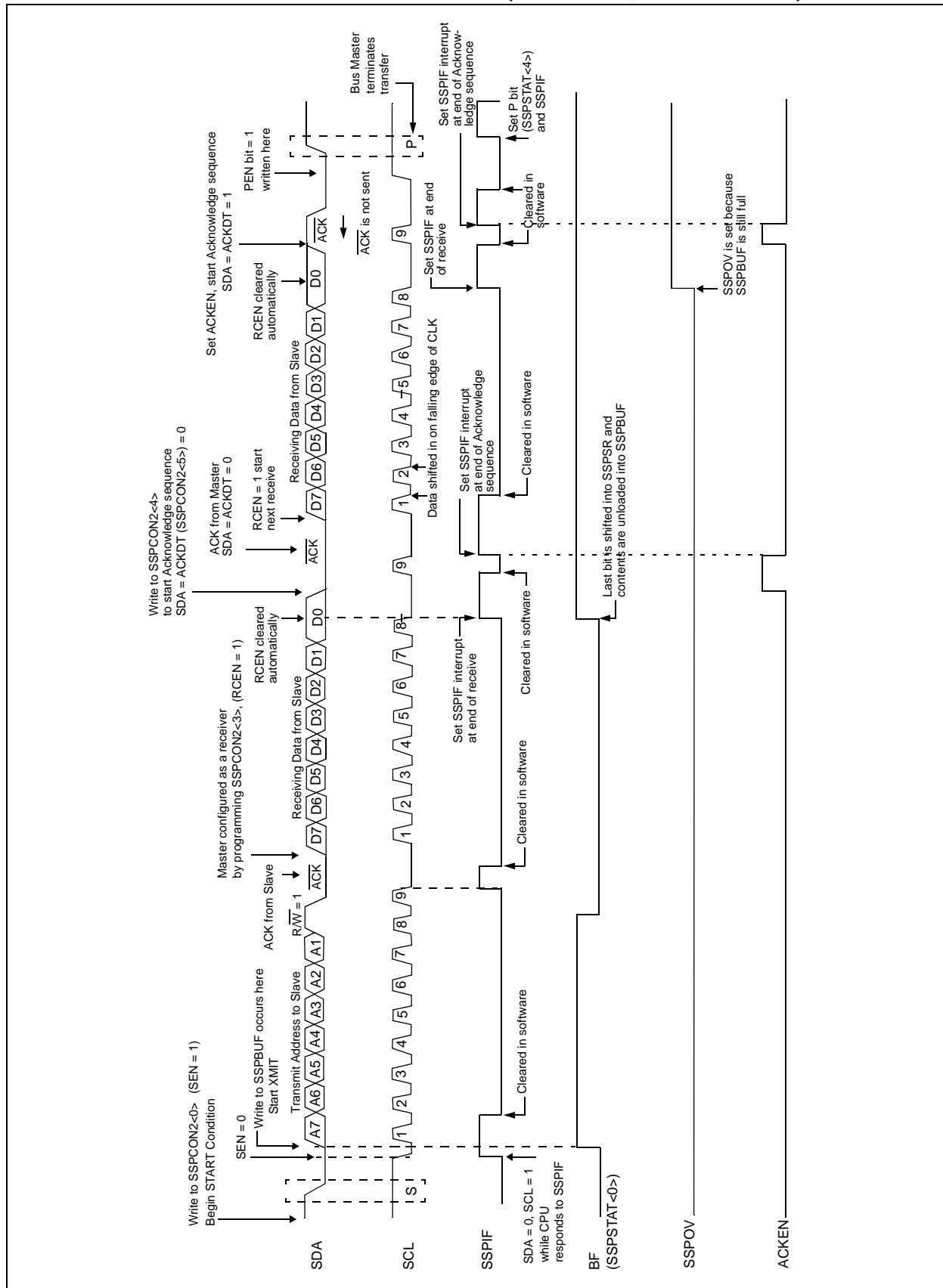
If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

# PIC18FXX2

**FIGURE 15-21: I<sup>2</sup>C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)**



**FIGURE 15-22:** I<sup>2</sup>C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



## 15.4.12 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 de-asserted (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 15-23).

### 15.4.12.1 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).

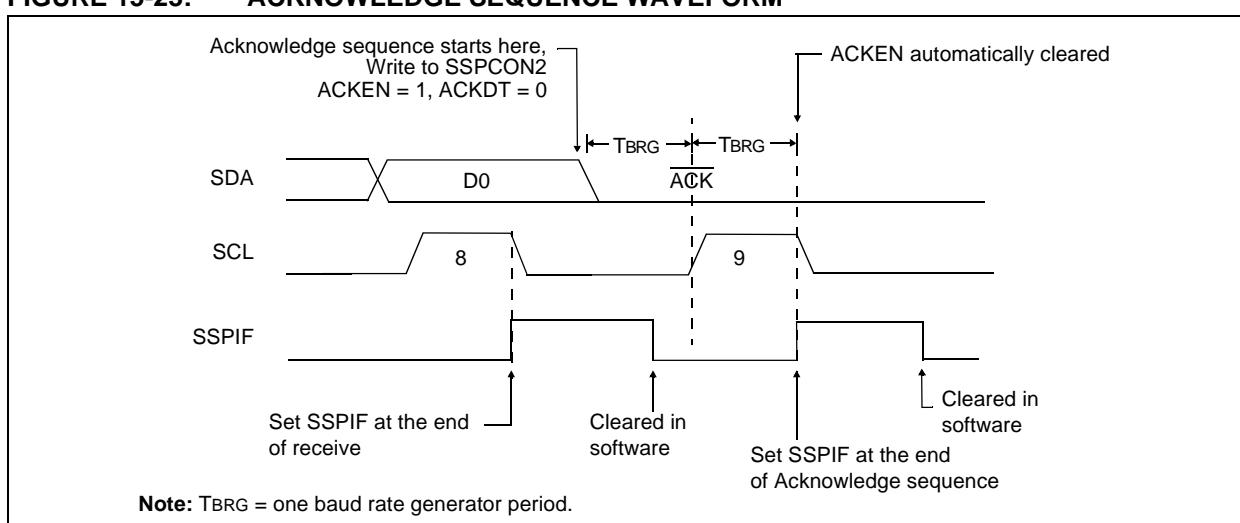
## 15.4.13 STOP CONDITION TIMING

A STOP bit is asserted on the SDA pin at the end of a receive/transmit by setting the STOP sequence enable bit, PEN (SSPCON2<2>). At the end of a receive/transmit the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the baud rate generator is reloaded and counts down to 0. When the baud rate generator times out, the SCL pin will be brought high, and one TBRG (baud rate generator rollover count) later, the SDA pin will be de-asserted. When the SDA pin is sampled high while SCL is high, the P bit (SSPSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 15-24).

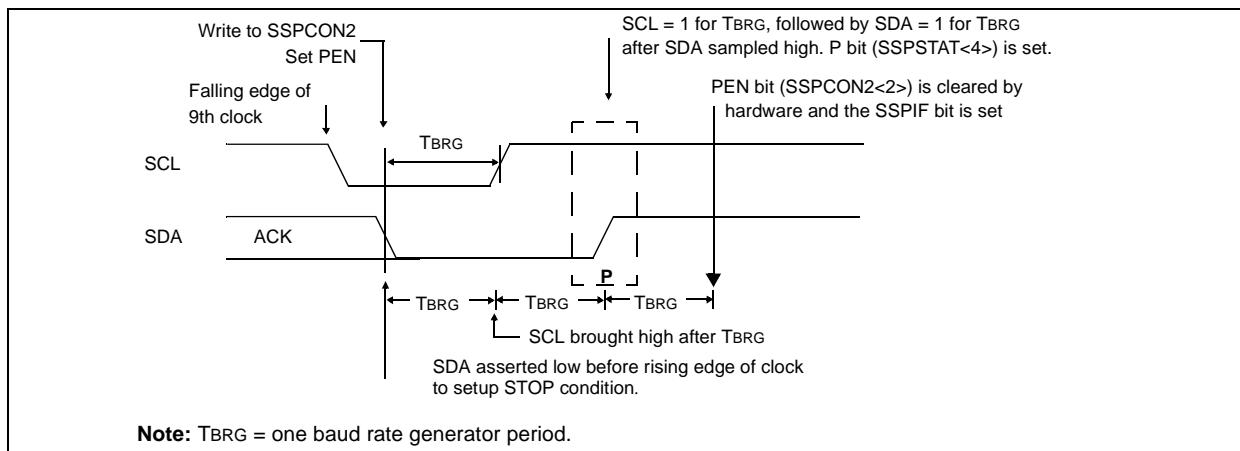
### 15.4.13.1 WCOL Status Flag

If the user writes the SSPBUF when a STOP sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 15-23: ACKNOWLEDGE SEQUENCE WAVEFORM**



**FIGURE 15-24: STOP CONDITION RECEIVE OR TRANSMIT MODE**



#### 15.4.14 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).

#### 15.4.15 EFFECT OF A RESET

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

#### 15.4.16 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 SSP 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

#### 15.4.17 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 15-25).

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 de-asserted, 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.

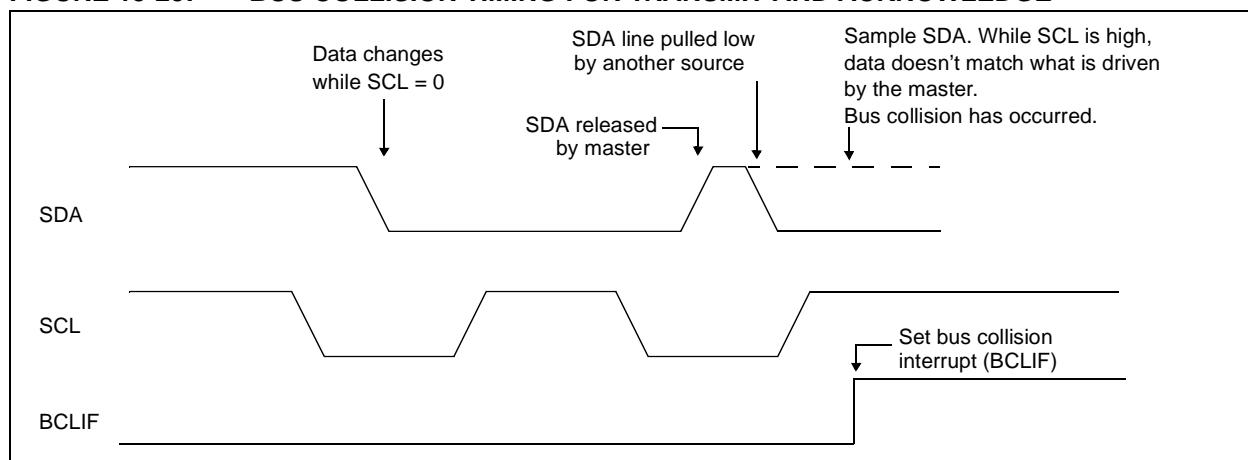
If a START, Repeated START, STOP, or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are de-asserted, and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service 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 15-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**



## 15.4.17.1 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 15-26).
- SCL is sampled low before SDA is asserted low (Figure 15-27).

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:

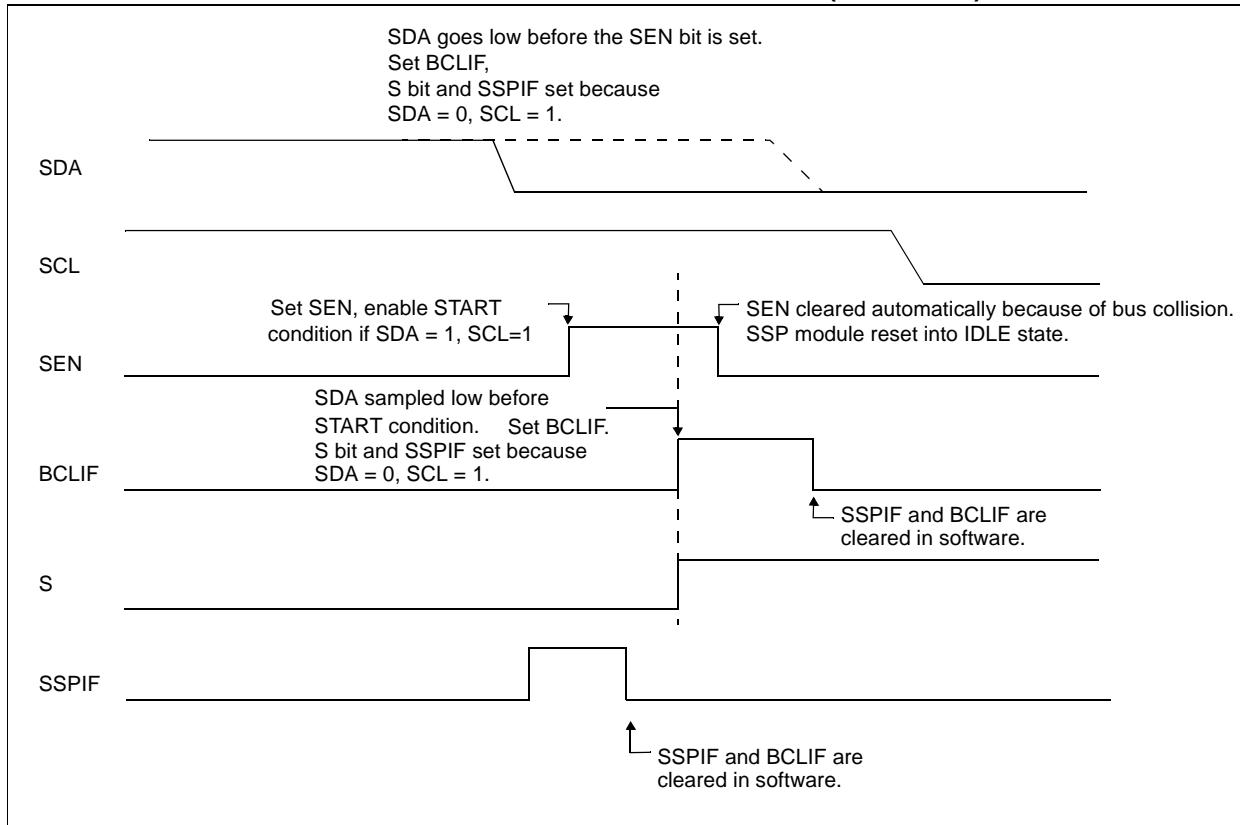
- the START condition is aborted,
- the BCLIF flag is set, and
- the MSSP module is reset to its IDLE state (Figure 15-26).

The START condition begins with the SDA and SCL pins de-asserted. 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.

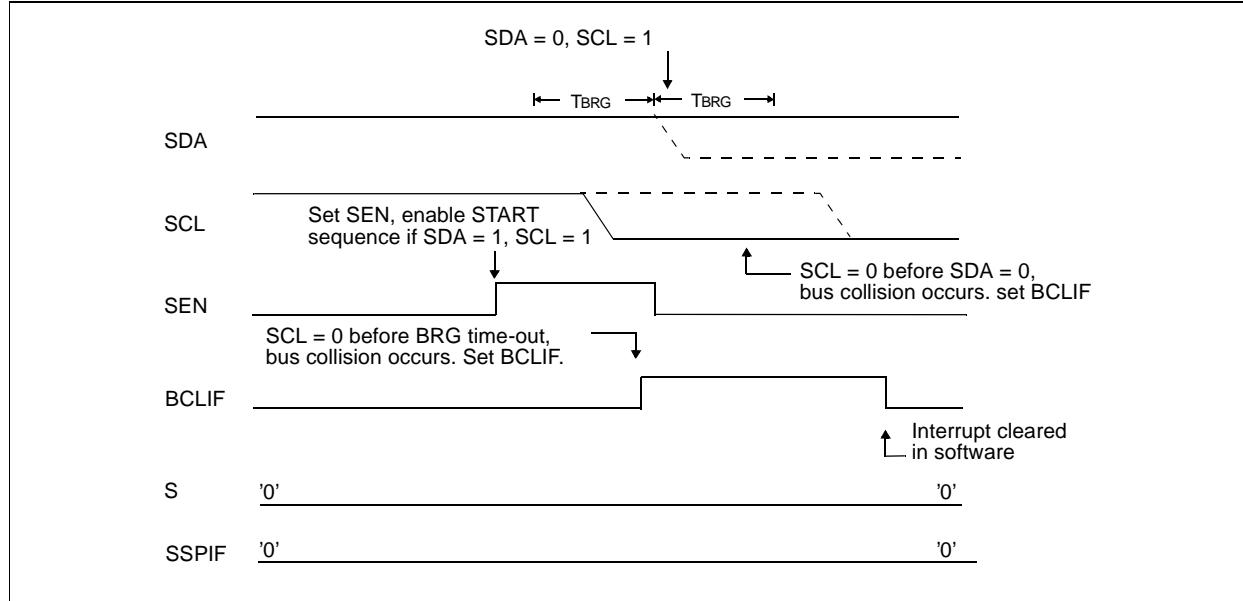
If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 15-28). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The baud rate generator is then reloaded and counts down to 0, and during this time, if the SCL pins are sampled as '0', a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

**Note:** The reason that bus collision is not a factor during a START condition is that no two bus masters can assert a START condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision, because the two masters must be allowed to arbitrate the first address following the START condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated START or STOP conditions.

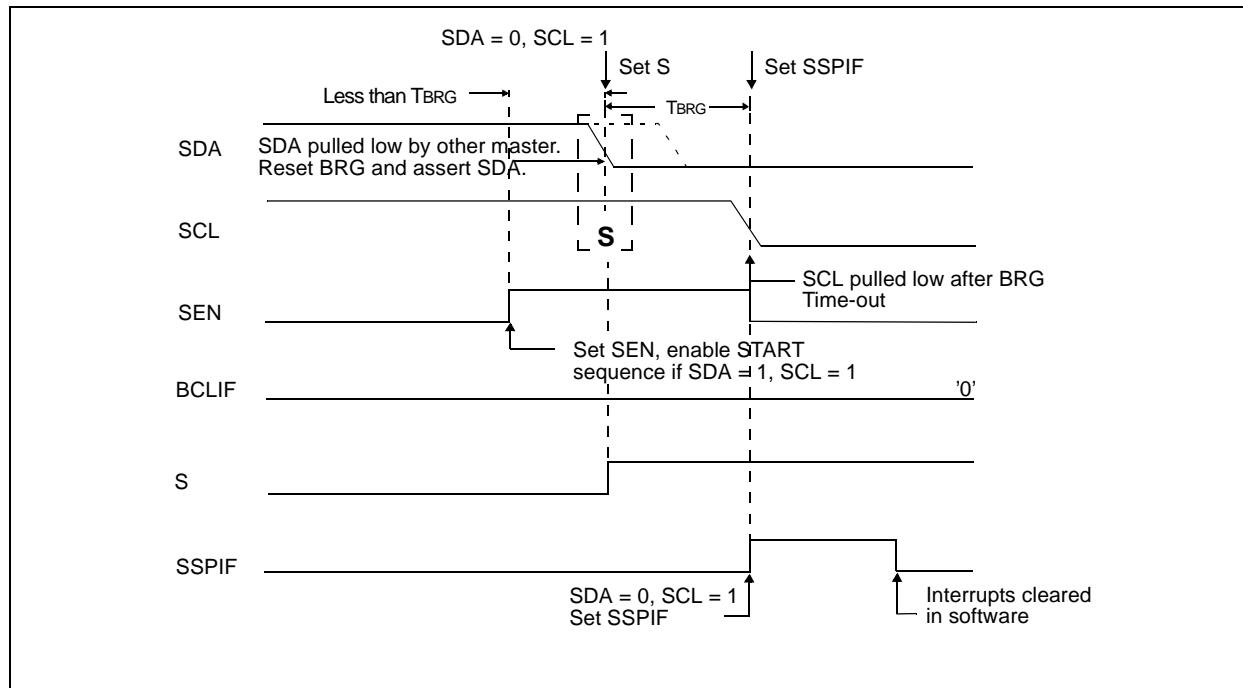
**FIGURE 15-26: BUS COLLISION DURING START CONDITION (SDA ONLY)**



**FIGURE 15-27: BUS COLLISION DURING START CONDITION (SCL = 0)**



**FIGURE 15-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION**



## 15.4.17.2 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 de-asserts 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 de-asserted, and when sampled high, the SDA pin is sampled.

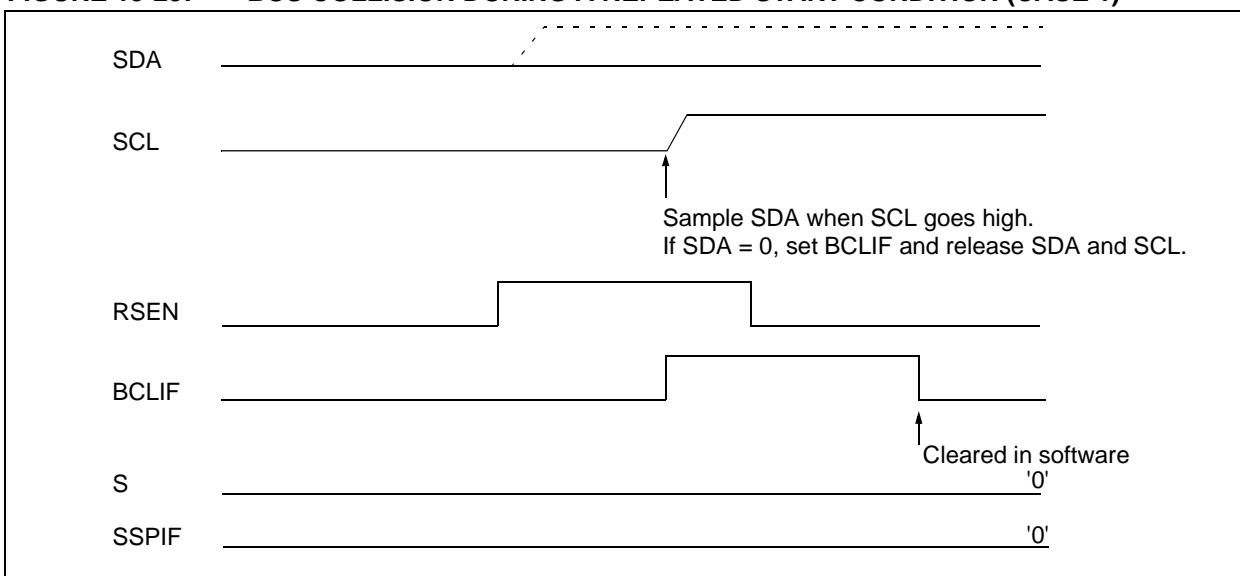
If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', Figure 15-29). If SDA is sampled high, the BRG is

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.

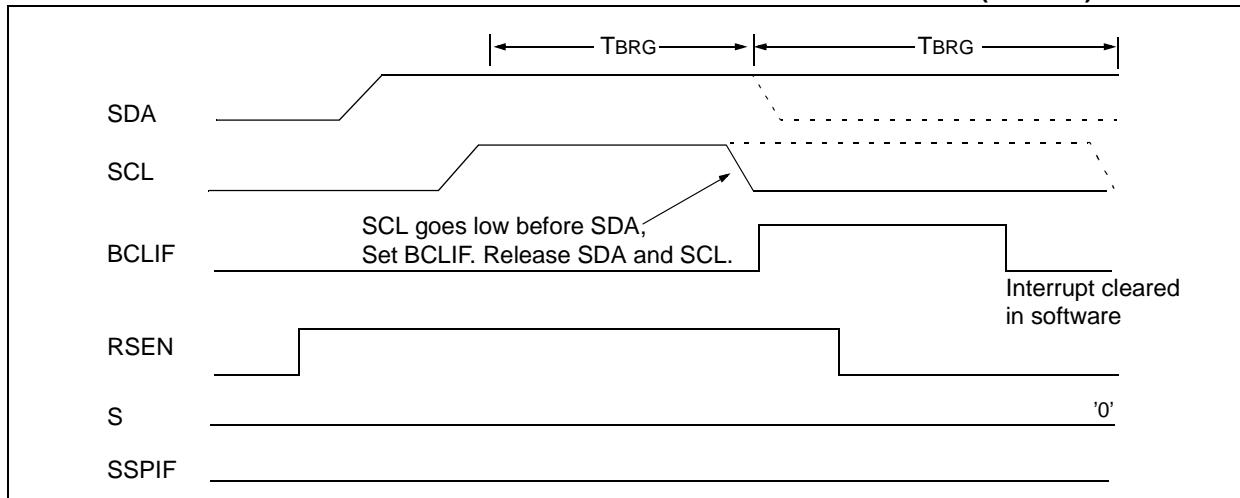
If SCL goes from high to low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated START condition, Figure 15-30.

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 15-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**



**FIGURE 15-30: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)**



### 15.4.17.3 Bus Collision During a STOP Condition

Bus collision occurs during a STOP condition if:

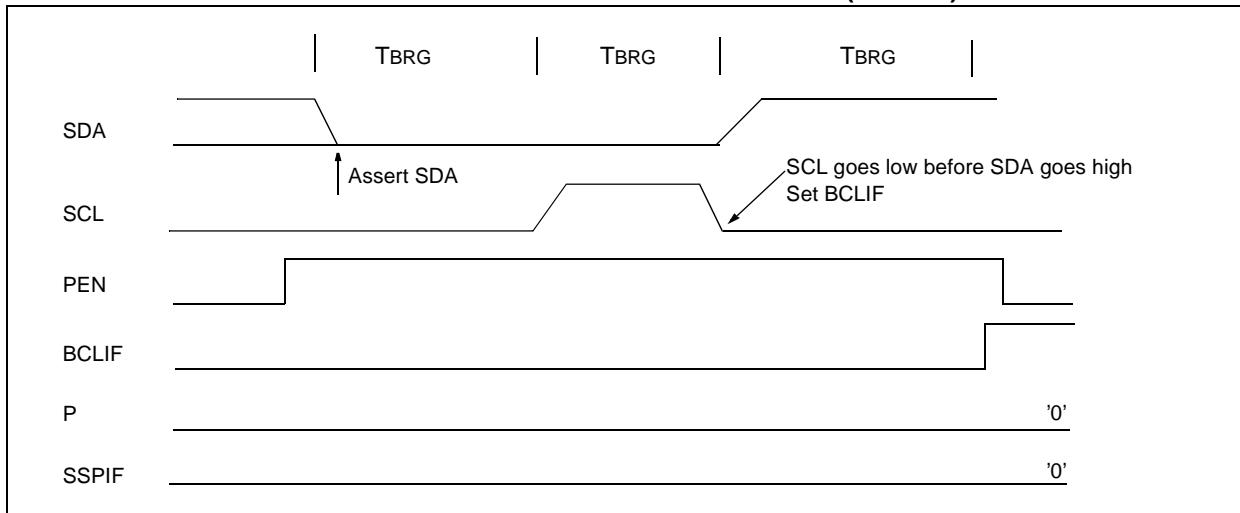
- After the SDA pin has been de-asserted and allowed to float high, SDA is sampled low after the BRG has timed out.
- After the SCL pin is de-asserted, SCL is sampled low before SDA goes high.

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 15-31). 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 15-32).

**FIGURE 15-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)**



**FIGURE 15-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)**



# **PIC18FXX2**

---

---

## **NOTES:**

## 16.0 ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART can be configured as a full duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers, or it can 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 USART can be configured in the following modes:

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

In order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter:

- bit SPEN (RCSTA<7>) must be set (= 1),
- bit TRISC<6> must be cleared (= 0), and
- bit TRISC<7> must be set (=1).

Register 16-1 shows the Transmit Status and Control Register (TXSTA) and Register 16-2 shows the Receive Status and Control Register (RCSTA).

# **PIC18FXX2**

**REGISTER 16-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER**

- |       |   |
|-------|---|
| bit 7 | <b>CSRC:</b> Clock Source Select bit<br><br><u>Asynchronous mode:</u><br>Don't care<br><br><u>Synchronous mode:</u><br>1 = Master mode (clock generated internally from BRG)<br>0 = Slave mode (clock from external source) |
| bit 6 | <b>TX9:</b> 9-bit Transmit Enable bit<br>1 = Selects 9-bit transmission<br>0 = Selects 8-bit transmission   |
| bit 5 | <b>TXEN:</b> Transmit Enable bit<br>1 = Transmit enabled<br>0 = Transmit disabled<br><br><b>Note:</b> SREN/CREN overrides TXEN in SYNC mode.  |
| bit 4 | <b>SYNC:</b> USART Mode Select bit<br>1 = Synchronous mode<br>0 = Asynchronous mode   |
| bit 3 | <b>Unimplemented:</b> Read as '0'   |
| bit 2 | <b>BRGH:</b> High Baud Rate Select bit<br><br><u>Asynchronous mode:</u><br>1 = High speed<br>0 = Low speed<br><br><u>Synchronous mode:</u><br>Unused in this mode   |
| bit 1 | <b>TRMT:</b> Transmit Shift Register Status bit<br>1 = TSR empty<br>0 = TSR full  |
| bit 0 | <b>TX9D:</b> 9th bit of Transmit Data<br>Can be Address/Data bit or a parity bit.   |

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

## REGISTER 16-2: 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

- bit 7      **SPEN:** Serial Port Enable bit  
               1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)  
               0 = Serial port disabled
- bit 6      **RX9:** 9-bit Receive Enable bit  
               1 = Selects 9-bit reception  
               0 = Selects 8-bit reception
- bit 5      **SREN:** Single Receive Enable bit  
Asynchronous mode:  
               Don't care  
Synchronous mode - Master:  
               1 = Enables single receive  
               0 = Disables single receive  
               This bit is cleared after reception is complete.  
Synchronous mode - Slave:  
               Don't care
- bit 4      **CREN:** Continuous Receive Enable bit  
Asynchronous mode:  
               1 = Enables receiver  
               0 = Disables receiver  
Synchronous mode:  
               1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)  
               0 = Disables continuous receive
- bit 3      **ADDEN:** Address Detect Enable bit  
Asynchronous mode 9-bit (RX9 = 1):  
               1 = Enables address detection, enable interrupt and load of 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
- bit 2      **FERR:** Framing Error bit  
               1 = Framing error (can be updated by reading RCREG register and receive next valid byte)  
               0 = No framing error
- bit 1      **OERR:** Overrun Error bit  
               1 = Overrun error (can be cleared by clearing bit CREN)  
               0 = No overrun error
- bit 0      **RX9D:** 9th bit of Received Data  
               This can be Address/Data bit or a parity bit, and must be calculated by user firmware.

## 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

## 16.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit baud rate generator. The SPBRG register controls the period of a free running 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA<2>) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 16-1 shows the formula for computation of the baud rate for different USART modes, which only apply in Master mode (internal clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRG register can be calculated using the formula in Table 16-1. From this, the error in baud rate can be determined.

Example 16-1 shows the calculation of the baud rate error for the following conditions:

- Fosc = 16 MHz
- Desired Baud Rate = 9600
- BRGH = 0
- SYNC = 0

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the  $Fosc/(16(X + 1))$  equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

### 16.1.1 SAMPLING

The data on the RC7/RX/DT 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.

### EXAMPLE 16-1: CALCULATING BAUD RATE ERROR

Desired Baud Rate	= FOSC / (64 (X + 1))
Solving for X:	
X	= ((FOSC / Desired Baud Rate) / 64) - 1
X	= ((16000000 / 9600) / 64) - 1
X	= [25.042] = 25
Calculated Baud Rate	= 16000000 / (64 (25 + 1))
	= 9615
Error	= $\frac{(\text{Calculated Baud Rate} - \text{Desired Baud Rate})}{\text{Desired Baud Rate}}$
	= $\frac{(9615 - 9600)}{9600}$
	= 0.16%

TABLE 16-1: BAUD RATE FORMULA

SYNC	BRGH = 0 (Low Speed)	BRGH = 1 (High Speed)
0	(Asynchronous) Baud Rate = Fosc/(64(X+1))	Baud Rate = Fosc/(16(X+1))
1	(Synchronous) Baud Rate = Fosc/(4(X+1))	N/A

Legend: X = value in SPBRG (0 to 255)

TABLE 16-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used by the BRG.

**TABLE 16-3: BAUD RATES FOR SYNCHRONOUS MODE**

BAUD RATE (Kbps)	Fosc = 40 MHz			33 MHz			25 MHz			20 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	NA	-	-	NA	-	-	NA	-	-
19.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
76.8	76.92	+0.16	129	77.10	+0.39	106	77.16	+0.47	80	76.92	+0.16	64
96	96.15	+0.16	103	95.93	-0.07	85	96.15	+0.16	64	96.15	+0.16	51
300	303.03	+1.01	32	294.64	-1.79	27	297.62	-0.79	20	294.12	-1.96	16
500	500	0	19	485.30	-2.94	16	480.77	-3.85	12	500	0	9
HIGH	10000	-	0	8250	-	0	6250	-	0	5000	-	0
LOW	39.06	-	255	32.23	-	255	24.41	-	255	19.53	-	255
BAUD RATE (Kbps)	Fosc = 16 MHz			10 MHz			7.15909 MHz			5.0688 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	NA	-	-	9.62	+0.23	185	9.60	0	131
19.2	19.23	+0.16	207	19.23	+0.16	129	19.24	+0.23	92	19.20	0	65
76.8	76.92	+0.16	51	75.76	-1.36	32	77.82	+1.32	22	74.54	-2.94	16
96	95.24	-0.79	41	96.15	+0.16	25	94.20	-1.88	18	97.48	+1.54	12
300	307.70	+2.56	12	312.50	+4.17	7	298.35	-0.57	5	316.80	+5.60	3
500	500	0	7	500	0	4	447.44	-10.51	3	422.40	-15.52	2
HIGH	4000	-	0	2500	-	0	1789.80	-	0	1267.20	-	0
LOW	15.63	-	255	9.77	-	255	6.99	-	255	4.95	-	255
BAUD RATE (Kbps)	Fosc = 4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-	0.30	+1.14	26
1.2	NA	-	-	NA	-	-	1.20	+0.16	207	1.17	-2.48	6
2.4	NA	-	-	NA	-	-	2.40	+0.16	103	2.73	+13.78	2
9.6	9.62	+0.16	103	9.62	+0.23	92	9.62	+0.16	25	8.20	-14.67	0
19.2	19.23	+0.16	51	19.04	-0.83	46	19.23	+0.16	12	NA	-	-
76.8	76.92	+0.16	12	74.57	-2.90	11	83.33	+8.51	2	NA	-	-
96	1000	+4.17	9	99.43	+3.57	8	83.33	-13.19	2	NA	-	-
300	333.33	+11.11	2	298.30	-0.57	2	250	-16.67	0	NA	-	-
500	500	0	1	447.44	-10.51	1	NA	-	-	NA	-	-
HIGH	1000	-	0	894.89	-	0	250	-	0	8.20	-	0
LOW	3.91	-	255	3.50	-	255	0.98	-	255	0.03	-	255

# PIC18FXX2

---

**TABLE 16-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)**

BAUD RATE (Kbps)	Fosc = 40 MHz		33 MHz		25 MHz		20 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	2.40	-0.07	214	2.40	-0.15	162
9.6	9.62	+0.16	64	9.55	-0.54	53	9.53	-0.76	40
19.2	18.94	-1.36	32	19.10	-0.54	26	19.53	+1.73	19
76.8	78.13	+1.73	7	73.66	-4.09	6	78.13	+1.73	4
96	89.29	-6.99	6	103.13	+7.42	4	97.66	+1.73	3
300	312.50	+4.17	1	257.81	-14.06	1	NA	-	312.50
500	625	+25.00	0	NA	-	-	NA	-	-
HIGH	625	-	0	515.63	-	0	390.63	-	0
LOW	2.44	-	255	2.01	-	255	1.53	-	255
<hr/>									
BAUD RATE (Kbps)	Fosc = 16 MHz		10 MHz		7.15909 MHz		5.0688 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-
1.2	1.20	+0.16	207	1.20	+0.16	129	1.20	+0.23	92
2.4	2.40	+0.16	103	2.40	+0.16	64	2.38	-0.83	46
9.6	9.62	+0.16	25	9.77	+1.73	15	9.32	-2.90	11
19.2	19.23	+0.16	12	19.53	+1.73	7	18.64	-2.90	5
76.8	83.33	+8.51	2	78.13	+1.73	1	111.86	+45.65	0
96	83.33	-13.19	2	78.13	-18.62	1	NA	-	NA
300	250	-16.67	0	156.25	-47.92	0	NA	-	NA
500	NA	-	-	NA	-	-	NA	-	-
HIGH	250	-	0	156.25	-	0	111.86	-	0
LOW	0.98	-	255	0.61	-	255	0.44	-	255
<hr/>									
BAUD RATE (Kbps)	Fosc = 4 MHz		3.579545 MHz		1 MHz		32.768 kHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	0.30	-0.16	207	0.30	+0.23	185	0.30	+0.16	51
1.2	1.20	+1.67	51	1.19	-0.83	46	1.20	+0.16	12
2.4	2.40	+1.67	25	2.43	+1.32	22	2.23	-6.99	6
9.6	8.93	-6.99	6	9.32	-2.90	5	7.81	-18.62	1
19.2	20.83	+8.51	2	18.64	-2.90	2	15.63	-18.62	0
76.8	62.50	-18.62	0	55.93	-27.17	0	NA	-	NA
96	NA	-	-	NA	-	-	NA	-	-
300	NA	-	-	NA	-	-	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-
HIGH	62.50	-	0	55.93	-	0	15.63	-	0
LOW	0.24	-	255	0.22	-	255	0.06	-	255

**TABLE 16-5: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 1)**

BAUD RATE (Kbps)	Fosc = 40 MHz		33 MHz		25 MHz		20 MHz		
	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)	KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	NA	-	-
9.6	NA	-	-	9.60	-0.07	214	9.59	-0.15	162
19.2	19.23	+0.16	129	19.28	+0.39	106	19.30	+0.47	80
76.8	75.76	-1.36	32	76.39	-0.54	26	78.13	+1.73	19
96	96.15	+0.16	25	98.21	+2.31	20	97.66	+1.73	15
300	312.50	+4.17	7	294.64	-1.79	6	312.50	+4.17	4
500	500	0	4	515.63	+3.13	3	520.83	+4.17	2
HIGH	2500	-	0	2062.50	-	0	1562.50	-	0
LOW	9.77	-	255	8.06	-	255	6.10	-	255
<hr/>									
BAUD RATE (Kbps)	Fosc = 16 MHz		SPBRG value (decimal)	10 MHz		SPBRG value (decimal)	7.15909 MHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	NA	-	-
1.2	NA	-	-	NA	-	-	NA	-	-
2.4	NA	-	-	NA	-	-	2.41	+0.23	185
9.6	9.62	+0.16	103	9.62	+0.16	64	9.52	-0.83	46
19.2	19.23	+0.16	51	18.94	-1.36	32	19.45	+1.32	22
76.8	76.92	+0.16	12	78.13	+1.73	7	74.57	-2.90	5
96	100	+4.17	9	89.29	-6.99	6	89.49	-6.78	4
300	333.33	+11.11	2	312.50	+4.17	1	447.44	+49.15	0
500	500	0	1	625	+25.00	0	447.44	-10.51	0
HIGH	1000	-	0	625	-	0	447.44	-	0
LOW	3.91	-	255	2.44	-	255	1.75	-	255
<hr/>									
BAUD RATE (Kbps)	Fosc = 4 MHz		SPBRG value (decimal)	3.579545 MHz		SPBRG value (decimal)	1 MHz		SPBRG value (decimal)
	KBAUD	% ERROR		KBAUD	% ERROR		KBAUD	% ERROR	SPBRG value (decimal)
0.3	NA	-	-	NA	-	-	0.30	+0.16	207
1.2	1.20	+0.16	207	1.20	+0.23	185	1.20	+0.16	51
2.4	2.40	+0.16	103	2.41	+0.23	92	2.40	+0.16	25
9.6	9.62	+0.16	25	9.73	+1.32	22	8.93	-6.99	6
19.2	19.23	+0.16	12	18.64	-2.90	11	20.83	+8.51	2
76.8	NA	-	-	74.57	-2.90	2	62.50	-18.62	0
96	NA	-	-	111.86	+16.52	1	NA	-	-
300	NA	-	-	223.72	-25.43	0	NA	-	-
500	NA	-	-	NA	-	-	NA	-	-
HIGH	250	-	0	55.93	-	0	62.50	-	0
LOW	0.98	-	255	0.22	-	255	0.24	-	255

## 16.2 USART Asynchronous Mode

In this mode, the USART uses standard non-return-to-zero (NRZ) format (one START bit, eight or nine data bits and one STOP bit). The most common data format is 8-bits. An on-chip dedicated 8-bit baud rate generator can be used to derive standard baud rate frequencies from the oscillator. The USART transmits and receives the LSb first. The USART'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 bit BRGH (TXSTA<2>). Parity is not supported by the hardware, but can be implemented in software (and stored as the ninth data bit). Asynchronous mode is stopped during SLEEP.

Asynchronous mode is selected by clearing bit SYNC (TXSTA<4>).

The USART Asynchronous module consists of the following important elements:

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

### 16.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 16-1. The heart of the transmitter is the Transmit (serial) Shift Register (TSR). The shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the STOP bit has been transmitted from the previous load. As soon as the STOP bit is transmitted, the TSR is loaded with new data from the TXREG register (if available). Once the TXREG register transfers the data to the TSR register (occurs in one Tcy), the TXREG register is empty and

flag bit TXIF (PIR1<4>) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set, regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicated the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. Status bit TRMT is a read-only bit, which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

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

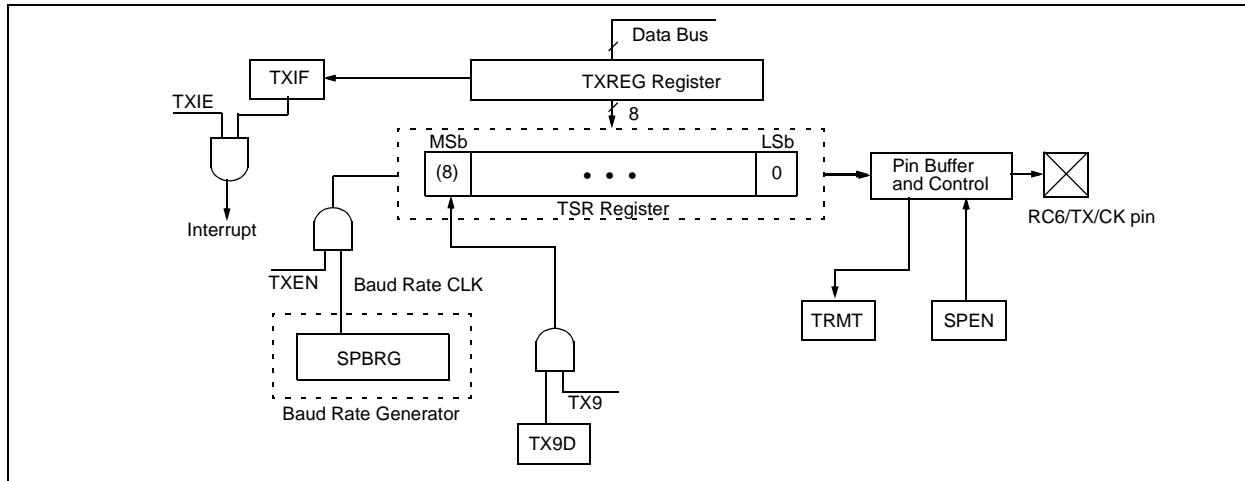
**2:** Flag bit TXIF is set when enable bit TXEN is set.

To set up an asynchronous transmission:

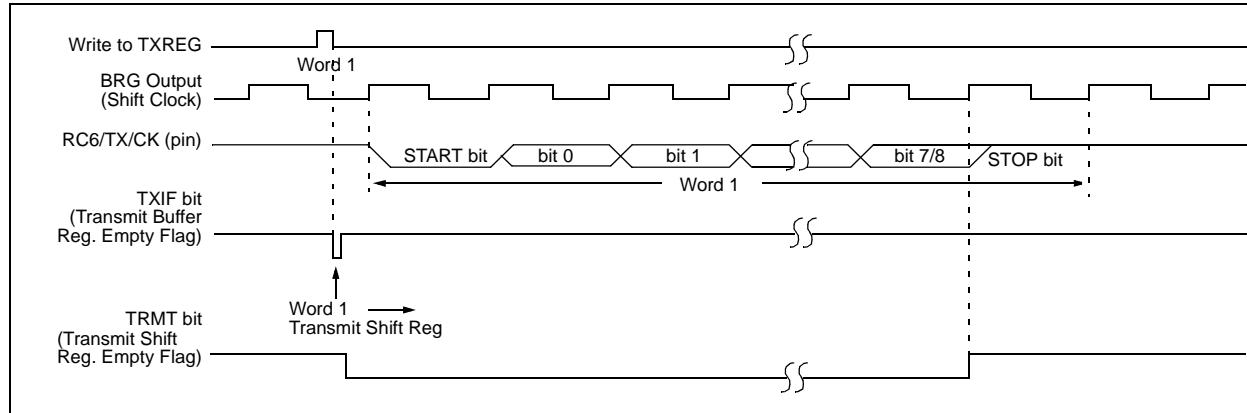
1. Initialize the SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 16.1).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, 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).

**Note:** TXIF is not cleared immediately upon loading data into the transmit buffer TXREG. The flag bit becomes valid in the second instruction cycle following the load instruction.

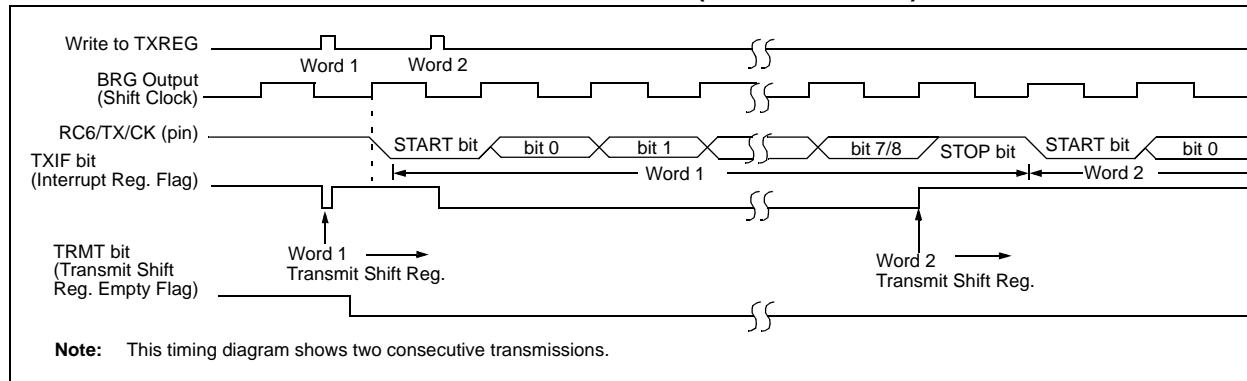
**FIGURE 16-1: USART TRANSMIT BLOCK DIAGRAM**



**FIGURE 16-2: ASYNCHRONOUS TRANSMISSION**



**FIGURE 16-3: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)**



**TABLE 16-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'.

Shaded cells are not used for Asynchronous Transmission.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

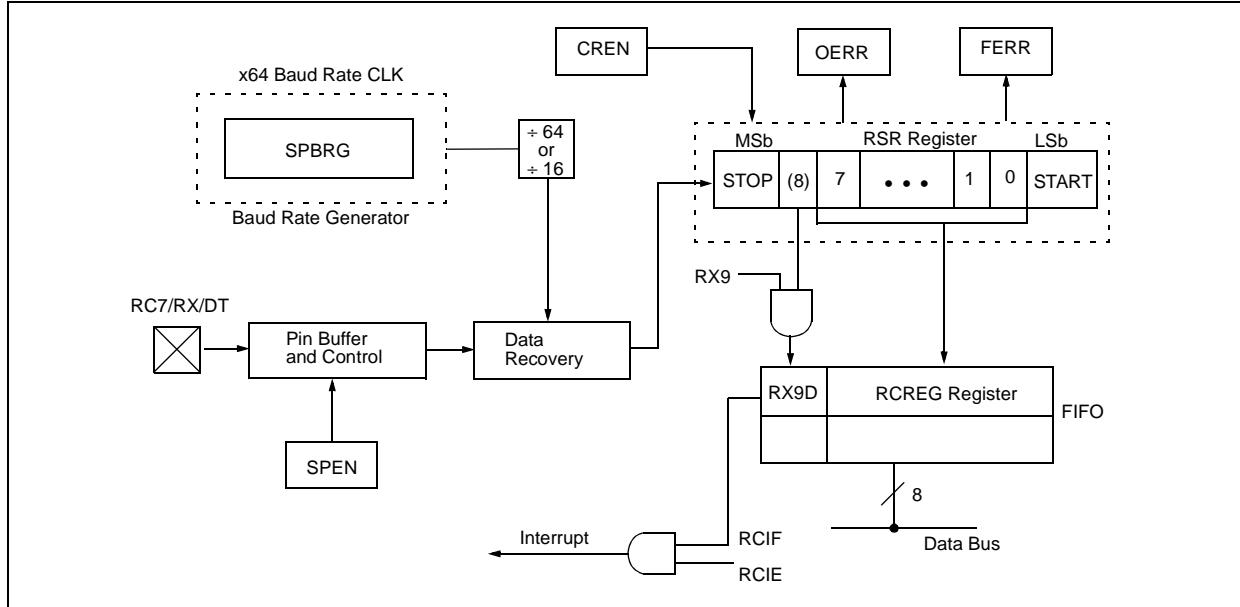
## 16.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 16-4. The data is received on the RC7/RX/DT 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 SPBRG register for the appropriate baud rate. If a high speed baud rate is desired, set bit BRGH (Section 16.1).
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 ninth 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 16-4: USART RECEIVE BLOCK DIAGRAM**



## 16.2.3 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 SPBRG register for the appropriate baud rate. If a high speed baud rate is required, set the BRGH bit.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCIF bit will be set when reception is complete. The interrupt will be acknowledged if the RCIE and GIE bits are set.
8. Read the RCSTA register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREG to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 16-5: ASYNCHRONOUS RECEPTION

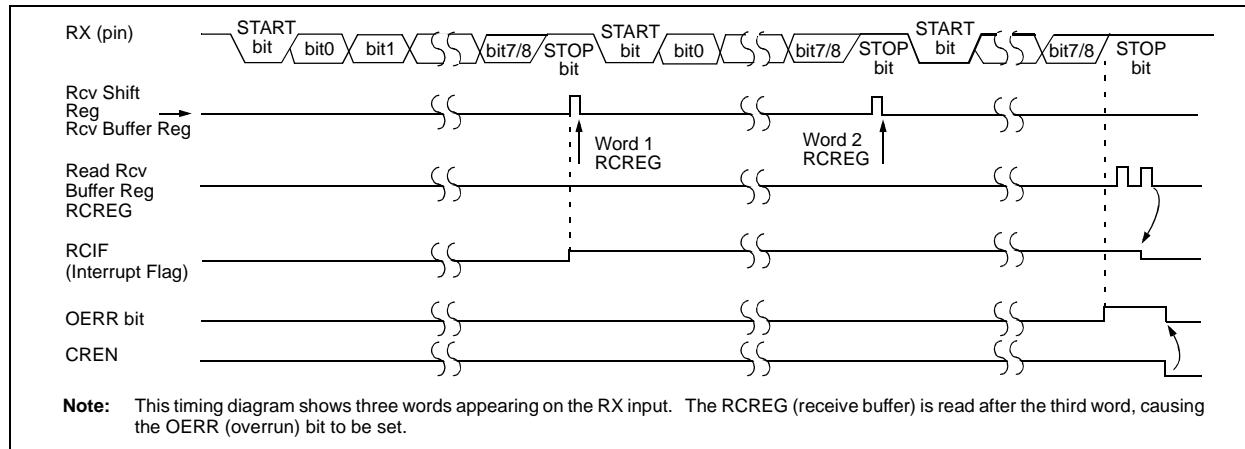


TABLE 16-7: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented locations read as '0'.

Shaded cells are not used for Asynchronous Reception.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

## 16.3 USART Synchronous Master Mode

In Synchronous Master 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 RC6/TX/CK and RC7/RX/DT I/O pins to CK (clock) and DT (data) lines, respectively. The Master mode indicates that the processor transmits the master clock on the CK line. The Master mode is entered by setting bit CSRC (TXSTA<7>).

### 16.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 16-1. The heart of the transmitter is the Transmit (serial) Shift Register (TSR). The shift register obtains its data from the read/write transmit buffer register TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available). Once the TXREG register transfers the data to the TSR register (occurs in one Tcycle), the TXREG is empty and interrupt bit TXIF (PIR1<4>) is set. The interrupt can be enabled/disabled by setting/clearing enable bit TXIE

(PIE1<4>). Flag bit TXIF will be set, regardless of the state of enable bit TXIE, and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register. While flag bit TXIF indicates the status of the TXREG register, another bit TRMT (TXSTA<1>) shows the status of the TSR register. TRMT is a read only bit, which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory, so it is not available to the user.

To set up a Synchronous Master Transmission:

1. Initialize the SPBRG register for the appropriate baud rate (Section 16.1).
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.

**Note:** TXIF is not cleared immediately upon loading data into the transmit buffer TXREG. The flag bit becomes valid in the second instruction cycle following the load instruction.

TABLE 16-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

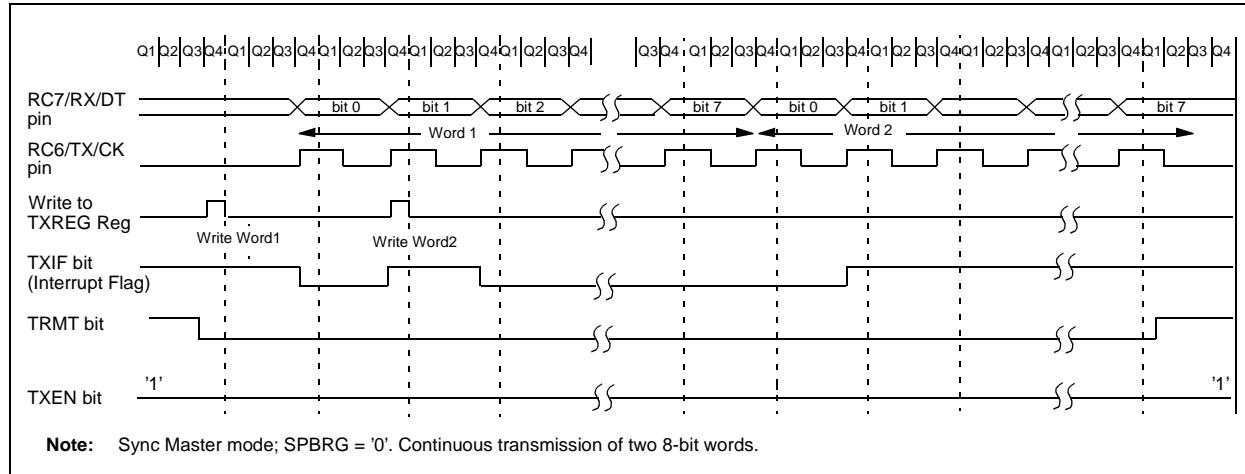
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RBIE	TMROIF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'.

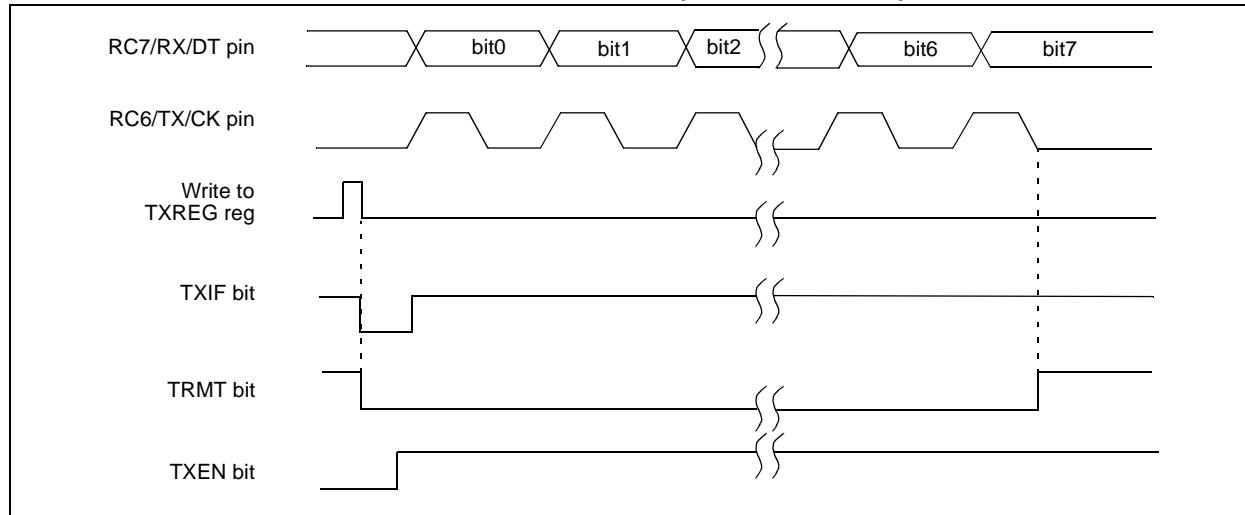
Shaded cells are not used for Synchronous Master Transmission.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

**FIGURE 16-6: SYNCHRONOUS TRANSMISSION**



**FIGURE 16-7: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**



# PIC18FXX2

## 16.3.2 USART SYNCHRONOUS MASTER RECEPTION

Once Synchronous mode is selected, reception is enabled by setting either enable bit SREN (RCSTA<5>), or enable bit CREN (RCSTA<4>). Data is sampled on the RC7/RX/DT 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 SPBRG register for the appropriate baud rate (Section 16.1).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Ensure bits CREN and SREN are clear.

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

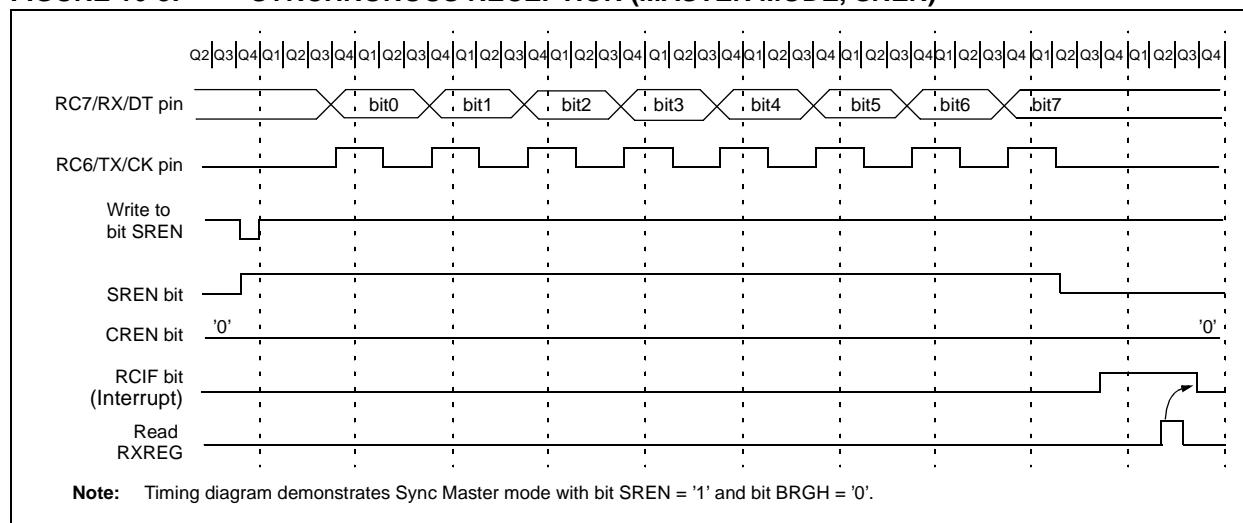
**TABLE 16-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INTOIE	RBIE	TMROIF	INTOIF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for Synchronous Master Reception.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

**FIGURE 16-8: SYNCHRONOUS RECESSION (MASTER MODE, SREN)**



## 16.4 USART Synchronous Slave Mode

Synchronous Slave mode differs from the Master mode in the fact that the shift clock is supplied externally at the RC6/TX/CK pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in SLEEP mode. Slave mode is entered by clearing bit CSRC (TXSTA<7>).

### 16.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes are identical, except in the case of the SLEEP mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

- The first word will immediately transfer to the TSR register and transmit.
- The second word will remain in 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 TXREG register.
- If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 16-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other Resets
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'.

Shaded cells are not used for Synchronous Slave Transmission.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

## 16.4.2 USART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of the SLEEP mode and bit SREN, which is a "don't care" in Slave mode.

If receive is enabled by setting bit CREN prior to the SLEEP instruction, then a word may be received during SLEEP. On completely receiving the word, the RSR register will transfer the data to the RCREG register, and if enable bit RCIE bit is set, the interrupt generated 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 Reception:

1. Enable the synchronous master serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. If interrupts are desired, set enable bit RCIE.
3. If 9-bit reception is desired, set bit RX9.
4. To enable reception, set enable bit CREN.
5. Flag bit RCIF will be set when reception is complete. An interrupt will be generated if enable bit RCIE was set.
6. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 16-11: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'.

Shaded cells are not used for Synchronous Slave Reception.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

## 17.0 COMPATIBLE 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) converter module has five inputs for the PIC18F2X2 devices and eight for the PIC18F4X2 devices. This module has the ADCON0 and ADCON1 register definitions that are compatible with the mid-range A/D module.

The A/D allows conversion of an analog input signal to a corresponding 10-bit digital number.

The A/D module has four registers. These registers are:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)

The ADCON0 register, shown in Register 17-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 17-2, configures the functions of the port pins.

### REGISTER 17-1: ADCON0 REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
bit 7						bit 0	

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits (ADCON0 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-3 **CHS2:CHS0:** Analog Channel Select bits

- 000 = channel 0, (AN0)
- 001 = channel 1, (AN1)
- 010 = channel 2, (AN2)
- 011 = channel 3, (AN3)
- 100 = channel 4, (AN4)
- 101 = channel 5, (AN5)
- 110 = channel 6, (AN6)
- 111 = channel 7, (AN7)

**Note:** The PIC18F2X2 devices do not implement the full 8 A/D channels; the unimplemented selections are reserved. Do not select any unimplemented channel.

bit 2 **GO/DONE:** A/D Conversion Status bit

When ADON = 1:

- 1 = A/D conversion in progress (setting this bit starts the A/D conversion which is automatically cleared by hardware when the A/D conversion is complete)
- 0 = A/D conversion not in progress

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

bit 0 **ADON:** A/D On bit

- 1 = A/D converter module is powered up

- 0 = A/D converter module is shut-off and consumes no operating current

#### 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

# PIC18FXX2

## REGISTER 17-2: ADCON1 REGISTER

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7   **ADFM:** A/D Result Format Select bit  
 1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.  
 0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 6   **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-4   **Unimplemented:** Read as '0'

bit 3-0   **PCFG3:PCFG0:** A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	Vss	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	Vss	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	Vss	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	Vss	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	Vss	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	Vss	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	Vss	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	Vss	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	Vss	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input    D = Digital I/O

C/R = # of analog input channels / # of A/D voltage references

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

**Note:** On any device RESET, the port pins that are multiplexed with analog functions (ANx) are forced to be an analog input.

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+ pin and RA2/AN2/VREF- pin.

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.

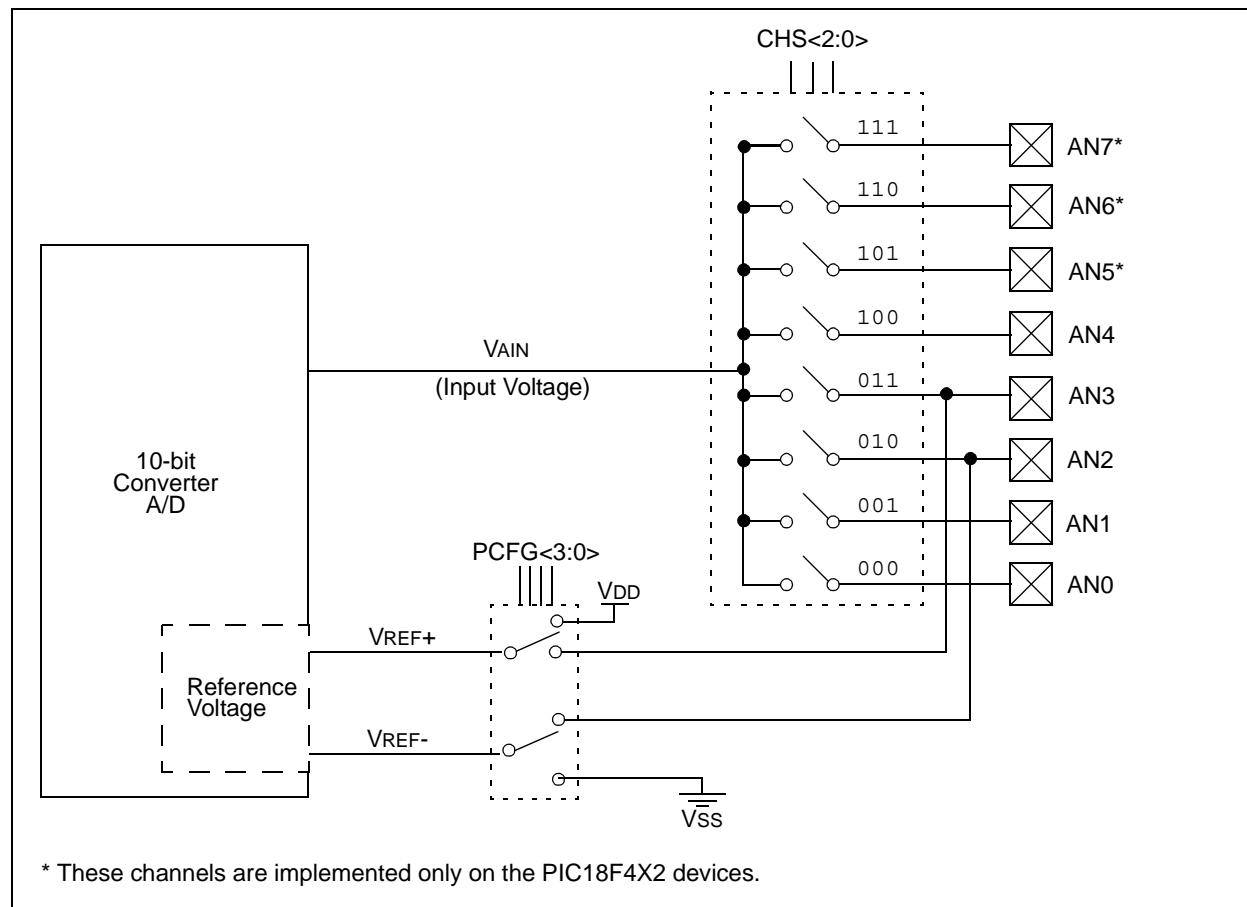
The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

A device RESET forces all registers to their RESET state. This forces the A/D module to be turned off and any conversion is aborted.

Each port pin associated with the A/D converter can be configured as an analog input (RA3 can also be a voltage reference) or as a digital I/O.

The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH/ADRESL registers, the GO/DONE bit (ADCON0<2>) is cleared, and A/D interrupt flag bit, ADIF is set. The block diagram of the A/D module is shown in Figure 17-1.

**FIGURE 17-1: A/D BLOCK DIAGRAM**



The value that is in the ADRESH/ADRESL registers is not modified for a Power-on Reset. The ADRESH/ADRESL registers will contain unknown data after a Power-on Reset.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see Section 17.1. After this acquisition time has elapsed, the A/D conversion can be started. The following steps should be followed for doing 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 conversion clock (ADCON0)
    - Turn on A/D module (ADCON0)
  2. Configure A/D interrupt (if desired):
    - Clear ADIF bit
    - Set ADIE bit
    - Set GIE bit
    - Set PEIE bit
  3. Wait the required acquisition time.
  4. Start conversion:
    - Set GO/DONE bit (ADCON0)

- Wait for A/D conversion to complete, by either:
  - Polling for the GO/ $\overline{\text{DONE}}$  bit to be cleared (interrupts disabled)

OR

- Waiting for the A/D interrupt

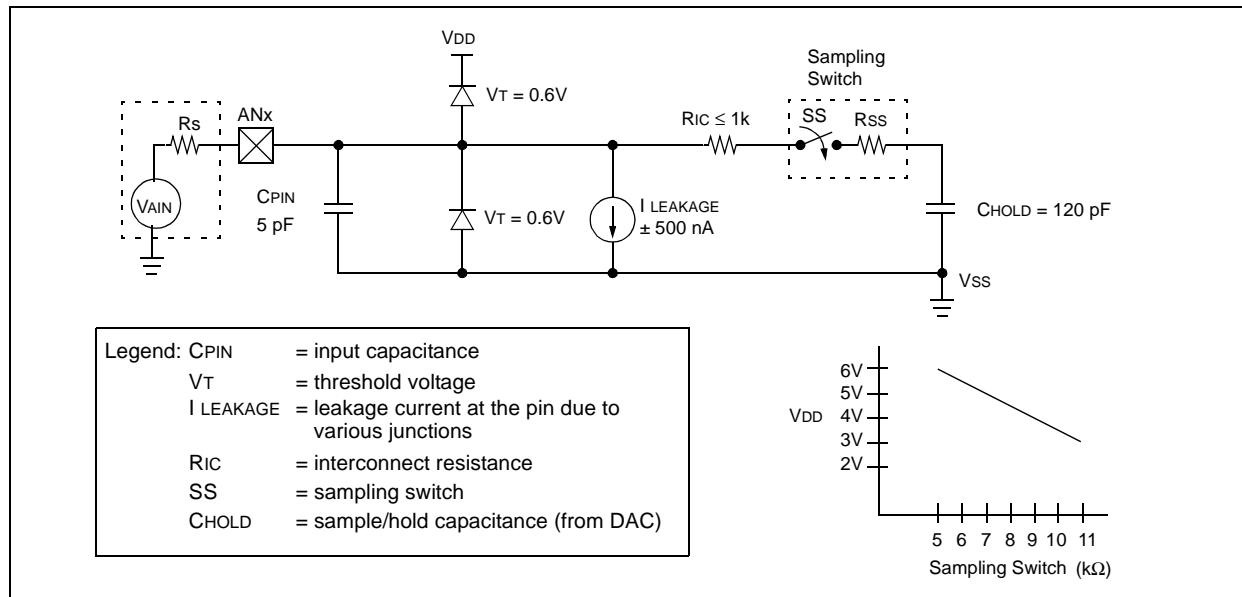
6. Read A/D Result registers (ADRESH/ADRESL); clear bit ADIF if required.
  7. For next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2 TAD is required before the next acquisition starts.

## 17.1 A/D Acquisition Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor ( $C_{HOLD}$ ) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 17-2. The source impedance ( $R_S$ ) and the internal sampling switch ( $R_{SS}$ ) impedance directly affect the time required to charge the capacitor  $C_{HOLD}$ . The sampling switch ( $R_{SS}$ ) impedance varies over the device voltage ( $V_{DD}$ ). The source impedance affects the offset voltage at the analog input (due to pin leakage current). **The maximum recommended impedance for analog sources is  $2.5\text{ k}\Omega$ .** After the analog input channel is selected (changed), this acquisition must be done before the conversion can be started.

**Note:** When the conversion is started, the holding capacitor is disconnected from the input pin.

**FIGURE 17-2:** ANALOG INPUT MODEL



To calculate the minimum acquisition time, Equation 17-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

## EQUATION 17-1: ACQUISITION TIME

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \end{aligned}$$

## EQUATION 17-2: A/D MINIMUM CHARGING TIME

$$\begin{aligned} V_{HOLD} &= (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))}) \\ \text{or} \\ T_C &= -(120 \text{ pF})(1 \text{ k}\Omega + R_{SS} + R_S) \ln(1/2048) \end{aligned}$$

Example 17-1 shows the calculation of the minimum required acquisition time,  $T_{ACQ}$ . This calculation is based on the following application system assumptions:

- $C_{HOLD}$  = 120 pF
- $R_s$  = 2.5 k $\Omega$
- Conversion Error  $\leq$  1/2 LSb
- $V_{DD}$  = 5V  $\rightarrow R_{SS} = 7 \text{ k}\Omega$
- Temperature = 50°C (system max.)
- $V_{HOLD}$  = 0V @ time = 0

## EXAMPLE 17-1: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$$\begin{aligned} T_{ACQ} &= T_{AMP} + T_C + T_{COFF} \\ \text{Temperature coefficient is only required for temperatures } > 25^\circ\text{C.} \\ T_{ACQ} &= 2 \mu\text{s} + T_C + [(Temp - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})] \\ T_C &= -C_{HOLD} (R_{IC} + R_{SS} + R_S) \ln(1/2048) \\ &= -120 \text{ pF} (1 \text{ k}\Omega + 7 \text{ k}\Omega + 2.5 \text{ k}\Omega) \ln(0.0004883) \\ &= -120 \text{ pF} (10.5 \text{ k}\Omega) \ln(0.0004883) \\ &= -1.26 \mu\text{s} (-7.6246) \\ &= 9.61 \mu\text{s} \\ T_{ACQ} &= 2 \mu\text{s} + 9.61 \mu\text{s} + [(50^\circ\text{C} - 25^\circ\text{C})(0.05 \mu\text{s}/^\circ\text{C})] \\ &= 11.61 \mu\text{s} + 1.25 \mu\text{s} \\ &= 12.86 \mu\text{s} \end{aligned}$$

## 17.2 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 12 TAD per 10-bit conversion. The source of the A/D conversion clock is software selectable. The seven possible options for TAD are:

- 2 Tosc
- 4 Tosc
- 8 Tosc
- 16 Tosc
- 32 Tosc
- 64 Tosc
- Internal A/D module RC oscillator (2-6  $\mu$ s)

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 1.6  $\mu$ s.

Table 17-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

## 17.3 Configuring Analog Port Pins

The ADCON1, TRISA and TRISE registers control the operation of the A/D port pins. The port pins that are desired as analog inputs, must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS2:CHS0 bits and the TRIS bits.

- Note 1:** When reading the port register, all pins configured as analog input channels will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.
- 2:** Analog levels on any pin that is defined as a digital input (including the AN4:AN0 pins) may cause the input buffer to consume current that is out of the device's specification.

**TABLE 17-1: TAD vs. DEVICE OPERATING FREQUENCIES**

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	ADCS2:ADCS0	PIC18FXX2	PIC18LFXX2
2 Tosc	000	1.25 MHz	666 kHz
4 Tosc	100	2.50 MHz	1.33 MHz
8 Tosc	001	5.00 MHz	2.67 MHz
16 Tosc	101	10.00 MHz	5.33 MHz
32 Tosc	010	20.00 MHz	10.67 MHz
64 Tosc	110	40.00 MHz	21.33 MHz
RC	011	—	—

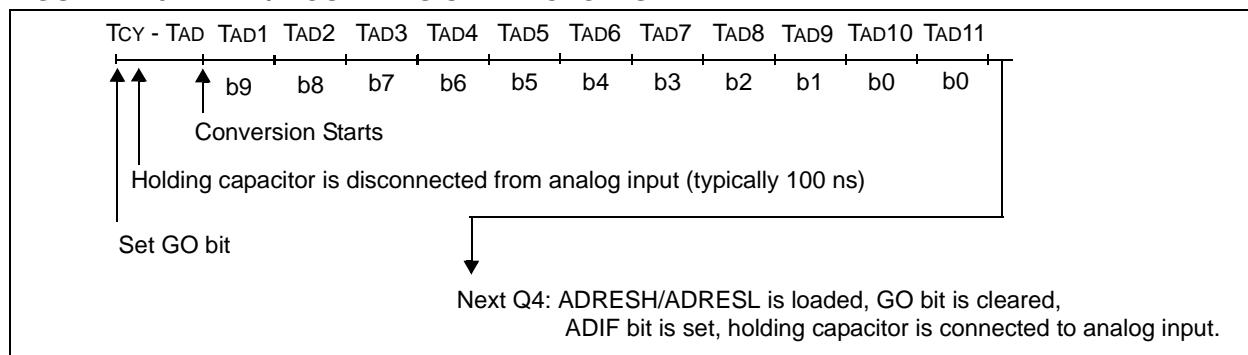
## 17.4 A/D Conversions

Figure 17-3 shows the operation of the A/D converter after the GO bit has been set. 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. That is, 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 aborted, a 2 TAD wait is required before the next acquisition is started. After this 2 TAD wait, acquisition on the selected channel is automatically started. The GO/DONE bit can then be set to start the conversion.

**Note:** The GO/DONE bit should **NOT** be set in the same instruction that turns on the A/D.

**FIGURE 17-3: A/D CONVERSION TAD CYCLES**

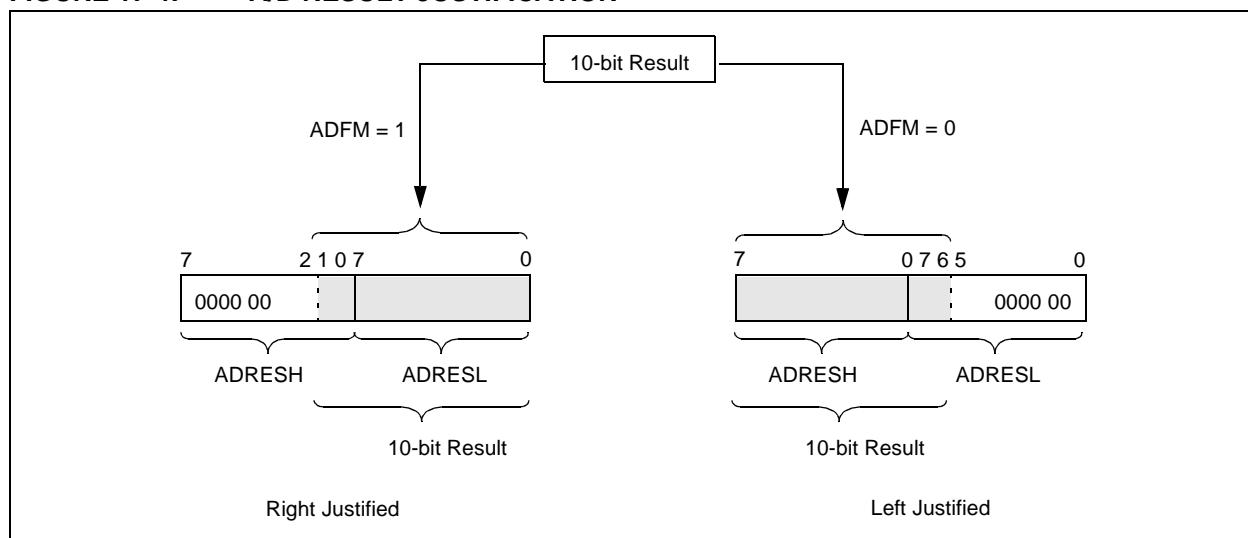


### 17.4.1 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is the location where the 10-bit A/D result is loaded at the completion of the A/D conversion. This register pair is 16-bits wide. The A/D module gives the flexibility to left or right justify the 10-bit result in the 16-bit result register. The A/D

Format Select bit (ADFM) controls this justification. Figure 17-4 shows the operation of the A/D result justification. The extra bits are loaded with '0's. When an A/D result will not overwrite these locations (A/D disable), these registers may be used as two general purpose 8-bit registers.

**FIGURE 17-4: A/D RESULT JUSTIFICATION**



## 17.5 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 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 done before the “special event trigger” sets the GO/DONE bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the “special event trigger” will be ignored by the A/D module, but will still reset the Timer1 (or Timer3) counter.

**TABLE 17-2: SUMMARY OF A/D REGISTERS**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF <sup>(1)</sup>	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE <sup>(1)</sup>	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP <sup>(1)</sup>	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 0000
ADRESH	A/D Result Register								xxxx xxxx	uuuu uuuu
ADRESL	A/D Result Register								xxxx xxxx	uuuu uuuu
ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON	0000 00-0	0000 00-0
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	---- -000	---- -000
PORTA	—	RA6	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
TRISA	—	PORTA Data Direction Register							--11 1111	--11 1111
PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -000	---- -000
LATE	—	—	—	—	—	LATE2	LATE1	LATE0	---- -xxx	---- -uuu
TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

**Note 1:** The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18F2X2 devices; always maintain these bits clear.

## 18.0 LOW VOLTAGE DETECT

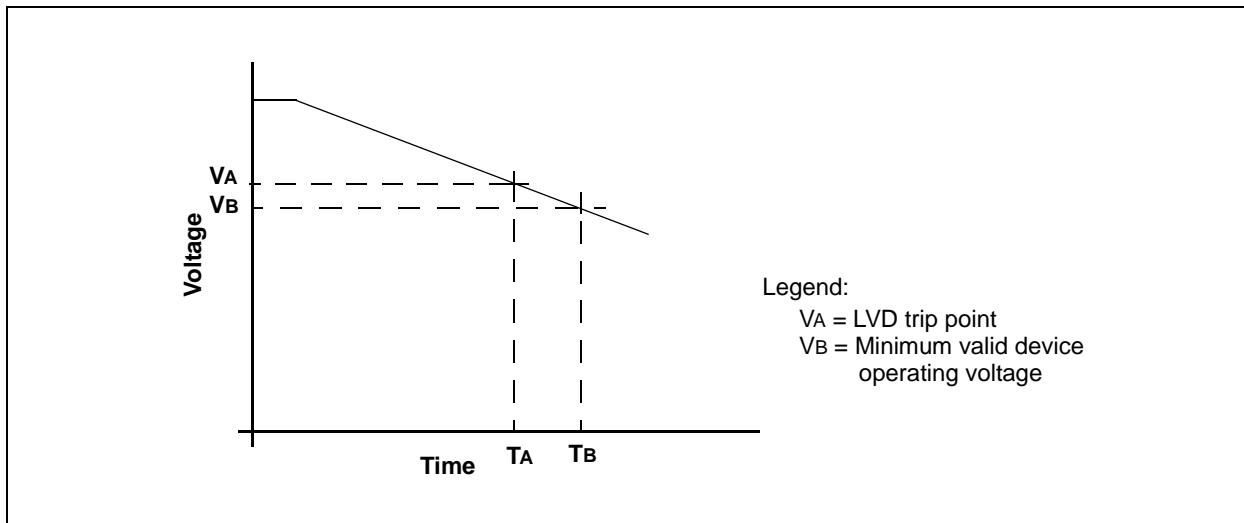
In many applications, the ability to determine if the device voltage ( $V_{DD}$ ) is below a specified voltage level is a desirable feature. A window of operation for the application can be created, where the application software can do “housekeeping tasks” before the device voltage exits the valid operating range. This can be done using the Low Voltage Detect module.

This module is a software programmable circuitry, where a device voltage trip point can be specified. When the voltage of the device becomes lower than the specified point, an interrupt flag is set. If the interrupt is enabled, the program execution will branch to the interrupt vector address and the software can then respond to that interrupt source.

The Low Voltage Detect circuitry is completely under software control. This allows the circuitry to be “turned off” by the software, which minimizes the current consumption for the device.

Figure 18-1 shows a possible application voltage curve (typically for batteries). Over time, the device voltage decreases. When the device voltage equals voltage  $V_A$ , the LVD logic generates an interrupt. This occurs at time  $T_A$ . The application software then has the time, until the device voltage is no longer in valid operating range, to shutdown the system. Voltage point  $V_B$  is the minimum valid operating voltage specification. This occurs at time  $T_B$ . The difference  $T_B - T_A$  is the total time for shutdown.

**FIGURE 18-1: TYPICAL LOW VOLTAGE DETECT APPLICATION**



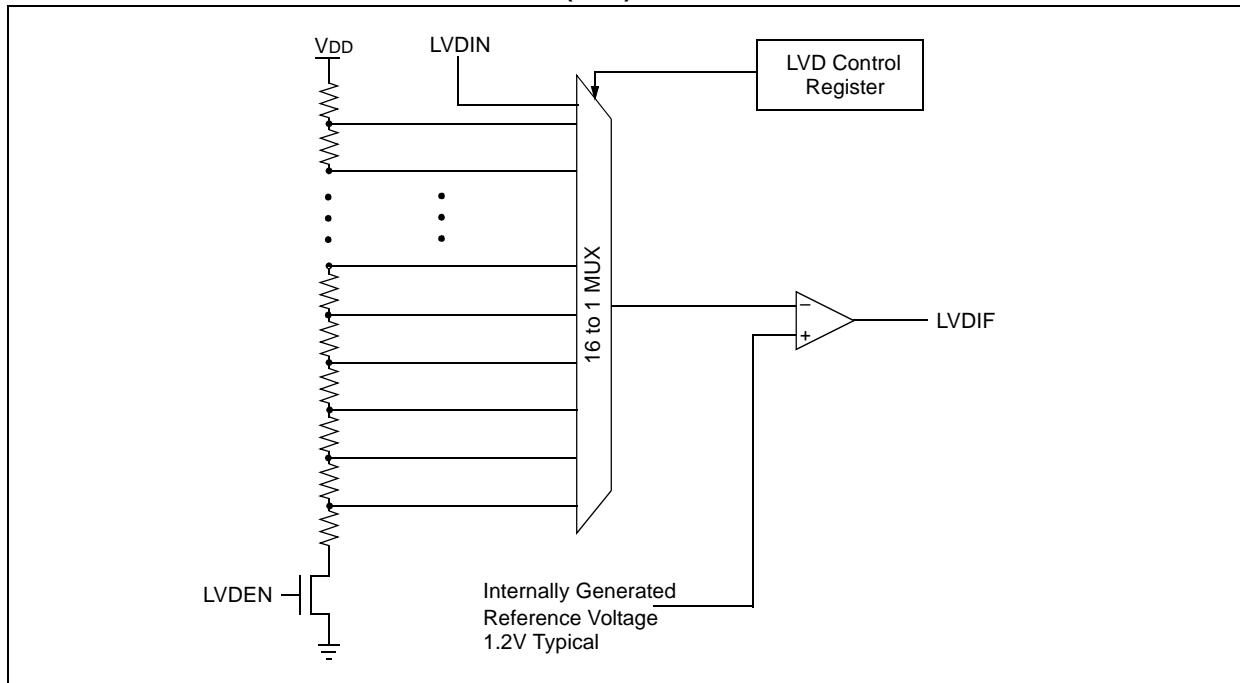
The block diagram for the LVD module is shown in Figure 18-2. A comparator uses an internally generated reference voltage as the set point. When the selected tap output of the device voltage crosses the set point (is lower than), the LVDIF bit is set.

Each node in the resistor divider represents a “trip point” voltage. The “trip point” voltage is the minimum supply voltage level at which the device can operate before the LVD module asserts an interrupt. When the

supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the 1.2V internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal setting the LVDIF bit. This voltage is software programmable to any one of 16 values (see Figure 18-2). The trip point is selected by programming the LVDL3:LVDL0 bits (LVDCON<3:0>).

# PIC18FXX2

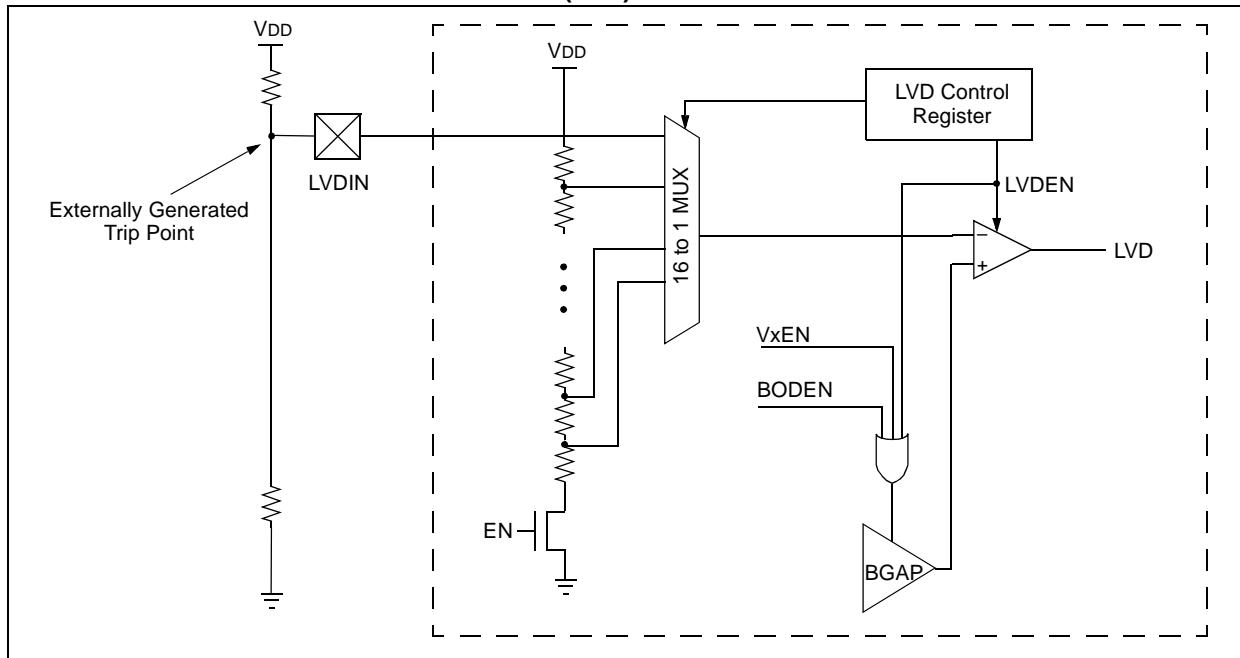
**FIGURE 18-2: LOW VOLTAGE DETECT (LVD) BLOCK DIAGRAM**



The LVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits **LVDL3:LVDL0** are set to **1111**. In this state, the comparator input is multiplexed from the external input pin,

**LVDIN** (Figure 18-3). This gives users flexibility, because it allows them to configure the Low Voltage Detect interrupt to occur at any voltage in the valid operating range.

**FIGURE 18-3: LOW VOLTAGE DETECT (LVD) WITH EXTERNAL INPUT BLOCK DIAGRAM**



## 18.1 Control Register

The Low Voltage Detect Control register controls the operation of the Low Voltage Detect circuitry.

## REGISTER 18-1: LVDCON REGISTER

U-0	U-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0
bit 7				bit 0			

- |         |  |
|---------|--|
| bit 7-6 | <b>Unimplemented:</b> Read as '0'  |
| bit 5   | <b>IRVST:</b> Internal Reference Voltage Stable Flag bit<br>1 = Indicates that the Low Voltage Detect logic will generate the interrupt flag at the specified voltage range<br>0 = Indicates that the Low Voltage Detect logic will not generate the interrupt flag at the specified voltage range and the LVD interrupt should not be enabled   |
| bit 4   | <b>LVDEN:</b> Low Voltage Detect Power Enable bit<br>1 = Enables LVD, powers up LVD circuit<br>0 = Disables LVD, powers down LVD circuit   |
| bit 3-0 | <b>LVDL3:LVDL0:</b> Low Voltage Detection Limit bits<br>1111 = External analog input is used (input comes from the LVDIN pin)<br>1110 = 4.5V - 4.77V<br>1101 = 4.2V - 4.45V<br>1100 = 4.0V - 4.24V<br>1011 = 3.8V - 4.03V<br>1010 = 3.6V - 3.82V<br>1001 = 3.5V - 3.71V<br>1000 = 3.3V - 3.50V<br>0111 = 3.0V - 3.18V<br>0110 = 2.8V - 2.97V<br>0101 = 2.7V - 2.86V<br>0100 = 2.5V - 2.65V<br>0011 = 2.4V - 2.54V<br>0010 = 2.2V - 2.33V<br>0001 = 2.0V - 2.12V<br>0000 = Reserved |

**Note:** LVDL3:LVDL0 modes which result in a trip point below the valid operating voltage of the device are not tested.

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

## 18.2 Operation

Depending on the power source for the device voltage, the voltage normally decreases relatively slowly. This means that the LVD module does not need to be constantly operating. To decrease the current requirements, the LVD circuitry only needs to be enabled for short periods, where the voltage is checked. After doing the check, the LVD module may be disabled.

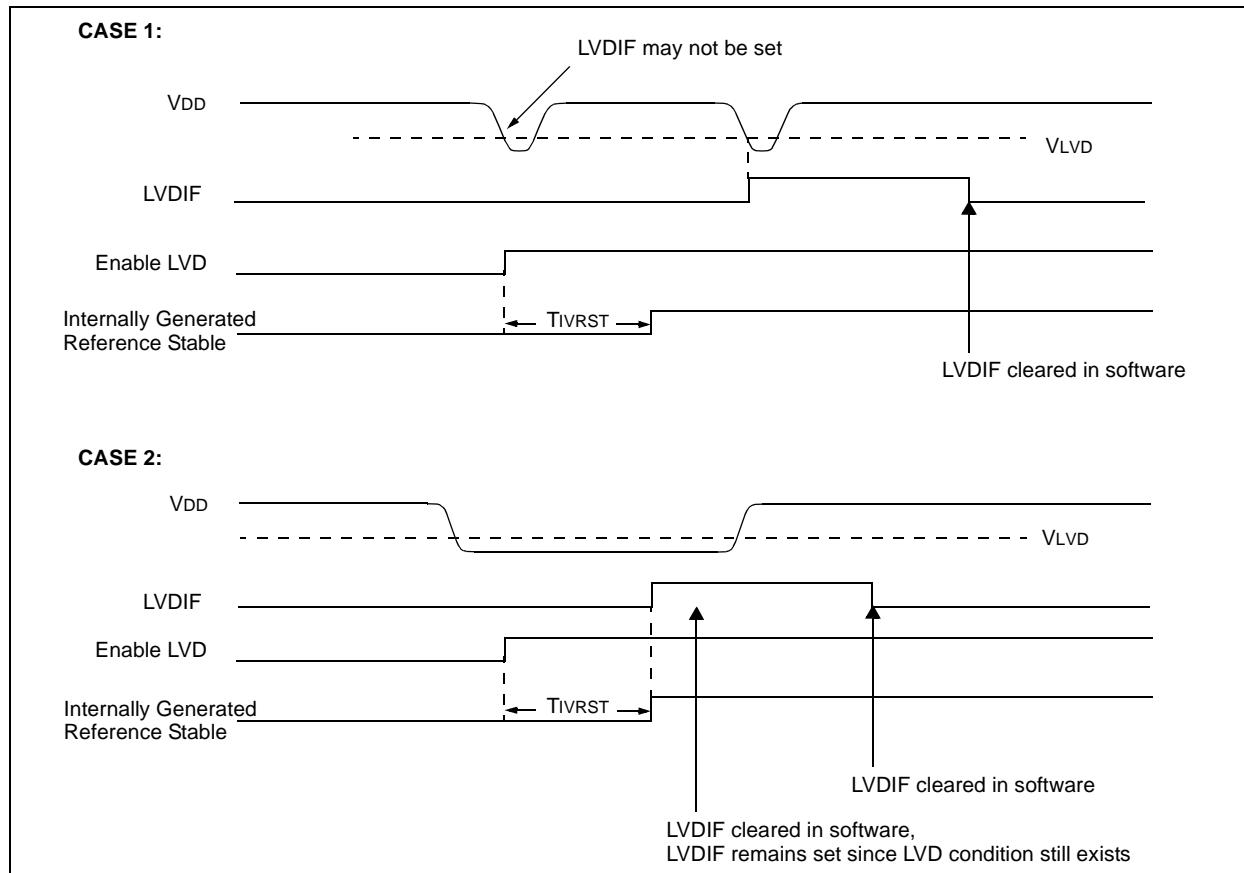
Each time that the LVD module is enabled, the circuitry requires some time to stabilize. After the circuitry has stabilized, all status flags may be cleared. The module will then indicate the proper state of the system.

The following steps are needed to set up the LVD module:

1. Write the value to the LVLD3:LVLD0 bits (LVDCON register), which selects the desired LVD Trip Point.
2. Ensure that LVD interrupts are disabled (the LVDIE bit is cleared or the GIE bit is cleared).
3. Enable the LVD module (set the LVDEN bit in the LVDCON register).
4. Wait for the LVD module to stabilize (the IRVST bit to become set).
5. Clear the LVD interrupt flag, which may have falsely become set until the LVD module has stabilized (clear the LVDIF bit).
6. Enable the LVD interrupt (set the LVDIE and the GIE bits).

Figure 18-4 shows typical waveforms that the LVD module may be used to detect.

**FIGURE 18-4: LOW VOLTAGE DETECT WAVEFORMS**



### 18.2.1 REFERENCE VOLTAGE SET POINT

The Internal Reference Voltage of the LVD module may be used by other internal circuitry (the Programmable Brown-out Reset). If these circuits are disabled (lower current consumption), the reference voltage circuit requires a time to become stable before a low voltage condition can be reliably detected. This time is invariant of system clock speed. This start-up time is specified in electrical specification parameter 36. The low voltage interrupt flag will not be enabled until a stable reference voltage is reached. Refer to the waveform in Figure 18-4.

### 18.2.2 CURRENT CONSUMPTION

When the module is enabled, the LVD comparator and voltage divider are enabled and will consume static current. The voltage divider can be tapped from multiple places in the resistor array. Total current consumption, when enabled, is specified in electrical specification parameter #D022B.

### 18.3 Operation During SLEEP

When enabled, the LVD circuitry continues to operate during SLEEP. If the device voltage crosses the trip point, the LVDIF bit will be set and the device will wake-up from SLEEP. Device execution will continue from the interrupt vector address if interrupts have been globally enabled.

### 18.4 Effects of a RESET

A device RESET forces all registers to their RESET state. This forces the LVD module to be turned off.

# **PIC18FXX2**

---

---

## **NOTES:**

## 19.0 SPECIAL FEATURES OF THE CPU

There are several features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving Operating modes and offer code protection. These are:

- OSC Selection
- RESET
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- SLEEP
- Code Protection
- ID Locations
- In-Circuit Serial Programming

All PIC18FXX2 devices have a Watchdog Timer, which is permanently enabled via the configuration bits or software controlled. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in RESET until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a fixed delay on power-up only, designed to keep the part in RESET while the power supply stabilizes. With these two timers on-chip, most applications need no external RESET circuitry.

SLEEP mode is designed to offer a very low current Power-down mode. The user can wake-up from SLEEP through external RESET, Watchdog Timer Wake-up or through an interrupt. Several oscillator options are also made available to allow the part to fit the application. The RC oscillator option saves system cost, while the LP crystal option saves power. A set of configuration bits are used to select various options.

## 19.1 Configuration Bits

The configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1'), to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h - 3FFFFFh), which can only be accessed using Table Reads and Table Writes.

Programming the configuration registers is done in a manner similar to programming the FLASH memory (see Section 5.5.1). The only difference is the configuration registers are written a byte at a time. The sequence of events for programming configuration registers is:

1. Load table pointer with address of configuration register being written.
2. Write a single byte using the TBLWT instruction.
3. Set EEPGD to point to program memory, set the CFGS bit to access configuration registers, and set WREN to enable byte writes.
4. Disable interrupts.
5. Write 55h to EECON2.
6. Write AAh to EECON2.
7. Set the WR bit. This will begin the write cycle.
8. CPU will stall for duration of write (approximately 2 ms using internal timer).
9. Execute a NOP.
10. Re-enable interrupts.



**REGISTER 19-2: CONFIGURATION REGISTER 2 LOW (CONFIG2L: BYTE ADDRESS 300002h)**

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	BORV1	BORV0	BOREN	PWRLEN
bit 7				bit 0			

- |         |  |
|---------|--|
| bit 7-4 | <b>Unimplemented:</b> Read as '0'                |
| bit 3-2 | <b>BORV1:BORV0:</b> Brown-out Reset Voltage bits |
|         | 11 = VBOR set to 2.5V                            |
|         | 10 = VBOR set to 2.7V                            |
|         | 01 = VBOR set to 4.2V                            |
|         | 00 = VBOR set to 4.5V                            |
| bit 1   | <b>BOREN:</b> Brown-out Reset Enable bit         |
|         | 1 = Brown-out Reset enabled                      |
|         | 0 = Brown-out Reset disabled                     |
| bit 0   | <b>PWR滕:</b> Power-up Timer Enable bit           |
|         | 1 = PWRT disabled                                |
|         | 0 = PWRT enabled                                 |

### Legend:

R = Readable bit                  P = Programmable bit                  U = Unimplemented bit, read as '0'  
- n = Value when device is unprogrammed                  u = Unchanged from programmed state

**REGISTER 19-3: CONFIGURATION REGISTER 2 HIGH (CONFIG2H: BYTE ADDRESS 300003h)**

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN
bit 7				bit 0			

- |         |  |
|---------|--|
| bit 7-4 | <b>Unimplemented:</b> Read as '0'                          |
| bit 3-1 | <b>WDTPS2:WDTPS0:</b> Watchdog Timer Postscale Select bits |
|         | 111 = 1:128  |
|         | 110 = 1:64   |
|         | 101 = 1:32   |
|         | 100 = 1:16   |
|         | 011 = 1:8  |
|         | 010 = 1:4  |
|         | 001 = 1:2  |
|         | 000 = 1:1  |
| bit 0   | <b>WDTEN:</b> Watchdog Timer Enable bit                    |
|         | 1 = WDT enabled  |
|         | 0 = WDT disabled (control is placed on the SWDTEN bit)     |

### Legend:

R = Readable bit      P = Programmable bit      U = Unimplemented bit, read as '0'  
- n = Value when device is unprogrammed      u = Unchanged from programmed state

# **PIC18FXX2**

**REGISTER 19-4: CONFIGURATION REGISTER 3 HIGH (CONFIG3H: BYTE ADDRESS 300005h)**

bit 7-1      **Unimplemented:** Read as '0'

bit 0      **CCP2MX**: CCP2 Mux bit

1 = CCP2 input/output is multiplexed with RC1  
0 = CCP2 input/output is multiplexed with RB3

---

**Legend:**

R = Readable bit      P = Programmable bit      U = Unimplemented bit, read as '0'  
- n = Value when device is unprogrammed      u = Unchanged from programmed state

**REGISTER 19-5: CONFIGURATION REGISTER 4 LOW (CONFIG4L: BYTE ADDRESS 300006h)**

R/P-1	U-0	U-0	U-0	U-0	R/P-1	U-0	R/P-1
BKBUG	—	—	—	—	LVP	—	STVREN
bit 7				bit 0			

**bit 7      DEBUG:** Background Debugger Enable bit

**1** = Background Debugger disabled. RB6 and RB7 configured as general purpose I/O pins.  
**0** = Background Debugger enabled. RB6 and RB7 are dedicated to In-Circuit Debug.

bit 6-3      **Unimplemented:** Read as '0'

bit 2      **LVP**: Low Voltage ICSP Enable bit

1 = Low Voltage ICSP enabled  
0 = Low Voltage ICSP disabled

bit 1      Unimplemented; Read as '0'

bit 0 **STVREN**: Stack Full/I Underflow Reset Enable bit

**STVREN:** Stack Full/Underflow Reset Enable S

### Legend:

R = Readable bit      C = Clearable bit      U = Unimplemented bit, read as '0'  
 - n = Value when device is unprogrammed      U = Unchanged from programmed state

## REGISTER 19-6: CONFIGURATION REGISTER 5 LOW (CONFIG5L: 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 <sup>(1)</sup>	CP1	CP0

bit 7 bit 0

- 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<sup>(1)</sup>  
     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 (000200-001FFFh) not code protected  
     0 = Block 0 (000200-001FFFh) code protected

**Note 1:** Unimplemented in PIC18FX42 devices; maintain this bit set.

## Legend:

R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

## REGISTER 19-7: CONFIGURATION REGISTER 5 HIGH (CONFIG5H: BYTE ADDRESS 300009h)

R/C-1	R/C-1	U-0						
CPD	CPB	—	—	—	—	—	—	—

bit 7 bit 0

- bit 7     **CPD:** Data EEPROM Code Protection bit  
     1 = Data EEPROM not code protected  
     0 = Data EEPROM code protected
- bit 6     **CPB:** Boot Block Code Protection bit  
     1 = Boot Block (000000-0001FFh) not code protected  
     0 = Boot Block (000000-0001FFh) code protected
- bit 5-0   **Unimplemented:** Read as '0'

## Legend:

R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

# PIC18FXX2

## REGISTER 19-8: CONFIGURATION REGISTER 6 LOW (CONFIG6L: 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 <sup>(1)</sup>	WRT1	WRT0

bit 7

bit 0

bit 7-4 **Unimplemented:** Read as '0'bit 3 **WRT3:** Write Protection bit<sup>(1)</sup>

1 = Block 3 (006000-007FFFh) not write protected

0 = Block 3 (006000-007FFFh) write protected

bit 2 **WRT2:** Write Protection bit<sup>(1)</sup>

1 = Block 2 (004000-005FFFh) not write protected

0 = Block 2 (004000-005FFFh) write protected

bit 1 **WRT1:** Write Protection bit

1 = Block 1 (002000-003FFFh) not write protected

0 = Block 1 (002000-003FFFh) write protected

bit 0 **WRT0:** Write Protection bit

1 = Block 0 (000200h-001FFFh) not write protected

0 = Block 0 (000200h-001FFFh) write protected

**Note 1:** Unimplemented in PIC18FX42 devices; maintain this bit set.

Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

- n = Value when device is unprogrammed

u = Unchanged from programmed state

## REGISTER 19-9: CONFIGURATION REGISTER 6 HIGH (CONFIG6H: BYTE ADDRESS 30000Bh)

R/C-1	R/C-1	C-1	U-0	U-0	U-0	U-0	U-0
WRTD	WRTB	WRTC	—	—	—	—	—

bit 7

bit 0

bit 7 **WRTD:** Data EEPROM Write Protection bit

1 = Data EEPROM not write protected

0 = Data EEPROM write protected

bit 6 **WRTB:** Boot Block Write Protection bit

1 = Boot Block (000000-0001FFh) not write protected

0 = Boot Block (000000-0001FFh) write protected

bit 5 **WRTC:** Configuration Register Write Protection bit

1 = Configuration registers (300000-3000FFh) not write protected

0 = Configuration registers (300000-3000FFh) write protected

**Note:** This bit is read only, and cannot be changed in User mode.

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

## REGISTER 19-10: CONFIGURATION REGISTER 7 LOW (CONFIG7L: 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 <sup>(1)</sup>	EBTR1	EBTR0

bit 7

bit 0

- 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<sup>(1)</sup>  
 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 (000200h-001FFFh) not protected from Table Reads executed in other blocks  
 0 = Block 0 (000200h-001FFFh) protected from Table Reads executed in other blocks

**Note 1:** Unimplemented in PIC18FX42 devices; maintain this bit set.

Legend:

R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

## REGISTER 19-11: CONFIGURATION REGISTER 7 HIGH (CONFIG7H: 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

- bit 7   **Unimplemented:** Read as '0'
- bit 6   **EBTRB:** Boot Block Table Read Protection bit  
 1 = Boot Block (000000-0001FFh) not protected from Table Reads executed in other blocks  
 0 = Boot Block (000000-0001FFh) protected from Table Reads executed in other blocks
- bit 5-0   **Unimplemented:** Read as '0'

Legend:

R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

# PIC18FXX2

---

## REGISTER 19-12: DEVICE ID REGISTER 1 FOR PIC18FXX2 (DEVID1: BYTE ADDRESS 3FFFFEh)

R	R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	
bit 7								bit 0

bit 7-5 **DEV2:DEV0:** Device ID bits

000 = PIC18F252

001 = PIC18F452

100 = PIC18F242

101 = PIC18F442

bit 4-0 **REV4:REV0:** Revision ID bits

These bits are used to indicate the device revision.

Legend:

R = Readable bit      P =Programmable bit      U = Unimplemented bit, read as '0'

- n = Value when device is unprogrammed      u = Unchanged from programmed state

## REGISTER 19-13: DEVICE ID REGISTER 2 FOR PIC18FXX2 (DEVID2: BYTE ADDRESS 3FFFFFh)

R	R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	
bit 7								bit 0

bit 7-0 **DEV10:DEV3:** Device ID bits

These bits are used with the DEV2:DEV0 bits in the Device ID Register 1 to identify the part number.

Legend:

R = Readable bit      P =Programmable bit      U = Unimplemented bit, read as '0'

- n = Value when device is unprogrammed      u = Unchanged from programmed state

## 19.2 Watchdog Timer (WDT)

The Watchdog Timer is a free running on-chip RC oscillator, which does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKI pin. That means that the WDT will run, even if the clock on the OSC1/CLKI and OSC2/CLKO/RA6 pins of the device has been stopped, for example, by execution of a SLEEP instruction.

During normal operation, a WDT time-out generates a device RESET (Watchdog Timer Reset). If the device is in SLEEP mode, a WDT time-out causes the device to wake-up and continue with normal operation (Watchdog Timer Wake-up). The TO bit in the RCON register will be cleared upon a WDT time-out.

The Watchdog Timer is enabled/disabled by a device configuration bit. If the WDT is enabled, software execution may not disable this function. When the WDTEN configuration bit is cleared, the SWDTEN bit enables/disables the operation of the WDT.

The WDT time-out period values may be found in the Electrical Specifications (Section 22.0) under parameter D031. Values for the WDT postscaler may be assigned using the configuration bits.

**Note:** The CLRWD and SLEEP instructions clear the WDT and the postscaler, if assigned to the WDT and prevent it from timing out and generating a device RESET condition.

**Note:** When a CLRWD instruction is executed and the postscaler is assigned to the WDT, the postscaler count will be cleared, but the postscaler assignment is not changed.

### 19.2.1 CONTROL REGISTER

Register 19-14 shows the WDTCON register. This is a readable and writable register, which contains a control bit that allows software to override the WDT enable configuration bit, only when the configuration bit has disabled the WDT.

#### REGISTER 19-14: WDTCON REGISTER

U-0	R/W-0						
—	—	—	—	—	—	—	SWDTEN

bit 7

bit 0

bit 7-1   **Unimplemented:** Read as '0'bit 0   **SWDTEN:** Software Controlled Watchdog Timer Enable bit

1 = Watchdog Timer is on

0 = Watchdog Timer is turned off if the WDTEN configuration bit in the configuration register = '0'

## Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

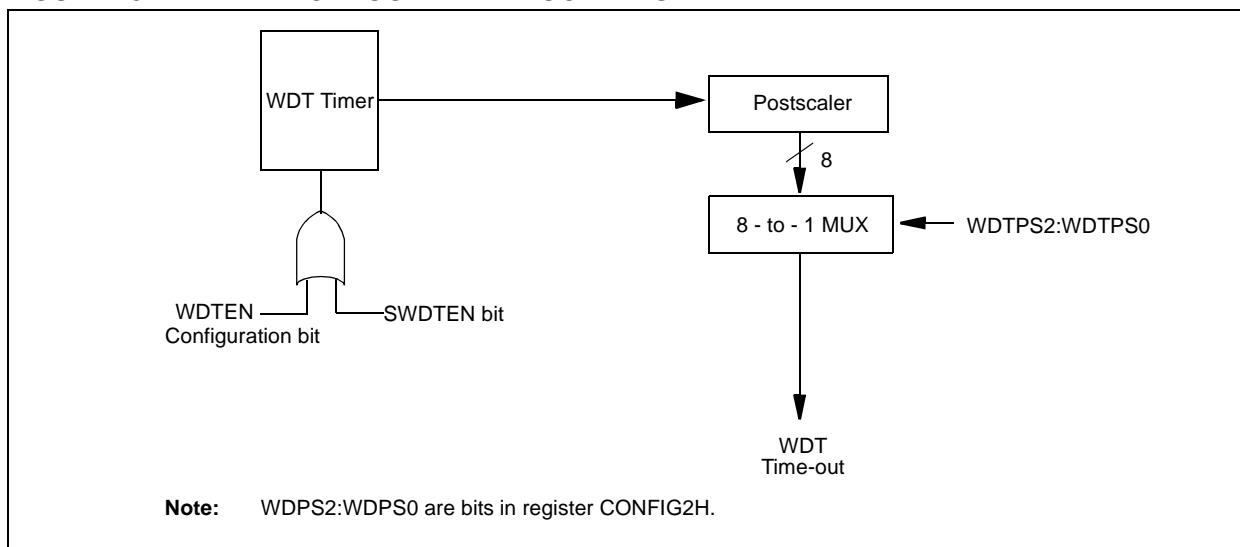
- n = Value at POR

# PIC18FXX2

## 19.2.2 WDT POSTSCALER

The WDT has a postscaler that can extend the WDT Reset period. The postscaler is selected at the time of the device programming, by the value written to the CONFIG2H configuration register.

**FIGURE 19-1: WATCHDOG TIMER BLOCK DIAGRAM**



**TABLE 19-2: SUMMARY OF WATCHDOG TIMER REGISTERS**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG2H	—	—	—	—	WDTPS2	WDTPS2	WDTPS0	WDTEN
RCON	IPEN	—	—	RI	TO	PD	POR	BOR
WDTCON	—	—	—	—	—	—	—	SWDTEN

Legend: Shaded cells are not used by the Watchdog Timer.

### 19.3 Power-down Mode (SLEEP)

Power-down mode is entered by executing a `SLEEP` instruction.

If enabled, the Watchdog Timer will be cleared, but keeps running, the `PD` bit (`RCON<3>`) is cleared, the `TO` (`RCON<4>`) bit is set, and the oscillator driver is turned off. The I/O ports maintain the status they had before the `SLEEP` instruction was executed (driving high, low or hi-impedance).

For lowest current consumption in this mode, place all I/O pins at either `VDD` or `Vss`, ensure no external circuitry is drawing current from the I/O pin, power-down the A/D and disable external clocks. Pull all I/O pins that are hi-impedance inputs, high or low externally, to avoid switching currents caused by floating inputs. The `T0CKI` input should also be at `VDD` or `Vss` for lowest current consumption. The contribution from on-chip pull-ups on PORTB should be considered.

The `MCLR` pin must be at a logic high level (`VIHMC`).

#### 19.3.1 WAKE-UP FROM SLEEP

The device can wake-up from `SLEEP` through one of the following events:

1. External RESET input on `MCLR` pin.
2. Watchdog Timer Wake-up (if WDT was enabled).
3. Interrupt from INT pin, RB port change or a Peripheral Interrupt.

The following peripheral interrupts can wake the device from `SLEEP`:

1. PSP read or write.
2. TMR1 interrupt. Timer1 must be operating as an asynchronous counter.
3. TMR3 interrupt. Timer3 must be operating as an asynchronous counter.
4. CCP Capture mode interrupt.
5. Special event trigger (Timer1 in Asynchronous mode using an external clock).
6. MSSP (START/STOP) bit detect interrupt.
7. MSSP transmit or receive in Slave mode (SPI/I<sup>2</sup>C).
8. USART RX or TX (Synchronous Slave mode).
9. A/D conversion (when A/D clock source is RC).
10. EEPROM write operation complete.
11. LVD interrupt.

Other peripherals cannot generate interrupts, since during `SLEEP`, no on-chip clocks are present.

External `MCLR` Reset will cause a device `RESET`. All other events are considered a continuation of program execution and will cause a “wake-up”. The `TO` and `PD` bits in the `RCON` register can be used to determine the cause of the device `RESET`. The `PD` bit, which is set on power-up, is cleared when `SLEEP` is invoked. The `TO` bit is cleared, if a WDT time-out occurred (and caused wake-up).

When the `SLEEP` instruction is being executed, the next instruction (`PC + 2`) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the `GIE` bit. If the `GIE` bit is clear (disabled), the device continues execution at the instruction after the `SLEEP` instruction. If the `GIE` bit is set (enabled), the device executes the instruction after the `SLEEP` instruction and then branches to the interrupt address. In cases where the execution of the instruction following `SLEEP` is not desirable, the user should have a `NOP` after the `SLEEP` instruction.

#### 19.3.2 WAKE-UP USING INTERRUPTS

When global interrupts are disabled (`GIE` cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

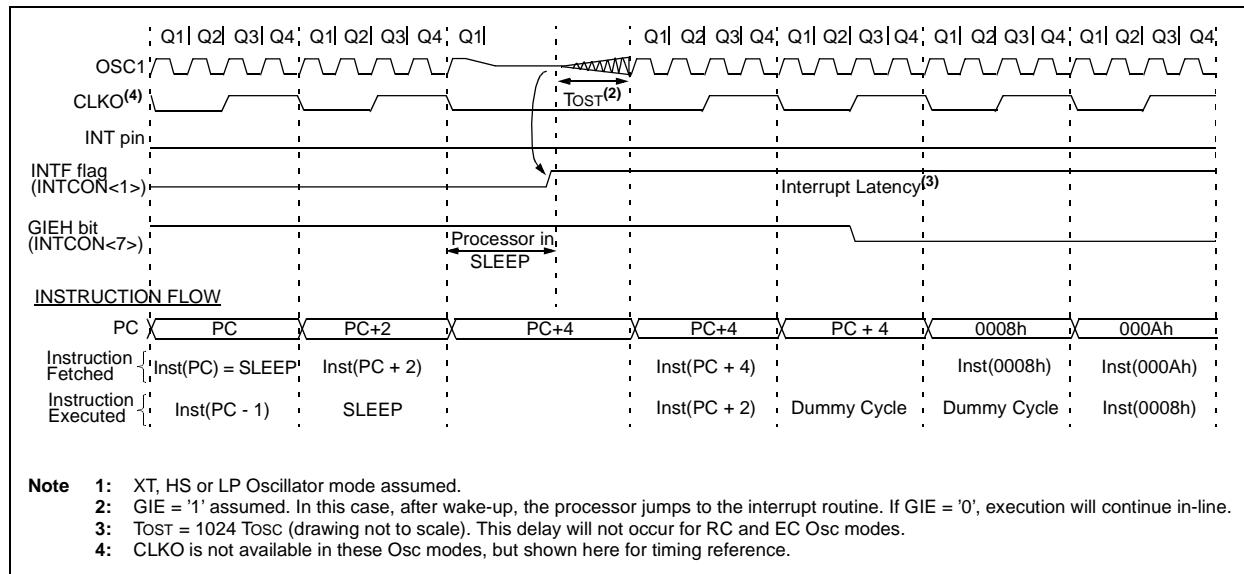
- If an interrupt condition (interrupt flag bit and interrupt enable bits are set) occurs **before** the execution of a `SLEEP` instruction, the `SLEEP` instruction will complete as a `NOP`. Therefore, the WDT and WDT postscaler will not be cleared, the `TO` bit will not be set and `PD` bits will not be cleared.
- If the interrupt condition occurs **during or after** the execution of a `SLEEP` instruction, the device will immediately wake-up from `SLEEP`. The `SLEEP` instruction will be completely executed before the wake-up. Therefore, the WDT and WDT postscaler will be cleared, the `TO` bit will be set and the `PD` bit will be cleared.

Even if the flag bits were checked before executing a `SLEEP` instruction, it may be possible for flag bits to become set before the `SLEEP` instruction completes. To determine whether a `SLEEP` instruction executed, test the `PD` bit. If the `PD` bit is set, the `SLEEP` instruction was executed as a `NOP`.

To ensure that the WDT is cleared, a `CLRWDT` instruction should be executed before a `SLEEP` instruction.

# PIC18FXX2

**FIGURE 19-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT<sup>(1,2)</sup>**



## 19.4 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 FLASH devices differs significantly from other PICmicro devices.

The user program memory is divided into five blocks. One of these is a boot block of 512 bytes. 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 19-3 shows the program memory organization for 16- and 32-Kbyte devices, and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 19-3.

**FIGURE 19-3: CODE PROTECTED PROGRAM MEMORY FOR PIC18F2XX/4XX**

MEMORY SIZE/DEVICE		Address Range	Block Code Protection Controlled By:
16 Kbytes (PIC18FX42)	32 Kbytes (PIC18FX52)		
Boot Block	Boot Block	000000h 0001FFh	CPB, WRTB, EBTRB
Block 0	Block 0	000200h 001FFFh	CP0, WRT0, EBTR0
Block 1	Block 1	002000h 003FFFh	CP1, WRT1, EBTR1
Unimplemented Read 0's	Block 2	004000h 005FFFh	CP2, WRT2, EBTR2
Unimplemented Read 0's	Block 3	006000h 007FFFh	CP3, WRT3, EBTR3
Unimplemented Read 0's	Unimplemented Read 0's	008000h  1FFFFFFh	(Unimplemented Memory Space)

**TABLE 19-3: SUMMARY OF CODE PROTECTION REGISTERS**

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h CONFIG5L	—	—	—	—	CP3	CP2	CP1	CP0
300009h CONFIG5H	CPD	CPB	—	—	—	—	—	—
30000Ah CONFIG6L	—	—	—	—	WRT3	WRT2	WRT1	WRT0
30000Bh CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—
30000Ch CONFIG7L	—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0
30000Dh CONFIG7H	—	EBTRB	—	—	—	—	—	—

Legend: Shaded cells are unimplemented.

## 19.4.1 PROGRAM MEMORY CODE PROTECTION

The user 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 User mode, the CPn bits have no direct effect. CPn bits inhibit external reads and writes. A block of user memory may be protected from Table Writes if the WRTn configuration bit is '0'. The EBTRn bits control Table Reads. For a block of user memory with the EBTRn bit set to '0', a Table Read instruction that executes from within that block is allowed to read. A Table Read instruction that executes from a location

outside of that block is not allowed to read, and will result in reading '0's. Figures 19-4 through 19-6 illustrate Table Write and Table Read protection.

**Note:** Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full chip erase or block erase function. The full chip erase and block erase functions can only be initiated via ICSP or an external programmer.

**FIGURE 19-4: TABLE WRITE (WRTn) DISALLOWED**

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 000FFF PC = 001FFE	000000h 0001FFh 000200h  TBLWT *  001FFFFh 002000h  TBLWT *  003FFFFh 004000h  005FFFFh 006000h  007FFFFh	WRTB,EBTRB = 11
PC = 004FFE		WRT0,EBTR0 = 01
		WRT1,EBTR1 = 11
		WRT2,EBTR2 = 11
		WRT3,EBTR3 = 11

**Results:** All Table Writes disabled to Blockn whenever WRTn = '0'.

**FIGURE 19-5: EXTERNAL BLOCK TABLE READ (EBTRn) DISALLOWED**

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 000FFF	000000h 0001FFh 000200h	WRTB,EBTRB = 11
PC = 002FFE	001FFFh 002000h TBLRD *	WRT0,EBTR0 = 10
	003FFFh 004000h	WRT1,EBTR1 = 11
	005FFFh 006000h	WRT2,EBTR2 = 11
	007FFFh	WRT3,EBTR3 = 11

**Results:** All Table Reads from external blocks to Blockn are disabled whenever EBTRn = '0'.  
TABLAT register returns a value of "0".

**FIGURE 19-6: EXTERNAL BLOCK TABLE READ (EBTRn) ALLOWED**

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 000FFF	000000h 0001FFh 000200h	WRTB,EBTRB = 11
PC = 001FFE	001FFFh 002000h TBLRD *	WRT0,EBTR0 = 10
	003FFFh 004000h	WRT1,EBTR1 = 11
	005FFFh 006000h	WRT2,EBTR2 = 11
	007FFFh	WRT3,EBTR3 = 11

**Results:** Table Reads permitted within Blockn, even when EBTRBn = '0'.  
TABLAT register returns the value of the data at the location TBLPTR.

## 19.4.2 DATA EEPROM CODE PROTECTION

The entire Data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of Data EEPROM. WRTD inhibits external writes to Data EEPROM. The CPU can continue to read and write Data EEPROM regardless of the protection bit settings.

## 19.4.3 CONFIGURATION REGISTER PROTECTION

The configuration registers can be write protected. The WRTC bit controls protection of the configuration registers. In User mode, the WRTC bit is readable only. WRTC can only be written via ICSP or an external programmer.

## 19.5 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 accessible 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.

The sequence for programming the ID locations is similar to programming the FLASH memory (see Section 5.5.1).

## 19.6 In-Circuit Serial Programming

PIC18FXXX 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.

## 19.7 In-Circuit Debugger

When the DEBUG bit in configuration register CONFIG4L is programmed to a '0', the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some of the resources are not available for general use. Table 19-4 shows which features are consumed by the background debugger.

**TABLE 19-4: 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, VDD, GND, RB7 and RB6. This will interface to the In-Circuit Debugger module available from Microchip or one of the third party development tool companies.

## 19.8 Low Voltage ICSP Programming

The LVP bit configuration register CONFIG4L enables low voltage ICSP programming. This mode allows the microcontroller to be programmed via ICSP using a VDD source in the operating voltage range. This only means that VPP does not have to be brought to VIHH, but can instead be left at the normal operating voltage. In this mode, the RB5/PGM pin is dedicated to the programming function and ceases to be a general purpose I/O pin. During programming, VDD is applied to the MCLR/VPP pin. To enter Programming mode, VDD must be applied to the RB5/PGM, provided the LVP bit is set. The LVP bit defaults to a ('1') from the factory.

- Note 1:** The High Voltage Programming mode is always available, regardless of the state of the LVP bit, by applying VIHH to the MCLR pin.
- 2:** While in low voltage ICSP mode, the RB5 pin can no longer be used as a general purpose I/O pin, and should be held low during normal operation to protect against inadvertent ICSP mode entry.
- 3:** When using low voltage ICSP programming (LVP), the pull-up on RB5 becomes disabled. If TRISB bit 5 is cleared, thereby setting RB5 as an output, LATB bit 5 must also be cleared for proper operation.

If Low Voltage Programming mode is not used, the LVP bit can be programmed to a '0' and RB5/PGM becomes a digital I/O pin. However, the LVP bit may only be programmed when programming is entered with VIHH on MCLR/VPP.

It should be noted that once the LVP bit is programmed to 0, only the High Voltage Programming mode is available and only High Voltage Programming mode can be used to program the device.

When using low voltage ICSP, the part must be supplied 4.5V to 5.5V, if a bulk erase will be executed. This includes reprogramming of the code protect bits from an on-state to off-state. For all other cases of low voltage ICSP, the part may be programmed at the normal operating voltage. This means unique user IDs, or user code can be reprogrammed or added.

## 20.0 INSTRUCTION SET SUMMARY

The PIC18FXXX instruction set adds many enhancements to the previous PICmicro instruction sets, while maintaining an easy migration from these PICmicro instruction sets.

Most instructions are a single program memory word (16-bits), but there are three instructions that require two program memory locations.

Each single word instruction is a 16-bit word divided into an OPCODE, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18FXXX instruction set summary in Table 20-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 20-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The destination of the result (specified by 'd')
3. The accessed memory (specified by 'a')

The file register designator 'f' specifies which file register is to be used by the instruction.

The destination designator 'd' specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the WREG register. If 'd' is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The bit in the file register (specified by 'b')
3. The accessed memory (specified by 'a')

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by 'k')
- The desired FSR register to load the literal value into (specified by 'f')
- No operand required (specified by '—')

The **control** instructions may use some of the following operands:

- A program memory address (specified by 'n')
- The mode of the Call or Return instructions (specified by 's')
- The mode of the Table Read and Table Write instructions (specified by 'm')
- No operand required (specified by '—')

All instructions are a single word, except for three double-word instructions. These three instructions were made double-word instructions so that all the required information is available in these 32 bits. In the second word, the 4-MSbs are 1's. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1  $\mu$ 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 20-1 shows the general formats that the instructions can have.

All examples use the format 'nnh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 20-2, lists the instructions recognized by the Microchip Assembler (MPASM<sup>TM</sup>).

Section 20.1 provides a description of each instruction.

# PIC18FXX2

---

**TABLE 20-1: 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.
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 (0x00 to 0xFF)
fs	12-bit Register file address (0x000 to 0xFFFF). This is the source address.
fd	12-bit Register file address (0x000 to 0xFFFF). This is the destination address.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value)
label	Label name
mm	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
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)
u	Unused or Unchanged
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.
TBLPTR	21-bit Table Pointer (points to a Program Memory location)
TABLAT	8-bit Table Latch
TOS	Top-of-Stack
PC	Program Counter
PCL	Program Counter Low Byte
PCH	Program Counter High Byte
PCLATH	Program Counter High Byte Latch
PCLATU	Program Counter Upper Byte Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
TO	Time-out bit
PD	Power-down bit
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
[ ]	Optional
( )	Contents
→	Assigned to
< >	Register bit field
ε	In the set of
italics	User defined term (font is courier)

---

**FIGURE 20-1: GENERAL FORMAT FOR INSTRUCTIONS**

Byte-oriented file register operations	Example Instruction																																																
<table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">10</td><td style="padding: 2px;">9</td><td style="padding: 2px;">8</td><td style="padding: 2px;">7</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td><td style="border: 1px solid black; padding: 2px;">d</td><td style="border: 1px solid black; padding: 2px;">a</td><td colspan="3" style="border: 1px solid black; padding: 2px; text-align: center;">f (FILE #)</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>	15	10	9	8	7	0	OPCODE	d	a	f (FILE #)			ADDWF MYREG, W, B																																				
15	10	9	8	7	0																																												
OPCODE	d	a	f (FILE #)																																														
<b>Byte to Byte move operations (2-word)</b> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">12</td><td style="padding: 2px;">11</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td><td colspan="3" style="border: 1px solid black; padding: 2px; text-align: center;">f (Source FILE #)</td></tr> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">12</td><td style="padding: 2px;">11</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">1111</td><td colspan="3" style="border: 1px solid black; padding: 2px; text-align: center;">f (Destination FILE #)</td></tr> </table> <p>f = 12-bit file register address</p>	15	12	11	0	OPCODE	f (Source FILE #)			15	12	11	0	1111	f (Destination FILE #)			MOVFF MYREG1, MYREG2																																
15	12	11	0																																														
OPCODE	f (Source FILE #)																																																
15	12	11	0																																														
1111	f (Destination FILE #)																																																
<b>Bit-oriented file register operations</b> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">12</td><td style="padding: 2px;">11</td><td style="padding: 2px;">9</td><td style="padding: 2px;">8</td><td style="padding: 2px;">7</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td><td style="border: 1px solid black; padding: 2px;">b (BIT #)</td><td style="border: 1px solid black; padding: 2px;">a</td><td colspan="4" style="border: 1px solid black; padding: 2px; text-align: center;">f (FILE #)</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>	15	12	11	9	8	7	0	OPCODE	b (BIT #)	a	f (FILE #)				BSF MYREG, bit, B																																		
15	12	11	9	8	7	0																																											
OPCODE	b (BIT #)	a	f (FILE #)																																														
<b>Literal operations</b> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">8</td><td style="padding: 2px;">7</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td><td colspan="3" style="border: 1px solid black; padding: 2px; text-align: center;">k (literal)</td></tr> </table> <p>k = 8-bit immediate value</p>	15	8	7	0	OPCODE	k (literal)			MOVLW 0x7F																																								
15	8	7	0																																														
OPCODE	k (literal)																																																
<b>Control operations</b> <p><b>CALL, GOTO and Branch operations</b></p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">8</td><td style="padding: 2px;">7</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td><td colspan="3" style="border: 1px solid black; padding: 2px; text-align: center;">n&lt;7:0&gt; (literal)</td></tr> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">12</td><td style="padding: 2px;">11</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">1111</td><td colspan="3" style="border: 1px solid black; padding: 2px; text-align: center;">n&lt;19:8&gt; (literal)</td></tr> </table> <p>n = 20-bit immediate value</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">8</td><td style="padding: 2px;">7</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td><td style="border: 1px solid black; padding: 2px;">S</td><td colspan="2" style="border: 1px solid black; padding: 2px; text-align: center;">n&lt;7:0&gt; (literal)</td></tr> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">12</td><td style="padding: 2px;">11</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;"></td><td colspan="3" style="border: 1px solid black; padding: 2px; text-align: center;">n&lt;19:8&gt; (literal)</td></tr> </table> <p>S = Fast bit</p> <table style="margin-left: auto; margin-right: auto;"> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">11</td><td style="padding: 2px;">10</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td><td colspan="3" style="border: 1px solid black; padding: 2px; text-align: center;">n&lt;10:0&gt; (literal)</td></tr> <tr> <td style="padding: 2px;">15</td><td style="padding: 2px;">8</td><td style="padding: 2px;">7</td><td style="padding: 2px;">0</td></tr> <tr> <td style="border: 1px solid black; padding: 2px;">OPCODE</td><td colspan="3" style="border: 1px solid black; padding: 2px; text-align: center;">n&lt;7:0&gt; (literal)</td></tr> </table> <p>CALL MYFUNC</p> <p>BRA MYFUNC</p> <p>BC MYFUNC</p>	15	8	7	0	OPCODE	n<7:0> (literal)			15	12	11	0	1111	n<19:8> (literal)			15	8	7	0	OPCODE	S	n<7:0> (literal)		15	12	11	0		n<19:8> (literal)			15	11	10	0	OPCODE	n<10:0> (literal)			15	8	7	0	OPCODE	n<7:0> (literal)			GOTO Label
15	8	7	0																																														
OPCODE	n<7:0> (literal)																																																
15	12	11	0																																														
1111	n<19:8> (literal)																																																
15	8	7	0																																														
OPCODE	S	n<7:0> (literal)																																															
15	12	11	0																																														
	n<19:8> (literal)																																																
15	11	10	0																																														
OPCODE	n<10:0> (literal)																																																
15	8	7	0																																														
OPCODE	n<7:0> (literal)																																																

# PIC18FXX2

---

**TABLE 20-2: PIC18FXXX INSTRUCTION SET**

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb	Lsb					
<b>BYTE-ORIENTED FILE REGISTER OPERATIONS</b>									
ADDWF f, d, a	Add WREG and f	1	0010 01da0	ffff ffff	C, DC, Z, OV, N		1, 2		
ADDWFC f, d, a	Add WREG and Carry bit to f	1	0010 0da	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	ffff ffff	None				
			1111 ffff ffff 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				
NEGF f, a	Negate f	1	0110 110a	ffff ffff	C, DC, Z, OV, N		1, 2		
RLCF f, d, a	Rotate Left f through Carry	1	0011 01da	ffff ffff	C, Z, N				
RLNCF f, d, a	Rotate Left f (No Carry)	1	0100 01da	ffff ffff	Z, N		1, 2		
RRCF f, d, a	Rotate Right f through Carry	1	0011 00da	ffff ffff	C, Z, N				
RRNCF f, d, a	Rotate Right f (No Carry)	1	0100 00da	ffff ffff	Z, N				
SETf f, a	Set f	1	0110 100a	ffff ffff	None				
SUBFWB f, d, a	Subtract f from WREG with borrow	1	0101 01da	ffff ffff	C, DC, Z, OV, N		1, 2		
SUBWF f, d, a	Subtract WREG from f	1	0101 11da	ffff ffff	C, DC, Z, OV, N				
SUBWFB f, d, a	Subtract WREG from f with borrow	1	0101 10da	ffff ffff	C, DC, Z, OV, N		1, 2		
SWAPf f, d, a	Swap nibbles in f	1	0011 10da	ffff ffff	None		4		
TSTFSZ f, a	Test f, skip if 0	1 (2 or 3)	0110 011a	ffff ffff	None		1, 2		
XORWF f, d, a	Exclusive OR WREG with f	1	0001 10da	ffff ffff	Z, N				
<b>BIT-ORIENTED FILE REGISTER OPERATIONS</b>									
BCF f, b, a	Bit Clear f	1	1001 bbba	ffff ffff	None		1, 2		
BSF f, b, a	Bit Set f	1	1000 bbba	ffff ffff	None		1, 2		
BTFSC f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011 bbba	ffff ffff	None		3, 4		
BTFSS f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010 bbba	ffff ffff	None		3, 4		
BTG f, d, a	Bit Toggle f	1	0111 bbba	ffff ffff	None		1, 2		

- Note 1:** When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.
- 3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a **NOP**.
- 4:** Some instructions are 2-word instructions. The second word of these instructions will be executed as a **NOP**, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.
- 5:** If the Table Write starts the write cycle to internal memory, the write will continue until terminated.

TABLE 20-2: PIC18FXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
			MSb			LSb		
<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	2	1110	0001	nnnn	nnnn	None	
BOV n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA n	Branch Unconditionally	1 (2)	1101	0nnn	nnnn	nnnn	None	
BZ n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL n, s	Call subroutine1st 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 address1st 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 TMRO register (and, where applicable, d = 1), the prescaler will be cleared if assigned.
- 3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4:** Some instructions are 2-word instructions. The second word of these instructions will be executed as a NOP, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.
- 5:** If the Table Write starts the write cycle to internal memory, the write will continue until terminated.

# PIC18FXX2

---

TABLE 20-2: PIC18FXXX 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 FSRx 1st word	2	1110	1110	00ff	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 (5)	0000	0000	0000	1100	None		
TBLWT*+	Table Write with post-increment		0000	0000	0000	1101	None		
TBLWT*-	Table Write with post-decrement		0000	0000	0000	1110	None		
TBLWT+*	Table Write with pre-increment		0000	0000	0000	1111	None		

- Note**
- 1: When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
  - 2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.
  - 3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
  - 4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a NOP, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.
  - 5: If the Table Write starts the write cycle to internal memory, the write will continue until terminated.

## 20.1 Instruction Set

<b>ADDLW</b>	<b>ADD literal to W</b>											
Syntax:	[ <i>label</i> ] ADDLW <i>k</i>											
Operands:	0 ≤ <i>k</i> ≤ 255											
Operation:	$(W) + k \rightarrow W$											
Status Affected:	N, OV, C, DC, Z											
Encoding:	0000	1111	kkkk	kkkk								
Description:	The contents of W are added to the 8-bit literal 'k' and the result is placed in W.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<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: ADDLW 0x15

Before Instruction

W = 0x10

After Instruction

W = 0x25

<b>ADDWF</b>	<b>ADD W to f</b>			
Syntax:	[ <i>label</i> ] ADDWF <i>f</i> [,d [,a]]			
Operands:	0 ≤ <i>f</i> ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	$(W) + (f) \rightarrow \text{dest}$			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0010	01da	ffff	ffff
Description:	Add W to register 'f'. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the BSR is used.			
Words:	1			
Cycles:	1			

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: ADDWF REG, 0, 0

Before Instruction

W = 0x17  
REG = 0xC2

After Instruction

W = 0xD9  
REG = 0xC2

# PIC18FXX2

---

<b>ADDWFC</b>	<b>ADD W and Carry bit to f</b>								
Syntax:	[label] ADDWFC f [,d [,a]]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(W) + (f) + (C) → dest								
Status Affected:	N,OV,C,DC,Z								
Encoding:	0010 00da ffff ffff								
Description:	Add W, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the BSR will not be overridden.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: ADDWFC REG, 0, 1

Before Instruction

Carry bit = 1  
REG = 0x02  
W = 0x4D

After Instruction

Carry bit = 0  
REG = 0x02  
W = 0x50

<b>ANDLW</b>	<b>AND literal with W</b>								
Syntax:	[label] ANDLW k								
Operands:	0 ≤ k ≤ 255								
Operation:	(W) .AND. k → W								
Status Affected:	N,Z								
Encoding:	0000 1011 kkkk kkkk								
Description:	The contents of W are ANDed with the 8-bit literal 'k'. The result is placed in W.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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 W</td> </tr> </tbody> </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 0x5F

Before Instruction

W = 0xA3

After Instruction

W = 0x03

<b>ANDWF</b>	<b>AND W with f</b>								
Syntax:	[ <i>label</i> ] 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:	0001 01da ffff ffff								
Description:	The contents of W are AND'ed with register 'f'. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the BSR will not be overridden (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: ANDWF REG, 0, 0

Before Instruction

W = 0x17  
REG = 0xC2

After Instruction

W = 0x02  
REG = 0xC2

<b>BC</b>	<b>Branch if Carry</b>												
Syntax:	[ <i>label</i> ] BC n												
Operands:	$-128 \leq n \leq 127$												
Operation:	if carry bit is '1' $(PC) + 2 + 2n \rightarrow PC$												
Status Affected:	None												
Encoding:	1110 0010 nnnn nnnn												
Description:	If the Carry bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC+2+2n$ . This instruction is then a two-cycle instruction.												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:	<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:

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)

# PIC18FXX2

---

<b>BCF</b>		<b>Bit Clear f</b>	
Syntax:	[ <i>label</i> ] BCF f,b[,a]		
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]		
Operation:	0 → f<b>		
Status Affected:	None		
Encoding:	1001 bbba ffff ffff		
Description:	Bit 'b' in register 'f' is cleared. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
	Q1      Q2      Q3      Q4		
	Decode      Read register 'f'      Process Data      Write register 'f'		

Example: BCF FLAG\_REG, 7, 0

Before Instruction  
FLAG\_REG = 0xC7  
After Instruction  
FLAG\_REG = 0x47

<b>BN</b>		<b>Branch if Negative</b>	
Syntax:	[ <i>label</i> ] BN n		
Operands:	-128 ≤ n ≤ 127		
Operation:	if negative bit is '1' (PC) + 2 + 2n → PC		
Status Affected:	None		
Encoding:	1110 0110 nnnn nnnn		
Description:	If the Negative bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.		
Words:	1		
Cycles:	1(2)		
Q Cycle Activity:			
If Jump:			
	Q1      Q2      Q3      Q4		
	Decode      Read literal 'n'      Process Data      Write to PC		
	No operation      No operation      No operation      No operation		
If No Jump:			
	Q1      Q2      Q3      Q4		
	Decode      Read literal 'n'      Process Data      No operation		

Example: HERE BN Jump

Before Instruction  
PC = address (HERE)  
After Instruction  
If Negative = 1;  
PC = address (Jump)  
If Negative = 0;  
PC = address (HERE+2)

BNC	Branch if Not Carry															
Syntax:	[ <i>label</i> ] BNC <i>n</i>															
Operands:	$-128 \leq n \leq 127$															
Operation:	if carry bit is '0' $(PC) + 2 + 2n \rightarrow PC$															
Status Affected:	None															
Encoding:	1110	0011	nnnn	nnnn												
Description:	<p>If the Carry bit is '0', then the program will branch.</p> <p>The 2's complement number '<math>2n</math>' 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:																
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													

<u>Example:</u>	HERE	BNC	Jump
<b>Before Instruction</b>			
PC	=	address	(HERE)
<b>After Instruction</b>			
If Carry	=	0;	
PC	=	address	(Jump)
If Carry	=	1;	
PC	=	address	(HERE+2)

BNN	Branch if Not Negative															
Syntax:	[ <i>label</i> ] BNN <i>n</i>															
Operands:	$-128 \leq n \leq 127$															
Operation:	if negative bit is '0' $(PC) + 2 + 2n \rightarrow PC$															
Status Affected:	None															
Encoding:	1110	0111	nnnn	nnnn												
Description:	<p>If the Negative bit is '0', then the program will branch.</p> <p>The 2's complement number '<math>2n</math>' 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:																
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													

<u>Example:</u>	HERE	BNN	Jump
<b>Before Instruction</b>			
PC	=	address	(HERE)
<b>After Instruction</b>			
If Negative	=	0;	
PC	=	address	(Jump)
If Negative	=	1;	
PC	=	address	(HERE+2)

# PIC18FXX2

---

<b>BNOV</b>		<b>Branch if Not Overflow</b>	
Syntax:	[ <i>label</i> ] BNOV n		
Operands:	-128 ≤ n ≤ 127		
Operation:	if overflow bit is '0' (PC) + 2 + 2n → PC		
Status Affected:	None		
Encoding:	1110 0101 nnnn nnnn		
Description:	If the Overflow bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.		
Words:	1		
Cycles:	1(2)		
Q Cycle Activity:			
If Jump:			
	Q1 Q2 Q3 Q4		
	Decode Read literal 'n' No operation	Process Data No operation	Write to PC No operation
If No Jump:			
	Q1 Q2 Q3 Q4		
	Decode Read literal 'n' No operation	Process Data No operation	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:	[ <i>label</i> ] BNZ n		
Operands:	-128 ≤ n ≤ 127		
Operation:	if zero bit is '0' (PC) + 2 + 2n → PC		
Status Affected:	None		
Encoding:	1110 0001 nnnn nnnn		
Description:	If the Zero bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.		
Words:	1		
Cycles:	1(2)		
Q Cycle Activity:			
If Jump:			
	Q1 Q2 Q3 Q4		
	Decode Read literal 'n' No operation	Process Data No operation	Write to PC No operation
If No Jump:			
	Q1 Q2 Q3 Q4		
	Decode Read literal 'n' No operation	Process Data No operation	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)

<b>BRA</b>	<b>Unconditional Branch</b>												
Syntax:	[ <i>label</i> ] BRA n												
Operands:	-1024 ≤ n ≤ 1023												
Operation:	(PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1101</td><td>0nnn</td><td>nnnn</td><td>nnnn</td></tr> </table>	1101	0nnn	nnnn	nnnn								
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:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read literal 'n'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write to PC</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">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										

Example:      HERE      BRA    Jump

Before Instruction  
 PC                =    address (HERE)

After Instruction  
 PC                =    address (Jump)

<b>BSF</b>	<b>Bit Set f</b>								
Syntax:	[ <i>label</i> ] BSF f,b[,a]								
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]								
Operation:	1 → f<b>								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1000</td><td>bbba</td><td>ffff</td><td>ffff</td></tr> </table>	1000	bbba	ffff	ffff				
1000	bbba	ffff	ffff						
Description:	Bit 'b' in register 'f' is set. If 'a' is 0 Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read register 'f'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write register 'f'</td> </tr> </tbody> </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:      BSF      FLAG\_REG, 7, 1

Before Instruction  
 FLAG\_REG = 0x0A

After Instruction  
 FLAG\_REG = 0x8A

# PIC18FXX2

---

<b>BTFSC</b>	<b>Bit Test File, Skip if Clear</b>
Syntax:	[ <i>label</i> ] BTFSC f,b[,a]
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]
Operation:	skip if (f<b>) = 0
Status Affected:	None
Encoding:	
Description:	<p>If bit 'b' in register 'f' is 0, then the next instruction is skipped.</p> <p>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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).</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 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)

<b>BTFSS</b>	<b>Bit Test File, Skip if Set</b>
Syntax:	[ <i>label</i> ] BTFSS f,b[,a]
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]
Operation:	skip if (f<b>) = 1
Status Affected:	None
Encoding:	
Description:	<p>If bit 'b' in register 'f' is 1, then the next instruction is skipped.</p> <p>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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).</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 BTFSS FLAG, 1, 0  
 FALSE :  
 TRUE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1>	=	0;
PC	=	address (FALSE)
If FLAG<1>	=	1;
PC	=	address ('TRUE')

<b>BTG</b>	<b>Bit Toggle f</b>								
Syntax:	[ <i>label</i> ] BTG f,b[ <i>a</i> ]								
Operands:	$0 \leq f \leq 255$ $0 \leq b \leq 7$ $a \in [0,1]$								
Operation:	$(f<b>) \rightarrow f<b>$								
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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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 register 'f'</td></tr> </tbody> </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 [0x75]

After Instruction:

PORTC = 0110 0101 [0x65]

<b>BOV</b>	<b>Branch if Overflow</b>												
Syntax:	[ <i>label</i> ] BOV n												
Operands:	$-128 \leq n \leq 127$												
Operation:	if overflow 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>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="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 '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:

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)

# PIC18FXX2

---

<b>BZ</b>		<b>Branch if Zero</b>	
Syntax:	[ <i>label</i> ] BZ n		
Operands:	-128 ≤ n ≤ 127		
Operation:	if Zero bit is '1' (PC) + 2 + 2n → PC		
Status Affected:	None		
Encoding:	1110 0000 nnnn nnnn		
Description:	If the Zero bit is '1', then the program will branch.  The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.		
Words:	1		
Cycles:	1(2)		
Q Cycle Activity:			
If Jump:			
	Q1 Q2 Q3 Q4		
	Decode Read literal 'n' No operation	Process Data No operation	Write to PC No operation

If No Jump:			
Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation
No operation	No operation	No operation	No operation

Example: HERE BZ Jump

Before Instruction

PC = address (HERE)

After Instruction

If Zero	=	1;
PC	=	address (Jump)
If Zero	=	0;
PC	=	address (HERE+2)

<b>CALL</b>		<b>Subroutine Call</b>	
Syntax:	[ <i>label</i> ] CALL k [.s]		
Operands:	0 ≤ k ≤ 1048575		
	s ∈ [0,1]		
Operation:	(PC) + 4 → TOS, k → PC<20:1>, if s = 1 (W) → WS, (STATUS) → STATUS,SS, (BSR) → BSRS		
Status Affected:	None		
Encoding:	1110 110s k <sub>7</sub> kkk k <sub>19</sub> kkk k <sub>19</sub> kkk k <sub>7</sub> kkk k <sub>19</sub> kkk k <sub>7</sub> kkk k <sub>19</sub> kkk k <sub>7</sub> kkk k <sub>19</sub> kkk		
1st word (k<7:0>)	1110	110s	k <sub>7</sub> kkk
2nd word(k<19:8>)	1111	k <sub>19</sub> kkk	kkkk
Description:	Subroutine call of entire 2 Mbyte memory range. First, return address (PC+ 4) is pushed onto the return stack. If 's' = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUS,SS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-cycle instruction.		

Words:	2
Cycles:	2
Q Cycle Activity:	
Q1 Q2 Q3 Q4	
Decode	Read literal 'k'<7:0>, Push PC to stack
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

<b>CLRF</b>	<b>Clear f</b>				
Syntax:	[ <i>label</i> ] 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"><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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).				
Words:	1				
Cycles:	1				
Q Cycle Activity:					
	Q1            Q2            Q3            Q4				
	Decode	Read register 'f'	Process Data	Write register 'f'	

Example: CLRF FLAG\_REG,1

Before Instruction

FLAG\_REG = 0x5A

After Instruction

FLAG\_REG = 0x00

<b>CLRWDT</b>	<b>Clear Watchdog Timer</b>				
Syntax:	[ <i>label</i> ] CLRWDT				
Operands:	None				
Operation:	$000h \rightarrow WDT$ , $000h \rightarrow WDT$ postscaler, $1 \rightarrow \overline{TO}$ , $1 \rightarrow \overline{PD}$				
Status Affected:	$\overline{TO}, \overline{PD}$				
Encoding:	<table border="1"><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 TO and PD are set.				
Words:	1				
Cycles:	1				
Q Cycle Activity:					
	Q1            Q2            Q3            Q4				
	Decode	No operation	Process Data	No operation	

Example: CLRWDT

Before Instruction

WDT Counter = ?

After Instruction

WDT Counter	=	0x00
WDT Postscaler	=	0
TO	=	1
PD	=	1

# PIC18FXX2

---

<b>COMF</b>	<b>Complement f</b>								
Syntax:	[ label ] COMF f [,d [,a]]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f) → dest								
Status Affected:	N, Z								
Encoding:	0001 11da ffff ffff								
Description:	The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<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 = 0x13  
After Instruction  
REG = 0x13  
W = 0xEC

<b>CPFSEQ</b>	<b>Compare f with W, skip if f = W</b>
Syntax:	[ label ] CPFSEQ f [,a]
Operands:	0 ≤ f ≤ 255 a ∈ [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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2)
<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 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				
Syntax:	[ label ] CPFSGT f [,a]				
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$				
Operation:	$(f) - (W)$ , skip if $(f) > (W)$ (unsigned comparison)				
Status Affected:	None				
Encoding:	<table border="1"><tr><td>0110</td><td>010a</td><td>ffff</td><td>ffff</td></tr></table>	0110	010a	ffff	ffff
0110	010a	ffff	ffff		
Description:	Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction. If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).				
Words:	1				
Cycles:	1(2)				
	<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 CPFSGT REG, 0  
NGREATER :  
GREATER :

Before Instruction

PC = Address (HERE)  
W = ?

After Instruction

If REG > W;  
PC = Address (GREATER)  
If REG ≤ W;  
PC = Address (NGREATER)

CPFSLT	Compare f with W, skip if f < W				
Syntax:	[ label ] CPFSLT f [,a]				
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$				
Operation:	$(f) - (W)$ , skip if $(f) < (W)$ (unsigned comparison)				
Status Affected:	None				
Encoding:	<table border="1"><tr><td>0110</td><td>000a</td><td>ffff</td><td>ffff</td></tr></table>	0110	000a	ffff	ffff
0110	000a	ffff	ffff		
Description:	Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the BSR will not be overridden (default).				
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 CPFSLT REG, 1  
NLESS :  
LESS :

Before Instruction

PC = Address (HERE)  
W = ?

After Instruction

If REG < W;  
PC = Address (LESS)  
If REG ≥ W;  
PC = Address (NLESS)

# PIC18FXX2

---

DAW	Decimal Adjust W Register								
Syntax:	[ label ] DAW								
Operands:	None								
Operation:	If [W<3:0> >9] or [DC = 1] then (W<3:0>) + 6 → W<3:0>; else (W<3:0>) → W<3:0>;  If [W<7:4> >9] or [C = 1] then (W<7:4>) + 6 → W<7:4>; else (W<7:4>) → W<7:4>;								
Status Affected:	C								
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				
0000	0000	0000	0111						
Description:	DAW adjusts the eight-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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
Q1	Q2	Q3	Q4						
Decode	Read register W	Process Data	Write W						

Example1: DAW

Before Instruction

```

W      = 0xA5
C      = 0
DC     = 0

```

After Instruction

```

W      = 0x05
C      = 1
DC     = 0

```

Example 2:

Before Instruction

```

W      = 0xCE
C      = 0
DC     = 0

```

After Instruction

```

W      = 0x34
C      = 1
DC     = 0

```

DECF	Decrement f								
Syntax:	[ label ] DECF f [,d [,a]]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f) – 1 → dest								
Status Affected:	C, DC, N, OV, Z								
Encoding:	<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	01da	ffff	ffff						
Description:	Decrement register 'f'. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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: DECF CNT, 1, 0

Before Instruction

```

CNT    = 0x01
Z      = 0

```

After Instruction

```

CNT    = 0x00
Z      = 1

```

<b>DECFSZ</b>	<b>Decrement f, skip if 0</b>
Syntax:	[ <i>label</i> ] 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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2)
<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	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE DECFSZ CNT, 1, 1  
GOTO LOOP  
CONTINUE

Before Instruction

PC = Address (HERE)

After Instruction

CNT = CNT - 1  
If CNT = 0;  
PC = Address (CONTINUE)  
If CNT ≠ 0;  
PC = Address (HERE+2)

<b>DCFSNZ</b>	<b>Decrement f, skip if not 0</b>
Syntax:	[ <i>label</i> ] 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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2)
<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	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE DCFSNZ TEMP, 1, 0  
ZERO :  
NZERO :

Before Instruction

TEMP = ?

After Instruction

TEMP = TEMP - 1,  
If TEMP = 0;  
PC = Address (ZERO)  
If TEMP ≠ 0;  
PC = Address (NZERO)

# PIC18FXX2

---

GOTO	Unconditional Branch				INCF	Increment f			
Syntax:	[ <i>label</i> ] GOTO k				Syntax:	[ <i>label</i> ] INCF f [,d [,a]]			
Operands:	0 ≤ k ≤ 1048575				Operands:	0 ≤ f ≤ 255			
Operation:	k → PC<20:1>				d ∈ [0,1]				
Status Affected:	None				a ∈ [0,1]				
Encoding:					Operation:	(f) + 1 → dest			
1st word (k<7:0>)	1110	1111	k <sub>7</sub> kkk	kkk <sub>0</sub>	Status Affected:	C, DC, N, OV, Z			
2nd word(k<19:8>)	1111	k <sub>19</sub> kkk	kkkk	kkk <sub>8</sub>	Encoding:		0010	10da	ffff
Description:	GOTO allows an unconditional branch anywhere within entire 2 Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle instruction.				Description:	The contents of register 'f' are incremented. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).			
Words:	2				Words:	1			
Cycles:	2				Cycles:	1			
Q Cycle Activity:					Q Cycle Activity:				
	Q1	Q2	Q3	Q4		Q1	Q2	Q3	Q4
	Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC		Decode	Read register 'f'	Process Data	Write to destination
	No operation	No operation	No operation	No operation					

Example: GOTO THERE

After Instruction

PC = Address (THERE)

Example: INCF CNT, 1, 0

Before Instruction

CNT	=	0xFF
Z	=	0
C	=	?
DC	=	?

After Instruction

CNT	=	0x00
Z	=	1
C	=	1
DC	=	1

<b>INCFSZ</b>	<b>Increment f, skip if 0</b>							
Syntax:	[ <i>label</i> ] 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:	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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).							
Words:	1							
Cycles:	1(2)							
<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	Write to destination				

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE INCFSZ CNT, 1, 0  
NZERO :  
ZERO :

Before Instruction

PC = Address (HERE)

After Instruction

CNT = CNT + 1  
If CNT = 0;  
PC = Address (ZERO)  
If CNT ≠ 0;  
PC = Address (NZERO)

<b>INFSNZ</b>	<b>Increment f, skip if not 0</b>							
Syntax:	[ <i>label</i> ] 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:	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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).							
Words:	1							
Cycles:	1(2)							
<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	Write to destination				

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE INFSNZ REG, 1, 0  
ZERO :  
NZERO :

Before Instruction

PC = Address (HERE)

After Instruction

REG = REG + 1  
If REG ≠ 0;  
PC = Address (NZERO)  
If REG = 0;  
PC = Address (ZERO)

# PIC18FXX2

---

IORLW	Inclusive OR literal with W								
Syntax:	[ <i>label</i> ] IORLW <i>k</i>								
Operands:	$0 \leq k \leq 255$								
Operation:	(W) .OR. <i>k</i> $\rightarrow$ 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 OR'ed with the eight-bit literal ' <i>k</i> '. The result is placed in W.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read literal '<i>k</i>'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write to W</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal ' <i>k</i> '	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read literal ' <i>k</i> '	Process Data	Write to W						

Example: IORLW 0x35

Before Instruction

W = 0x9A

After Instruction

W = 0xBF

IORWF	Inclusive OR W with f								
Syntax:	[ <i>label</i> ] IORWF f [, <i>d</i> [, <i>a</i> ]]								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	(W) .OR. (f) $\rightarrow$ dest								
Status Affected:	N, Z								
Encoding:	<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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read register 'f'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: IORWF RESULT, 0, 1

Before Instruction

RESULT = 0x13  
W = 0x91

After Instruction

RESULT = 0x13  
W = 0x93

LFSR	Load FSR												
Syntax:	[ <i>label</i> ] LFSR f,k												
Operands:	$0 \leq f \leq 2$ $0 \leq k \leq 4095$												
Operation:	$k \rightarrow \text{FSR}_f$												
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_{kk}</math></td> </tr> <tr> <td>1111</td> <td>0000</td> <td><math>k_7k_{kk}</math></td> <td>kkkk</td> </tr> </table>	1110	1110	00ff	$k_{11}k_{kk}$	1111	0000	$k_7k_{kk}$	kkkk				
1110	1110	00ff	$k_{11}k_{kk}$										
1111	0000	$k_7k_{kk}$	kkkk										
Description:	The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'.												
Words:	2												
Cycles:	2												
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">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, 0x3AB

After Instruction

```
FSR2H      = 0x03
FSR2L      = 0xAB
```

MOVF	Move f				
Syntax:	[ <i>label</i> ] MOVF f [,d [,a]]				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$f \rightarrow \text{dest}$				
Status Affected:	N, Z				
Encoding:	<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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).				
Words:	1				
Cycles:	1				

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write W

Example: MOVF REG, 0, 0

Before Instruction

```
REG      = 0x22
W       = 0xFF
```

After Instruction

```
REG      = 0x22
W       = 0x22
```

# PIC18FXX2

---

<b>MOVFF</b>	<b>Move f to f</b>
Syntax:	[ <i>label</i> ] MOVFF <i>f<sub>s</sub>,f<sub>d</sub></i>
Operands:	$0 \leq f_s \leq 4095$ $0 \leq f_d \leq 4095$
Operation:	$(f_s) \rightarrow f_d$
Status Affected:	None
Encoding:	
1st word (source)	1100      ffff      ffff      ffff <sub>s</sub>
2nd word (destin.)	1111      ffff      ffff      ffff <sub>d</sub>
Description:	The contents of source register ' <i>f<sub>s</sub></i> ' are moved to destination register ' <i>f<sub>d</sub></i> '. Location of source ' <i>f<sub>s</sub></i> ' can be anywhere in the 4096 byte data space (000h to FFFh), and location of destination ' <i>f<sub>d</sub></i> ' can also be anywhere from 000h to FFFh. Either source or destination can be W (a useful special situation). MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
<b>Note:</b>	The MOVFF instruction should not be used to modify interrupt settings while any interrupt is enabled. See Section 8.0 for more information.
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      =    0x33  
REG2      =    0x11

After Instruction

REG1      =    0x33,  
REG2      =    0x33

<b>MOVLB</b>	<b>Move literal to low nibble in BSR</b>		
Syntax:	[ <i>label</i> ] MOVLB <i>k</i>		
Operands:	$0 \leq k \leq 255$		
Operation:	<i>k</i> $\rightarrow$ BSR		
Status Affected:	None		
Encoding:	0000      0001      kkkk      kkkk		
Description:	The 8-bit literal 'k' is loaded into the Bank Select Register (BSR).		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR

Example:      MOVLB    5

Before Instruction  
BSR register = 0x02  
After Instruction  
BSR register = 0x05

**MOVLW      Move literal to W**

Syntax:	[ <i>label</i> ] MOVLW <i>k</i>			
Operands:	0 ≤ <i>k</i> ≤ 255			
Operation:	<i>k</i> → W			
Status Affected:	None			
Encoding:	0000	1110	kkkk	kkkk
Description:	The eight-bit literal ' <i>k</i> ' is loaded into W.			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
	Q1	Q2	Q3	Q4
	Decode	Read literal ' <i>k</i> '	Process Data	Write to W

Example:      MOVLW      0x5A

After Instruction

$$W = 0x5A$$

**MOVWF      Move W to f**

Syntax:	[ <i>label</i> ] MOVWF <i>f</i> [, <i>a</i> ]			
Operands:	0 ≤ <i>f</i> ≤ 255 <i>a</i> ∈ [0,1]			
Operation:	(W) → <i>f</i>			
Status Affected:	None			
Encoding:	0110	111a	ffff	ffff
Description:	Move data from W to register ' <i>f</i> '. Location ' <i>f</i> ' can be anywhere in the 256 byte bank. If ' <i>a</i> ' is 0, the Access Bank will be selected, overriding the BSR value. If ' <i>a</i> ' = 1, then the bank will be selected as per the BSR value (default).			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
	Q1	Q2	Q3	Q4
	Decode	Read register ' <i>f</i> '	Process Data	Write register ' <i>f</i> '

Example:      MOVWF      REG, 0

Before Instruction

$$\begin{aligned} W &= 0x4F \\ \text{REG} &= 0xFF \end{aligned}$$

After Instruction

$$\begin{aligned} W &= 0x4F \\ \text{REG} &= 0x4F \end{aligned}$$

# PIC18FXX2

---

MULLW	Multiply Literal with W											
Syntax:	[label]	MULLW	k									
Operands:	0 ≤ k ≤ 255											
Operation:	(W) x k → PRODH:PRODL											
Status Affected:	None											
Encoding:	0000	1101	kkkk	kkkk								
Description:	An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<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 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 0xC4

Before Instruction

W	=	0xE2
PRODH	=	?
PRODL	=	?

After Instruction

W	=	0xE2
PRODH	=	0xAD
PRODL	=	0x08

MULWF	Multiply W with f											
Syntax:	[label]	MULWF	f	[a]								
Operands:	0 ≤ f ≤ 255											
a ∈ [0,1]												
Operation:	(W) x (f) → PRODH:PRODL											
Status Affected:	None											
Encoding:	0000	001a	ffff	ffff								
Description:	An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<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 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	=	0xC4
REG	=	0xB5
PRODH	=	?
PRODL	=	?

After Instruction

W	=	0xC4
REG	=	0xB5
PRODH	=	0x8A
PRODL	=	0x94

<b>NEGF</b>	<b>Negate f</b>								
Syntax:	[ <i>label</i> ] NEGF f [,a]								
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$								
Operation:	$(\bar{f}) + 1 \rightarrow f$								
Status Affected:	N, OV, C, DC, Z								
Encoding:	<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:	Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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:      NEGREG, 1

Before Instruction

REG = 0011 1010 [0x3A]

After Instruction

REG = 1100 0110 [0xC6]

<b>NOP</b>	<b>No Operation</b>								
Syntax:	[ <i>label</i> ] 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.

# PIC18FXX2

---

<b>POP</b>		<b>Pop Top of Return Stack</b>					
Syntax:	[ <i>label</i> ] 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:	The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.						
Words:	1						
Cycles:	1						
Q Cycle Activity:							
	Q1	Q2	Q3	Q4			
	Decode	No operation	POP TOS value	No operation			

<u>Example:</u>	POP	
	GOTO	NEW
<b>Before Instruction</b>		
TOS Stack (1 level down)		
	= 0031A2h	
	= 014332h	
<b>After Instruction</b>		
TOS	= 014332h	
PC	= NEW	

<b>PUSH</b>		<b>Push Top of Return Stack</b>					
Syntax:	[ <i>label</i> ] 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:	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 to implement a software stack by modifying TOS, and then push it onto the return stack.						
Words:	1						
Cycles:	1						
Q Cycle Activity:							
	Q1	Q2	Q3	Q4			
	Decode	PUSH PC+2 onto return stack	No operation	No operation			

<u>Example:</u>	PUSH	
<b>Before Instruction</b>		
TOS PC		
	= 00345Ah	
	= 000124h	
<b>After Instruction</b>		
PC	= 000126h	
TOS	= 000126h	
Stack (1 level down)	= 00345Ah	

<b>RCALL</b>	<b>Relative Call</b>												
Syntax:	[ <i>label</i> ] RCALL n												
Operands:	-1024 ≤ n ≤ 1023												
Operation:	(PC) + 2 → TOS, (PC) + 2 + 2n → 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="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Start reset</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Start reset	No operation	No operation	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Start reset	No operation	No operation										
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)

<b>RESET</b>	<b>Reset</b>								
Syntax:	[ <i>label</i> ] 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="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Start reset</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">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

# PIC18FXX2

---

RETFIE	Return from Interrupt				
Syntax:	[ <i>label</i> ] RETFIE [s]				
Operands:	s ∈ [0,1]				
Operation:	(TOS) → PC, 1 → GIE/GIEH or PEIE/GIEL, if s = 1 (WS) → W, (STATUS) → STATUS, (BSRS) → BSR, PCLATU, PCLATH are unchanged.				
Status Affected:	GIE/GIEH, PEIE/GIEL.				
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:

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:	[ <i>label</i> ] RETLW k												
Operands:	0 ≤ k ≤ 255												
Operation:	k → W, (TOS) → 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:	W is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.												
Words:	1												
Cycles:	2												
Q Cycle Activity:	<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 = 0x07

After Instruction

W = value of kn

RETURN	Return from Subroutine												
Syntax:	[ <i>label</i> ] RETURN [s]												
Operands:	s ∈ [0,1]												
Operation:	(TOS) → PC, if s = 1 (WS) → W, (STATUSS) → STATUS, (BSRS) → BSR, PCLATU, PCLATH are unchanged												
Status Affected:	None												
Encoding:	0000 0000 0001 001s												
Description:	Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers occurs (default).												
Words:	1												
Cycles:	2												
Q Cycle Activity:													
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>No operation</td><td>Process Data</td><td>pop PC from stack</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	No operation	Process Data	pop PC from stack	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	No operation	Process Data	pop PC from stack										
No operation	No operation	No operation	No operation										

Example: RETURN

After Interrupt  
PC = TOS

RLCF	Rotate Left f through Carry								
Syntax:	[ <i>label</i> ] RLCF f [d,a]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f<n>) → dest<n+1>, (f<7>) → C, (C) → dest<0>								
Status Affected:	C, N, Z								
Encoding:	0011 01da ffff ffff								
Description:	The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: RLCF REG, 0, 0

Before Instruction

REG = 1110 0110  
C = 0

After Instruction

REG = 1110 0110  
W = 1100 1100  
C = 1

# PIC18FXX2

---

RLNCF	Rotate Left f (no carry)								
Syntax:	[ label ] RLNCF f [,d [,a]]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f<n>) → dest<n+1>, (f<7>) → dest<0>								
Status Affected:	N, Z								
Encoding:	0100 01da ffff ffff								
Description:	The contents of register 'f' are rotated one bit to the left. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
									
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">Q1</th> <th style="width: 25%;">Q2</th> <th style="width: 25%;">Q3</th> <th style="width: 25%;">Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

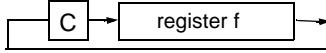
Example:      RLNCF      REG, 1, 0

Before Instruction

REG = 1010 1011

After Instruction

REG = 0101 0111

RRCF	Rotate Right f through Carry								
Syntax:	[ label ] RRCF f [,d [,a]]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f<n>) → dest<n-1>, (f<0>) → C, (C) → dest<7>								
Status Affected:	C, N, Z								
Encoding:	0011 00da ffff ffff								
Description:	The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
									
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 25%;">Q1</th> <th style="width: 25%;">Q2</th> <th style="width: 25%;">Q3</th> <th style="width: 25%;">Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example:      RRCF      REG, 0, 0

Before Instruction

REG = 1110 0110  
C = 0

After Instruction

REG = 1110 0110  
W = 0111 0011  
C = 0

RRNCF	Rotate Right f (no carry)								
Syntax:	[ label ] RRNCF f [,d [,a]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f<n>) → dest<n-1> (f<0>) → dest<7>								
Status Affected:	N, Z								
Encoding:	0100 00da ffff ffff								
Description:	The contents of register 'f' are rotated one bit to the right. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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 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'						

SETF	Set f								
Syntax:	[ label ] SETF f [,a]								
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]								
Operation:	FFh → f								
Status Affected:	None								
Encoding:	0110 100a ffff ffff								
Description:	The contents of the specified register are set to FFh. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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 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: SETF REG, 1

Before Instruction			
REG = 0x5A			
After Instruction			
REG = 0xFF			

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

# PIC18FXX2

---

SLEEP	Enter SLEEP mode								
Syntax:	[ <i>label</i> ] SLEEP								
Operands:	None								
Operation:	00h → WDT, 0 → WDT postscaler, 1 → <u>TO</u> , 0 → <u>PD</u>								
Status Affected:	<u>TO</u> , <u>PD</u>								
Encoding:	<table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0011</td></tr></table>	0000	0000	0000	0011				
0000	0000	0000	0011						
Description:	The power-down status bit (PD) is cleared. The time-out status bit (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

TO = ?  
PD = ?

After Instruction

TO = 1 †  
PD = 0

† If WDT causes wake-up, this bit is cleared.

SUBFWB	Subtract f from W with borrow								
Syntax:	[ <i>label</i> ] SUBFWB f [,d [,a]]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(W) – (f) – ( <u>C</u> ) → dest								
Status Affected:	N, OV, C, DC, Z								
Encoding:	<table border="1"><tr><td>0101</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>	0101	01da	ffff	ffff				
0101	01da	ffff	ffff						
Description:	Subtract register 'f' and carry flag (borrow) from W (2's complement method). If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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 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:	[label] SUBLW k				
Operands:	0 ≤ k ≤ 255				
Operation:	k - (W) → W				
Status Affected:	N, OV, C, DC, Z				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>1000</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1000	kkkk	kkkk
0000	1000	kkkk	kkkk		
Description:	W is subtracted from the eight-bit literal 'k'. The result is placed in W.				
Words:	1				
Cycles:	1				
Q Cycle Activity:					
Q1            Q2            Q3            Q4					
Decode      Read literal 'k'      Process Data      Write to W					

Example 1:      SUBLW 0x02

Before Instruction

W	=	1
C	=	?

After Instruction

W	=	1
C	=	1
Z	=	0
N	=	0

; result is positive

Example 2:      SUBLW 0x02

Before Instruction

W	=	2
C	=	?

After Instruction

W	=	0
C	=	1
Z	=	1
N	=	0

; result is zero

Example 3:      SUBLW 0x02

Before Instruction

W	=	3
C	=	?

After Instruction

W	=	FF
C	=	0
Z	=	0
N	=	1

; (2's complement)  
; result is negative

<b>SUBWF</b>	<b>Subtract W from f</b>				
Syntax:	[label] SUBWF f [,d [,a]]				
Operands:	0 ≤ f ≤ 255				
	d ∈ [0,1]				
	a ∈ [0,1]				
Operation:	(f) - (W) → dest				
Status Affected:	N, OV, C, DC, Z				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0101</td><td>11da</td><td>ffff</td><td>ffff</td></tr> </table>	0101	11da	ffff	ffff
0101	11da	ffff	ffff		
Description:	Subtract W from register 'f' (2's complement method). If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).				
Words:	1				
Cycles:	1				
Q Cycle Activity:					
Q1            Q2            Q3            Q4					
Decode      Read register 'f'      Process Data      Write to destination					

Example 1:      SUBWF REG, 1, 0

Before Instruction

REG	=	3
W	=	2
C	=	?

After Instruction

REG	=	1
W	=	2
C	=	1
Z	=	0
N	=	0

; result is positive

Example 2:      SUBWF REG, 0, 0

Before Instruction

REG	=	2
W	=	2
C	=	?

After Instruction

REG	=	2
W	=	0
C	=	1
Z	=	1
N	=	0

; result is zero

Example 3:      SUBWF REG, 1, 0

Before Instruction

REG	=	1
W	=	2
C	=	?

After Instruction

REG	=	FFh
W	=	2
C	=	0
Z	=	0
N	=	1

; (2's complement)  
; result is negative

# PIC18FXX2

---

SUBWFB	Subtract W from f with Borrow								
Syntax:	[ <i>label</i> ] SUBWFB f [,d [,a]]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f) – (W) – ( $\bar{C}$ ) → dest								
Status Affected:	N, OV, C, DC, Z								
Encoding:	<table border="1"><tr><td>0101</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0101	10da	ffff	ffff				
0101	10da	ffff	ffff						
Description:	Subtract W and the carry flag (borrow) from register 'f' (2's complement method). If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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 1: SUBWFB REG, 1, 0

Before Instruction

REG	=	0x19	(0001 1001)
W	=	0x0D	(0000 1101)
C	=	1	

After Instruction

REG	=	0x0C	(0000 1011)
W	=	0x0D	(0000 1101)
C	=	1	
Z	=	0	
N	=	0	; result is positive

Example 2: SUBWFB REG, 0, 0

Before Instruction

REG	=	0x1B	(0001 1011)
W	=	0x1A	(0001 1010)
C	=	0	

After Instruction

REG	=	0x1B	(0001 1011)
W	=	0x00	
C	=	1	
Z	=	1	; result is zero
N	=	0	

Example 3: SUBWFB REG, 1, 0

Before Instruction

REG	=	0x03	(0000 0011)
W	=	0x0E	(0000 1101)
C	=	1	

After Instruction

REG	=	0xF5	(1111 0100) ; [2's comp]
W	=	0x0E	(0000 1101)
C	=	0	
Z	=	0	
N	=	1	; result is negative

SWAPF	Swap f								
Syntax:	[ <i>label</i> ] SWAPF f [,d [,a]]								
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]								
Operation:	(f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0>								
Status Affected:	None								
Encoding:	<table border="1"><tr><td>0011</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0011	10da	ffff	ffff				
0011	10da	ffff	ffff						
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<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: SWAPF REG, 1, 0

Before Instruction

REG	=	0x53
-----	---	------

After Instruction

REG	=	0x35
-----	---	------

TBLRD	Table Read				
Syntax:	[ <i>label</i> ] TBLRD ( *; *+; *-; +* )				
Operands:	None				
Operation:	if TBLRD *, (Prog Mem (TBLPTR)) → TABLAT; TBLPTR - No Change; if TBLRD *+, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) +1 → TBLPTR; if TBLRD *-, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) -1 → TBLPTR; if TBLRD +*, (TBLPTR) +1 → TBLPTR; (Prog Mem (TBLPTR)) → TABLAT;				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>10nn 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. The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2 Mbyte address range.</p> <p style="margin-left: 20px;">TBLPTR[0] = 0: Least Significant Byte of Program Memory Word</p> <p style="margin-left: 20px;">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 (cont'd)
Example1:	TBLRD *+ ;
Before Instruction	
TABLAT	= 0x55
TBLPTR	= 0x00A356
MEMORY(0x00A356)	= 0x34
After Instruction	
TABLAT	= 0x34
TBLPTR	= 0x00A357
Example2:	TBLRD +* ;
Before Instruction	
TABLAT	= 0xAA
TBLPTR	= 0x01A357
MEMORY(0x01A357)	= 0x12
MEMORY(0x01A358)	= 0x34
After Instruction	
TABLAT	= 0x34
TBLPTR	= 0x01A358

# PIC18FXX2

---

TBLWT	Table Write												
Syntax:	[ <i>label</i> ] 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" style="display: inline-table; vertical-align: middle;"> <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 the TBLPTR to determine which of the 8 holding registers the TABLAT data is written to. The 8 holding registers are used to program the contents of Program Memory (P.M.). See Section 5.0 for information on writing to FLASH memory.</p> <p>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 style="margin-left: 40px;">TBLPTR[0] = 0: Least Significant Byte of Program Memory Word</p> <p style="margin-left: 40px;">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" 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>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 or Memory)</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register or Memory)				
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 or Memory)										

TBLWT	Table Write (Continued)
Example1:	TBLWT *+;
Before Instruction	
TABLAT	= 0x55
TBLPTR	= 0x00A356
HOLDING REGISTER (0x00A356)	= 0xFF
After Instructions (table write completion)	
TABLAT	= 0x55
TBLPTR	= 0x00A357
HOLDING REGISTER (0x00A356)	= 0x55
Example 2:	TBLWT +*;
Before Instruction	
TABLAT	= 0x34
TBLPTR	= 0x01389A
HOLDING REGISTER (0x01389A)	= 0xFF
HOLDING REGISTER (0x01389B)	= 0xFF
After Instruction (table write completion)	
TABLAT	= 0x34
TBLPTR	= 0x01389B
HOLDING REGISTER (0x01389A)	= 0xFF
HOLDING REGISTER (0x01389B)	= 0x34

TSTFSZ	Test f, skip if 0								
Syntax:	[ <i>label</i> ] 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:	If 'f' = 0, the next instruction, fetched during the current instruction execution, is discarded and a NOP is executed, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1(2) <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.								
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>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						

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 = 0x00,  
PC = Address (ZERO)  
If CNT ≠ 0x00,  
PC = Address (NZERO)

XORLW	Exclusive OR literal with W								
Syntax:	[ <i>label</i> ] XORLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	(W) .XOR. k → 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 0xAF

Before Instruction

W = 0xB5

After Instruction

W = 0x1A

# PIC18FXX2

---

## XORWF      Exclusive OR W with f

Syntax:	[ <i>label</i> ] XORWF f [,d [,a]								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	(W) .XOR. (f) $\rightarrow$ dest								
Status Affected:	N, Z								
Encoding:	<table border="1"><tr><td>0001</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0001	10da	ffff	ffff				
0001	10da	ffff	ffff						
Description:	Exclusive OR the contents of W with register 'f'. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in the register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"><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						

Example:      XORWF REG, 1, 0

Before Instruction

REG = 0xAF  
W = 0xB5

After Instruction

REG = 0x1A  
W = 0xB5

## 21.0 DEVELOPMENT SUPPORT

The PICmicro® 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 C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/  
MPLIB™ Object Librarian
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Entry-Level Development  
Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM 2 Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 17 Demonstration Board
  - KEELoQ® Demonstration Board

### 21.1 MPLAB Integrated Development Environment Software

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

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or 'C')
- One touch assemble (or compile) and download to PICmicro emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

### 21.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PICmicro MCU's.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. 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, an absolute LST file that contains source lines and generated machine code, and a COD file 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.

### 21.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI 'C' compilers for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

## 21.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can also link relocatable objects from pre-compiled libraries, using directives from a linker script.

The MPLIB object librarian is a librarian for pre-compiled code to be used with the MPLINK object linker. When a routine from a library is called from another 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 MPLIB object librarian manages the creation and modification of library files.

The MPLINK object linker features include:

- Integration with MPASM assembler and MPLAB C17 and MPLAB C18 C compilers.
- Allows all memory areas to be defined as sections to provide link-time flexibility.

The MPLIB object librarian features include:

- Easier linking because single libraries can be included instead of many smaller files.
- Helps keep code maintainable by grouping related modules together.
- Allows libraries to be created and modules to be added, listed, replaced, deleted or extracted.

## 21.5 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC-hosted environment by simulating the PICmicro series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user-defined key press, to any of the pins. The execution can be performed in single step, execute until break, or trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and the MPLAB C18 C compilers and the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent multi-project software development tool.

## 21.6 MPLAB ICE High Performance Universal In-Circuit Emulator with MPLAB IDE

The MPLAB ICE universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PICmicro microcontrollers (MCUs). Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment (IDE), 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 universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PICmicro microcontrollers.

The MPLAB ICE in-circuit emulator system has been designed as a real-time emulation system, with advanced features that are generally found on more expensive development tools. The PC platform and Microsoft® Windows environment were chosen to best make these features available to you, the end user.

## 21.7 ICEPIC In-Circuit Emulator

The ICEPIC low cost, in-circuit emulator is a solution for the Microchip Technology PIC16C5X, PIC16C6X, PIC16C7X and PIC16CXXX families of 8-bit One-Time-Programmable (OTP) microcontrollers. The modular system can support different subsets of PIC16C5X or PIC16CXXX products through the use of interchangeable personality modules, or daughter boards. The emulator is capable of emulating without target application circuitry being present.

## 21.8 MPLAB ICD In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD, is a powerful, low cost, run-time development tool. This tool is based on the FLASH PICmicro MCUs and can be used to develop for this and other PICmicro microcontrollers. The MPLAB ICD utilizes the in-circuit debugging capability built into the FLASH devices. This feature, along with Microchip's In-Circuit Serial Programming™ 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 watching variables, single-stepping and setting break points. Running at full speed enables testing hardware in real-time.

## 21.9 PRO MATE II Universal Device Programmer

The PRO MATE II universal device programmer is a full-featured programmer, capable of operating in stand-alone mode, as well as PC-hosted mode. The PRO MATE II device programmer is CE compliant.

The PRO MATE II device programmer has programmable VDD and VPP supplies, which allow it to verify programmed memory at VDD min and VDD max for maximum reliability. It has an LCD display for instructions and error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode, the PRO MATE II device programmer can read, verify, or program PICmicro devices. It can also set code protection in this mode.

## 21.10 PICSTART Plus Entry Level 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 all PICmicro devices with 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.

## 21.11 PICDEM 1 Low Cost PICmicro Demonstration Board

The PICDEM 1 demonstration board is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The user can program the sample microcontrollers provided with the PICDEM 1 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The user can also connect the PICDEM 1 demonstration board to the MPLAB ICE in-circuit emulator and download the firmware to the emulator for testing. A prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push button switches and eight LEDs connected to PORTB.

## 21.12 PICDEM 2 Low Cost PIC16CXX Demonstration Board

The PICDEM 2 demonstration board is a simple demonstration board that supports the PIC16C62, PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM 2 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 2 demonstration board to test firmware. A prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push button switches, a potentiometer for simulated analog input, a serial EEPROM to demonstrate usage of the I<sup>2</sup>C™ bus and separate headers for connection to an LCD module and a keypad.

## 21.13 PICDEM 3 Low Cost PIC16CXXX Demonstration Board

The PICDEM 3 demonstration board is a simple demonstration board that supports the PIC16C923 and PIC16C924 in the PLCC package. It will also support future 44-pin PLCC microcontrollers with an LCD Module. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM 3 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer with an adapter socket, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 3 demonstration board to test firmware. A prototype area has been provided to the user for adding hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push button switches, a potentiometer for simulated analog input, a thermistor and separate headers for connection to an external LCD module and a keypad. Also provided on the PICDEM 3 demonstration board is a LCD panel, with 4 commons and 12 segments, that is capable of displaying time, temperature and day of the week. The PICDEM 3 demonstration board provides an additional RS-232 interface and Windows software for showing the demultiplexed LCD signals on a PC. A simple serial interface allows the user to construct a hardware demultiplexer for the LCD signals.

## 21.14 PICDEM 17 Demonstration Board

The PICDEM 17 demonstration board is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756A, PIC17C762 and PIC17C766. All necessary hardware is included to run basic demo programs, which are supplied on a 3.5-inch disk. A programmed sample is included and the user may erase it and program it with the other sample programs using the PRO MATE II device programmer, or the PICSTART Plus development programmer, and easily debug and test the sample code. In addition, the PICDEM 17 demonstration board supports downloading of programs to and executing out of external FLASH memory on board. The PICDEM 17 demonstration board is also usable with the MPLAB ICE in-circuit emulator, or the PICMASTER emulator and all of the sample programs can be run and modified using either emulator. Additionally, a generous prototype area is available for user hardware.

## 21.15 KEELOQ Evaluation and Programming Tools

KEELOQ evaluation and programming tools support Microchip's HCS Secure Data Products. The HCS evaluation kit includes a LCD display to show changing codes, a decoder to decode transmissions and a programming interface to program test transmitters.

**TABLE 21-1: DEVELOPMENT TOOLS FROM MICROCHIP**

\* Contact the Microchip Technology Inc. web site at [www.microchip.com](http://www.microchip.com) for information on how to use the MPLAB® ICID In-Circuit Debugger (DV164001) with PIC16C62.

\*\*\* Contact Microchip Technology Inc. for availability date.

<sup>†</sup> Donations and grants in millions of dollars.

# **PIC18FXX2**

---

---

## **NOTES:**

## 22.0 ELECTRICAL CHARACTERISTICS

### Absolute Maximum Ratings <sup>(†)</sup>

Ambient temperature under bias.....	-55°C to +125°C
Storage temperature .....	-65°C to +150°C
Voltage on any pin with respect to Vss (except VDD, <u>MCLR</u> , and RA4) .....	-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
Voltage on RA4 with respect to Vss .....	0V to +8.5V
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>OL</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 PORTA, PORTB, and PORTE ( <b>Note 3</b> ) (combined) .....	200 mA
Maximum current sourced by PORTA, PORTB, and PORTE ( <b>Note 3</b> ) (combined).....	200 mA
Maximum current sunk by PORTC and PORTD ( <b>Note 3</b> ) (combined).....	200 mA
Maximum current sourced by PORTC and PORTD ( <b>Note 3</b> ) (combined).....	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 pin, inducing currents greater than 80 mA, may cause latchup. Thus, a series resistor of 50-100Ω should be used when applying a “low” level to the MCLR/VPP pin, rather than pulling this pin directly to Vss.

**3:** PORTD and PORTE not available on the PIC18F2X2 devices.

<sup>†</sup> 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.

# PIC18FXX2

FIGURE 22-1: PIC18FXX2 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

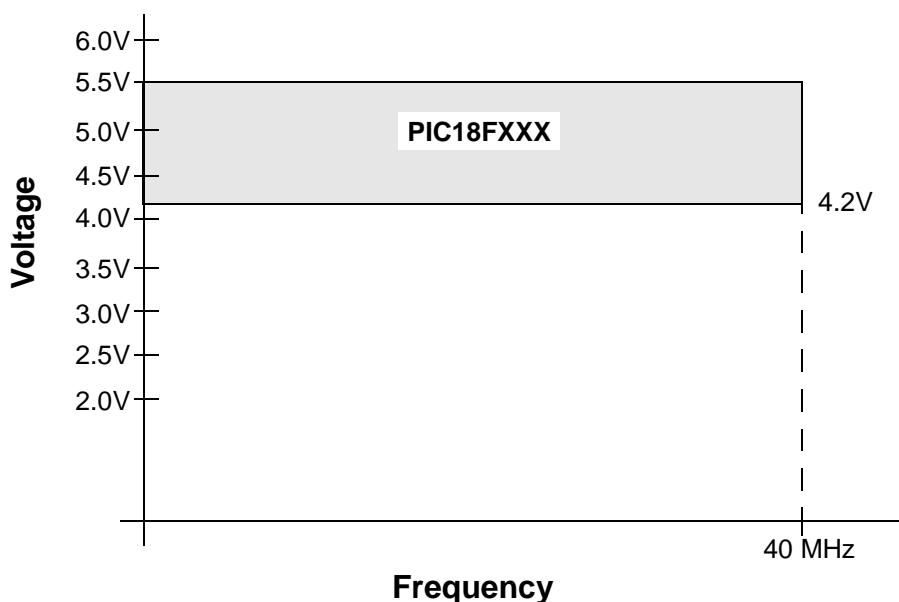
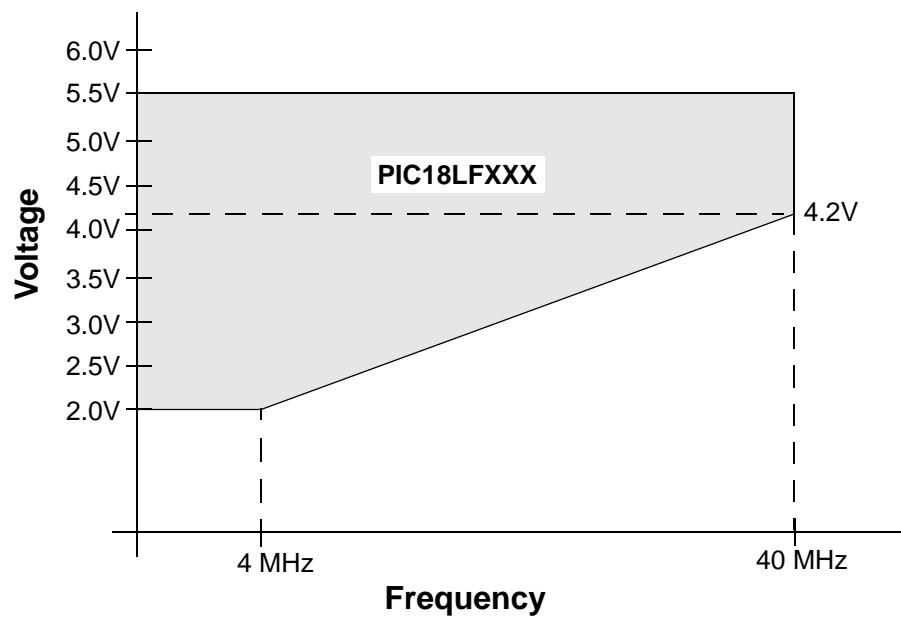


FIGURE 22-2: PIC18LFXX2 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



$$F_{MAX} = (16.36 \text{ MHz/V}) (V_{DDAPPMIN} - 2.0V) + 4 \text{ MHz}$$

**Note:**  $V_{DDAPPMIN}$  is the minimum voltage of the PICmicro® device in the application.

## 22.1 DC Characteristics: PIC18FXX2 (Industrial, Extended) PIC18LFXX2 (Industrial)

<b>PIC18LFXX2 (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>PIC18FXX2 (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.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions	
			<b>Supply Voltage</b>					
D001	VDD	PIC18LFXX2	2.0	—	5.5	V	HS, XT, RC and LP Osc mode	
D001		PIC18FXX2	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 3.1 (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 3.1 (Power-on Reset) for details	
			<b>Brown-out Reset Voltage</b>					
D005	VBOR	PIC18LFXX2						
		BORV1:BORV0 = 11	1.98	—	2.14	V	$85^{\circ}\text{C} \geq \text{T} \geq 25^{\circ}\text{C}$	
		BORV1:BORV0 = 10	2.67	—	2.89	V		
		BORV1:BORV0 = 01	4.16	—	4.5	V		
		BORV1:BORV0 = 00	4.45	—	4.83	V		
D005		PIC18FXX2						
		BORV1:BORV0 = 1x	N.A.	—	N.A.	V	Not in operating voltage range of device	
		BORV1:BORV0 = 01	4.16	—	4.5	V		
		BORV1:BORV0 = 00	4.45	—	4.83	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:** The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.  
The test conditions for all IDD measurements in active Operation mode are:  
 OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD  
 MCLR = Vdd; WDT enabled/disabled as specified.
- 3:** 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 hi-impedance state and tied to VDD or Vss, and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR,...).
- 4:** For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula  $I_r = V_{DD}/2REXT$  (mA) with REXT in kOhm.
- 5:** The LVD and BOR modules share a large portion of circuitry. The  $\Delta I_{BOR}$  and  $\Delta I_{LVD}$  currents are not additive. Once one of these modules is enabled, the other may also be enabled without further penalty.

# PIC18FXX2

---

## 22.1 DC Characteristics: PIC18FXX2 (Industrial, Extended) PIC18LFXX2 (Industrial) (Continued)

PIC18LFXX2 (Industrial)			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial				
PIC18FXX2 (Industrial, Extended)			Standard Operating Conditions (unless otherwise stated) 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
D010	IDD	Supply Current <sup>(2,4)</sup>					
		PIC18LFXX2	—	.5	1	mA	XT osc configuration VDD = 2.0V, $+25^{\circ}\text{C}$ , FOSC = 4 MHz
			—	.5	1.25	mA	VDD = 2.0V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ , FOSC = 4 MHz
			—	1.2	2	mA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ , FOSC = 4 MHz
			—	.3	1	mA	RC osc configuration VDD = 2.0V, $+25^{\circ}\text{C}$ , FOSC = 4 MHz
			—	.3	1	mA	VDD = 2.0V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ , FOSC = 4 MHz
			—	1.5	3	mA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ , FOSC = 4 MHz
			—	.3	1	mA	RCIO osc configuration VDD = 2.0V, $+25^{\circ}\text{C}$ , FOSC = 4 MHz
			—	.3	1	mA	VDD = 2.0V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ , FOSC = 4 MHz
			—	.75	3	mA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ , FOSC = 4 MHz
D010		PIC18FXX2	—	1.2	1.5	mA	XT osc configuration VDD = 4.2V, $+25^{\circ}\text{C}$ , FOSC = 4 MHz
			—	1.2	2	mA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ , FOSC = 4 MHz
			—	1.2	3	mA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ , FOSC = 4 MHz
			—	1.5	3	mA	RC osc configuration VDD = 4.2V, $+25^{\circ}\text{C}$ , FOSC = 4 MHz
			—	1.5	4	mA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ , FOSC = 4 MHz
			—	1.6	4	mA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ , FOSC = 4 MHz
			—	.75	2	mA	RCIO osc configuration VDD = 4.2V, $+25^{\circ}\text{C}$ , FOSC = 4 MHz
			—	.75	3	mA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ , FOSC = 4 MHz
			—	.8	3	mA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$ , FOSC = 4 MHz
D010A		PIC18LFXX2	—	14	30	$\mu\text{A}$	LP osc, FOSC = 32 kHz, WDT disabled VDD = 2.0V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D010A		PIC18FXX2	—	40	70	$\mu\text{A}$	LP osc, FOSC = 32 kHz, WDT disabled VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	50	100	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$

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:** The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active Operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD  
MCLR = VDD; WDT enabled/disabled as specified.

- 3: 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 hi-impedance state and tied to VDD or VSS, and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR,...).
- 4: For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula  $I_R = VDD/2REXT$  (mA) with REXT in kOhm.
- 5: The LVD and BOR modules share a large portion of circuitry. The  $\Delta I_{BOR}$  and  $\Delta I_{LVD}$  currents are not additive. Once one of these modules is enabled, the other may also be enabled without further penalty.

## 22.1 DC Characteristics: PIC18FXX2 (Industrial, Extended) PIC18LFXX2 (Industrial) (Continued)

PIC18LFXX2 (Industrial)			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial					
PIC18FXX2 (Industrial, Extended)			Standard Operating Conditions (unless otherwise stated) 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	
D010C	IDD	<b>Supply Current<sup>(2,4)</sup> (Continued)</b>						
		PIC18LFXX2	—	10	25	mA	EC, ECIO osc configurations VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	
D010C		PIC18FXX2	—	10	25	mA	EC, ECIO osc configurations VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$	
D013		PIC18LFXX2	—	.6	2	mA	HS osc configuration Fosc = 4 MHz, VDD = 2.0V	
			—	10	15	mA	Fosc = 25 MHz, VDD = 5.5V	
			—	15	25	mA	HS + PLL osc configurations Fosc = 10 MHz, VDD = 5.5V	
D013		PIC18FXX2	—	10	15	mA	HS osc configuration Fosc = 25 MHz, VDD = 5.5V	
			—	15	25	mA	HS + PLL osc configurations Fosc = 10 MHz, VDD = 5.5V	
D014		PIC18LFXX2	—	15	55	μA	Timer1 osc configuration Fosc = 32 kHz, VDD = 2.0V	
D014		PIC18FXX2	—	—	200	μA	Timer1 osc configuration Fosc = 32 kHz, VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	
			—	—	250	μA	Fosc = 32 kHz, VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$	
D020	IPD	<b>Power-down Current<sup>(3)</sup></b>						
		PIC18LFXX2	—	.08	.9	μA	VDD = 2.0V, $+25^{\circ}\text{C}$	
D020			—	.1	4	μA	VDD = 2.0V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	
			—	3	10	μA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	
D021B		PIC18FXX2	—	.1	.9	μA	VDD = 4.2V, $+25^{\circ}\text{C}$	
			—	3	10	μA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$	
			—	15	25	μA	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$	

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:** The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.
- The test conditions for all IDD measurements in active Operation mode are:
- OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD
  - MCLR = VDD; WDT enabled/disabled as specified.
- 3:** 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 hi-impedance state and tied to VDD or VSS, and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR,...).
- 4:** For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula  $I_r = VDD/2REXT$  (mA) with REXT in kOhm.
- 5:** The LVD and BOR modules share a large portion of circuitry. The  $\Delta I_{BOR}$  and  $\Delta I_{LVD}$  currents are not additive. Once one of these modules is enabled, the other may also be enabled without further penalty.

# PIC18FXX2

---

## 22.1 DC Characteristics: PIC18FXX2 (Industrial, Extended) PIC18LFXX2 (Industrial) (Continued)

PIC18LFXX2 (Industrial)			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial				
PIC18FXX2 (Industrial, Extended)			Standard Operating Conditions (unless otherwise stated) 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
D022	$\Delta I_{WDT}$	<b>Module Differential Current</b>					
		<b>Watchdog Timer</b> PIC18LFXX2	—	.75	1.5	$\mu\text{A}$	VDD = 2.0V, $+25^{\circ}\text{C}$
			—	2	8	$\mu\text{A}$	VDD = 2.0V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D022		<b>Watchdog Timer</b> PIC18FXX2	—	10	25	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	7	15	$\mu\text{A}$	VDD = 4.2V, $+25^{\circ}\text{C}$
			—	10	25	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D022A	$\Delta I_{BOR}$	<b>Brown-out Reset<sup>(5)</sup></b> PIC18LFXX2	—	29	35	$\mu\text{A}$	VDD = 2.0V, $+25^{\circ}\text{C}$
			—	29	45	$\mu\text{A}$	VDD = 2.0V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	33	50	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D022A		<b>Brown-out Reset<sup>(5)</sup></b> PIC18FXX2	—	36	40	$\mu\text{A}$	VDD = 4.2V, $+25^{\circ}\text{C}$
			—	36	50	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	36	65	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$
D022B	$\Delta I_{LVD}$	<b>Low Voltage Detect<sup>(5)</sup></b> PIC18LFXX2	—	29	35	$\mu\text{A}$	VDD = 2.0V, $+25^{\circ}\text{C}$
			—	29	45	$\mu\text{A}$	VDD = 2.0V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	33	50	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D022B		<b>Low Voltage Detect<sup>(5)</sup></b> PIC18FXX2	—	33	40	$\mu\text{A}$	VDD = 4.2V, $+25^{\circ}\text{C}$
			—	33	50	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	33	65	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$
D025	$\Delta I_{TMR1}$	<b>Timer1 Oscillator</b> PIC18LFXX2	—	5.2	30	$\mu\text{A}$	VDD = 2.0V, $+25^{\circ}\text{C}$
			—	5.2	40	$\mu\text{A}$	VDD = 2.0V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	6.5	50	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
D025		<b>Timer1 Oscillator</b> PIC18FXX2	—	6.5	40	$\mu\text{A}$	VDD = 4.2V, $+25^{\circ}\text{C}$
			—	6.5	50	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	6.5	65	$\mu\text{A}$	VDD = 4.2V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$

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:** The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

The test conditions for all IDD measurements in active Operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD  
MCLR = VDD; WDT enabled/disabled as specified.

- 3:** 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 hi-impedance state and tied to VDD or Vss, and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR,...).
- 4:** For RC osc configuration, current through REXT is not included. The current through the resistor can be estimated by the formula  $I_R = VDD/2REXT$  (mA) with REXT in kOhm.
- 5:** The LVD and BOR modules share a large portion of circuitry. The  $\Delta I_{BOR}$  and  $\Delta I_{LVD}$  currents are not additive. Once one of these modules is enabled, the other may also be enabled without further penalty.

## 22.2 DC Characteristics: PIC18FXX2 (Industrial, Extended) PIC18LFXX2 (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated)			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D030 D030A D031 D032 D032A D033	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 ≤ VDD ≤ 5.5V
		with Schmitt Trigger buffer RC3 and RC4	Vss	0.2 VDD	V	
			Vss	0.3 VDD	V	
		<u>MCLR</u>	Vss	0.2 VDD	V	
		OSC1 (in XT, HS and LP modes) and T1OSI	Vss	0.3 VDD	V	
D033		OSC1 (in RC and EC mode) <sup>(1)</sup>	Vss	0.2 VDD	V	
D040 D040A D041 D042 D042A D043	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 ≤ VDD ≤ 5.5V
		with Schmitt Trigger buffer RC3 and RC4	0.8 VDD	VDD	V	
			0.7 VDD	VDD	V	
		<u>MCLR</u> , OSC1 (EC mode)	0.8 VDD	VDD	V	
		OSC1 (in XT, HS and LP modes) and T1OSI	0.7 VDD	VDD	V	
D043		OSC1 (RC mode) <sup>(1)</sup>	0.9 VDD	VDD	V	
D060 D061 D063	IIL	<b>Input Leakage Current<sup>(2,3)</sup></b> I/O ports	.02	±1	µA	Vss ≤ VPIN ≤ VDD, Pin at hi-impedance
		<u>MCLR</u>	—	±1	µA	Vss ≤ VPIN ≤ VDD
		OSC1	—	±1	µA	Vss ≤ VPIN ≤ VDD
D070	IPU IPURB	<b>Weak Pull-up Current</b> PORTB weak pull-up current	50	450	µA	VDD = 5V, VPIN = Vss

**Note 1:** In RC oscillator configuration, the OSC1/CLK1 pin is a Schmitt Trigger input. It is not recommended that the PICmicro 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.
- 4:** Parameter is characterized but not tested.

# PIC18FXX2

---

## 22.2 DC Characteristics: PIC18FXX2 (Industrial, Extended) PIC18LFXX2 (Industrial) (Continued)

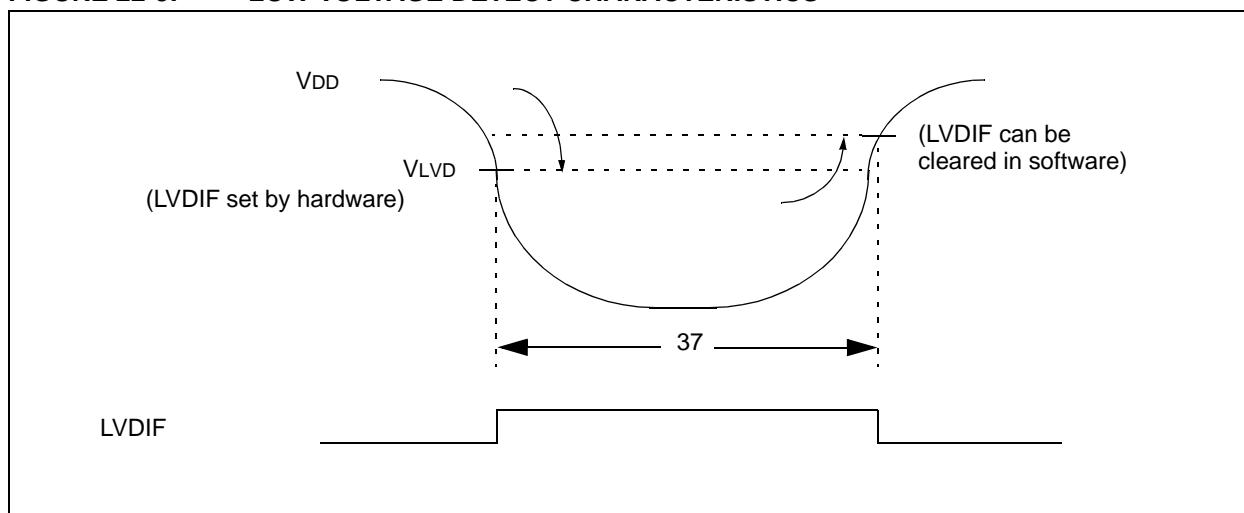
DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended			
Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
D080	V <sub>OL</sub>	<b>Output Low Voltage</b> I/O ports	—	0.6	V	I <sub>OL</sub> = 8.5 mA, V <sub>DD</sub> = 4.5V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	0.6	V	I <sub>OL</sub> = 7.0 mA, V <sub>DD</sub> = 4.5V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$
			—	0.6	V	I <sub>OL</sub> = 1.6 mA, V <sub>DD</sub> = 4.5V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			—	0.6	V	I <sub>OL</sub> = 1.2 mA, V <sub>DD</sub> = 4.5V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$
D090	V <sub>OH</sub>	<b>Output High Voltage<sup>(3)</sup></b> I/O ports	V <sub>DD</sub> – 0.7	—	V	I <sub>OH</sub> = -3.0 mA, V <sub>DD</sub> = 4.5V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			V <sub>DD</sub> – 0.7	—	V	I <sub>OH</sub> = -2.5 mA, V <sub>DD</sub> = 4.5V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$
			V <sub>DD</sub> – 0.7	—	V	I <sub>OH</sub> = -1.3 mA, V <sub>DD</sub> = 4.5V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
			V <sub>DD</sub> – 0.7	—	V	I <sub>OH</sub> = -1.0 mA, V <sub>DD</sub> = 4.5V, $-40^{\circ}\text{C}$ to $+125^{\circ}\text{C}$
D150	V <sub>OD</sub>	<b>Open Drain High Voltage</b>	—	8.5	V	RA4 pin
D100 <sup>(4)</sup>	C <sub>osc2</sub>	<b>Capacitive Loading Specs on Output Pins</b> OSC2 pin	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
			—	50	pF	To meet the AC Timing Specifications
D101	C <sub>IO</sub>	All I/O pins and OSC2 (in RC mode)	—	400	pF	In I <sup>2</sup> C mode

**Note 1:** In RC oscillator configuration, the OSC1/CLK1 pin is a Schmitt Trigger input. It is not recommended that the PICmicro 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.

**4:** Parameter is characterized but not tested.

**FIGURE 22-3: LOW VOLTAGE DETECT CHARACTERISTICS****TABLE 22-1: 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 $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$ for extended					
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions	
D420	VLVD	LVD Voltage on VDD transition high to low	LVV = 0001	1.98	2.06	2.14	V	$\text{T} \geq 25^{\circ}\text{C}$
			LVV = 0010	2.18	2.27	2.36	V	$\text{T} \geq 25^{\circ}\text{C}$
			LVV = 0011	2.37	2.47	2.57	V	$\text{T} \geq 25^{\circ}\text{C}$
			LVV = 0100	2.48	2.58	2.68	V	
			LVV = 0101	2.67	2.78	2.89	V	
			LVV = 0110	2.77	2.89	3.01	V	
			LVV = 0111	2.98	3.1	3.22	V	
			LVV = 1000	3.27	3.41	3.55	V	
			LVV = 1001	3.47	3.61	3.75	V	
			LVV = 1010	3.57	3.72	3.87	V	
			LVV = 1011	3.76	3.92	4.08	V	
			LVV = 1100	3.96	4.13	4.3	V	
			LVV = 1101	4.16	4.33	4.5	V	
			LVV = 1110	4.45	4.64	4.83	V	

# PIC18FXX2

---

**TABLE 22-2: MEMORY PROGRAMMING REQUIREMENTS**

DC Characteristics			Standard Operating Conditions (unless otherwise stated)				
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D110	VPP	<b>Internal Program Memory Programming Specifications</b>	9.00	—	13.25	V	
D113	IDDP	Voltage on MCLR/VPP pin	—	—	10	mA	
D120	ED	<b>Data EEPROM Memory</b>	100K	1M	—	E/W	-40°C to +85°C
D121	VDRW	Cell Endurance 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°C to +85°C
D130	EP	<b>Program FLASH Memory</b>	10K	100K	—	E/W	-40°C to +85°C
D131	VPR	Cell Endurance VDD for Read	V <sub>MIN</sub>	—	5.5	V	V <sub>MIN</sub> = Minimum operating voltage
D132	VIE	VDD for Block Erase	4.5	—	5.5	V	Using ICSP port
D132A	VIW	VDD for Externally Timed Erase or Write	4.5	—	5.5	V	Using ICSP port
D132B	VPEW	VDD for Self-timed Write	V <sub>MIN</sub>	—	5.5	V	V <sub>MIN</sub> = Minimum operating voltage
D133	TIE	ICSP Block Erase Cycle Time	—	4	—	ms	V <sub>DD</sub> ≥ 4.5V
D133A	TIW	ICSP Erase or Write Cycle Time (externally timed)	1	—	—	ms	V <sub>DD</sub> ≥ 4.5V
D133A	TIW	Self-timed Write Cycle Time	—	2	—	ms	
D134	TRETD	Characteristic Retention	40	—	—	Year	Provided no other specifications are violated

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Refer to Section 6.8 for a more detailed discussion on data EEPROM endurance.

## 22.3 AC (Timing) Characteristics

### 22.3.1 TIMING PARAMETER SYMOLOGY

The timing parameter symbols have been created following one of the following formats:

- |             |  |
|-------------|--|
| 1. TppS2ppS | 3. Tcc:ST      ( $I^2C$ specifications only) |
| 2. TppS     | 4. Ts      ( $I^2C$ specifications only)     |

T	
F	Frequency
	T      Time

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$RD$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T1CKI
mc	MCLR	wr	$\overline{WR}$

Uppercase letters and their meanings:

S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (Hi-impedance)	V	Valid
L	Low	Z	Hi-impedance
$I^2C$ only	AA output access	High	High
		Low	Low
BUF	Bus free		

Tcc:ST ( $I^2C$  specifications only)

CC			
HD	Hold	SU	Setup
ST		STO	STOP condition
DAT	DATA input hold		
STA	START condition		

# PIC18FXX2

---

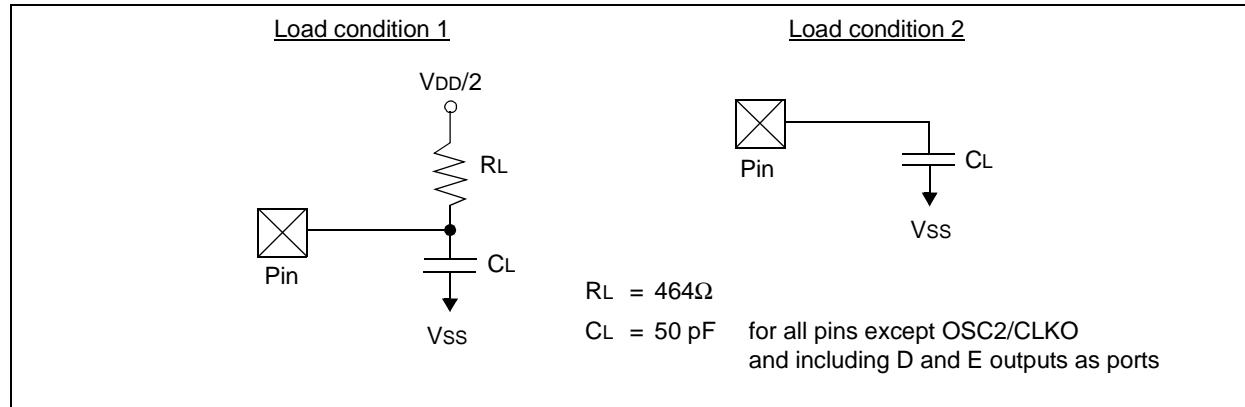
## 22.3.2 TIMING CONDITIONS

The temperature and voltages specified in Table 22-3 apply to all timing specifications unless otherwise noted. Figure 22-4 specifies the load conditions for the timing specifications.

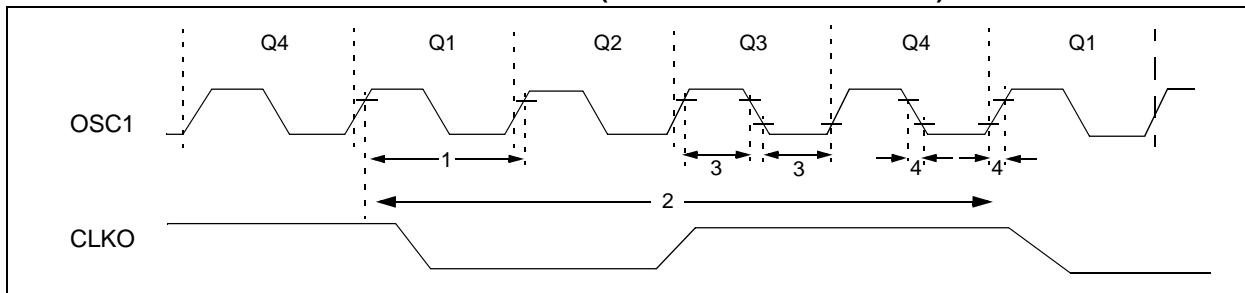
**TABLE 22-3: TEMPERATURE AND VOLTAGE SPECIFICATIONS - AC**

AC CHARACTERISTICS	Standard Operating Conditions (unless otherwise stated)	
	Operating temperature	-40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended
Operating voltage VDD range as described in DC spec Section 22.1 and Section 22.2. LC parts operate for industrial temperatures only.		

**FIGURE 22-4: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**



## 22.3.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 22-5: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)****TABLE 22-4: EXTERNAL CLOCK TIMING REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	Fosc	External CLK1 Frequency <sup>(1)</sup>	DC	40	MHz	EC, ECIO, -40°C to +85°C
		Oscillator Frequency <sup>(1)</sup>	DC	25	MHz	EC, ECIO, +85°C to +125°C
			DC	4	MHz	RC osc
			0.1	4	MHz	XT osc
			4	25	MHz	HS osc
			4	10	MHz	HS + PLL osc, -40°C to +85°C
			4	6.25	MHz	HS + PLL osc, +85°C to +125°C
			5	200	kHz	LP Osc mode
1	Tosc	External CLK1 Period <sup>(1)</sup>	25	—	ns	EC, ECIO, -40°C to +85°C
		Oscillator Period <sup>(1)</sup>	40	—	ns	EC, ECIO, +85°C to +125°C
			250	—	ns	RC osc
			250	10,000	ns	XT osc
			40	250	ns	HS osc
			100	250	ns	HS + PLL osc, -40°C to +85°C
			160	250	ns	HS + PLL osc, +85°C to +125°C
			25	—	μs	LP osc
2	TCY	Instruction Cycle Time <sup>(1)</sup>	100	—	ns	TCY = 4/Fosc, -40°C to +85°C
			160	—	ns	TCY = 4/Fosc, +85°C to +125°C
3	TosL, TosH	External Clock in (OSC1) High or Low Time	30	—	ns	XT osc
			2.5	—	μs	LP osc
			10	—	ns	HS osc
4	TosR, TosF	External Clock in (OSC1) Rise or Fall Time	—	20	ns	XT osc
			—	50	ns	LP osc
			—	7.5	ns	HS osc

**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.

# PIC18FXX2

TABLE 22-5: PLL CLOCK TIMING SPECIFICATIONS (V<sub>DD</sub> = 4.2 TO 5.5V)

Param No.	Sym	Characteristic	Min	Ty†	Max	Units	Conditions
—	FOSC	Oscillator Frequency Range	4	—	10	MHz	HS mode only
—	F <sub>SYS</sub>	On-chip VCO System Frequency	16	—	40	MHz	HS mode only
—	t <sub>rc</sub>	PLL Start-up Time (Lock Time)	—	—	2	ms	
—	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.

FIGURE 22-6: CLKO AND I/O TIMING

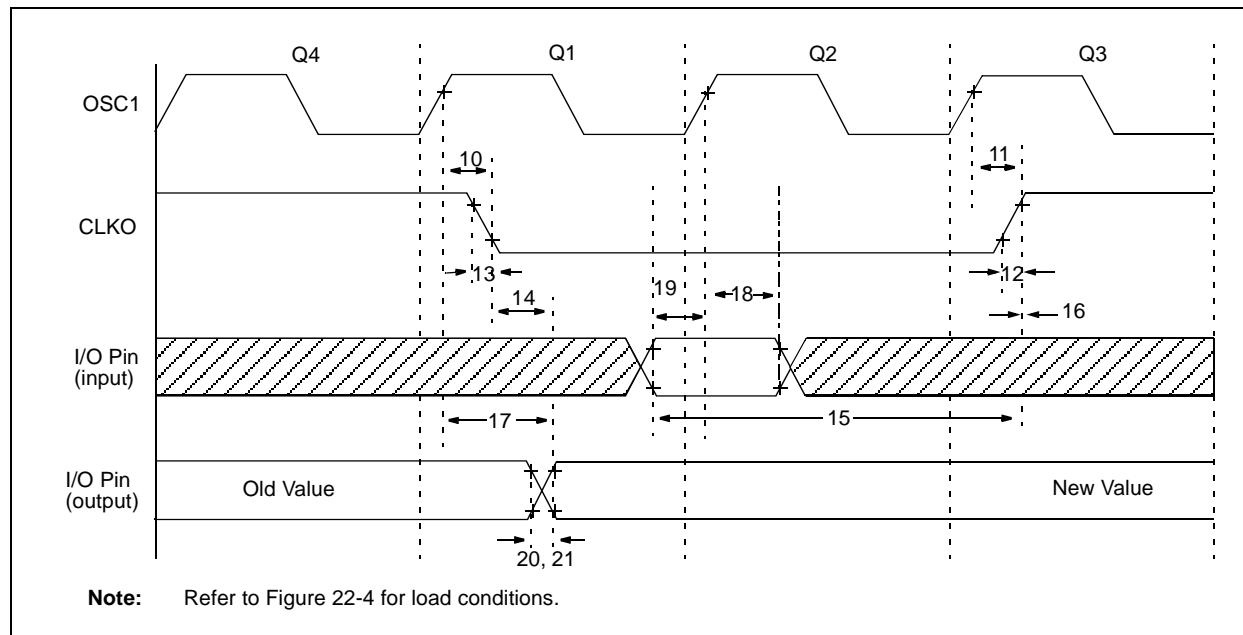


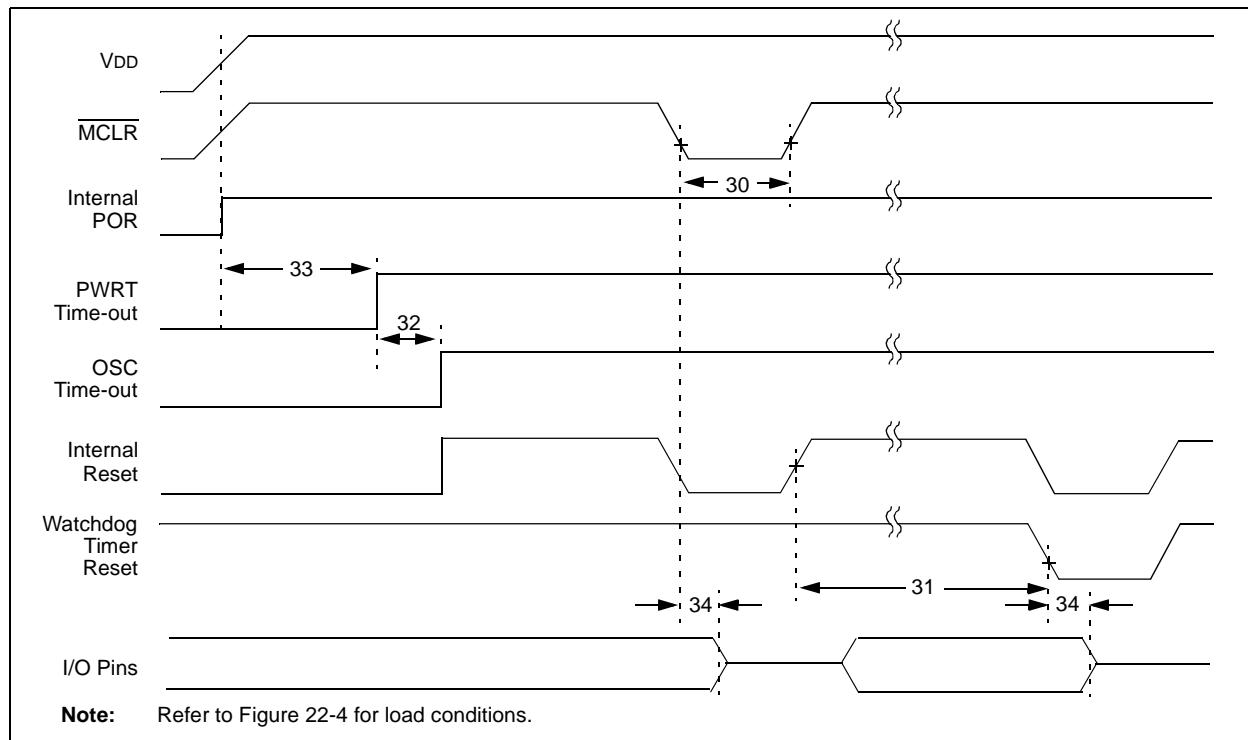
TABLE 22-6: CLKO AND I/O TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
10	TosH2ckL	OSC1↑ to CLKO↓	—	75	200	ns	(Note 1)
11	TosH2ckH	OSC1↑ to CLKO↑	—	75	200	ns	(Note 1)
12	TckR	CLKO rise time	—	35	100	ns	(Note 1)
13	TckF	CLKO fall time	—	35	100	ns	(Note 1)
14	TckL2ioV	CLKO↓ to Port out valid	—	—	0.5 TCY + 20	ns	(Note 1)
15	TioV2ckH	Port in valid before CLKO ↑	0.25 TCY + 25	—	—	ns	(Note 1)
16	TckH2iol	Port in hold after CLKO ↑	0	—	—	ns	(Note 1)
17	TosH2ioV	OSC1↑ (Q1 cycle) to Port out valid	—	50	150	ns	
18	TosH2iol	OSC1↑ (Q2 cycle) to Port input invalid (I/O in hold time)	PIC18FXXX	100	—	—	ns
18A			PIC18LFXXX	200	—	—	ns
19	TioV2osH	Port input valid to OSC1↑ (I/O in setup time)	0	—	—	ns	
20	TioR	Port output rise time	PIC18FXXX	—	10	25	ns
20A			PIC18LFXXX	—	—	60	ns VDD = 2V
21	TioF	Port output fall time	PIC18FXXX	—	10	25	ns
21A			PIC18LFXXX	—	—	60	ns VDD = 2V
22††	TINP	INT pin high or low time	TCY	—	—	ns	
23††	TRBP	RB7:RB4 change INT high or low time	TCY	—	—	ns	
24††	TRCP	RC7:RC4 change INT high or low time	20	—	—	ns	

†† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in RC mode, where CLKO output is 4 x Tosc.

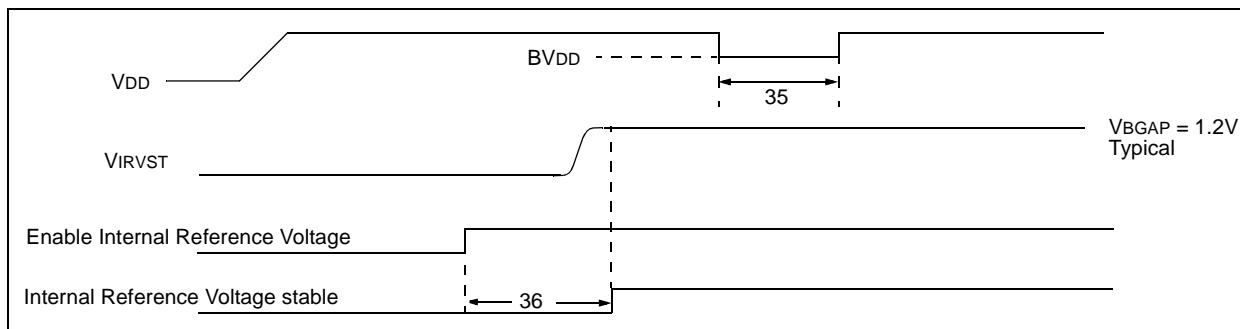
FIGURE 22-7: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



# PIC18FXX2

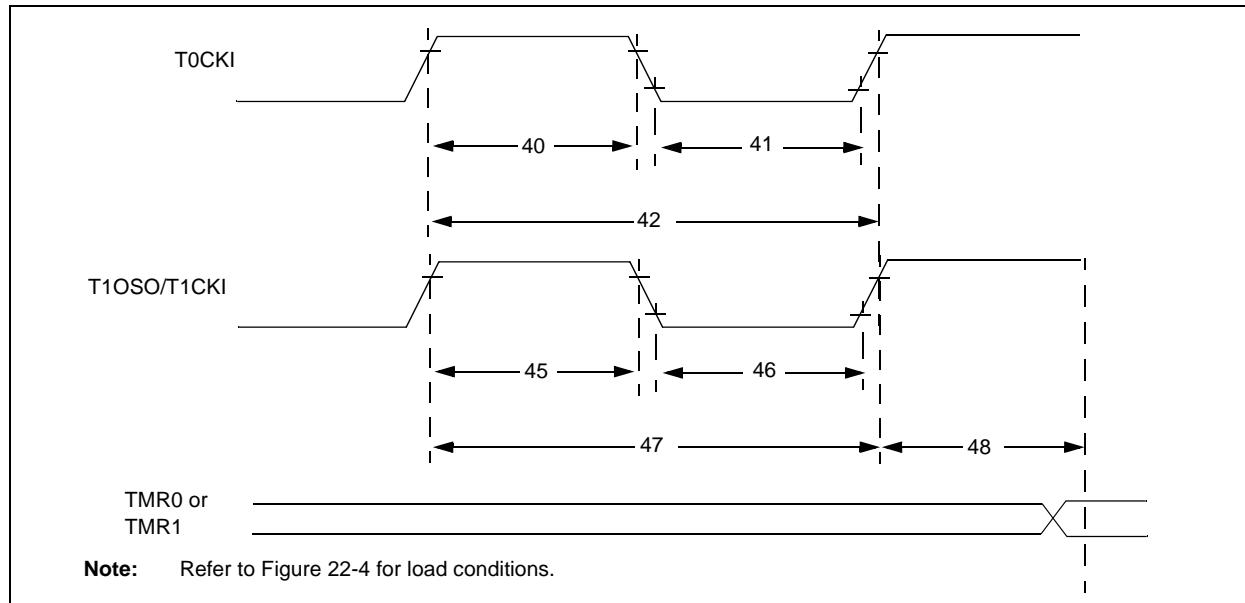
---

**FIGURE 22-8: BROWN-OUT RESET TIMING**



**TABLE 22-7: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS**

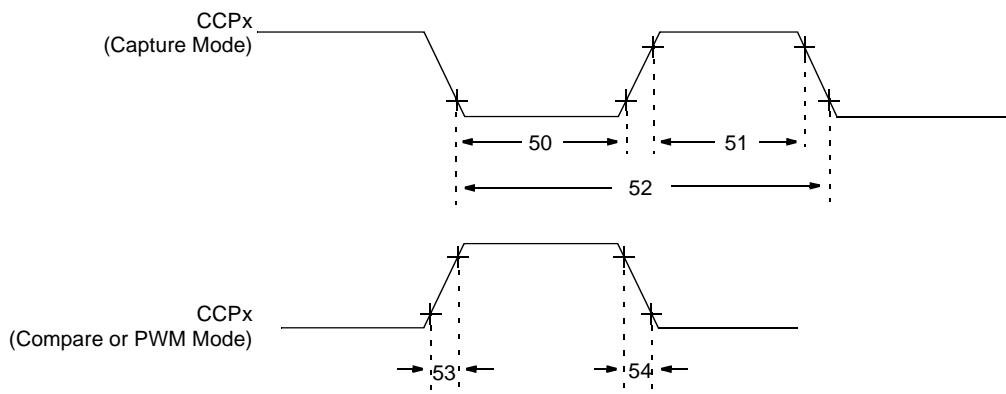
Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
30	TmCL	MCLR Pulse Width (low)	2	—	—	μs	
31	TWDT	Watchdog Timer Time-out Period (No Postscaler)	7	18	33	ms	
32	TOST	Oscillation Start-up Timer Period	1024 Tosc	—	1024 Tosc	—	Tosc = OSC1 period
33	TPWRT	Power up Timer Period	28	72	132	ms	
34	TIOZ	I/O Hi-impedance from MCLR Low or Watchdog Timer Reset	—	2	—	μs	
35	TBOR	Brown-out Reset Pulse Width	200	—	—	μs	$VDD \leq BVDD$ (see D005)
36	TIVRST	Time for Internal Reference Voltage to become stable	—	20	500	μs	
37	TLVD	Low Voltage Detect Pulse Width	200	—	—	μs	$VDD \leq VLVD$ (see D420)

**FIGURE 22-9: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS****TABLE 22-8: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
40	Tt0H	T0CKI High Pulse Width	No Prescaler	0.5TCY + 20	—	ns	
			With Prescaler	10	—	ns	
41	Tt0L	T0CKI Low Pulse Width	No Prescaler	0.5TCY + 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 $\frac{TCY + 40}{N}$	—	ns	
45	Tt1H	T1CKI High Time	Synchronous, no prescaler	0.5TCY + 20	—	ns	
			Synchronous, with prescaler	PIC18FXXX	10	—	
				PIC18LFXXX	25	—	
			Asynchronous	PIC18FXXX	30	—	
46	Tt1L	T1CKI Low Time	Synchronous, no prescaler	0.5TCY + 5	—	ns	
			Synchronous, with prescaler	PIC18FXXX	10	—	
				PIC18LFXXX	25	—	
			Asynchronous	PIC18FXXX	30	—	
47	Tt1P	T1CKI input period	Synchronous	Greater of: 20 ns or $\frac{TCY + 40}{N}$	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous	60	—	ns	
	Ft1	T1CKI oscillator input frequency range		DC	50	kHz	
48	Tcke2tmr1	Delay from external T1CKI clock edge to timer increment		2 Tosc	7 Tosc	—	

# PIC18FXX2

**FIGURE 22-10: CAPTURE/COMPARE/PWM TIMINGS (CCP1 AND CCP2)**

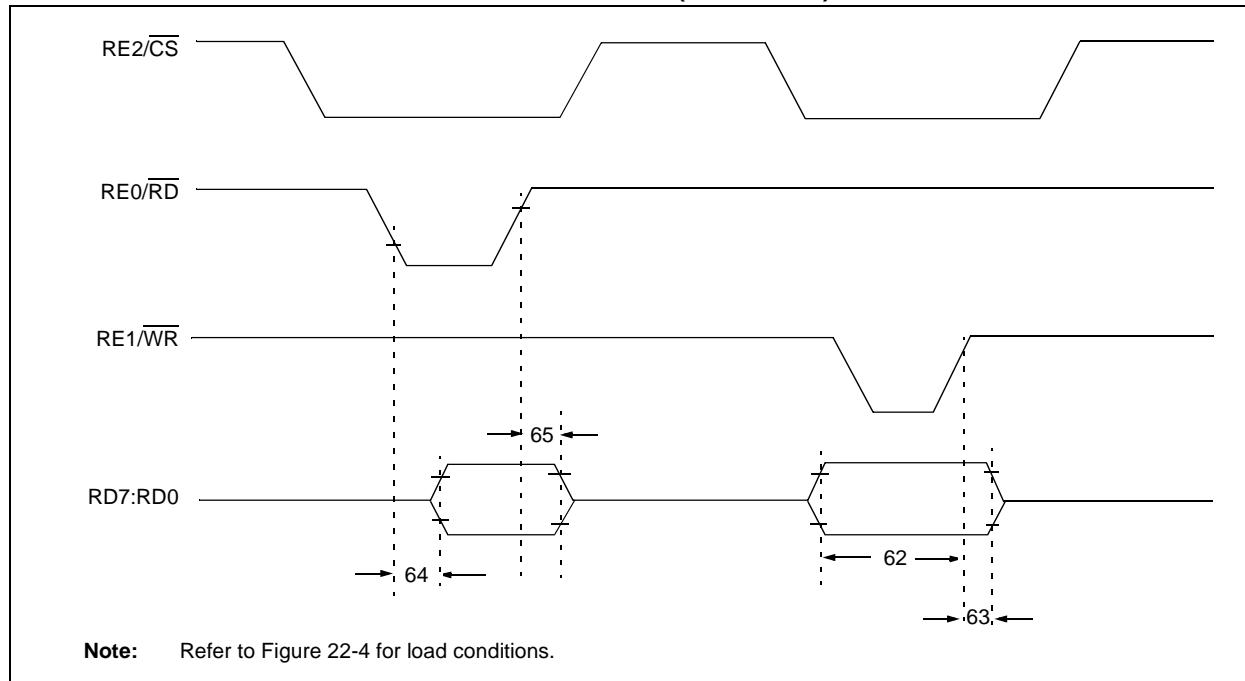


**Note:** Refer to Figure 22-4 for load conditions.

**TABLE 22-9: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP1 AND CCP2)**

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
50	TccL	CCPx input low time	No Prescaler	0.5 TCY + 20	—	ns	
			With Prescaler	PIC18FXXX PIC18LFXXX	10 20	— —	
		CCPx input high time	No Prescaler	0.5 TCY + 20	—	ns	
51	TccH	CCPx input high time	With Prescaler	PIC18FXXX PIC18LFXXX	10 20	— —	ns
			No Prescaler	0.5 TCY + 20	—	ns	
52	TccP	CCPx input period		$\frac{3 \text{ TCY} + 40}{N}$	—	ns	N = prescale value (1,4 or 16)
53	TccR	CCPx output fall time		PIC18FXXX PIC18LFXXX	— —	25 60	ns
		VDD = 2V					
54	TccF	CCPx output fall time		PIC18FXXX PIC18LFXXX	— —	25 60	ns
		VDD = 2V					

**FIGURE 22-11: PARALLEL SLAVE PORT TIMING (PIC18F4X2)**

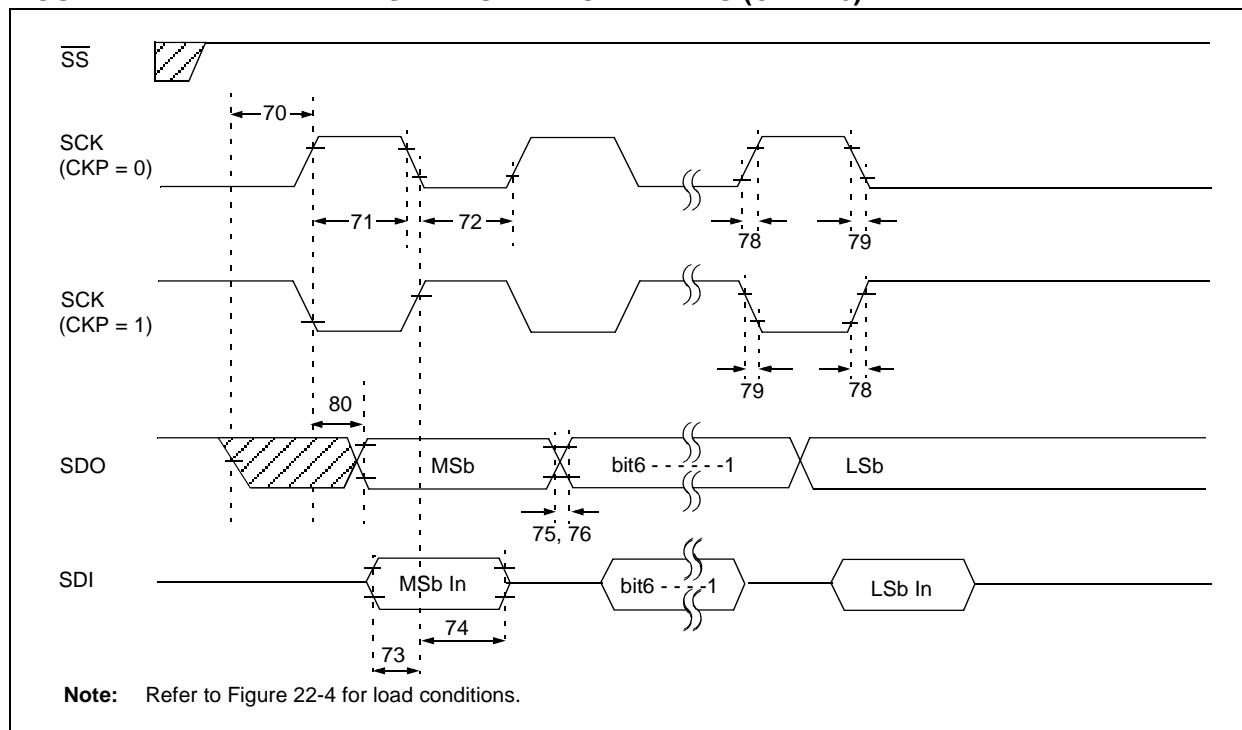


**TABLE 22-10: PARALLEL SLAVE PORT REQUIREMENTS (PIC18F4X2)**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
62	TdtV2wrH	Data in valid before $\overline{WR} \uparrow$ or $\overline{CS} \uparrow$ (setup time)	20 25	— —	ns ns	Extended Temp. Range
63	TwrH2dtl	$\overline{WR} \uparrow$ or $\overline{CS} \uparrow$ to data-in invalid (hold time)	PIC18FXXX	20	—	ns
			PIC18LFXXX	35	—	ns $V_{DD} = 2V$
64	TrdL2dtV	$\overline{RD} \downarrow$ and $\overline{CS} \downarrow$ to data-out valid	— —	80 90	ns ns	Extended Temp. Range
65	TrdH2dtI	$\overline{RD} \uparrow$ or $\overline{CS} \downarrow$ to data-out invalid	10	30	ns	
66	TibflNH	Inhibit of the IBF flag bit being cleared from $\overline{WR} \uparrow$ or $\overline{CS} \uparrow$	—	3 TCY		

# PIC18FXX2

**FIGURE 22-12: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)**

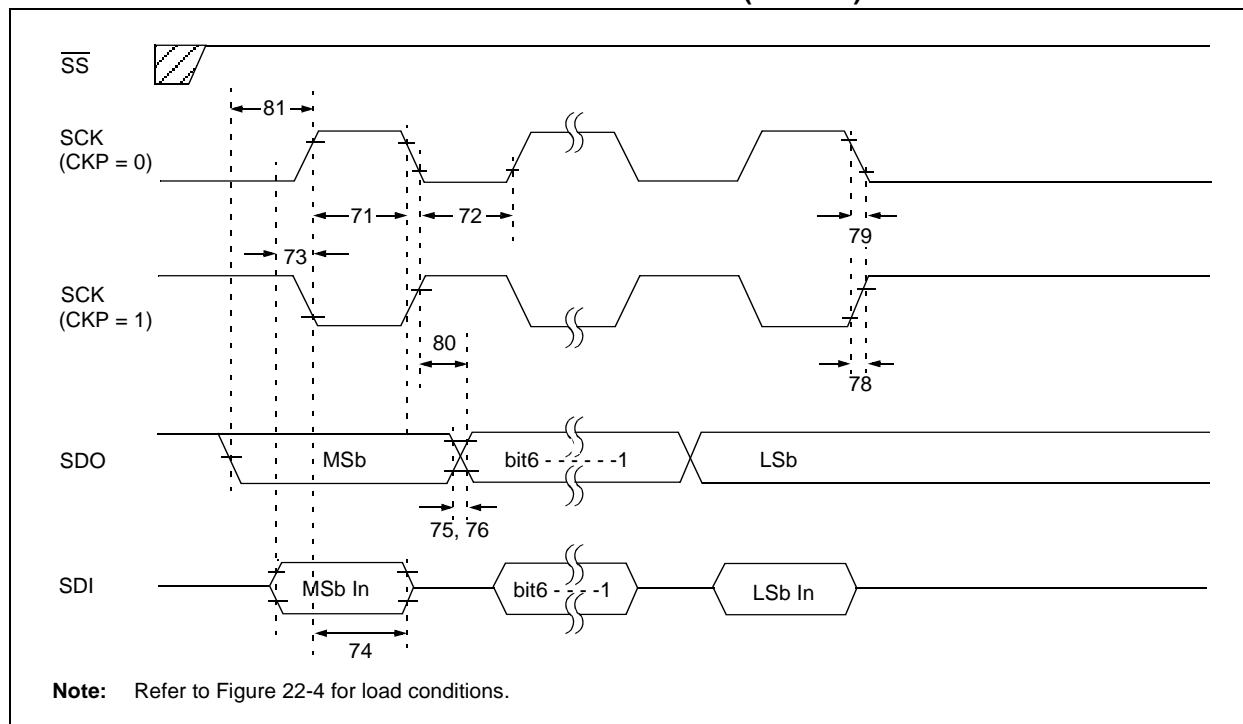


**TABLE 22-11: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)**

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2sch, TssL2scl	$\overline{SS} \downarrow$ to $SCK \downarrow$ or $SCK \uparrow$ input		TCY	—	ns	
71 71A	TscH	SCK input high time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
72 72A	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		100	—	ns	
73A	TB2B	Last clock edge of Byte1 to the 1st clock edge of Byte2		1.5 TCY + 40	—	ns	(Note 2)
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	ns	
75	TdoR	SDO data output rise time	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	VDD = 2V
76	TdoF	SDO data output fall time	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	VDD = 2V
78	TscR	SCK output rise time (Master mode)	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	VDD = 2V
79	TscF	SCK output fall time (Master mode)	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	VDD = 2V
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC18FXXX	—	50	ns	
			PIC18LFXXX	—	150	ns	VDD = 2V

**Note 1:** Requires the use of Parameter # 73A.

**2:** Only if Parameter # 71A and # 72A are used.

**FIGURE 22-13: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)****TABLE 22-12: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)**

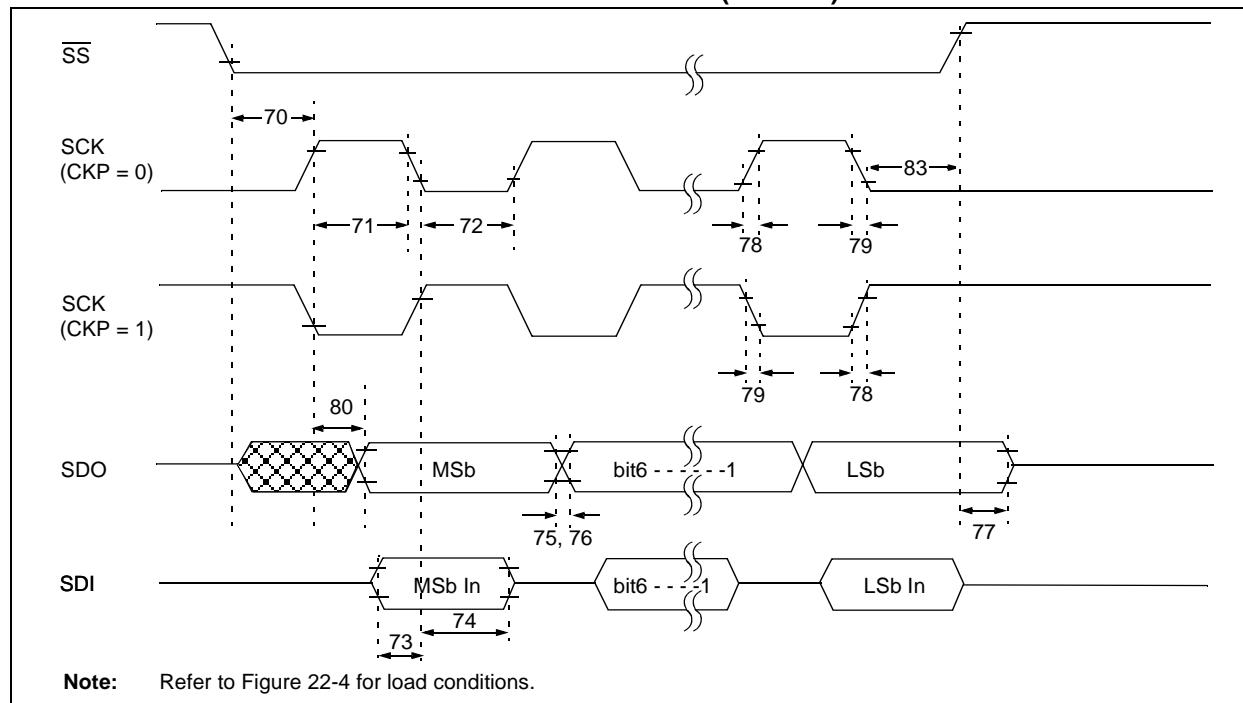
Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
71	TscH	SCK input high time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
71A	TscL	Single Byte	40	—	ns		(Note 1)
72	TscL	SCK input low time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
72A	TscH2diL, TscL2diL	Single Byte	40	—	ns		(Note 1)
73	TdiV2scH, TdiV2scL	Setup time of SDI data input to SCK edge		100	—	ns	
73A	Tb2B	Last clock edge of Byte1 to the 1st clock edge of Byte2		1.5 TCY + 40	—	ns	(Note 2)
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	ns	
75	TdoR	SDO data output rise time	PIC18FXXX	—	25	ns	
75A	TdoF	PIC18LFXXX	—	60	ns	VDD = 2V	
76	TdoF	SDO data output fall time	PIC18FXXX	—	25	ns	
76A	TscR	PIC18LFXXX	—	60	ns	VDD = 2V	
78	TscF	SCK output rise time (Master mode)	PIC18FXXX	—	25	ns	
78A	TscH2doV, TscL2doV	PIC18LFXXX	—	60	ns	VDD = 2V	
79	TscF	SCK output fall time (Master mode)	PIC18FXXX	—	25	ns	
79A	TdoV2scH, TdoV2scL	PIC18LFXXX	—	60	ns	VDD = 2V	
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC18FXXX	—	50	ns	
80A	TdoV2scH, TdoV2scL	PIC18LFXXX	—	150	ns	VDD = 2V	
81	TdoV2scH, TdoV2scL	SDO data output setup to SCK edge		TCY	—	ns	

**Note 1:** Requires the use of Parameter # 73A.

**2:** Only if Parameter # 71A and # 72A are used.

# PIC18FXX2

**FIGURE 22-14: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)**



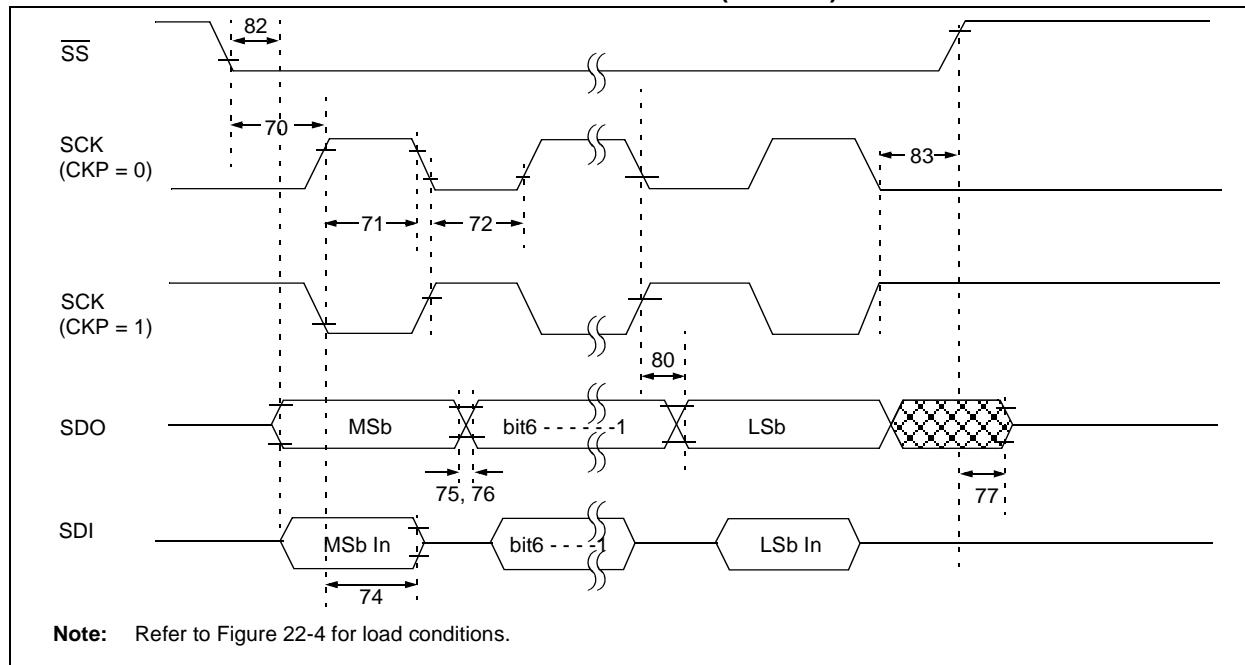
**TABLE 22-13: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING (CKE = 0))**

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS} \downarrow$ to SCK $\downarrow$ or SCK $\uparrow$ input		TCY	—	ns	
71 71A	TscH	SCK input high time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
72 72A	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		100	—	ns	
73A	TB2B	Last clock edge of Byte1 to the first clock edge of Byte2		1.5 TCY + 40	—	ns	(Note 2)
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	ns	
75	TdoR	SDO data output rise time	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	VDD = 2V
76	TdoF	SDO data output fall time	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	VDD = 2V
77	TssH2doZ	$\overline{SS} \uparrow$ to SDO output hi-impedance		10	50	ns	
78	Tscr	SCK output rise time (Master mode)	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	VDD = 2V
79	TscF	SCK output fall time (Master mode)	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	VDD = 2V
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC18FXXX	—	50	ns	
			PIC18LFXXX	—	150	ns	VDD = 2V
83	TscH2ssH, TscL2ssH	$\overline{SS} \uparrow$ 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 22-15: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)**



**TABLE 22-14: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)**

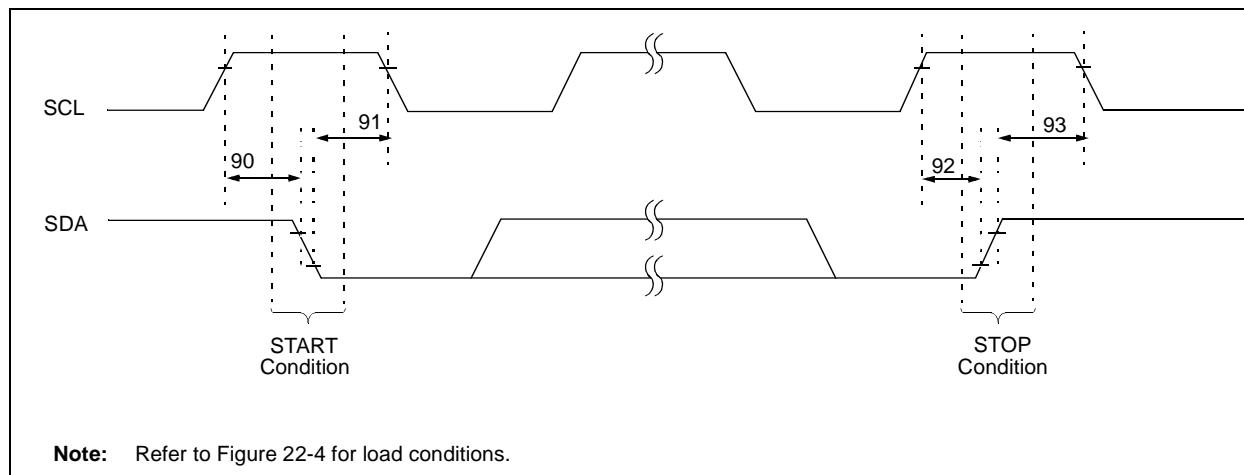
Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS} \downarrow$ to $SCK \downarrow$ or $SCK \uparrow$ input		T <sub>CY</sub>	—	ns	
71 71A	TscH	SCK input high time (Slave mode)	Continuous	1.25 T <sub>CY</sub> + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
72 72A	TscL	SCK input low time (Slave mode)	Continuous	1.25 T <sub>CY</sub> + 30	—	ns	
			Single Byte	40	—	ns	(Note 1)
73A	TB2B	Last clock edge of Byte1 to the first clock edge of Byte2		1.5 T <sub>CY</sub> + 40	—	ns	(Note 2)
74	TscH2diL, TscL2diL	Hold time of SDI data input to SCK edge		100	—	ns	
75	TdoR	SDO data output rise time	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	V <sub>DD</sub> = 2V
76	TdoF	SDO data output fall time	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	V <sub>DD</sub> = 2V
77	TssH2doZ	$\overline{SS} \uparrow$ to SDO output hi-impedance		10	50	ns	
78	TscR	SCK output rise time (Master mode)	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	V <sub>DD</sub> = 2V
79	TscF	SCK output fall time (Master mode)	PIC18FXXX	—	25	ns	
			PIC18LFXXX	—	60	ns	V <sub>DD</sub> = 2V
80	TscH2doV, TscL2doV	SDO data output valid after SCK edge	PIC18FXXX	—	50	ns	
			PIC18LFXXX	—	150	ns	V <sub>DD</sub> = 2V
82	TssL2doV	SDO data output valid after $\overline{SS} \downarrow$ edge	PIC18FXXX	—	50	ns	
			PIC18LFXXX	—	150	ns	V <sub>DD</sub> = 2V
83	TscH2ssH, TscL2ssH	$\overline{SS} \uparrow$ after SCK edge		1.5 T <sub>CY</sub> + 40	—	ns	

**Note 1:** Requires the use of Parameter # 73A.

**2:** Only if Parameter # 71A and # 72A are used.

# PIC18FXX2

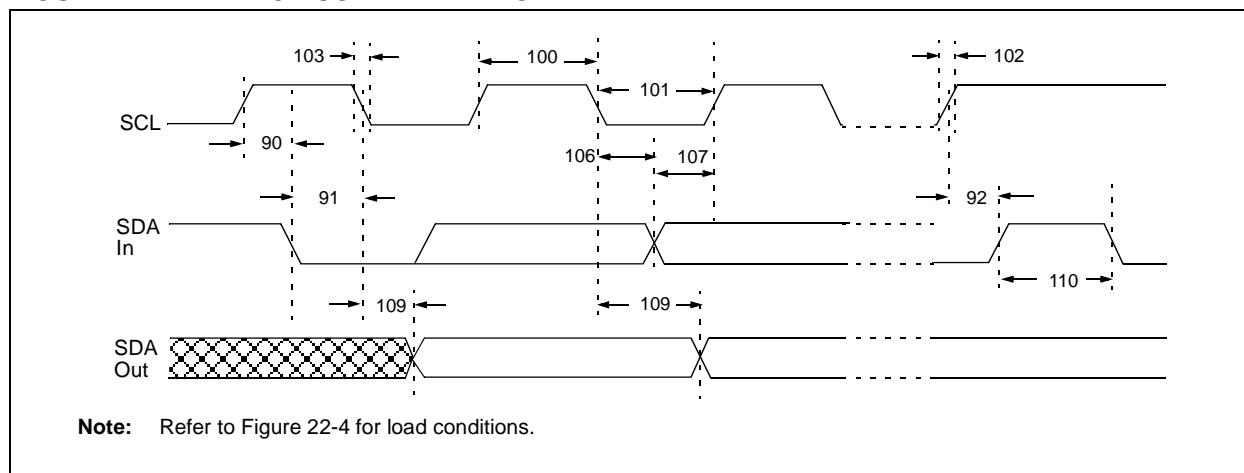
**FIGURE 22-16: I<sup>2</sup>C BUS START/STOP BITS TIMING**



**TABLE 22-15: I<sup>2</sup>C BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)**

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	START condition	100 kHz mode	4700	—	ns	Only relevant for Repeated START condition
			400 kHz mode	600	—		
91	THD:STA	START condition	100 kHz mode	4000	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—		
92	TSU:STO	STOP condition	100 kHz mode	4700	—	ns	
			400 kHz mode	600	—		
93	THD:STO	STOP condition	100 kHz mode	4000	—	ns	
			400 kHz mode	600	—		

**FIGURE 22-17: I<sup>2</sup>C BUS DATA TIMING**



**TABLE 22-16: I<sup>2</sup>C BUS DATA REQUIREMENTS (SLAVE MODE)**

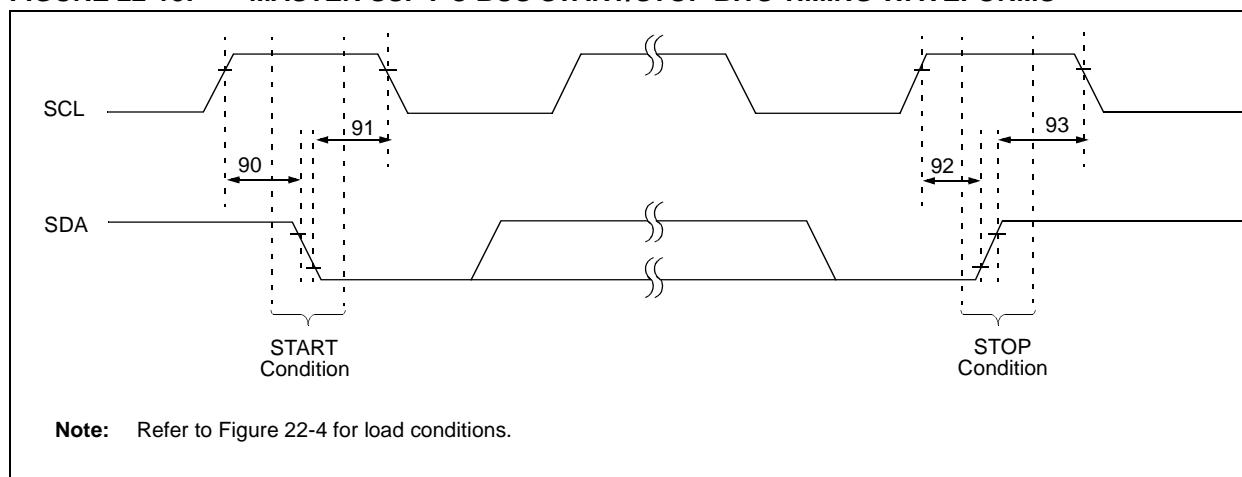
Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock high time	100 kHz mode	4.0	—	μs	PIC18FXXX must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	μs	PIC18FXXX must operate at a minimum of 10 MHz
			SSP Module	1.5 TCY	—		
101	TLOW	Clock low time	100 kHz mode	4.7	—	μs	PIC18FXXX must operate at a minimum of 1.5 MHz
			400 kHz mode	1.3	—	μs	PIC18FXXX must operate at a minimum of 10 MHz
			SSP 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	—	1000	ns	VDD ≥ 4.2V
			400 kHz mode	20 + 0.1 CB	300	ns	VDD ≥ 4.2V
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	C <sub>B</sub>	Bus capacitive loading		—	400	pF	

**Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of START or STOP conditions.

- 2:** A Fast mode I<sup>2</sup>C bus device can be used in a Standard mode I<sup>2</sup>C bus system, but the requirement TSU:DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line. TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I<sup>2</sup>C bus specification) before the SCL line is released.

# PIC18FXX2

**FIGURE 22-18: MASTER SSP I<sup>2</sup>C BUS START/STOP BITS TIMING WAVEFORMS**



**TABLE 22-17: MASTER SSP I<sup>2</sup>C BUS START/STOP BITS REQUIREMENTS**

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	START condition Setup time	100 kHz mode	2(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 22-19: MASTER SSP I<sup>2</sup>C BUS DATA TIMING**

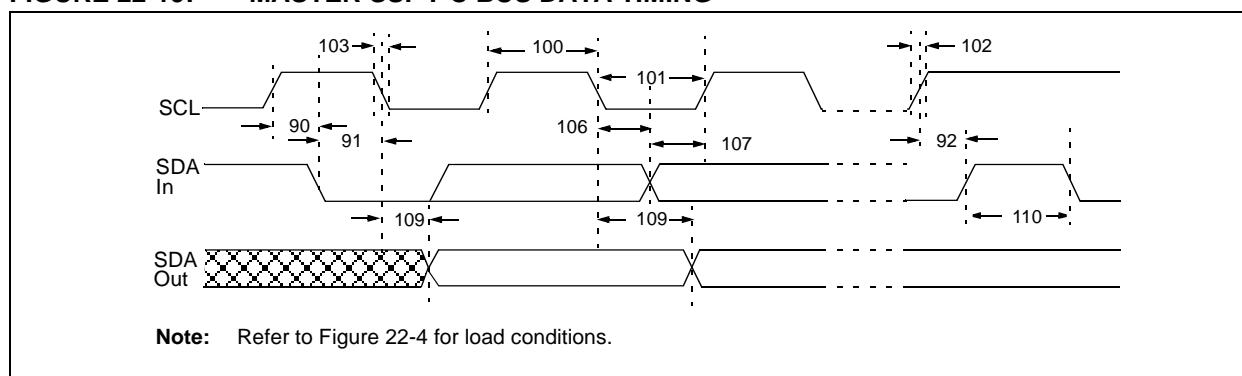


TABLE 22-18: MASTER SSP I<sup>2</sup>C BUS DATA REQUIREMENTS

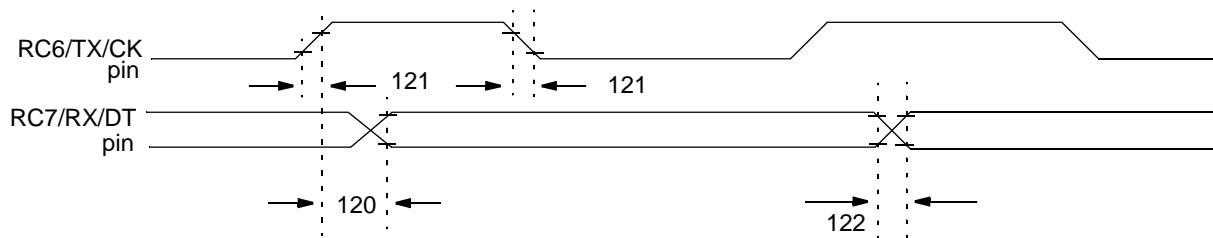
Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock high time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			400 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			1 MHz mode <sup>(1)</sup>	2(Tosc)(BRG + 1)	—	ms	
101	TLOW	Clock low time	100 kHz mode	2(Tosc)(BRG + 1)	—	ms	
			400 kHz mode	2(Tosc)(BRG + 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	—	1000	ns	VDD ≥ 4.2V
			400 kHz mode	20 + 0.1 CB	300	ns	VDD ≥ 4.2V
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 ≥ 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.

# PIC18FXX2

**FIGURE 22-20: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

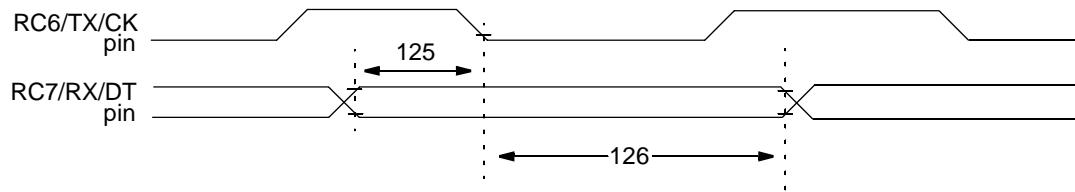


**Note:** Refer to Figure 22-4 for load conditions.

**TABLE 22-19: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
120	TckH2dtV	SYNC XMIT (MASTER & SLAVE) Clock high to data out valid	PIC18FXXX	—	50	ns
			PIC18LFXXX	—	150	ns VDD = 2V
121	Tckr	Clock out rise time and fall time (Master mode)	PIC18FXXX	—	25	ns
			PIC18LFXXX	—	60	ns VDD = 2V
122	Tdtr	Data out rise time and fall time	PIC18FXXX	—	25	ns
			PIC18LFXXX	—	60	ns VDD = 2V

**FIGURE 22-21: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**



**Note:** Refer to Figure 22-4 for load conditions.

**TABLE 22-20: USART 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)	PIC18FXXX	15	—	ns
			PIC18LFXXX	20	—	ns VDD = 2V

**TABLE 22-21: A/D CONVERTER CHARACTERISTICS: PIC18FXX2 (INDUSTRIAL, EXTENDED)  
PIC18LFXX2 (INDUSTRIAL)**

Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
A01	NR	Resolution	—	—	10	bit	
A03	EIL	Integral linearity error	—	—	<±1	LSb	VREF = VDD = 5.0V
A04	EDL	Differential linearity error	—	—	<±1	LSb	VREF = VDD = 5.0V
A05	EG	Gain error	—	—	<±1	LSb	VREF = VDD = 5.0V
A06	EOFF	Offset error	—	—	<±1.5	LSb	VREF = VDD = 5.0V
A10	—	Monotonicity	guaranteed <sup>(2)</sup>			—	VSS ≤ VAIN ≤ VREF
A20	VREF	Reference Voltage (VREFH – VREFL)	1.8V 3V	— —	—	V	VDD < 3.0V VDD ≥ 3.0V
A21	VREFH	Reference voltage High	AVSS	—	AVDD + 0.3V	V	
A22	VREFL	Reference voltage Low	AVss – 0.3V	—	VREFH	V	
A25	VAIN	Analog input voltage	AVss – 0.3V	—	AVDD + 0.3V	V	VDD ≥ 2.5V ( <b>Note 3</b> )
A30	ZAIN	Recommended impedance of analog voltage source	—	—	2.5	kΩ	( <b>Note 4</b> )
A50	IREF	VREF input current ( <b>Note 1</b> )	— —	— —	5 150	μA	During VAIN acquisition During A/D conversion cycle

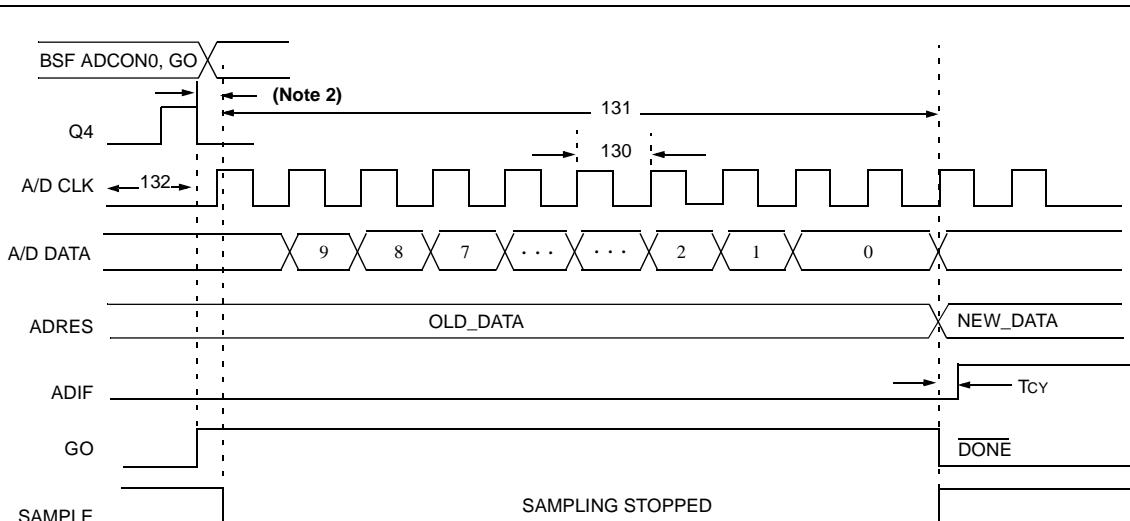
**Note 1:**  $V_{SS} \leq V_{AIN} \leq V_{REF}$

**2:** The A/D conversion result never decreases with an increase in the Input Voltage, and has no missing codes.

**3:** For  $V_{DD} < 2.5V$ ,  $V_{AIN}$  should be limited to <.5  $V_{DD}$ .

**4:** Maximum allowed impedance for analog voltage source is 10 kΩ. This requires higher acquisition times.

**FIGURE 22-22: A/D CONVERSION TIMING**



**Note 1:** If the A/D clock source is selected as RC, a time of  $T_{CY}$  is added before the A/D clock starts.  
This allows the SLEEP instruction to be executed.

**2:** This is a minimal RC delay (typically 100 nS), which also disconnects the holding capacitor from the analog input.

# PIC18FXX2

---

TABLE 22-22: A/D CONVERSION REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
130	TAD	A/D clock period	PIC18FXXX	1.6	20 <sup>(4)</sup>	μs	TOSC based
			PIC18FXXX	2.0	6.0	μs	A/D RC mode
131	TCNV	Conversion time (not including acquisition time) <b>(Note 1)</b>		11	12	TAD	
132		Acquisition time <b>(Note 2)</b>		5	—	μs	VREF = VDD = 5.0V
135		Switching Time from convert → sample		—	(Note 3)	μs	VREF = VDD = 2.5V

**Note 1:** ADRES register may be read on the following Tcy cycle.

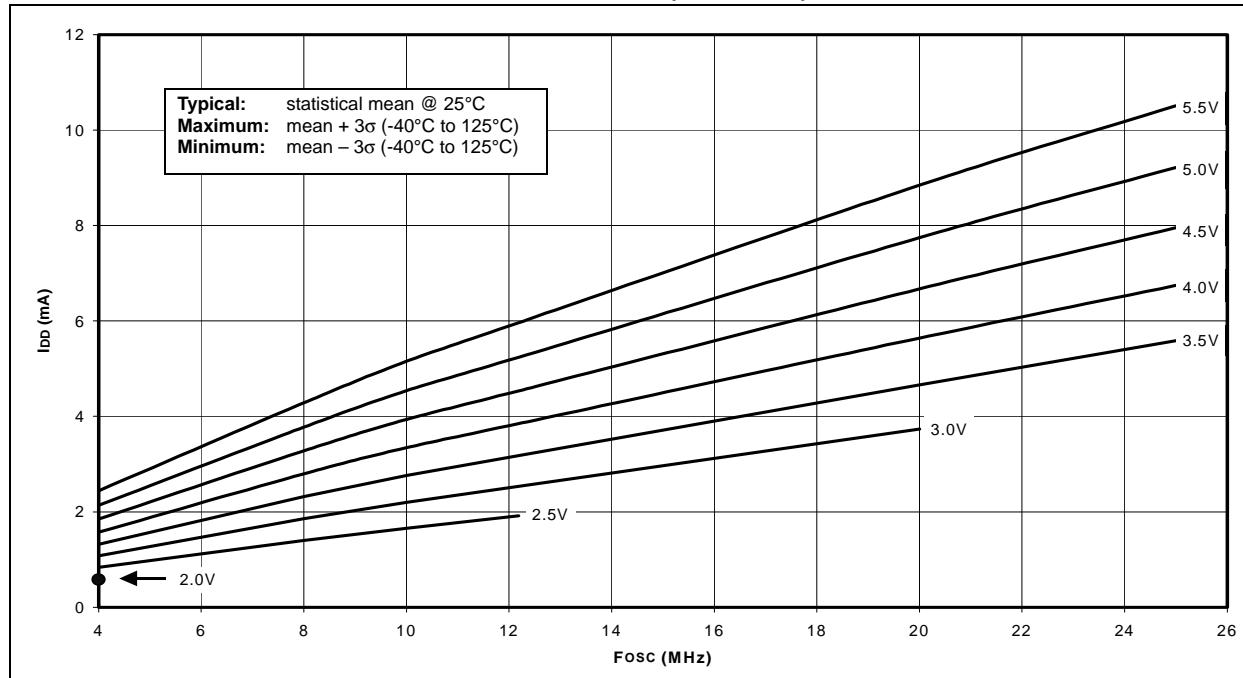
- 2: The time for the holding capacitor to acquire the “New” input voltage, when the new input value has not changed by more than 1 LSB from the last sampled voltage. The source impedance ( $R_s$ ) on the input channels is  $50\Omega$ . See Section 17.0 for more information on acquisition time consideration.
- 3: On the next Q4 cycle of the device clock.
- 4: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

## 23.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

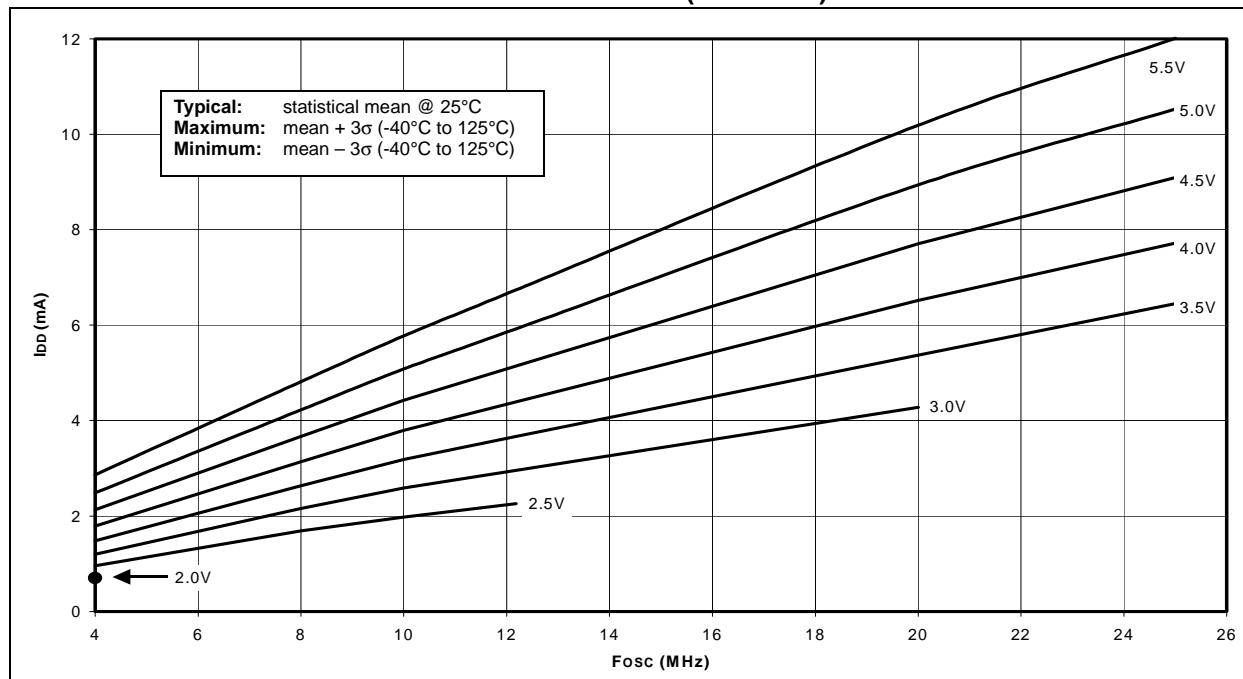
**Note:** The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "Maximum" or "minimum" represents (mean + 3 $\sigma$ ) or (mean - 3 $\sigma$ ) respectively, where  $\sigma$  is a standard deviation, over the whole temperature range.

**FIGURE 23-1: TYPICAL IDD vs. Fosc OVER VDD (HS MODE)**



**FIGURE 23-2: MAXIMUM IDD vs. Fosc OVER VDD (HS MODE)**



# PIC18FXX2

FIGURE 23-3: TYPICAL IDD vs. Fosc OVER VDD (HS/PLL MODE)

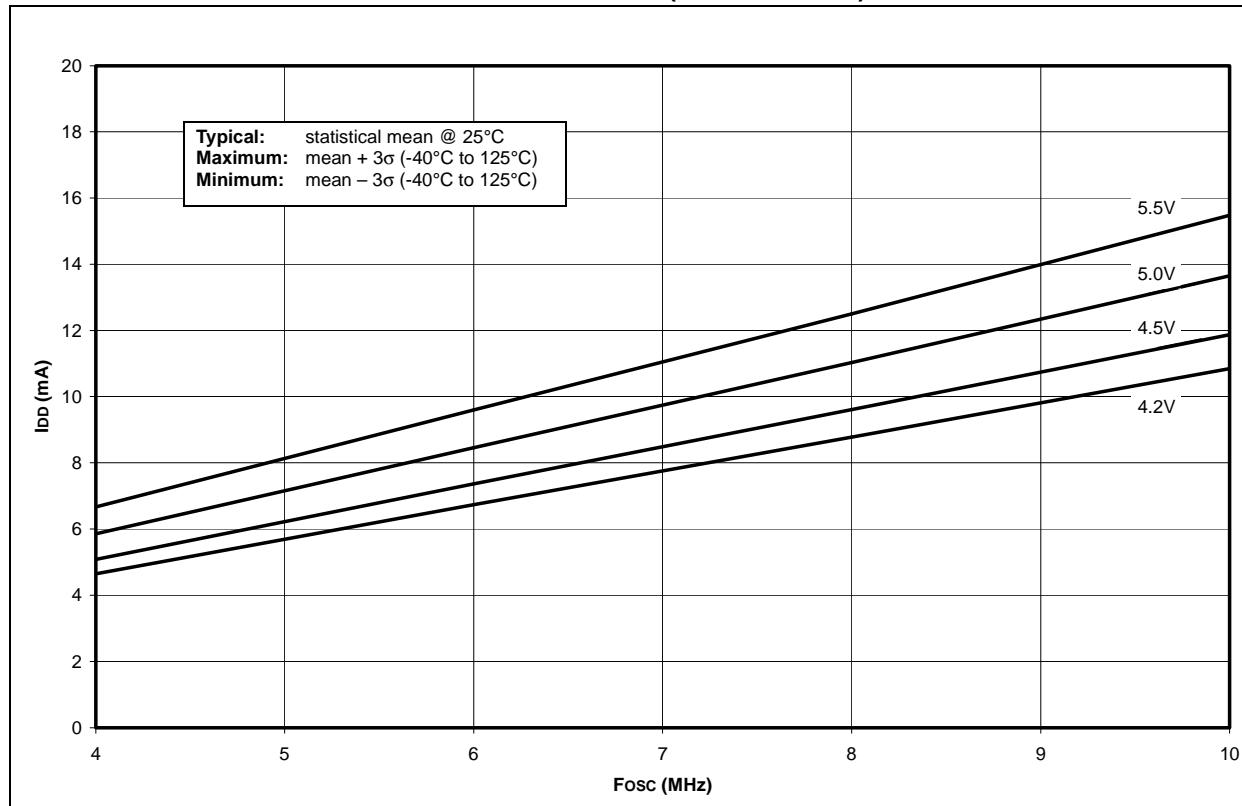
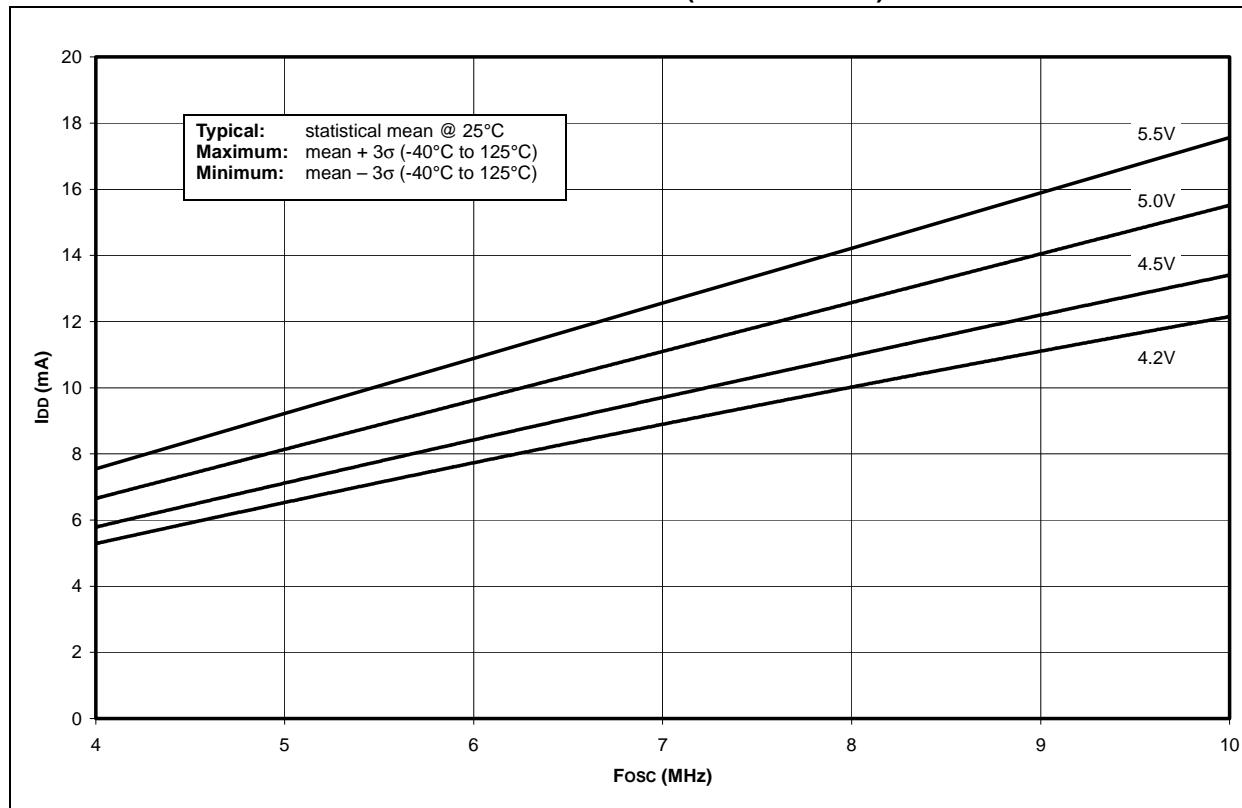
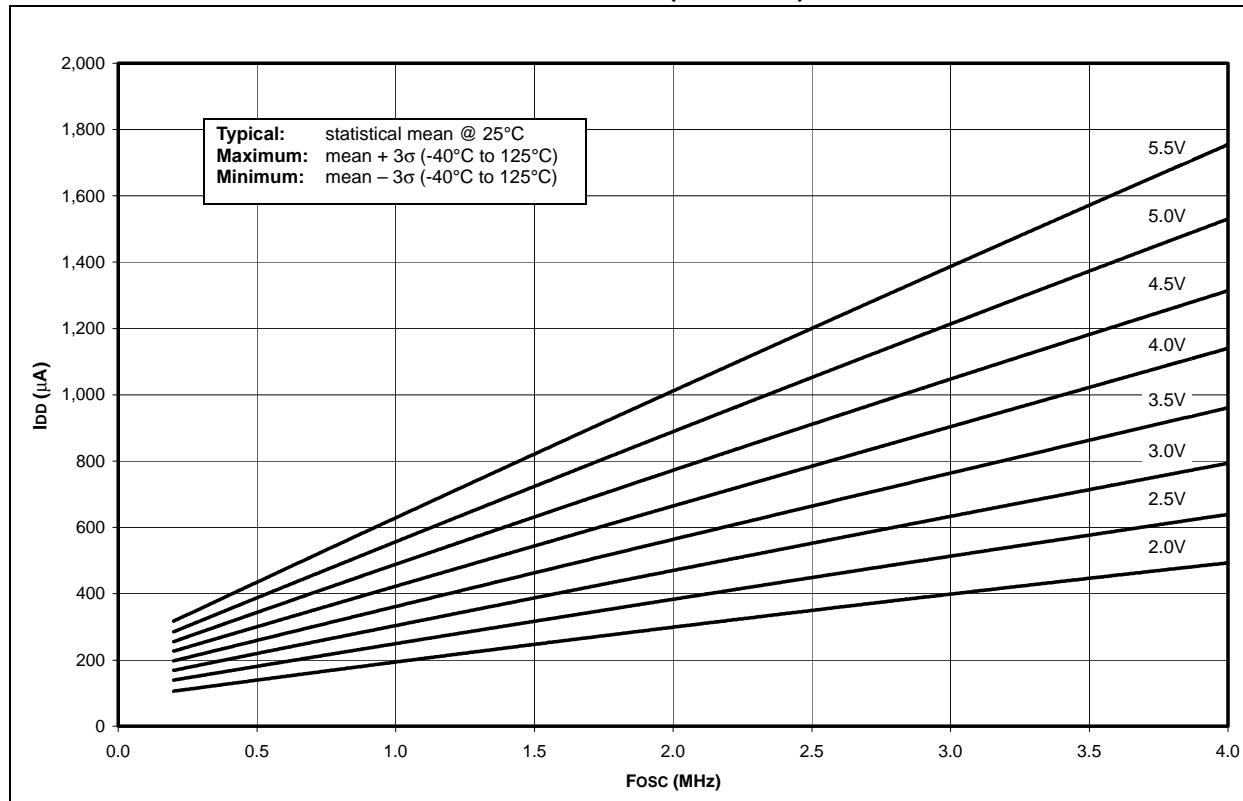
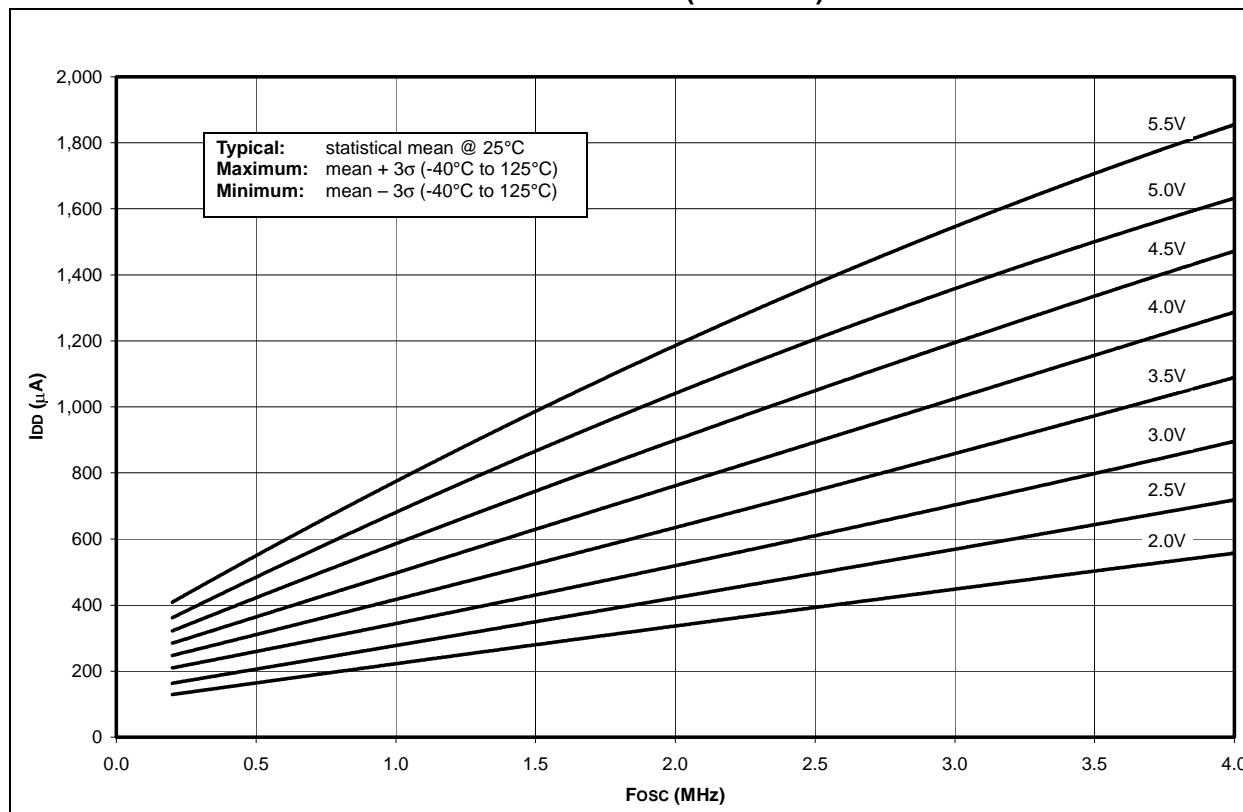


FIGURE 23-4: MAXIMUM IDD vs. Fosc OVER VDD (HS/PLL MODE)



**FIGURE 23-5: TYPICAL IDD vs. Fosc OVER VDD (XT MODE)****FIGURE 23-6: MAXIMUM IDD vs. Fosc OVER VDD (XT MODE)**

# PIC18FXX2

FIGURE 23-7: TYPICAL IDD vs. Fosc OVER VDD (LP MODE)

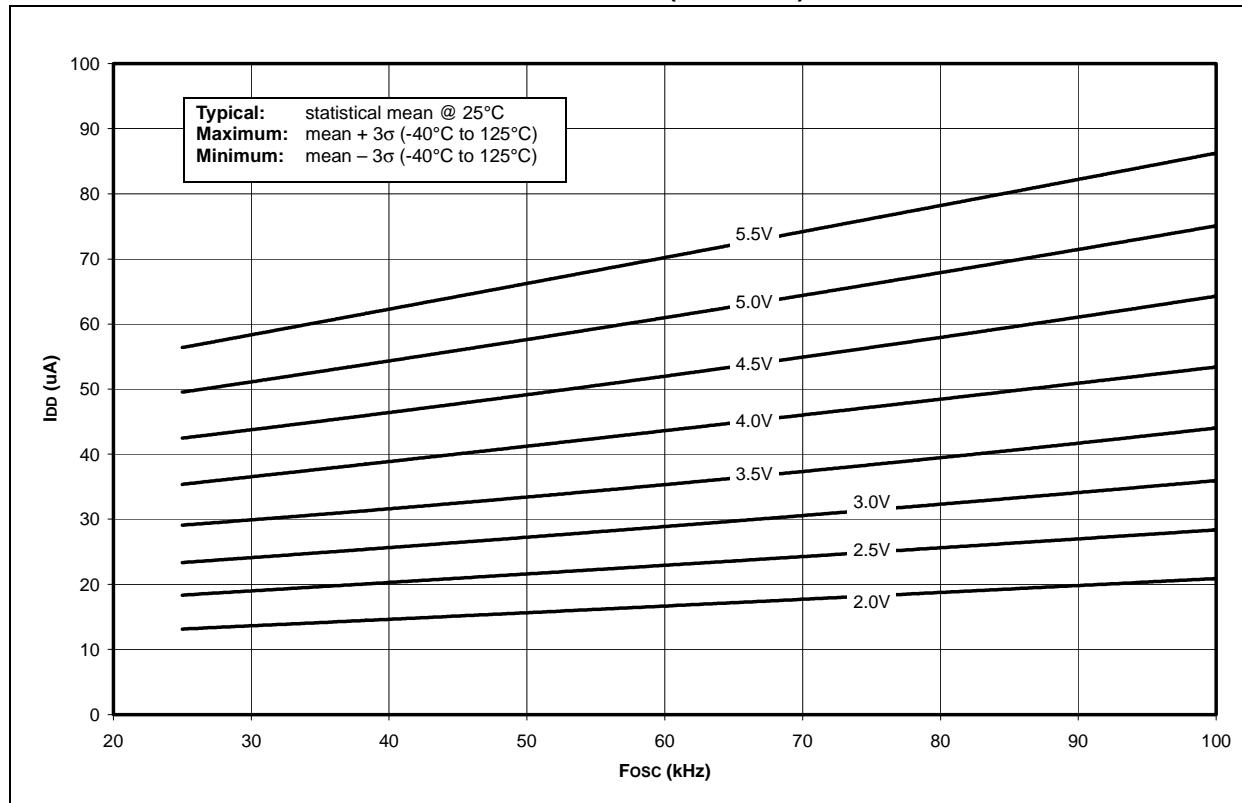
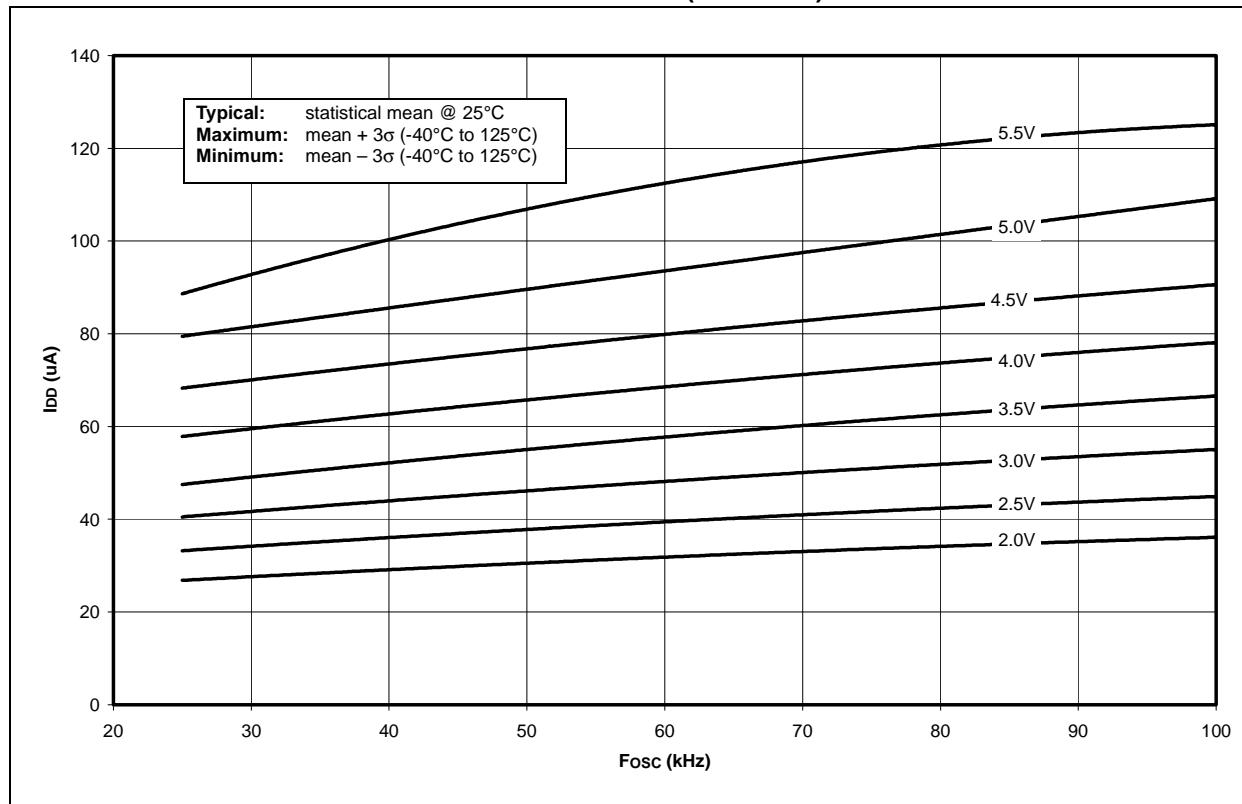
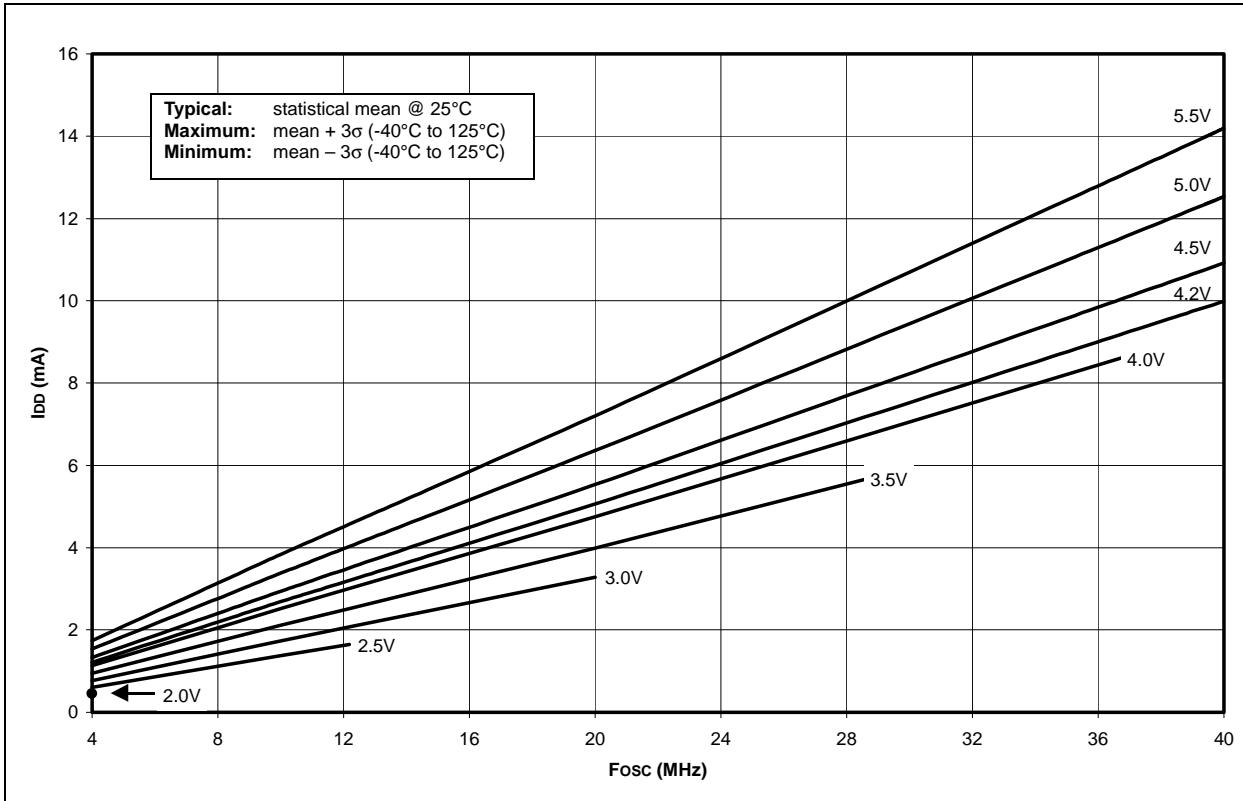
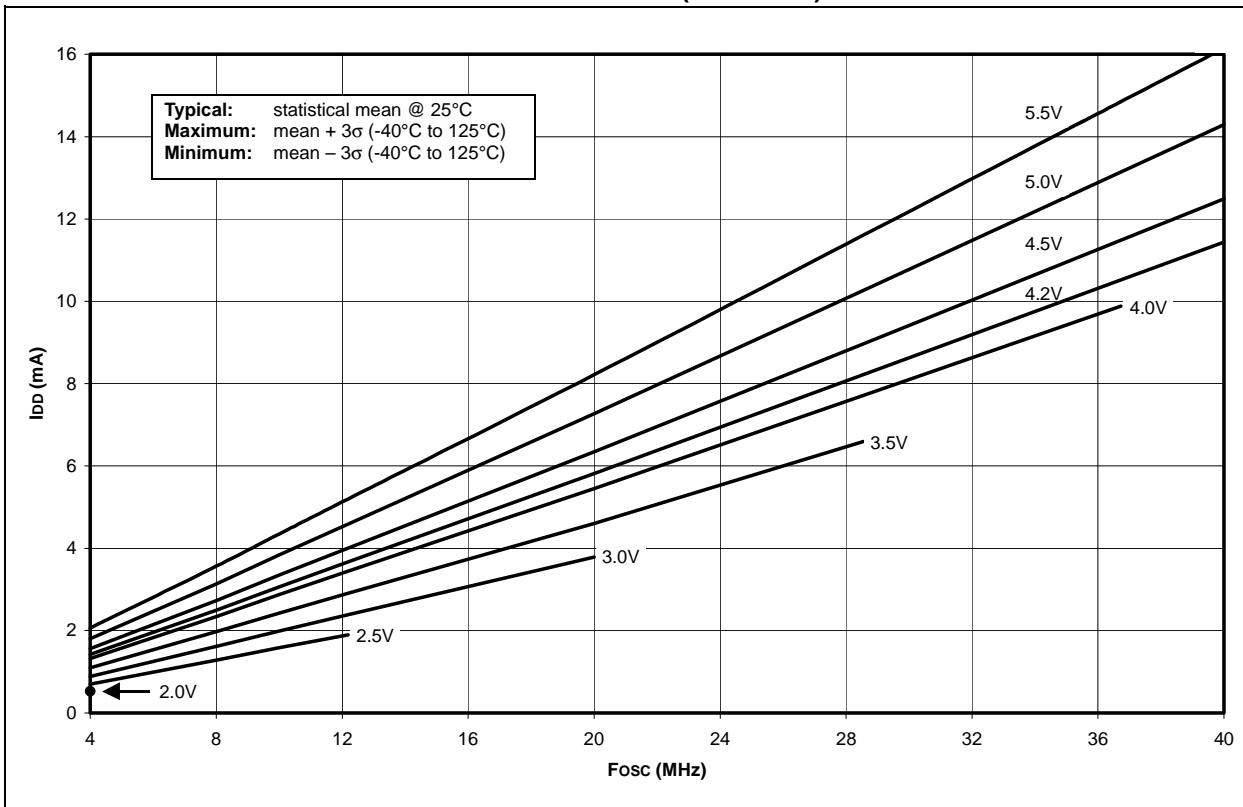


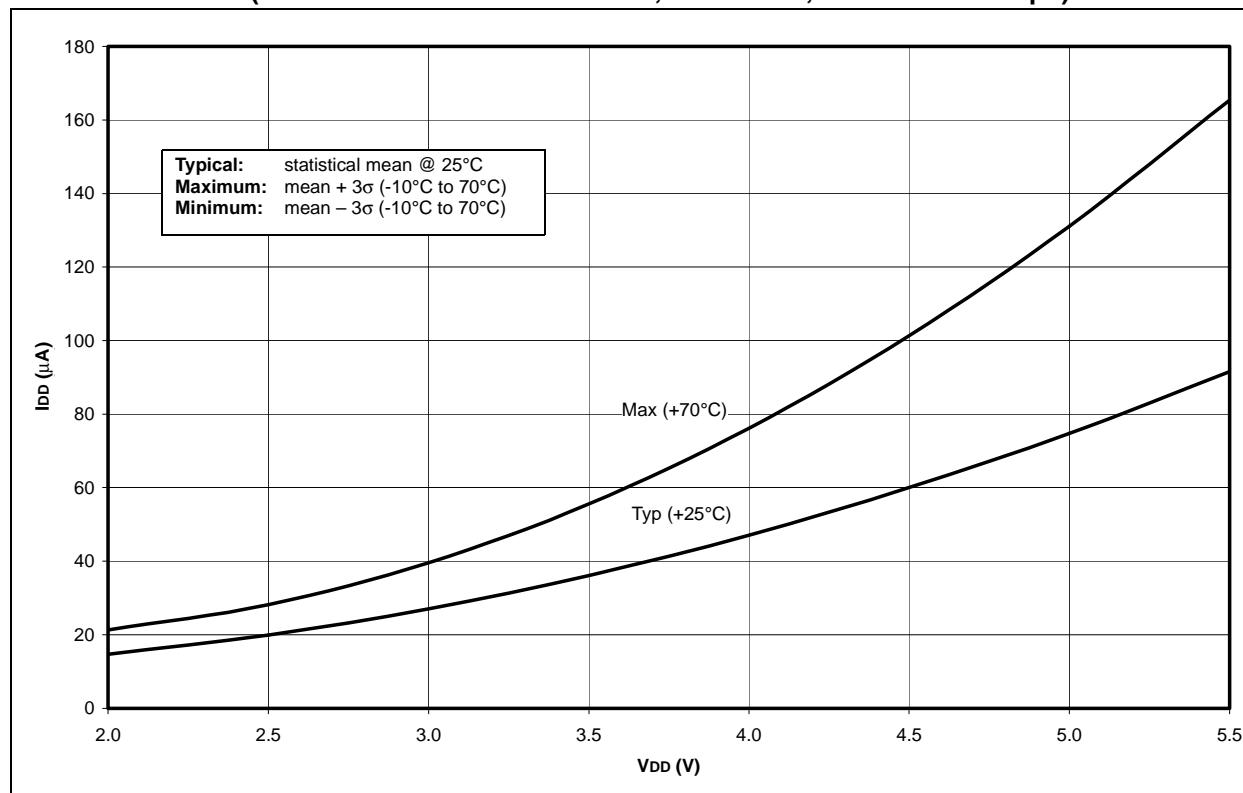
FIGURE 23-8: MAXIMUM IDD vs. Fosc OVER VDD (LP MODE)



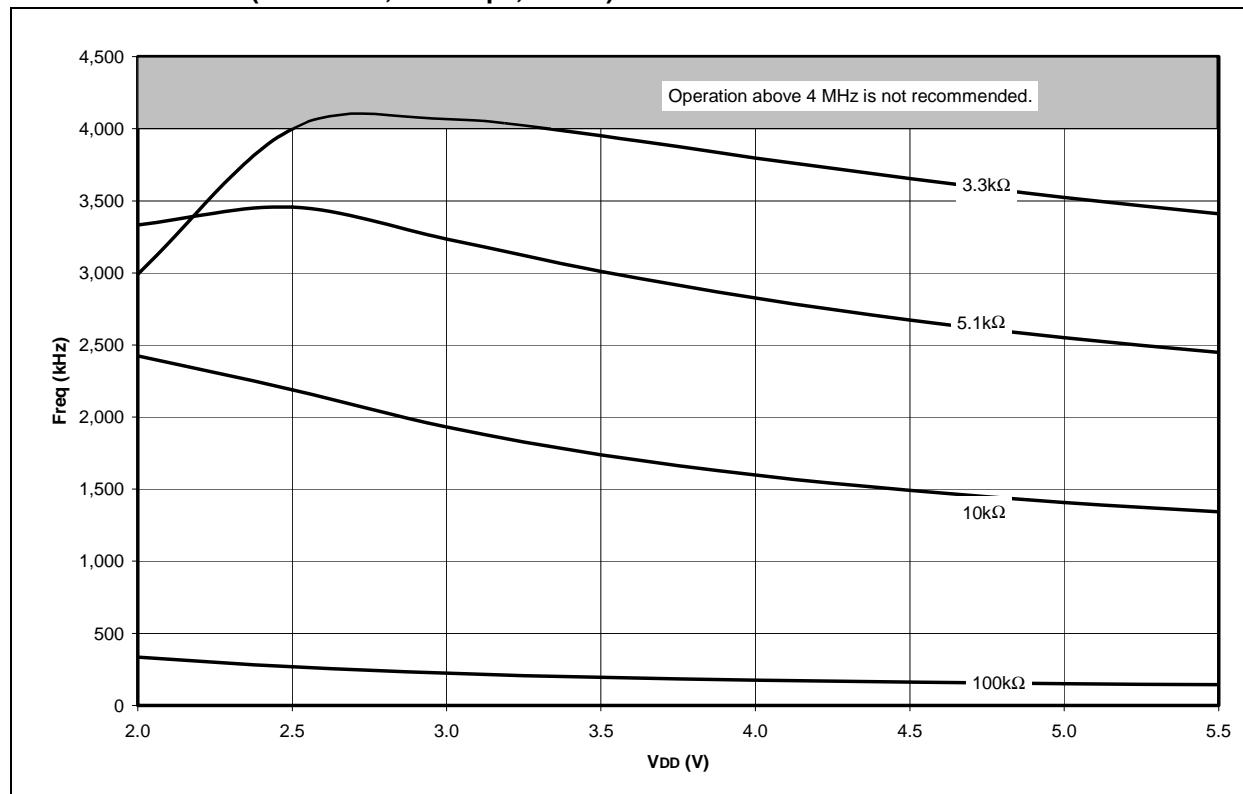
**FIGURE 23-9: TYPICAL IDD VS. FOSC OVER VDD (EC MODE)****FIGURE 23-10: MAXIMUM IDD VS. FOSC OVER VDD (EC MODE)**

# PIC18FXX2

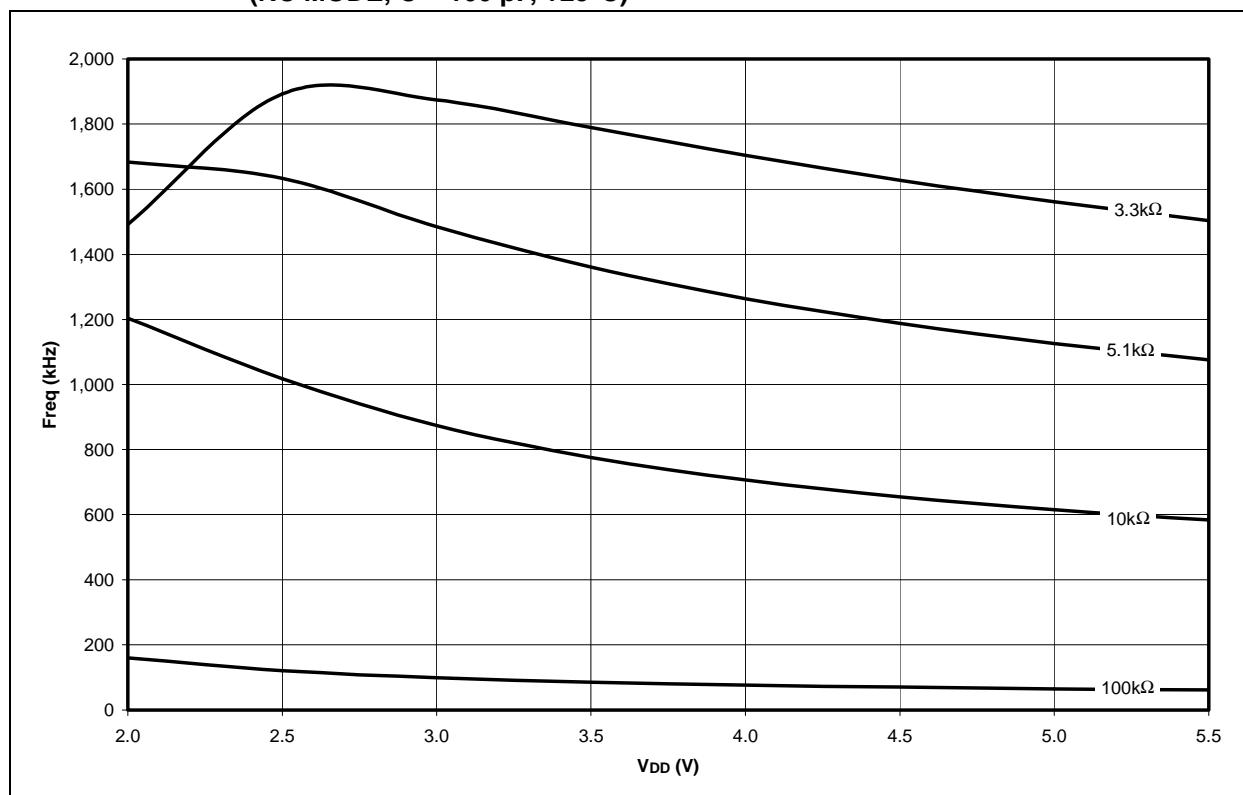
**FIGURE 23-11: TYPICAL AND MAXIMUM IDD vs. VDD  
(TIMER1 AS MAIN OSCILLATOR, 32.768 kHz, C1 AND C2 = 47 pF)**



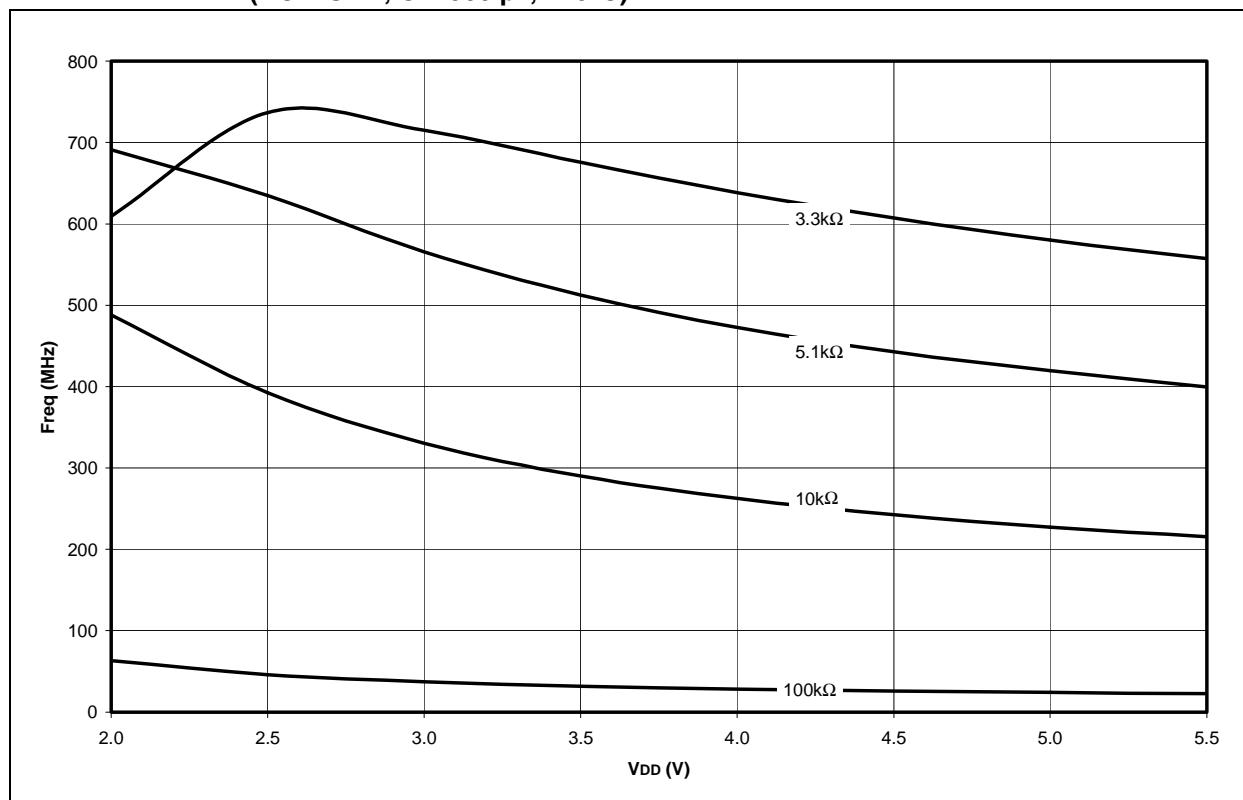
**FIGURE 23-12: AVERAGE FOSC vs. VDD FOR VARIOUS VALUES OF R  
(RC MODE, C = 20 pF, +25°C)**



**FIGURE 23-13:** AVERAGE Fosc vs. VDD FOR VARIOUS VALUES OF R  
(RC MODE, C = 100 pF, +25°C)

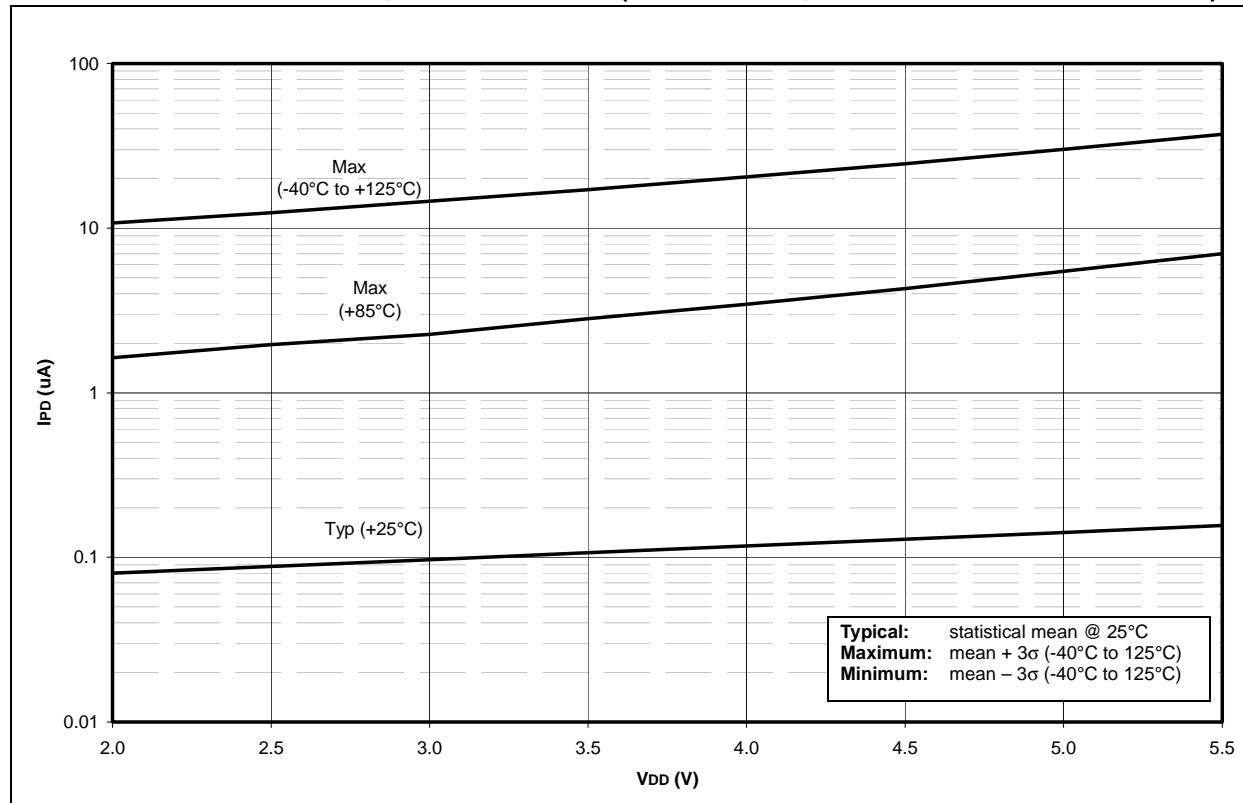


**FIGURE 23-14:** AVERAGE Fosc vs. VDD FOR VARIOUS VALUES OF R  
(RC MODE, C = 300 pF, +25°C)

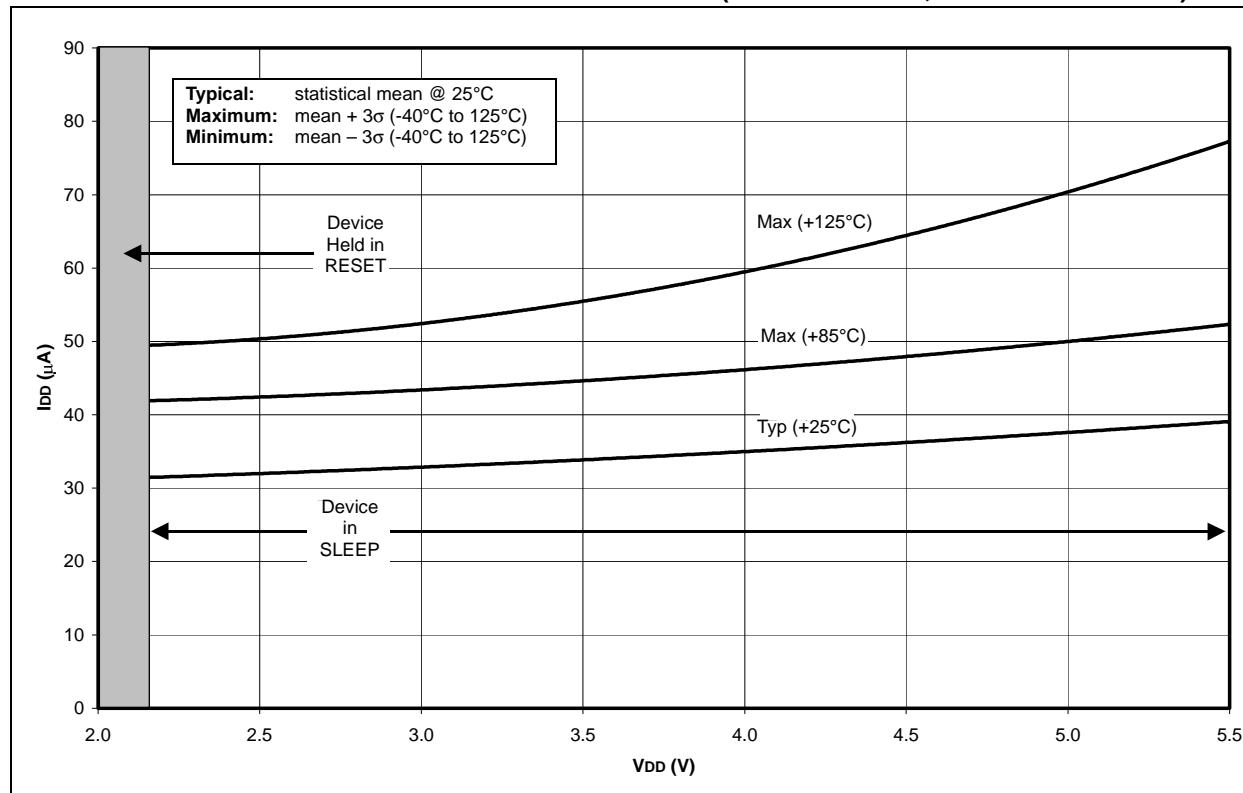


# PIC18FXX2

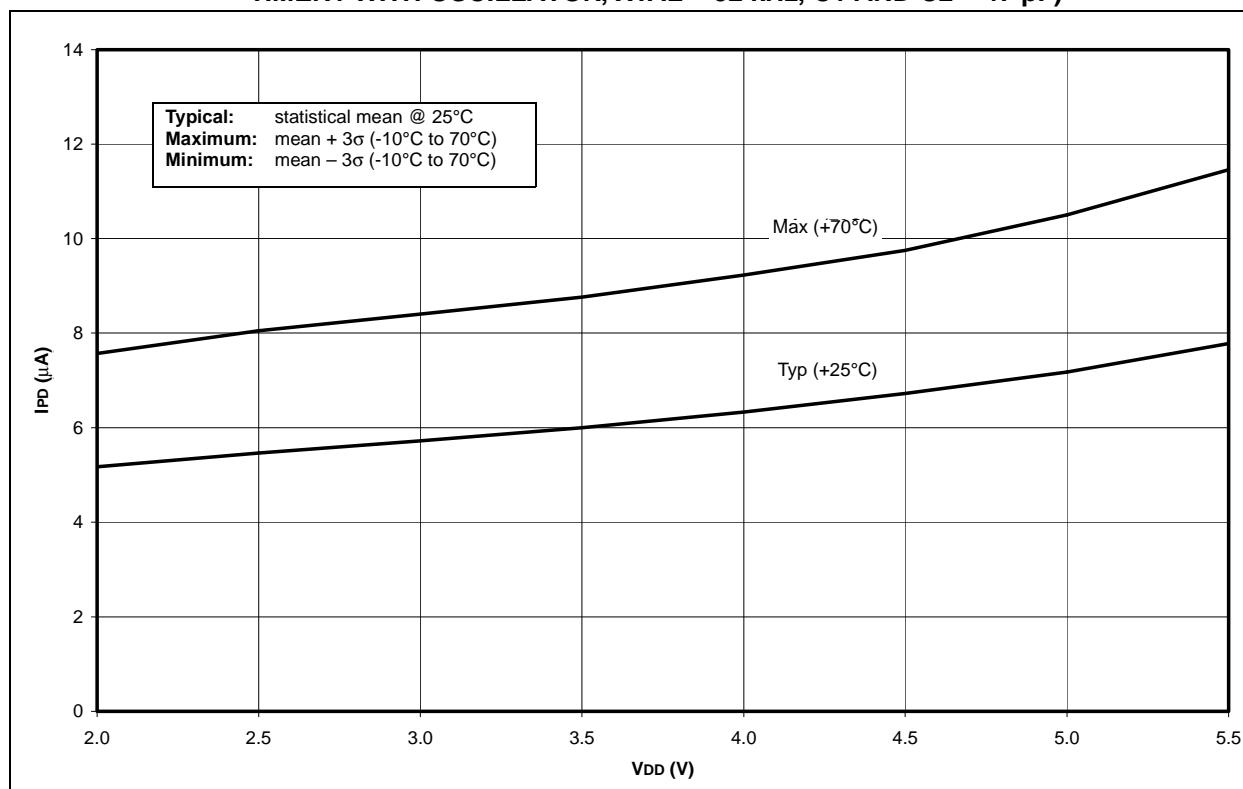
**FIGURE 23-15: IPD VS. VDD, -40°C TO +125°C (SLEEP MODE, ALL PERIPHERALS DISABLED)**



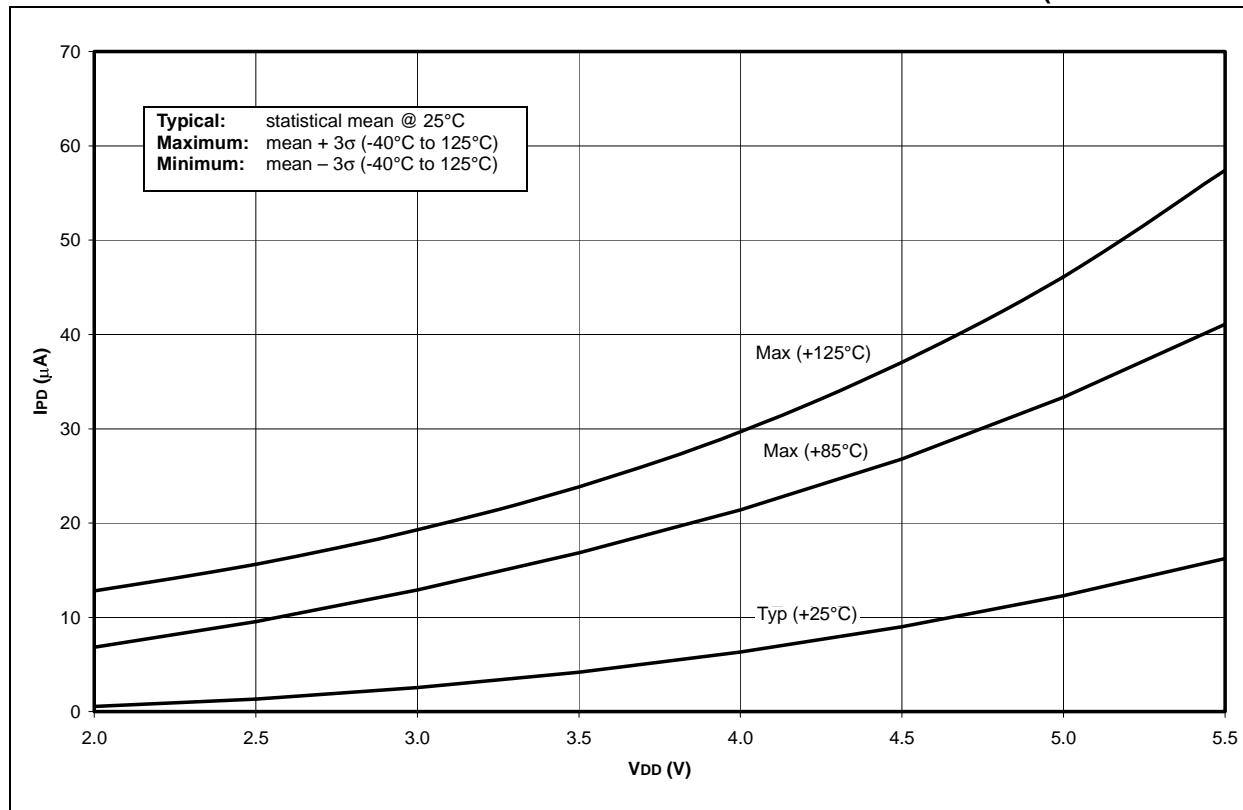
**FIGURE 23-16: ΔIBOR VS. VDD OVER TEMPERATURE (BOR ENABLED, V<sub>BOR</sub> = 2.00 - 2.16V)**



**FIGURE 23-17: TYPICAL AND MAXIMUM  $\Delta I_{TMR1}$  vs. V<sub>DD</sub> OVER TEMPERATURE (-10°C TO +70°C, TIMER1 WITH OSCILLATOR, XTAL = 32 kHz, C<sub>1</sub> AND C<sub>2</sub> = 47 pF)**

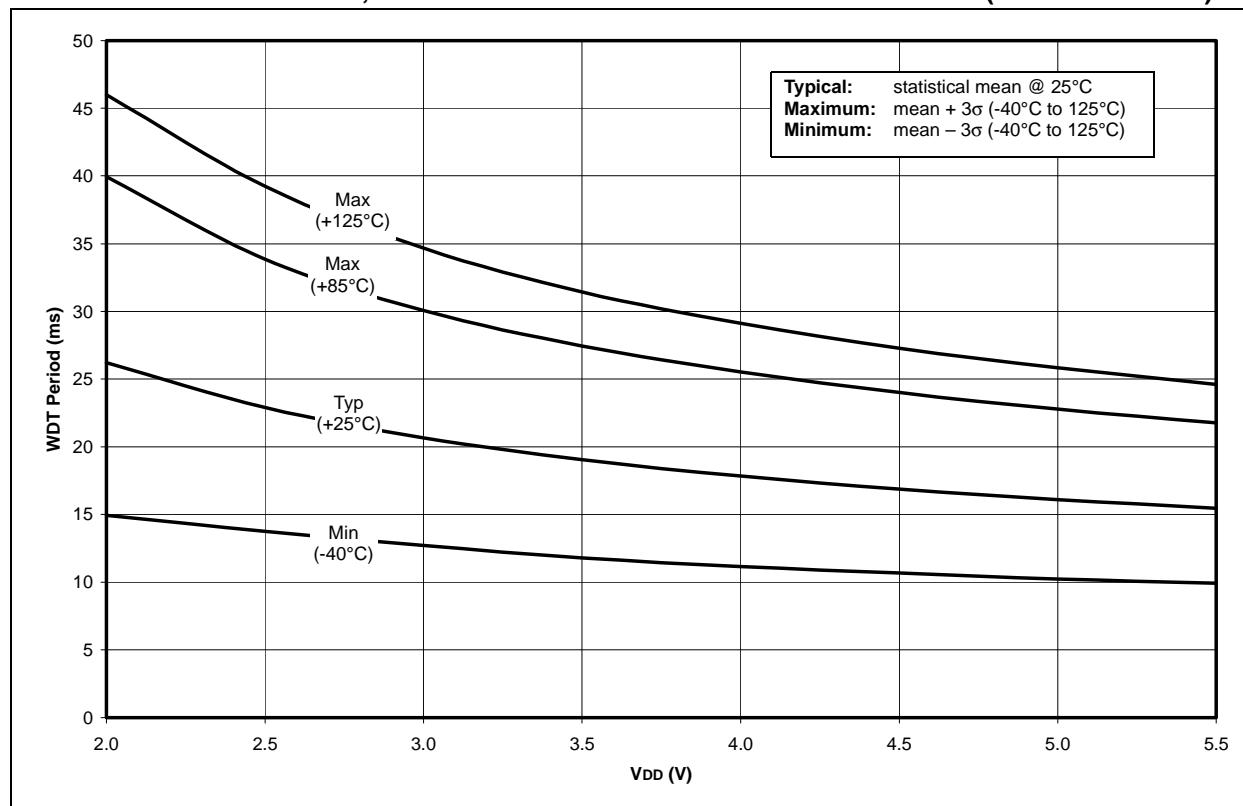


**FIGURE 23-18: TYPICAL AND MAXIMUM  $\Delta I_{WDT}$  vs. V<sub>DD</sub> OVER TEMPERATURE (WDT ENABLED)**

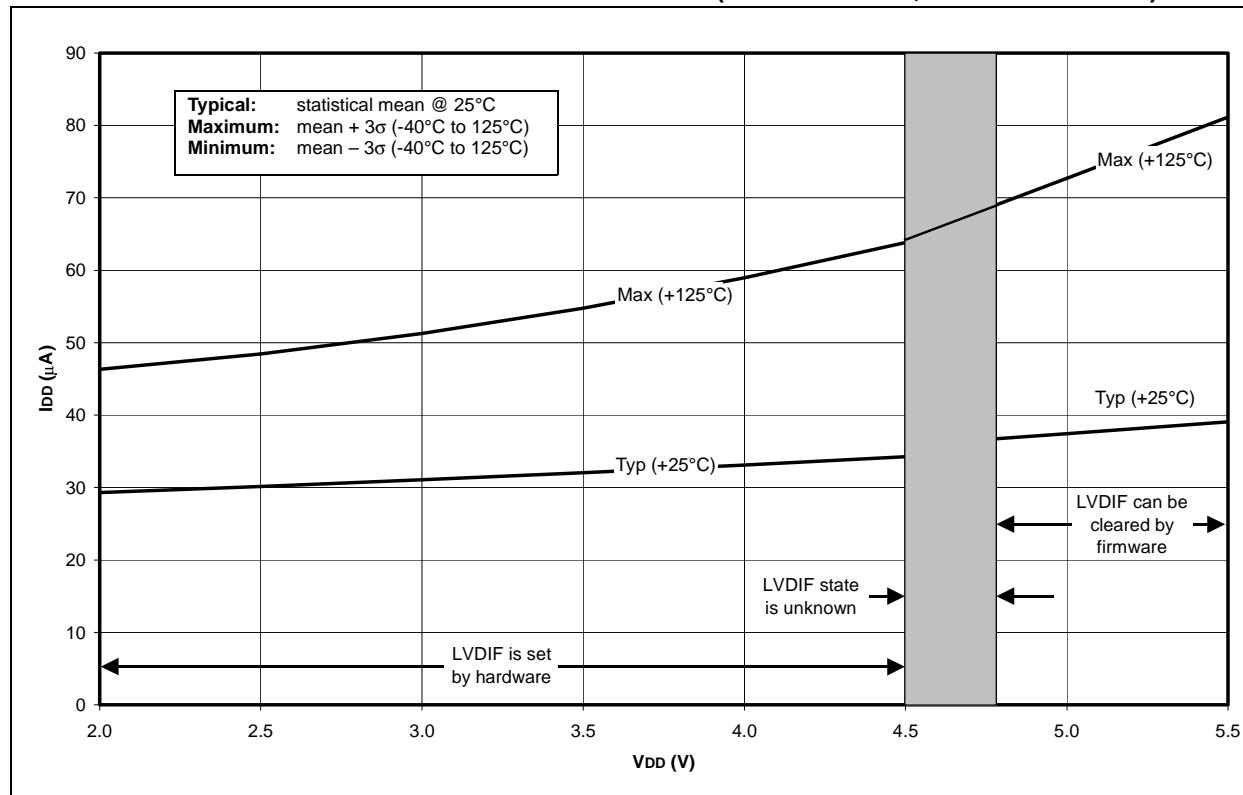


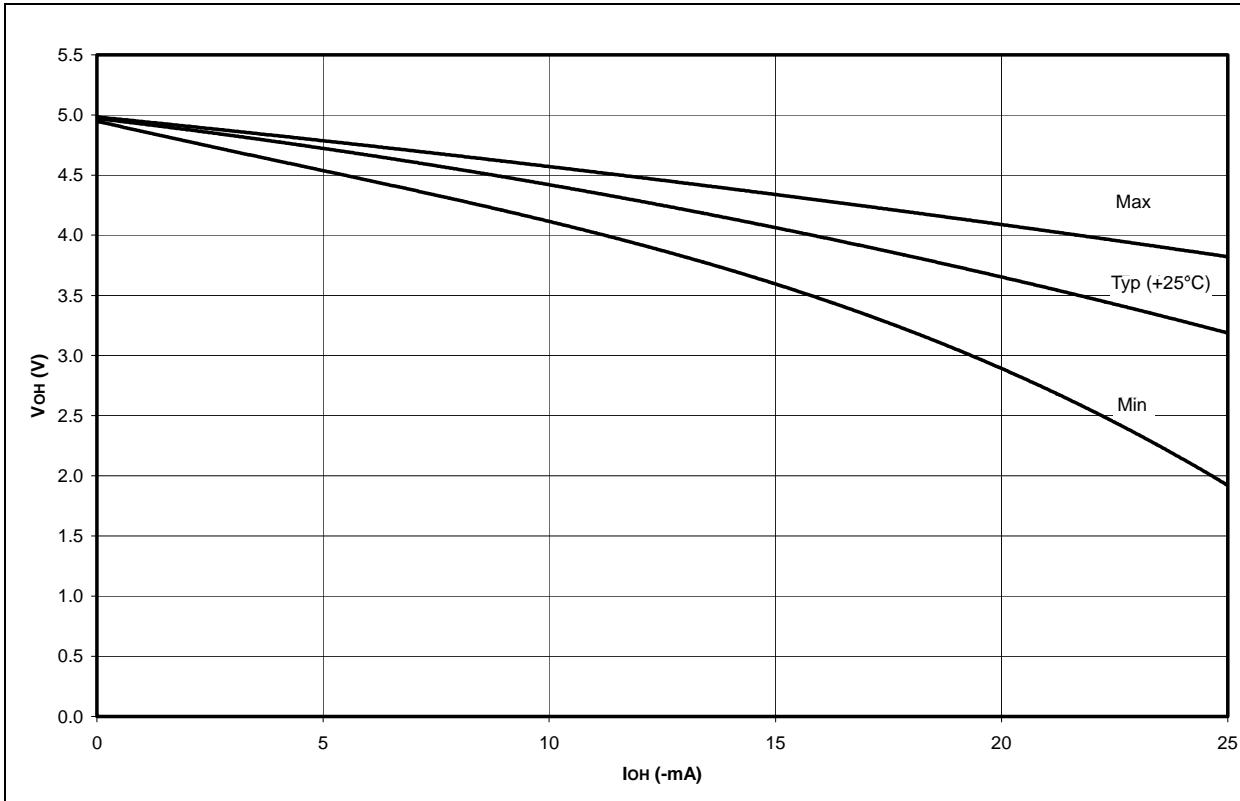
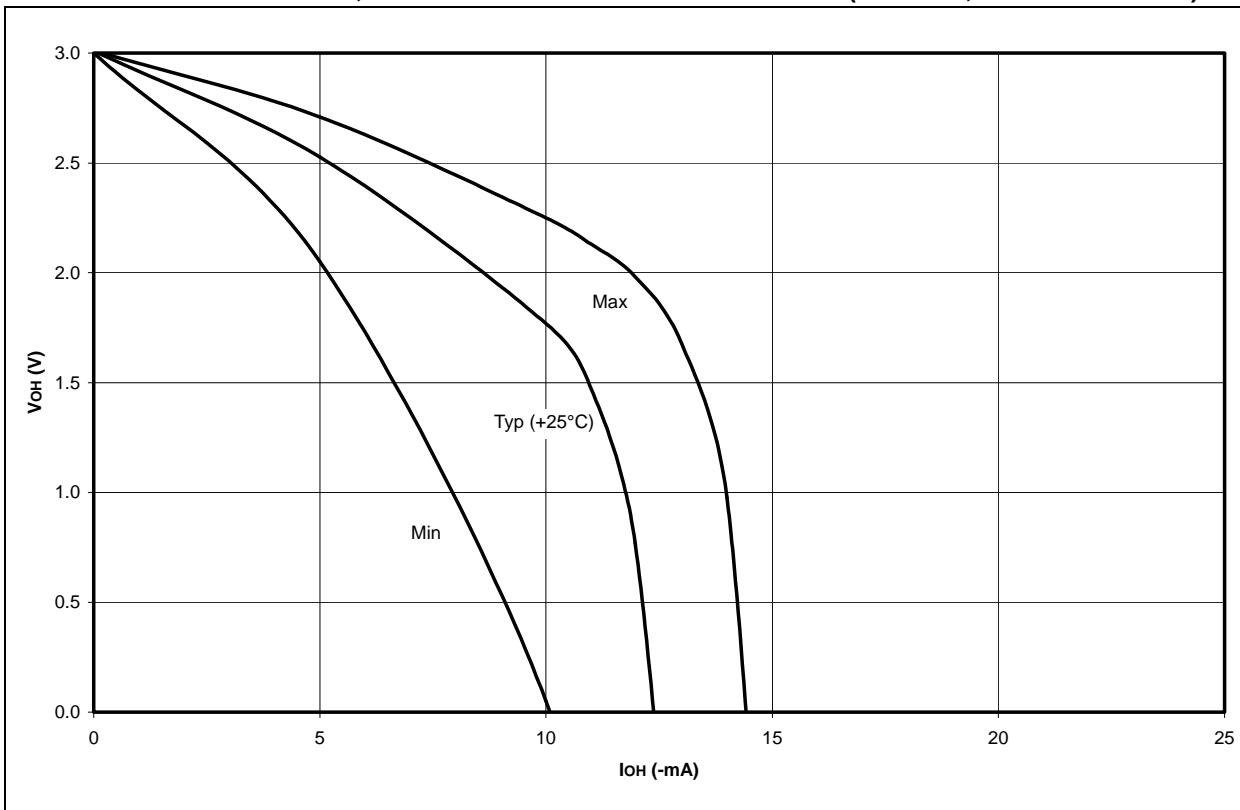
# PIC18FXX2

**FIGURE 23-19: TYPICAL, MINIMUM AND MAXIMUM WDT PERIOD vs. VDD (-40°C TO +125°C)**



**FIGURE 23-20:  $\Delta I_{LVD}$  vs. VDD OVER TEMPERATURE (LVD ENABLED,  $V_{LVD} = 4.5 - 4.78V$ )**



**FIGURE 23-21: TYPICAL, MINIMUM AND MAXIMUM  $V_{OH}$  vs.  $I_{OH}$  ( $V_{DD} = 5V$ ,  $-40^{\circ}C$  TO  $+125^{\circ}C$ )****FIGURE 23-22: TYPICAL, MINIMUM AND MAXIMUM  $V_{OH}$  vs.  $I_{OH}$  ( $V_{DD} = 3V$ ,  $-40^{\circ}C$  TO  $+125^{\circ}C$ )**

# PIC18FXX2

FIGURE 23-23: TYPICAL AND MAXIMUM VOL VS. IOL (VDD = 5V, -40°C TO +125°C)

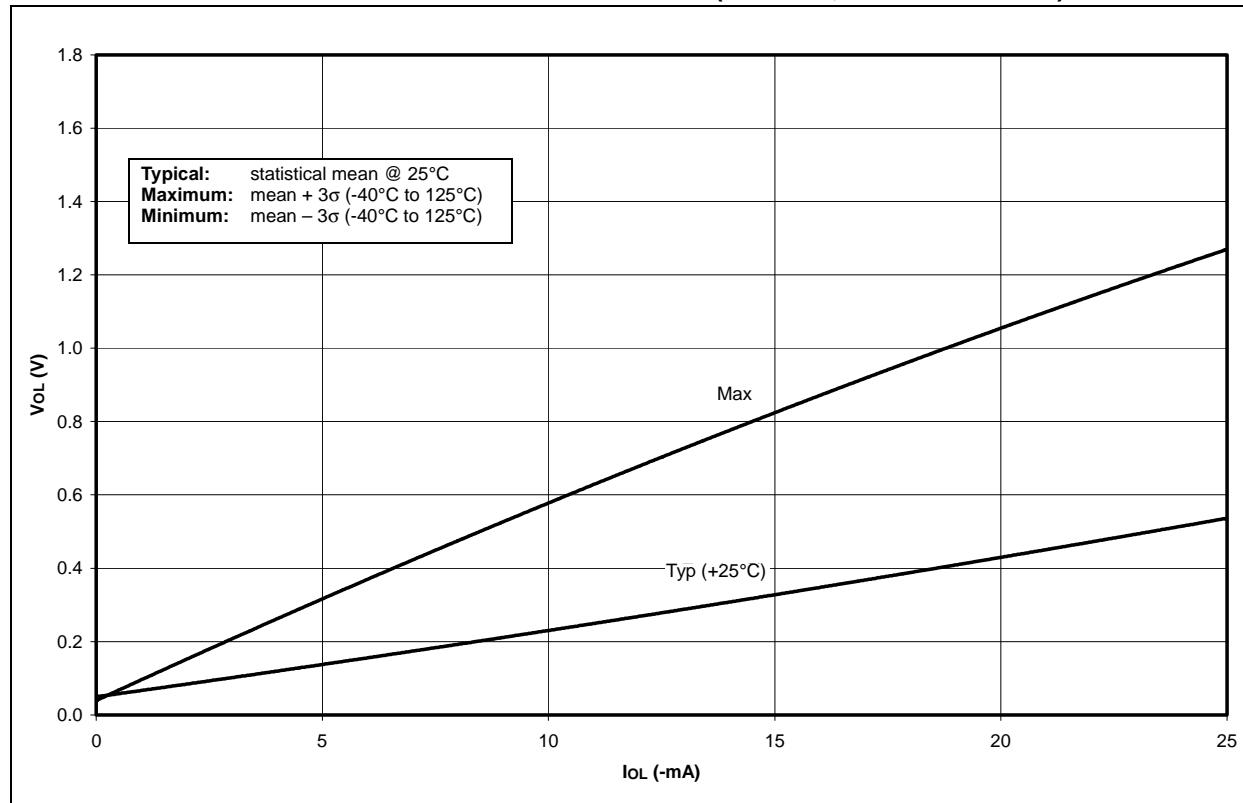


FIGURE 23-24: TYPICAL AND MAXIMUM VOL VS. IOL (VDD = 3V, -40°C TO +125°C)

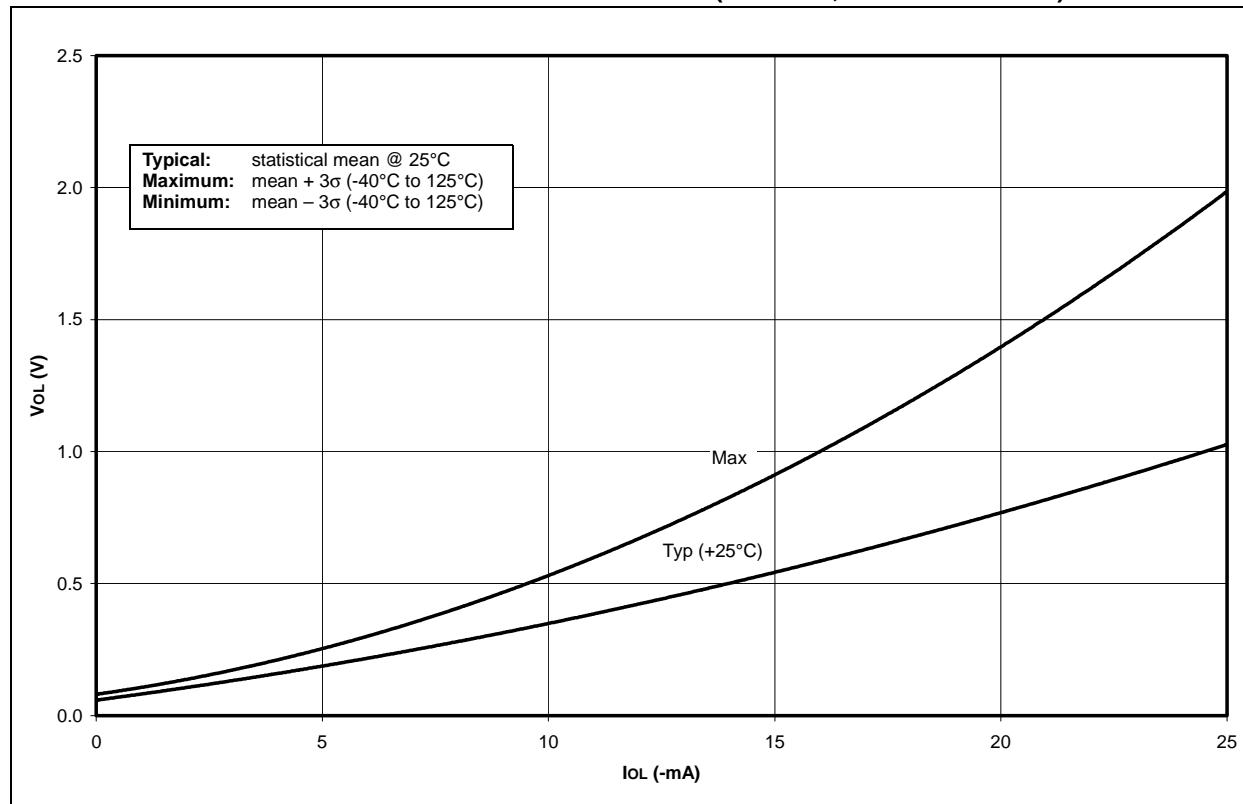


FIGURE 23-25: MINIMUM AND MAXIMUM VIN VS. VDD (ST INPUT, -40°C TO +125°C)

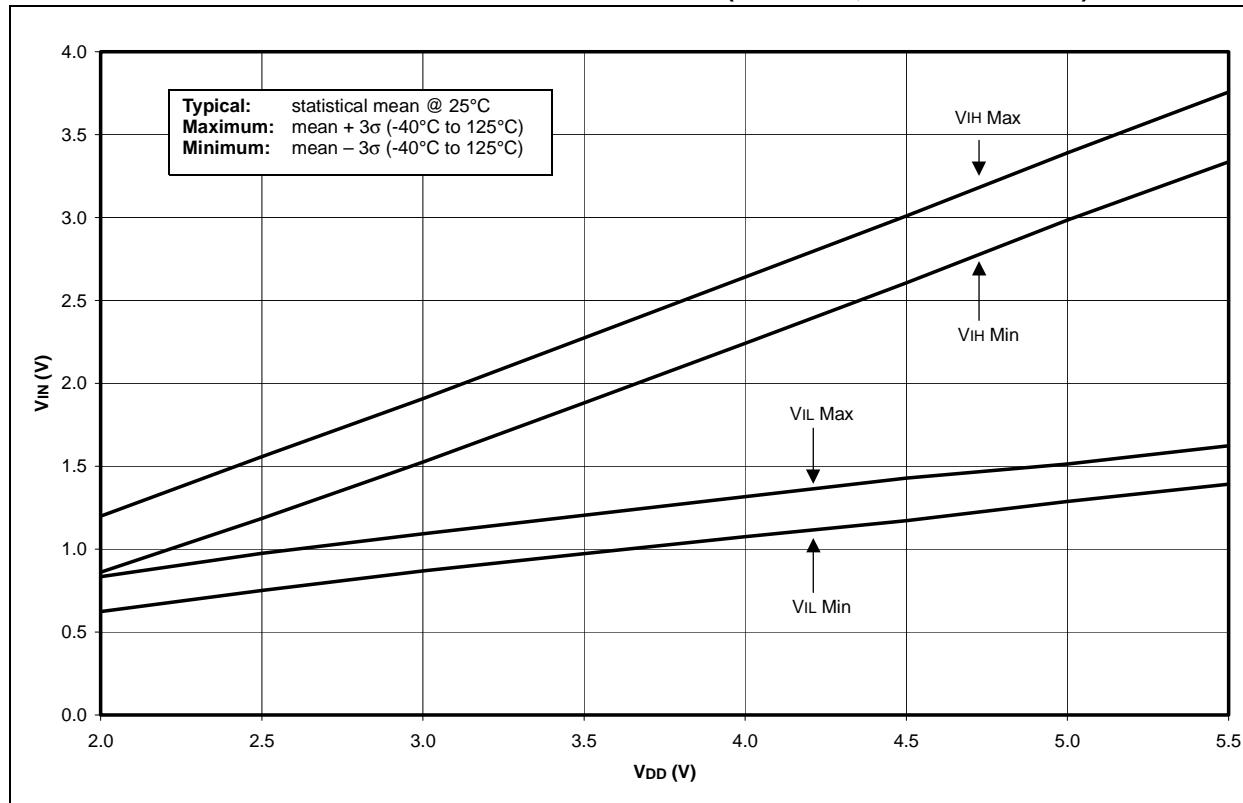
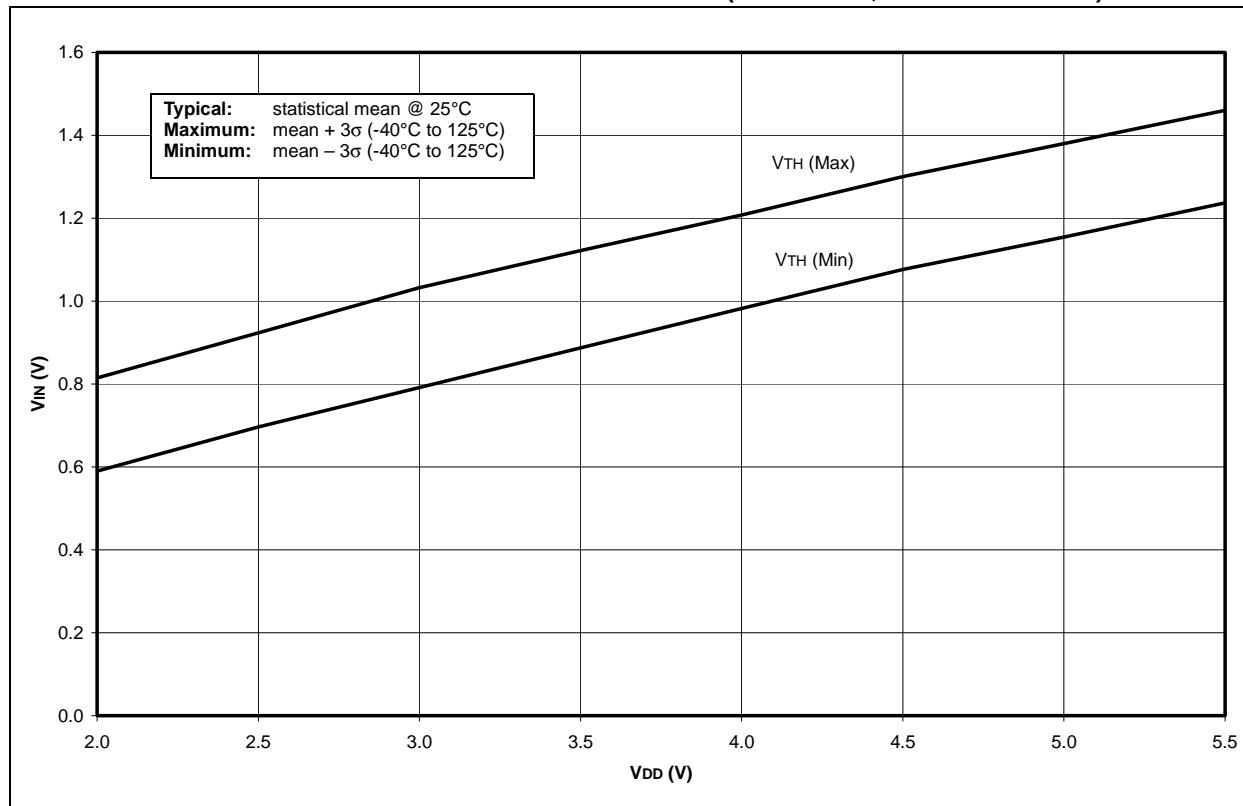


FIGURE 23-26: MINIMUM AND MAXIMUM VIN VS. VDD (TTL INPUT, -40°C TO +125°C)



# PIC18FXX2

FIGURE 23-27: MINIMUM AND MAXIMUM  $V_{IN}$  VS.  $V_{DD}$  ( $I^2C$  INPUT,  $-40^\circ C$  TO  $+125^\circ C$ )

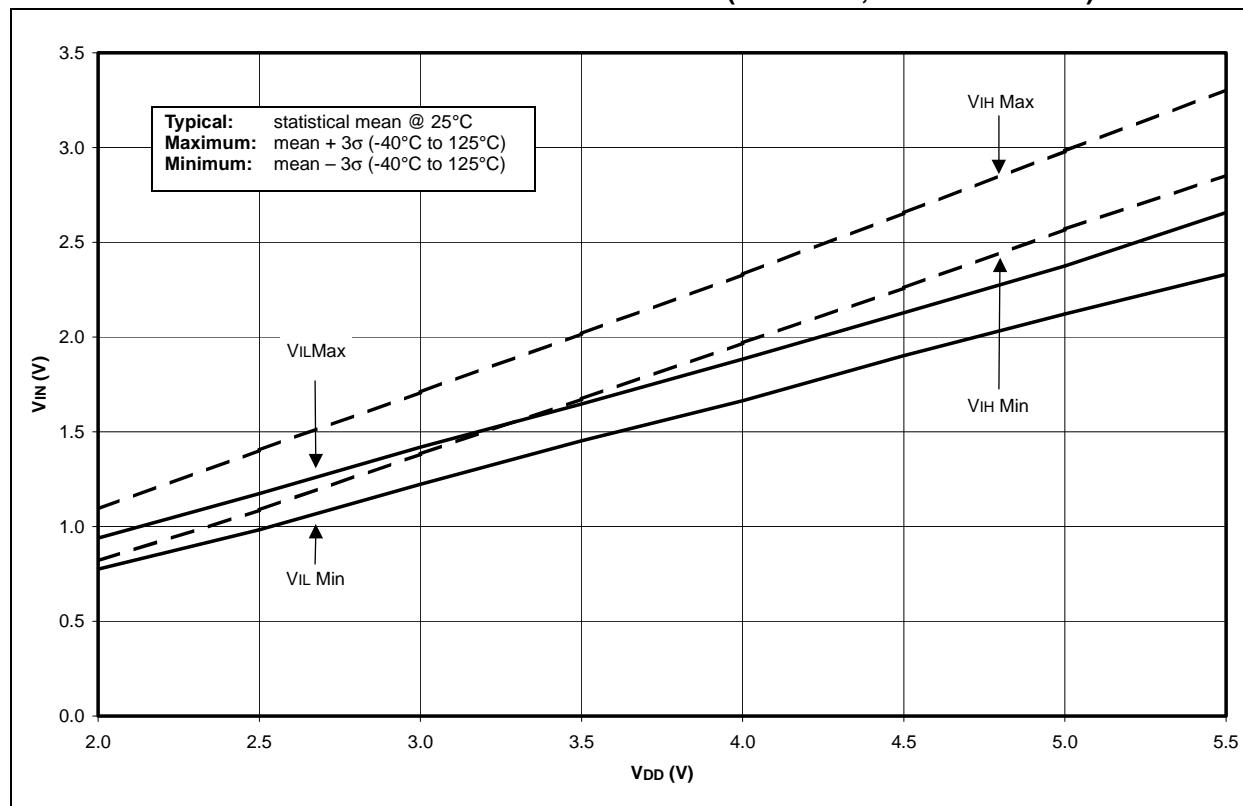
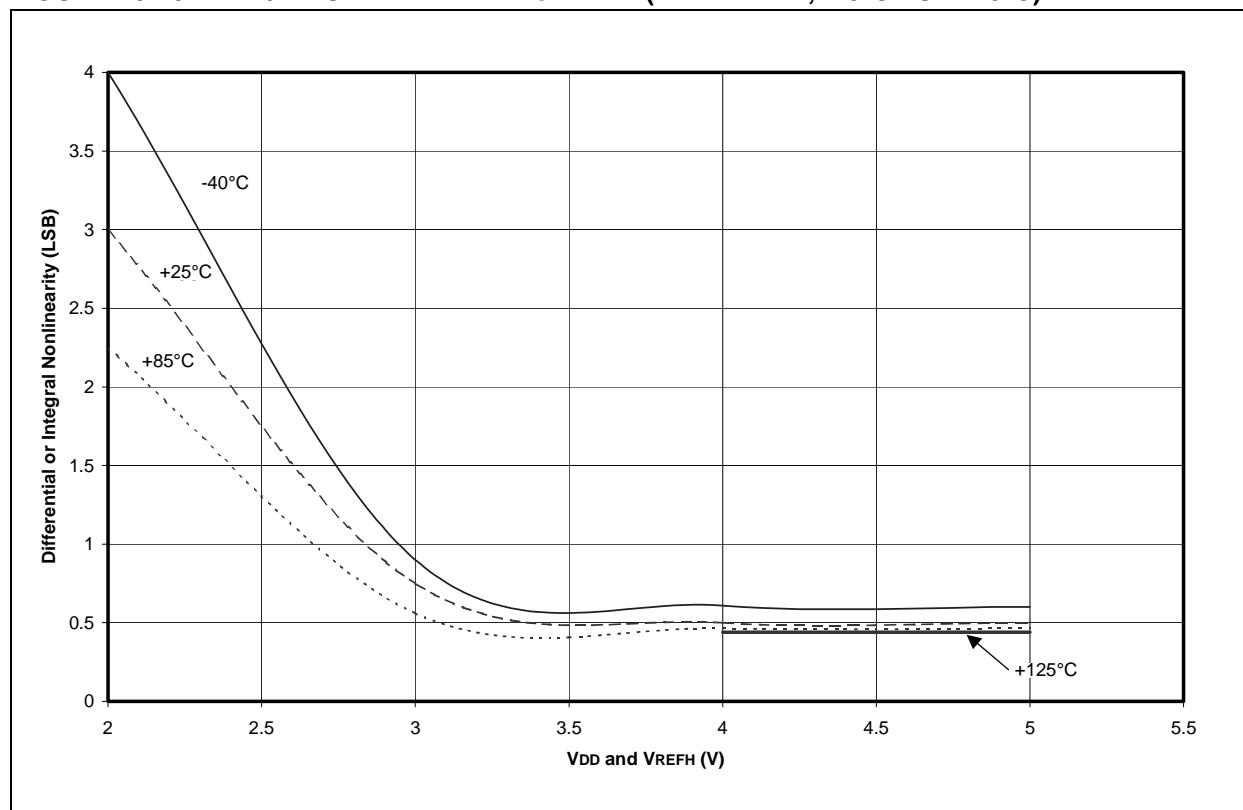
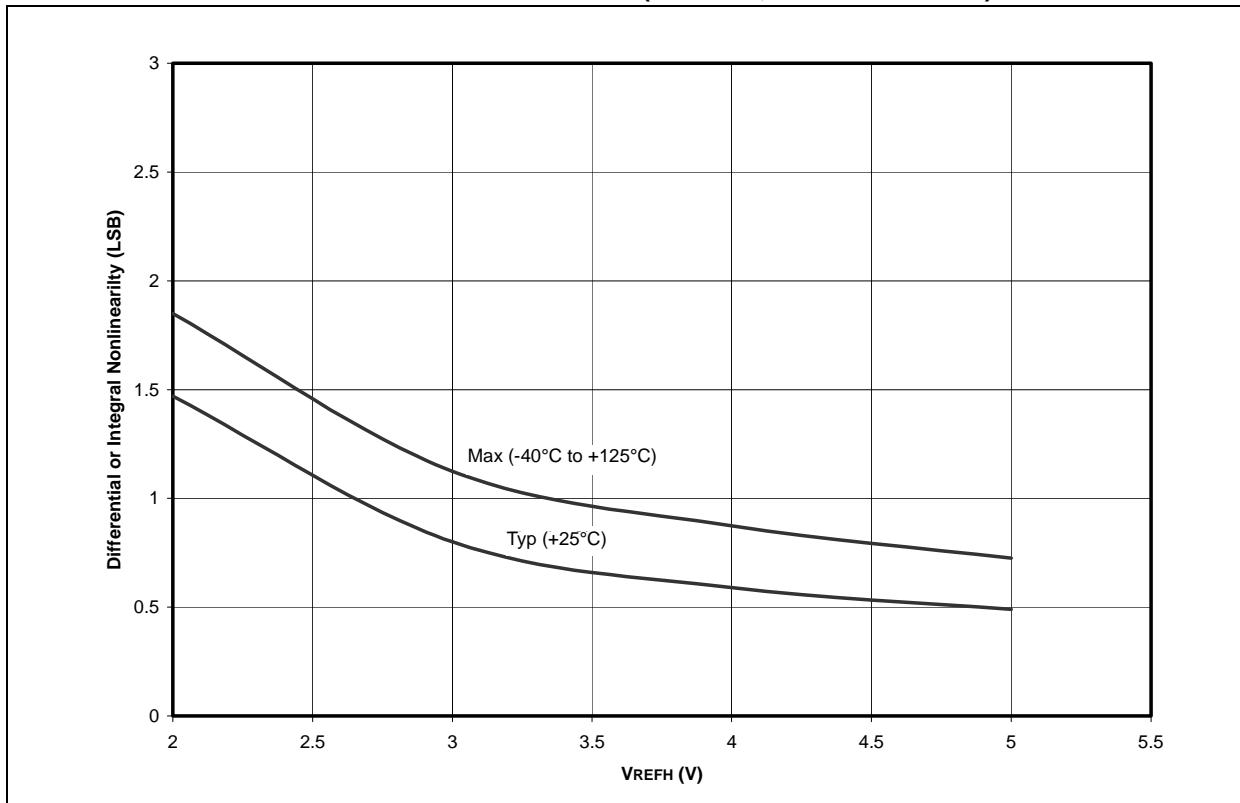


FIGURE 23-28: A/D NON-LINEARITY VS.  $V_{REFH}$  ( $V_{DD} = V_{REFH}$ ,  $-40^\circ C$  TO  $+125^\circ C$ )



**FIGURE 23-29: A/D NON-LINEARITY vs. VREFH ( $V_{DD} = 5V$ ,  $-40^{\circ}\text{C}$  TO  $+125^{\circ}\text{C}$ )**

# **PIC18FXX2**

---

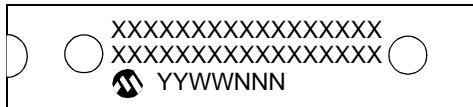
---

## **NOTES:**

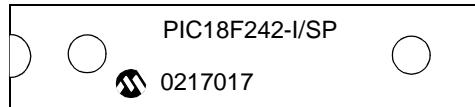
## 24.0 PACKAGING INFORMATION

### 24.1 Package Marking Information

28-Lead PDIP (Skinny DIP)



Example



28-Lead SOIC



Example



**Legend:**

XX...X	Customer specific information*
Y	Year code (last digit of calendar year)
YY	Year code (last 2 digits of calendar year)
WW	Week code (week of January 1 is week '01')
NNN	Alphanumeric traceability code

**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.

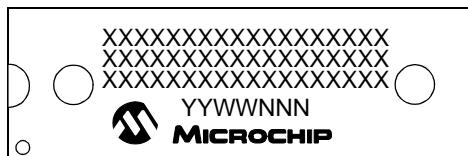
- \* Standard PICmicro device marking consists of Microchip part number, year code, week code, and traceability code. For PICmicro device marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

# PIC18FXX2

---

## Package Marking Information (Cont'd)

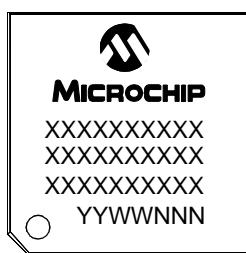
40-Lead PDIP



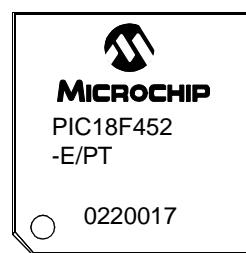
Example



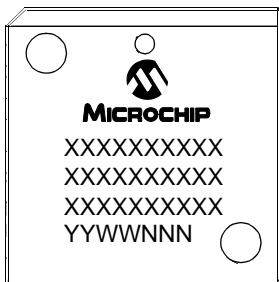
44-Lead TQFP



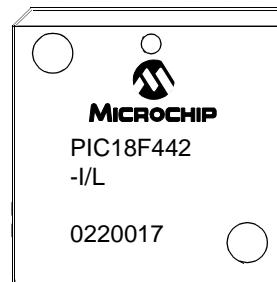
Example



44-Lead PLCC



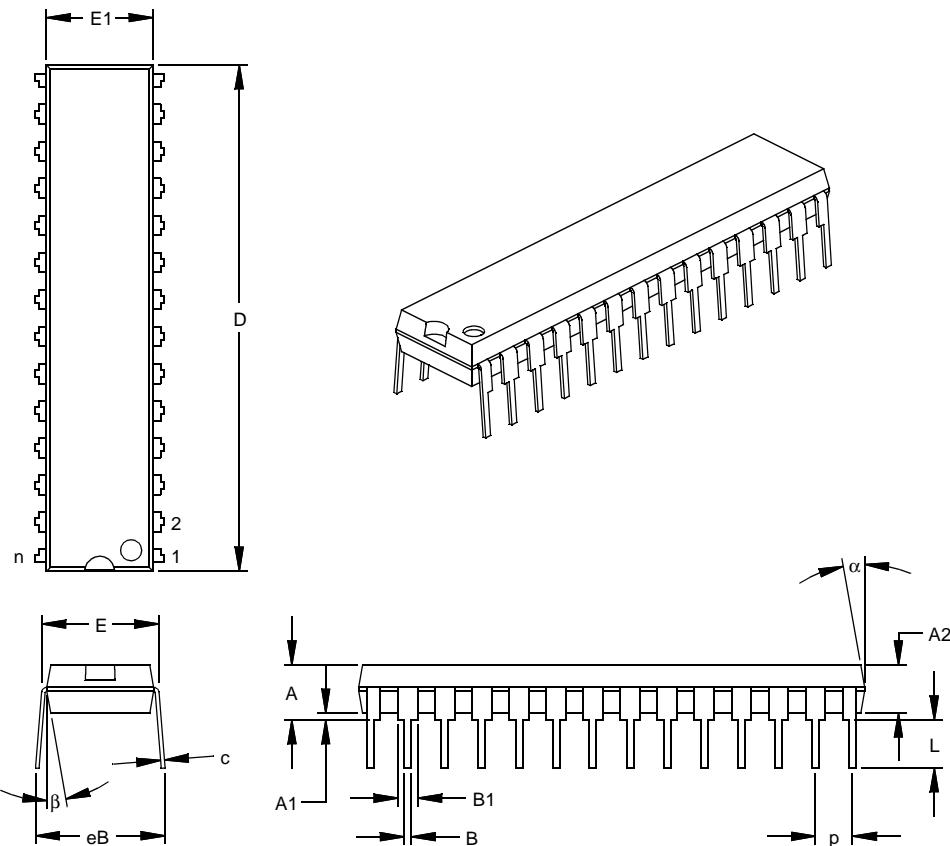
Example



## 24.2 Package Details

The following sections give the technical details of the packages.

### 28-Lead Skinny Plastic Dual In-line (SP) – 300 mil (PDIP)



		Units	INCHES*			MILLIMETERS		
Dimension Limits			MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n			28			28	
Pitch	p			.100			2.54	
Top to Seating Plane	A	.140	.150	.160	3.56	3.81	4.06	
Molded Package Thickness	A2	.125	.130	.135	3.18	3.30	3.43	
Base to Seating Plane	A1	.015			0.38			
Shoulder to Shoulder Width	E	.300	.310	.325	7.62	7.87	8.26	
Molded Package Width	E1	.275	.285	.295	6.99	7.24	7.49	
Overall Length	D	1.345	1.365	1.385	34.16	34.67	35.18	
Tip to Seating Plane	L	.125	.130	.135	3.18	3.30	3.43	
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38	
Upper Lead Width	B1	.040	.053	.065	1.02	1.33	1.65	
Lower Lead Width	B	.016	.019	.022	0.41	0.48	0.56	
Overall Row Spacing	§	eB	.320	.350	.430	8.13	8.89	10.92
Mold Draft Angle Top	alpha	5	10	15	5	10	15	
Mold Draft Angle Bottom	beta	5	10	15	5	10	15	

\* Controlling Parameter

§ Significant Characteristic

Notes:

Dimension D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

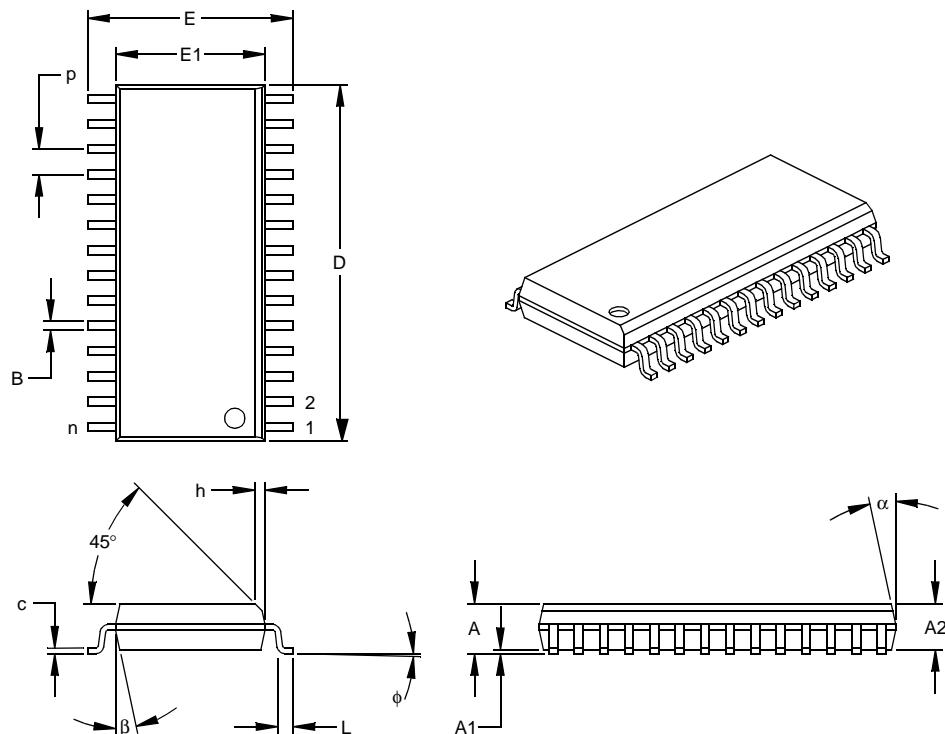
JEDEC Equivalent: MO-095

Drawing No. C04-070

# PIC18FXX2

---

## 28-Lead Plastic Small Outline (SO) – Wide, 300 mil (SOIC)



Dimension	Units	INCHES*			MILLIMETERS		
		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.050			1.27	
Overall Height	A	.093	.099	.104	2.36	2.50	2.64
Molded Package Thickness	A2	.088	.091	.094	2.24	2.31	2.39
Standoff §	A1	.004	.008	.012	0.10	0.20	0.30
Overall Width	E	.394	.407	.420	10.01	10.34	10.67
Molded Package Width	E1	.288	.295	.299	7.32	7.49	7.59
Overall Length	D	.695	.704	.712	17.65	17.87	18.08
Chamfer Distance	h	.010	.020	.029	0.25	0.50	0.74
Foot Length	L	.016	.033	.050	0.41	0.84	1.27
Foot Angle Top	φ	0	4	8	0	4	8
Lead Thickness	c	.009	.011	.013	0.23	0.28	0.33
Lead Width	B	.014	.017	.020	0.36	0.42	0.51
Mold Draft Angle Top	α	0	12	15	0	12	15
Mold Draft Angle Bottom	β	0	12	15	0	12	15

\* Controlling Parameter

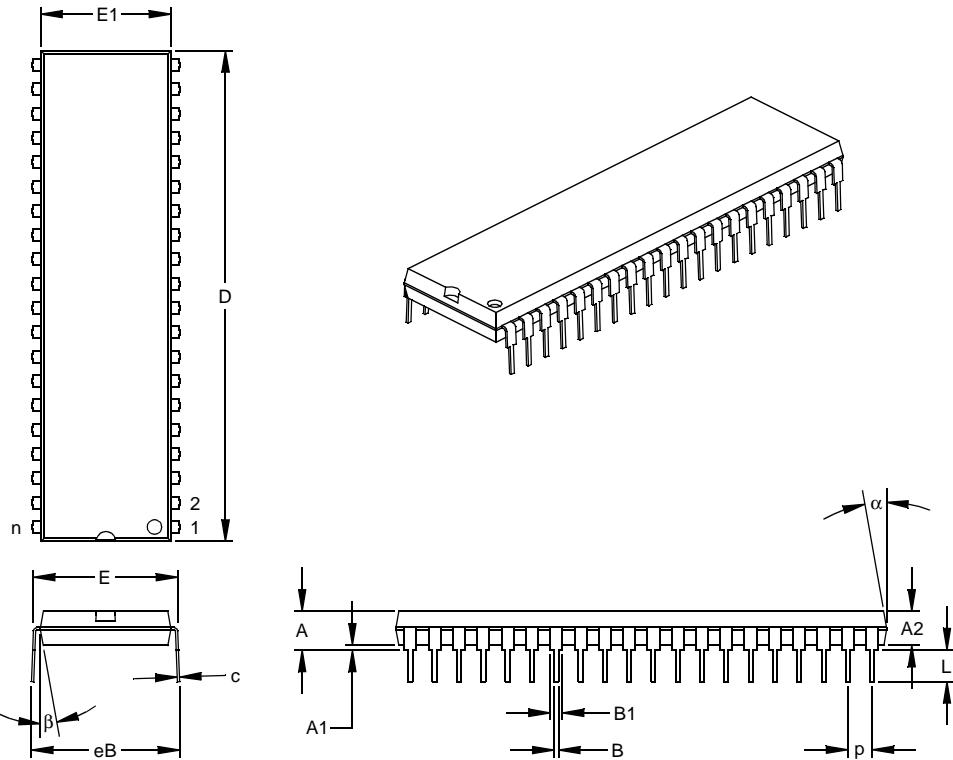
§ Significant Characteristic

### Notes:

Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

JEDEC Equivalent: MS-013  
Drawing No. C04-052

## 40-Lead Plastic Dual In-line (P) – 600 mil (PDIP)



Dimension	Limits	INCHES*			MILLIMETERS			
		MIN	NOM	MAX	MIN	NOM	MAX	
Number of Pins	n		.40			.40		
Pitch	p		.100			2.54		
Top to Seating Plane	A	.160	.175	.190	4.06	4.45	4.83	
Molded Package Thickness	A2	.140	.150	.160	3.56	3.81	4.06	
Base to Seating Plane	A1	.015			0.38			
Shoulder to Shoulder Width	E	.595	.600	.625	15.11	15.24	15.88	
Molded Package Width	E1	.530	.545	.560	13.46	13.84	14.22	
Overall Length	D	2.045	2.058	2.065	51.94	52.26	52.45	
Tip to Seating Plane	L	.120	.130	.135	3.05	3.30	3.43	
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38	
Upper Lead Width	B1	.030	.050	.070	0.76	1.27	1.78	
Lower Lead Width	B	.014	.018	.022	0.36	0.46	0.56	
Overall Row Spacing	§	eB	.620	.650	.680	15.75	16.51	17.27
Mold Draft Angle Top	alpha	5	10	15	5	10	15	
Mold Draft Angle Bottom	beta	5	10	15	5	10	15	

\* Controlling Parameter

§ Significant Characteristic

### Notes:

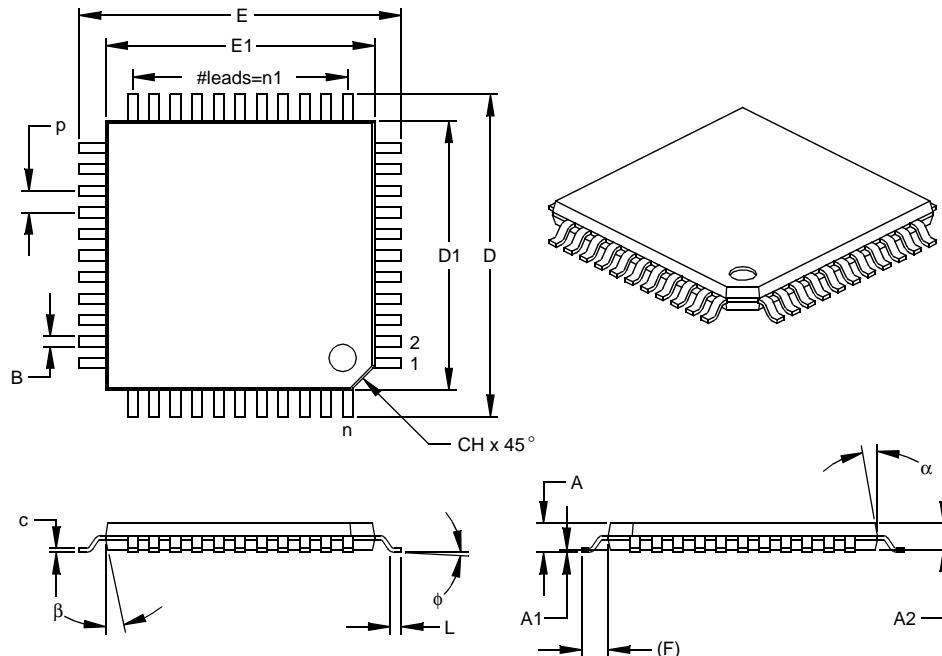
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

JEDEC Equivalent: MO-011

Drawing No. C04-016

# PIC18FXX2

## 44-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)



Dimension Limits	Units	INCHES			MILLIMETERS*		
		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n			.44			.44
Pitch	P		.031				.80
Pins per Side	n1			.11			.11
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	.95	1.00	1.05
Standoff §	A1	.002	.004	.006	.05	.10	.15
Foot Length	L	.018	.024	.030	.45	.60	.75
Footprint (Reference)	(F)		.039		1.00		
Foot Angle	phi	0	3.5	7	0	3.5	7
Overall Width	E	.463	.472	.482	11.75	12.00	12.25
Overall Length	D	.463	.472	.482	11.75	12.00	12.25
Molded Package Width	E1	.390	.394	.398	9.90	10.00	10.10
Molded Package Length	D1	.390	.394	.398	9.90	10.00	10.10
Lead Thickness	c	.004	.006	.008	.09	.15	.20
Lead Width	B	.012	.015	.017	.30	.38	.44
Pin 1 Corner Chamfer	CH	.025	.035	.045	.64	.89	1.14
Mold Draft Angle Top	alpha	5	10	15	5	10	15
Mold Draft Angle Bottom	beta	5	10	15	5	10	15

\* Controlling Parameter

§ Significant Characteristic

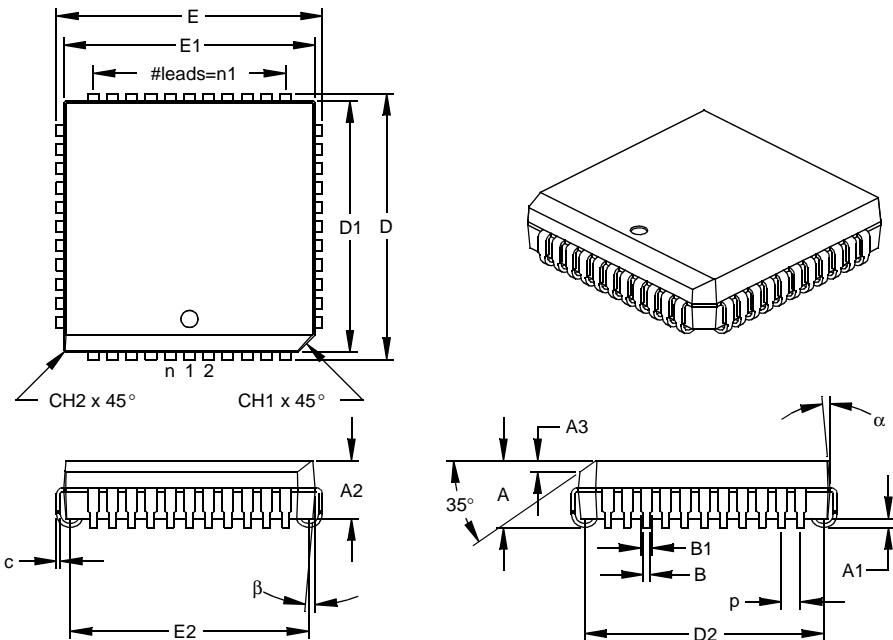
### Notes:

Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

JEDEC Equivalent: MS-026

Drawing No. C04-076

## 44-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)



Units		INCHES*			MILLIMETERS		
Dimension	Limits	MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		44			44	
Pitch	p		.050			1.27	
Pins per Side	n1		11			11	
Overall Height	A	.165	.173	.180	4.19	4.39	4.57
Molded Package Thickness	A2	.145	.153	.160	3.68	3.87	4.06
Standoff §	A1	.020	.028	.035	0.51	0.71	0.89
Side 1 Chamfer Height	A3	.024	.029	.034	0.61	0.74	0.86
Corner Chamfer 1	CH1	.040	.045	.050	1.02	1.14	1.27
Corner Chamfer (others)	CH2	.000	.005	.010	0.00	0.13	0.25
Overall Width	E	.685	.690	.695	17.40	17.53	17.65
Overall Length	D	.685	.690	.695	17.40	17.53	17.65
Molded Package Width	E1	.650	.653	.656	16.51	16.59	16.66
Molded Package Length	D1	.650	.653	.656	16.51	16.59	16.66
Footprint Width	E2	.590	.620	.630	14.99	15.75	16.00
Footprint Length	D2	.590	.620	.630	14.99	15.75	16.00
Lead Thickness	c	.008	.011	.013	0.20	0.27	0.33
Upper Lead Width	B1	.026	.029	.032	0.66	0.74	0.81
Lower Lead Width	B	.013	.020	.021	0.33	0.51	0.53
Mold Draft Angle Top	α	0	5	10	0	5	10
Mold Draft Angle Bottom	β	0	5	10	0	5	10

\* Controlling Parameter

§ Significant Characteristic

Notes:

Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.

JEDEC Equivalent: MO-047

Drawing No. C04-048

# **PIC18FXX2**

---

---

## **NOTES:**

## APPENDIX A: REVISION HISTORY

### Revision A (June 2001)

Original data sheet for the PIC18FXX2 family.

### Revision B (August 2002)

This revision includes the DC and AC Characteristics Graphs and Tables. The Electrical Specifications in Section 22.0 have been updated and there have been minor corrections to the data sheet text.

## APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table B-1.

**TABLE B-1: DEVICE DIFFERENCES**

Feature	PIC18F242	PIC18F252	PIC18F442	PIC18F452
Program Memory (Kbytes)	16	32	16	32
Data Memory (Bytes)	768	1536	768	1536
A/D Channels	5	5	8	8
Parallel Slave Port (PSP)	No	No	Yes	Yes
Package Types	28-pin DIP 28-pin SOIC	28-pin DIP 28-pin SOIC	40-pin DIP 44-pin PLCC 44-pin TQFP	40-pin DIP 44-pin PLCC 44-pin TQFP

## APPENDIX C: CONVERSION CONSIDERATIONS

This appendix discusses the considerations for converting from previous versions of a device to the ones listed in this data sheet. Typically, these changes are due to the differences in the process technology used. An example of this type of conversion is from a PIC16C74A to a PIC16C74B.

**Not Applicable**

## APPENDIX D: MIGRATION FROM BASELINE TO ENHANCED DEVICES

This section discusses how to migrate from a Baseline device (i.e., PIC16C5X) to an Enhanced MCU device (i.e., PIC18FXXX).

The following are the list of modifications over the PIC16C5X microcontroller family:

**Not Currently Available**

## APPENDIX E: MIGRATION FROM MID-RANGE TO ENHANCED DEVICES

A detailed discussion of the differences between the mid-range MCU devices (i.e., PIC16CXXX) and the enhanced devices (i.e., PIC18FXXX) is provided in AN716, "Migrating Designs from PIC16C74A/74B to PIC18F442". 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 PIC18FXXX Migration". This Application Note is available as Literature Number DS00726.

# **PIC18FXX2**

---

---

## **NOTES:**

## INDEX

### A

A/D .....	181
A/D Converter Flag (ADIF Bit) .....	183
A/D Converter Interrupt, Configuring .....	184
Acquisition Requirements .....	184
ADCON0 Register .....	181
ADCON1 Register .....	181
ADRESH Register .....	181
ADRESH/ADRESL Registers .....	183
ADRESL Register .....	181
Analog Port Pins .....	99, 100
Analog Port Pins, Configuring .....	186
Associated Registers .....	188
Configuring the Module .....	184
Conversion Clock (TAD) .....	186
Conversion Status (GO/DONE Bit) .....	183
Conversions .....	187
Converter Characteristics .....	287
Equations	
Acquisition Time .....	185
Minimum Charging Time .....	185
Examples	
Calculating the Minimum Required	
Acquisition Time .....	185
Result Registers .....	187
Special Event Trigger (CCP) .....	120, 188
TAD vs. Device Operating Frequencies .....	186
Use of the CCP2 Trigger .....	188
Absolute Maximum Ratings .....	259
AC (Timing) Characteristics .....	269
Load Conditions for Device Timing	
Specifications .....	270
Parameter Symbology .....	269
Temperature and Voltage Specifications - AC .....	270
Timing Conditions .....	270
ACKSTAT Status Flag .....	155
ADCON0 Register .....	181
GO/DONE Bit .....	183
ADCON1 Register .....	181
ADDLW .....	217
ADDWF .....	217
ADDWFC .....	218
ADRESH Register .....	181
ADRESH/ADRESL Registers .....	183
ADRESL Register .....	181
Analog-to-Digital Converter. See A/D .....	
ANDLW .....	218
ANDWF .....	219
Assembler	
MPASM Assembler .....	253

### B

Baud Rate Generator .....	151
BC .....	219
BCF .....	220
BF Status Flag .....	155

### Block Diagrams

A/D Converter .....	183
Analog Input Model .....	184
Baud Rate Generator .....	151
Capture Mode Operation .....	119
Compare Mode Operation .....	120
Low Voltage Detect	
External Reference Source .....	190
Internal Reference Source .....	190
MSSP	
I <sup>2</sup> C Mode .....	134
MSSP (SPI Mode) .....	125
On-Chip Reset Circuit .....	25
Parallel Slave Port (PORTD and PORTE) .....	100
PIC18F2X2 .....	8
PIC18F4X2 .....	9
PLL .....	19
PORTC (Peripheral Output Override) .....	93
PORTD (I/O Mode) .....	95
PORTE (I/O Mode) .....	97
PWM Operation (Simplified) .....	122
RA3:RA0 and RA5 Port Pins .....	87
RA4/T0CKI Pin .....	88
RA6 Pin .....	88
RB2:RB0 Port Pins .....	91
RB3 Pin .....	91
RB7:RB4 Port Pins .....	90
Table Read Operation .....	55
Table Write Operation .....	56
Table Writes to FLASH Program Memory .....	61
Timer0 in 16-bit Mode .....	104
Timer0 in 8-bit Mode .....	104
Timer1 .....	108
Timer1 (16-bit R/W Mode) .....	108
Timer2 .....	112
Timer3 .....	114
Timer3 (16-bit R/W Mode) .....	114
USART	
Asynchronous Receive .....	174
Asynchronous Transmit .....	172
Watchdog Timer .....	204
BN .....	220
BNC .....	221
BNN .....	221
BNOV .....	222
BNZ .....	222
BOR. See Brown-out Reset .....	
BOV .....	225
BRA .....	223
BRG. See Baud Rate Generator .....	
Brown-out Reset (BOR) .....	26
BSF .....	223
BTFSCL .....	224
BTFSR .....	224
BTG .....	225
Bus Collision During a STOP Condition .....	163
BZ .....	226

## C

CALL .....	226
Capture (CCP Module) .....	119
Associated Registers .....	121
CCP Pin Configuration .....	119
CCPR1H:CCPR1L Registers .....	119
Software Interrupt .....	119
Timer1/Timer3 Mode Selection .....	119
Capture/Compare/PWM (CCP) .....	117
Capture Mode. See Capture	
CCP1 .....	118
CCPR1H Register .....	118
CCPR1L Register .....	118
CCP2 .....	118
CCPR2H Register .....	118
CCPR2L Register .....	118
Compare Mode. See Compare	
Interaction of Two CCP Modules .....	118
PWM Mode. See PWM	
Timer Resources .....	118
Clocking Scheme/Instruction Cycle .....	39
CLRF .....	227
CLRWDT .....	227
Code Examples	
16 x 16 Signed Multiply Routine .....	72
16 x 16 Unsigned Multiply Routine .....	72
8 x 8 Signed Multiply Routine .....	71
8 x 8 Unsigned Multiply Routine .....	71
Changing Between Capture Prescalers .....	119
Data EEPROM Read .....	67
Data EEPROM Refresh Routine .....	68
Data EEPROM Write .....	67
Erasing a FLASH Program Memory Row .....	60
Fast Register Stack .....	39
How to Clear RAM (Bank1) Using	
Indirect Addressing .....	50
Initializing PORTA .....	87
Initializing PORTB .....	90
Initializing PORTC .....	93
Initializing PORTD .....	95
Initializing PORTE .....	97
Loading the SSPBUF (SSPSR) Register .....	128
Reading a FLASH Program Memory Word .....	59
Saving STATUS, WREG and BSR	
Registers in RAM .....	85
Writing to FLASH Program Memory .....	62-63
Code Protection .....	195
COMF .....	228
Compare (CCP Module) .....	120
Associated Registers .....	121
CCP Pin Configuration .....	120
CCPR1 Register .....	120
Software Interrupt .....	120
Special Event Trigger .....	109, 115, 120, 188
Timer1/Timer3 Mode Selection .....	120
Configuration Bits .....	195
Context Saving During Interrupts .....	85
Conversion Considerations .....	314
CPFSEQ .....	228
CPFSGT .....	229
CPFSLT .....	229

## D

Data EEPROM Memory	
Associated Registers .....	69
EEADR Register .....	65
EECON1 Register .....	65
EECON2 Register .....	65
Operation During Code Protect .....	68
Protection Against Spurious Write .....	68
Reading .....	67
Using .....	68
Write Verify .....	68
Writing .....	67
Data Memory .....	42
General Purpose Registers .....	42
Map for PIC18F242/442 .....	43
Map for PIC18F252/452 .....	44
Special Function Registers .....	42
DAW .....	230
DC and AC Characteristics	
Graphs and Tables .....	289
DC Characteristics .....	261, 265
DCFSNZ .....	231
DECFL .....	230
DECFSZ .....	231
Development Support .....	253
Device Differences .....	313
Device Overview .....	7
Features .....	7
Direct Addressing .....	51
Example .....	49
E	
Electrical Characteristics .....	259
Errata .....	5
F	
Firmware Instructions .....	211
FLASH Program Memory .....	55
Associated Registers .....	63
Control Registers .....	56
Erase Sequence .....	60
Erasing .....	60
Operation During Code Protect .....	63
Reading .....	59
TABLAT Register .....	58
Table Pointer .....	58
Boundaries Based on Operation .....	58
Table Pointer Boundaries .....	58
Table Reads and Table Writes .....	55
Block Diagrams	
Reads from FLASH Program Memory .....	59
Writing to .....	61
Protection Against Spurious Writes .....	63
Unexpected Termination .....	63
Write Verify .....	63
G	
General Call Address Support .....	148
GOTO .....	232

## I

I/O Ports .....	87
I <sup>2</sup> C (MSSP Module)	
ACK Pulse .....	139
Read/Write Bit Information (R/W Bit) .....	139
I <sup>2</sup> C (SSP Module)	
ACK Pulse .....	138
I <sup>2</sup> C Master Mode Reception .....	155
I <sup>2</sup> C Mode	
Clock Stretching .....	144
I <sup>2</sup> C Mode (MSSP Module) .....	134
Registers .....	134
I <sup>2</sup> C Module	
ACK Pulse .....	138, 139
Acknowledge Sequence Timing .....	158
Baud Rate Generator .....	151
Bus Collision	
Repeated START Condition .....	162
START Condition .....	160
Clock Arbitration .....	152
Effect of a RESET .....	159
General Call Address Support .....	148
Master Mode .....	149
Operation .....	150
Repeated START Condition Timing .....	154
Master Mode START Condition .....	153
Master Mode Transmission .....	155
Multi-Master Communication, Bus Collision	
and Arbitration .....	159
Multi-Master Mode .....	159
Operation .....	138
Read/Write Bit Information (R/W Bit) .....	138, 139
Serial Clock (RC3/SCK/SCL) .....	139
Slave Mode .....	138
Addressing .....	138
Reception .....	139
Transmission .....	139
Slave Mode Timing (10-bit Reception, SEN = 0) .....	142
Slave Mode Timing (10-bit Reception, SEN = 1) .....	147
Slave Mode Timing (10-bit Transmission) .....	143
Slave Mode Timing (7-bit Reception, SEN = 0) .....	140
Slave Mode Timing (7-bit Reception, SEN = 1) .....	146
Slave Mode Timing (7-bit Transmission) .....	141
SLEEP Operation .....	159
STOP Condition Timing .....	158
ICEPIC In-Circuit Emulator .....	254
ID Locations .....	195, 210
INCFSZ .....	232
INCSFSZ .....	233
In-Circuit Debugger .....	210
In-Circuit Serial Programming (ICSP) .....	195, 210
Indirect Addressing .....	51
INDF and FSR Registers .....	50
Indirect Addressing Operation .....	51
Indirect File Operand .....	42
INFSNZ .....	233
Instruction Cycle .....	39
Instruction Flow/Pipelining .....	40
Instruction Format .....	213

Instruction Set .....	211
ADDLW .....	217
ADDWF .....	217
ADDWFC .....	218
ANDLW .....	218
ANDWF .....	219
BC .....	219
BCF .....	220
BN .....	220
BNC .....	221
BNN .....	221
BNOV .....	222
BNZ .....	222
BOV .....	225
BRA .....	223
BSF .....	223
BTFSR .....	224
BTFSZ .....	224
BTG .....	225
BZ .....	226
CALL .....	226
CLRF .....	227
CLRWDT .....	227
COMF .....	228
CPFSEQ .....	228
CPFGT .....	229
CPFSLT .....	229
DAW .....	230
DCFSNZ .....	231
DECFSZ .....	230
DECFSZ .....	231
GOTO .....	232
INCF .....	232
INCFSZ .....	233
INFSNZ .....	233
IORLW .....	234
IORWF .....	234
LFSR .....	235
MOV .....	235
MOVFF .....	236
MOVLB .....	236
MOVLW .....	237
MOVWF .....	237
MULLW .....	238
MULWF .....	238
NEGF .....	239
NOP .....	239
POP .....	240
PUSH .....	240
RCALL .....	241
RESET .....	241
RETFIE .....	242
RETLW .....	242
RETURN .....	243
RLCF .....	243
RLNCF .....	244
RRCF .....	244
RRNCF .....	245
SETF .....	245
SLEEP .....	246
SUBFWB .....	246
SUBLB .....	247
SUBWF .....	247
SUBWFB .....	248
SWAPF .....	248

TBLRD .....	249
TBLWT .....	250
TSTFSZ .....	251
XORLW .....	251
XORWF .....	252
Summary Table .....	214
Instructions in Program Memory .....	40
Two-Word Instructions .....	41
INT Interrupt (RB0/INT). See Interrupt Sources	
INTCON Register	
RBIF Bit .....	90
INTCON Registers .....	75-77
Inter-Integrated Circuit. See I <sup>2</sup> C	
Interrupt Sources .....	195
A/D Conversion Complete .....	184
Capture Complete (CCP) .....	119
Compare Complete (CCP) .....	120
INT0 .....	85
Interrupt-on-Change (RB7:RB4) .....	90
PORTB, Interrupt-on-Change .....	85
RB0/INT Pin, External .....	85
TMR0 .....	85
TMR0 Overflow .....	105
TMR1 Overflow .....	107, 109
TMR2 to PR2 Match .....	112
TMR2 to PR2 Match (PWM) .....	111, 122
TMR3 Overflow .....	113, 115
USART Receive/Transmit Complete .....	165
Interrupts .....	73
Logic .....	74
Interrupts, Enable Bits	
CCP1 Enable (CCP1IE Bit) .....	119
Interrupts, Flag Bits	
A/D Converter Flag (ADIF Bit) .....	183
CCP1 Flag (CCP1IF Bit) .....	119
CCP1IF Flag (CCP1IF Bit) .....	120
Interrupt-on-Change (RB7:RB4) Flag (RBIF Bit) .....	90
IORLW .....	234
IORWF .....	234
IPR Registers .....	82-83
<b>K</b>	
KEELOQ Evaluation and Programming Tools .....	256
<b>L</b>	
LFSR .....	235
Lookup Tables	
Computed GOTO .....	41
Table Reads, Table Writes .....	41
Low Voltage Detect .....	189
Converter Characteristics .....	267
Effects of a RESET .....	193
Operation .....	192
Current Consumption .....	193
During SLEEP .....	193
Reference Voltage Set Point .....	193
Typical Application .....	189
LVD. See Low Voltage Detect. ....	189
<b>M</b>	
Master SSP (MSSP) Module Overview .....	125
Master Synchronous Serial Port (MSSP). See MSSP.	
Master Synchronous Serial Port. See MSSP	
Memory Organization	
Data Memory .....	42
Program Memory .....	35
Memory Programming Requirements .....	268
Migration from Baseline to Enhanced Devices .....	314
Migration from High-End to Enhanced Devices .....	315
Migration from Mid-Range to Enhanced Devices .....	315
MOVF .....	235
MOVFF .....	236
MOVLB .....	236
MOVlw .....	237
MOVWF .....	237
MPLAB C17 and MPLAB C18 C Compilers .....	253
MPLAB ICD In-Circuit Debugger .....	255
MPLAB ICE High Performance Universal In-Circuit Emulator with MPLAB IDE .....	254
MPLAB Integrated Development	
Environment Software .....	253
MPLINK Object Linker/MPLIB Object Librarian .....	254
MSSP .....	125
Control Registers (general) .....	125
Enabling SPI I/O .....	129
Operation .....	128
Typical Connection .....	129
MSSP Module	
SPI Master Mode .....	130
SPI Master./Slave Connection .....	129
SPI Slave Mode .....	131
MULLW .....	238
MULWF .....	238
<b>N</b>	
NEGF .....	239
NOP .....	239
<b>O</b>	
Opcode Field Descriptions .....	212
OPTION_REG Register	
PSA Bit .....	105
T0CS Bit .....	105
T0PS2:T0PS0 Bits .....	105
T0SE Bit .....	105
Oscillator Configuration .....	17
EC .....	17
ECIO .....	17
HS .....	17
HS + PLL .....	17
LP .....	17
RC .....	17
RCIO .....	17
XT .....	17
Oscillator Selection .....	195
Oscillator, Timer1 .....	107, 109, 115
Oscillator, Timer3 .....	113
Oscillator, WDT .....	203

## P

Packaging .....	305	RC7/RX/DT .....	15
Details .....	307	RD0/PSP0 .....	16
Marking Information .....	305	RD1/PSP1 .....	16
Parallel Slave Port		RD2/PSP2 .....	16
PORTD .....	100	RD3/PSP3 .....	16
Parallel Slave Port (PSP) .....	95, 100	RD4/PSP4 .....	16
Associated Registers .....	101	RD5/PSP5 .....	16
RE0/RD/AN5 Pin .....	99, 100	RD6/PSP6 .....	16
RE1/WR/AN6 Pin .....	99, 100	RD7/PSP7 .....	16
RE2/CS/AN7 Pin .....	99, 100	RE0/RD/AN5 .....	16
Select (PSPMODE Bit) .....	95, 100	RE1/WR/AN6 .....	16
PIC18F2X2 Pin Functions		RE2/CS/AN7 .....	16
MCLR/VPP .....	10	VDD .....	16
OSC1/CLKI .....	10	Vss .....	16
OSC2/CLKO/RA6 .....	10	<b>PIC18FXX2 Voltage-Frequency Graph</b>	
RA0/AN0 .....	10	(Industrial) .....	260
RA1/AN1 .....	10	<b>PIC18LFXX2 Voltage-Frequency Graph</b>	
RA2/AN2/VREF- .....	10	(Industrial) .....	260
RA3/AN3/VREF+ .....	10	<b>PICDEM 1 Low Cost PICmicro</b>	
RA4/TOCKI .....	10	Demonstration Board .....	255
RA5/AN4/SS/LVDIN .....	10	<b>PICDEM 17 Demonstration Board</b> .....	256
RB0/INT0 .....	11	<b>PICDEM 2 Low Cost PIC16CXX</b>	
RB1/INT1 .....	11	Demonstration Board .....	255
RB2/INT2 .....	11	<b>PICDEM 3 Low Cost PIC16CXXX</b>	
RB3/CCP2 .....	11	Demonstration Board .....	256
RB4 .....	11	<b>PICSTART Plus Entry Level Development</b>	
RB5/PGM .....	11	Programmer .....	255
RB6/PGC .....	11	<b>PIE Registers</b> .....	80–81
RB7/PGD .....	11	<b>Pinout I/O Descriptions</b>	
RC0/T1OSO/T1CKI .....	12	PIC18F2X2 .....	10
RC1/T1OSI/CCP2 .....	12	<b>PIR Registers</b> .....	78–79
RC2/CCP1 .....	12	<b>PLL Lock Time-out</b> .....	26
RC3/SCK/SCL .....	12	<b>Pointer, FSR</b> .....	50
RC4/SDI/SDA .....	12	<b>POP</b> .....	240
RC5/SDO .....	12	<b>POR. See Power-on Reset</b>	
RC6/TX/CK .....	12	<b>PORTA</b>	
RC7/RX/DT .....	12	Associated Registers .....	89
VDD .....	12	LATA Register .....	87
Vss .....	12	PORTA Register .....	87
<b>PIC18F4X2 Pin Functions</b>		TRISA Register .....	87
MCLR/VPP .....	13	<b>PORTB</b>	
OSC1/CLKI .....	13	Associated Registers .....	92
OSC2/CLKO .....	13	LATB Register .....	90
RA0/AN0 .....	13	PORTB Register .....	90
RA1/AN1 .....	13	RB0/INT Pin, External .....	85
RA2/AN2/VREF- .....	13	RB7:RB4 Interrupt-on-Change Flag (RBIF Bit) .....	90
RA3/AN3/VREF+ .....	13	TRISB Register .....	90
RA4/TOCKI .....	13	<b>PORTC</b>	
RA5/AN4/SS/LVDIN .....	13	Associated Registers .....	94
RB0/INT .....	14	LATC Register .....	93
RB1 .....	14	PORTC Register .....	93
RB2 .....	14	RC3/SCK/SCL Pin .....	139
RB3 .....	14	RC7/RX/DT Pin .....	168
RB4 .....	14	TRISC Register .....	93, 165
RB5/PGM .....	14	<b>PORTD</b>	
RB6/PGC .....	14	Associated Registers .....	96
RB7/PGD .....	14	LATD Register .....	95
RC0/T1OSO/T1CKI .....	15	Parallel Slave Port (PSP) Function .....	95
RC1/T1OSI/CCP2 .....	15	PORTD Register .....	95
RC2/CCP1 .....	15	TRISD Register .....	95
RC3/SCK/SCL .....	15		
RC4/SDI/SDA .....	15		
RC5/SDO .....	15		
RC6/TX/CK .....	15		

<b>PORTE</b>	
Analog Port Pins .....	99, 100
Associated Registers .....	99
LATE Register .....	97
PORTE Register .....	97
PSP Mode Select (PSPMODE Bit) .....	95, 100
RE0/ <u>RD</u> /AN5 Pin .....	99, 100
RE1/ <u>WR</u> /AN6 Pin .....	99, 100
RE2/ <u>CS</u> /AN7 Pin .....	99, 100
TRISE Register .....	97
<b>Postscaler, WDT</b>	
Assignment (PSA Bit) .....	105
Rate Select (T0PS2:T0PS0 Bits) .....	105
Switching Between Timer0 and WDT .....	105
<b>Power-down Mode. See SLEEP</b>	
<b>Power-on Reset (POR)</b>	26
Oscillator Start-up Timer (OST) .....	26
Power-up Timer (PWRT) .....	26
<b>Prescaler, Capture</b>	119
<b>Prescaler, Timer0</b>	105
Assignment (PSA Bit) .....	105
Rate Select (T0PS2:T0PS0 Bits) .....	105
Switching Between Timer0 and WDT .....	105
<b>Prescaler, Timer2</b>	122
<b>PRO MATE II Universal Device Programmer</b>	255
<b>Product Identification System</b>	327
<b>Program Counter</b>	
PCL Register .....	39
PCLATH Register .....	39
PCLATU Register .....	39
<b>Program Memory</b>	
Interrupt Vector .....	35
Map and Stack for PIC18F442/242 .....	36
Map and Stack for PIC18F452/252 .....	36
RESET Vector .....	35
<b>Program Verification and Code Protection</b>	207
Associated Registers .....	207
<b>Programming, Device Instructions</b>	211
<b>PSP. See Parallel Slave Port.</b>	
<b>Pulse Width Modulation. See PWM (CCP Module).</b>	
<b>PUSH</b>	240
<b>PWM (CCP Module)</b>	122
Associated Registers .....	123
CCPR1H:CCPR1L Registers .....	122
Duty Cycle .....	122
Example Frequencies/Resolutions .....	123
Period .....	122
Setup for PWM Operation .....	123
TMR2 to PR2 Match .....	111, 122
<b>Q</b>	
<b>Q Clock</b>	122
<b>R</b>	
<b>RAM. See Data Memory</b>	
<b>RC Oscillator</b>	18
<b>RCALL</b>	241
<b>RCSTA Register</b>	
SPEN Bit .....	165
<b>Register File</b>	42
<b>Registers</b>	
ADCON0 (A/D Control 0) .....	181
ADCON1 (A/D Control 1) .....	182
CCP1CON and CCP2CON	
(Capture/Compare/PWM Control) .....	117
CONFIG1H (Configuration 1 High) .....	196
CONFIG2H (Configuration 2 High) .....	197
CONFIG2L (Configuration 2 Low) .....	197
CONFIG3H (Configuration 3 High) .....	198
CONFIG4L (Configuration 4 Low) .....	198
CONFIG5H (Configuration 5 High) .....	199
CONFIG5L (Configuration 5 Low) .....	199
CONFIG6H (Configuration 6 High) .....	200
CONFIG6L (Configuration 6 Low) .....	200
CONFIG7H (Configuration 7 High) .....	201
CONFIG7L (Configuration 7 Low) .....	201
DEVID1 (Device ID Register 1) .....	202
DEVID2 (Device ID Register 2) .....	202
EECON1 (Data EEPROM Control 1) .....	57, 66
File Summary .....	46–48
INTCON (Interrupt Control) .....	75
INTCON2 (Interrupt Control 2) .....	76
INTCON3 (Interrupt Control 3) .....	77
IPR1 (Peripheral Interrupt Priority 1) .....	82
IPR2 (Peripheral Interrupt Priority 2) .....	83
LVDCON (LVD Control) .....	191
OSCCON (Oscillator Control) .....	21
PIE1 (Peripheral Interrupt Enable 1) .....	80
PIE2 (Peripheral Interrupt Enable 2) .....	81
PIR1 (Peripheral Interrupt Request 1) .....	78
PIR2 (Peripheral Interrupt Request 2) .....	79
RCON (Register Control) .....	84
RCON (RESET Control) .....	53
RCSTA (Receive Status and Control) .....	167
SSPCON1 (MSSP Control 1)	
I <sup>2</sup> C Mode .....	136
SPI Mode .....	127
SSPCON2 (MSSP Control 2)	
I <sup>2</sup> C Mode .....	137
SSPSTAT (MSSP Status)	
I <sup>2</sup> C Mode .....	135
SPI Mode .....	126
STATUS .....	52
STKPTR (Stack Pointer) .....	38
T0CON (Timer0 Control) .....	103
T1CON (Timer 1 Control) .....	107
T2CON (Timer 2 Control) .....	111
T3CON (Timer3 Control) .....	113
TRISE .....	98
TXSTA (Transmit Status and Control) .....	166
WDTCON (Watchdog Timer Control) .....	203
<b>RESET</b>	25, 195, 241
Brown-out Reset (BOR) .....	195
<u>MCLR</u> Reset (During SLEEP) .....	25
<u>MCLR</u> Reset (Normal Operation) .....	25
Oscillator Start-up Timer (OST) .....	195
Power-on Reset (POR) .....	25, 195
Power-up Timer (PWRT) .....	195
Programmable Brown-out Reset (BOR) .....	25
RESET Instruction .....	25
Stack Full Reset .....	25
Stack Underflow Reset .....	25
Watchdog Timer (WDT) Reset .....	25

RETFIE .....	242
RETLW .....	242
RETURN .....	243
Revision History .....	313
RLCF .....	243
RLNCF .....	244
RRCF .....	244
RRNCF .....	245
<b>S</b>	
SCI. See USART	
SCK .....	125
SDI .....	125
SDO .....	125
Serial Clock, SCK .....	125
Serial Communication Interface. See USART	
Serial Data In, SDI .....	125
Serial Data Out, SDO .....	125
Serial Peripheral Interface. See SPI	
SETF .....	245
Slave Select Synchronization .....	131
Slave Select, SS .....	125
SLEEP .....	195, 205, 246
Software Simulator (MPLAB SIM) .....	254
Special Event Trigger. See Compare	
Special Features of the CPU .....	195
Configuration Registers .....	196–201
Special Function Registers .....	42
Map .....	45
SPI	
Master Mode .....	130
Serial Clock .....	125
Serial Data In .....	125
Serial Data Out .....	125
Slave Select .....	125
SPI Clock .....	130
SPI Mode .....	125
SPI Master/Slave Connection .....	129
SPI Module	
Associated Registers .....	133
Bus Mode Compatibility .....	133
Effects of a RESET .....	133
Master/Slave Connection .....	129
Slave Mode .....	131
Slave Select Synchronization .....	131
Slave Synch Timing .....	131
SLEEP Operation .....	133
SS .....	125
SSP	
I <sup>2</sup> C Mode. See I <sup>2</sup> C	
SPI Mode .....	125
SPI Mode. See SPI	
SSPBUF Register .....	130
SSPSR Register .....	130
TMR2 Output for Clock Shift .....	111, 112
SSPOV Status Flag .....	155
SSPSTAT Register	
R/W Bit .....	138, 139
Status Bits	
Significance and the Initialization Condition for RCON Register .....	27
SUBFWB .....	246
SUBLW .....	247
SUBWF .....	247
SUBWFB .....	248
SWAPF .....	248

## T

TABLAT Register .....	58
Table Pointer Operations (table) .....	58
TBLPTR Register .....	58
TBLRD .....	249
TBLWT .....	250
Time-out Sequence .....	26
Time-out in Various Situations .....	27
Timer0 .....	103
16-bit Mode Timer Reads and Writes .....	105
Associated Registers .....	105
Clock Source Edge Select (T0SE Bit) .....	105
Clock Source Select (T0CS Bit) .....	105
Operation .....	105
Overflow Interrupt .....	105
Prescaler. See Prescaler, Timer0	
Timer1 .....	107
16-bit Read/Write Mode .....	109
Associated Registers .....	110
Operation .....	108
Oscillator .....	107, 109
Overflow Interrupt .....	107, 109
Special Event Trigger (CCP) .....	109, 120
TMR1H Register .....	107
TMR1L Register .....	107
Timer2 .....	111
Associated Registers .....	112
Operation .....	111
Postscaler. See Postscaler, Timer2	
PR2 Register .....	111, 122
Prescaler. See Prescaler, Timer2	
SSP Clock Shift .....	111, 112
TMR2 Register .....	111
TMR2 to PR2 Match Interrupt .....	111, 112, 122
Timer3 .....	113
Associated Registers .....	115
Operation .....	114
Oscillator .....	113, 115
Overflow Interrupt .....	113, 115
Special Event Trigger (CCP) .....	115
TMR3H Register .....	113
TMR3L Register .....	113
Timing Diagrams	
Bus Collision	
Transmit and Acknowledge .....	159
A/D Conversion .....	287
Acknowledge Sequence .....	158
Baud Rate Generator with Clock Arbitration .....	152
BRG Reset Due to SDA Arbitration During START Condition .....	161
Brown-out Reset (BOR) .....	274
Bus Collision	
Start Condition (SDA Only) .....	160
Bus Collision During a Repeated START Condition (Case 1) .....	162
Bus Collision During a Repeated START Condition (Case 2) .....	162
Bus Collision During a START Condition (SCL = 0) .....	161
Bus Collision During a STOP Condition (Case 1) .....	163
Bus Collision During a STOP Condition (Case 2) .....	163
Capture/Compare/PWM (CCP1 and CCP2) .....	276
CLKO and I/O .....	272
Clock Synchronization .....	145

Example SPI Master Mode (CKE = 0) .....	278	USART Synchronous Transmission (Through TXEN) .....	177
Example SPI Master Mode (CKE = 1) .....	279	Wake-up from SLEEP via Interrupt .....	206
Example SPI Slave Mode (CKE = 0) .....	280	Timing Diagrams Requirements	
Example SPI Slave Mode (CKE = 1) .....	281	Master SSP I <sup>2</sup> C Bus START/STOP Bits .....	284
External Clock (All Modes except PLL) .....	271	Timing Requirements	
First START Bit Timing .....	153	A/D Conversion .....	288
I <sup>2</sup> C Bus Data .....	282	Capture/Compare/PWM (CCP1 and CCP2) .....	276
I <sup>2</sup> C Bus START/STOP Bits .....	282	CLKO and I/O .....	273
I <sup>2</sup> C Master Mode (Reception, 7-bit Address) .....	157	Example SPI Mode (Master Mode, CKE = 0) .....	278
I <sup>2</sup> C Master Mode (Transmission, 7 or 10-bit Address) .....	156	Example SPI Mode (Master Mode, CKE = 1) .....	279
I <sup>2</sup> C Slave Mode Timing (10-bit Reception, SEN = 0) .....	142	Example SPI Mode (Slave Mode, CKE = 0) .....	280
I <sup>2</sup> C Slave Mode Timing (10-bit Transmission) .....	143	Example SPI Slave Mode (CKE = 1) .....	281
I <sup>2</sup> C Slave Mode Timing (7-bit Reception, SEN = 0) .....	140	External Clock .....	271
I <sup>2</sup> C Slave Mode Timing (7-bit Reception, SEN = 1) .....	146, 147	I <sup>2</sup> C Bus Data (Slave Mode) .....	283
I <sup>2</sup> C Slave Mode Timing (7-bit Transmission) .....	141	Master SSP I <sup>2</sup> C Bus Data .....	285
Low Voltage Detect .....	192	Parallel Slave Port (PIC18F4X2) .....	277
Master SSP I <sup>2</sup> C Bus Data .....	284	RESET, Watchdog Timer, Oscillator Start-up Timer, Power-up Timer and Brown-out Reset Requirements .....	274
Master SSP I <sup>2</sup> C Bus START/STOP Bits .....	284	Timer0 and Timer1 External Clock .....	275
Parallel Slave Port (PIC18F4X2) .....	277	USART Synchronous Receive .....	286
Parallel Slave Port (Read) .....	101	USART Synchronous Transmission .....	286
Parallel Slave Port (Write) .....	100	Timing Specifications	
PWM Output .....	122	PLL Clock .....	272
Repeat START Condition .....	154	TRISE Register	
RESET, Watchdog Timer (WDT), Oscillator Start-up Timer (OST) and Power-up Timer (PWRT) .....	273	PSPMODE Bit .....	95, 100
Slave Synchronization .....	131	TSTFSZ .....	251
Slaver Mode General Call Address Sequence (7 or 10-bit Address Mode) .....	148	Two-Word Instructions	
Slow Rise Time (MCLR Tied to VDD) .....	33	Example Cases .....	41
SPI Mode (Master Mode) .....	130	TXSTA Register	
SPI Mode (Slave Mode with CKE = 0) .....	132	BRGH Bit .....	168
SPI Mode (Slave Mode with CKE = 1) .....	132	<b>U</b>	
Stop Condition Receive or Transmit Mode .....	158	Universal Synchronous Asynchronous Receiver Transmitter. See USART	
Time-out Sequence on POR w/PLL Enabled (MCLR Tied to VDD) .....	33		
Time-out Sequence on Power-up (MCLR Not Tied to VDD)			
Case 1 .....	32	USART .....	165
Case 2 .....	32	Asynchronous Mode .....	172
Time-out Sequence on Power-up (MCLR Tied to VDD) .....	32	Associated Registers, Receive .....	175
Timer0 and Timer1 External Clock .....	275	Associated Registers, Transmit .....	173
Timing for Transition Between Timer1 and OSC1 (HS with PLL) .....	23	Receiver .....	174
Transition Between Timer1 and OSC1 (HS, XT, LP) .....	22	Transmitter .....	172
Transition Between Timer1 and OSC1 (RC, EC) .....	23	Baud Rate Generator (BRG) .....	168
Transition from OSC1 to Timer1 Oscillator .....	22	Associated Registers .....	168
USART Asynchronous Master Transmission .....	173	Baud Rate Error, Calculating .....	168
USART Asynchronous Master Transmission (Back to Back) .....	173	Baud Rate Formula .....	168
USART Asynchronous Reception .....	175	Baud Rates for Asynchronous Mode (BRGH = 0) .....	170
USART Synchronous Receive (Master/Slave) .....	286	Baud Rates for Asynchronous Mode (BRGH = 1) .....	171
USART Synchronous Reception (Master Mode, SREN) .....	178	Baud Rates for Synchronous Mode .....	169
USART Synchronous Transmission .....	177	High Baud Rate Select (BRGH Bit) .....	168
USART Synchronous Transmission (Master/Slave) .....	286	Sampling .....	168

## W

Wake-up from SLEEP .....	195, 205
Using Interrupts .....	205
Watchdog Timer (WDT) .....	195, 203
Associated Registers .....	204
Control Register .....	203
Postscaler .....	203, 204
Programming Considerations .....	203
RC Oscillator .....	203
Time-out Period .....	203
WCOL .....	153
WCOL Status Flag .....	153, 155, 158
WWW, On-Line Support .....	5

## X

XORLW .....	251
XORWF .....	252

# **PIC18FXX2**

---

---

## **NOTES:**

## ON-LINE SUPPORT

Microchip provides on-line support on the Microchip World Wide Web site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape® or Microsoft® Internet Explorer. Files are also available for FTP download from our FTP site.

### Connecting to the Microchip Internet Web Site

The Microchip web site is available at the following URL:

**[www.microchip.com](http://www.microchip.com)**

The file transfer site is available by using an FTP service to connect to:

**<ftp://ftp.microchip.com>**

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events

## SYSTEMS INFORMATION AND UPGRADE HOT LINE

The Systems Information and Upgrade Line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive the most current upgrade kits. The Hot Line Numbers are:

1-800-755-2345 for U.S. and most of Canada, and  
1-480-792-7302 for the rest of the world.

092002

# PIC18FXX2

---

---

## 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: PIC18FXX2

Literature Number: DS39564B

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?

---

## PIC18FXX2 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>	-	X	XX	XXX	
Device		Temperature Range	Package	Pattern	Examples:
Device		PIC18FXX2 <sup>(1)</sup> , PIC18FXX2T <sup>(2)</sup> ; VDD range 4.2V to 5.5V PIC18LFXX2 <sup>(1)</sup> , PIC18LFXX2T <sup>(2)</sup> ; VDD range 2.5V to 5.5V			a) PIC18LF452 - I/P 301 = Industrial temp., PDIP package, Extended VDD limits, QTP pattern #301. b) PIC18LF242 - I/SO = Industrial temp., SOIC package, Extended VDD limits. c) PIC18F442 - E/P = Extended temp., PDIP package, normal VDD limits.
Temperature Range	I	= -40°C to +85°C (Industrial)			
	E	= -40°C to +125°C (Extended)			
Package	PT	= TQFP (Thin Quad Flatpack)			<b>Note 1:</b> F = Standard Voltage range LF = Wide Voltage Range
	SO	= SOIC			<b>2:</b> T = in tape and reel - SOIC, PLCC, and TQFP packages only.
	SP	= Skinny Plastic DIP			
	P	= PDIP			
	L	= PLCC			
Pattern		QTP, SQTP, Code or Special Requirements (blank otherwise)			

### Sales and Support

#### **Data Sheets**

Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office
2. The Microchip Corporate Literature Center U.S. FAX: (480) 792-7277
3. The Microchip Worldwide Site ([www.microchip.com](http://www.microchip.com))

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

#### **New Customer Notification System**

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



# 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: 480-792-7627  
Web Address: <http://www.microchip.com>

#### Rocky Mountain

2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7966 Fax: 480-792-4338

#### Atlanta

500 Sugar Mill Road, Suite 200B  
Atlanta, GA 30350  
Tel: 770-640-0034 Fax: 770-640-0307

#### Boston

2 Lan Drive, Suite 120  
Westford, MA 01886  
Tel: 978-692-3848 Fax: 978-692-3821

#### Chicago

333 Pierce Road, Suite 180  
Itasca, IL 60143  
Tel: 630-285-0071 Fax: 630-285-0075

#### Dallas

4570 Westgrove Drive, Suite 160  
Addison, TX 75001  
Tel: 972-818-7423 Fax: 972-818-2924

#### Detroit

Tri-Atria Office Building  
32255 Northwestern Highway, Suite 190  
Farmington Hills, MI 48334  
Tel: 248-538-2250 Fax: 248-538-2260

#### Kokomo

2767 S. Albright Road  
Kokomo, Indiana 46902  
Tel: 765-864-8360 Fax: 765-864-8387

#### Los Angeles

18201 Von Karman, Suite 1090  
Irvine, CA 92612  
Tel: 949-263-1888 Fax: 949-263-1338

#### New York

150 Motor Parkway, Suite 202  
Hauppauge, NY 11788  
Tel: 631-273-5305 Fax: 631-273-5335

#### San Jose

Microchip Technology Inc.  
2107 North First Street, Suite 590  
San Jose, CA 95131  
Tel: 408-436-7950 Fax: 408-436-7955

#### Toronto

6285 Northam Drive, Suite 108  
Mississauga, Ontario L4V 1X5, Canada  
Tel: 905-673-0699 Fax: 905-673-6509

### ASIA/PACIFIC

#### Australia

Microchip Technology Australia Pty Ltd  
Suite 22, 41 Rawson Street  
Epping 2121, NSW  
Australia  
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

#### China - Beijing

Microchip Technology Consulting (Shanghai)  
Co., Ltd., Beijing Liaison Office  
Unit 915  
Bei Hai Wan Tai Bldg.  
No. 6 Chaoyangmen Beidajie  
Beijing, 100027, No. China  
Tel: 86-10-85282100 Fax: 86-10-85282104

#### China - Chengdu

Microchip Technology Consulting (Shanghai)  
Co., Ltd., Chengdu Liaison Office  
Rm. 2401, 24th Floor,  
Ming Xing Financial Tower  
No. 88 TIDU Street  
Chengdu 610016, China  
Tel: 86-28-86766200 Fax: 86-28-86766599

#### China - Fuzhou

Microchip Technology Consulting (Shanghai)  
Co., Ltd., Fuzhou Liaison Office  
Unit 28F, World Trade Plaza  
No. 71 Wusi Road  
Fuzhou 350001, China  
Tel: 86-591-7503506 Fax: 86-591-7503521

#### China - Shanghai

Microchip Technology Consulting (Shanghai)  
Co., Ltd.  
Room 701, Bldg. B  
Far East International Plaza  
No. 317 Xian Xia Road  
Shanghai, 200051  
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

#### China - Shenzhen

Microchip Technology Consulting (Shanghai)  
Co., Ltd., Shenzhen Liaison Office  
Rm. 1315, 13/F, Shenzhen Kerry Centre,  
Renminnan Lu  
Shenzhen 518001, China  
Tel: 86-755-82350361 Fax: 86-755-82366086

#### China - Hong Kong SAR

Microchip Technology Hongkong Ltd.  
Unit 901-6, Tower 2, Metroplaza  
223 Hing Fong Road  
Kwai Fong, N.T., Hong Kong  
Tel: 852-2401-1200 Fax: 852-2401-3431

#### India

Microchip Technology Inc.  
India Liaison Office  
Divyasree Chambers  
1 Floor, Wing A (A3/A4)  
No. 11, O'Shaugnessy Road  
Bangalore, 560 025, India  
Tel: 91-80-2290061 Fax: 91-80-2290062

### Japan

Microchip Technology Japan K.K.  
Benex S-1 6F  
3-18-20, Shinyokohama  
Kohoku-Ku, Yokohama-shi  
Kanagawa, 222-0033, Japan  
Tel: 81-45-471-6166 Fax: 81-45-471-6122

### Korea

Microchip Technology Korea  
168-1, Youngbo Bldg. 3 Floor  
Samsung-Dong, Kangnam-Ku  
Seoul, Korea 135-882  
Tel: 82-2-554-7200 Fax: 82-2-558-5934

### Singapore

Microchip Technology Singapore Pte Ltd.  
200 Middle Road  
#07-02 Prime Centre  
Singapore, 188980  
Tel: 65-6334-8870 Fax: 65-6334-8850

### Taiwan

Microchip Technology (Barbados) Inc.,  
Taiwan Branch  
11F-3, No. 207  
Tung Hua North Road  
Taipei, 105, Taiwan  
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

### EUROPE

#### Austria

Microchip Technology Austria GmbH  
Durisolstrasse 2  
A-4600 Wels  
Austria  
Tel: 43-7242-2244-399  
Fax: 43-7242-2244-393

#### Denmark

Microchip Technology Nordic ApS  
Regus Business Centre  
Lautrup høj 1-3  
Ballerup DK-2750 Denmark  
Tel: 45 4420 9895 Fax: 45 4420 9910

#### France

Microchip Technology SARL  
Parc d'Activite du Moulin de Massy  
43 Rue du Saule Trapu  
Batiment A - 1er Eteage  
91300 Massy, France  
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

#### Germany

Microchip Technology GmbH  
Steinheilstrasse 10  
D-85737 Ismaning, Germany  
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

#### Italy

Microchip Technology SRL  
Centro Direzionale Colleoni  
Palazzo Taurus 1 V. Le Colleoni 1  
20041 Agrate Brianza  
Milan, Italy  
Tel: 39-039-65791-1 Fax: 39-039-6899883

#### United Kingdom

Microchip Ltd.  
505 Eskdale Road  
Winnersh Triangle  
Wokingham  
Berkshire, England RG41 5TU  
Tel: 44 118 921 5869 Fax: 44-118 921-5820

08/01/02