



---

# **PIC18F13K50/14K50**

## **Data Sheet**

**20-Pin USB Flash Microcontrollers  
with nanoWatt XLP™ Technology**

---

---

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. **MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE.** Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

#### Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, rfPIC, SmartShunt and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, nanoWatt XLP, PICkit, PICDEM, PICDEM.net, PICtail, PIC<sup>32</sup> logo, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

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

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

 Printed on recycled paper.

---

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

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



# MICROCHIP PIC18F1XK50/PIC18LF1XK50

## 20-Pin USB Flash Microcontrollers with nanoWatt XLP™ Technology

### Universal Serial Bus Features:

- USB V2.0 Compliant SIE
- Full Speed (12 Mb/s) and Low Speed (1.5 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 16 Endpoints (8 bidirectional)
- 256-byte Dual Access RAM for USB
- Input-change interrupt on D+/D- for detecting physical connection to USB host

### High Performance RISC CPU:

- C Compiler Optimized Architecture:
  - Optional extended instruction set designed to optimize re-entrant code
  - 256 bytes, data EEPROM
  - Up to 16 Kbytes linear program memory addressing
  - Up to 768 bytes linear data memory addressing
- Priority levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier

### Flexible Oscillator Structure:

- CPU divider to run the core slower than the USB peripheral
- 16 MHz Internal Oscillator Block:
  - Software selectable frequencies, 31 kHz to 16 MHz
  - Provides a complete range of clock speeds from 31 kHz to 32 MHz when used with PLL
  - User tunable to compensate for frequency drift
- Four Crystal modes, up to 48 MHz
- External Clock modes, up to 48 MHz
- 4X Phase Lock Loop (PLL)
- Secondary oscillator using Timer1 at 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if primary or secondary oscillator stops
- Two-speed Oscillator Start-up

### Special Microcontroller Features:

- Full 5.5V Operation – PIC18F1XK50
- 1.8V-3.6V Operation – PIC18LF1XK50
- Self-programmable under Software Control
- Programmable Brown-out Reset (BOR)
  - With software enable option
- Extended Watchdog Timer (WDT)
  - Programmable period from 4ms to 131s
- Single-supply 3V In-Circuit Serial Programming™ (ICSP™) via two pins

### Extreme Low-Power Management

#### PIC18LF1XK50 with nanoWatt XLP™:

- Sleep Mode: 24nA
- Watchdog Timer: 450nA
- Timer1 Oscillator: 790nA @ 32 kHz

### Analog Features:

- Analog-to-Digital Converter (ADC) module :
  - 10-bit resolution, 9 external channels
  - Auto acquisition capability
  - Conversion available during Sleep
  - Internal 1.024V Fixed Voltage Reference (FVR) channel
  - Independent input multiplexing
- Dual Analog Comparators
  - Rail-to-rail operation
  - Independent input multiplexing
- Voltage Reference module:
  - Programmable (% of VDD), 16 steps
  - Two 16-level voltage ranges using VREF pins
  - Programmable Fixed Voltage Reference (FVR), 3 levels
- On-chip 3.2V LDO Regulator – (PIC18F1XK50)

### Peripheral Highlights:

- 14 I/O Pins plus 1 Input-only pin:
  - High-current sink/source 25 mA/25 mA
  - 7 Programmable weak pull-ups
  - 7 Programmable Interrupt-on-change pins
  - 3 programmable external interrupts
  - Programmable slew rate
- Enhanced Capture/Compare/PWM (ECCP) module:
  - One, two, three, or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and Auto-restart
- Master Synchronous Serial Port (MSSP) module:
  - 3-wire SPI (supports all 4 modes)
  - I<sup>2</sup>C™ Master and Slave modes (Slave mode address masking)
- Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module:
  - Supports RS-485, RS-232 and LIN 2.0
  - RS-232 operation using internal oscillator
  - Auto-Baud Detect
  - Auto-Wake-up on Break
- SR Latch mode

# PIC18F1XK50/PIC18LF1XK50

Device	Program Memory		Data Memory		I/O <sup>(1)</sup>	10-bit A/D (ch) <sup>(2)</sup>	ECCP (PWM)	MSSP		EUSART	Comp.	Timers 8/16-bit	USB
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I <sup>2</sup> C <sup>TM</sup>				
PIC18F13K50/ PIC18LF13K50	8K	4096	512 <sup>(3)</sup>	256	15	11	1	Y	Y	1	2	1/3	Y
PIC18F14K50/ PIC18LF14K50	16K	8192	768 <sup>(3)</sup>	256	15	11	1	Y	Y	1	2	1/3	Y

**Note 1:** One pin is input only.

**2:** Channel count includes internal Fixed Voltage Reference (FVR) and Programmable Voltage Reference (CVREF) channels.

**3:** Includes the dual port RAM used by the USB module which is shared with the data memory.

## Pin Diagrams

20-pin PDIP, SSOP, SOIC (300 MIL)

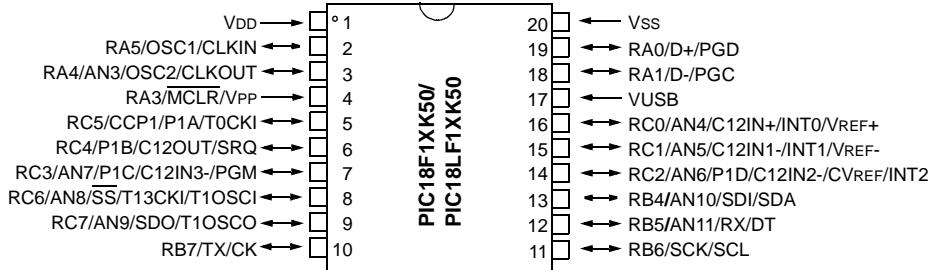


TABLE 1: PIC18F1XK50/PIC18LF1XK50 PIN SUMMARY

Pin	I/O	Analog	Comparator	Reference	ECCP	EUSART	MSSP	Timers	Interrupts	Pull-up	USB	Basic
19	RA0								IOC		D+	PGD
18	RA1								IOC		D-	PGC
4	RA3 <sup>(1)</sup>								IOC	Y		MCLR/VPP
3	RA4	AN3							IOC	Y		OSC2/CLKOUT
2	RA5								IOC	Y		OSC1/CLKIN
13	RB4	AN10					SDI/SDA		IOC	Y		
12	RB5	AN11				RX/DT			IOC	Y		
11	RB6						SCL/SCK		IOC	Y		
10	RB7					TX/CK			IOC	Y		
16	RC0	AN4	C12IN+	VREF+					INT0			
15	RC1	AN5	C12IN1-	VREF-					INT1			
14	RC2	AN6	C12IN2-	CVREF	P1D				INT2			
7	RC3	AN7	C12IN3-		P1C							PGM
6	RC4		C12OUT		P1B							SRQ
5	RC5				CCP1/P1A			T0CKI				
8	RC6	AN8					SS	T13CKI/T1OSCI				
9	RC7	AN9					SDO	T1OSCO				
17										VUSB		
1											VDD	
20											VSS	

**Note 1:** Input only

# PIC18F1XK50/PIC18LF1XK50

---

## Table of Contents

1.0	Device Overview .....	5
2.0	Oscillator Module (With Fail-Safe Clock Monitor).....	11
3.0	Memory Organization.....	25
4.0	Flash Program Memory.....	47
5.0	Data EEPROM Memory .....	57
6.0	8 x 8 Hardware Multiplier.....	61
7.0	Interrupts .....	63
8.0	Low Dropout (LDO) Voltage Regulator .....	77
9.0	I/O Ports .....	79
10.0	Timer0 Module .....	97
11.0	Timer1 Module .....	101
12.0	Timer2 Module .....	107
13.0	Timer3 Module .....	109
14.0	Enhanced Capture/Compare/PWM (ECCP) Module.....	113
15.0	Master Synchronous Serial Port (MSSP) Module .....	135
16.0	Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) .....	177
17.0	Analog-to-Digital Converter (ADC) Module .....	205
18.0	Comparator Module.....	219
19.0	Power-Managed Modes .....	231
20.0	SR Latch.....	239
21.0	Voltage References .....	241
22.0	Universal Serial Bus (USB) .....	245
23.0	Reset .....	271
24.0	Special Features of the CPU.....	285
25.0	Instruction Set Summary .....	303
26.0	Development Support.....	353
27.0	Electrical Specifications.....	357
28.0	DC and AC Characteristics Graphs and Tables.....	391
29.0	Packaging Information.....	393
	Appendix A: Revision History.....	397
	Appendix B: Device Differences .....	397
	Index .....	399
	The Microchip Web Site .....	409
	Customer Change Notification Service .....	409
	Customer Support.....	409
	Reader Response .....	410
	Product Identification System .....	411

## TO OUR VALUED CUSTOMERS

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

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

### Most Current Data Sheet

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

<http://www.microchip.com>

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

### Errata

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

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

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

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

### Customer Notification System

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

## 1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F13K50
- PIC18LF13K50
- PIC18F14K50
- PIC18LF14K50

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Flash program memory. On top of these features, the PIC18F1XK50/PIC18LF1XK50 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

### 1.1 New Core Features

#### 1.1.1 nanoWatt XLP™ TECHNOLOGY

All of the devices in the PIC18F1XK50/PIC18LF1XK50 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 27.0 "Electrical Specifications"** for values.

#### 1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F1XK50/PIC18LF1XK50 family offer ten different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes, using crystals or ceramic resonators
- External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O)
- External RC Oscillator modes with the same pin options as the External Clock modes
- An internal oscillator block which contains a 16 MHz HFINTOSC oscillator and a 31 kHz LFINTOSC oscillator which together provide 8 user selectable clock frequencies, from 31 kHz to 16 MHz. This option frees the two oscillator pins for use as additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the high-speed crystal and internal oscillator modes, which allows clock speeds of up to 48 MHz. Used with the internal oscillator, the PLL gives users a complete selection of clock speeds, from 31 kHz to 32 MHz – all without using an external crystal or clock circuit.

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

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

# PIC18F1XK50/PIC18LF1XK50

---

## 1.2 Other Special Features

- **Memory Endurance:** The Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 1K for program memory and 100K for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-programmability:** These devices can write to their own program memory spaces under internal software control. Using a bootloader routine located in the code protected Boot Block, it is possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18F1XK50/ PIC18LF1XK50 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Addressing mode. This extension has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **Enhanced CCP module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include:
  - Auto-Shutdown, for disabling PWM outputs on interrupt or other select conditions
  - Auto-Restart, to reactivate outputs once the condition has cleared
  - Output steering to selectively enable one or more of 4 outputs to provide the PWM signal.
- **Enhanced Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include automatic baud rate detection and a 16-bit Baud Rate Generator for improved resolution.
- **10-bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduce code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit postscaler, allowing an extended time-out range that is stable across operating voltage and temperature. See **Section 27.0 “Electrical Specifications”** for time-out periods.

## 1.3 Details on Individual Family Members

Devices in the PIC18F1XK50/PIC18LF1XK50 family are available in 20-pin packages. Block diagrams for the two groups are shown in Figure 1-1.

The devices are differentiated from each other in the following ways:

1. Flash program memory:
  - 8 Kbytes for PIC18F13K50/PIC18LF13K50
  - 16 Kbytes for PIC18F14K50/PIC18LF14K50
2. On-chip 3.2V LDO regulator for PIC18F13K50 and PIC18F14K50.

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

The pinouts for all devices are listed in Table 1 and I/O description are in Table 1-2.

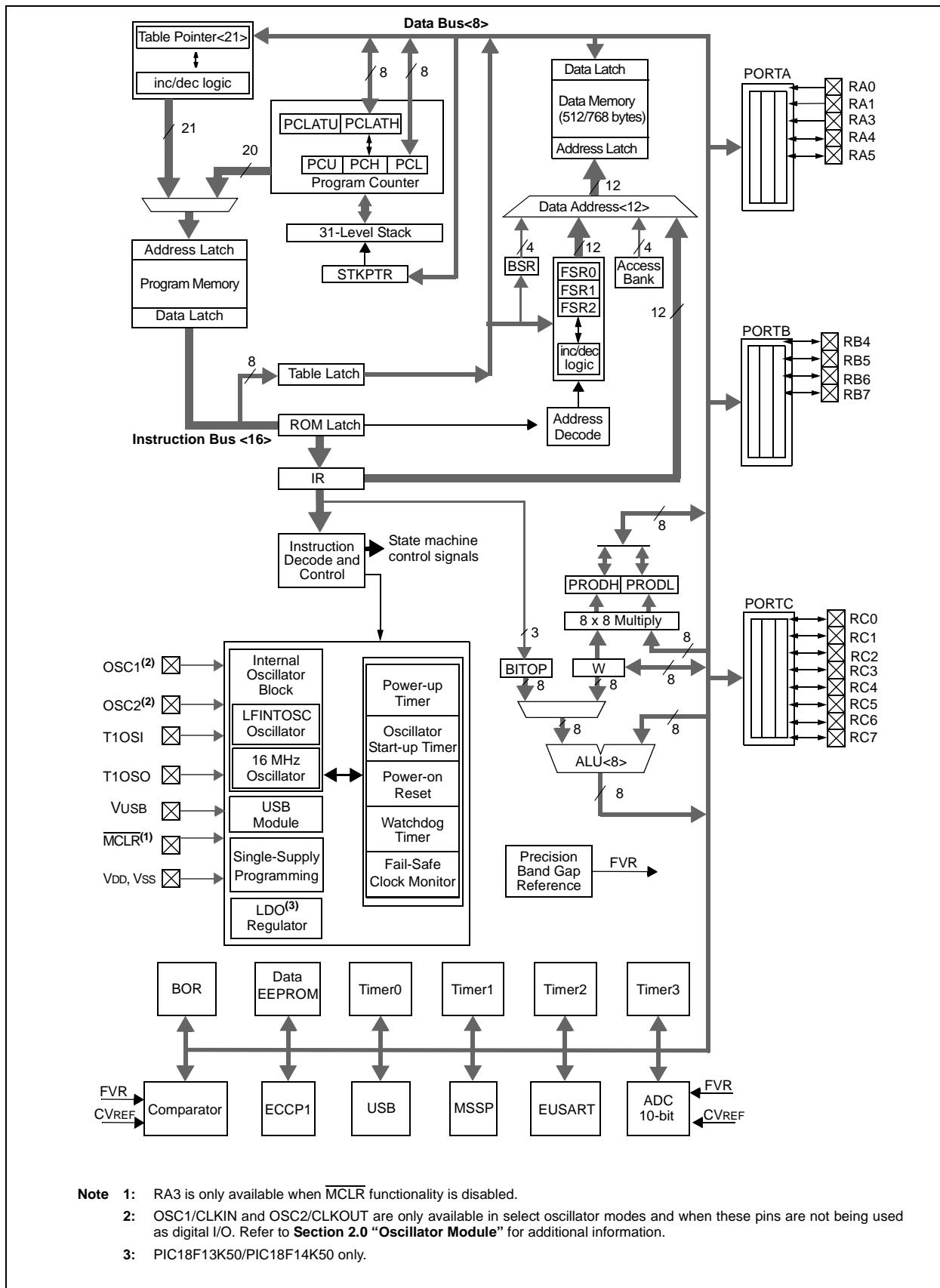
# PIC18F1XK50/PIC18LF1XK50

TABLE 1-1: DEVICE FEATURES FOR THE PIC18F1XK50/PIC18LF1XK50 (20-PIN DEVICES)

Features	PIC18F13K50	PIC18LF13K50	PIC18F14K50	PIC18LF14K50
LDO Regulator	Yes	No	Yes	No
Program Memory (Bytes)	8K		16K	
Program Memory (Instructions)	4096		8192	
Data Memory (Bytes)	512		768	
Operating Frequency		DC – 48 MHz		
Interrupt Sources		30		
I/O Ports		Ports A, B, C		
Timers		4		
Enhanced Capture/ Compare/PWM Modules		1		
Serial Communications		MSSP, Enhanced USART, USB		
10-Bit Analog-to-Digital Module		9 Input Channels		
Resets (and Delays)		POR, BOR, RESET Instruction, Stack Full, Stack Underflow, MCLR, WDT (PWRT, OST)		
Instruction Set		75 Instructions, 83 with Extended Instruction Set Enabled		
Packages		20-Pin PDIP, SSOP, SOIC (300 mil)		

# PIC18F1XK50/PIC18LF1XK50

FIGURE 1-1: PIC18F1XK50/PIC18LF1XK50 BLOCK DIAGRAM



# PIC18F1XK50/PIC18LF1XK50

**TABLE 1-2: PIC18F1XK50/PIC18LF1XK50 PINOUT I/O DESCRIPTIONS**

Pin Name	Pin Number	Pin Type	Buffer Type	Description
RA0/D+/PGD RA0 D+ PGD	19	I I/O I/O	TTL XCVR ST	Digital input USB differential plus line (input/output) ICSP™ programming data pin
RA1/D-/PGC RA1 D- PGC	18	I I/O I/O	TTL XCVR ST	Digital input USB differential minus line (input/output) ICSP™ programming clock pin
RA3/MCLR/VPP RA3 MCLR VPP	4	I I P	ST ST —	Master Clear (input) or programming voltage (input) Digital input Active-low Master Clear with internal pull-up High voltage programming input
RA4/AN3/OSC2/CLKOUT RA4 AN3 OSC2 CLKOUT	3	I/O I O O	TTL Analog XTAL CMOS	Digital I/O ADC channel 3 Oscillator crystal output. Connect to crystal or resonator in Crystal Oscillator mode In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate
RA5/OSC1/CLKIN RA5 OSC1  CLKIN	2	I/O I  I	TTL XTAL  CMOS	Digital I/O Oscillator crystal input or external clock input ST buffer when configured in RC mode; analog otherwise External clock source input. Always associated with the pin function OSC1 (See related OSC1/CLKIN, OSC2, CLKOUT pins)
RB4/AN10/SDI/SDA RB4 AN10 SDI SDA	13	I/O I I I/O	TTL Analog ST ST	Digital I/O ADC channel 10 SPI data in I <sup>2</sup> C™ data I/O
RB5/AN11/RX/DT RB5 AN11 RX DT	12	I/O I I I/O	TLL Analog ST ST	Digital I/O ADC channel 11 EUSART asynchronous receive EUSART synchronous data (see related RX/TX)
RB6/SCK/SCI RB6 SCK SCI	11	I/O I/O I/O	TLL 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
RB7/TX/CK RB7 TX CK	10	I/O O I/O	TLL CMOS ST	Digital I/O EUSART asynchronous transmit EUSART synchronous clock (see related RX/DT)

**Legend:** TTL = TTL compatible input

ST = Schmitt Trigger input

O = Output

XTAL= Crystal Oscillator

CMOS = CMOS compatible input or output

I = Input

P = Power

XCVR = USB Differential Transceiver

# PIC18F1XK50/PIC18LF1XK50

TABLE 1-2: PIC18F1XK50/PIC18LF1XK50 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
RC0/AN4/C12IN+/INT0/VREF+	16	I/O I I I I	ST Analog Analog ST Analog	Digital I/O ADC channel 4 Comparator C1 and C2 non-inverting input External interrupt 0 Comparator reference voltage (high) input
RC1/AN5/C12IN-/INT1/VREF-	15	I/O I I I I	ST Analog Analog ST Analog	Digital I/O ADC channel 5 Comparator C1 and C2 non-inverting input External interrupt 0 Comparator reference voltage (low) input
RC2/AN6/P1D/C12IN2-/CVREF/INT2	14	I/O I O I O I	ST Analog CMOS Analog Analog ST	Digital I/O ADC channel 6 Enhanced CCP1 PWM output Comparator C1 and C2 inverting input Comparator reference voltage output External interrupt 0
RC3/AN7/P1C/C12IN3-/PGM	7	I/O I O I I/O	ST Analog CMOS Analog ST	Digital I/O ADC channel 7 Enhanced CCP1 PWM output Comparator C1 and C2 inverting input Low-Voltage ICSP Programming enable pin
RC4/P1B/C12OUT/SRQ	6	I/O O O O	ST CMOS CMOS CMOS	Digital I/O Enhanced CCP1 PWM output Comparator C1 and C2 output SR Latch output
RC5/CCP1/P1A/T0CKI	5	I/O I/O O I	ST ST CMOS ST	Digital I/O Capture 1 input/Compare 1 output/PWM 1 output Enhanced CCP1 PWM output Timer0 external clock input
RC6/AN8/SS/T13CKI/T1OSCI	8	I/O I I I I	ST Analog TTL ST XTAL	Digital I/O ADC channel 8 SPI slave select input Timer0 and Timer3 external clock input Timer1 oscillator input
RC7/AN9/SDO/T1OSCO	9	I/O I O O	ST Analog CMOS XTAL	Digital I/O ADC channel 9 SPI data out Timer1 oscillator output
VSS	20	P	—	Ground reference for logic and I/O pins
VDD	1	P	—	Positive supply for logic and I/O pins
VUSB	17	P	—	Positive supply for USB transceiver

Legend: TTL = TTL compatible input

ST = Schmitt Trigger input

O = Output

XTAL= Crystal Oscillator

CMOS = CMOS compatible input or output

I = Input

P = Power

XCVR = USB Differential Transceiver

## 2.0 OSCILLATOR MODULE

### 2.1 Overview

The oscillator module has a variety of clock sources and features that allow it to be used in a wide range of applications, maximizing performance and minimizing power consumption. Figure 2-1 illustrates a block diagram of the oscillator module.

Key features of the oscillator module include:

- System Clock Selection
  - Primary External Oscillator
  - Secondary External Oscillator
  - Internal Oscillator
- Oscillator Start-up Timer
- System Clock Selection
- Clock Switching
- 4x Phase Lock Loop Frequency Multiplier
- CPU Clock Divider
- USB Operation
  - Low Speed
  - Full Speed
- Two-Speed Start-up Mode
- Fail-Safe Clock Monitoring

### 2.2 System Clock Selection

The SCS bits of the OSCCON register select between the following clock sources:

- Primary External Oscillator
- Secondary External Oscillator
- Internal Oscillator

**Note:** The frequency of the system clock will be referred to as Fosc throughout this document.

**TABLE 2-1: SYSTEM CLOCK SELECTION**

Configuration	Selection
SCS <1:0>	System Clock
1x	Internal Oscillator
01	Secondary External Oscillator
00 (Default after Reset)	Oscillator defined by FOSC<3:0>

The default state of the SCS bits sets the system clock to be the oscillator defined by the FOSC bits of the CONFIG1H Configuration register. The system clock will always be defined by the FOSC bits until the SCS bits are modified in software.

When the Internal Oscillator is selected as the system clock, the IRFC bits of the OSCCON register and the INTSRC bit of the OSCTUNE register will select either the LFINTOSC or the HFINTOSC. The LFINTOSC is selected when the IRFC<2:0> = 000 and the INTSRC bit is clear. All other combinations of the IRFC bits and the INTSRC bit will select the HFINTOSC as the system clock.

### 2.3 Primary External Oscillator

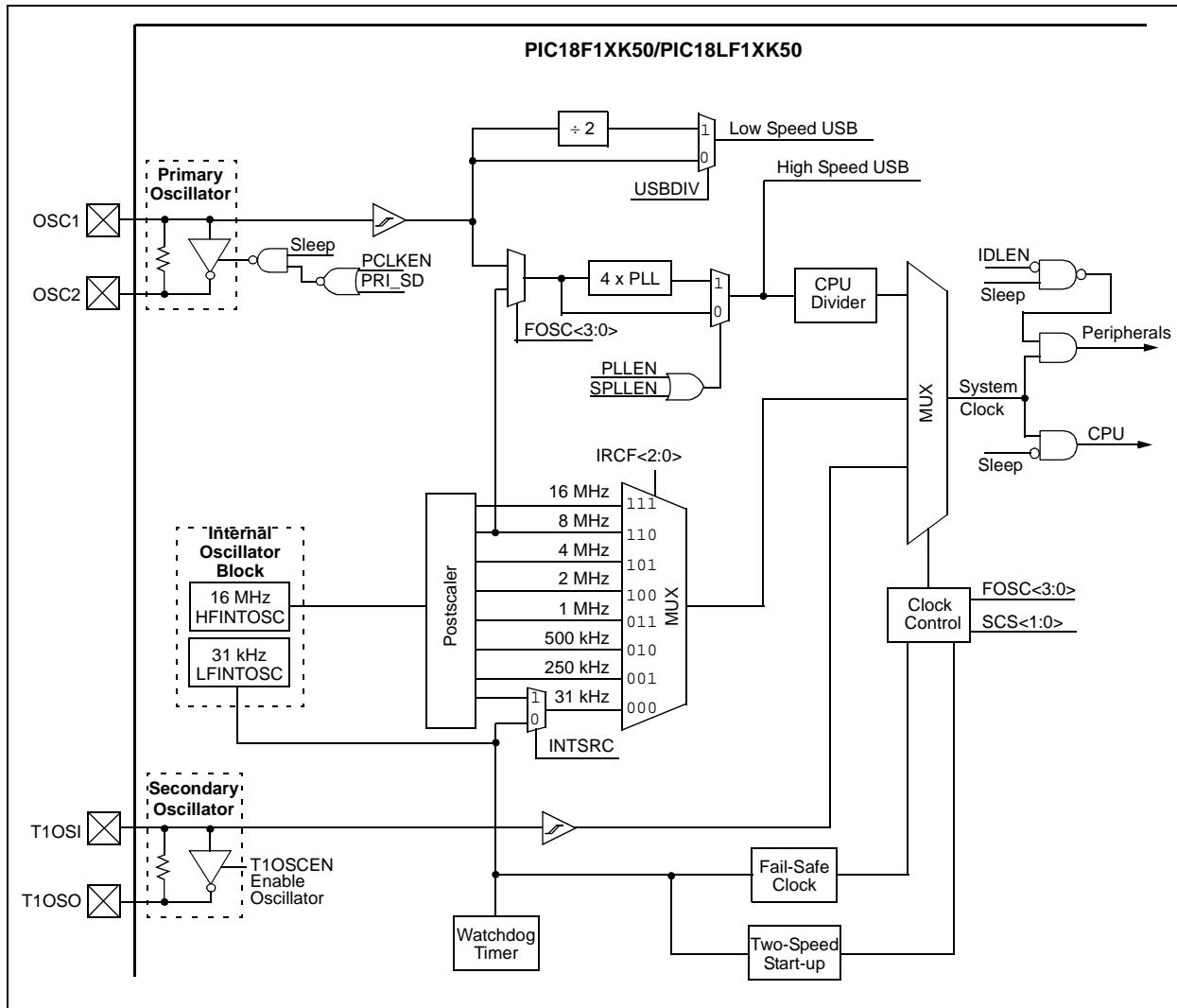
The Primary External Oscillator's mode of operation is selected by setting the FOSC<3:0> bits of the CONFIG1H Configuration register. The oscillator can be set to the following modes:

- LP: Low-Power Crystal
- XT: Crystal/Ceramic Resonator
- HS: High-Speed Crystal Resonator
- RC: External RC Oscillator
- EC: External Clock

Additionally, the Primary External Oscillator may be shut-down under firmware control to save power.

# PIC18F1XK50/PIC18LF1XK50

FIGURE 2-1: PIC® MCU CLOCK SOURCE BLOCK DIAGRAM



### 2.3.1 PRIMARY EXTERNAL OSCILLATOR SHUT-DOWN

The Primary External Oscillator can be enabled or disabled via software. To enable software control of the Primary External Oscillator, the PCLKEN bit of the CONFIG1H Configuration register must be set. With the PCLKEN bit set, the Primary External Oscillator is controlled by the PRI\_SD bit of the OSCCON2 register. The Primary External Oscillator will be enabled when the PRI\_SD bit is set, and disabled when the PRI\_SD bit is clear.

**Note:** The Primary External Oscillator cannot be shut down when it is selected as the System Clock. To shut down the oscillator, the system clock source must be either the Secondary Oscillator or the Internal Oscillator.

### 2.3.2 LP, XT AND HS OSCILLATOR MODES

The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to OSC1 and OSC2 (Figure 2-2). The mode selects a low, medium or high gain setting of the internal inverter-amplifier to support various resonator types and speed.

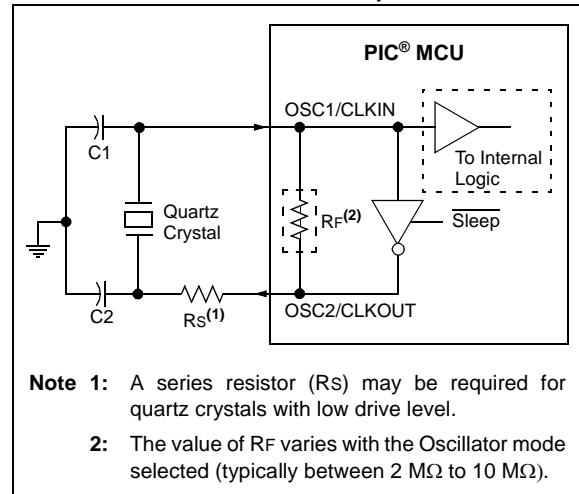
**LP** Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is best suited to drive resonators with a low drive level specification, for example, tuning fork type crystals.

**XT** Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. XT mode current consumption is the medium of the three modes. This mode is best suited to drive resonators with a medium drive level specification.

**HS** Oscillator mode selects the highest gain setting of the internal inverter-amplifier. HS mode current consumption is the highest of the three modes. This mode is best suited for resonators that require a high drive setting.

Figure 2-2 and Figure 2-3 show typical circuits for quartz crystal and ceramic resonators, respectively.

**FIGURE 2-2: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)**

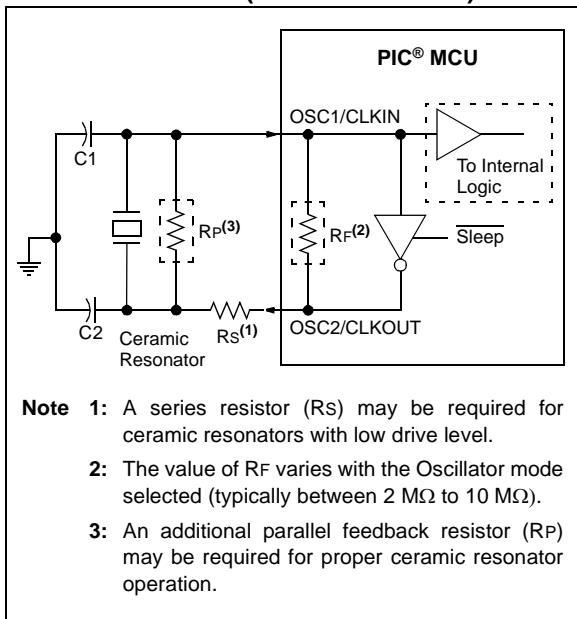


**Note 1:** Quartz crystal characteristics vary according to type, package and manufacturer. The user should consult the manufacturer data sheets for specifications and recommended application.

- 2:** Always verify oscillator performance over the VDD and temperature range that is expected for the application.
- 3:** For oscillator design assistance, reference the following Microchip Applications Notes:
  - AN826, "Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices" (DS00826)
  - AN849, "Basic PIC® Oscillator Design" (DS00849)
  - AN943, "Practical PIC® Oscillator Analysis and Design" (DS00943)
  - AN949, "Making Your Oscillator Work" (DS00949)

# PIC18F1XK50/PIC18LF1XK50

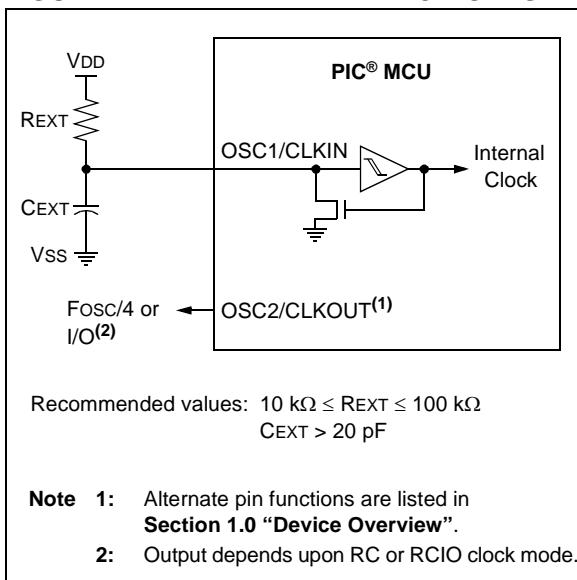
**FIGURE 2-3: CERAMIC RESONATOR OPERATION (XT OR HS MODE)**



### 2.3.3 EXTERNAL RC

The External Resistor-Capacitor (RC) mode supports the use of an external RC circuit. This allows the designer maximum flexibility in frequency choice while keeping costs to a minimum when clock accuracy is not required. In RC mode, the RC circuit connects to OSC1, allowing OSC2 to be configured as an IO or as CLKOUT. The CLKOUT function is selected by the FOSC bits of the CONFIG1H Configuration register. When OSC2 is configured as CLKOUT, the frequency at the pin is the frequency of the RC oscillator divided by 4. Figure 2-4 shows the external RC mode connections.

**FIGURE 2-4: EXTERNAL RC MODES**



The RC oscillator frequency is a function of the supply voltage, the resistor  $R_{EXT}$ , the capacitor  $C_{EXT}$  and the operating temperature. Other factors affecting the oscillator frequency are:

- Input threshold voltage variation
- Component tolerances
- Variation in capacitance due to packaging

### 2.3.4 EXTERNAL CLOCK

The External Clock (EC) mode allows an externally generated logic level clock to be used as the system's clock source. When operating in this mode, the external clock source is connected to the OSC1 allowing OSC2 to be configured as an I/O or as CLKOUT. The CLKOUT function is selected by the FOSC bits of the CONFIG1H Configuration register. When OSC2 is configured as CLKOUT, the frequency at the pin is the frequency of the EC oscillator divided by 4.

Three different power settings are available for EC mode. The power settings allow for a reduced  $IDD$  of the device, if the EC clock is known to be in a specific range. If there is an expected range of frequencies for the EC clock, select the power mode for the highest frequency.

EC	Low power	$0 - 250\text{ kHz}$
EC	Medium power	$250\text{ kHz} - 4\text{ MHz}$
EC	High power	$4 - 48\text{ MHz}$

### 2.4 Secondary External Oscillator

The Secondary External Oscillator is designed to drive an external  $32.768\text{ kHz}$  crystal. This oscillator is enabled or disabled by the T1OSCEN bit of the T1CON register. See **Section 11.0 “Timer1 Module”** for more information.

## 2.5 Internal Oscillator

The internal oscillator module contains two independent oscillators which are:

- LFINTOSC: Low-Frequency Internal Oscillator
- HFINTOSC: High-Frequency Internal Oscillator

When operating with either oscillator, OSC1 will be an I/O and OSC2 will be either an I/O or CLKOUT. The CLKOUT function is selected by the FOSC bits of the CONFIG1H Configuration register. When OSC2 is configured as CLKOUT, the frequency at the pin is the frequency of the Internal Oscillator divided by 4.

### 2.5.1 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a 31 kHz internal clock source. The LFINTOSC oscillator is the clock source for:

- Power-up Timer
- Watchdog Timer
- Fail-Safe Clock Monitor

The LFINTOSC is enabled when any of the following conditions are true:

- Power-up Timer is enabled (PWRTE = 0)
- Watchdog Timer is enabled (WDTEN = 1)
- Watchdog Timer is enabled by software (WDTEN = 0 and SWDTEN = 1)
- Fail-Safe Clock Monitor is enabled (FCMEM = 1)
- SCS1 = 1 and IRCF<2:0> = 000 and INTSRC = 0
- FOSC<3:0> selects the internal oscillator as the primary clock and IRCF<2:0> = 000 and INTSRC = 0
- IESO = 1 (Two-Speed Start-up) and IRCF<2:0> = 000 and INTSRC = 0

### 2.5.2 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a precision oscillator that is factory-calibrated to operate at 16 MHz. The output of the HFINTOSC connects to a postscaler and a multiplexer (see Figure 2-1). One of eight frequencies can be selected using the IRCF<2:0> bits of the OSCCON register. The following frequencies are available from the HFINTOSC:

- 16 MHz
- 8 MHz
- 4 MHz
- 2 MHz
- 1 MHz (Default after Reset)
- 500 kHz
- 250 kHz
- 31 kHz

The HFIOFS bit of the OSCCON register indicates whether the HFINTOSC is stable.

- Note 1:** Selecting 31 kHz from the HFINTOSC oscillator requires IRCF<2:0> = 000 and the INTSRC bit of the OSCTUNE register to be set. If the INTSRC bit is clear, the system clock will come from the LFINTOSC.
- 2:** Additional adjustments to the frequency of the HFINTOSC can be made via the OSCTUNE registers. See Register 2-3 for more details

The HFINTOSC is enabled if any of the following conditions are true:

- SCS1 = 1 and IRCF<2:0> ≠ 000
- SCS1 = 1 and IRCF<2:0> = 000 and INTSRC = 1
- FOSC<3:0> selects the internal oscillator as the primary clock and
  - IRCF<2:0> ≠ 000 or
  - IRCF<2:0> = 000 and INTSRC = 1
- IESO = 1 (Two-Speed Start-up) and
  - IRCF<2:0> ≠ 000 or
  - IRCF<2:0> = 000 and INTSRC = 1
- FCMEM = 1 (Fail Safe Clock Monitoring) and
  - IRCF<2:0> ≠ 000 or
  - IRCF<2:0> = 000 and INTSRC = 1

# PIC18F1XK50/PIC18LF1XK50

## 2.6 Oscillator Control

The Oscillator Control (OSCCON) (Register 2-1) and the Oscillator Control 2 (OSCCON2) (Register 2-2) registers control the system clock and frequency selection options.

### REGISTER 2-1: OSCCON: OSCILLATOR CONTROL REGISTER

R/W-0	R/W-0	R/W-1	R/W-1	R-q	R-0	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS <sup>(1)</sup>	HFIOFS	SCS1	SCS0
bit 7	bit 0						

#### Legend:

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

bit 7	<b>IDLEN:</b> Idle Enable bit 1 = Device enters Idle mode on SLEEP instruction 0 = Device enters Sleep mode on SLEEP instruction
bit 6-4	<b>IRCF&lt;2:0&gt;:</b> Internal Oscillator Frequency Select bits 111 = 16 MHz 110 = 8 MHz 101 = 4 MHz 100 = 2 MHz 011 = 1 MHz <sup>(3)</sup> 010 = 500 kHz 001 = 250 kHz 000 = 31 kHz <sup>(2)</sup>
bit 3	<b>OSTS:</b> Oscillator Start-up Time-out Status bit <sup>(1)</sup> 1 = Device is running from the clock defined by FOSC<2:0> of the CONFIG1 register 0 = Device is running from the internal oscillator (HFINTOSC or LFINTOSC)
bit 2	<b>HFIOFS:</b> HFINTOSC Frequency Stable bit 1 = HFINTOSC frequency is stable 0 = HFINTOSC frequency is not stable
bit 1-0	<b>SCS&lt;1:0&gt;:</b> System Clock Select bits 1x = Internal oscillator block 01 = Secondary (Timer1) oscillator 00 = Primary clock (determined by CONFIG1H[FOSC<3:0>]).

- Note 1:** Reset state depends on state of the IESO Configuration bit.  
**2:** Source selected by the INTSRC bit of the OSCTUNE register, see text.  
**3:** Default output frequency of HFINTOSC on Reset.

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 2-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R-x
—	—	—	—	—	PRI_SD	HFIOFL	LFIOFS
bit 7							

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

q = depends on condition

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-3      **Unimplemented:** Read as '0'

bit 2      **PRI\_SD:** Primary Oscillator Drive Circuit shutdown bit

1 = Oscillator drive circuit on

0 = Oscillator drive circuit off (zero power)

bit 1      **HFIOFL:** HFINTOSC Frequency Locked bit

1 = HFINTOSC is in lock

0 = HFINTOSC has not yet locked

bit 0      **LFIOFS:** LFINTOSC Frequency Stable bit

1 = LFINTOSC is stable

0 = LFINTOSC is not stable

# PIC18F1XK50/PIC18LF1XK50

## 2.6.1 OSCTUNE REGISTER

The HFINTOSC is factory calibrated, but can be adjusted in software by writing to the TUN<5:0> bits of the OSCTUNE register (Register 2-3).

The default value of the TUN<5:0> is '000000'. The value is a 6-bit two's complement number.

When the OSCTUNE register is modified, the HFINTOSC frequency will begin shifting to the new frequency. Code execution continues during this shift, while giving no indication that the shift has occurred.

OSCTUNE does not affect the LFINTOSC frequency. The operation of features that depend on the LFINTOSC clock source frequency, such as the Power-up Timer

(PWRT), Watchdog Timer (WDT), Fail-Safe Clock Monitor (FSCM) and peripherals, are *not* affected by the change in frequency.

The OSCTUNE register also implements the INTSRC and SPLLEN bits, which control certain features of the internal oscillator block.

The INTSRC bit allows users to select which internal oscillator provides the clock source when the 31 kHz frequency option is selected. This is covered in greater detail in **Section 2.5.1 “LFINTOSC”**.

The SPLLEN bit controls the operation of the frequency multiplier. For more details about the function of the SPLLEN bit see **Section 2.9 “4x Phase Lock Loop Frequency Multiplier”**

## REGISTER 2-3: OSCTUNE: OSCILLATOR TUNING 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
INTSRC	SPLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7

**INTSRC:** Internal Oscillator Low-Frequency Source Select bit

1 = 31.25 kHz device clock derived from 16 MHz HFINTOSC source (divide-by-512 enabled)

0 = 31 kHz device clock derived directly from LFINTOSC internal oscillator

bit 6

**SPLLEN:** Software Controlled Frequency Multiplier PLL bit

1 = PLL enabled (for HFINTOSC 8 MHz only)

0 = PLL disabled

bit 5-0

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

011111 = Maximum frequency

011110 =

•••

000001 =

000000 = Oscillator module is running at the factory calibrated frequency.

111111 =

•••

100000 = Minimum frequency

## 2.7 Oscillator Start-up Timer

The Primary External Oscillator, when configured for LP, XT or HS modes, incorporates an Oscillator Start-up Timer (OST). The OST ensures that the oscillator starts and provides a stable clock to the oscillator module. The OST times out when 1024 oscillations on OSC1 have occurred. During the OST period, with the system clock set to the Primary External Oscillator, the program counter does not increment suspending program execution. The OST period will occur following:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Wake-up from Sleep
- Oscillator being enabled
- Expiration of Power-up Timer (PWRT)

In order to minimize latency between external oscillator start-up and code execution, the Two-Speed Start-up mode can be selected. See **Section 2.12 “Two-Speed Start-up Mode”** for more information.

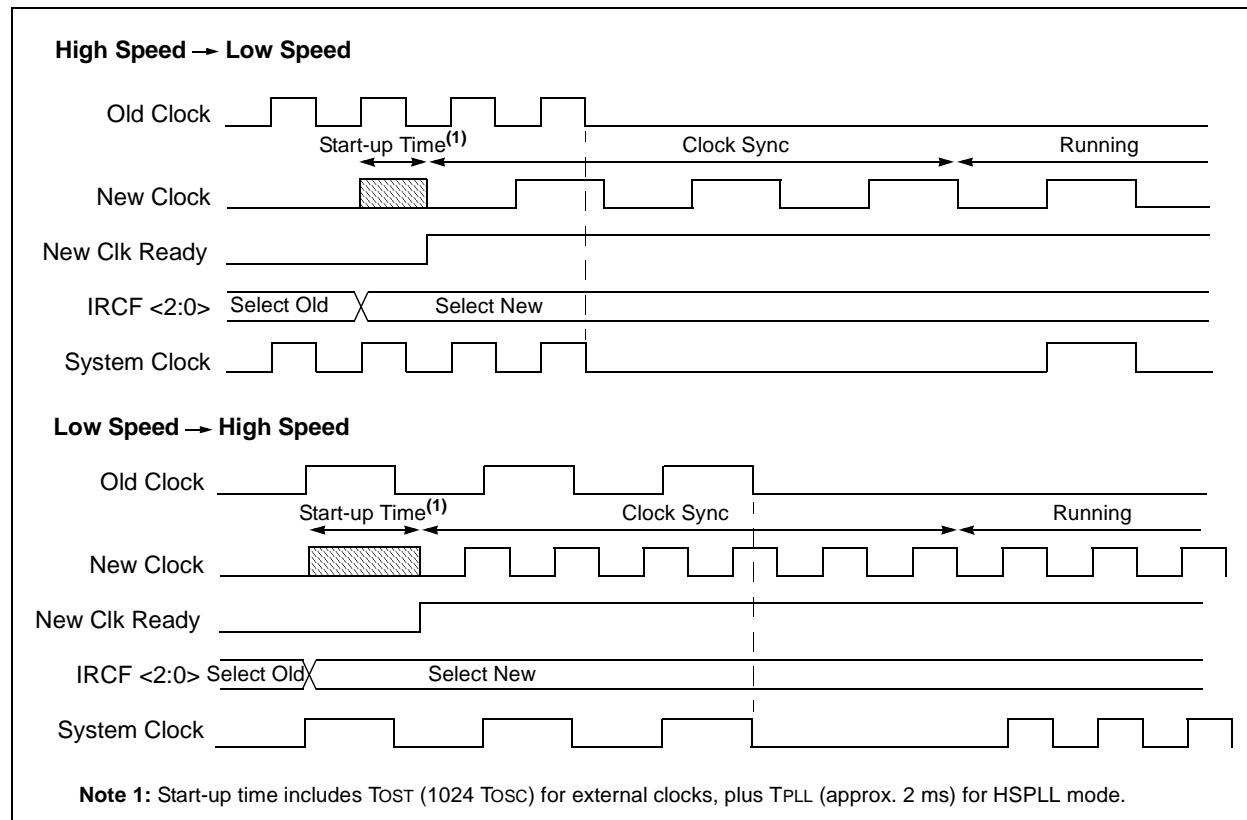
## 2.8 Clock Switching

The device contains circuitry to prevent clock “glitches” due to a change of the system clock source. To accomplish this, a short pause in the system clock occurs during the clock switch. If the new clock source is not stable (e.g., OST is active), the device will continue to execute from the old clock source until the new clock source becomes stable. The timing of a clock switch is as follows:

1. SCS<1:0> bits of the OSCCON register are modified.
2. The system clock will continue to operate from the old clock until the new clock is ready.
3. Clock switch circuitry waits for two consecutive rising edges of the old clock after the new clock is ready.
4. The system clock is held low, starting at the next falling edge of the old clock.
5. Clock switch circuitry waits for an additional two rising edges of the new clock.
6. On the next falling edge of the new clock, the low hold on the system clock is release and the new clock is switched in as the system clock.
7. Clock switch is complete.

Refer to Figure 2-5 for more details.

**FIGURE 2-5: CLOCK SWITCH TIMING**



# PIC18F1XK50/PIC18LF1XK50

TABLE 2-2: EXAMPLES OF DELAYS DUE TO CLOCK SWITCHING

Switch From	Switch To	Oscillator Delay
Sleep/POR	LFINTOSC HFINTOSC	Oscillator Warm-up Delay (TWARM)
Sleep/POR	LP, XT, HS	1024 clock cycles
Sleep/POR	EC, RC	8 Clock Cycles

## 2.9 4x Phase Lock Loop Frequency Multiplier

A Phase Locked Loop (PLL) circuit is provided as an option for users who wish to use a lower-frequency external oscillator or to operate at 32 MHz with the HFINTOSC. The PLL is designed for an input frequency from 4 MHz to 12 MHz. The PLL multiplies its input frequency by a factor of four when the PLL is enabled. This may be useful for customers who are concerned with EMI, due to high-frequency crystals.

Two bits control the PLL: the PLLEN bit of the CONFIG1H Configuration register and the SPLLEN bit of the OSCTUNE register. The PLL is enabled when the PLLEN bit is set and it is under software control when the PLLEN bit is cleared.

TABLE 2-3: PLL CONFIGURATION

PLLLEN	SPLLEN	PLL Status
1	x	PLL enabled
0	1	PLL enabled
0	0	PLL disabled

**Note:** The HFINTOSC may use the PLL when the postscaler is set to 8 MHz and the FOSC<3:0> bits of the CONFIG1H Configuration register are selected for Internal Oscillator operation.

## 2.10 CPU Clock Divider

The CPU Clock Divider allows the system clock to run at a slower speed than the Low/Full Speed USB module clock while sharing the same clock source. Only the oscillator defined by the settings of the FOSC bits of the CONFIG1H Configuration register may be used with the CPU Clock Divider. The CPU Clock Divider is controlled by the CPUDIV bits of the CONFIG1L Configuration register. Setting the CPUDIV bits will set the system clock to:

- Equal the clock speed of the USB module
- Half the clock speed of the USB module
- One third the clock speed of the USB module
- One fourth the clock speed of the USB module

For more information on the CPU Clock Divider, see Figure 2-1 and Register 24-1 CONFIG1L.

## 2.11 USB Operation

The USB module is designed to operate in two different modes:

- Low Speed
- Full Speed

Because of timing requirements imposed by the USB specifications, the Primary External Oscillator is required for the USB module. The FOSC bits of the CONFIG1H Configuration register must be set to either External Clock (EC) High-power or HS mode with a clock frequency of 6, 12 or 48 MHz.

### 2.11.1 LOW SPEED OPERATION

For Low Speed USB operation, a 6 MHz clock is required for the USB module. To generate the 6 MHz clock, only 2 Oscillator modes are allowed:

- EC High-power mode
- HS mode

Table 2-4 shows the recommended Clock mode for low-speed operation.

### 2.11.2 FULL-SPEED OPERATION

For full-speed USB operation, a 48 MHz clock is required for the USB module. To generate the 48 MHz clock, only 2 Oscillator modes are allowed:

- EC High-power mode
- HS mode

Table 2-5 shows the recommended Clock mode for full-speed operation.

# PIC18F1XK50/PIC18LF1XK50

TABLE 2-4: LOW SPEED USB CLOCK SETTINGS

Clock Mode	Clock Frequency	USBDIV	4x PLL Enabled	CPUDIV<1:0>	System Clock Frequency (MHz)
EC High/HS	12 MHz	1	Yes	00	48
				01	24
				10	16
				11	12
		0	No	00	12
				01	6
				10	4
				11	3
	6 MHz	1	Yes	00	24
				01	12
				10	8
				11	6
		0	No	00	6
				01	3
				10	2
				11	1.5

**Note:** The system clock frequency in Table 2-4 only applies if the OSCCON register bits SCS<1:0> = 00. By changing these bits, the system clock can operate down to 31 kHz.

TABLE 2-5: FULL-SPEED USB CLOCK SETTINGS

Clock Mode	Clock Frequency	4x PLL Enabled	CPUDIV<1:0>	System Clock Frequency (MHz)
EC High	48 MHz	No	00	48
			01	24
			10	16
			11	12
EC High/HS	12 MHz	Yes	00	48
			01	24
			10	16
			11	12

**Note:** The system clock frequency in the above table only applies if the OSCCON register bits SCS<1:0> = 00. By changing these bits, the system clock can operate down to 31 kHz.

## 2.12 Two-Speed Start-up Mode

Two-Speed Start-up mode provides additional power savings by minimizing the latency between external Oscillator Start-up Timer (OST) and code execution. In applications that make heavy use of the Sleep mode, Two-Speed Start-up will remove the OST period, which can reduce the overall power consumption of the device.

Two-Speed Start-up mode is enabled by setting the IESO bit of the CONFIG1H Configuration register. With Two-Speed Start-up enabled, the device will execute instructions using the internal oscillator during the Primary External Oscillator OST period.

When the system clock is set to the Primary External Oscillator and the oscillator is configured for LP, XT or HS modes, the device will not execute code during the OST period. The OST will suspend program execution until 1024 oscillations are counted. Two-Speed Start-up mode minimizes the delay in code execution by operating from the internal oscillator while the OST is active. The system clock will switch back to the Primary External Oscillator after the OST period has expired.

Two-speed Start-up will become active after:

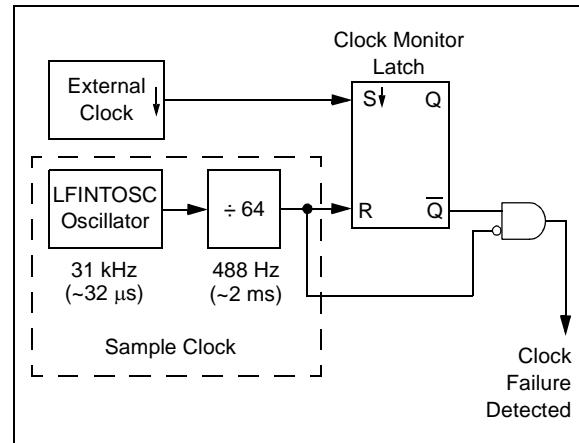
- Power-on Reset (POR)
- Power-up Timer (PWRT), if enabled
- Wake-up from Sleep

The OSTS bit of the OSCCON register reports which oscillator the device is currently using for operation. The device is running from the oscillator defined by the FOSC bits of the CONFIG1H Configuration register when the OSTS bit is set. The device is running from the internal oscillator when the OSTS bit is clear.

## 2.13 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM can detect oscillator failure any time after the Oscillator Start-up Timer (OST) has expired. The FSCM is enabled by setting the FCMEN bit in the CONFIG1H Configuration register. The FSCM is applicable to all external oscillator modes (LP, XT, HS, EC and RC).

FIGURE 2-6: FSCM BLOCK DIAGRAM



### 2.13.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64. See Figure 2-6. Inside the fail detector block is a latch. The external clock sets the latch on each falling edge of the external clock. The sample clock clears the latch on each rising edge of the sample clock. A failure is detected when an entire half-cycle of the sample clock elapses before the primary clock goes low.

### 2.13.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM switches the device clock to an internal clock source and sets the bit flag OSCFIF of the PIR2 register. The OSCFIF flag will generate an interrupt if the OSCFIE bit of the PIE2 register is also set. The device firmware can then take steps to mitigate the problems that may arise from a failed clock. The system clock will continue to be sourced from the internal clock source until the device firmware successfully restarts the external oscillator and switches back to external operation. An automatic transition back to the failed clock source will not occur.

The internal clock source chosen by the FSCM is determined by the IR<sub>CF</sub><2:0> bits of the OSCCON register. This allows the internal oscillator to be configured before a failure occurs.

### 2.13.3 FAIL-SAFE CONDITION CLEARING

The Fail-Safe condition is cleared by either one of the following:

- Any Reset
- By toggling the SCS1 bit of the OSCCON register

Both of these conditions restart the OST. While the OST is running, the device continues to operate from the INTOSC selected in OSCCON. When the OST times out, the Fail-Safe condition is cleared and the device automatically switches over to the external clock source. The Fail-Safe condition need not be cleared before the OSCFIF flag is cleared.

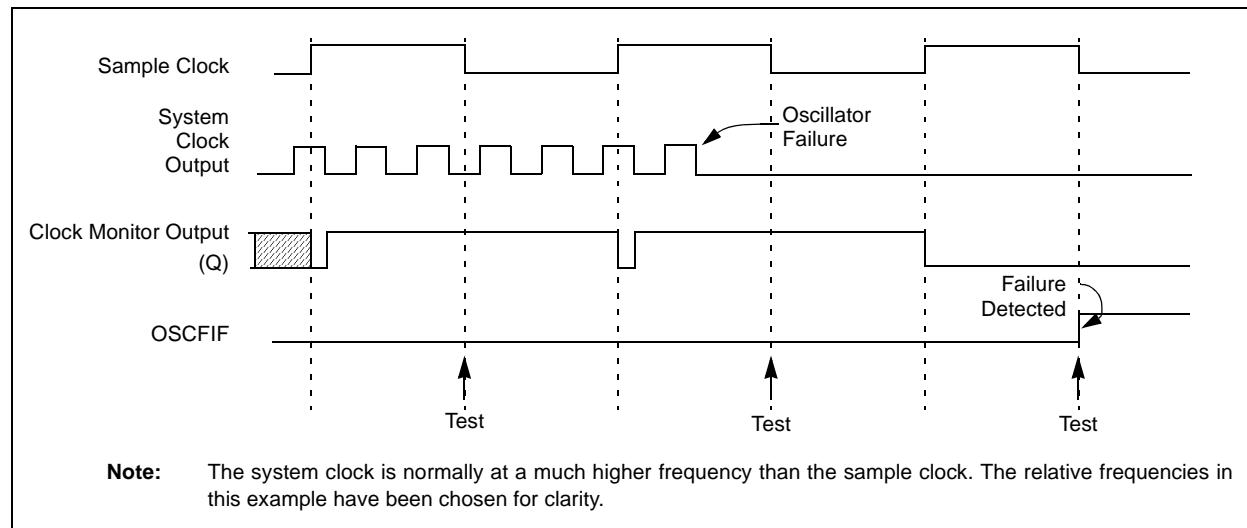
### 2.13.4 RESET OR WAKE-UP FROM SLEEP

The FSCM is designed to detect an oscillator failure after the Oscillator Start-up Timer (OST) has expired. The OST is used after waking up from Sleep and after

any type of Reset. The OST is not used with the EC or RC Clock modes so that the FSCM will be active as soon as the Reset or wake-up has completed. When the FSCM is enabled, the Two-Speed Start-up is also enabled. Therefore, the device will always be executing code while the OST is operating.

**Note:** Due to the wide range of oscillator start-up times, the Fail-Safe circuit is not active during oscillator start-up (i.e., after exiting Reset or Sleep). After an appropriate amount of time, the user should check the OSTS bit of the OSCCON register to verify the oscillator start-up and that the system clock switchover has successfully completed.

**FIGURE 2-7: FSCM TIMING DIAGRAM**



**TABLE 2-6: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK SOURCES**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
CONFIG1H	IESO	FCMEN	PCLKEN	PLLEN	FOSC3	FOSC2	FOSC1	FOSC0	283
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	283
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	HFIofs	SCS1	SCS0	283
OSCTUNE	INTSRC	SPLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0	283
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	283
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	285
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	101

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

**Note 1:** Other (non Power-up) Resets include MCLR Reset and Watchdog Timer Reset during normal operation.

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 3.0 MEMORY ORGANIZATION

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

- Program Memory
- Data RAM
- Data EEPROM

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

Additional detailed information on the operation of the Flash program memory is provided in **Section 4.0 “Flash Program Memory”**. Data EEPROM is discussed separately in **Section 5.0 “Data EEPROM Memory”**.

## 3.1 Program Memory Organization

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

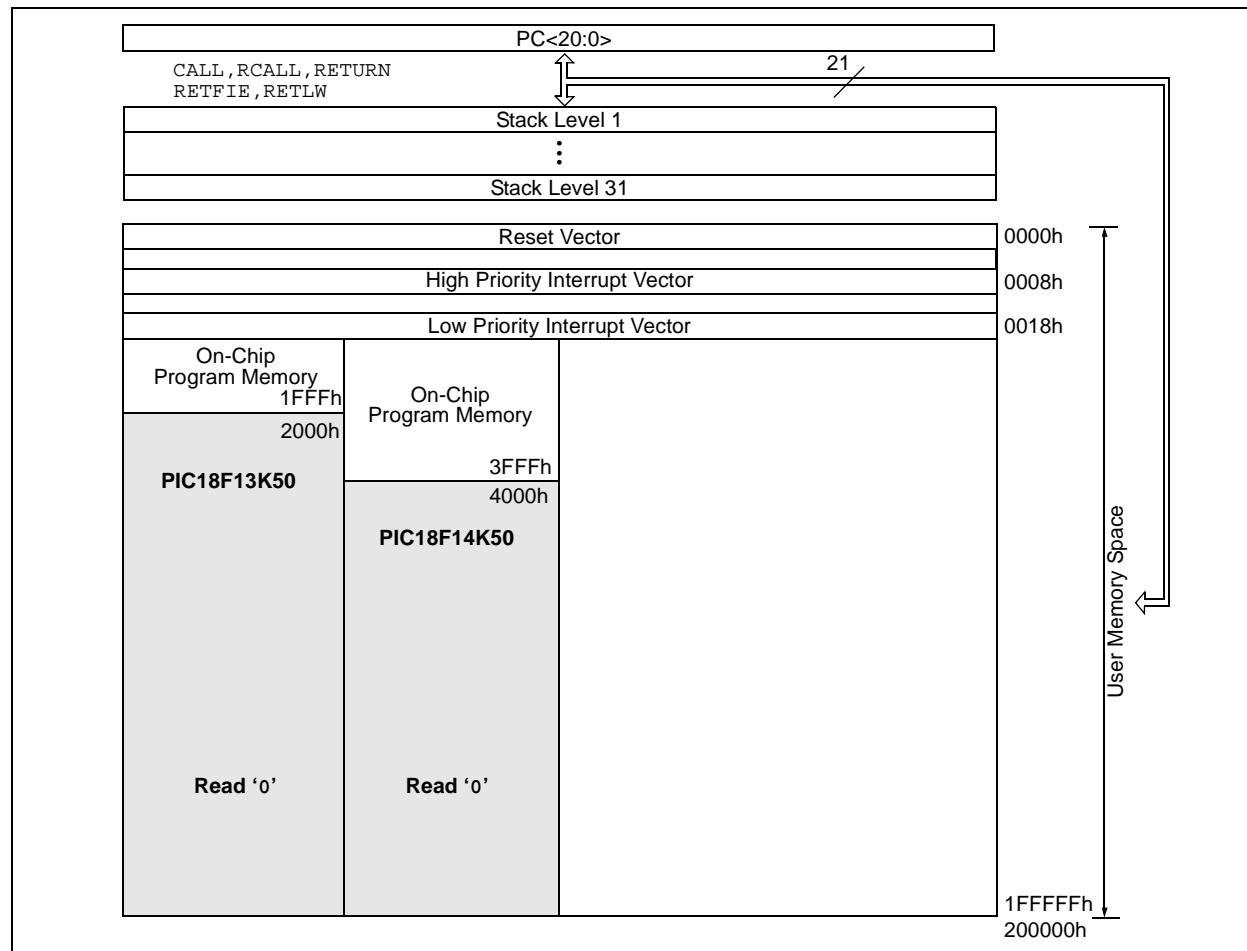
This family of devices contain the following:

- PIC18F13K50: 8 Kbytes of Flash Memory, up to 4,096 single-word instructions
- PIC18F14K50: 16 Kbytes of Flash Memory, up to 8,192 single-word instructions

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

The program memory map for PIC18F1XK50/PIC18LF1XK50 devices is shown in Figure 3-1. Memory block details are shown in Figure 24-2.

**FIGURE 3-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F1XK50/PIC18LF1XK50 DEVICES**



# PIC18F1XK50/PIC18LF1XK50

## 3.1.1 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<sub>15:8</sub> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<sub>20:16</sub> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 3.1.4.1 “Computed GOTO”**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit (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.

## 3.1.2 RETURN ADDRESS STACK

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

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

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

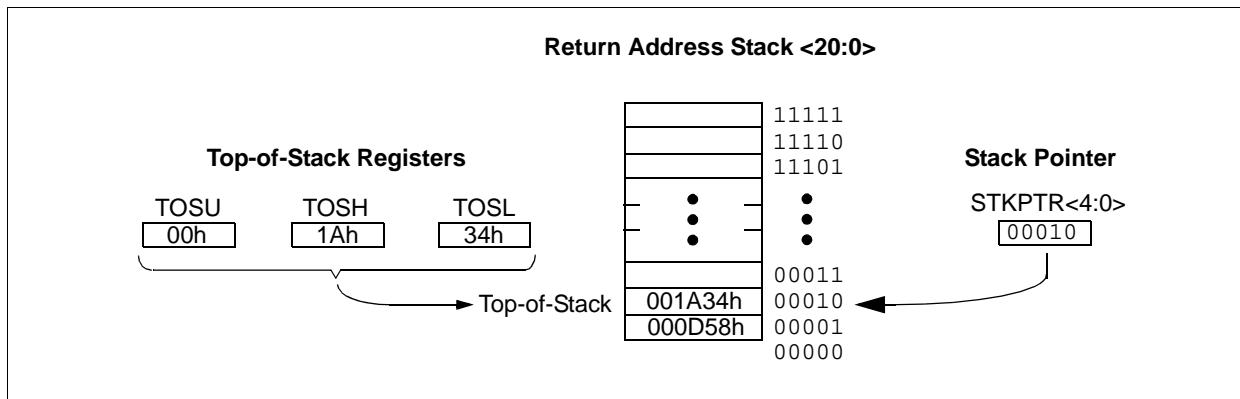
The Stack Pointer is initialized to ‘00000’ after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of ‘00000’; this is only a Reset value. Status bits indicate if the stack is full or has overflowed or has underflowed.

### 3.1.2.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 3-2). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

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

**FIGURE 3-2: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**



### 3.1.2.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 3-1) contains the Stack Pointer value, the STKFUL (stack full) bit and the STKUNF (stack underflow) bits. The value of the Stack Pointer can be 0 through 31. The Stack Pointer increments before values are pushed onto the stack and decrements after values are popped off the stack. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for return stack maintenance.

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

The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 24.1 “Configuration Bits”** for a description of the device Configuration bits.) If STVREN is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit and reset the device. The STKFUL bit will remain set and the Stack Pointer will be set to zero.

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

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

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

### 3.1.2.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable feature. The PIC18 instruction set includes two instructions, **PUSH** and **POP**, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

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

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

### REGISTER 3-1: STKPTR: STACK POINTER REGISTER

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

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented

C = Clearable only bit

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

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

# PIC18F1XK50/PIC18LF1XK50

### 3.1.2.4 Stack Full and Underflow Resets

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

### 3.1.3 FAST REGISTER STACK

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

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

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

Example 3-1 shows a source code example that uses the fast register stack during a subroutine call and return.

#### EXAMPLE 3-1: 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
```

### 3.1.4 LOOK-UP TABLES IN PROGRAM MEMORY

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

- Computed GOTO
- Table Reads

#### 3.1.4.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 3-2.

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

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

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

#### EXAMPLE 3-2: COMPUTED GOTO USING AN OFFSET VALUE

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

#### 3.1.4.2 Table Reads and Table Writes

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

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

Table read and table write operations are discussed further in **Section 4.1 “Table Reads and Table Writes”**.

## 3.2 PIC18 Instruction Cycle

### 3.2.1 CLOCKING SCHEME

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

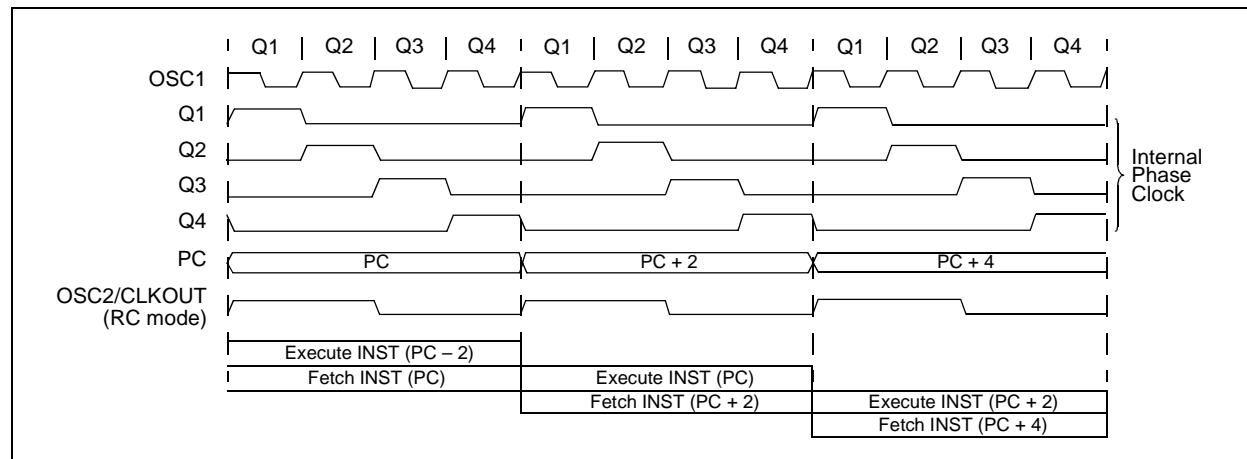
### 3.2.2 INSTRUCTION FLOW/PIPELINING

An “Instruction Cycle” consists of four Q cycles: Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), then two cycles are required to complete the instruction (Example 3-3).

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

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

**FIGURE 3-3: CLOCK/INSTRUCTION CYCLE**



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

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

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

# PIC18F1XK50/PIC18LF1XK50

## 3.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as either two bytes or four bytes in program memory. The Least Significant Byte (LSB) of an instruction word is always stored in a program memory location with an even address (LSb = 0). To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSb will always read '0' (see **Section 3.1.1 "Program Counter"**).

Figure 3-4 shows an example of how instruction words are stored in the program memory.

The **CALL** and **GOTO** instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 3-4 shows how the instruction **GOTO 0006h** is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 25.0 "Instruction Set Summary"** provides further details of the instruction set.

**FIGURE 3-4: INSTRUCTIONS IN PROGRAM MEMORY**

Program Memory Byte Locations →		Word Address ↓	
		LSB = 1	LSB = 0
Instruction 1:	MOVLW 055h		000000h
Instruction 2:	GOTO 0006h		000002h
Instruction 3:	MOVFF 123h, 456h	0Fh	00004h
		EFh	00006h
		F0h	00008h
		C1h	0000Ah
		F4h	0000Ch
			000023h
			000025h
			000027h
			000029h
			00002Bh
			00002Dh
			00002Fh
			000031h
			000033h

## 3.2.4 TWO-WORD INSTRUCTIONS

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

The use of '1111' in the 4 MSbs of an instruction specifies a special form of **NOP**. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed

and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a **NOP** is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 3-4 shows how this works.

**Note:** See **Section 3.6 "PIC18 Instruction Execution and the Extended Instruction Set"** for information on two-word instructions in the extended instruction set.

## EXAMPLE 3-4: TWO-WORD INSTRUCTIONS

CASE 1:	
Object Code	Source Code
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; No, skip this word
1111 0100 0101 0110	; Execute this word as a NOP
0010 0100 0000 0000	ADDWF REG3 ; continue code
CASE 2:	
Object Code	Source Code
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; Yes, execute this word
1111 0100 0101 0110	; 2nd word of instruction
0010 0100 0000 0000	ADDWF REG3 ; continue code

### 3.3 Data Memory Organization

**Note:** The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See **Section 3.5 “Data Memory and the Extended Instruction Set”** for more information.

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. Figure 3-5 and Figure 3-6 show the data memory organization for the PIC18F1XK50/PIC18LF1XK50 devices.

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

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

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to SFRs and the lower portion of GPR Bank 0 without using the Bank Select Register (BSR). **Section 3.3.3 “Access Bank”** provides a detailed description of the Access RAM.

#### 3.3.1 USB RAM

Part of the data memory is actually mapped to a special dual access RAM. When the USB module is disabled, the GPRs in these banks are used like any other GPR in the data memory space.

When the USB module is enabled, the memory in these banks is allocated as buffer RAM for USB operation. This area is shared between the microcontroller core and the USB Serial Interface Engine (SIE) and is used to transfer data directly between the two.

It is theoretically possible to use the areas of USB RAM that are not allocated as USB buffers for normal scratchpad memory or other variable storage. In practice, the dynamic nature of buffer allocation makes this risky at best. Additional information on USB RAM and buffer operation is provided in **Section 22.0 “Universal Serial Bus (USB)”**

#### 3.3.2 BANK SELECT REGISTER (BSR)

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

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

The value of the BSR indicates the bank in data memory; the 8 bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 3-5 and Figure 3-6.

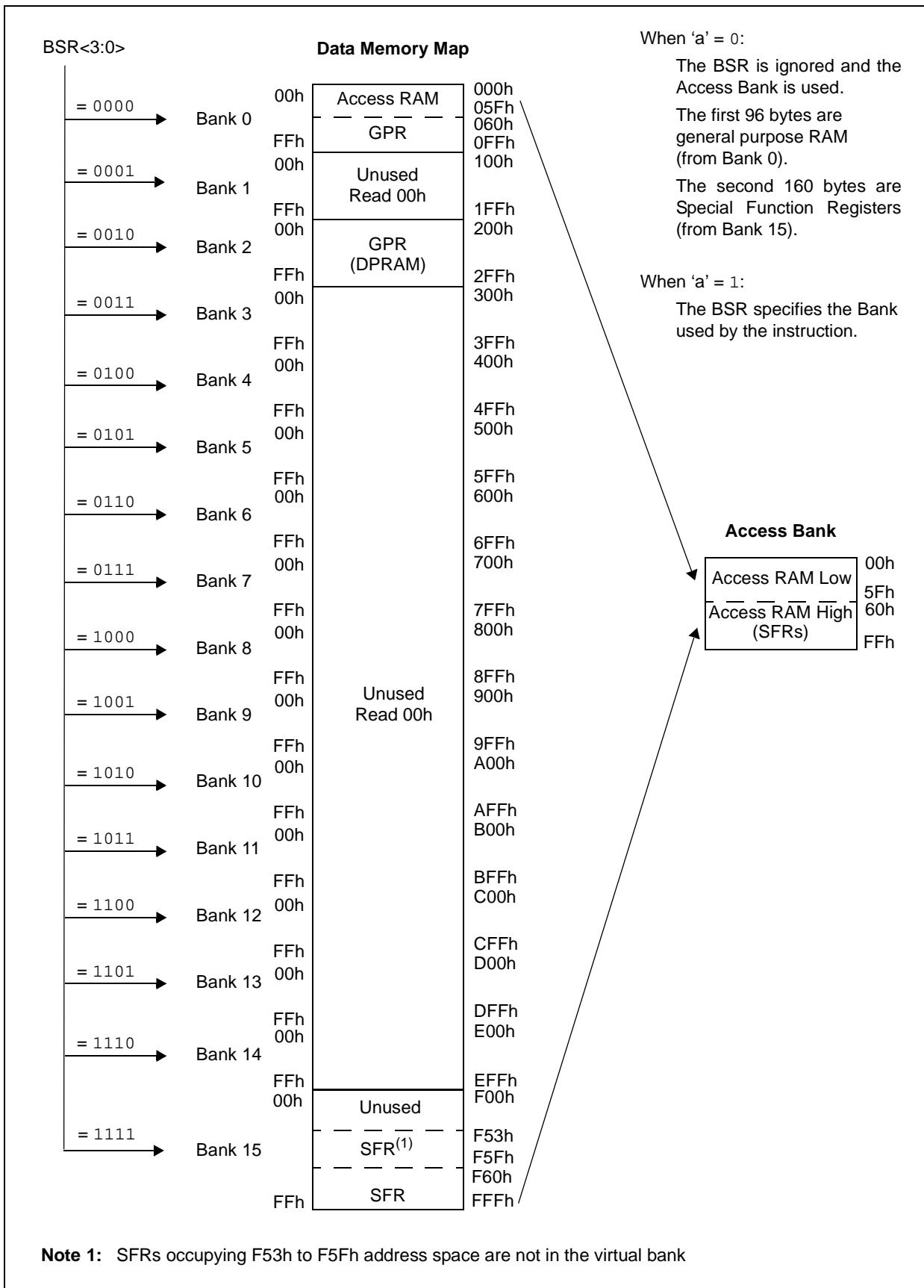
Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 0Fh will end up resetting the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

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory maps in Figure 3-5 and Figure 3-6 indicate which banks are implemented.

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

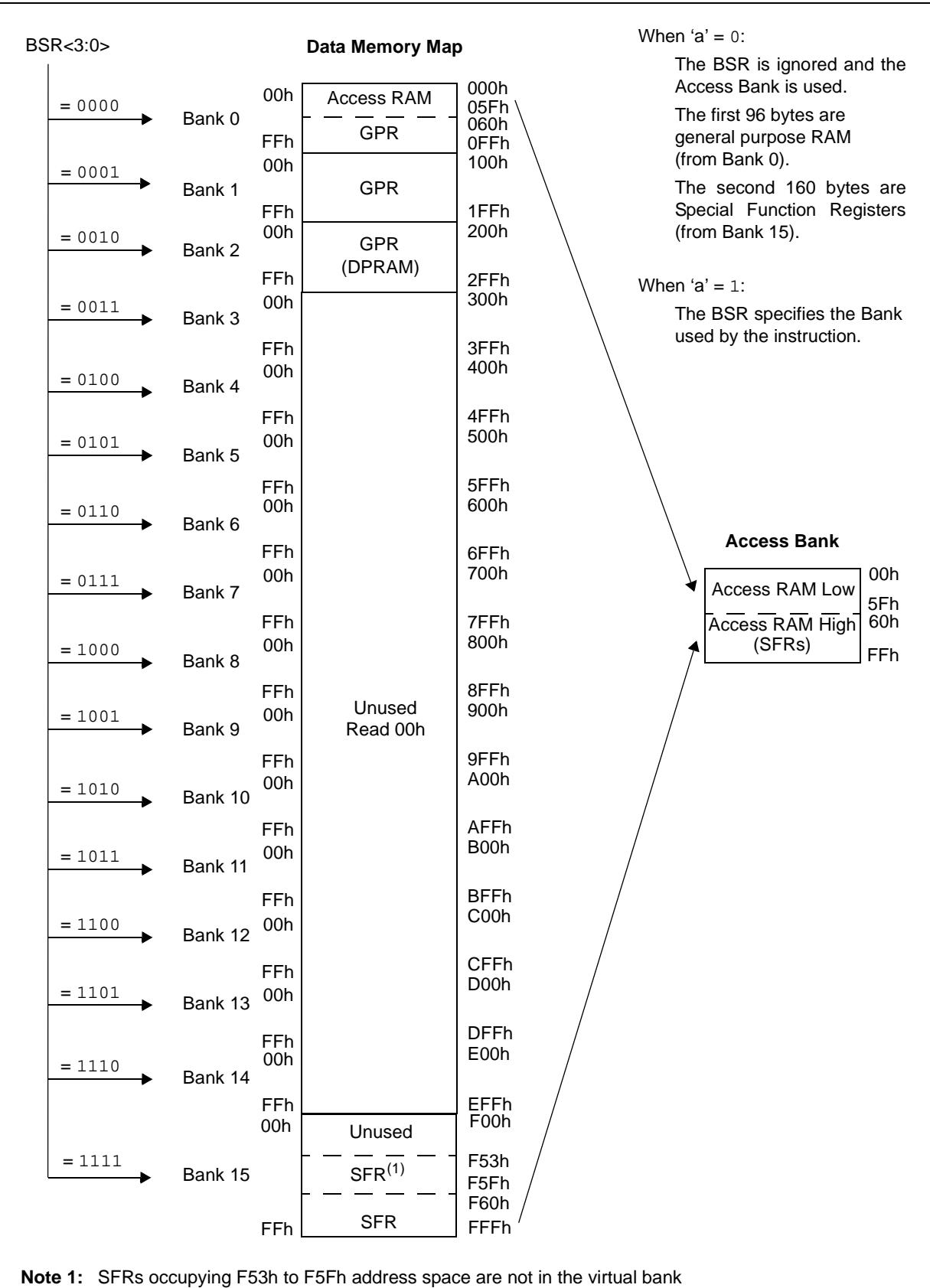
# PIC18F1XK50/PIC18LF1XK50

FIGURE 3-5: DATA MEMORY MAP FOR PIC18F13K50/PIC18LF13K50 DEVICES



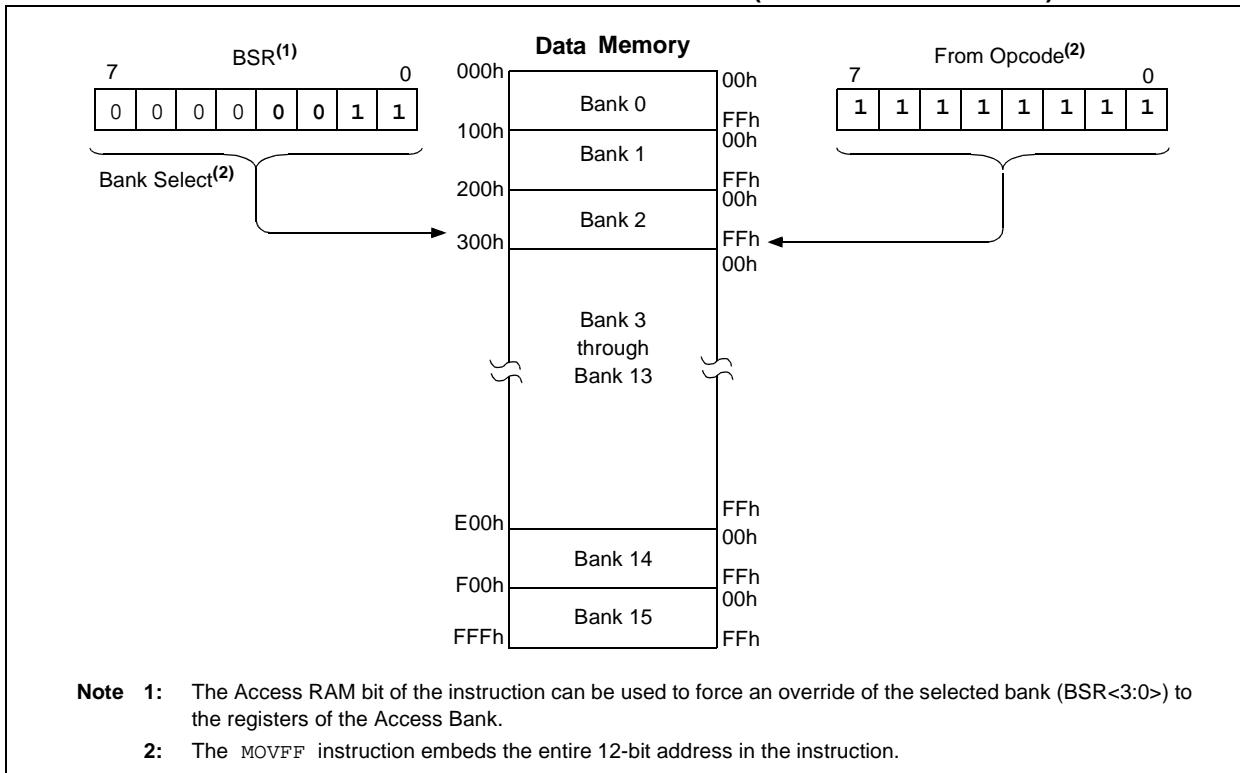
# PIC18F1XK50/PIC18LF1XK50

FIGURE 3-6: DATA MEMORY MAP FOR PIC18F14K50/PIC18LF14K50 DEVICES



# PIC18F1XK50/PIC18LF1XK50

FIGURE 3-7: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)



### 3.3.3 ACCESS BANK

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

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Block 15. The lower half is known as the “Access RAM” and is composed of GPRs. This upper half is also where the device’s SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figure 3-5 and Figure 3-6).

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

Using this “forced” addressing allows the instruction to operate on a data address in a single cycle, without updating the BSR first. For 8-bit addresses of 60h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 60h is a good place for data values that the user might need to access rapidly, such as immediate computational results or common program variables. Access RAM also allows for faster and more code efficient context saving and switching of variables.

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = 1). This is discussed in more detail in **Section 3.5.3 “Mapping the Access Bank in Indexed Literal Offset Mode”**.

### 3.3.4 GENERAL PURPOSE REGISTER FILE

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

### 3.3.5 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top portion of Bank 15 (F60h to FFFh). A list of these registers is given in Table 3-1 and Table 3-2.

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

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

# PIC18F1XK50/PIC18LF1XK50

TABLE 3-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F1XK50/PIC18LF1XK50 DEVICES

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FD7h	TMR0H	FAFh	SPBRG	F87h	__(2)	F5Fh	UEIR
FFEh	TOSH	FD6h	TMR0L	FAEh	RCREG	F86h	__(2)	F5Eh	UFRMH
FFDh	TOSL	FD5h	T0CON	FADh	TXREG	F85h	__(2)	F5Dh	UFRML
FFCh	STKPTR	FD4h	__(2)	FACh	TXSTA	F84h	__(2)	F5Ch	UADDR
FFBh	PCLATU	FD3h	OSCCON	FABh	RCSTA	F83h	__(2)	F5Bh	UEIE
FFAh	PCLATH	FD2h	OSCCON2	FAAh	EEADRH	F82h	PORTC	F5Ah	UEP7
FF9h	PCL	FD1h	WDTCON	FA9h	EEADR	F81h	PORTB	F59h	UEP6
FF8h	TBLPTRU	FD0h	RCON	FA8h	EEDATA	F80h	PORTA	F58h	UEP5
FF7h	TBLPTRH	FCFh	TMR1H	FA7h	EECON2 <sup>(1)</sup>	F7Fh	ANSELH	F57h	UEP4
FF6h	TBLPTRL	FCEh	TMR1L	FA6h	EECON1	F7Eh	ANSEL	F56h	UEP3
FF5h	TABLAT	FCDh	T1CON	FA5h	__(2)	F7Dh	__(2)	F55h	UEP2
FF4h	PRODH	FCCh	TMR2	FA4h	__(2)	F7Ch	__(2)	F54h	UEP1
FF3h	PRODL	FCBh	PR2	FA3h	__(2)	F7Bh	__(2)	F53h	UEP0
FF2h	INTCON	FCAh	T2CON	FA2h	IPR2	F7Ah	IOCB		
FF1h	INTCON2	FC9h	SSPBUF	FA1h	PIR2	F79h	IOCA		
FF0h	INTCON3	FC8h	SSPADD	FA0h	PIE2	F78h	WPUB		
FEFh	INDF0 <sup>(1)</sup>	FC7h	SSPSTAT	F9Fh	IPR1	F77h	WPUA		
FEEh	POSTINC0 <sup>(1)</sup>	FC6h	SSPCON1	F9Eh	PIR1	F76h	SLRCON		
FEDh	POSTDEC0 <sup>(1)</sup>	FC5h	SSPCON2	F9Dh	PIE1	F75h	__(2)		
FECh	PREINC0 <sup>(1)</sup>	FC4h	ADRESH	F9Ch	__(2)	F74h	__(2)		
FEBh	PLUSW0 <sup>(1)</sup>	FC3h	ADRESL	F9Bh	OSCTUNE	F73h	__(2)		
FEAh	FSR0H	FC2h	ADCON0	F9Ah	__(2)	F72h	__(2)		
FE9h	FSR0L	FC1h	ADCON1	F99h	__(2)	F71h	__(2)		
FE8h	WREG	FC0h	ADCON2	F98h	__(2)	F70h	__(2)		
FE7h	INDF1 <sup>(1)</sup>	FBFh	CCPR1H	F97h	__(2)	F6Fh	SSPMASK		
FE6h	POSTINC1 <sup>(1)</sup>	FBEh	CCPR1L	F96h	__(2)	F6Eh	__(2)		
FE5h	POSTDEC1 <sup>(1)</sup>	FBDh	CCP1CON	F95h	__(2)	F6Dh	CM1CON0		
FE4h	PREINC1 <sup>(1)</sup>	FBCh	REFCON2	F94h	TRISC	F6Ch	CM2CON1		
FE3h	PLUSW1 <sup>(1)</sup>	FBBh	REFCON1	F93h	TRISB	F6Bh	CM2CON0		
FE2h	FSR1H	FBAh	REFCON0	F92h	TRISA	F6Ah	__(2)		
FE1h	FSR1L	FB9h	PSTRCON	F91h	__(2)	F69h	SRCON1		
FE0h	BSR	FB8h	BAUDCON	F90h	__(2)	F68h	SRCON0		
FDFh	INDF2 <sup>(1)</sup>	FB7h	PWM1CON	F8Fh	__(2)	F67h	__(2)		
FDEh	POSTINC2 <sup>(1)</sup>	FB6h	ECCP1AS	F8Eh	__(2)	F66h	__(2)		
FDDh	POSTDEC2 <sup>(1)</sup>	FB5h	__(2)	F8Dh	__(2)	F65h	__(2)		
FDCh	PREINC2 <sup>(1)</sup>	FB4h	__(2)	F8Ch	__(2)	F64h	UCON		
FDBh	PLUSW2 <sup>(1)</sup>	FB3h	TMR3H	F8Bh	LATC	F63h	USTAT		
FDAh	FSR2H	FB2h	TMR3L	F8Ah	LATB	F62h	UIR		
FD9h	FSR2L	FB1h	T3CON	F89h	LATA	F61h	UCFG		
FD8h	STATUS	FB0h	SPBRGH	F88h	__(2)	F60h	UIE		

Note 1: This is not a physical register.

2: Unimplemented registers are read as '0'.

# PIC18F1XK50/PIC18LF1XK50

**TABLE 3-2: REGISTER FILE SUMMARY (PIC18F1XK50/PIC18LF1XK50)**

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	279, 26
TOSH	Top-of-Stack, High Byte (TOS<15:8>)								0000 0000	279, 26
TOSL	Top-of-Stack, Low Byte (TOS<7:0>)								0000 0000	279, 26
STKPTR	STKFUL	STKUNF	—	SP4	SP3	SP2	SP1	SP0	00-0 0000	279, 27
PCLATU	—	—	—	Holding Register for PC<20:16>					---0 0000	279, 26
PCLATH	Holding Register for PC<15:8>								0000 0000	279, 26
PCL	PC, Low Byte (PC<7:0>)								0000 0000	279, 26
TBLPTRU	—	—	—	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					---0 0000	279, 50
TBLPTRH	Program Memory Table Pointer, High Byte (TBLPTR<15:8>)								0000 0000	279, 50
TBLPTRL	Program Memory Table Pointer, Low Byte (TBLPTR<7:0>)								0000 0000	279, 50
TABLAT	Program Memory Table Latch								0000 0000	279, 50
PRODH	Product Register, High Byte								xxxx xxxx	279, 61
PRODL	Product Register, Low Byte								xxxx xxxx	279, 61
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	0000 000x	279, 65
INTCON2	RABPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	1111 -1-1	279, 66
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	279, 67
INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								N/A	279, 43
POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								N/A	279, 43
POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								N/A	279, 43
PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								N/A	279, 43
PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W								N/A	279, 43
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0, High Byte				---- 0000	279, 43
FSR0L	Indirect Data Memory Address Pointer 0, Low Byte								xxxx xxxx	279, 43
WREG	Working Register								xxxx xxxx	279
INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								N/A	279, 43
POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								N/A	279, 43
POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								N/A	279, 43
PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								N/A	279, 43
PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W								N/A	279, 43
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1, High Byte				---- 0000	280, 43
FSR1L	Indirect Data Memory Address Pointer 1, Low Byte								xxxx xxxx	280, 43
BSR	—	—	—	—	Bank Select Register				---- 0000	280, 31
INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								N/A	280, 43
POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								N/A	280, 43
POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								N/A	280, 43
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								N/A	280, 43
PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W								N/A	280, 43
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2, High Byte				---- 0000	280, 43
FSR2L	Indirect Data Memory Address Pointer 2, Low Byte								xxxx xxxx	280, 43
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	280, 41

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

**Note 1:** The SBOREN bit is only available when the BOREN<1:0> Configuration bits = 01; otherwise it is disabled and reads as '0'. See **Section 23.4 "Brown-out Reset (BOR)"**.

**2:** The RA3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0). Otherwise, RA3 reads as '0'. This bit is read-only.

**3:** Bits RA0 and RA1 are available only when USB is disabled.

# PIC18F1XK50/PIC18LF1XK50

TABLE 3-2: REGISTER FILE SUMMARY (PIC18F1XK50/PIC18LF1XK50) (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
TMR0H	Timer0 Register, High Byte								0000 0000	280, 99
TMR0L	Timer0 Register, Low Byte								xxxx xxxx	280, 99
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	280, 97
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOSF	SCS1	SCS0	0011 qq00	280, 16
OSCCON2	—	—	—	—	—	PRI_SD	HFIOFL	LFIOFS	---- -10x	280, 17
WDTCON	—	—	—	—	—	—	—	SWDTEN	--- ---0	280, 297
RCON	IPEN	SBOREN <sup>(1)</sup>	—	RI	TO	PD	POR	BOR	0q-1 11q0	271, 278, 74
TMR1H	Timer1 Register, High Byte								xxxx xxxx	280, 106
TMR1L	Timer1 Register, Low Bytes								xxxx xxxx	280, 106
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0000 0000	280, 101
TMR2	Timer2 Register								0000 0000	280, 108
PR2	Timer2 Period Register								1111 1111	280, 108
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	280, 107
SSPBUF	SSP Receive Buffer/Transmit Register								xxxx xxxx	280, 143, 144
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	280, 144
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000	280, 137, 146
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	280, 137, 146
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	280, 147
ADRESH	A/D Result Register, High Byte								xxxx xxxx	281, 217
ADRESL	A/D Result Register, Low Byte								xxxx xxxx	281, 217
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	-00 0000	281, 211
ADCON1	—	—	—	—	PVCFG1	PVCFG0	NVCFG1	NVCFG0	--- 0000	281, 212
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	281, 213
CCPR1H	Capture/Compare/PWM Register 1, High Byte								xxxx xxxx	281, 133
CCPR1L	Capture/Compare/PWM Register 1, Low Byte								xxxx xxxx	281, 133
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	281, 113
REFCON2	—	—	—	DAC1R4	DAC1R3	DAC1R2	DAC1R1	DAC1R0	---0 0000	281, 244
REFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	—	D1NSS	000- 00-0	281, 243
REFCON0	FVR1EN	FVR1ST	FVR1S1	FVR1S0	TSEN	TSRS	—	—	0001 00--	281, 243
PSTRCON	—	—	—	STRSYNC	STRD	STRC	STRB	STRA	---0 0001	281, 129
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTYP	BRG16	—	WUE	ABDEN	0100 0-00	281, 188
PWM1CON	PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0	0000 0000	281, 128
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	0000 0000	281, 125
TMR3H	Timer3 Register, High Byte								xxxx xxxx	281, 111
TMR3L	Timer3 Register, Low Byte								xxxx xxxx	281, 111
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	0-00 0000	281, 109

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

Note 1: The SBOREN bit is only available when the BOREN<1:0> Configuration bits = 01; otherwise it is disabled and reads as '0'. See Section 23.4 "Brown-out Reset (BOR)".

- 2: The RA3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0). Otherwise, RA3 reads as '0'. This bit is read-only.
- 3: Bits RA0 and RA1 are available only when USB is disabled.

# PIC18F1XK50/PIC18LF1XK50

TABLE 3-2: REGISTER FILE SUMMARY (PIC18F1XK50/PIC18LF1XK50) (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
SPBRGH	EUSART Baud Rate Generator Register, High Byte								0000 0000	281, 177
SPBRG	EUSART Baud Rate Generator Register, Low Byte								0000 0000	281, 177
RCREG	EUSART Receive Register								0000 0000	281, 178
TXREG	EUSART Transmit Register								0000 0000	281, 177
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	281, 186
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	281, 187
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	0000 0000	281, 48, 57
EEADRH	—	—	—	—	—	—	EEADR9	EEADR8	---- --00	281, 48, 57
EEDATA	EEPROM Data Register								0000 0000	281, 48, 57
EECON2	EEPROM Control Register 2 (not a physical register)								0000 0000	281, 48, 57
EECON1	EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	281, 49, 57
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	1111 111-	282, 73
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	0000 000-	282, 69
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	0000 000-	282, 71
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-111 1111	282, 72
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	282, 68
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	282, 70
OSCTUNE	INTSRC	SPLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0	0000 0000	18, 282
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111	282, 90
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	1111 ----	282, 85
TRISA	—	—	TRISA5	TRISA4	—	—	—	—	--11 ----	282, 79
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	xxxx xxxx	282, 90
LATB	LATB7	LATB6	LATB5	LATB4	—	—	—	—	xxxx ----	282, 85
LATA	—	—	LATA5	LATA4	—	—	—	—	--xx ----	282, 79
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	282, 90
PORTB	RB7	RB6	RB5	RB4	—	—	—	—	xxxx ----	282, 85
PORTA	—	—	RA5	RA4	RA3 <sup>(2)</sup>	—	RA1 <sup>(3)</sup>	RA0 <sup>(3)</sup>	--xx x-xx	282, 79
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	---- 1111	282, 95
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	1111 1---	282, 94
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—	0000 ----	282, 85
IOCA	—	—	IOCA5	IOCA4	IOCA3	—	IOCA1	IOCA0	--00 0-00	282, 79
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	—	—	—	—	1111 ----	282, 85
WPUA	—	—	WPUA5	WPUA4	WPUA3	—	—	—	--11 1---	279, 85
SLRCON	—	—	—	—	SLRC	SLRB	SLRA	—	---- -111	282, 96
SSPMASK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	1111 1111	282, 154
CM1CON0	C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH1	C1CH0	0000 1000	282, 225
CM2CON1	MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC	0000 0000	282, 226
CM2CON0	C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH1	C2CH0	0000 1000	282, 226
SRCON1	SRSPE	SRSCKE	SRSC2E	SRSC1E	SRRPE	SRRCKE	SRR2E	SRR1E	0000 0000	282, 239
SRCON0	SRLEN	SRCLK2	SRCLK1	SRCLK0	SRQEN	SRNQEN	SRPS	SRPR	0000 0000	282, 238

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

**Note 1:** The SBOREN bit is only available when the BOREN<1:0> Configuration bits = 01; otherwise it is disabled and reads as '0'. See Section 23.4 "Brown-out Reset (BOR)".

**2:** The RA3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0). Otherwise, RA3 reads as '0'. This bit is read-only.

**3:** Bits RA0 and RA1 are available only when USB is disabled.

# PIC18F1XK50/PIC18LF1XK50

TABLE 3-2: REGISTER FILE SUMMARY (PIC18F1XK50/PIC18LF1XK50) (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:
UCON	—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—	-0x0 000-	282, 246
USTAT	—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI	—	-xxx xxx-	283, 250
UIR	—	SOFIF	STALLIF	IDLEIF	TRNIF	ACTVIF	UERRIF	URSTIF	-000 0000	283, 260
UCFG	UTEYE	—	—	UPUEN	—	FSEN	PPB1	PPB0	0--0 -000	283, 248
UIE	—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE	-000 0000	283, 262
UEIR	BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF	0--0 0000	283, 263
UFRMH	—	—	—	—	—	FRM10	FRM9	FRM8	--- -xxx	283, 246
UFRML	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0	xxxx xxxx	283, 246
UADDR	—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	-000 0000	283, 252
UEIE	BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE	0--0 0000	283, 264
UEP7	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	283, 251
UEP6	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	283, 251
UEP5	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	283, 251
UEP4	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	283, 251
UEP3	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	283, 251
UEP2	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	283, 251
UEP1	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	283, 251
UEP0	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	279, 251

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

Note 1: The SBOREN bit is only available when the BOREN<1:0> Configuration bits = 01; otherwise it is disabled and reads as '0'. See Section 23.4 "Brown-out Reset (BOR)".

- 2: The RA3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0). Otherwise, RA3 reads as '0'. This bit is read-only.
- 3: Bits RA0 and RA1 are available only when USB is disabled.

### 3.3.6 STATUS REGISTER

The STATUS register, shown in Register 3-2, contains the arithmetic status of the ALU. As with any other SFR, it can be the operand for any instruction.

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

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

For other instructions that do not affect Status bits, see the instruction set summaries in Table 25-2 and Table 25-3.

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

### REGISTER 3-2: STATUS: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC <sup>(1)</sup>	C <sup>(1)</sup>
bit 7							
bit 0							

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5

**Unimplemented:** Read as '0'

bit 4

**N:** Negative bit

This bit is used for signed arithmetic (two's complement). It indicates whether the result was negative (ALU MSB = 1).

1 = Result was negative

0 = Result was positive

bit 3

**OV:** Overflow bit

This bit is used for signed arithmetic (two's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7 of the result) to change state.

1 = Overflow occurred for signed arithmetic (in this arithmetic operation)

0 = No overflow occurred

bit 2

**Z:** Zero bit

1 = The result of an arithmetic or logic operation is zero

0 = The result of an arithmetic or logic operation is not zero

bit 1

**DC:** Digit Carry/Borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)<sup>(1)</sup>

1 = A carry-out from the 4th low-order bit of the result occurred

0 = No carry-out from the 4th low-order bit of the result

bit 0

**C:** Carry/Borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)<sup>(1)</sup>

1 = A carry-out from the Most Significant bit of the result occurred

0 = No carry-out from the Most Significant bit of the result occurred

**Note 1:** For Borrow, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high-order or low-order bit of the source register.

## 3.4 Data Addressing Modes

**Note:** The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 3.5 “Data Memory and the Extended Instruction Set”** for more information.

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

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in **Section 3.5.1 “Indexed Addressing with Literal Offset”**.

### 3.4.1 INHERENT AND LITERAL ADDRESSING

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

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

### 3.4.2 DIRECT ADDRESSING

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

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of direct addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 3.3.4 “General Purpose Register File”**) or a location in the Access Bank (**Section 3.3.3 “Access Bank”**) as the data source for the instruction.

The Access RAM bit ‘a’ determines how the address is interpreted. When ‘a’ is ‘1’, the contents of the BSR (**Section 3.3.2 “Bank Select Register (BSR)”**) are used with the address to determine the complete 12-bit address of the register. When ‘a’ is ‘0’, the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

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

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

### 3.4.3 INDIRECT ADDRESSING

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

The registers for indirect addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the example of clearing an entire RAM bank in Example 3-5.

### EXAMPLE 3-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

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

### 3.4.3.1 FSR Registers and the INDF Operand

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

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

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

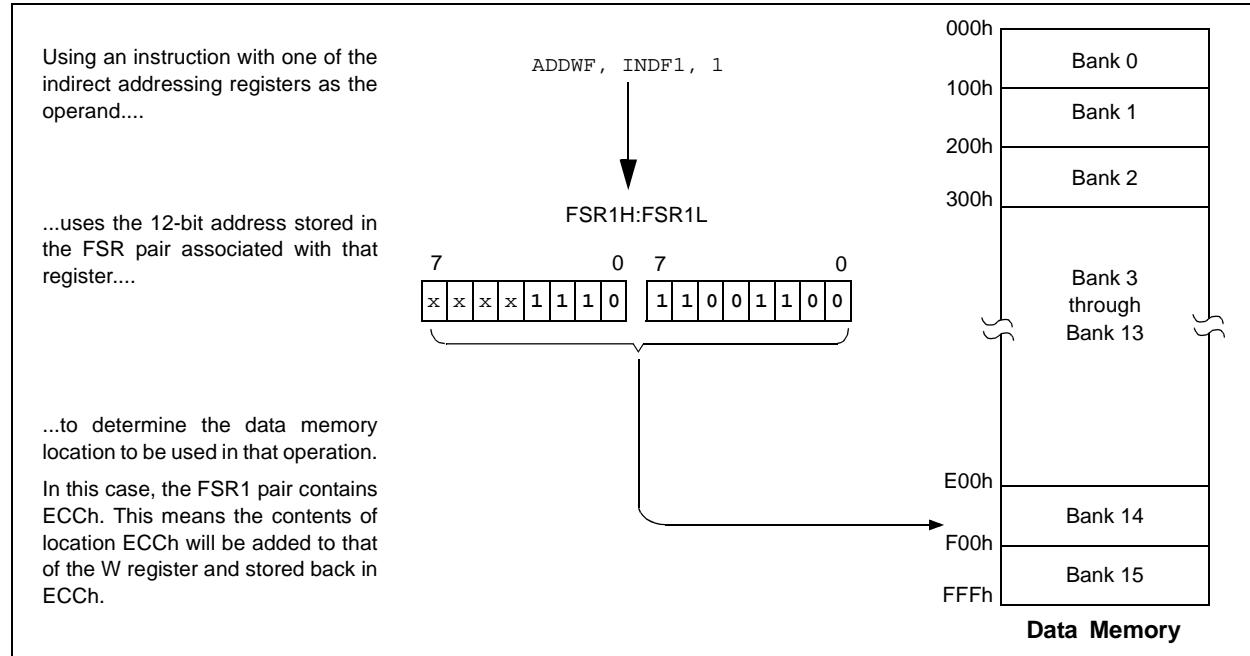
### 3.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

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

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

In this context, accessing an INDF register uses the value in the associated FSR register without changing it. Similarly, accessing a PLUSW register gives the FSR value an offset by that in the W register; however, neither W nor the FSR is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR register.

**FIGURE 3-8: INDIRECT ADDRESSING**



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

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

### 3.4.3.3 Operations by FSRs on FSRs

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

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

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

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

## 3.5 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different; this is due to the introduction of a new addressing mode for the data memory space.

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

### 3.5.1 INDEXED ADDRESSING WITH LITERAL OFFSET

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

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

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

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

### 3.5.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

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

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is '1'), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in Figure 3-9.

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 25.2.1 “Extended Instruction Syntax”**.

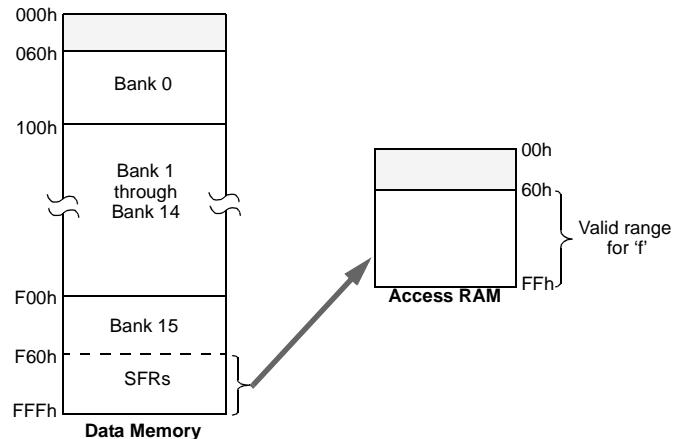
**FIGURE 3-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

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

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

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

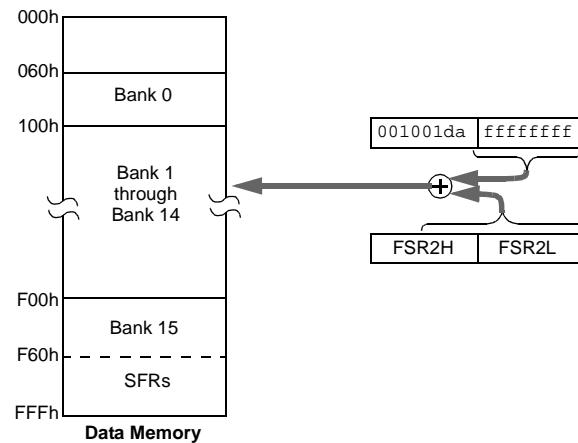
Locations below 60h are not available in this addressing mode.



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

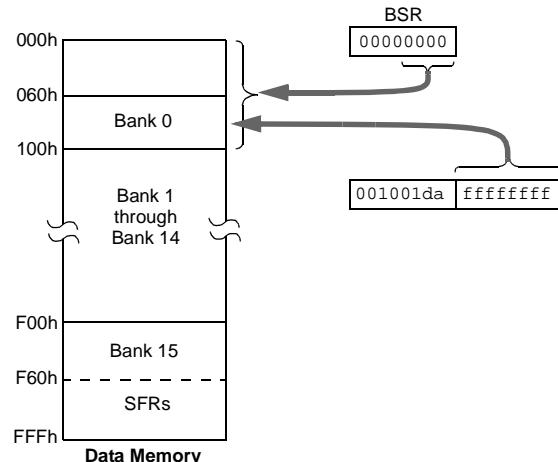
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

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



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

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



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

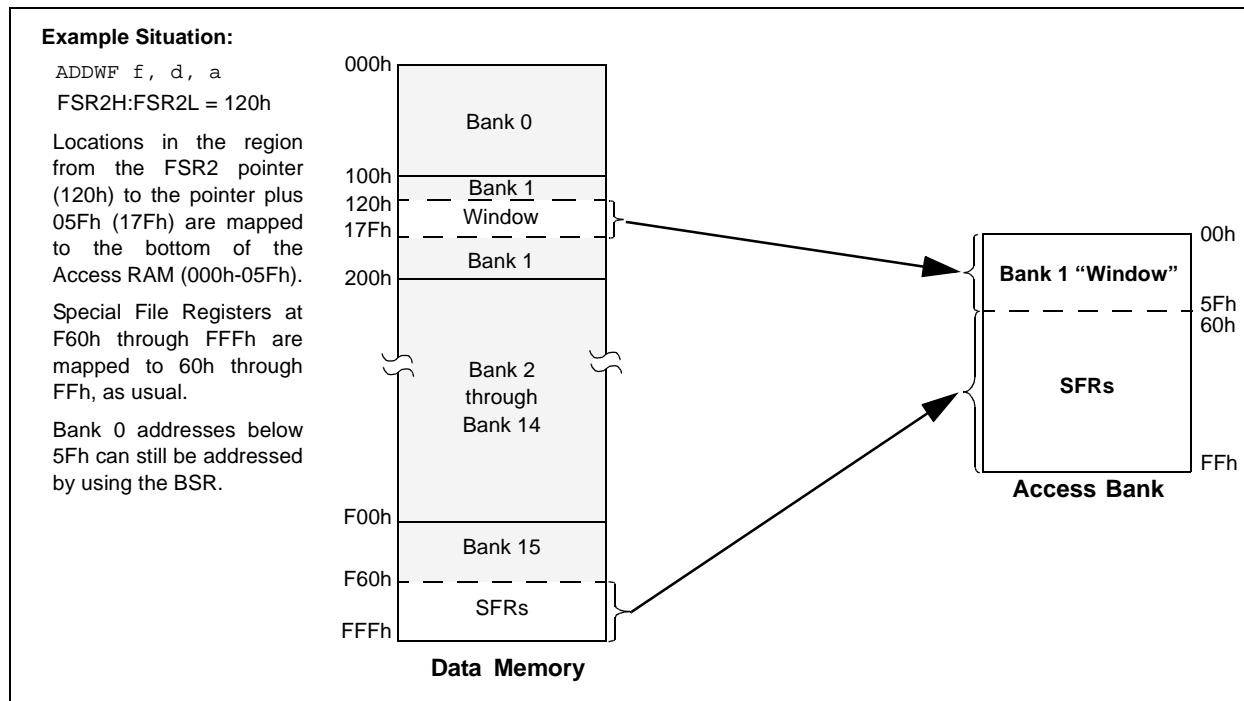
The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the bottom section of Bank 0, this mode maps the contents from a user defined “window” that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see **Section 3.3.3 “Access Bank”**). An example of Access Bank remapping in this addressing mode is shown in Figure 3-10.

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

### 3.6 PIC18 Instruction Execution and the Extended Instruction Set

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

**FIGURE 3-10: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING**



## 4.0 FLASH PROGRAM MEMORY

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

A read from program memory is executed one byte at a time. A write to program memory is executed on blocks of 32 or 8 bytes at a time depending on the specific device (See Table 4-1). Program memory is erased in blocks of 64 bytes at a time. The difference between the write and erase block sizes requires from 1 to 8 block writes to restore the contents of a single block erase. A bulk erase operation can not be issued from user code.

**TABLE 4-1: WRITE/ERASE BLOCK SIZES**

Device	Write Block Size (bytes)	Erase Block Size (bytes)
PIC18F13K50	8	64
PIC18F14K50	16	64

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.

## 4.1 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)

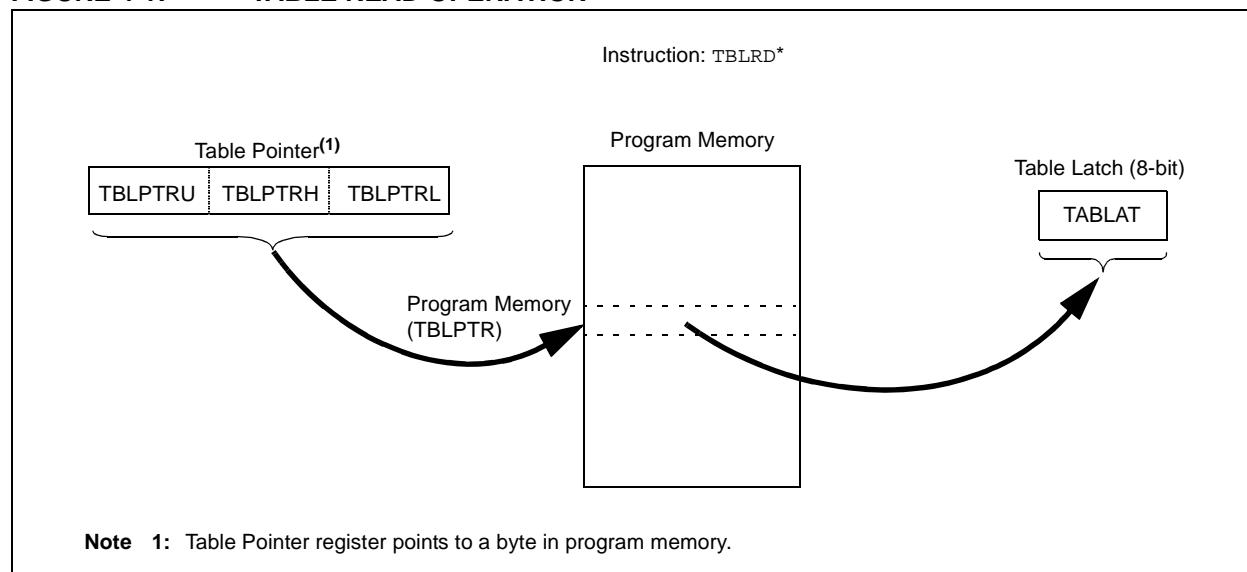
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

The table read operation retrieves one byte of data directly from program memory and places it into the TABLAT register. Figure 4-1 shows the operation of a table read.

The table write operation stores one byte of data from the TABLAT register into a write block holding register. The procedure to write the contents of the holding registers into program memory is detailed in **Section 4.5 “Writing to Flash Program Memory”**. Figure 4-2 shows the operation of a table write with program memory and data RAM.

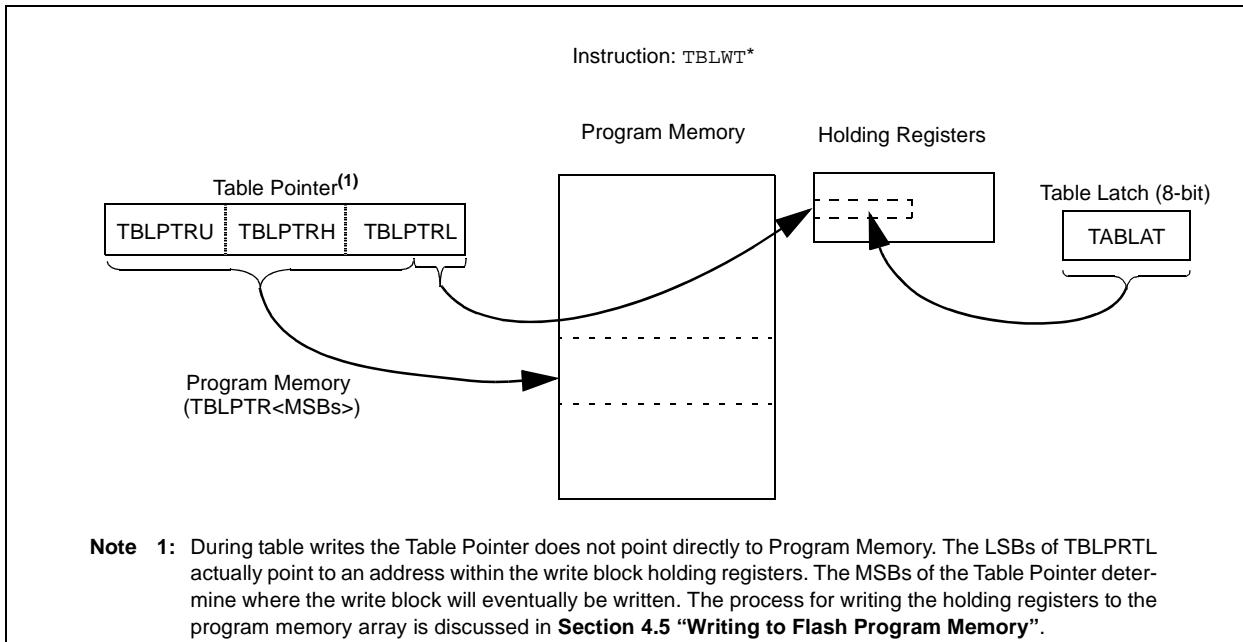
Table operations work with byte entities. Tables containing data, rather than program instructions, are not required to be word aligned. Therefore, a table can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need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

**FIGURE 4-1: TABLE READ OPERATION**



# PIC18F1XK50/PIC18LF1XK50

FIGURE 4-2: TABLE WRITE OPERATION



## 4.2 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

### 4.2.1 EECON1 AND EECON2 REGISTERS

The EECON1 register (Register 4-1) is the control register for memory accesses. The EECON2 register is not a physical register; it is used exclusively in the memory write and erase sequences. Reading EECON2 will read all '0's.

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

The CFGS control bit determines if the access will be to the Configuration/Calibration registers or to program memory/data EEPROM memory. When CFGS is set, subsequent operations will operate on Configuration registers regardless of EEPGD (see **Section 24.0 “Special Features of the CPU”**). When CFGS is clear, memory selection access is determined by EEPGD.

The FREE bit allows the program memory erase operation. When FREE is set, an 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. The WREN bit is clear on power-up.

The WRERR bit is set by hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

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

The WR control bit initiates write operations. The WR bit cannot be cleared, only set, by firmware. Then WR bit is cleared by hardware at the completion of the write operation.

**Note:** The EEIF interrupt flag bit of the PIR2 register is set when the write is complete. The EEIF flag stays set until cleared by firmware.

## REGISTER 4-1: EECON1: DATA EEPROM CONTROL 1 REGISTER

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

S = Bit can be set by software, but not cleared

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

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

# PIC18F1XK50/PIC18LF1XK50

## 4.2.2 TABLAT – TABLE LATCH REGISTER

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

## 4.2.3 TBLPTR – TABLE POINTER REGISTER

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

The Table Pointer register, TBLPTR, is used by the TBLRD and TBLWT instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations are shown in Table 4-2. These operations on the TBLPTR affect only the low-order 21 bits.

## 4.2.4 TABLE POINTER BOUNDARIES

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

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

When a TBLWT is executed the byte in the TABLAT register is written, not to Flash memory but, to a holding register in preparation for a program memory write. The holding registers constitute a write block which varies depending on the device (See Table 4-1). The 3, 4, or 5 LSbs of the TBLPTRL register determine which specific address within the holding register block is written to. The MSBs of the Table Pointer have no effect during TBLWT operations.

When a program memory write is executed the entire holding register block is written to the Flash memory at the address determined by the MSbs of the TBLPTR. The 3, 4, or 5 LSbs are ignored during Flash memory writes. For more detail, see **Section 4.5 “Writing to Flash Program Memory”**.

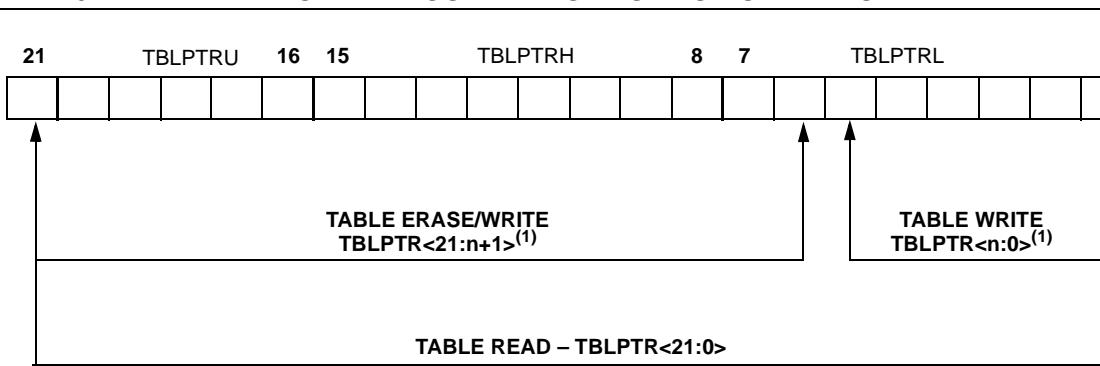
When an erase of program memory is executed, the 16 MSbs of the Table Pointer register (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

Figure 4-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

**TABLE 4-2: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS**

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

**FIGURE 4-3: TABLE POINTER BOUNDARIES BASED ON OPERATION**



**Note 1:** n = 3, 4, 5, or 6 for block sizes of 8, 16, 32 or 64 bytes, respectively.

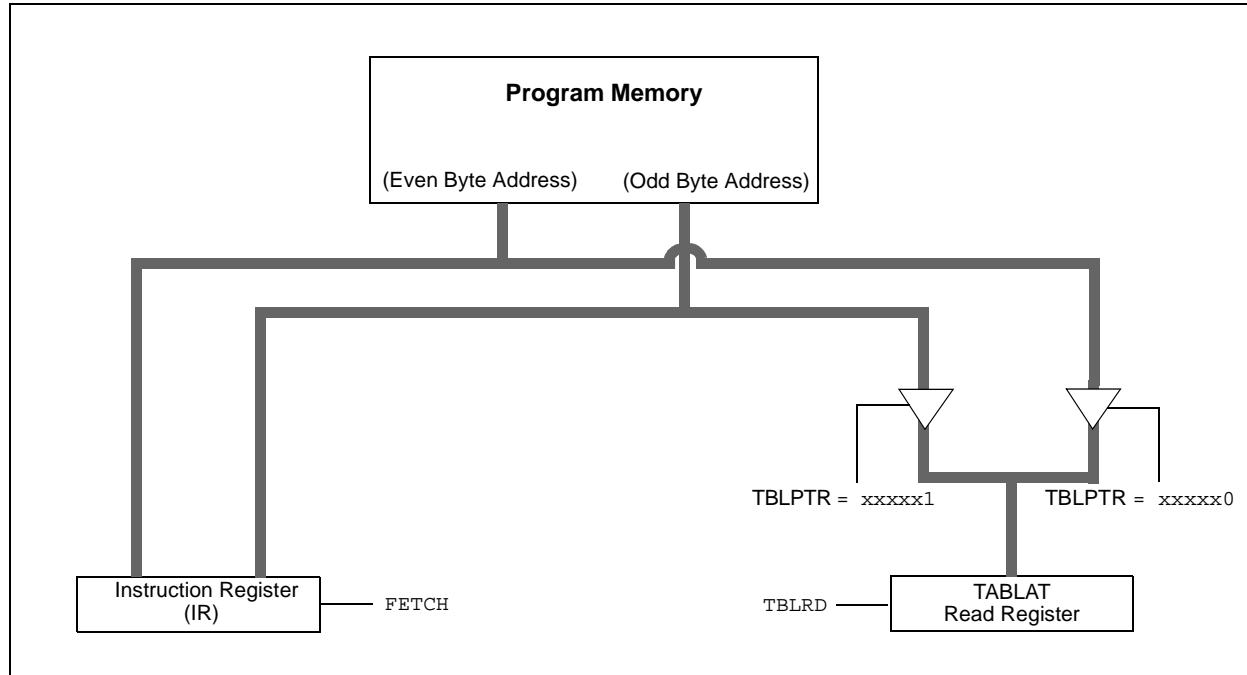
## 4.3 Reading the Flash Program Memory

The TBLRD instruction retrieves data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

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

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

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



## EXAMPLE 4-1: 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*+
    MOVfw TABLAT, W           ; read into TABLAT and increment
    MOVf  WORD_ODD             ; get data

```

# PIC18F1XK50/PIC18LF1XK50

## 4.4 Erasing Flash Program Memory

The minimum erase block is 32 words or 64 bytes. Only through the use of an external programmer, or through ICSPTM 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. The TBLPTR<5:0> bits 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.

The write initiate sequence for EECON2, shown as steps 4 through 6 in **Section 4.4.1 “Flash Program Memory Erase Sequence”**, is used to guard against accidental writes. This is sometimes referred to as a long write.

A long write is necessary for erasing the internal Flash. Instruction execution is halted during the long write cycle. The long write is terminated by the internal programming timer.

### EXAMPLE 4-2: ERASING A FLASH PROGRAM MEMORY 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	
ERASE_BLOCK	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 block Erase operation
	BCF	INTCON, GIE	; disable interrupts
Required Sequence	MOVLW	55h	
	MOVWF	EECON2	; write 55h
	MOVLW	0AAh	
	MOVWF	EECON2	; write 0AAh
	BSF	EECON1, WR	; start erase (CPU stall)
	BSF	INTCON, GIE	; re-enable interrupts

## 4.5 Writing to Flash Program Memory

The programming block size is 8 or 32 bytes, depending on the device (See Table 4-1). 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 only as many holding registers as there are bytes in a write block (See Table 4-1).

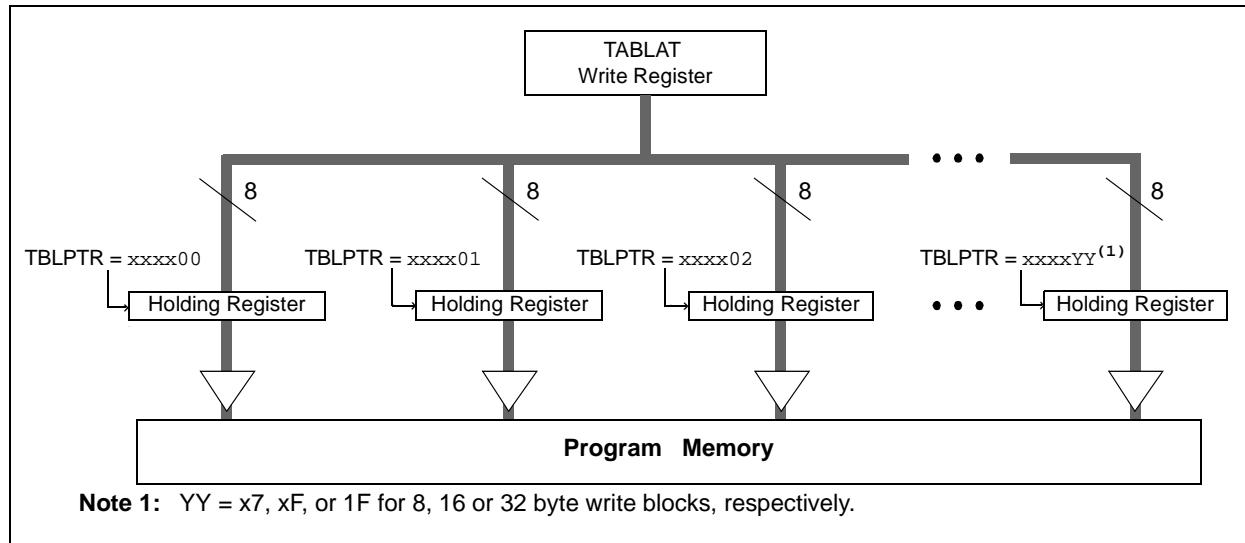
Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction may need to be executed 8, or 32 times, depending on the device, for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. After all the holding registers have been written, the programming operation of that block of memory is started by configuring the EECON1 register for a program memory write and performing the long write sequence.

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

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

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

**FIGURE 4-5: TABLE WRITES TO FLASH PROGRAM MEMORY**



### 4.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

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

1. Read 64 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load Table Pointer register with address being erased.
4. Execute the block erase procedure.
5. Load Table Pointer register with address of first byte being written.
6. Write the 8 or 32 byte block into the holding registers with auto-increment.
7. Set the EECON1 register for the write operation:
  - set EEPGD bit to point to program memory;
  - clear the CFGS bit to access program memory;
  - set WREN to enable byte writes.

8. Disable interrupts.
9. Write 55h to EECON2.
10. Write 0AAh to EECON2.
11. Set the WR bit. This will begin the write cycle.
12. The CPU will stall for duration of the write (about 2 ms using internal timer).
13. Re-enable interrupts.
14. Repeat steps 6 to 13 for each block until all 64 bytes are written.
15. Verify the memory (table read).

This procedure will require about 6 ms to update each write block of memory. An example of the required code is given in Example 4-3.

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

# PIC18F1XK50/PIC18LF1XK50

## EXAMPLE 4-3: 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  FSR0L
        MOVLW  CODE_ADDR_UPPER          ; Load TBLPTR with the base
        MOVWF  TBLPTRU
        MOVLW  CODE_ADDR_HIGH           ; address of the memory block
        MOVWF  TBLPTRH
        MOVLW  CODE_ADDR_LOW
        MOVWF  TBLPTRL

READ_BLOCK
        TBLRD*+
        MOVF   TABLAT, W
        MOVWF  POSTINC0
        DECFSZ COUNTER
        BRA    READ_BLOCK
        ; read into TABLAT, and inc
        ; get data
        ; store data
        ; done?
        ; repeat

MODIFY_WORD
        MOVLW  BUFFER_ADDR_HIGH          ; point to buffer
        MOVWF  FSROH
        MOVLW  BUFFER_ADDR_LOW
        MOVWF  FSR0L
        MOVLW  NEW_DATA_LOW             ; update buffer word
        MOVWF  POSTINC0
        MOVLW  NEW_DATA_HIGH
        MOVWF  INDF0

ERASE_BLOCK
        MOVLW  CODE_ADDR_UPPER          ; load TBLPTR with the base
        MOVWF  TBLPTRU
        MOVLW  CODE_ADDR_HIGH           ; address of the memory block
        MOVWF  TBLPTRH
        MOVLW  CODE_ADDR_LOW
        MOVWF  TBLPTRL
        BSF    EECON1, EEPGD
        BCF    EECON1, CFGS
        BSF    EECON1, WREN
        BSF    EECON1, FREE
        BCF    INTCON, GIE
        ; point to Flash program memory
        ; access Flash program memory
        ; enable write to memory
        ; enable Erase operation
        ; disable interrupts

Required Sequence
        MOVLW  55h
        MOVWF  EECON2
        ; write 55h
        MOVLW  0AAh
        MOVWF  EECON2
        ; write 0AAh
        BSF    EECON1, WR
        ; start erase (CPU stall)
        BSF    INTCON, GIE
        ; re-enable interrupts
        TBLRD*-
        MOVLW  BUFFER_ADDR_HIGH          ; dummy read decrement
        MOVWF  FSROH
        MOVLW  BUFFER_ADDR_LOW
        MOVWF  FSR0L

WRITE_BUFFER_BACK
        MOVLW  BlockSize
        MOVWF  COUNTER
        MOVLW  D'64'/BlockSize
        MOVWF  COUNTER2
        ; number of bytes in holding register
        ; number of write blocks in 64 bytes

WRITE_BYTE_TO_HREGS
        MOVF   POSTINC0, W
        MOVWF  TABLAT
        TBLWT+*
        ; get low byte of buffer data
        ; present data to table latch
        ; write data, perform a short write
        ; to internal TBLWT holding register.
```

### EXAMPLE 4-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)

PROGRAM_MEMORY	DECFSZ	COUNTER	; loop until holding registers are full	
	BRA	WRITE_WORD_TO_HREGS		
	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		
	MOVWF	EECON2	; write 55h	
	MOVLW	0AAh		
	MOVWF	EECON2		
Required Sequence	BSF	EECON1, WR	; write 0AAh	
	DCFSZ	COUNTER2	; start program (CPU stall)	
	BRA	WRITE_BYTE_TO_HREGS	; repeat for remaining write blocks	
	BSF	INTCON, GIE	;	
	BCF	EECON1, WREN	; re-enable interrupts	
; disable write to memory				

#### 4.5.2 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.

#### 4.5.3 UNEXPECTED TERMINATION OF WRITE OPERATION

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

#### 4.5.4 PROTECTION AGAINST SPURIOUS WRITES

To protect against spurious writes to Flash program memory, the write initiate sequence must also be followed. See **Section 24.0 “Special Features of the CPU”** for more detail.

#### 4.6 Flash Program Operation During Code Protection

See **Section 24.3 “Program Verification and Code Protection”** for details on code protection of Flash program memory.

**TABLE 4-3: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					279
TBPLTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								279
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								279
TABLAT	Program Memory Table Latch								279
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
EECON2	EEPROM Control Register 2 (not a physical register)								281
EECON1	EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD	281
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	282
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	282
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	282

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

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 5.0 DATA EEPROM MEMORY

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

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

- EECON1
- EECON2
- EEDATA
- EEADR
- EEADRH

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

The EEPROM data memory is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer; it will vary with voltage and temperature as well as from chip-to-chip. Please refer to parameter US122 (Table 27-13 in **Section 27.0 “Electrical Specifications”**) for exact limits.

### 5.1 EEADR and EEADRH Registers

The EEADR register is used to address the data EEPROM for read and write operations. The 8-bit range of the register can address a memory range of 256 bytes (00h to FFh). The EEADRH register expands the range to 1024 bytes by adding an additional two address bits.

### 5.2 EECON1 and EECON2 Registers

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

The EECON1 register (Register 5-1) is the control register for data and program memory access. Control bit EEPGD determines if the access will be to program or data EEPROM memory. When the EEPGD bit is clear, operations will access the data EEPROM memory. When the EEPGD bit is set, program memory is accessed.

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

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear.

The WRERR bit is set by hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

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

The WR control bit initiates write operations. The bit can be set but not cleared by software. It is cleared only by hardware at the completion of the write operation.

**Note:** The EEIF interrupt flag bit of the PIR2 register is set when the write is complete. It must be cleared by software.

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

The RD bit cannot be set when accessing program memory (EEPGD = 1). Program memory is read using table read instructions. See **Section 4.1 “Table Reads and Table Writes”** regarding table reads.

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

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 5-1: EECON1: DATA EEPROM CONTROL 1 REGISTER

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

S = Bit can be set by software, but not cleared

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

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

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

The basic process is shown in Example 5-1.

## 5.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADR register and the data written to the EEDATA register. The sequence in Example 5-2 must be followed to initiate the write cycle.

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

### EXAMPLE 5-1: DATA EEPROM READ

```

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

```

### EXAMPLE 5-2: DATA EEPROM WRITE

	MOVLW  DATA_EE_ADDR_LOW ;
	MOVWF  EEADR              ; Data Memory Address to write
	MOVLW  DATA_EE_ADDR_HI   ;
	MOVWF  EEADRH             ;
	MOVLW  DATA_EE_DATA      ;
	MOVWF  EEDATA              ; Data Memory Value to write
	BCF    EECON1, EEPGD       ; Point to DATA memory
	BCF    EECON1, CFGS        ; Access EEPROM
	BSF    EECON1, WREN        ; Enable writes
	BCF    INTCON, GIE         ; Disable Interrupts
	MOVLW  55h                ;
<b>Required Sequence</b>	MOVWF  EECON2              ; Write 55h
	MOVLW  0AAh                ;
	MOVWF  EECON2              ; Write 0AAh
	BSF    EECON1, WR          ; Set WR bit to begin write
	BSF    INTCON, GIE         ; Enable Interrupts
	; User code execution
	BCF    EECON1, WREN        ; Disable writes on write complete (EEIF set)

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

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

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

## 5.5 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.

# PIC18F1XK50/PIC18LF1XK50

## 5.6 Operation During Code-Protect

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

The microcontroller itself can both read and write to the internal data EEPROM, regardless of the state of the code-protect Configuration bit. Refer to **Section 24.0 “Special Features of the CPU”** for additional information.

## 5.7 Protection Against Spurious Write

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

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

## 5.8 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). When variables in one section change frequently, while variables in another section do not change, it is possible to exceed the total number of write cycles to the EEPROM without exceeding the total number of write cycles to a single byte. If this is the case, then 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.

### EXAMPLE 5-3: DATA EEPROM REFRESH ROUTINE

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

TABLE 5-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page	
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMROIF	INT0IF	RABIF	279	
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	281	
EEADRH	—	—	—	—	—	—	EEADR9	EEADR8	281	
EEDATA	EEPROM Data Register									281
EECON2	EEPROM Control Register 2 (not a physical register)									281
EECON1	EEPGD	CFGs	—	FREE	WRERR	WREN	WR	RD	281	
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	282	
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	282	
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	282	

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

## 6.0 8 x 8 HARDWARE MULTIPLIER

### 6.1 Introduction

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

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 6-1.

### 6.2 Operation

Example 6-1 shows the instruction sequence for an 8 x 8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 6-2 shows the sequence to do an 8 x 8 signed multiplication. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

**TABLE 6-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS**

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 40 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	6.9 $\mu$ s	27.6 $\mu$ s	69 $\mu$ s
	Hardware multiply	1	1	100 ns	400 ns	1 $\mu$ s
8 x 8 signed	Without hardware multiply	33	91	9.1 $\mu$ s	36.4 $\mu$ s	91 $\mu$ s
	Hardware multiply	6	6	600 ns	2.4 $\mu$ s	6 $\mu$ s
16 x 16 unsigned	Without hardware multiply	21	242	24.2 $\mu$ s	96.8 $\mu$ s	242 $\mu$ s
	Hardware multiply	28	28	2.8 $\mu$ s	11.2 $\mu$ s	28 $\mu$ s
16 x 16 signed	Without hardware multiply	52	254	25.4 $\mu$ s	102.6 $\mu$ s	254 $\mu$ s
	Hardware multiply	35	40	4.0 $\mu$ s	16.0 $\mu$ s	40 $\mu$ s

### EXAMPLE 6-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

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

### EXAMPLE 6-2: 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
```

# PIC18F1XK50/PIC18LF1XK50

Example 6-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 6-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES<3:0>).

## EQUATION 6-1: 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 6-3: 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 6-4 shows the sequence to do a 16 x 16 signed multiply. Equation 6-2 shows the algorithm used. The 32-bit result is stored in four registers (RES<3:0>). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

## EQUATION 6-2: 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 6-4: 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       ;
;
BTFS  ARG2H, 7       ; ARG2H:ARG2L neg?
BRA   SIGN_ARG1      ; no, check ARG1
MOVF  ARG1L, W       ;
SUBWF RES2           ;
MOVF  ARG1H, W       ;
SUBWFB RES3          ;
;
SIGN_ARG1
BTFS  ARG1H, 7       ; ARG1H:ARG1L neg?
BRA   CONT_CODE      ; no, done
MOVF  ARG2L, W       ;
SUBWF RES2           ;
MOVF  ARG2H, W       ;
SUBWFB RES3          ;
;
CONT_CODE
:

```

## 7.0 INTERRUPTS

The PIC18F1XK50/PIC18LF1XK50 devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 0008h and the low priority interrupt vector is at 0018h. A high priority interrupt event will interrupt a low priority interrupt that may be in progress.

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

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

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

In general, interrupt sources have three bits to control their operation. They are:

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

## 7.1 Mid-Range Compatibility

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® microcontroller mid-range devices. In Compatibility mode, the interrupt priority bits of the IPRx registers have no effect. The PEIE bit of the INTCON register is the global interrupt enable for the peripherals. The PEIE bit disables only the peripheral interrupt sources and enables the peripheral interrupt sources when the GIE bit is also set. The GIE bit of the INTCON register is the global interrupt enable which enables all non-peripheral interrupt sources and disables all interrupt sources, including the peripherals. All interrupts branch to address 0008h in Compatibility mode.

## 7.2 Interrupt Priority

The interrupt priority feature is enabled by setting the IPEN bit of the RCON register. When interrupt priority is enabled the GIE and PEIE global interrupt enable bits of Compatibility mode are replaced by the GIEH high priority, and GIEL low priority, global interrupt enables. When set, the GIEH bit of the INTCON register enables all interrupts that have their associated IPRx register or INTCONx register priority bit set (high priority). When clear, the GIEL bit disables all interrupt sources including those selected as low priority. When clear, the GIEL bit of the INTCON register disables only the interrupts that have their associated priority bit cleared (low priority). When set, the GIEL bit enables the low priority sources when the GIEH bit is also set.

When the interrupt flag, enable bit and appropriate global interrupt enable bit are all set, the interrupt will vector immediately to address 0008h for high priority, or 0018h for low priority, depending on level of the interrupting source's priority bit. Individual interrupts can be disabled through their corresponding interrupt enable bits.

## 7.3 Interrupt Response

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. The GIE bit is the global interrupt enable when the IPEN bit is cleared. When the IPEN bit is set, enabling interrupt priority levels, the GIEH bit is the high priority global interrupt enable and the GIEL bit is the low priority global interrupt enable. High priority interrupt sources can interrupt a low priority interrupt. Low priority interrupts are not processed while high priority interrupts are in progress.

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

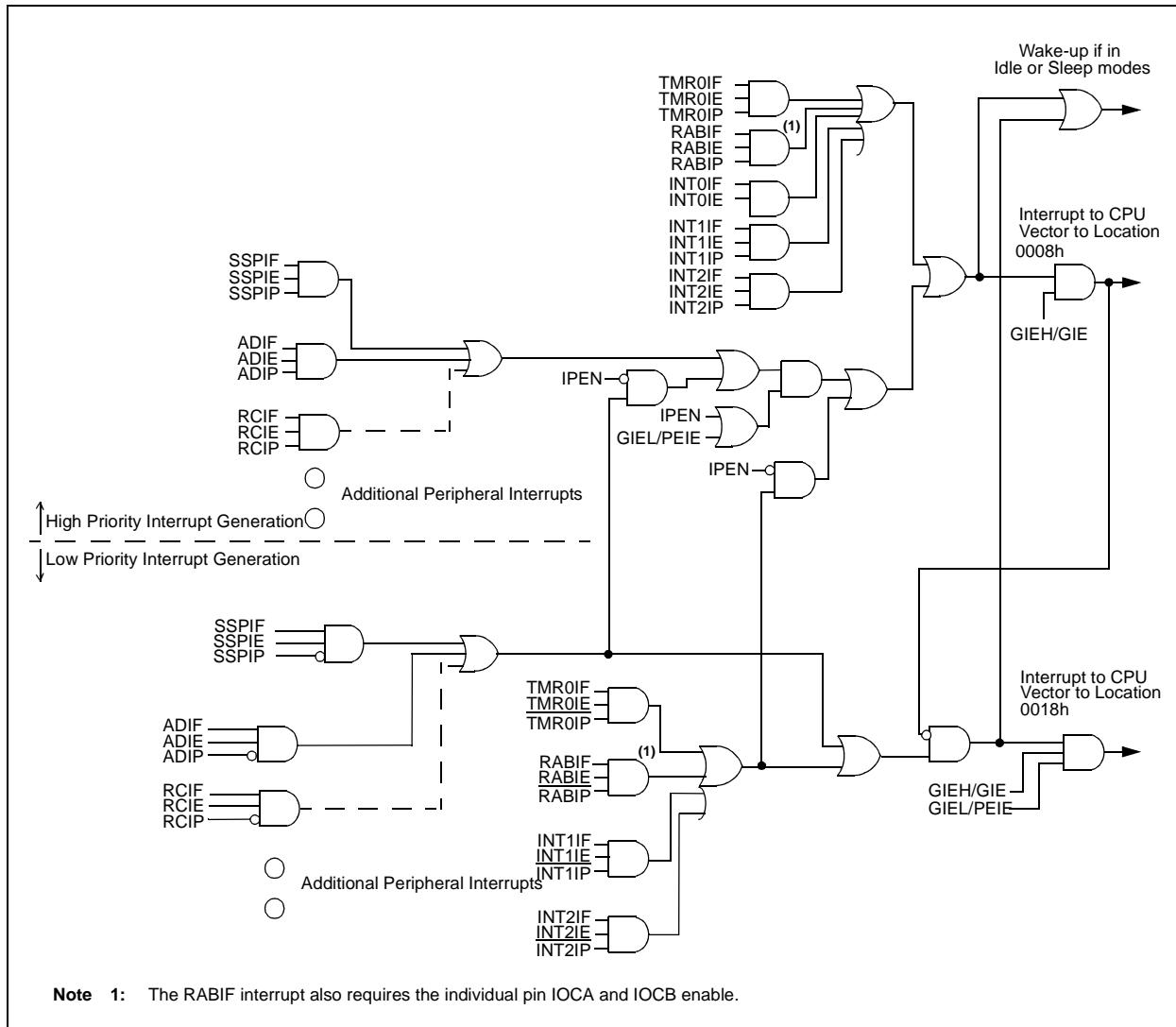
The "return from interrupt" instruction, RETFIE, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

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

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

# PIC18F1XK50/PIC18LF1XK50

FIGURE 7-1: PIC18 INTERRUPT LOGIC



## 7.4 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 7-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF
bit 7							bit 0

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>GIE/GIEH:</b> Global Interrupt Enable bit <u>When IPEN = 0:</u> 1 = Enables all unmasked interrupts 0 = Disables all interrupts including peripherals <u>When IPEN = 1:</u> 1 = Enables all high priority interrupts 0 = Disables all interrupts including low priority.
bit 6	<b>PEIE/GIEL:</b> Peripheral Interrupt Enable bit <u>When IPEN = 0:</u> 1 = Enables all unmasked peripheral interrupts 0 = Disables all peripheral interrupts <u>When IPEN = 1:</u> 1 = Enables all low priority interrupts 0 = Disables all low priority interrupts
bit 5	<b>TMROIE:</b> TMR0 Overflow Interrupt Enable bit 1 = Enables the TMR0 overflow interrupt 0 = Disables the TMR0 overflow interrupt
bit 4	<b>INT0IE:</b> INT0 External Interrupt Enable bit 1 = Enables the INT0 external interrupt 0 = Disables the INT0 external interrupt
bit 3	<b>RABIE:</b> RA and RB Port Change Interrupt Enable bit <sup>(2)</sup> 1 = Enables the RA and RB port change interrupt 0 = Disables the RA and RB port change interrupt
bit 2	<b>TMR0IF:</b> TMR0 Overflow Interrupt Flag bit 1 = TMR0 register has overflowed (must be cleared by software) 0 = TMR0 register did not overflow
bit 1	<b>INT0IF:</b> INT0 External Interrupt Flag bit 1 = The INT0 external interrupt occurred (must be cleared by software) 0 = The INT0 external interrupt did not occur
bit 0	<b>RABIF:</b> RA and RB Port Change Interrupt Flag bit <sup>(1)</sup> 1 = At least one of the RA <5:3> or RB<7:4> pins changed state (must be cleared by software) 0 = None of the RA<5:3> or RB<7:4> pins have changed state

- Note 1:** A mismatch condition will continue to set the RABIF bit. Reading PORTA and PORTB will end the mismatch condition and allow the bit to be cleared.
- 2:** RA and RB port change interrupts also require the individual pin IOCA and IOCB enable.

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 7-2: INTCON2: INTERRUPT CONTROL 2 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
RABPU	INTEDG0	INTEDG1	INTEDG2	—	TMROIP	—	RABIP
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>RABPU:</b> PORTA and PORTB Pull-up Enable bit 1 = All PORTA and PORTB pull-ups are disabled 0 = PORTA and PORTB pull-ups are enabled provided that the pin is an input and the corresponding WPUA and WPUB bits are set.
bit 6	<b>INTEDG0:</b> External Interrupt 0 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 5	<b>INTEDG1:</b> External Interrupt 1 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 4	<b>INTEDG2:</b> External Interrupt 2 Edge Select bit 1 = Interrupt on rising edge 0 = Interrupt on falling edge
bit 3	<b>Unimplemented:</b> Read as '0'
bit 2	<b>TMROIP:</b> TMR0 Overflow Interrupt Priority bit 1 = High priority 0 = Low priority
bit 1	<b>Unimplemented:</b> Read as '0'
bit 0	<b>RABIP:</b> RA and RB Port Change Interrupt Priority bit 1 = High priority 0 = Low priority

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

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 7-3: INTCON3: INTERRUPT CONTROL 3 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

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

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

# PIC18F1XK50/PIC18LF1XK50

## 7.5 PIR Registers

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

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

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

### REGISTER 7-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

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

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7

**Unimplemented:** Read as '0'

bit 6

**ADIF:** A/D Converter Interrupt Flag bit

1 = An A/D conversion completed (must be cleared by software)  
0 = The A/D conversion is not complete or has not been started

bit 5

**RCIF:** EUSART Receive Interrupt Flag bit

1 = The EUSART receive buffer, RCREG, is full (cleared when RCREG is read)  
0 = The EUSART receive buffer is empty

bit 4

**TXIF:** EUSART Transmit Interrupt Flag bit

1 = The EUSART transmit buffer, TXREG, is empty (cleared when TXREG is written)  
0 = The EUSART transmit buffer is full

bit 3

**SSPIF:** Master Synchronous Serial Port Interrupt Flag bit

1 = The transmission/reception is complete (must be cleared by software)  
0 = Waiting to transmit/receive

bit 2

**CCP1IF:** CCP1 Interrupt Flag bit

Capture mode:

1 = A TMR1 register capture occurred (must be cleared by software)  
0 = No TMR1 register capture occurred

Compare mode:

1 = A TMR1 register compare match occurred (must be cleared by 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 by software)  
0 = No TMR2 to PR2 match occurred

bit 0

**TMR1IF:** TMR1 Overflow Interrupt Flag bit

1 = TMR1 register overflowed (must be cleared by software)  
0 = TMR1 register did not overflow

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

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 7-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>OSCFIF:</b> Oscillator Fail Interrupt Flag bit 1 = Device oscillator failed, clock input has changed to HFINTOSC (must be cleared by software) 0 = Device clock operating
bit 6	<b>C1IF:</b> Comparator C1 Interrupt Flag bit 1 = Comparator C1 output has changed (must be cleared by software) 0 = Comparator C1 output has not changed
bit 5	<b>C2IF:</b> Comparator C2 Interrupt Flag bit 1 = Comparator C2 output has changed (must be cleared by software) 0 = Comparator C2 output has not changed
bit 4	<b>EEIF:</b> Data EEPROM/Flash Write Operation Interrupt Flag bit 1 = The write operation is complete (must be cleared by software) 0 = The write operation is not complete or has not been started
bit 3	<b>BCLIF:</b> Bus Collision Interrupt Flag bit 1 = A bus collision occurred (must be cleared by software) 0 = No bus collision occurred
bit 2	<b>USBIF:</b> USB Interrupt Flag bit 1 = USB has requested an interrupt (must be cleared in software) 0 = No USB interrupt request
bit 1	<b>TMR3IF:</b> TMR3 Overflow Interrupt Flag bit 1 = TMR3 register overflowed (must be cleared by software) 0 = TMR3 register did not overflow
bit 0	<b>Unimplemented:</b> Read as '0'

# PIC18F1XK50/PIC18LF1XK50

---

## 7.6 PIE Registers

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

### REGISTER 7-6: PIE1: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 1

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

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7

**Unimplemented:** Read as '0'

bit 6

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

1 = Enables the A/D interrupt

0 = Disables the A/D interrupt

bit 5

**RCIE:** EUSART Receive Interrupt Enable bit

1 = Enables the EUSART receive interrupt

0 = Disables the EUSART receive interrupt

bit 4

**TXIE:** EUSART Transmit Interrupt Enable bit

1 = Enables the EUSART transmit interrupt

0 = Disables the EUSART transmit interrupt

bit 3

**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

## REGISTER 7-7: PIE2: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>OSCFIE:</b> Oscillator Fail Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 6	<b>C1IE:</b> Comparator C1 Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 5	<b>C2IE:</b> Comparator C2 Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 4	<b>EEIE:</b> Data EEPROM/Flash Write Operation Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 3	<b>BCLIE:</b> Bus Collision Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 2	<b>USBIE:</b> USB Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 1	<b>TMR3IE:</b> TMR3 Overflow Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 0	<b>Unimplemented:</b> Read as '0'

# PIC18F1XK50/PIC18LF1XK50

---

## 7.7 IPR Registers

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

### REGISTER 7-8: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

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

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7

**Unimplemented:** Read as '0'

bit 6

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

1 = High priority

0 = Low priority

bit 5

**RCIP:** EUSART Receive Interrupt Priority bit

1 = High priority

0 = Low priority

bit 4

**TXIP:** EUSART Transmit Interrupt Priority bit

1 = High priority

0 = Low priority

bit 3

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

1 = High priority

0 = Low priority

bit 2

**CCP1IP:** CCP1 Interrupt Priority bit

1 = High priority

0 = Low priority

bit 1

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

1 = High priority

0 = Low priority

bit 0

**TMR1IP:** TMR1 Overflow Interrupt Priority bit

1 = High priority

0 = Low priority

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

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 7-9: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>OSCFIP:</b> Oscillator Fail Interrupt Priority bit 1 = High priority 0 = Low priority
bit 6	<b>C1IP:</b> Comparator C1 Interrupt Priority bit 1 = High priority 0 = Low priority
bit 5	<b>C2IP:</b> Comparator C2 Interrupt Priority bit 1 = High priority 0 = Low priority
bit 4	<b>EEIP:</b> Data EEPROM/Flash Write Operation Interrupt Priority bit 1 = High priority 0 = Low priority
bit 3	<b>BCLIP:</b> Bus Collision Interrupt Priority bit 1 = High priority 0 = Low priority
bit 2	<b>USBIP:</b> USB Interrupt Priority bit 1 = High priority 0 = Low priority
bit 1	<b>TMR3IP:</b> TMR3 Overflow Interrupt Priority bit 1 = High priority 0 = Low priority
bit 0	<b>Unimplemented:</b> Read as '0'

# PIC18F1XK50/PIC18LF1XK50

## 7.8 RCON Register

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

The operation of the SBOREN bit and the Reset flag bits is discussed in more detail in **Section 23.1 “RCON Register”**.

### REGISTER 7-10: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0	
IPEN	SBOREN <sup>(1)</sup>	—	RI	TO	PD	POR <sup>(2)</sup>	BOR	
bit 7								bit 0

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>IPEN:</b> Interrupt Priority Enable bit 1 = Enable priority levels on interrupts 0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)
bit 6	<b>SBOREN:</b> BOR Software Enable bit <sup>(1)</sup> <u>If BOREN&lt;1:0&gt; = 01:</u> 1 = BOR is enabled 0 = BOR is disabled <u>If BOREN&lt;1:0&gt; = 00, 10 or 11:</u> Bit is disabled and read as '0'.
bit 5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>RI:</b> RESET Instruction Flag bit 1 = The RESET instruction was not executed (set by firmware or Power-on Reset) 0 = The RESET instruction was executed causing a device Reset (must be set in firmware after a code-executed Reset occurs)
bit 3	<b>TO:</b> Watchdog Time-out Flag bit 1 = Set by power-up, CLRWDT instruction or SLEEP instruction 0 = A WDT time-out occurred
bit 2	<b>PD:</b> Power-down Detection Flag bit 1 = Set by power-up or by the CLRWDT instruction 0 = Set by execution of the SLEEP instruction
bit 1	<b>POR:</b> Power-on Reset Status bit <sup>(2)</sup> 1 = No Power-on Reset occurred 0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)
bit 0	<b>BOR:</b> Brown-out Reset Status bit <sup>(3)</sup> 1 = A Brown-out Reset has not occurred (set by firmware only) 0 = A Brown-out Reset occurred (must be set by firmware after a POR or Brown-out Reset occurs)

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

**2:** The actual Reset value of POR is determined by the type of device Reset. See the notes following this register and **Section 23.6 “Reset State of Registers”** for additional information.

**3:** See Table 23-3.

## 7.9 INTn Pin Interrupts

External interrupts on the RC0/INT0, RC1/INT1 and RC2/INT2 pins are edge-triggered. If the corresponding INTEDG<sub>x</sub> bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. When a valid edge appears on the RC<sub>x</sub>/INT<sub>x</sub> 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 by software in the Interrupt Service Routine before re-enabling the interrupt.

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

Interrupt priority for INT1 and INT2 is determined by the value contained in the interrupt priority bits, INT1IP and INT2IP of the INTCON3 register. There is no priority bit associated with INT0. It is always a high priority interrupt source.

## 7.10 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit, TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE of the INTCON register. Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP of the INTCON2 register. See **Section 10.0 “Timer0 Module”** for further details on the Timer0 module.

## 7.11 PORTA and PORTB Interrupt-on-Change

An input change on PORTA or PORTB sets flag bit, RABIF of the INTCON register. The interrupt can be enabled/disabled by setting/clearing enable bit, RABIE of the INTCON register. Pins must also be individually enabled with the IOCA and IOCB register. Interrupt priority for PORTA and PORTB interrupt-on-change is determined by the value contained in the interrupt priority bit, RABIP of the INTCON2 register.

## 7.12 Context Saving During Interrupts

During interrupts, the return PC address is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the fast return stack. If a fast return from interrupt is not used (see **Section 3.3 “Data Memory Organization”**), the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user's application, other registers may also need to be saved. Example 7-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

### EXAMPLE 7-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

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

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 8.0 LOW DROPOUT (LDO) VOLTAGE REGULATOR

The PIC18F1XK50 devices differ from the PIC18LF1XK50 devices due to an internal Low Dropout (LDO) voltage regulator. The PIC18F1XK50 contain an internal LDO, while the PIC18LF1XK50 do not.

The lithography of the die allows a maximum operating voltage of the nominal 3.6V on the internal digital logic. In order to continue to support 5.0V designs, a LDO voltage regulator is integrated on the die. The LDO voltage regulator allows for the internal digital logic to operate at 3.3V, while I/O's operate at 5.0V (VDD).

The LDO voltage regulator requires an external bypass capacitor for stability. The VUSB pin is required to have an external bypass capacitor. It is recommended that the capacitor be a ceramic cap between 0.22 to 0.47  $\mu$ F.

On power-up, the external capacitor will look like a large load on the LDO voltage regulator. To prevent erroneous operation, the device is held in Reset while a constant current source charges the external capacitor. After the cap is fully charged, the device is released from Reset. For more information, refer to **Section 27.0 “Electrical Specifications”**.

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 9.0 I/O PORTS

There are up to 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.

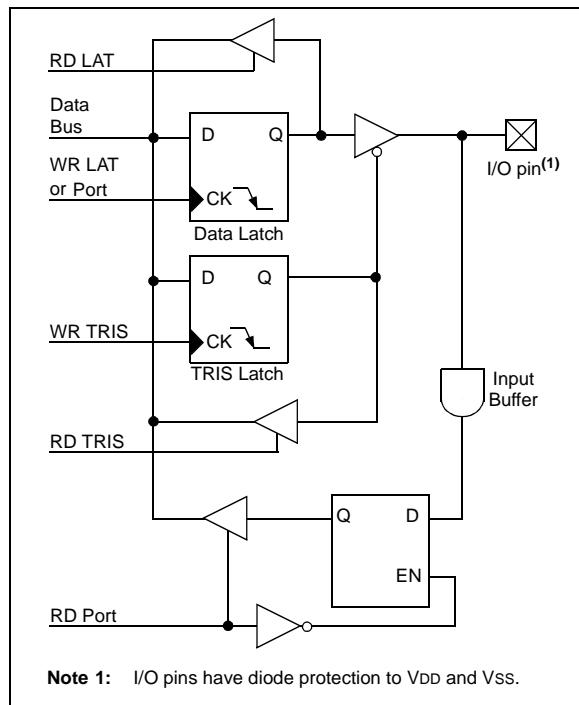
Each port has three registers for its operation. These registers are:

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

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

A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 9-1.

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



## 9.1 PORTA, TRISA and LATA Registers

PORTA is 5 bits wide. PORTA<5:4> bits are bidirectional ports and PORTA<3:1:0> bits are input-only ports. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., disable the output driver). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., enable the output driver and 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 PORTA Data Latch (LATA) register is also memory mapped. Read-modify-write operations on the LATA register read and write the latched output value for PORTA.

All of the PORTA pins are individually configurable as interrupt-on-change pins. Control bits in the IOCA register enable (when set) or disable (when clear) the interrupt function for each pin.

When set, the RABIE bit of the INTCON register enables interrupts on all pins which also have their corresponding IOCA bit set. When clear, the RABIE bit disables all interrupt-on-changes.

Only pins configured as inputs can cause this interrupt to occur (i.e., any pin configured as an output is excluded from the interrupt-on-change comparison).

For enabled interrupt-on-change pins, the values are compared with the old value latched on the last read of PORTA. The 'mismatch' outputs of the last read are OR'd together to set the PORTA Change Interrupt flag bit (RABIF) in the INTCON register.

This interrupt can wake the device from the Sleep mode, or any of the Idle modes. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

- a) Any read or write of PORTA to clear the mismatch condition (except when PORTA is the source or destination of a MOVFF instruction).
  - b) Clear the flag bit, RABIF.

A mismatch condition will continue to set the RABIF flag bit. Reading or writing PORTA will end the mismatch condition and allow the RABIF bit to be cleared. The latch holding the last read value is not affected by a MCLR nor Brown-out Reset. After either one of these Resets, the RABIF flag will continue to be set if a mismatch is present.

**Note 1:** If a change on the I/O pin should occur when the read operation is being executed (start of the Q2 cycle), then the RABIF interrupt flag may not get set. Furthermore, since a read or write on a port affects all bits of that port, care must be taken when using multiple pins in Interrupt-on-change mode. Changes on one pin may not be seen while servicing changes on another pin.

- 2: When configured for USB operation, interrupt-on-change functionality on RA0 and RA1 is automatically disabled.
  - 3: In order for the digital inputs to function on the RA<1:0> port pins, the interrupt-on-change pins must be enabled (IOCA<1:0> = 11) and the USB module must be disabled (USBEN = 0).

# PIC18F1XK50/PIC18LF1XK50

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

Each of the PORTA pins has an individually controlled weak internal pull-up. When set, each bit of the WPUA register enables the corresponding pin pull-up. When cleared, the RABPU bit of the INTCON2 register enables pull-ups on all pins which also have their corresponding WPUA bit set. When set, the RABPU bit disables all weak pull-ups. 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, RA4 is configured as analog inputs by default and read as '0'; RA<1:0> and RA<5:3> are configured as digital inputs.

RA0 and RA1 are multiplexed with the USB module and can serve as the differential data lines for the on-chip USB transceiver.

RA0 and RA1 do not have TRISA bits associated with them. As digital port pins, they can only function as digital inputs. When configured for USB operation, the data direction is determined by the configuration and status of the USB module at a given time.

RA3 is an input only pin. Its operation is controlled by the MCLRE bit of the CONFIG3H register. When selected as a port pin (MCLRE = 0), it functions as a digital input only pin; as such, it does not have TRIS or LAT bits associated with its operation.

**Note:** On a Power-on Reset, RA3 is enabled as a digital input only if Master Clear functionality is disabled.

Pins RA4 and RA5 are multiplexed with the main oscillator pins; they are enabled as oscillator or I/O pins by the selection of the main oscillator in the Configuration register (see **Section 24.1 “Configuration Bits”** for details). When they are not used as port pins, RA4 and RA5 and their associated TRIS and LAT bits read as '0'.

Pin RA4 is multiplexed with an analog input. The operation of pin RA4 as analog is selected by setting the ANS3 bit in the ANSEL register which is the default setting after a Power-on Reset.

**Note:** On a Power-on Reset, RA4 is configured as analog inputs and read as '0'.

## EXAMPLE 9-1: INITIALIZING PORTA

```
CLRF  PORTA ; Initialize PORTA by
              ; clearing output
              ; data latches
CLRF  LATA  ; Alternate method
              ; to clear output
              ; data latches
MOVLW 030h ; Value used to
              ; initialize data
              ; direction
MOVWF TRISA ; Set RA<5:4> as output
```

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 9-1: PORTA: PORTA REGISTER

U-0	U-0	R/W-x	R/W-x	R-x	U-0	R/W-x	R/W-x
—	—	RA5	RA4	RA3	—	RA1	RA0
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6	<b>Unimplemented:</b> Read as '0'
bit 5-3	<b>RA&lt;5:3&gt;:</b> PORTA I/O Pin bit <sup>(1)</sup>
	1 = Port pin is > VIH
	0 = Port pin is < VIL
bit 2	<b>Unimplemented:</b> Read as '0'
bit 1-0	<b>RA&lt;1:0&gt;:</b> PORTA I/O Pin bit
	1 = Port pin is > VIH
	0 = Port pin is < VIL

**Note 1:** The RA3 bit is only available when Master Clear Reset is disabled (MCLRE Configuration bit = 0). Otherwise, RA3 reads as '0'. This bit is read-only.

## REGISTER 9-2: TRISA: PORTA TRI-STATE REGISTER

U-0	U-0	R/W-1	R/W-1	U-0	U-0	U-0	U-0
—	—	TRISA5	TRISA4	—	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6	<b>Unimplemented:</b> Read as '0'
bit 5-4	<b>TRISA&lt;5:4&gt;:</b> PORTA Tri-State Control bit
	1 = PORTA pin configured as an input (tri-stated)
	0 = PORTA pin configured as an output

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

**Note 1:** TRISA<5:4> always reads '1' in XT, HS and LP Oscillator modes.

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 9-3: WPUA: WEAK PULL-UP PORTA REGISTER

U-0	U-0	R/W-1	R/W-1	RW-1	U-0	U-0	U-0
—	—	WPUA5	WPUA4	WPUA3	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

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

- bit 7-6      **Unimplemented:** Read as '0'  
 bit 5-3      **WPUA<5:3>:** Weak Pull-up Enable bit  
     1 = Pull-up enabled  
     0 = Pull-up disabled  
 bit 2      **Unimplemented:** Read as '0'  
 bit 1-0      **WPUA<1:0>:** Weak Pull-up Enable bit  
     1 = Pull-up enabled  
     0 = Pull-up disabled

## REGISTER 9-4: IOCA: INTERRUPT-ON-CHANGE PORTA REGISTER

U-0	U-0	R/W-0	R/W-0	R-0	U-0	R/W-0	R/W-0
—	—	IOCA5	IOCA4	IOCA3	—	IOCA1	IOCA0
bit 7							bit 0

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

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

- bit 7-6      **Unimplemented:** Read as '0'  
 bit 5-3      **IOCA<5:3>:** PORTA I/O Pin bit  
     1 = Interrupt-on-change enabled  
     0 = Interrupt-on-change disabled  
 bit 2      **Unimplemented:** Read as '0'  
 bit 1-0      **IOCA<1:0>:** PORTA I/O Pin bit  
     1 = Interrupt-on-change enabled  
     0 = Interrupt-on-change disabled

## REGISTER 9-5: LATA: PORTA DATA LATCH REGISTER

U-0	U-0	R/W-x	R/W-x	U-0	U-0	U-0	U-0
—	—	LATA5	LATA4	—	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

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

- bit 7-6      **Unimplemented:** Read as '0'  
 bit 5-4      **LATA<5:4>:** RA<5:4> Port I/O Output Latch Register bits  
 bit 3-0      **Unimplemented:** Read as '0'

**TABLE 9-1: PORTA I/O SUMMARY**

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RA0/KAI0/D+/PGD	RA0	— <sup>(1)</sup>	I	TTL	PORTA<0> data input; disabled when USB enabled.
	KAI0	— <sup>(1)</sup>	I	TTL	Interrupt-on-pin change; disabled when USB enabled.
	D+	— <sup>(1)</sup>	I	XCVR	USB bus differential plus line input (internal transceiver).
		— <sup>(1)</sup>	O	XCVR	USB bus differential plus line output (internal transceiver).
	PGD	— <sup>(1)</sup>	O	DIG	Serial execution data output for ICSP™.
		— <sup>(1)</sup>	I	ST	Serial execution data input for ICSP™.
RA1/KAI1/D-/PGC	RA1	— <sup>(1)</sup>	I	TTL	PORTA<1> data input; disabled when USB enabled.
	KAI1	— <sup>(1)</sup>	I	TTL	Interrupt-on-pin change; disabled when USB enabled.
	D-	— <sup>(1)</sup>	I	XCVR	USB bus differential minus line input (internal transceiver).
		— <sup>(1)</sup>	O	XCVR	USB bus differential minus line output (internal transceiver).
	PGC	— <sup>(1)</sup>	O	DIG	Serial execution clock output for ICSP™.
		— <sup>(1)</sup>	I	ST	Serial execution clock input for ICSP™.
MCLR/VPP/RA3/KAI3	MCLR	—	I	ST	External Master Clear input; enabled when MCLRE Configuration bit is set.
	VPP	—	I	ANA	High-voltage detection; used for ICSP™ mode entry detection. Always available, regardless of pin mode.
	RA3	— <sup>(2)</sup>	I	ST	PORTA<3> data input; enabled when MCLRE Configuration bit is clear; Programmable weak pull-up.
	KAI3	— <sup>(1)</sup>	I	TTL	Interrupt-on-pin change
RA4/KAI4/AN3/OSC2/CLKOUT	RA4	0	O	DIG	LATA<4> data output. Enabled in RCIO, INTIO2 and ECIO modes only.
		1	I	TTL	PORTA<4> data input; Programmable weak pull-up. Enabled in RCIO, INTIO2 and ECIO modes only.
	KAI4	1	I	TTL	Interrupt-on-pin change
	AN3	1	I	ANA	A/D input channel 3. Default configuration on POR.
	OSC2	x	O	ANA	Main oscillator feedback output connection (XT, HS and LP modes).
	CLKOUT	x	O	DIG	System cycle clock output (Fosc/4) in RC, INTIO1 and EC Oscillator modes.
RA5/KAI5/OSC1/CLKIN	RA5	0	O	DIG	LATA<5> data output. Disabled in external oscillator modes.
		1	I	TTL	PORTA<5> data input. Disabled in external oscillator modes; Programmable weak pull-up.
	KAI5	1	I	TTL	Interrupt-on-pin change
	OSC1	x	I	ANA	Main oscillator input connection.
	CLKIN	x	I	ANA	Main clock input connection.

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

**Note 1:** RA0 and RA1 do not have corresponding TRISA bits. In Port mode, these pins are input only. USB data direction is determined by the USB configuration.

**2:** RA3 does not have a corresponding TRISA bit. This pin is always an input regardless of mode.

# PIC18F1XK50/PIC18LF1XK50

---

TABLE 9-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTA	—	—	RA5 <sup>(1)</sup>	RA4 <sup>(1)</sup>	RA3 <sup>(2)</sup>	—	RA1 <sup>(3)</sup>	RA0 <sup>(3)</sup>	282
LATA	—	—	LATA5 <sup>(1)</sup>	LATA4 <sup>(1)</sup>	—	—	—	—	282
TRISA	—	—	TRISA5 <sup>(1)</sup>	TRISA4 <sup>(1)</sup>	—	—	—	—	282
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	282
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	282
IOCA	—	—	IOCA5	IOCA4	IOCA3 <sup>(2)</sup>	—	IOCA1 <sup>(3)</sup>	IOCA0 <sup>(3)</sup>	282
WPUA	—	—	WPUA5	WPUA4	WPUA3 <sup>(2)</sup>	—	—	—	282
UCON	—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—	282
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
INTCON2	RABPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	279

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

**Note 1:** RA<5:4> and their associated latch and data direction bits are enabled as I/O pins based on oscillator configuration; otherwise, they are read as '0'.

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

**3:** RA1 and RA0 are only available as port pins when the USB module is disabled (UCON<3> = 0).

## 9.2 PORTB, TRISB and LATB Registers

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

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

### EXAMPLE 9-2: INITIALIZING PORTB

```
CLRF    PORTB    ; Initialize PORTB by
                  ; clearing output
                  ; data latches
CLRF    LATB     ; Alternate method
                  ; to clear output
                  ; data latches
MOVLW  0F0h    ; Value used to
                  ; initialize data
                  ; direction
MOVWF  TRISB    ; Set RB<7:4> as outputs
```

All PORTB pins are individually configurable as interrupt-on-change pins. Control bits in the IOCB register enable (when set) or disable (when clear) the interrupt function for each pin.

When set, the RABIE bit of the INTCON register enables interrupts on all pins which also have their corresponding IOCB bit set. When clear, the RABIE bit disables all interrupt-on-changes.

Only pins configured as inputs can cause this interrupt to occur (i.e., any pin configured as an output is excluded from the interrupt-on-change comparison).

For enabled interrupt-on-change pins, the values are compared with the old value latched on the last read of PORTB. The 'mismatch' outputs of the last read are OR'd together to set the PORTB Change Interrupt flag bit (RABIF) in the INTCON register.

This interrupt can wake the device from the Sleep mode, or any of the Idle modes. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

- Any read or write of PORTB to clear the mismatch condition (except when PORTB is the source or destination of a MOVFF instruction).
- Clear the flag bit, RABIF.

A mismatch condition will continue to set the RABIF flag bit. Reading or writing PORTB will end the mismatch condition and allow the RABIF bit to be cleared. The latch

holding the last read value is not affected by a MCLR nor Brown-out Reset. After either one of these Resets, the RABIF flag will continue to be set if a mismatch is present.

**Note:** If a change on the I/O pin should occur when the read operation is being executed (start of the Q2 cycle), then the RABIF interrupt flag may not get set. Furthermore, since a read or write on a port affects all bits of that port, care must be taken when using multiple pins in Interrupt-on-change mode. Changes on one pin may not be seen while servicing changes on another pin.

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.

All PORTB pins have individually controlled weak internal pull-up. When set, each bit of the WPUB register enables the corresponding pin pull-up. When cleared, the RABPU bit of the INTCON2 register enables pull-ups on all pins which also have their corresponding WPUB bit set. When set, the RABPU bit disables all weak pull-ups. 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, RB<5:4> are configured as analog inputs by default and read as '0'.

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 9-6: PORTB: PORTB REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0
RB7	RB6	RB5	RB4	—	—	—	—
bit 7	bit 0						

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

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

bit 7-4      **RB<7:4>**: PORTB I/O Pin bit  
1 = Port pin is  $>V_{IH}$   
0 = Port pin is  $<V_{IL}$   
bit 3-0      **Unimplemented**: Read as '0'

## REGISTER 9-7: TRISB: PORTB TRI-STATE REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	U-0	U-0
TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—
bit 7	bit 0						

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

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

bit 7-4      **TRISB<7:4>**: PORTB Tri-State Control bit  
1 = PORTB pin configured as an input (tri-stated)  
0 = PORTB pin configured as an output  
bit 3-0      **Unimplemented**: Read as '0'

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 9-8: WPUB: WEAK PULL-UP PORTB REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	U-0	U-0
WPUB7	WPUB6	WPUB5	WPUB4	—	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-4      **WPUB<7:4>**: Weak Pull-up Enable bit

1 = Pull-up enabled

0 = Pull-up disabled

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

## REGISTER 9-9: IOCB: INTERRUPT-ON-CHANGE PORTB REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-4      **IOCB<7:4>**: Interrupt-on-change bits

1 = Interrupt-on-change enabled

0 = Interrupt-on-change disabled

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

## REGISTER 9-10: LATB: PORTB DATA LATCH REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0
LATB7	LATB6	LATB5	LATB4	—	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-4      **LATB<7:4>**: RB<7:4> Port I/O Output Latch Register bits

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

# PIC18F1XK50/PIC18LF1XK50

---



---

**TABLE 9-3: PORTB I/O SUMMARY**

Pin	Function	TRIS Setting	I/O	I/O Type	Description	
RB4/KBI4/AN10/ SDI/SDA	RB4	0	O	DIG	LATB<4> data output; not affected by analog input.	
		1	I	TTL	PORTB<4> data input; Programmable weak pull-up.	
	KBI4	1	I	TTL	Interrupt-on-pin change.	
	AN10	1	I	ANA	ADC input channel 10.	
	SDI	1	I	ST	SPI data input (MSSP module).	
	SDA	1	I	DIG	I <sup>2</sup> C <sup>TM</sup> data output (MSSP module); takes priority over port data.	
		1	O	I2C	I <sup>2</sup> C <sup>TM</sup> data input (MSSP module); input type depends on module setting.	
RB5/KBI5/AN11/ RX/DT	RB5	0	O	DIG	LATB<5> data output.	
		1	I	TTL	PORTB<5> data input; Programmable weak pull-up.	
	KBI5	1	I	TTL	Interrupt-on-pin change.	
	AN11	1	I	ANA	ADC input channel 11.	
	RX	1	I	ST	Asynchronous serial receive data input (USART module).	
		DT	1	O	DIG	Synchronous serial data output (USART module); takes priority over port data.
			1	I	ST	Synchronous serial data input (USART module). User must configure as an input.
RB6/KBI6/SCK/ SCL	RB6	0	O	DIG	LATB<6> data output.	
		1	I	TTL	PORTB<6> data input; Programmable weak pull-up.	
	KBI6	1	I	TTL	Interrupt-on-pin change.	
	SCK	0	O	DIG	SPI clock output (MSSP module); takes priority over port data.	
		1	I	ST	SPI clock input (MSSP module).	
	SCL	0	O	DIG	I <sup>2</sup> C <sup>TM</sup> clock output (MSSP module); takes priority over port data.	
		1	I	I2C	I <sup>2</sup> C <sup>TM</sup> clock input (MSSP module); input type depends on module setting.	
RB7/KBI7/TX/CK	RB7	0	O	DIG	LATB<7> data output.	
		1	I	TTL	PORTB<7> data input; Programmable weak pull-up.	
	KBI7	1	I	TTL	Interrupt-on-pin change.	
	TX	1	O	DIG	Asynchronous serial transmit data output (USART module); takes priority over port data. User must configure as output.	
	CK	1	O	DIG	Synchronous serial clock output (USART module); takes priority over port data.	
		1	I	ST	Synchronous serial clock input (USART module).	

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

# PIC18F1XK50/PIC18LF1XK50

TABLE 9-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTB	RB7	RB6	RB5	RB4	—	—	—	—	282
LATB	LATB7	LATB6	LATB5	LATB4	—	—	—	—	282
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	282
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	—	—	—	—	282
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—	282
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	282
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
INTCON2	<u>RABPU</u>	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	279
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	282
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	281
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	281
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	280

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

# PIC18F1XK50/PIC18LF1XK50

## 9.3 PORTC, TRISC and LATC Registers

PORTC is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., disable the output driver). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin).

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

All the pins on PORTC are implemented with Schmitt Trigger input buffer. Each pin is individually configurable as an input or output.

**Note:** On a Power-on Reset, RC<7:6> and RC<3:0> are configured as analog inputs and read as '0'.

### EXAMPLE 9-3: INITIALIZING PORTC

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

### REGISTER 9-11: PORTC: PORTC REGISTER

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| RC7   | RC6   | RC5   | RC4   | RC3   | RC2   | RC1   | RC0   |
| bit 7 |       |       |       |       |       |       | bit 0 |

#### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

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

bit 7-0      **RC<7:0>**: PORTC I/O Pin bit  
1 = Port pin is > VIH  
0 = Port pin is < VIL

### REGISTER 9-12: TRISC: PORTC TRI-STATE REGISTER

| R/W-1  |
|--------|--------|--------|--------|--------|--------|--------|--------|
| TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| bit 7  |        |        |        |        |        |        | bit 0  |

#### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

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

bit 7-0      **TRISC<7:0>**: PORTC Tri-State Control bit  
1 = PORTC pin configured as an input (tri-stated)  
0 = PORTC pin configured as an output

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 9-13: LATC: PORTC DATA LATCH REGISTER

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-0

**LATC<7:0>**: RB<7:0> Port I/O Output Latch Register bits

# PIC18F1XK50/PIC18LF1XK50

---

TABLE 9-14: PORTC I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RC0/AN4/ C12IN+/VREF+/ INT0	RC0	0	O	DIG	LATC<0> data output.
		1	I	ST	PORTC<0> data input.
	AN4	1	I	ANA	A/D input channel 4.
	C12IN+	1	I	ANA	Comparators C1 and C2 non-inverting input. Analog select is shared with ADC.
	VREF+	1	I	ANA	ADC and comparator voltage reference high input.
	INT0	1	I	ST	External Interrupt 0 input.
RC1/AN5/ C12IN1-/VREF-/ INT1	RC1	0	O	DIG	LATC<1> data output.
		1	I	ST	PORTC<1> data input.
	AN5	1	I	ANA	A/D input channel 5.
	C12IN1-	1	I	ANA	Comparators C1 and C2 inverting input. Analog select is shared with ADC.
	VREF-	1	I	ANA	ADC and comparator voltage reference low input.
	INT1	1	I	ST	External Interrupt 1 input.
RC2/AN6/ C12IN2-/CVREF/ P1D/INT2	RC2	0	O	DIG	LATC<2> data output.
		1	I	ST	PORTC<2> data input.
	AN6	1	I	ANA	A/D input channel 6.
	C12IN2-	1	I	ANA	Comparators C1 and C2 inverting input, channel 2. Analog select is shared with ADC.
	CVREF	x	O	ANA	Voltage reference output. Enabling this feature disables digital I/O.
	P1D	0	O	DIG	ECCP1 Enhanced PWM output, channel D. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data.
RC3/AN7/ C12IN3-/P1C/ PGM	RC3	0	O	DIG	LATC<3> data output.
		1	I	ST	PORTC<3> data input.
	AN7	1	I	ANA	A/D input channel 7.
	C12IN3-	1	I	ANA	Comparators C1 and C2 inverting input, channel 3. Analog select is shared with ADC.
	P1C	0	O	DIG	ECCP1 Enhanced PWM output, channel C. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data.
	PGM	x	I	ST	Single-Supply Programming mode entry (ICSP™). Enabled by LVP Configuration bit; all other pin functions disabled.
RC4/C12OUT/ P1B	RC4	0	O	DIG	LATC<4> data output.
		1	I	ST	PORTC<4> data input.
	C12OUT	0	O	DIG	Comparator 1 and 2 output; takes priority over port data.
	P1B	0	O	DIG	ECCP1 Enhanced PWM output, channel B. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data.

**Legend:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I<sup>2</sup>C/SMB = I<sup>2</sup>C/SMBus input buffer; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

# PIC18F1XK50/PIC18LF1XK50

TABLE 9-14: PORTC I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description		
RC5/CCP1/P1A/ T0CKI	RC5	0	O	DIG	LATC<5> data output.		
		1	I	ST	PORTC<5> data input.		
	CCP1	0	O	DIG	ECCP1 compare or PWM output; takes priority over port data.		
		1	I	ST	ECCP1 capture input.		
	P1A	0	0	DIG	ECCP1 Enhanced PWM output, channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data		
RC6/AN8/SS/ T13CKI/T1OSCI	RC6	0	O	DIG	LATC<6> data output.		
		1	I	ST	PORTC<6> data input.		
	AN8	1	I	ANA	A/D input channel 8.		
	SS	1	I	TTL	Slave select input for SSP (MSSP module)		
	T13CKI	1	I	ST	Timer1 and Timer3 counter input.		
	T1OSCI	x	O	ANA	Timer1 oscillator input; enabled when Timer1 oscillator enabled. Disables digital I/O.		
RC7/AN9/SDO/ T1OSCO	RC7	0	O	DIG	LATC<7> data output.		
		1	I	ST	PORTC<7> data input.		
	AN9	1	I	ANA	A/D input channel 9.		
	SDO	0	I	DIG	SPI data output (MSSP module); takes priority over port data.		
	T1OSCO	x	O	ANA	Timer1 oscillator output; enabled when Timer1 oscillator enabled. Disables digital I/O.		

**Legend:** DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I<sup>2</sup>C/SMB = I<sup>2</sup>C/SMBus input buffer; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

TABLE 9-5: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	282
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	282
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	282
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	282
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	280
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON	281
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	280
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	281
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	281
PSTRCON	—	—	—	STRSYNC	STRD	STRC	STRB	STRA	281
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	282
REFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	---	D1NSS	281
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
INTCON2	RAFPB	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	279
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	279

# PIC18F1XK50/PIC18LF1XK50

## 9.4 Port Analog Control

Some port pins are multiplexed with analog functions such as the Analog-to-Digital Converter and comparators. When these I/O pins are to be used as analog inputs it is necessary to disable the digital input buffer to avoid excessive current caused by improper biasing of the digital input. Individual control of the digital input buffers on pins which share analog functions is provided by the ANSEL and ANSELH registers. Setting an

ANS<sub>x</sub> bit high will disable the associated digital input buffer and cause all reads of that pin to return '0' while allowing analog functions of that pin to operate correctly.

The state of the ANS<sub>x</sub> bits has no affect on digital output functions. A pin with the associated TRIS<sub>x</sub> bit clear and ANS<sub>x</sub> bit set will still operate as a digital output but the Input mode will be analog.

### REGISTER 9-15: ANSEL: ANALOG SELECT REGISTER 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	U-0
ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—
bit 7	bit 0						

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7      **ANS7:** RC3 Analog Select Control bit  
1 = Digital input buffer of RC3 is disabled  
0 = Digital input buffer of RC3 is enabled
- bit 6      **ANS6:** RC2 Analog Select Control bit  
1 = Digital input buffer of RC2 is disabled  
0 = Digital input buffer of RC2 is enabled
- bit 5      **ANS5:** RC1 Analog Select Control bit  
1 = Digital input buffer of RC1 is disabled  
0 = Digital input buffer of RC1 is enabled
- bit 4      **ANS4:** RC0 Analog Select Control bit  
1 = Digital input buffer of RC0 is disabled  
0 = Digital input buffer of RC0 is enabled
- bit 3      **ANS3:** RA4 Analog Select Control bit  
1 = Digital input buffer of RA4 is disabled  
0 = Digital input buffer of RA4 is enabled
- bit 2-0      **Unimplemented:** Read as '0'

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 9-16: ANSELH: ANALOG SELECT REGISTER 2

U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	—	ANS11	ANS10	ANS9	ANS8
bit 7							

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-4      **Unimplemented:** Read as '0'

bit 3      **ANS11:** RB5 Analog Select Control bit

1 = Digital input buffer of RB5 is disabled  
0 = Digital input buffer of RB5 is enabled

bit 2      **ANS10:** RB4 Analog Select Control bit

1 = Digital input buffer of RB4 is disabled  
0 = Digital input buffer of RB4 is enabled

bit 1      **ANS9:** RC7 Analog Select Control bit

1 = Digital input buffer of RC7 is disabled  
0 = Digital input buffer of RC7 is enabled

bit 0      **ANS8:** RC6 Analog Select Control bit

1 = Digital input buffer of RC6 is disabled  
0 = Digital input buffer of RC6 is enabled

# PIC18F1XK50/PIC18LF1XK50

---

---

## 9.5 Port Slew Rate Control

The output slew rate of each port is programmable to select either the standard transition rate or a reduced transition rate of 0.1 times the standard to minimize EMI. The reduced transition time is the default slew rate for all ports.

### REGISTER 9-17: SLRCON: SLEW RATE CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1
—	—	—	—	—	SLRC	SLRB	SLRA
bit 7							bit 0

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-3

**Unimplemented:** Read as '0'

bit 2

**SLRC:** PORTC Slew Rate Control bit

1 = All outputs on PORTC slew at 0.1 times the standard rate

0 = All outputs on PORTC slew at the standard rate

bit 1

**SLRB:** PORTB Slew Rate Control bit

1 = All outputs on PORTB slew at 0.1 times the standard rate

0 = All outputs on PORTB slew at the standard rate

bit 0

**SLRA:** PORTA Slew Rate Control bit

1 = All outputs on PORTA slew at 0.1 times the standard rate<sup>(1)</sup>

0 = All outputs on PORTA slew at the standard rate

**Note 1:** The slew rate of RA4 defaults to standard rate when the pin is used as CLKOUT.

## 10.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 10-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

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

### 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						

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>TMR0ON:</b> Timer0 On/Off Control bit 1 = Enables Timer0 0 = Stops Timer0
bit 6	<b>T08BIT:</b> Timer0 8-bit/16-bit Control bit 1 = Timer0 is configured as an 8-bit timer/counter 0 = Timer0 is configured as a 16-bit timer/counter
bit 5	<b>T0CS:</b> Timer0 Clock Source Select bit 1 = Transition on T0CKI pin 0 = Internal instruction cycle clock (CLKOUT)
bit 4	<b>T0SE:</b> Timer0 Source Edge Select bit 1 = Increment on high-to-low transition on T0CKI pin 0 = Increment on low-to-high transition on T0CKI pin
bit 3	<b>PSA:</b> Timer0 Prescaler Assignment bit 1 = Timer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler. 0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.
bit 2-0	<b>T0PS&lt;2:0&gt;:</b> Timer0 Prescaler Select bits 111 = 1:256 prescale value 110 = 1:128 prescale value 101 = 1:64 prescale value 100 = 1:32 prescale value 011 = 1:16 prescale value 010 = 1:8 prescale value 001 = 1:4 prescale value 000 = 1:2 prescale value

# PIC18F1XK50/PIC18LF1XK50

## 10.1 Timer0 Operation

Timer0 can operate as either a timer or a counter; the mode is selected with the T0CS bit of the T0CON register. In Timer mode (T0CS = 0), the module increments on every clock by default unless a different prescaler value is selected (see **Section 10.3 “Prescaler”**). Timer0 incrementing is inhibited for two instruction cycles following a TMR0 register write. The user can work around this by adjusting the value written to the TMR0 register to compensate for the anticipated missing increments.

The Counter mode is selected by setting the T0CS bit (= 1). In this mode, Timer0 increments either on every rising or falling edge of the T0CKI pin. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE of the T0CON register; clearing this bit selects the rising edge. Restrictions on the external clock input are discussed below.

An external clock source can be used to drive Timer0; however, it must meet certain requirements (see Table 27-6) to ensure that the external clock can be synchronized with the internal phase clock (Tosc). There is a delay between synchronization and the onset of incrementing the timer/counter.

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

TMR0H is not the actual high byte of Timer0 in 16-bit mode; it is actually a buffered version of the real high byte of Timer0 which is neither directly readable nor writable (refer to Figure 10-2). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without the need to verify that the read of the high and low byte were valid. Invalid reads could otherwise occur due to a rollover between successive reads of the high and low byte.

Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. Writing to TMR0H does not directly affect Timer0. Instead, the high byte of Timer0 is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

**FIGURE 10-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)**

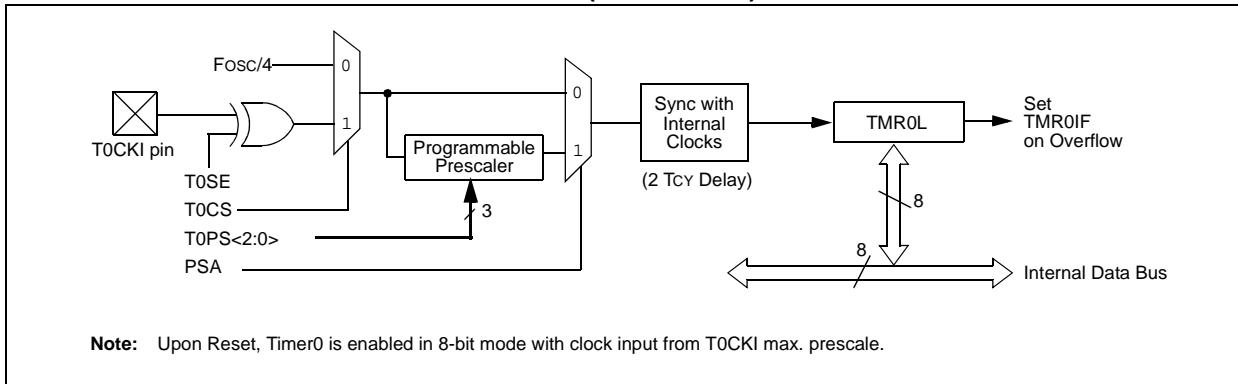
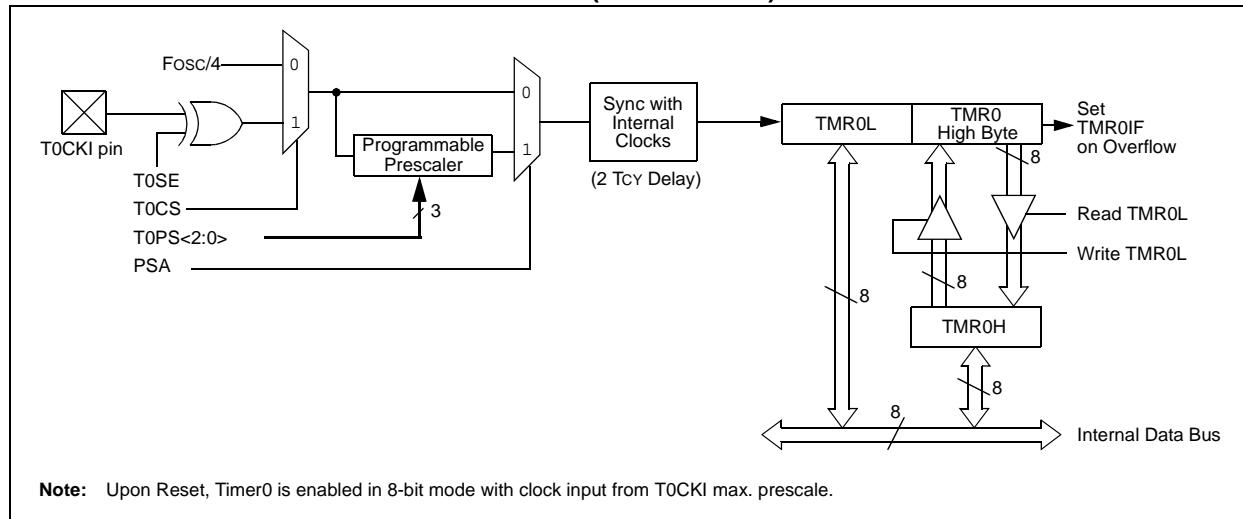


FIGURE 10-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



### 10.3 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not directly readable or writable; its value is set by the PSA and T0PS<2:0> bits of the T0CON register which determine the prescaler assignment and prescale ratio.

Clearing the PSA bit assigns the prescaler to the Timer0 module. When the prescaler is assigned, prescale values from 1:2 through 1:256 in integer power-of-2 increments are selectable.

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

**Note:** Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count but will not change the prescaler assignment.

#### 10.3.1 SWITCHING PRESCALER ASSIGNMENT

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

### 10.4 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or from FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF flag bit. The interrupt can be masked by clearing the TMR0IE bit of the INTCON register. Before re-enabling the interrupt, the TMR0IF bit must be cleared by software in the Interrupt Service Routine.

Since Timer0 is shut down in Sleep mode, the TMR0 interrupt cannot awaken the processor from Sleep.

TABLE 10-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TMR0L	Timer0 Register, Low Byte								280
TMR0H	Timer0 Register, High Byte								280
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	280
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282

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

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

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 11.0 TIMER1 MODULE

The Timer1 timer/counter module incorporates the following features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR1H and TMR1L)
- Selectable internal or external clock source and Timer1 oscillator options
- Interrupt-on-overflow
- Reset on CCP Special Event Trigger
- Device clock status flag (T1RUN)

A simplified block diagram of the Timer1 module is shown in Figure 11-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 11-2.

The module incorporates its own low-power oscillator to provide an additional clocking option. The Timer1 oscillator can also be used as a low-power clock source for the microcontroller in power-managed operation.

Timer1 can also be used to provide Real-Time Clock (RTC) functionality to applications with only a minimal addition of external components and code overhead.

Timer1 is controlled through the T1CON Control register (Register 11-1). It also contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON of the T1CON register.

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

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	<u>T1SYNC</u>	TMR1CS	TMR1ON
bit 7	bit 0						

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>RD16:</b> 16-bit Read/Write Mode Enable bit 1 = Enables register read/write of Timer1 in one 16-bit operation 0 = Enables register read/write of Timer1 in two 8-bit operations
bit 6	<b>T1RUN:</b> Timer1 System Clock Status bit 1 = Main system clock is derived from Timer1 oscillator 0 = Main system clock is derived from another source
bit 5-4	<b>T1CKPS&lt;1:0&gt;:</b> Timer1 Input Clock Prescale Select bits 11 = 1:8 Prescale value 10 = 1:4 Prescale value 01 = 1:2 Prescale value 00 = 1:1 Prescale value
bit 3	<b>T1OSCEN:</b> Timer1 Oscillator Enable bit 1 = Timer1 oscillator is enabled 0 = Timer1 oscillator is shut off The oscillator inverter and feedback resistor are turned off to eliminate power drain.
bit 2	<b>T1SYNC:</b> Timer1 External Clock Input Synchronization Select bit <u>When TMR1CS = 1:</u> 1 = Do not synchronize external clock input 0 = Synchronize external clock input <u>When TMR1CS = 0:</u> This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.
bit 1	<b>TMR1CS:</b> Timer1 Clock Source Select bit 1 = External clock from the T13CKI pin (on the rising edge) 0 = Internal clock (Fosc/4)
bit 0	<b>TMR1ON:</b> Timer1 On bit 1 = Enables Timer1 0 = Stops Timer1

# PIC18F1XK50/PIC18LF1XK50

## 11.1 Timer1 Operation

Timer1 can operate in one of the following modes:

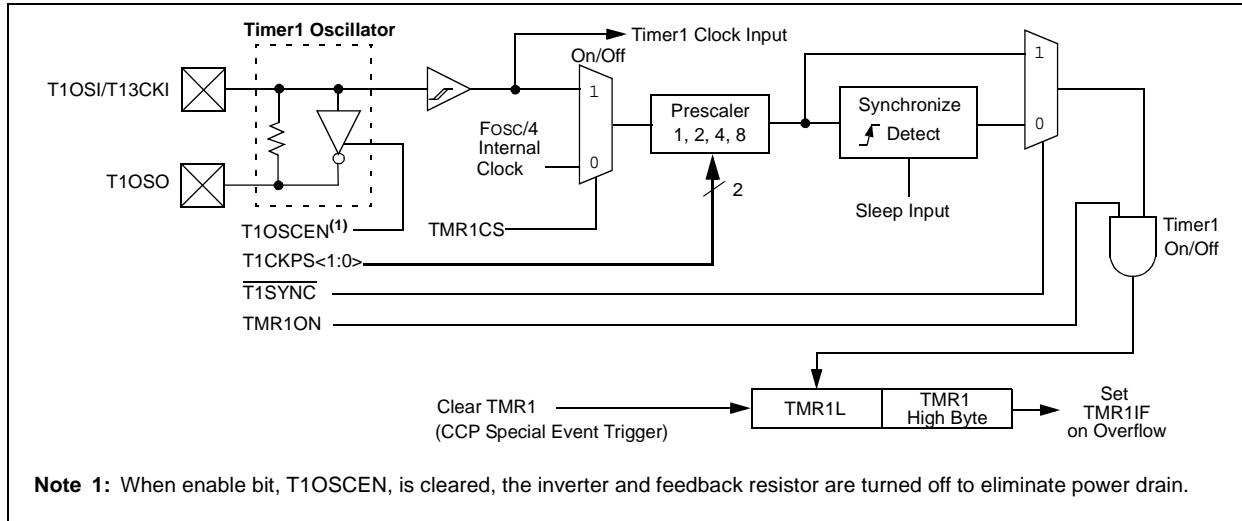
- Timer
- Synchronous Counter
- Asynchronous Counter

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

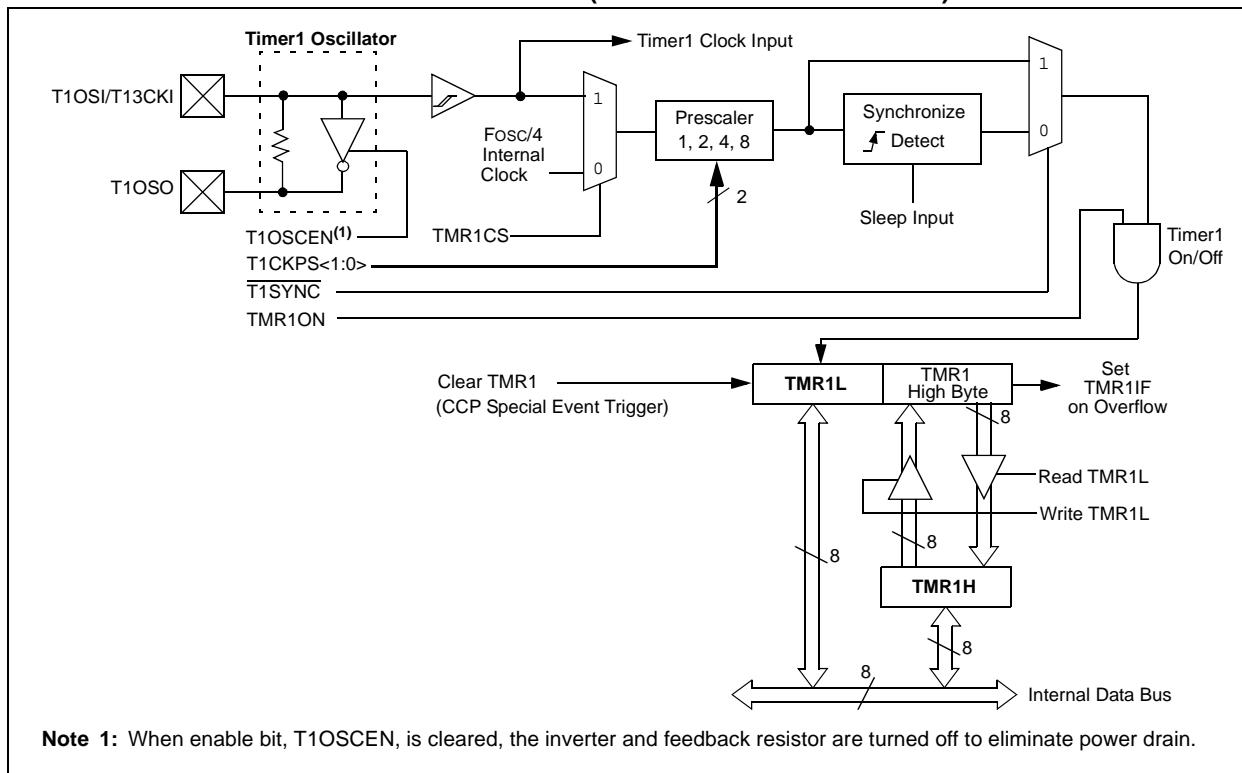
instruction cycle (Fosc/4). When the bit is set, Timer1 increments on every rising edge of either the Timer1 external clock input or the Timer1 oscillator, if enabled.

When the Timer1 oscillator is enabled, the digital circuitry associated with the T1OSI and T1OSO pins is disabled. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

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



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



## 11.2 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 of the T1CON register 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 the need to determine whether a read of the high byte, followed by a read of the low byte, has become invalid due to a rollover or carry between reads.

Writing to TMR1H does not directly affect Timer1. Instead, the high byte of Timer1 is updated with the contents of TMR1H when a write occurs to TMR1L. This allows all 16 bits of Timer1 to be updated at once.

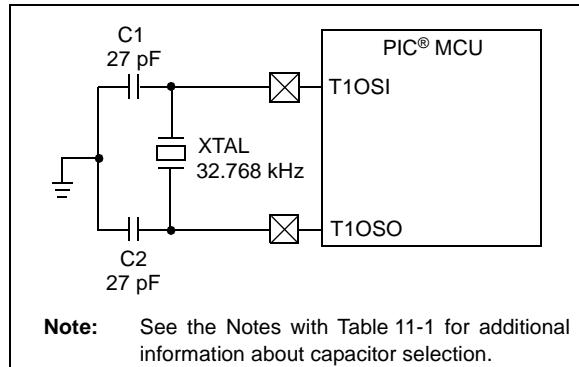
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

## 11.3 Timer1 Oscillator

An on-chip crystal oscillator circuit is incorporated between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting the Timer1 Oscillator Enable bit, T1OSCEN of the T1CON register. The oscillator is a low-power circuit rated for 32 kHz crystals. It will continue to run during all power-managed modes. The circuit for a typical LP oscillator is shown in Figure 11-3. 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.

**FIGURE 11-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR**



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

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

**Note 1:** Microchip suggests these values only 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.

### 11.3.1 USING TIMER1 AS A CLOCK SOURCE

The Timer1 oscillator is also available as a clock source in power-managed modes. By setting the clock select bits, SCS<1:0> of the OSCCON register, to '01', the device switches to SEC\_RUN mode; both the CPU and peripherals are clocked from the Timer1 oscillator. If the IDLEN bit of the OSCCON register is cleared and a SLEEP instruction is executed, the device enters SEC\_IDLE mode. Additional details are available in **Section 19.0 “Power-Managed Modes”**.

Whenever the Timer1 oscillator is providing the clock source, the Timer1 system clock status flag, T1RUN of the T1CON register, is set. This can be used to determine the controller's current clocking mode. It can also indicate which clock source is currently being used by the Fail-Safe Clock Monitor. If the Clock Monitor is enabled and the Timer1 oscillator fails while providing the clock, polling the T1RUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

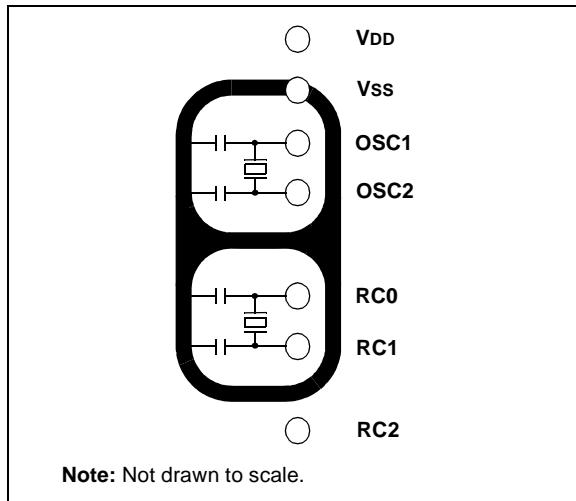
## 11.3.2 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

The Timer1 oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity.

The oscillator circuit, shown in Figure 11-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than Vss or Vdd.

If a high-speed circuit must be located near the oscillator (such as the CCP1 pin in Output Compare or PWM mode, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as shown in Figure 11-4, may be helpful when used on a single-sided PCB or in addition to a ground plane.

**FIGURE 11-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING**



## 11.4 Timer1 Interrupt

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

## 11.5 Resetting Timer1 Using the CCP Special Event Trigger

If either of the CCP modules is configured to use Timer1 and generate a Special Event Trigger in Compare mode (CCP1M<3:0> or CCP2M<3:0> = 1011), this signal will reset Timer1. The trigger from CCP2 will also start an A/D conversion if the A/D module is enabled (see **Section 14.3.4 “Special Event Trigger”** for more information).

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

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

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

**Note:** The Special Event Triggers from the CCP2 module will not set the TMR1IF interrupt flag bit of the PIR1 register.

## 11.6 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 11.3 “Timer1 Oscillator”** above) gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or supercapacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, RTCisr, shown in Example 11-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow triggers the interrupt and calls the routine, which increments the seconds counter by one; additional counters for minutes and hours are incremented on overflows of the less significant counters.

### EXAMPLE 11-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE

```

RTCinit
    MOVLW  80h          ; Preload TMR1 register pair
    MOVWF  TMR1H         ; for 1 second overflow
    CLRF   TMR1L
    MOVLW  b'000001111'  ; Configure for external clock,
    MOVWF  T1CON         ; Asynchronous operation, external oscillator
    CLRF   secs           ; Initialize timekeeping registers
    CLRF   mins           ;
    MOVLW  .12
    MOVWF  hours
    BSF    PIE1, TMR1IE   ; Enable Timer1 interrupt
    RETURN

RTCisr
    BSF    TMR1H, 7       ; Preload for 1 sec overflow
    BCF    PIR1, TMR1IF   ; Clear interrupt flag
    INCF   secs, F         ; Increment seconds
    MOVLW  .59             ; 60 seconds elapsed?
    CPFSGT secs
    RETURN                 ; No, done
    CLRF   secs           ; Clear seconds
    INCF   mins, F         ; Increment minutes
    MOVLW  .59             ; 60 minutes elapsed?
    CPFSGT mins
    RETURN                 ; No, done
    CLRF   mins           ; clear minutes
    INCF   hours, F        ; Increment hours
    MOVLW  .23             ; 24 hours elapsed?
    CPFSGT hours
    RETURN                 ; No, done
    CLRF   hours          ; Reset hours
    RETURN                 ; Done

```

Since the register pair is 16 bits wide, a 32.768 kHz clock source will take 2 seconds to count up to overflow. To force the overflow at the required one-second intervals, it is necessary to preload it; the simplest method is to set the MSb of TMR1H with a BSF instruction. Note that the TMR1L register is never preloaded or altered; doing so may introduce cumulative error over many cycles.

For this method to be accurate, Timer1 must operate in Asynchronous mode and the Timer1 overflow interrupt must be enabled (PIE1<0> = 1), as shown in the routine, RTCinit. The Timer1 oscillator must also be enabled and running at all times.

# PIC18F1XK50/PIC18LF1XK50

---

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	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
TMR1L	Timer1 Register, Low Byte								280
TMR1H	Timer1 Register, High Byte								280
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	280
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	282
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	280

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

## 12.0 TIMER2 MODULE

The Timer2 module timer incorporates the following features:

- 8-bit timer and period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable postscale (1:1 through 1:16)
- Interrupt on TMR2-to-PR2 match
- Optional use as the shift clock for the MSSP module

The module is controlled through the T2CON register (Register 12-1), which enables or disables the timer and configures the prescaler and postscale. Timer2 can be shut off by clearing control bit, TMR2ON of the T2CON register, to minimize power consumption.

A simplified block diagram of the module is shown in Figure 12-1.

## 12.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (Fosc/4). A 4-bit counter/prescaler on the clock input gives direct input, divide-by-4 and divide-by-16 prescale options; these are selected by the prescaler control bits, T2CKPS<1:0> of the T2CON register. The value of TMR2 is compared to that of the period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/postscale (see **Section 12.2 “Timer2 Interrupt”**).

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, whereas the PR2 register initializes to FFh. Both the prescaler and postscale counters are cleared on the following events:

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

TMR2 is not cleared when T2CON is written.

### REGISTER 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
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7	bit 0						

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>Unimplemented:</b> Read as '0'
bit 6-3	<b>T2OUTPS&lt;3:0&gt;:</b> Timer2 Output Postscale Select bits
	0000 = 1:1 Postscale
	0001 = 1:2 Postscale
	•
	•
	•
	1111 = 1:16 Postscale
bit 2	<b>TMR2ON:</b> Timer2 On bit
	1 = Timer2 is on
	0 = Timer2 is off
bit 1-0	<b>T2CKPS&lt;1:0&gt;:</b> Timer2 Clock Prescale Select bits
	00 = Prescaler is 1
	01 = Prescaler is 4
	1x = Prescaler is 16

# PIC18F1XK50/PIC18LF1XK50

## 12.2 Timer2 Interrupt

Timer2 can also generate an optional device interrupt. The Timer2 output signal (TMR2-to-PR2 match) provides the input for the 4-bit output counter/postscaler. This counter generates the TMR2 match interrupt flag which is latched in TMR2IF of the PIR1 register. The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE of the PIE1 register.

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

## 12.3 Timer2 Output

The unscaled output of TMR2 is available primarily to the CCP modules, where it is used as a time base for operations in PWM mode.

Timer2 can be optionally used as the shift clock source for the MSSP module operating in SPI mode. Additional information is provided in **Section 14.0 “Master Synchronous Serial Port (MSSP) Module”**.

FIGURE 12-1: TIMER2 BLOCK DIAGRAM

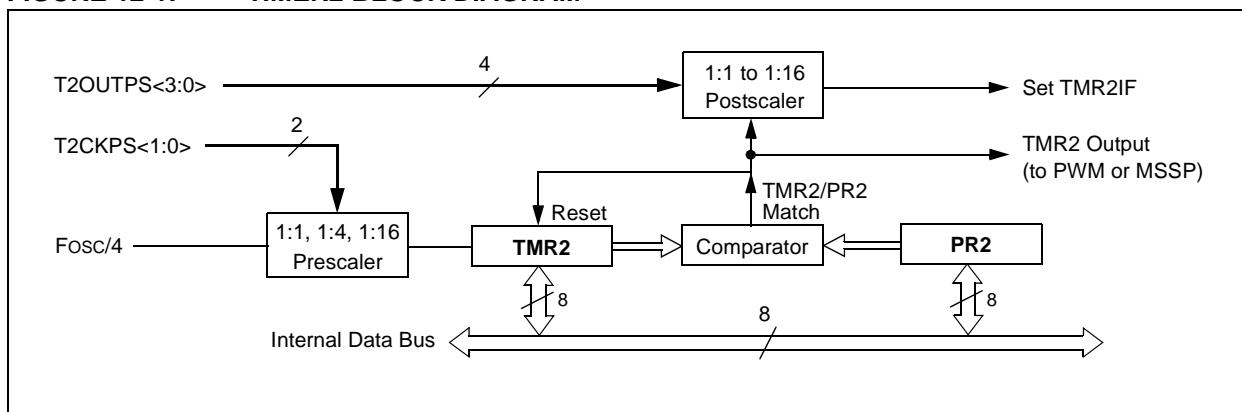


TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
TMR2	Timer2 Register								280
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	280
PR2	Timer2 Period Register								280

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

## 13.0 TIMER3 MODULE

The Timer3 module timer/counter incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR3H and TMR3L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Module Reset on CCP Special Event Trigger

A simplified block diagram of the Timer3 module is shown in Figure 13-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 13-2.

The Timer3 module is controlled through the T3CON register (Register 13-1). It also selects the clock source options for the CCP modules (see **Section 14.1.1 "CCP Module and Timer Resources"** for more information).

### REGISTER 13-1: T3CON: TIMER3 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
RD16	—	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC	TMR3CS	TMR3ON
bit 7	bit 0						

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<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	<b>Unimplemented:</b> Read as '0'
bit 5-4	<b>T3CKPS&lt;1:0&gt;:</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 3	<b>T3CCP1:</b> Timer3 and Timer1 to CCP1 Enable bits 1 = Timer3 is the clock source for compare/capture of ECCP1 0 = Timer1 is the clock source for compare/capture of ECCP1
bit 2	<b>T3SYNC:</b> Timer3 External Clock Input Synchronization Control bit (Not usable if the device 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 T13CKI (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

# PIC18F1XK50/PIC18LF1XK50

## 13.1 Timer3 Operation

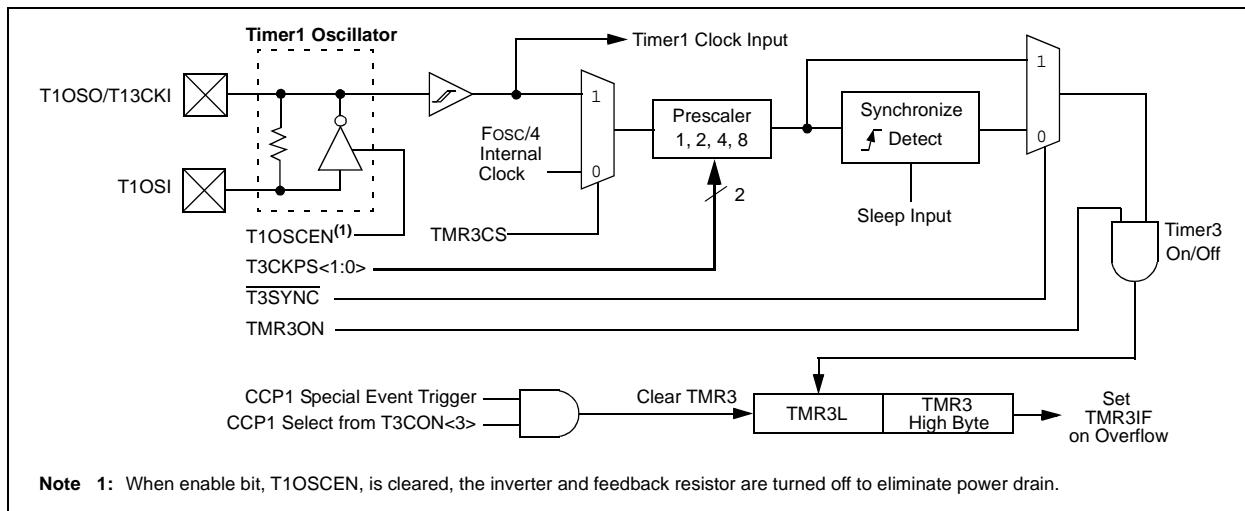
Timer3 can operate in one of three modes:

- Timer
- Synchronous Counter
- Asynchronous Counter

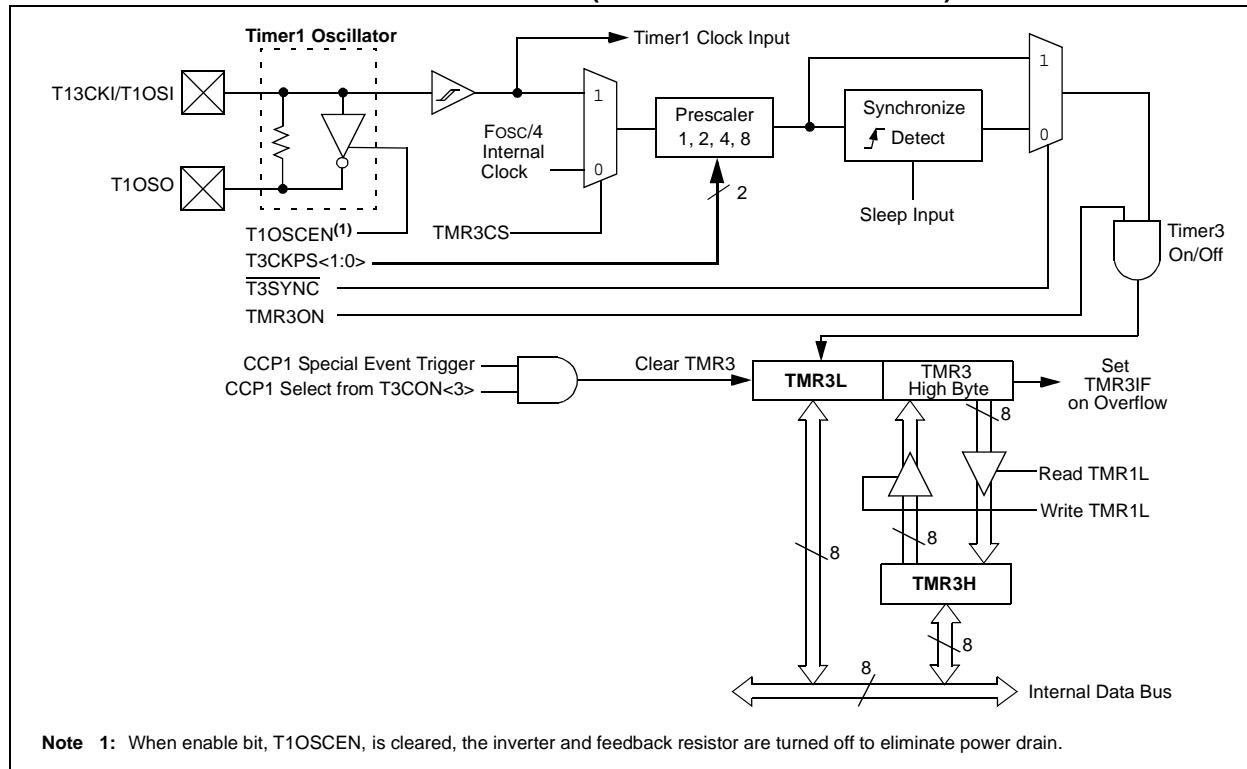
The operating mode is determined by the clock select bit, TMR3CS of the T3CON register. When TMR3CS is cleared (= 0), Timer3 increments on every internal instruction cycle (Fosc/4). When the bit is set, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator,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

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

FIGURE 13-1: TIMER3 BLOCK DIAGRAM



**FIGURE 13-2: TIMER3 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**



## 13.2 Timer3 16-Bit Read/Write Mode

Timer3 can be configured for 16-bit reads and writes (see Figure 13-2). When the RD16 control bit of the T3CON register is set, the address for TMR3H is mapped to a buffer register for the high byte of Timer3. A read from TMR3L will load the contents of the high byte of Timer3 into the Timer3 High Byte Buffer register. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, has become invalid due to a rollover between reads.

A write to the high byte of Timer3 must also take place through the TMR3H Buffer register. The Timer3 high byte is updated with the contents of TMR3H when a write occurs to TMR3L. This allows a user to write all 16 bits to both the high and low bytes of Timer3 at once.

The high byte of Timer3 is not directly readable or writable in this mode. All reads and writes must take place through the Timer3 High Byte Buffer register.

Writes to TMR3H do not clear the Timer3 prescaler. The prescaler is only cleared on writes to TMR3L.

## 13.3 Using the Timer1 Oscillator as the Timer3 Clock Source

The Timer1 internal oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN bit of the T1CON register. To use it as the Timer3 clock source, the TMR3CS bit must also be set. As previously noted, this also configures Timer3 to increment on every rising edge of the oscillator source.

The Timer1 oscillator is described in **Section 11.0 “Timer1 Module”**.

## 13.4 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and overflows to 0000h. The Timer3 interrupt, if enabled, is generated on overflow and is latched in interrupt flag bit, TMR3IF of the PIR2 register. This interrupt can be enabled or disabled by setting or clearing the Timer3 Interrupt Enable bit, TMR3IE of the PIE2 register.

# PIC18F1XK50/PIC18LF1XK50

---

## 13.5 Resetting Timer3 Using the CCP Special Event Trigger

If CCP1 module is configured to use Timer3 and to generate a Special Event Trigger in Compare mode (CCP1M<sub>3:0</sub>>), this signal will reset Timer3. It will also start an A/D conversion if the A/D module is enabled (see **Section 17.2.8 “Special Event Trigger”** for more information).

The module must be configured as either a timer or synchronous counter to take advantage of this feature. When used this way, the CCP1H:CCP1L register pair effectively becomes a period register for Timer3.

If Timer3 is running in Asynchronous Counter mode, the Reset operation may not work.

In the event that a write to Timer3 coincides with a Special Event Trigger from a CCP module, the write will take precedence.

**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	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	CCP2IF	282
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	CCP2IE	282
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	CCP2IP	282
TMR3L	Timer3 Register, Low Byte								281
TMR3H	Timer3 Register, High Byte								281
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	280
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	281
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	282

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

## 14.0 ENHANCED CAPTURE/COMPARE/PWM (ECCP) MODULE

PIC18F1XK50/PIC18LF1XK50 devices have one ECCP (Capture/Compare/PWM) module. The module contains a 16-bit register which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register.

CCP1 is implemented as a standard CCP module with enhanced PWM capabilities. These include:

- Provision for 2 or 4 output channels
- Output steering
- Programmable polarity
- Programmable dead-band control
- Automatic shutdown and restart.

The enhanced features are discussed in detail in [Section 14.4 “PWM \(Enhanced Mode\)”](#).

### REGISTER 14-1: CCP1CON: ENHANCED CAPTURE/COMPARE/PWM CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7				bit 0			

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6	<b>P1M&lt;1:0&gt;</b> : Enhanced PWM Output Configuration bits <u>If CCP1M&lt;3:2&gt; = 00, 01, 10:</u> xx = P1A assigned as Capture/Compare input/output; P1B, P1C, P1D assigned as port pins <u>If CCP1M&lt;3:2&gt; = 11:</u> 00 = Single output: P1A, P1B, P1C and P1D controlled by steering (See <a href="#">Section 14.4.7 “Pulse Steering Mode”</a> ). 01 = Full-bridge output forward: P1D modulated; P1A active; P1B, P1C inactive 10 = Half-bridge output: P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins 11 = Full-bridge output reverse: P1B modulated; P1C active; P1A, P1D inactive
bit 5-4	<b>DC1B&lt;1:0&gt;</b> : PWM Duty Cycle bit 1 and bit 0 <u>Capture mode:</u> Unused. <u>Compare mode:</u> Unused. <u>PWM mode:</u> These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.
bit 3-0	<b>CCP1M&lt;3:0&gt;</b> : Enhanced CCP Mode Select bits 0000 = Capture/Compare/PWM off (resets CCP module) 0001 = Reserved 0010 = Compare mode, toggle output on match 0011 = Reserved 0100 = Capture mode, every falling edge 0101 = Capture mode, every rising edge 0110 = Capture mode, every 4th rising edge 0111 = Capture mode, every 16th rising edge 1000 = Compare mode, initialize CCP1 pin low, set output on compare match (set CCP1IF) 1001 = Compare mode, initialize CCP1 pin high, clear output on compare match (set CCP1IF) 1010 = Compare mode, generate software interrupt only, CCP1 pin reverts to I/O state 1011 = Compare mode, trigger special event (ECCP resets TMR1 or TMR3, start A/D conversion, sets CC1IF bit) 1100 = PWM mode; P1A, P1C active-high; P1B, P1D active-high 1101 = PWM mode; P1A, P1C active-high; P1B, P1D active-low 1110 = PWM mode; P1A, P1C active-low; P1B, P1D active-high 1111 = PWM mode; P1A, P1C active-low; P1B, P1D active-low

# PIC18F1XK50/PIC18LF1XK50

---

---

In addition to the expanded range of modes available through the CCP1CON register and ECCP1AS register, the ECCP module has two additional registers associated with Enhanced PWM operation and auto-shutdown features. They are:

- PWM1CON (Dead-band delay)
- PSTRCON (output steering)

## 14.1 ECCP Outputs and Configuration

The enhanced CCP module may have up to four PWM outputs, depending on the selected operating mode. These outputs, designated P1A through P1D, are multiplexed with I/O pins on PORTD. The outputs that are active depend on the CCP operating mode selected. The pin assignments are summarized in Table 14-2.

To configure the I/O pins as PWM outputs, the proper PWM mode must be selected by setting the P1M<1:0> and CCP1M<3:0> bits. The appropriate TRISD direction bits for the port pins must also be set as outputs.

### 14.1.1 CCP MODULE AND TIMER RESOURCES

The CCP modules utilize Timers 1, 2 or 3, depending on the mode selected. Timer1 and Timer3 are available to modules in Capture or Compare modes, while Timer2 is available for modules in PWM mode.

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

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

The assignment of a particular timer to a module is determined by the Timer-to-CCP enable bits in the T3CON register (Register 13-1). The interactions between the two modules are summarized in Figure 14-1. In Asynchronous Counter mode, the capture operation will not work reliably.

## 14.2 Capture Mode

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

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

The event is selected by the mode select bits, CCP1M<3:0> of the CCP1CON register. When a capture is made, the interrupt request flag bit, CCP1IF, is set; it must be cleared by software. If another capture occurs before the value in register CCPR1 is read, the old captured value is overwritten by the new captured value.

### 14.2.1 CCP PIN CONFIGURATION

In Capture mode, the appropriate CCP1 pin should be configured as an input by setting the corresponding TRIS direction bit.

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

### 14.2.2 TIMER1/TIMER3 MODE SELECTION

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

be used with each CCP module is selected in the T3CON register (see **Section 14.1.1 “CCP Module and Timer Resources”**).

### 14.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCP1IE interrupt enable bit clear to avoid false interrupts. The interrupt flag bit, CCP1IF, should also be cleared following any such change in operating mode.

### 14.2.4 CCP PRESCALER

There are four prescaler settings in Capture mode; they are specified as part of the operating mode selected by the mode select bits (CCP1M<3:0>). Whenever the CCP module is turned off or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared; therefore, the first capture may be from a non-zero prescaler. Example 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.

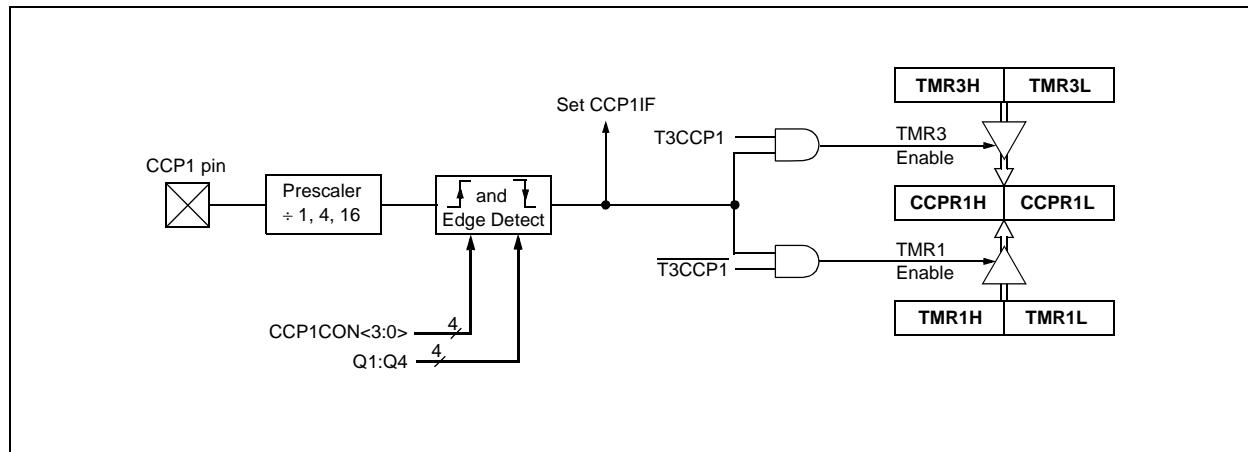
### EXAMPLE 14-1: CHANGING BETWEEN CAPTURE PRESCALERS

```

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

```

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



## 14.3 Compare Mode

In Compare mode, the 16-bit CCP1 register value is constantly compared against either the TMR1 or TMR3 register pair value. When a match occurs, the CCP1 pin can be:

- driven high
- driven low
- toggled (high-to-low or low-to-high)
- remain unchanged (that is, reflects the state of the I/O latch)

The action on the pin is based on the value of the mode select bits (CCP1M<3:0>). At the same time, the interrupt flag bit, CCP1IF, is set.

### 14.3.1 CCP PIN CONFIGURATION

The user must configure the CCP1 pin as an output by clearing the appropriate TRIS bit.

**Note:** Clearing the CCP1CON register will force the CCP1 compare output latch (depending on device configuration) to the default low level. This is not the PORTC I/O data latch.

### 14.3.2 TIMER1/TIMER3 MODE SELECTION

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

### 14.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCP1M<3:0> = 1010), the CCP1 pin is not affected. Only the CCP1IF interrupt flag is affected.

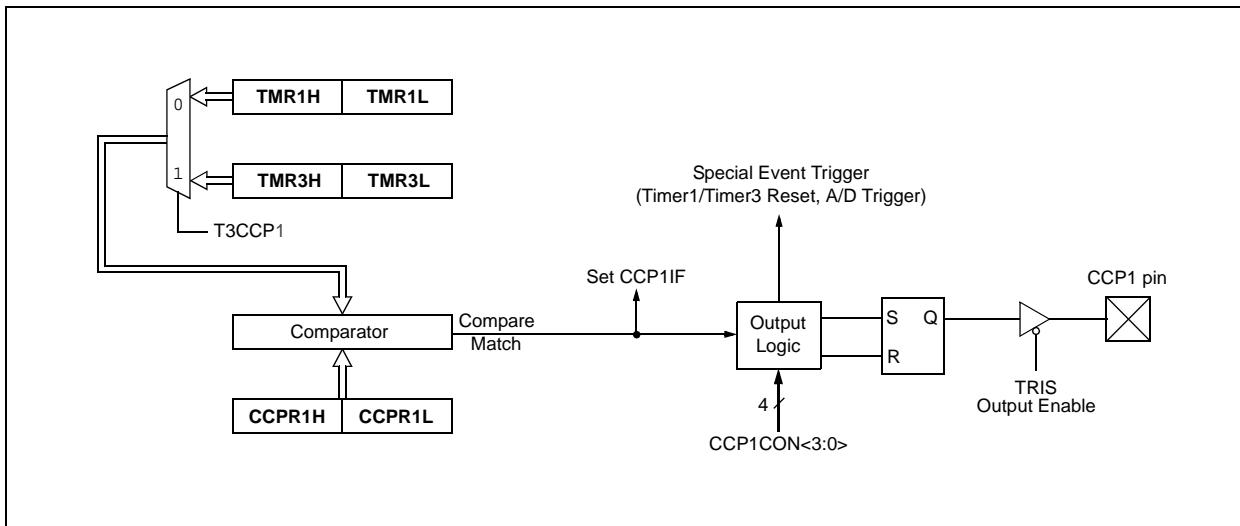
### 14.3.4 SPECIAL EVENT TRIGGER

The CCP module is equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode (CCP1M<3:0> = 1011).

The Special Event Trigger resets the timer register pair for whichever timer resource is currently assigned as the module's time base. This allows the CCP1 registers to serve as a programmable period register for either timer.

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

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



## 14.4 PWM (Enhanced Mode)

The Enhanced PWM Mode can generate a PWM signal on up to four different output pins with up to 10-bits of resolution. It can do this through four different PWM output modes:

- Single PWM
  - Half-Bridge PWM
  - Full-Bridge PWM, Forward mode
  - Full-Bridge PWM, Reverse mode

To select an Enhanced PWM mode, the P1M bits of the CCP1CON register must be set appropriately.

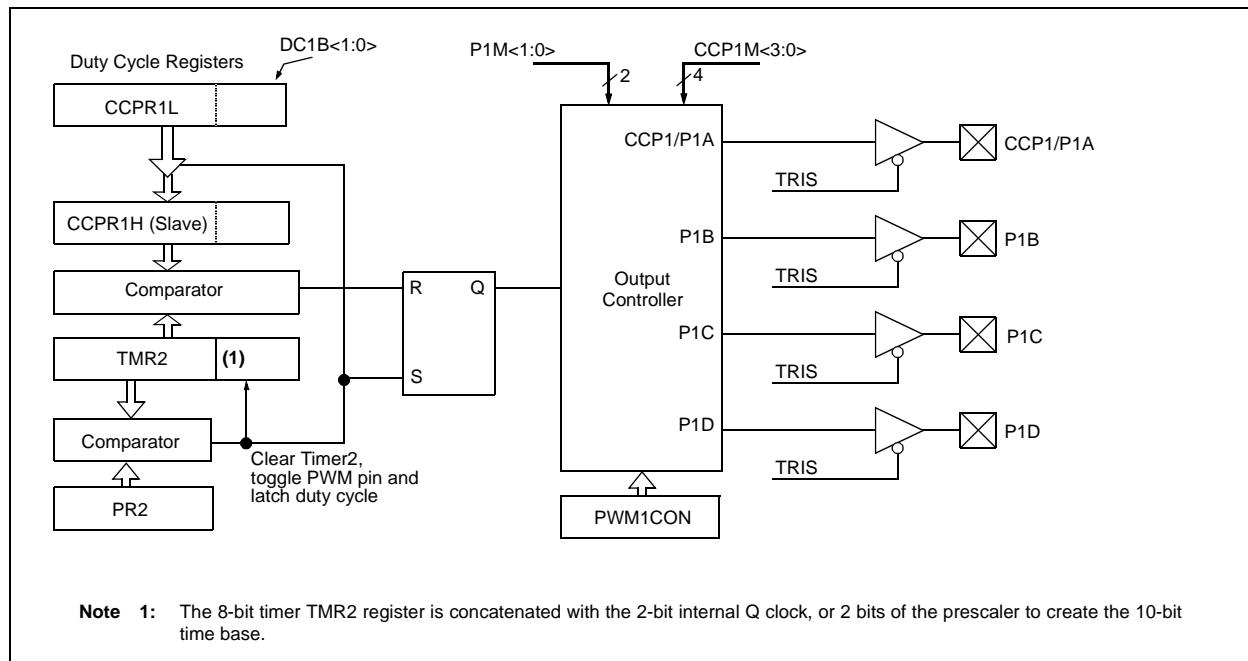
The PWM outputs are multiplexed with I/O pins and are designated P1A, P1B, P1C and P1D. The polarity of the PWM pins is configurable and is selected by setting the CCP1M bits in the CCP1CON register appropriately.

Table 14-1 shows the pin assignments for each Enhanced PWM mode.

Figure 14-3 shows an example of a simplified block diagram of the Enhanced PWM module.

**Note:** To prevent the generation of an incomplete waveform when the PWM is first enabled, the ECCP module waits until the start of a new PWM period before generating a PWM signal.

**FIGURE 14-3: EXAMPLE SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODE**



**Note 1:** The TRIS register value for each PWM output must be configured appropriately.

2: Any pin not used by an Enhanced PWM mode is available for alternate pin functions.

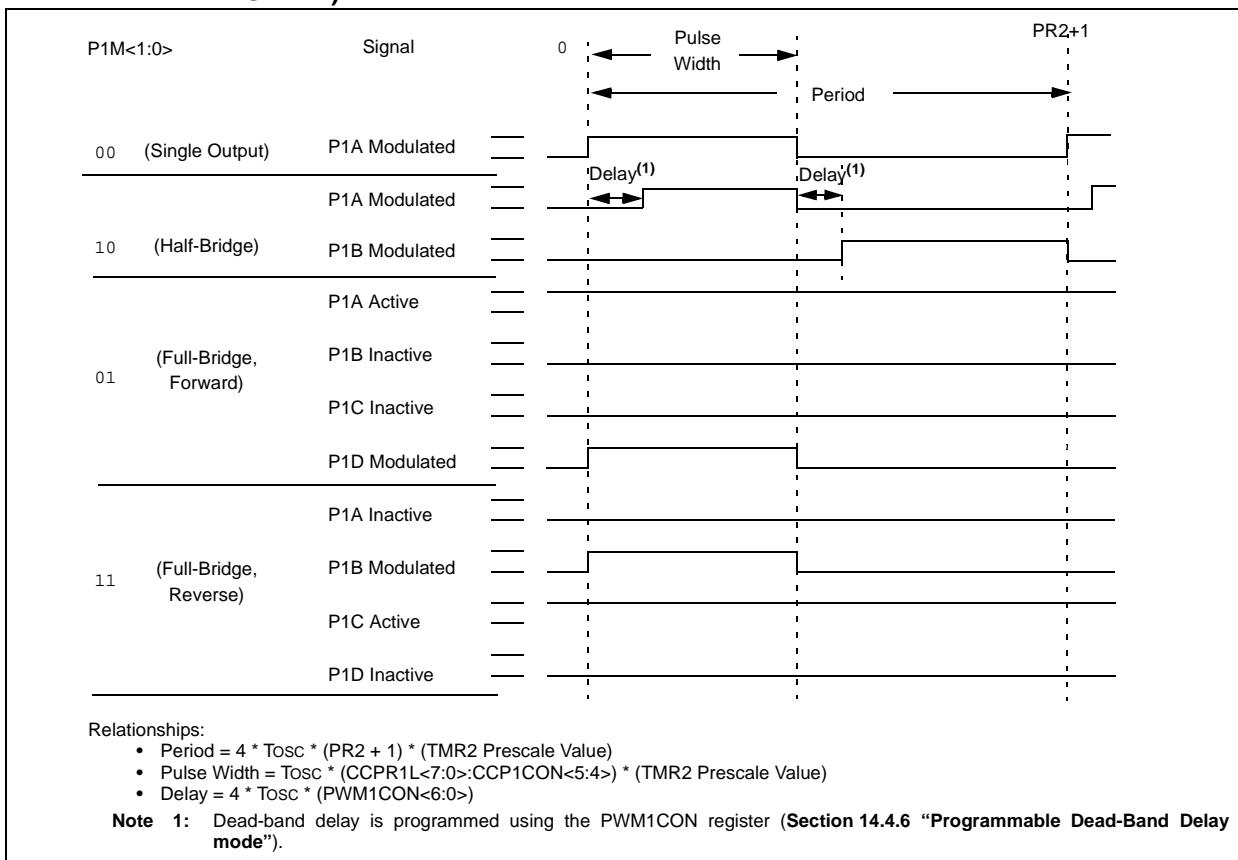
TABLE 14-2: EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES

ECCP Mode	P1M<1:0>	CCP1/P1A	P1B	P1C	P1D
Single	00	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>	Yes <sup>(1)</sup>
Half-Bridge	10	Yes	Yes	No	No
Full-Bridge, Forward	01	Yes	Yes	Yes	Yes
Full-Bridge, Reverse	11	Yes	Yes	Yes	Yes

**Note 1:** Outputs are enabled by pulse steering in Single mode. See Register 14-4.

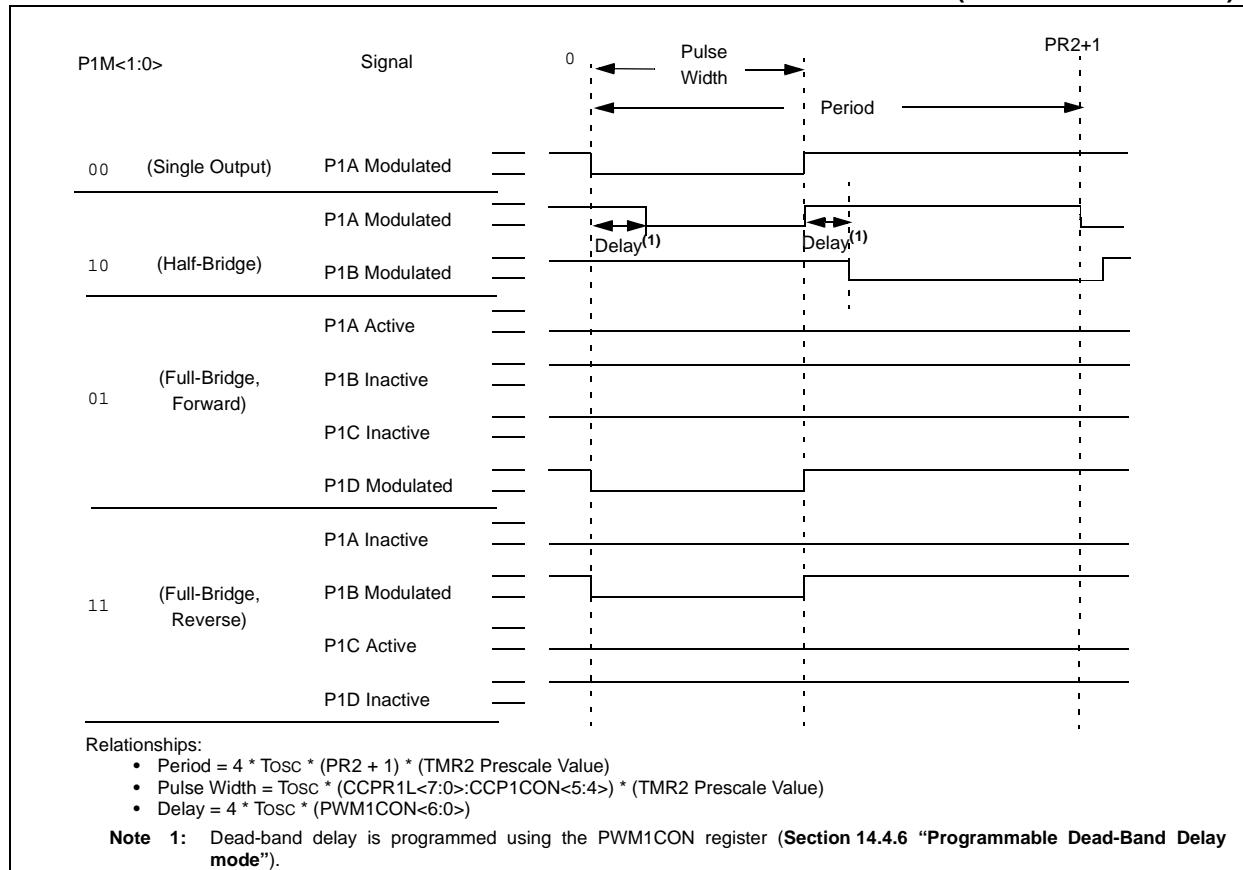
# PIC18F1XK50/PIC18LF1XK50

**FIGURE 14-4: EXAMPLE PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)**



# PIC18F1XK50/PIC18LF1XK50

**FIGURE 14-5: EXAMPLE ENHANCED PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)**



# PIC18F1XK50/PIC18LF1XK50

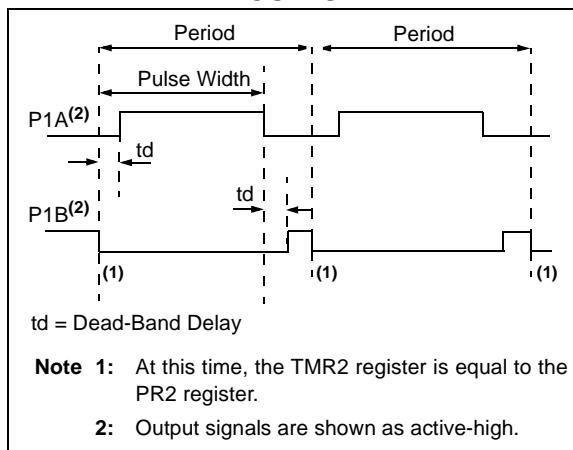
## 14.4.1 HALF-BRIDGE MODE

In Half-Bridge mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the CCP1/P1A pin, while the complementary PWM output signal is output on the P1B pin (see Figure 14-6). This mode can be used for Half-Bridge applications, as shown in Figure 14-7, or for Full-Bridge applications, where four power switches are being modulated with two PWM signals.

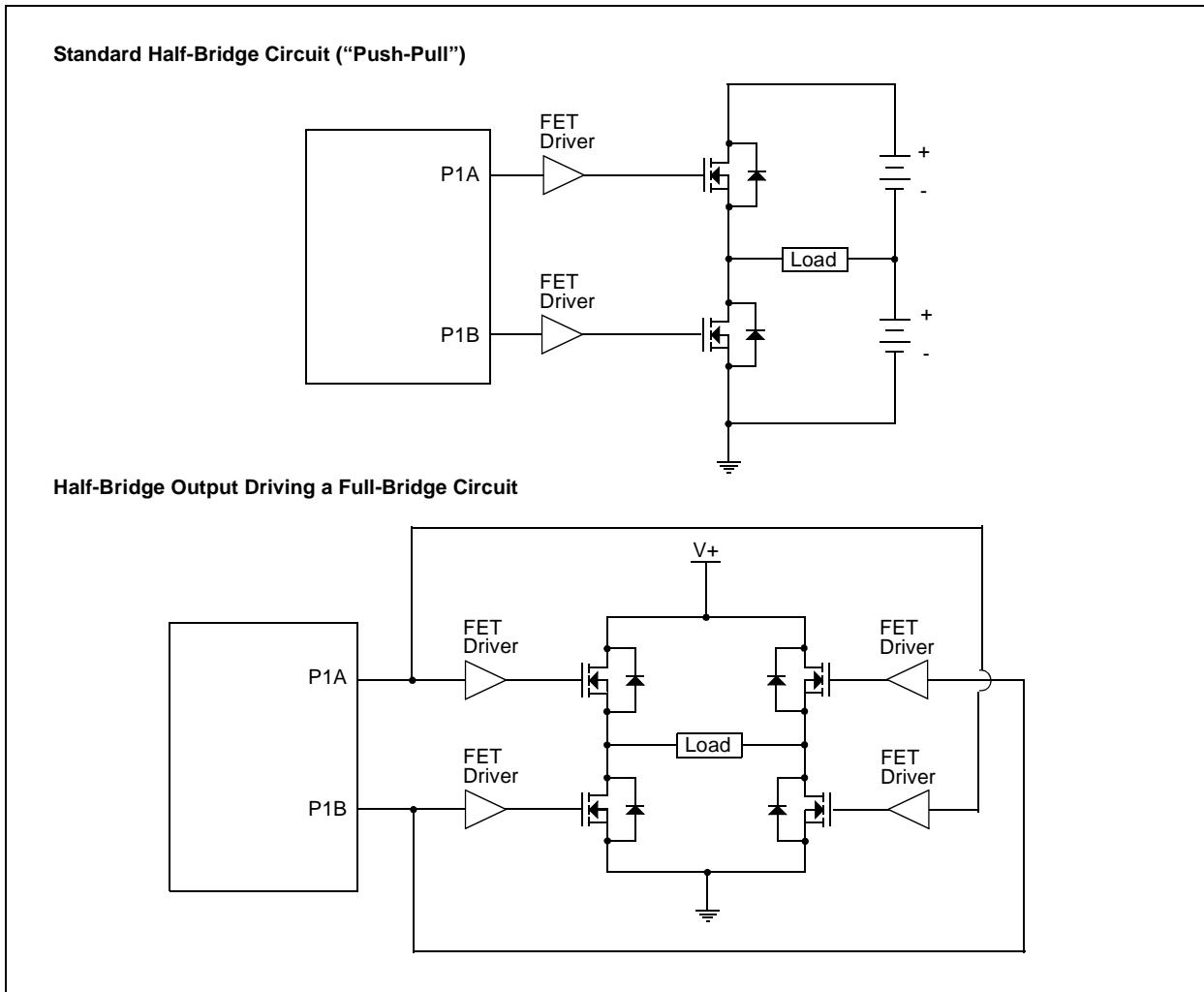
In Half-Bridge mode, the programmable dead-band delay can be used to prevent shoot-through current in Half-Bridge power devices. The value of the PDC<6:0> bits of the PWM1CON register sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See **Section 14.4.6 “Programmable Dead-Band Delay mode”** for more details of the dead-band delay operations.

Since the P1A and P1B outputs are multiplexed with the PORT data latches, the associated TRIS bits must be cleared to configure P1A and P1B as outputs.

**FIGURE 14-6: EXAMPLE OF HALF-BRIDGE PWM OUTPUT**



**FIGURE 14-7: EXAMPLE OF HALF-BRIDGE APPLICATIONS**



## 14.4.2 FULL-BRIDGE MODE

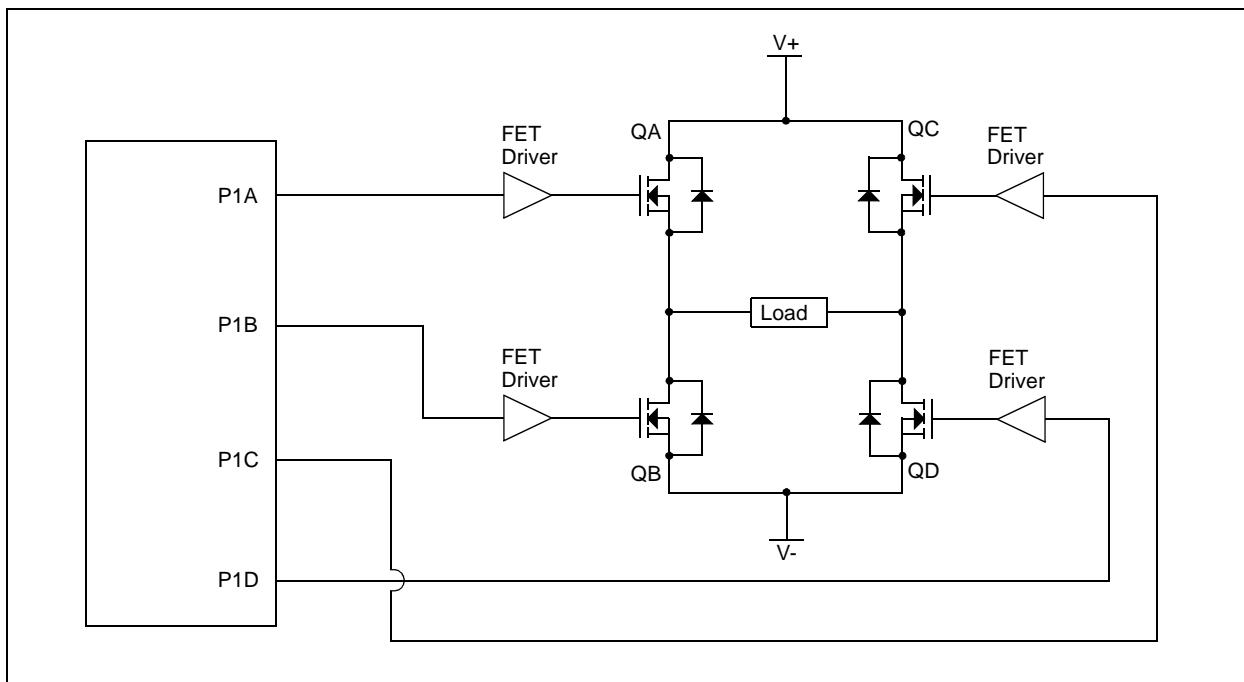
In Full-Bridge mode, all four pins are used as outputs. An example of Full-Bridge application is shown in Figure 14-8.

In the Forward mode, pin CCP1/P1A is driven to its active state, pin P1D is modulated, while P1B and P1C will be driven to their inactive state as shown in Figure 14-9.

In the Reverse mode, P1C is driven to its active state, pin P1B is modulated, while P1A and P1D will be driven to their inactive state as shown Figure 14-9.

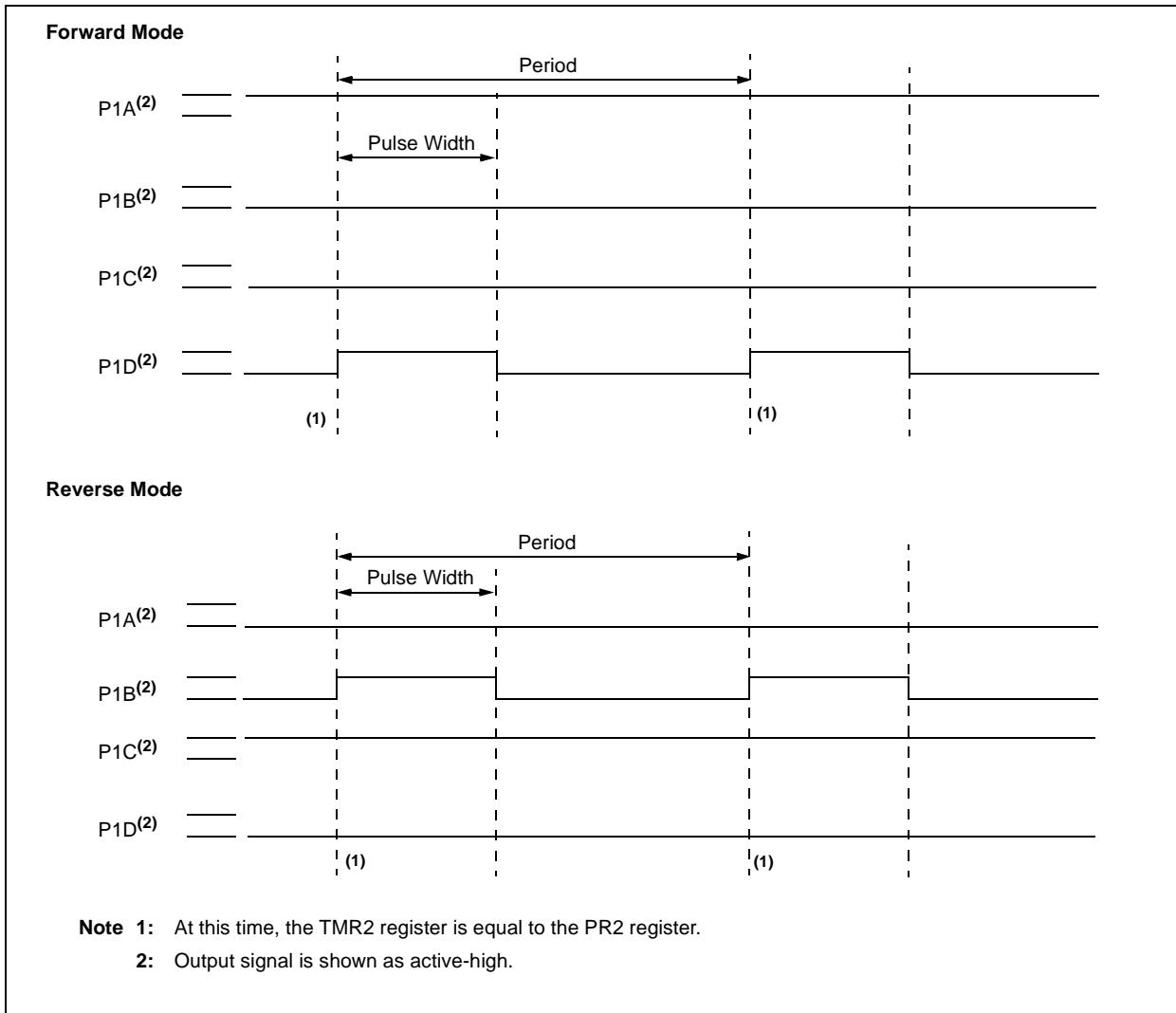
P1A, P1B, P1C and P1D outputs are multiplexed with the PORT data latches. The associated TRIS bits must be cleared to configure the P1A, P1B, P1C and P1D pins as outputs.

**FIGURE 14-8: EXAMPLE OF FULL-BRIDGE APPLICATION**



# PIC18F1XK50/PIC18LF1XK50

FIGURE 14-9: EXAMPLE OF FULL-BRIDGE PWM OUTPUT



### 14.4.2.1 Direction Change in Full-Bridge Mode

In the Full-Bridge mode, the P1M1 bit in the CCP1CON register allows users to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will change to the new direction on the next PWM cycle.

A direction change is initiated in software by changing the P1M1 bit of the CCP1CON register. The following sequence occurs prior to the end of the current PWM period:

- The modulated outputs (P1B and P1D) are placed in their inactive state.
- The associated unmodulated outputs (P1A and P1C) are switched to drive in the opposite direction.
- PWM modulation resumes at the beginning of the next period.

See Figure 14-10 for an illustration of this sequence.

The Full-Bridge mode does not provide dead-band delay. As one output is modulated at a time, dead-band delay is generally not required. There is a situation where dead-band delay is required. This situation occurs when both of the following conditions are true:

1. The direction of the PWM output changes when the duty cycle of the output is at or near 100%.
2. The turn off time of the power switch, including the power device and driver circuit, is greater than the turn on time.

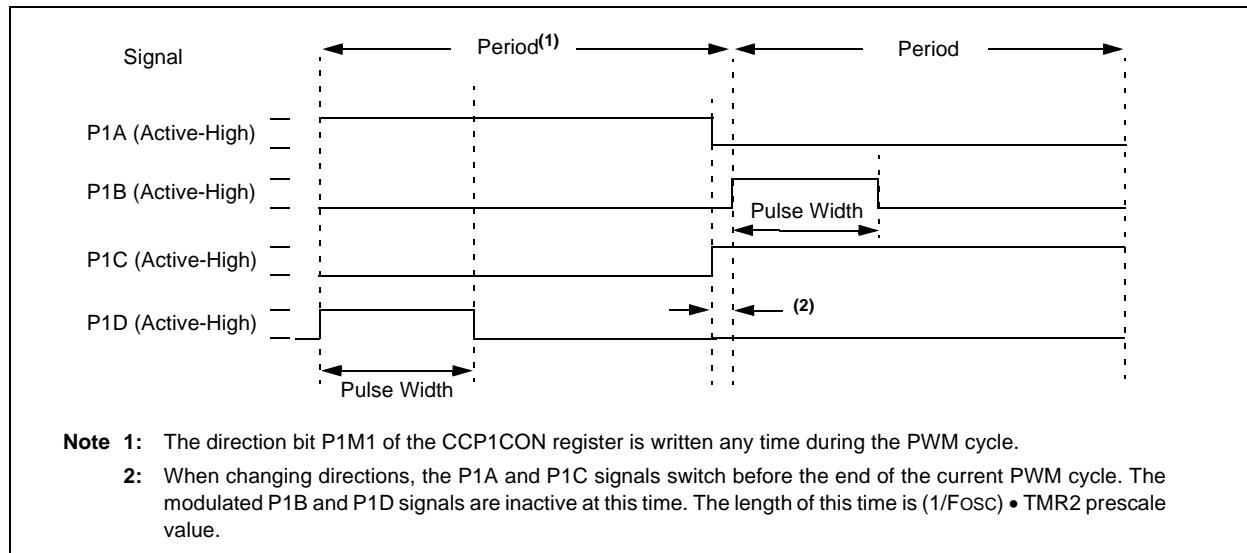
Figure 14-11 shows an example of the PWM direction changing from forward to reverse, at a near 100% duty cycle. In this example, at time  $t_1$ , the output P1A and P1D become inactive, while output P1C becomes active. Since the turn off time of the power devices is longer than the turn on time, a shoot-through current will flow through power devices QC and QD (see Figure 14-8) for the duration of ' $t$ '. The same phenomenon will occur to power devices QA and QB for PWM direction change from reverse to forward.

If changing PWM direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

1. Reduce PWM duty cycle for one PWM period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

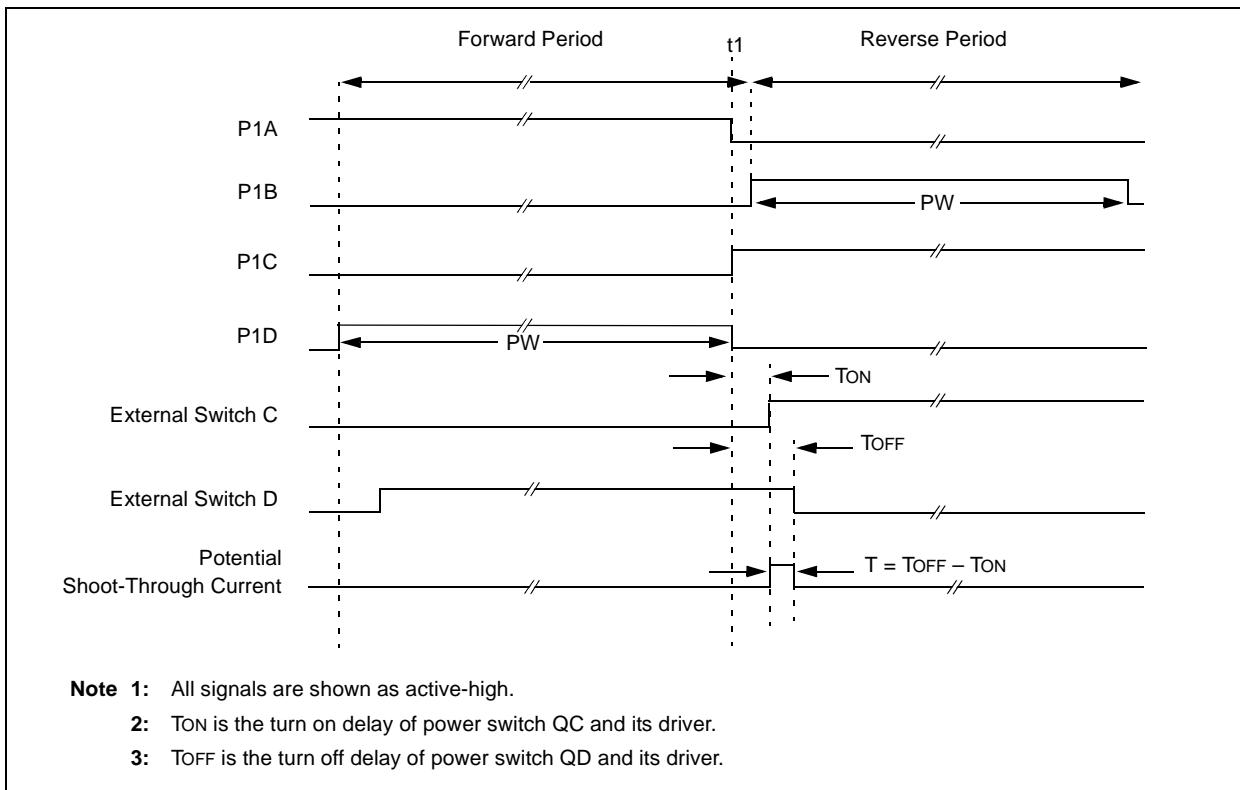
Other options to prevent shoot-through current may exist.

**FIGURE 14-10: EXAMPLE OF PWM DIRECTION CHANGE**



# PIC18F1XK50/PIC18LF1XK50

FIGURE 14-11: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



## 14.4.3 START-UP CONSIDERATIONS

When any PWM mode is used, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins.

**Note:** When the microcontroller is released from Reset, all of the I/O pins are in the high-impedance state. The external circuits must keep the power switch devices in the Off state until the microcontroller drives the I/O pins with the proper signal levels or activates the PWM output(s).

The CCP1M<1:0> bits of the CCP1CON register allow the user to choose whether the PWM output signals are active-high or active-low for each pair of PWM output pins (P1A/P1C and P1B/P1D). The PWM output polarities must be selected before the PWM pin output drivers are enabled. Changing the polarity configuration while the PWM pin output drivers are enable is not recommended since it may result in damage to the application circuits.

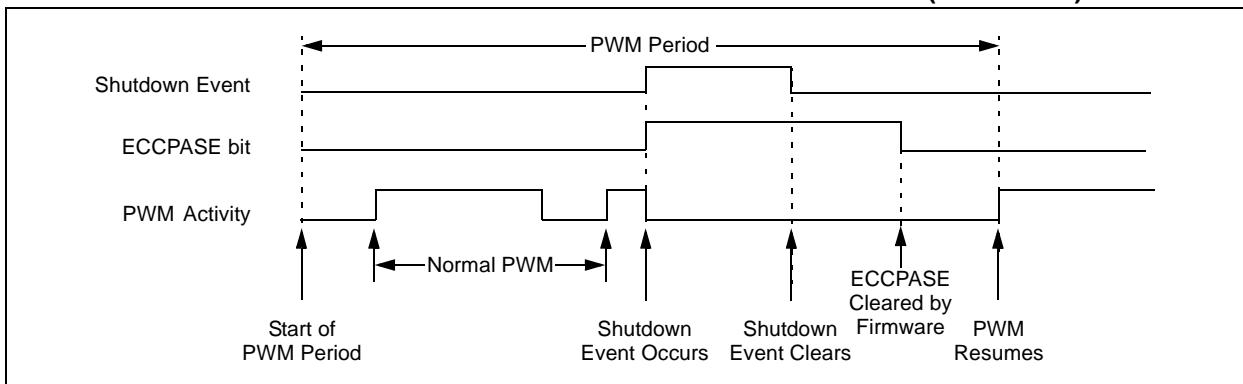
The P1A, P1B, P1C and P1D output latches may not be in the proper states when the PWM module is initialized. Enabling the PWM pin output drivers at the same time as the Enhanced PWM modes may cause damage to the application circuit. The Enhanced PWM modes must be enabled in the proper Output mode and complete a full PWM cycle before enabling the PWM pin output drivers. The completion of a full PWM cycle is indicated by the TMR2IF bit of the PIR1 register being set as the second PWM period begins.



# PIC18F1XK50/PIC18LF1XK50

- Note 1:** The auto-shutdown condition is a level-based signal, not an edge-based signal. As long as the level is present, the auto-shutdown will persist.
- 2:** Writing to the ECCPASE bit is disabled while an auto-shutdown condition persists.
- 3:** Once the auto-shutdown condition has been removed and the PWM restarted (either through firmware or auto-restart) the PWM signal will always restart at the beginning of the next PWM period.

**FIGURE 14-12: PWM AUTO-SHUTDOWN WITH FIRMWARE RESTART (PRSEN = 0)**

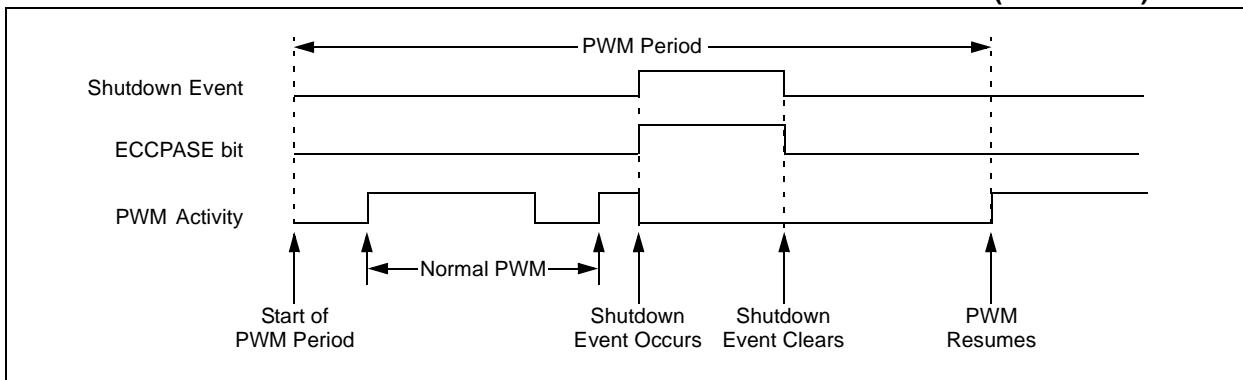


## 14.4.5 AUTO-RESTART MODE

The Enhanced PWM can be configured to automatically restart the PWM signal once the auto-shutdown condition has been removed. Auto-restart is enabled by setting the PRSEN bit in the PWM1CON register.

If auto-restart is enabled, the ECCPASE bit will remain set as long as the auto-shutdown condition is active. When the auto-shutdown condition is removed, the ECCPASE bit will be cleared via hardware and normal operation will resume.

**FIGURE 14-13: PWM AUTO-SHUTDOWN WITH AUTO-RESTART ENABLED (PRSEN = 1)**

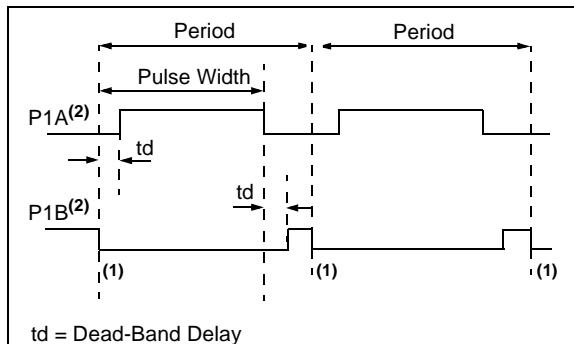


## 14.4.6 PROGRAMMABLE DEAD-BAND DELAY MODE

In Half-Bridge applications where all power switches are modulated at the PWM frequency, the power switches normally require more time to turn off than to turn on. If both the upper and lower power switches are switched at the same time (one turned on, and the other turned off), both switches may be on for a short period of time until one switch completely turns off. During this brief interval, a very high current (shoot-through current) will flow through both power switches, shorting the bridge supply. To avoid this potentially destructive shoot-through current from flowing during switching, turning on either of the power switches is normally delayed to allow the other switch to completely turn off.

In Half-Bridge mode, a digitally programmable dead-band delay is available to avoid shoot-through current from destroying the bridge power switches. The delay occurs at the signal transition from the non-active state to the active state. See Figure 14-14 for illustration. The lower seven bits of the associated PWM1CON register (Register 14-3) sets the delay period in terms of microcontroller instruction cycles (TCY or 4 Tosc).

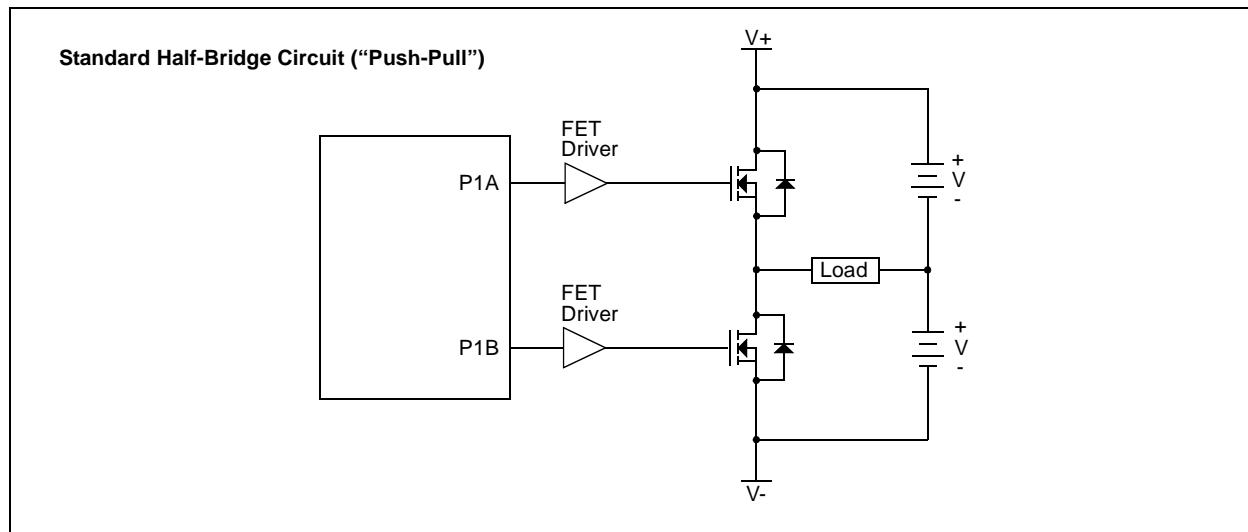
**FIGURE 14-14: EXAMPLE OF HALF-BRIDGE PWM OUTPUT**



**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

**2:** Output signals are shown as active-high.

**FIGURE 14-15: EXAMPLE OF HALF-BRIDGE APPLICATIONS**



# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 14-3: PWM1CON: ENHANCED PWM CONTROL REGISTER

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| PRSEN | PDC6  | PDC5  | PDC4  | PDC3  | PDC2  | PDC1  | PDC0  |
| bit 7 | bit 0 |       |       |       |       |       |       |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **PRSEN:** PWM Restart Enable bit  
1 = Upon auto-shutdown, the ECCPASE bit clears automatically once the shutdown event goes away; the PWM restarts automatically  
0 = Upon auto-shutdown, ECCPASE must be cleared by software to restart the PWM
- bit 6-0 **PDC<6:0>:** PWM Delay Count bits  
PDCn = Number of Fosc/4 (4 \* Tosc) cycles between the scheduled time when a PWM signal **should** transition active and the **actual** time it transitions active

## 14.4.7 PULSE STEERING MODE

In Single Output mode, pulse steering allows any of the PWM pins to be the modulated signal. Additionally, the same PWM signal can be simultaneously available on multiple pins.

Once the Single Output mode is selected ( $CCP1M<3:2> = 11$  and  $P1M<1:0> = 00$  of the CCP1CON register), the user firmware can bring out the same PWM signal to one, two, three or four output pins by setting the appropriate  $STR<D:A>$  bits of the PSTRCON register, as shown in Table 14-2.

**Note:** The associated TRIS bits must be set to output ('0') to enable the pin output driver in order to see the PWM signal on the pin.

While the PWM Steering mode is active, CCP1M<1:0> bits of the CCP1CON register select the PWM output polarity for the P1<D:A> pins.

The PWM auto-shutdown operation also applies to PWM Steering mode as described in **Section 14.4.4 “Enhanced PWM Auto-shutdown mode”**. An auto-shutdown event will only affect pins that have PWM outputs enabled.

## REGISTER 14-4: PSTRCON: PULSE STEERING CONTROL REGISTER<sup>(1)</sup>

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
—	—	—	STRSYNC	STRD	STRC	STRB	STRA
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5

**Unimplemented:** Read as '0'

bit 4

**STRSYNC:** Steering Sync bit

1 = Output steering update occurs on next PWM period

0 = Output steering update occurs at the beginning of the instruction cycle boundary

bit 3

**STRD:** Steering Enable bit D

1 = P1D pin has the PWM waveform with polarity control from CCP1M<1:0>

0 = P1D pin is assigned to port pin

bit 2

**STRC:** Steering Enable bit C

1 = P1C pin has the PWM waveform with polarity control from CCP1M<1:0>

0 = P1C pin is assigned to port pin

bit 1

**STRB:** Steering Enable bit B

1 = P1B pin has the PWM waveform with polarity control from CCP1M<1:0>

0 = P1B pin is assigned to port pin

bit 0

**STRA:** Steering Enable bit A

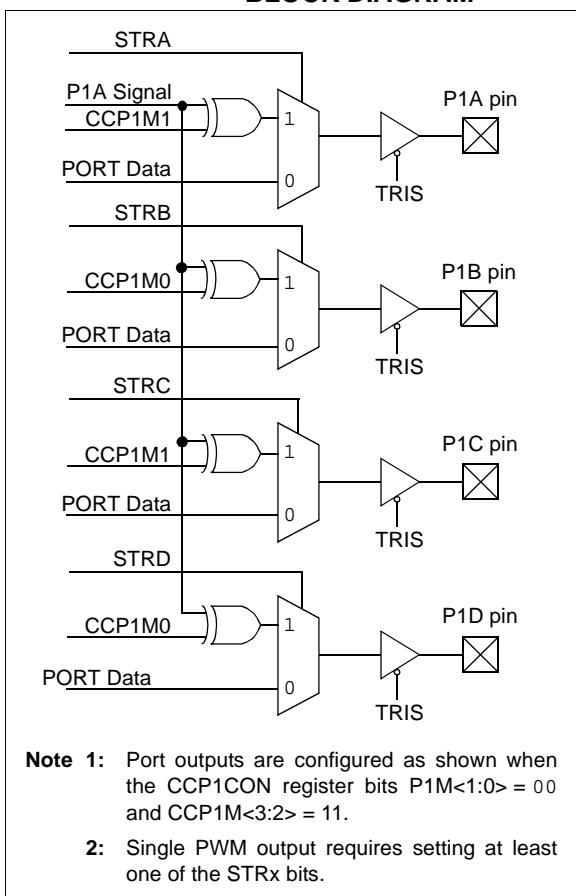
1 = P1A pin has the PWM waveform with polarity control from CCP1M<1:0>

0 = P1A pin is assigned to port pin

**Note 1:** The PWM Steering mode is available only when the CCP1CON register bits CCP1M<3:2> = 11 and P1M<1:0> = 00.

# PIC18F1XK50/PIC18LF1XK50

**FIGURE 14-16: SIMPLIFIED STEERING BLOCK DIAGRAM**



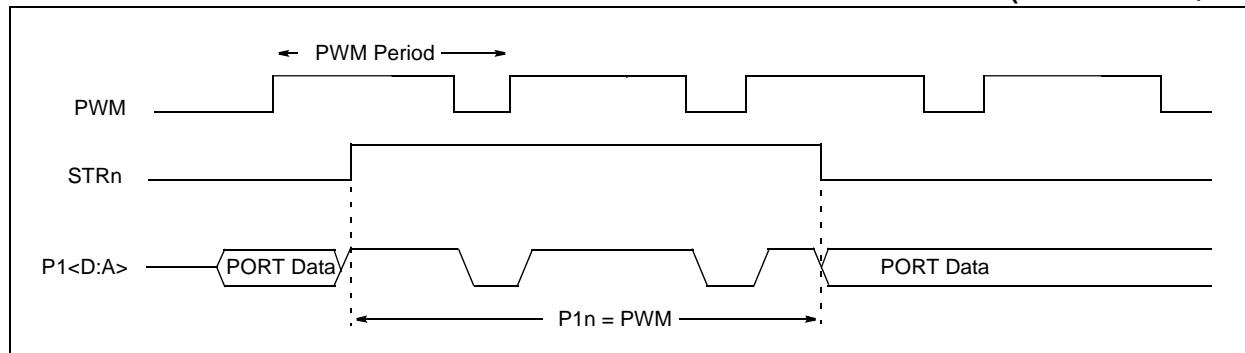
### 14.4.7.1 Steering Synchronization

The STRSYNC bit of the PSTRCON register gives the user two selections of when the steering event will happen. When the STRSYNC bit is '0', the steering event will happen at the end of the instruction that writes to the PSTRCON register. In this case, the output signal at the P1<D:A> pins may be an incomplete PWM waveform. This operation is useful when the user firmware needs to immediately remove a PWM signal from the pin.

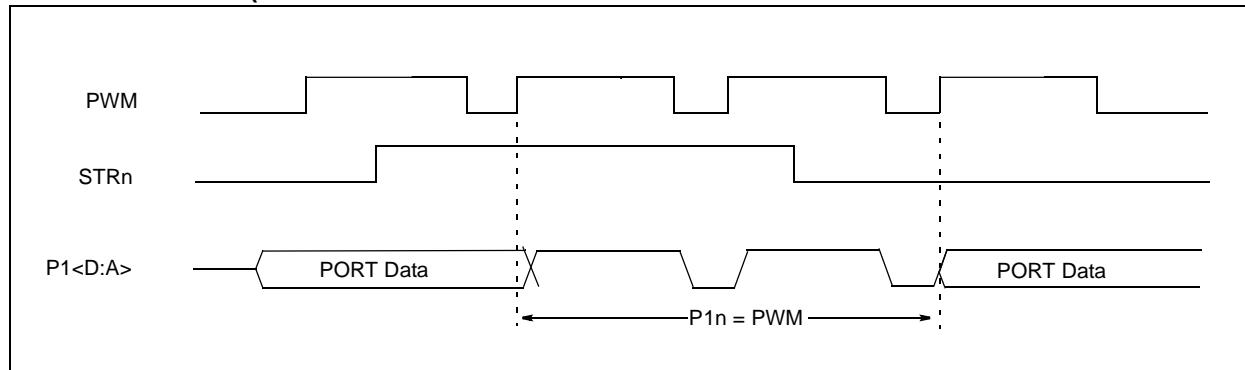
When the STRSYNC bit is '1', the effective steering update will happen at the beginning of the next PWM period. In this case, steering on/off the PWM output will always produce a complete PWM waveform.

Figures 14-17 and 14-18 illustrate the timing diagrams of the PWM steering depending on the STRSYNC setting.

**FIGURE 14-17: EXAMPLE OF STEERING EVENT AT END OF INSTRUCTION (STRSYNC = 0 )**



**FIGURE 14-18: EXAMPLE OF STEERING EVENT AT BEGINNING OF INSTRUCTION (STRSYNC = 1 )**



## 14.4.8 OPERATION IN POWER-MANAGED MODES

In Sleep mode, all clock sources are disabled. Timer2 will not increment and the state of the module will not change. If the ECCP pin is driving a value, it will continue to drive that value. When the device wakes up, it will continue from this state. If Two-Speed Start-ups are enabled, the initial start-up frequency from HFINTOSC and the postscaler may not be stable immediately.

In PRI\_IDLE mode, the primary clock will continue to clock the ECCP module without change. In all other power-managed modes, the selected power-managed mode clock will clock Timer2. Other power-managed mode clocks will most likely be different than the primary clock frequency.

### 14.4.8.1 Operation with Fail-Safe Clock Monitor

If the Fail-Safe Clock Monitor is enabled, a clock failure will force the device into the RC\_RUN Power-Managed mode and the OSCFIF bit of the PIR2 register will be set. The ECCP will then be clocked from the internal oscillator clock source, which may have a different clock frequency than the primary clock.

See the previous section for additional details.

### 14.4.9 EFFECTS OF A RESET

Both Power-on Reset and subsequent Resets will force all ports to Input mode and the CCP registers to their Reset states.

This forces the enhanced CCP module to reset to a state compatible with the standard CCP module.

# PIC18F1XK50/PIC18LF1XK50

TABLE 14-3: REGISTERS ASSOCIATED WITH ECCP1 MODULE AND TIMER1 TO TIMER3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
RCON	IPEN	SBOREN	—	RI	TO	PD	POR	BOR	278
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	282
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	282
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	282
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282
TMR1L	Timer1 Register, Low Byte								280
TMR1H	Timer1 Register, High Byte								280
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	—	TMR1CS	TMR1ON	280
TMR2	Timer2 Register								280
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	280
PR2	Timer2 Period Register								280
TMR3L	Timer3 Register, Low Byte								281
TMR3H	Timer3 Register, High Byte								281
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	—	TMR3CS	TMR3ON	281
CCPR1L	Capture/Compare/PWM Register 1, Low Byte								281
CCPR1H	Capture/Compare/PWM Register 1, High Byte								281
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	281
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	281
PWM1CON	PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0	281

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during ECCP operation.

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **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 SPI Mode

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

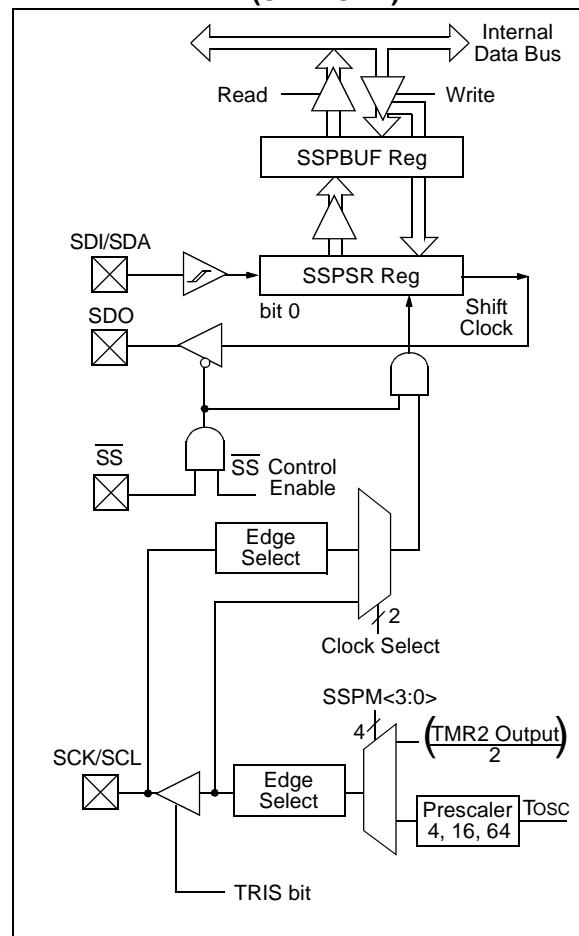
- Serial Data Out – SDO
- Serial Data In – SDI
- Serial Clock – SCK

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

- Slave Select – SS

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

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



# PIC18F1XK50/PIC18LF1XK50

## 15.2.1 REGISTERS

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

- SSPCON1 – Control Register
- SSPSTAT – STATUS register
- SSPBUF – Serial Receive/Transmit Buffer
- SSPSR – Shift Register (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 and out. SSPBUF provides indirect access to the SSPSR register. SSPBUF is the buffer register to which data bytes are written, and from which data bytes are read.

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							

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

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

bit 7	<b>SMP:</b> Sample bit  <u>SPI Master mode:</u> 1 = Input data sampled at end of data output time 0 = Input data sampled at middle of data output time  <u>SPI Slave mode:</u> SMP must be cleared when SPI is used in Slave mode.
bit 6	<b>CKE:</b> SPI Clock Select bit <sup>(1)</sup> 1 = Transmit occurs on transition from active to Idle clock state 0 = Transmit occurs on transition from Idle to active clock state
bit 5	<b>D/A:</b> Data/Address bit Used in I <sup>2</sup> C mode only.
bit 4	<b>P:</b> Stop bit Used in I <sup>2</sup> C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.
bit 3	<b>S:</b> Start bit Used in I <sup>2</sup> C mode only.
bit 2	<b>R/W:</b> Read/Write Information bit Used in I <sup>2</sup> C mode only.
bit 1	<b>UA:</b> Update Address bit Used in I <sup>2</sup> C mode only.
bit 0	<b>BF:</b> Buffer Full Status bit (Receive mode only) 1 = Receive complete, SSPBUF is full 0 = Receive not complete, SSPBUF is empty

**Note 1:** Polarity of clock state is set by the CKP bit of the SSPCON1 register.

## REGISTER 15-2: SSPCON1: MSSP CONTROL 1 REGISTER (SPI MODE)

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

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>WCOL:</b> Write Collision Detect bit (Transmit mode only) 1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared by software) 0 = No collision
bit 6	<b>SSPOV:</b> Receive Overflow Indicator bit <sup>(1)</sup> <b>SPI Slave mode:</b> 1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared by software). 0 = No overflow
bit 5	<b>SSPEN:</b> Synchronous Serial Port Enable bit <sup>(2)</sup> 1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins 0 = Disables serial port and configures these pins as I/O port pins
bit 4	<b>CKP:</b> Clock Polarity Select bit 1 = Idle state for clock is a high level 0 = Idle state for clock is a low level
bit 3-0	<b>SSPM&lt;3:0&gt;:</b> Synchronous Serial Port Mode Select bits <sup>(3)</sup> 0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin 0100 = SPI Slave mode, clock = SCK pin, SS pin control enabled 0011 = SPI Master mode, clock = TMR2 output/2 0010 = SPI Master mode, clock = Fosc/64 0001 = SPI Master mode, clock = Fosc/16 0000 = SPI Master mode, clock = Fosc/4

**Note 1:** In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

**2:** When enabled, these pins must be properly configured as input or output.

**3:** Bit combinations not specifically listed here are either reserved or implemented in I<sup>2</sup>C mode only.

# PIC18F1XK50/PIC18LF1XK50

---

## 15.2.2 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 of the SSPSTAT register, 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 of the SSPCON1 register, will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPBUF register to complete successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. The Buffer Full bit, BF of the SSPSTAT register, 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. 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

### 15.2.3 ENABLING SPI I/O

To enable the serial port, SSP Enable bit, SSPEN of the SSPCON1 register, must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize the SSPCON registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and SS pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI is automatically controlled by the SPI module
- SDO must have corresponding TRIS bit cleared
- SCK (Master mode) must have corresponding TRIS bit cleared
- SCK (Slave mode) must have corresponding TRIS bit set
- SS must have corresponding TRIS bit set

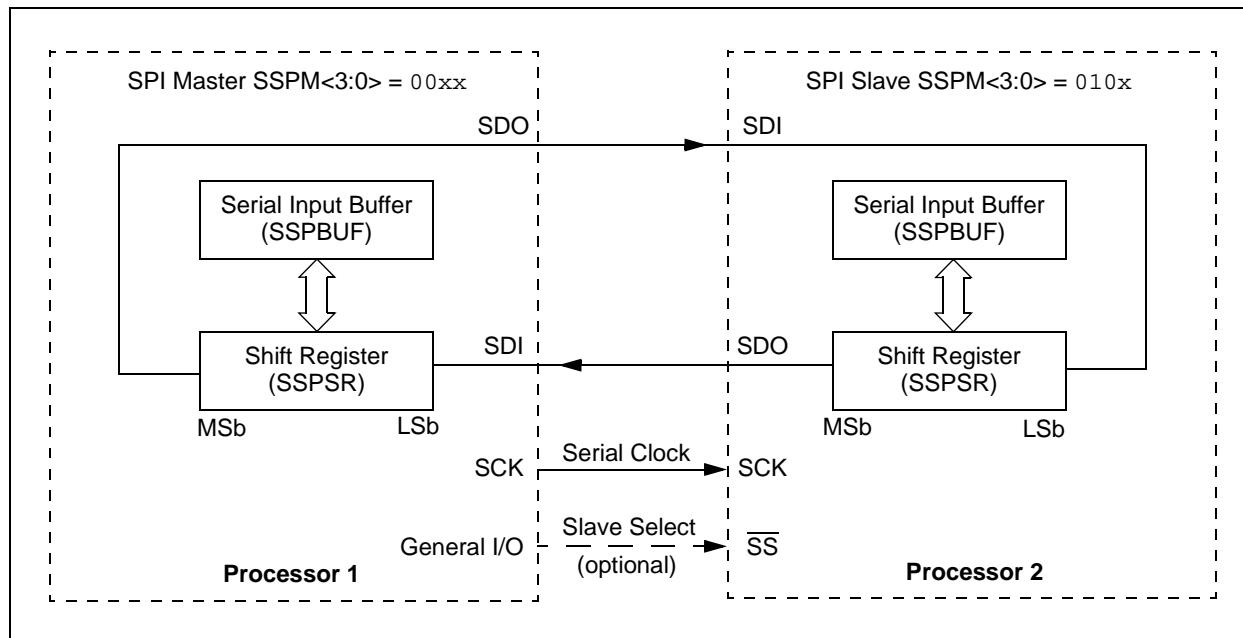
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

### 15.2.4 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: TYPICAL SPI MASTER/SLAVE CONNECTION**



# PIC18F1XK50/PIC18LF1XK50

## 15.2.5 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).

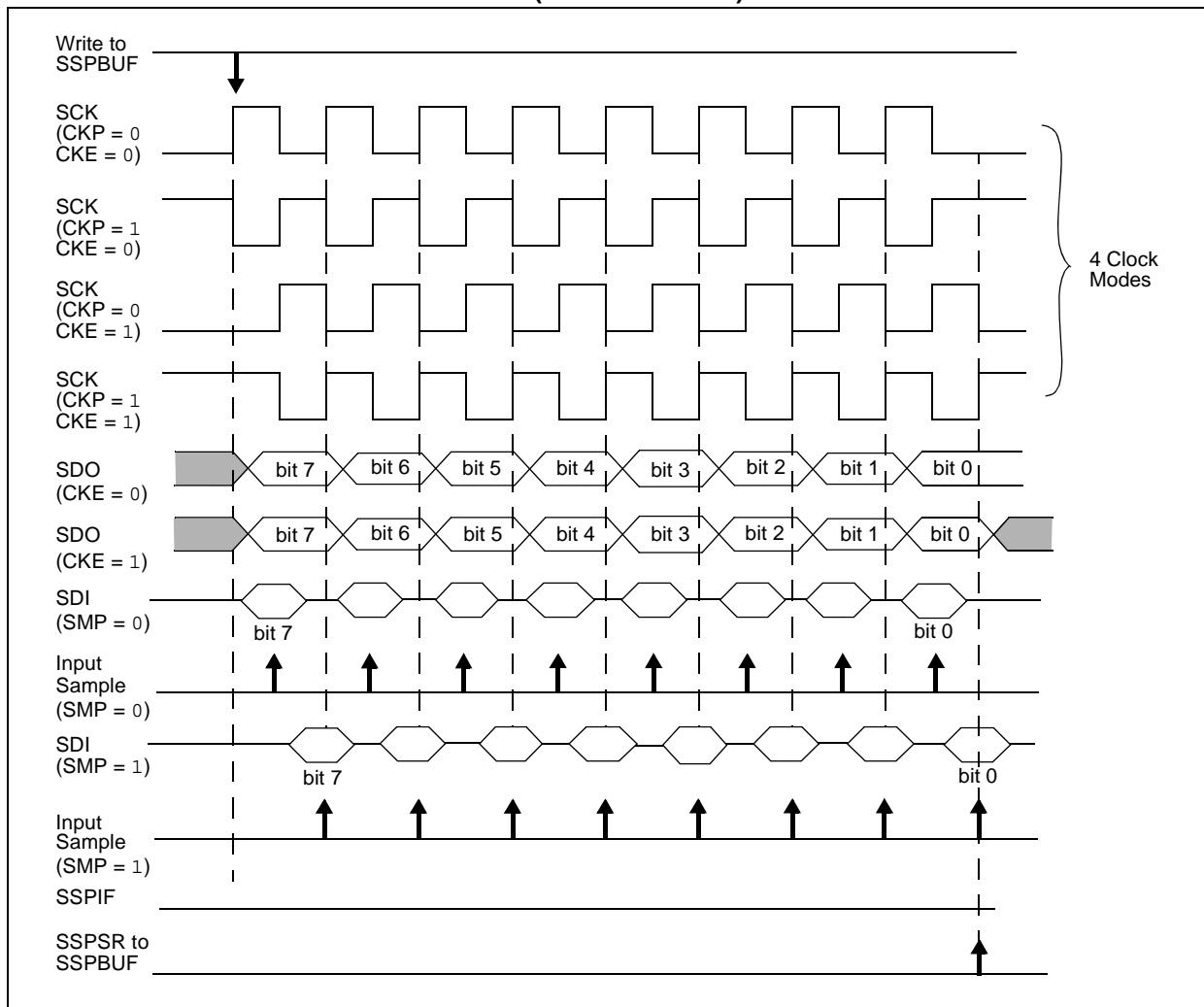
The clock polarity is selected by appropriately programming the CKP bit of the SSPCON1 register. This then, would give waveforms for SPI communication as shown in Figure 15-3, Figure 15-5 and Figure 15-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable 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 64 MHz) of 16.00 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)**



## 15.2.6 SLAVE MODE

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

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit of the SSPCON1 register.

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.

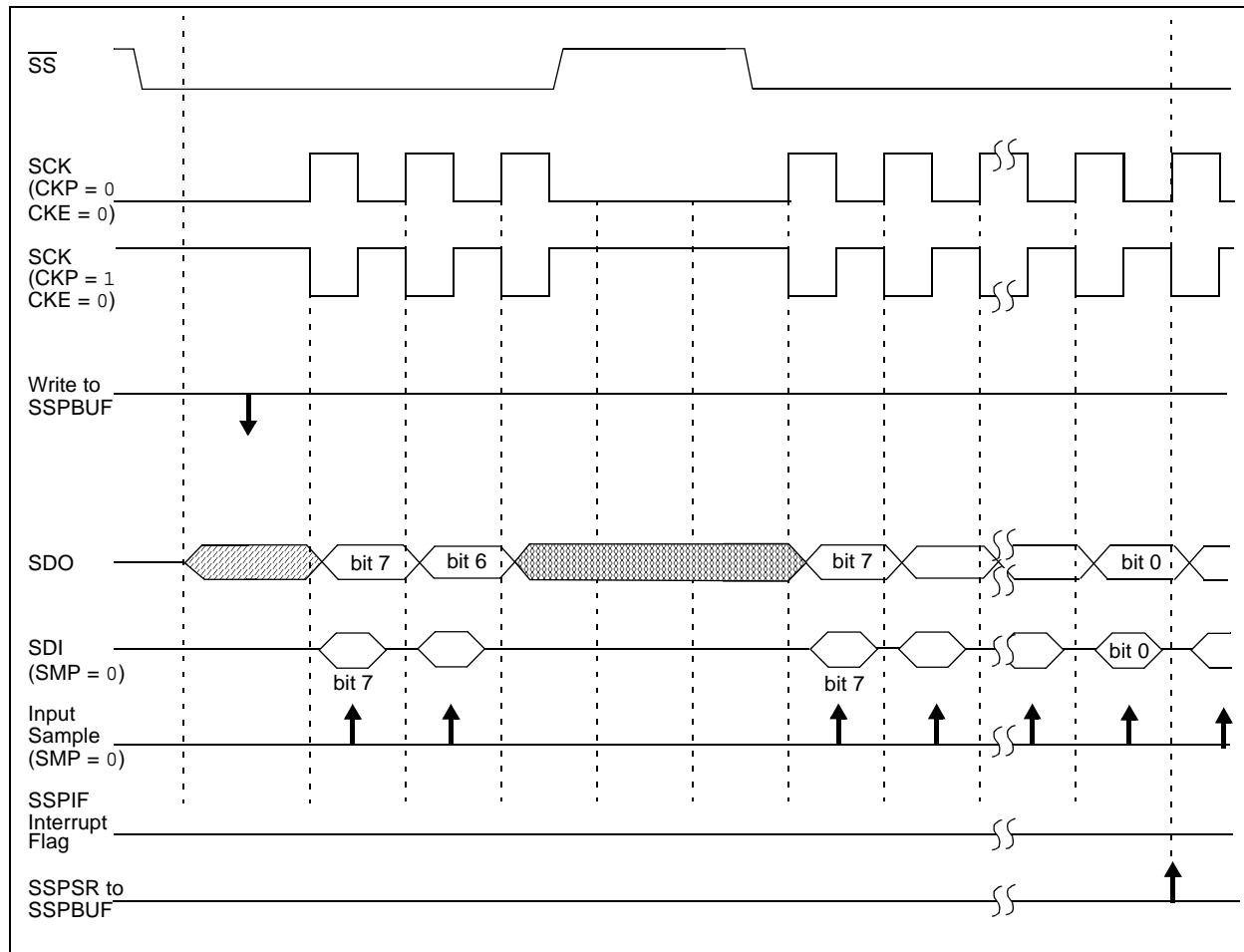
## 15.2.7 SLAVE SELECT SYNCHRONIZATION

The SS pin allows a Synchronous Slave mode. The SPI must be in Slave mode with SS pin control enabled (SSPCON1<3:0> = 0100). When the SS pin is low, transmission and reception are enabled and the SDO pin is driven. When the SS pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

- Note 1:** When the SPI is in Slave mode with SS pin control enabled (SSPCON1<3:0> = 0100), the SPI module will reset if the SS pin is set to VDD.
- 2:** When the SPI is used in Slave mode with CKE set the SS pin control must also 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.

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



# PIC18F1XK50/PIC18LF1XK50

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

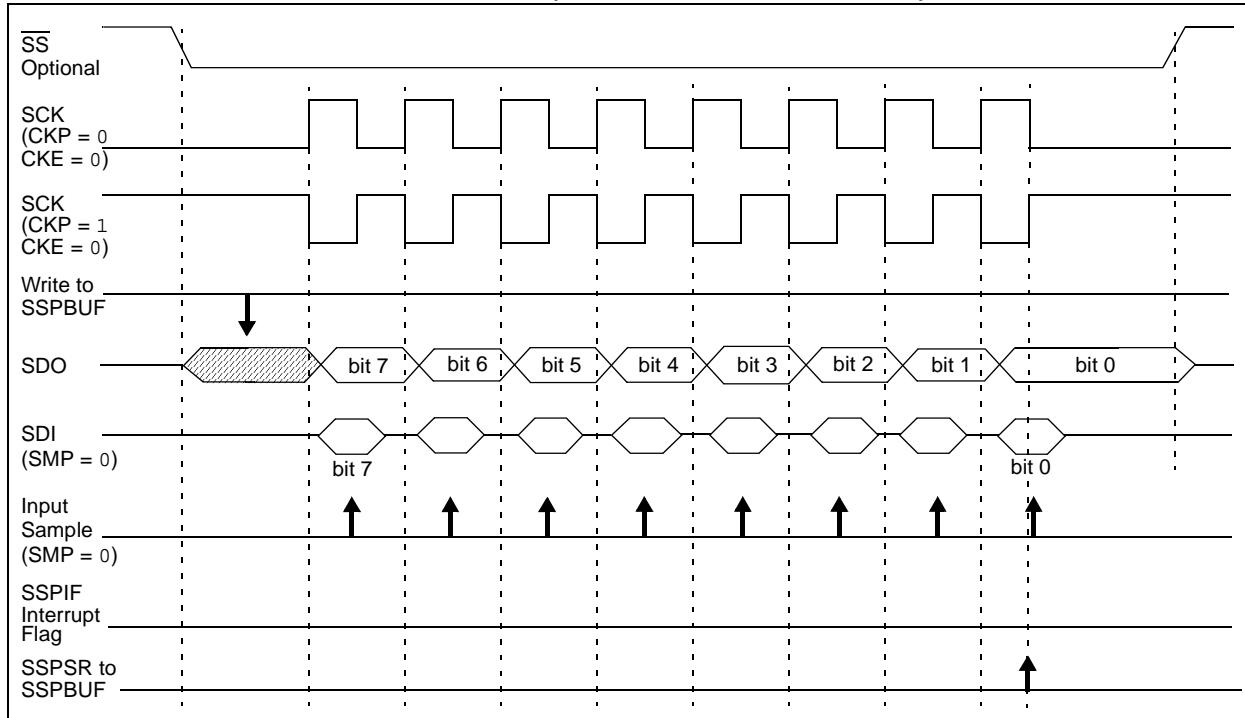
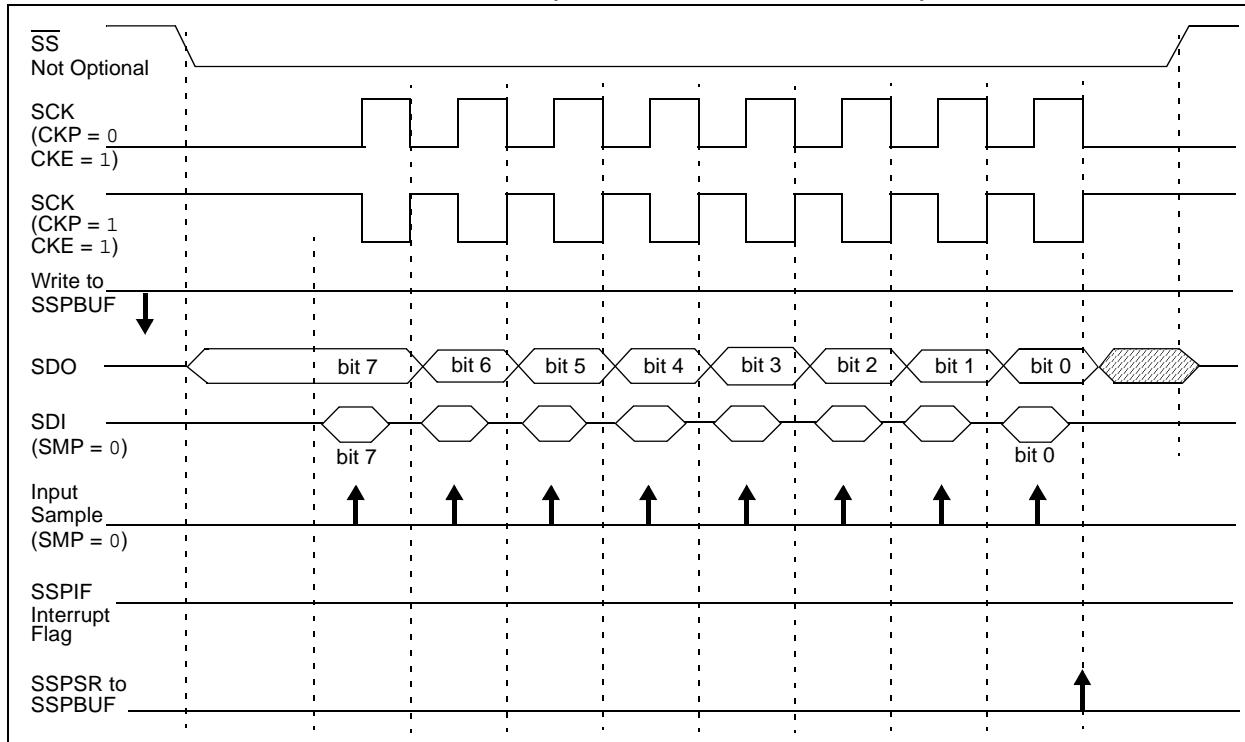


FIGURE 15-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



## 15.2.8 OPERATION IN POWER-MANAGED MODES

In SPI Master mode, module clocks may be operating at a different speed than when in full power mode; in the case of the Sleep mode, all clocks are halted.

In all Idle modes, a clock is provided to the peripherals. That clock could be from the primary clock source, the secondary clock (Timer1 oscillator at 32.768 kHz) or the INTOSC source. See **Section 19.0 “Power-Managed Modes”** for additional information.

In most cases, the speed that the master clocks SPI data is not important; however, this should be evaluated for each system.

When MSSP interrupts are enabled, after the master completes sending data, an MSSP interrupt will wake the controller:

- from Sleep, in slave mode
- from Idle, in slave or master mode

If an exit from Sleep or Idle mode is not desired, MSSP interrupts should be disabled.

In SPI master mode, when the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in any power-managed mode and data to be shifted into the SPI

Transmit/Receive Shift register. When all 8 bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

## 15.2.9 EFFECTS OF A RESET

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

## 15.2.10 BUS MODE COMPATIBILITY

Table 15-1 shows the compatibility between the standard SPI modes and the states of 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 an 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	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	282
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282
SSPBUF	SSP Receive Buffer/Transmit Register								280
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	280
SSPSTAT	SMP	CKE	D/Ā	P	S	R/W	UA	BF	280

**Legend:** Shaded cells are not used by the MSSP in SPI mode.

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

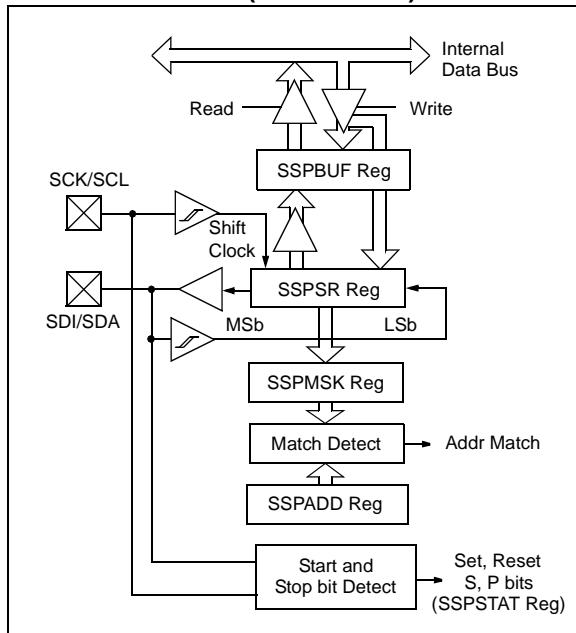
The MSSP module in I<sup>2</sup>C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial clock – SCL
- Serial data – SDA

**Note:** The user must configure these pins as inputs with the corresponding TRIS bits.

**FIGURE 15-7: MSSP BLOCK DIAGRAM (I<sup>2</sup>C<sup>TM</sup> MODE)**



### 15.3.1 REGISTERS

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

- MSSP Control Register 1 (SSPCON1)
- MSSP Control Register 2 (SSPCON2)
- MSSP Status register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible
- MSSP Address Register (SSPADD)
- MSSP Address Mask (SSPMSK)

SSPCON1, SSPCON2 and SSPSTAT are the control and STATUS registers in I<sup>2</sup>C mode operation. The SSPCON1 and SSPCON2 registers are readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

When the MSSP is configured in Master mode, the SSPADD register acts as the Baud Rate Generator reload value. When the MSSP is configured for I<sup>2</sup>C slave mode the SSPADD register holds the slave device address. The MSSP can be configured to respond to a range of addresses by qualifying selected bits of the address register with the SSPMSK register.

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 <sup>(1)</sup>	S <sup>(1)</sup>	R/W <sup>(2, 3)</sup>	UA	BF
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>SMP:</b> Slew Rate Control bit <u>In Master or Slave mode:</u> 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	<b>CKE:</b> SMBus Select bit <u>In Master or Slave mode:</u> 1 = Enable SMBus specific inputs 0 = Disable SMBus specific inputs
bit 5	<b>D/A:</b> Data/Address bit <u>In Master mode:</u> Reserved. <u>In Slave mode:</u> 1 = Indicates that the last byte received or transmitted was data 0 = Indicates that the last byte received was an address
bit 4	<b>P:</b> Stop bit <sup>(1)</sup> 1 = Indicates that a Stop bit has been detected last 0 = Stop bit was not detected last
bit 3	<b>S:</b> Start bit <sup>(1)</sup> 1 = Indicates that a Start bit has been detected last 0 = Start bit was not detected last
bit 2	<b>R/W:</b> Read/Write Information bit (I <sup>2</sup> C mode only) <sup>(2, 3)</sup> <u>In Slave mode:</u> 1 = Read 0 = Write <u>In Master mode:</u> 1 = Transmit is in progress 0 = Transmit is not in progress
bit 1	<b>UA:</b> Update Address bit (10-bit Slave mode only) 1 = Indicates that the user needs to update the address in the SSPADD register 0 = Address does not need to be updated
bit 0	<b>BF:</b> Buffer Full Status bit <u>In Transmit mode:</u> 1 = SSPBUF is full 0 = SSPBUF is empty <u>In Receive mode:</u> 1 = SSPBUF is full (does not include the ACK and Stop bits) 0 = SSPBUF is empty (does not include the ACK and Stop bits)

**Note 1:** This bit is cleared on Reset and when SSPEN is cleared.

**2:** This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not ACK bit.

**3:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the Master mode is active.

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 15-4: SSPCON1: MSSP CONTROL 1 REGISTER (I<sup>2</sup>C MODE)

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

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>WCOL:</b> Write Collision Detect bit
	<u>In Master Transmit mode:</u>
	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 by software)
	0 = No collision
	<u>In Slave Transmit mode:</u>
	1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared by software)
	0 = No collision
	<u>In Receive mode (Master or Slave modes):</u>
	This is a "don't care" bit.
bit 6	<b>SSPOV:</b> Receive Overflow Indicator bit
	<u>In Receive mode:</u>
	1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared by software)
	0 = No overflow
	<u>In Transmit mode:</u>
	This is a "don't care" bit in Transmit mode.
bit 5	<b>SSPEN:</b> 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
	When enabled, the SDA and SCL pins must be properly configured as inputs.
bit 4	<b>CKP:</b> SCK Release Control bit
	<u>In Slave mode:</u>
	1 = Release clock
	0 = Holds clock low (clock stretch), used to ensure data setup time
	<u>In Master mode:</u>
	Unused in this mode.
bit 3-0	<b>SSPM&lt;3:0&gt;:</b> 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
	Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

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

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT <sup>(2)</sup>	ACKEN <sup>(1)</sup>	RCEN <sup>(1)</sup>	PEN <sup>(1)</sup>	RSEN <sup>(1)</sup>	SEN <sup>(1)</sup>
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>GCEN:</b> General Call Enable bit (Slave mode only) 1 = Generate interrupt when a general call address 0x00 or 00h is received in the SSPSR 0 = General call address disabled
bit 6	<b>ACKSTAT:</b> Acknowledge Status bit (Master Transmit mode only) 1 = Acknowledge was not received from slave 0 = Acknowledge was received from slave
bit 5	<b>ACKDT:</b> Acknowledge Data bit (Master Receive mode only) <sup>(2)</sup> 1 = Not Acknowledge 0 = Acknowledge
bit 4	<b>ACKEN:</b> Acknowledge Sequence Enable bit (Master Receive mode only) <sup>(1)</sup> 1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware. 0 = Acknowledge sequence Idle
bit 3	<b>RCEN:</b> Receive Enable bit (Master mode only) <sup>(1)</sup> 1 = Enables Receive mode for I <sup>2</sup> C 0 = Receive Idle
bit 2	<b>PEN:</b> Stop Condition Enable bit (Master mode only) <sup>(1)</sup> 1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware. 0 = Stop condition Idle
bit 1	<b>RSEN:</b> Repeated Start Condition Enable bit (Master mode only) <sup>(1)</sup> 1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware. 0 = Repeated Start condition Idle
bit 0	<b>SEN:</b> Start Condition Enable/Stretch Enable bit <sup>(1)</sup> <u>In Master mode:</u> 1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware. 0 = Start condition Idle <u>In Slave mode:</u> 1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled) 0 = Clock stretching is disabled

**Note 1:** For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I<sup>2</sup>C module is not in the Idle mode, these bits may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

**2:** Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

## 15.3.2 OPERATION

The MSSP module functions are enabled by setting SSPEN bit of the SSPCON1 register.

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

- I<sup>2</sup>C Master mode, clock = (Fosc/(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 mode, slave is Idle

Selection of any I<sup>2</sup>C mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain, provided these pins are programmed to inputs by setting the appropriate TRIS bits

**Note:** To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

## 15.3.3 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs. 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 bit of the SSPSTAT register, is set before the transfer is received.
- The overflow bit, SSPOV bit of the SSPCON1 register, is set before the transfer is received.

In this case, the SSPSR register value is not loaded into the SSPBUF, but bit SSPIF of the PIR1 register 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 **Section 27.0 “Electrical Specifications”**.

## 15.3.3.1 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 ACK pulse is generated.
4. MSSP Interrupt Flag bit, SSPIF of the PIR1 register, 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 of the SSPSTAT register 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 UA of the SSPSTAT register are set).
2. Read the SSPBUF register (clears bit BF) and clear flag bit, SSPIF.
3. Update the SSPADD register with second (low) byte of address (clears bit UA and releases the SCL line).
4. Receive second (low) byte of address (bits SSPIF, BF and UA are set). If the address matches then the SCL is held until the next step. Otherwise the SCL line is not held.
5. Read the SSPBUF register (clears bit BF) and clear flag bit, SSPIF.
6. Update the SSPADD register with the first (high) byte of address. (This will clear bit UA and release a held SCL line.)
7. Receive Repeated Start condition.
8. Receive first (high) byte of address with R/W bit set (bits SSPIF, BF, R/W are set).
9. Read the SSPBUF register (clears bit BF) and clear flag bit, SSPIF.
10. Load SSPBUF with byte the slave is to transmit, sets the BF bit.
11. Set the CKP bit to release SCL.

### 15.3.3.2 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 bit of the SSPSTAT register is set, or bit SSPOV bit of the SSPCON1 register is set.

An MSSP interrupt is generated for each data transfer byte. Flag bit, SSPIF of the PIR1 register, must be cleared by software.

When the SEN bit of the SSPCON2 register is set, SCL will be held low (clock stretch) following each data transfer. The clock must be released by setting the CKP bit of the SSPCON1 register. See **Section 15.3.4 “Clock Stretching”** for more detail.

### 15.3.3.3 Transmission

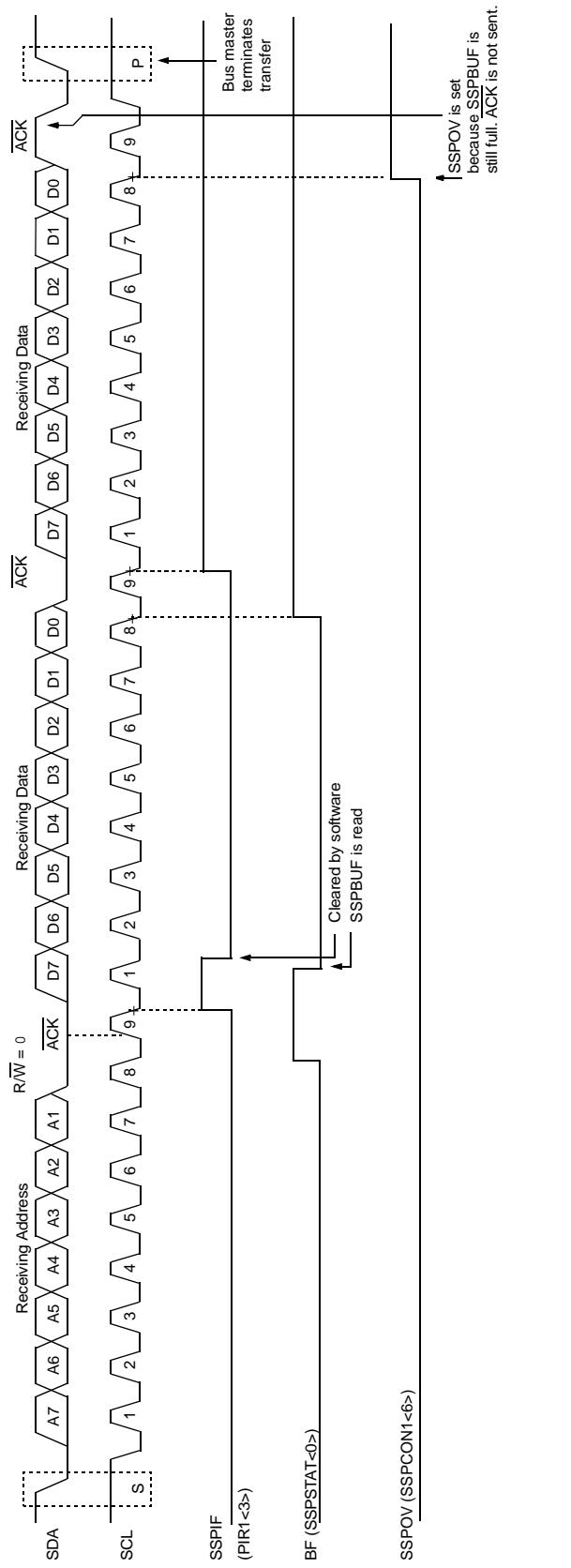
When the R/W bit of the incoming address byte is set and an address match occurs, the R/W bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The ACK pulse will be sent on the ninth bit and pin SCK/SCL is held low regardless of SEN (see **Section 15.3.4 “Clock Stretching”** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPBUF register which also loads the SSPSR register. Then pin SCK/SCL should be released by setting the CKP bit of the SSPCON1 register. 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 SCK/SCL must be released by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared by 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.

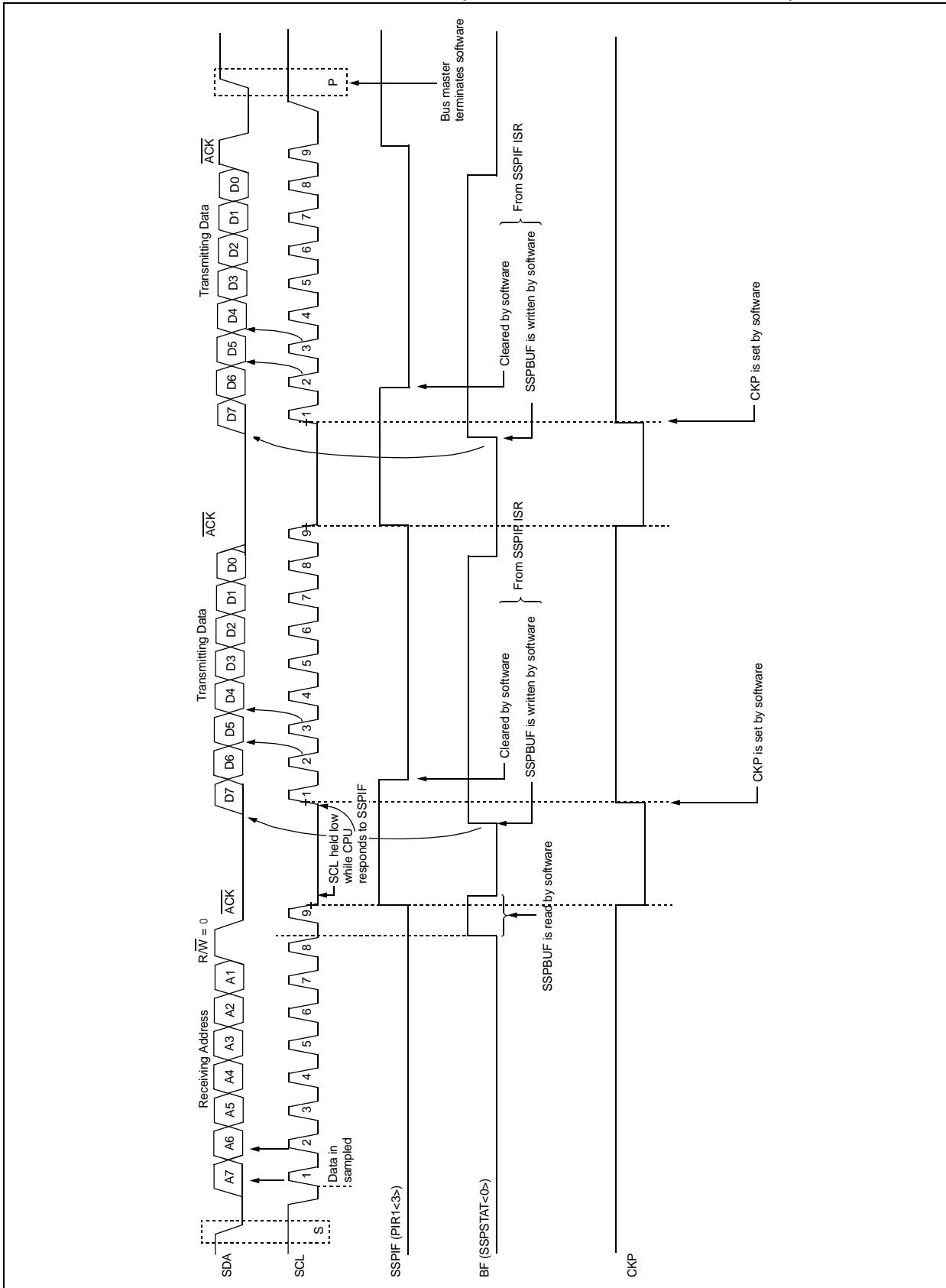
# PIC18F1XK50/PIC18LF1XK50

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



## **PIC18F1XK50/PIC18LF1XK50**

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



# PIC18F1XK50/PIC18LF1XK50

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

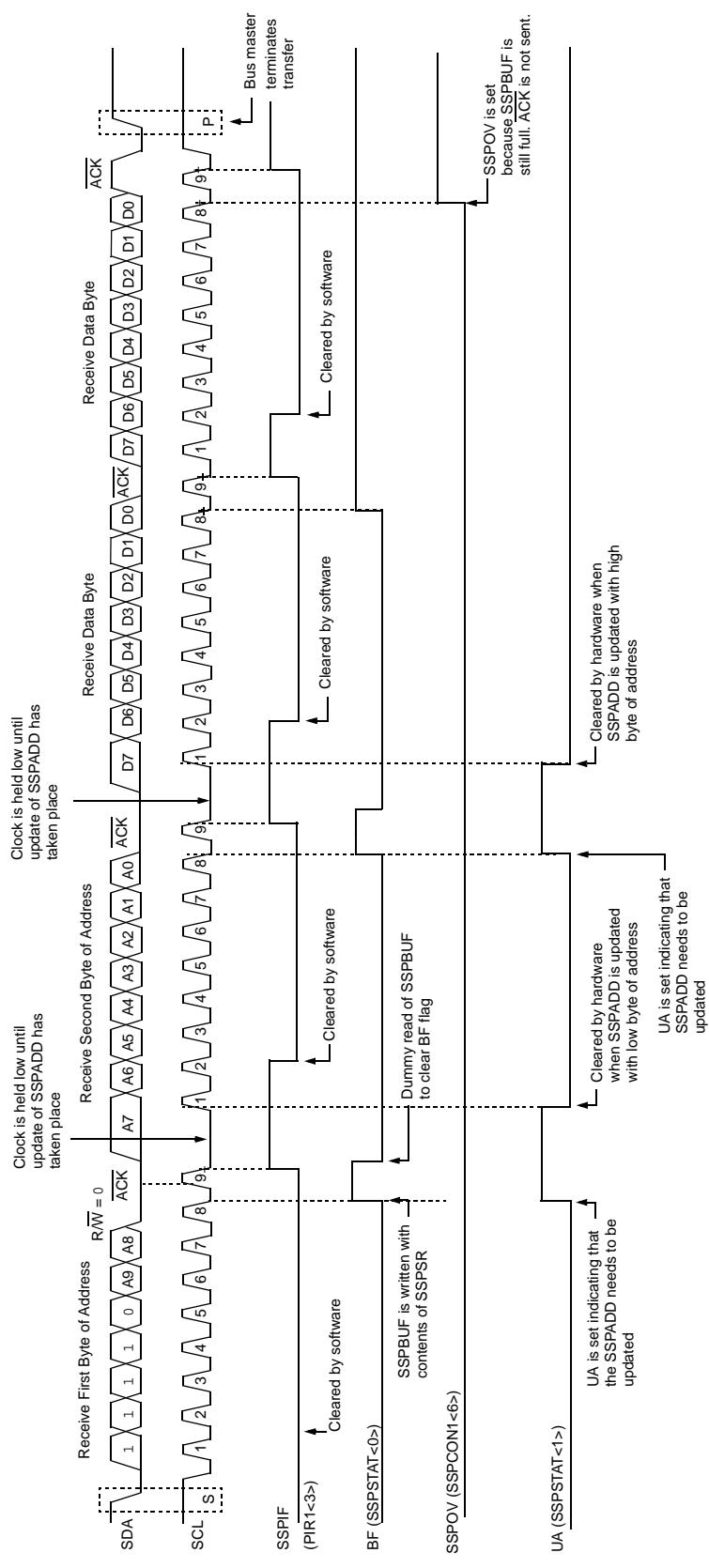
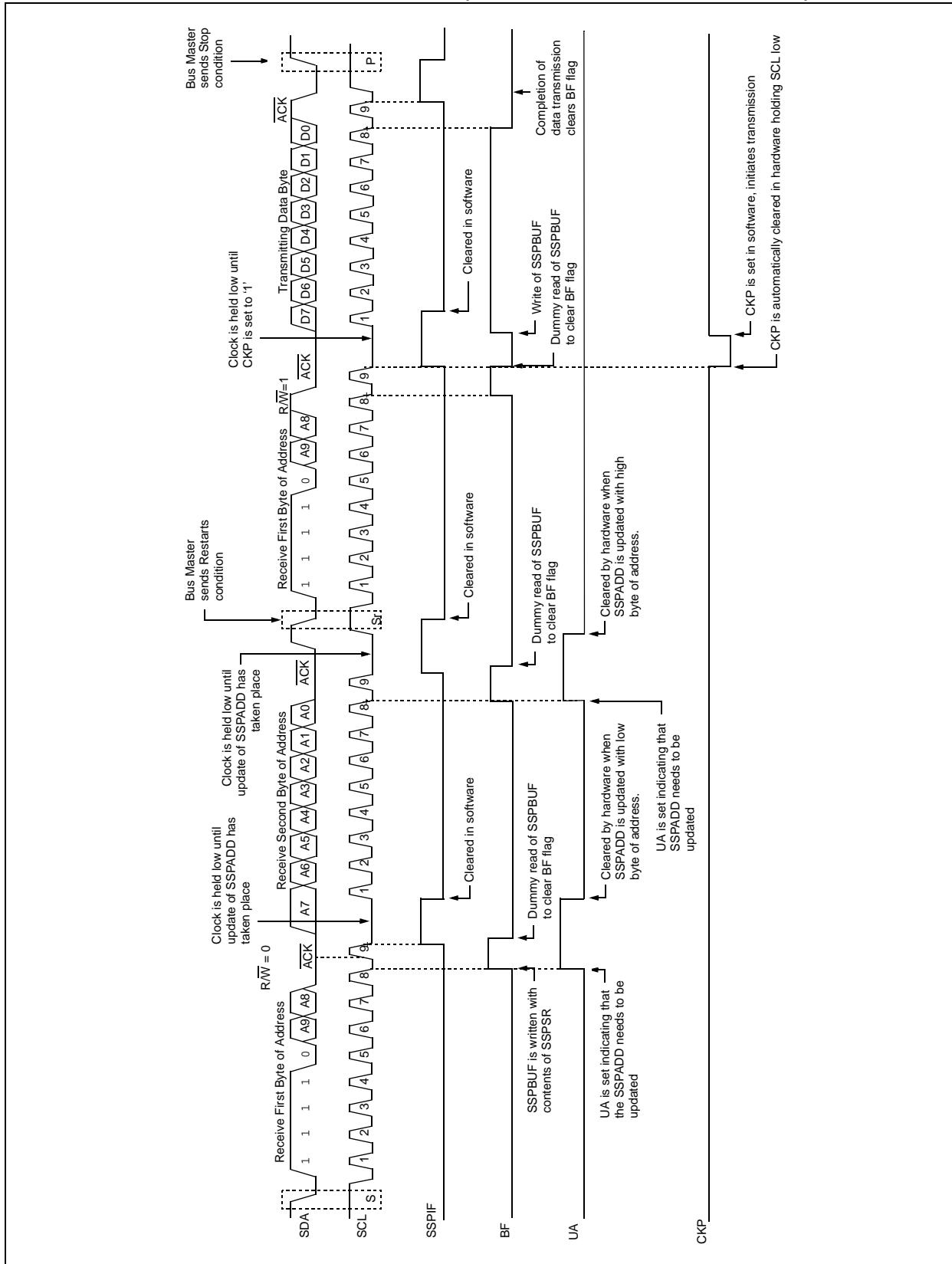


FIGURE 15-11: I<sup>2</sup>C<sup>TM</sup> SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)



# PIC18F1XK50/PIC18LF1XK50

### 15.3.3.4 SSP Mask Register

An SSP Mask (SSPMSK) register is available in I<sup>2</sup>C Slave mode as a mask for the value held in the SSPSR register during an address comparison operation. A zero ('0') bit in the SSPMSK register has the effect of making the corresponding bit in the SSPSR register a "don't care".

This register is reset to all '1's upon any Reset condition and, therefore, has no effect on standard SSP operation until written with a mask value.

This register must be initiated prior to setting SSPM<3:0> bits to select the I<sup>2</sup>C Slave mode (7-bit or 10-bit address).

The SSP Mask register is active during:

- 7-bit Address mode: address compare of A<7:1>.
- 10-bit Address mode: address compare of A<7:0> only. The SSP mask has no effect during the reception of the first (high) byte of the address.

### REGISTER 15-6: SSPMSK: SSP MASK REGISTER

| R/W-1               |
|-------|-------|-------|-------|-------|-------|-------|---------------------|
| MSK7  | MSK6  | MSK5  | MSK4  | MSK3  | MSK2  | MSK1  | MSK0 <sup>(1)</sup> |
| bit 7 | bit 0 |       |       |       |       |       |                     |

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-1

#### MSK<7:1>: Mask bits

- 1 = The received address bit n is compared to SSPADD<n> to detect I<sup>2</sup>C address match  
0 = The received address bit n is not used to detect I<sup>2</sup>C address match

bit 0

#### MSK<0>: Mask bit for I<sup>2</sup>C Slave mode, 10-bit Address<sup>(1)</sup>

I<sup>2</sup>C Slave mode, 10-bit Address (SSPM<3:0> = 0111):

- 1 = The received address bit 0 is compared to SSPADD<0> to detect I<sup>2</sup>C address match  
0 = The received address bit 0 is not used to detect I<sup>2</sup>C address match

**Note 1:** The MSK0 bit is used only in 10-bit slave mode. In all other modes, this bit has no effect.

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 15-7: SSPADD: MSSP ADDRESS AND BAUD RATE REGISTER (I<sup>2</sup>C MODE)

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| ADD7  | ADD6  | ADD5  | ADD4  | ADD3  | ADD2  | ADD1  | ADD0  |
| bit 7 |       |       |       |       |       |       | bit 0 |

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

### Master mode:

bit 7-0      **ADD<7:0>:** Baud Rate Clock Divider bits  
SCL pin clock period = ((ADD<7:0> + 1) \*4)/Fosc

### 10-Bit Slave mode — Most significant address byte:

bit 7-3      **Not used:** Unused for Most Significant Address Byte. Bit state of this register is a "don't care." Bit pattern sent by master is fixed by I<sup>2</sup>C specification and must be equal to '11110'. However, those bits are compared by hardware and are not affected by the value in this register.  
bit 2-1      **ADD<9:8>:** Two Most Significant bits of 10-bit address  
bit 0      **Not used:** Unused in this mode. Bit state is a "don't care."

### 10-Bit Slave mode — Least significant address byte:

bit 7-0      **ADD<7:0>:** Eight Least Significant bits of 10-bit address

### 7-Bit Slave mode:

bit 7-1      **ADD<6:0>:** 7-bit address  
bit 0      **Not used:** Unused in this mode. Bit state is a "don't care."

### 15.3.4 CLOCK STRETCHING

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

The SEN bit of the SSPCON2 register 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.

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

In 7-bit Slave Receive mode, on the falling edge of the ninth clock at the end of the ACK sequence if the BF bit is set, the CKP bit of 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 data transfer sequence (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 by 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.

#### 15.3.4.2 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.

#### 15.3.4.3 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. 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 data transfer 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 by software regardless of the state of the BF bit.

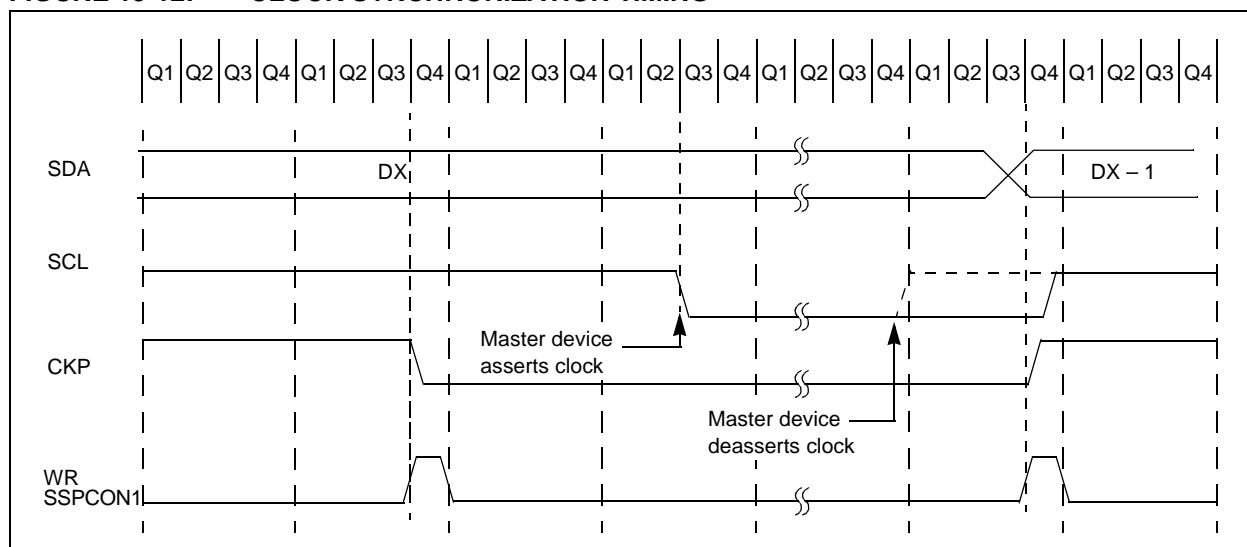
#### 15.3.4.4 Clock Stretching for 10-bit Slave Transmit Mode

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 automatic with the hardware clearing CKP, as in 7-bit Slave Transmit mode (see Figure 15-11).

#### 15.3.4.5 Clock Synchronization and the CKP bit

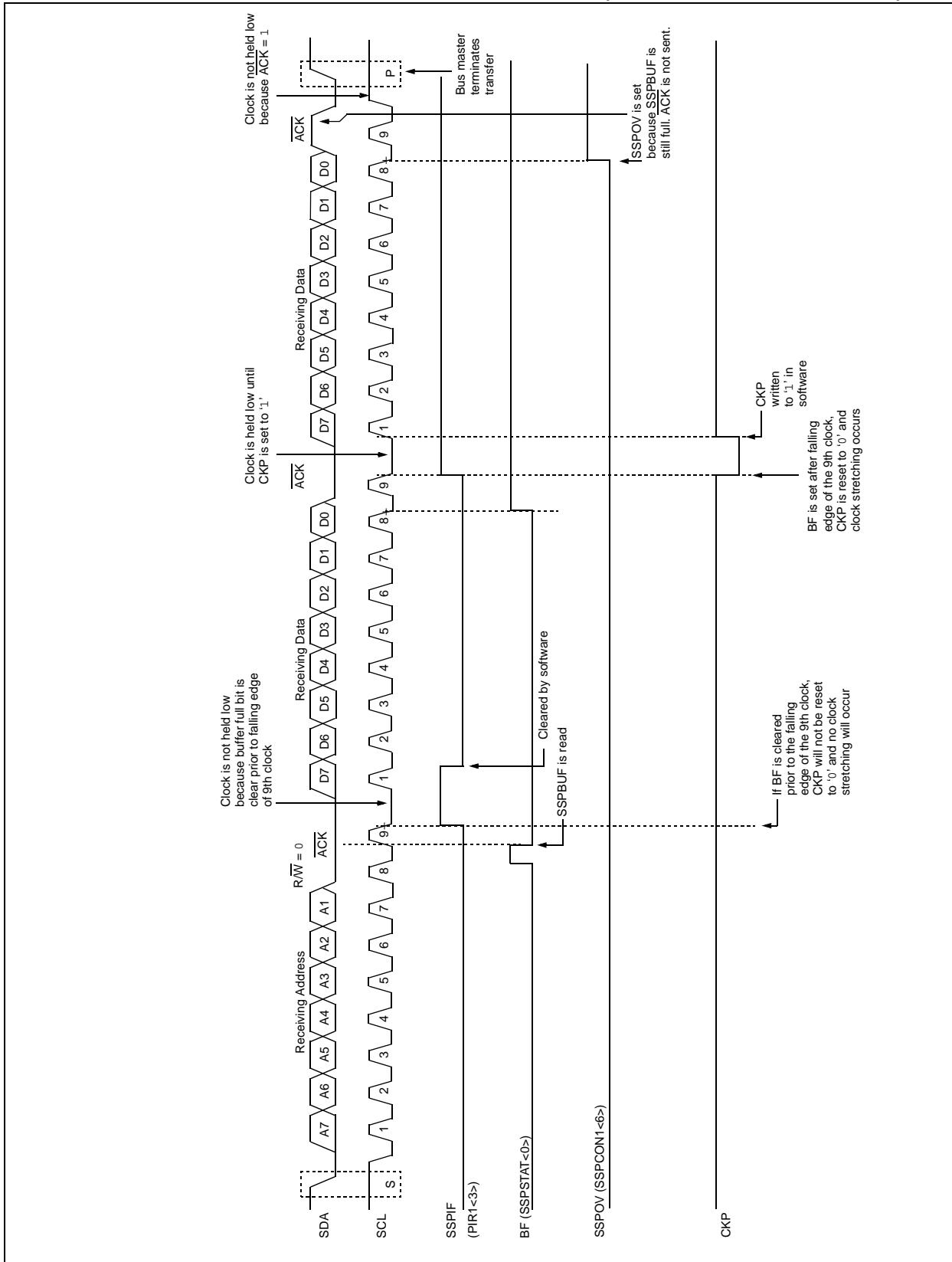
When the CKP bit is cleared, the SCL output is forced to '0'. However, clearing the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I<sup>2</sup>C master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I<sup>2</sup>C bus have deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 15-12).

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



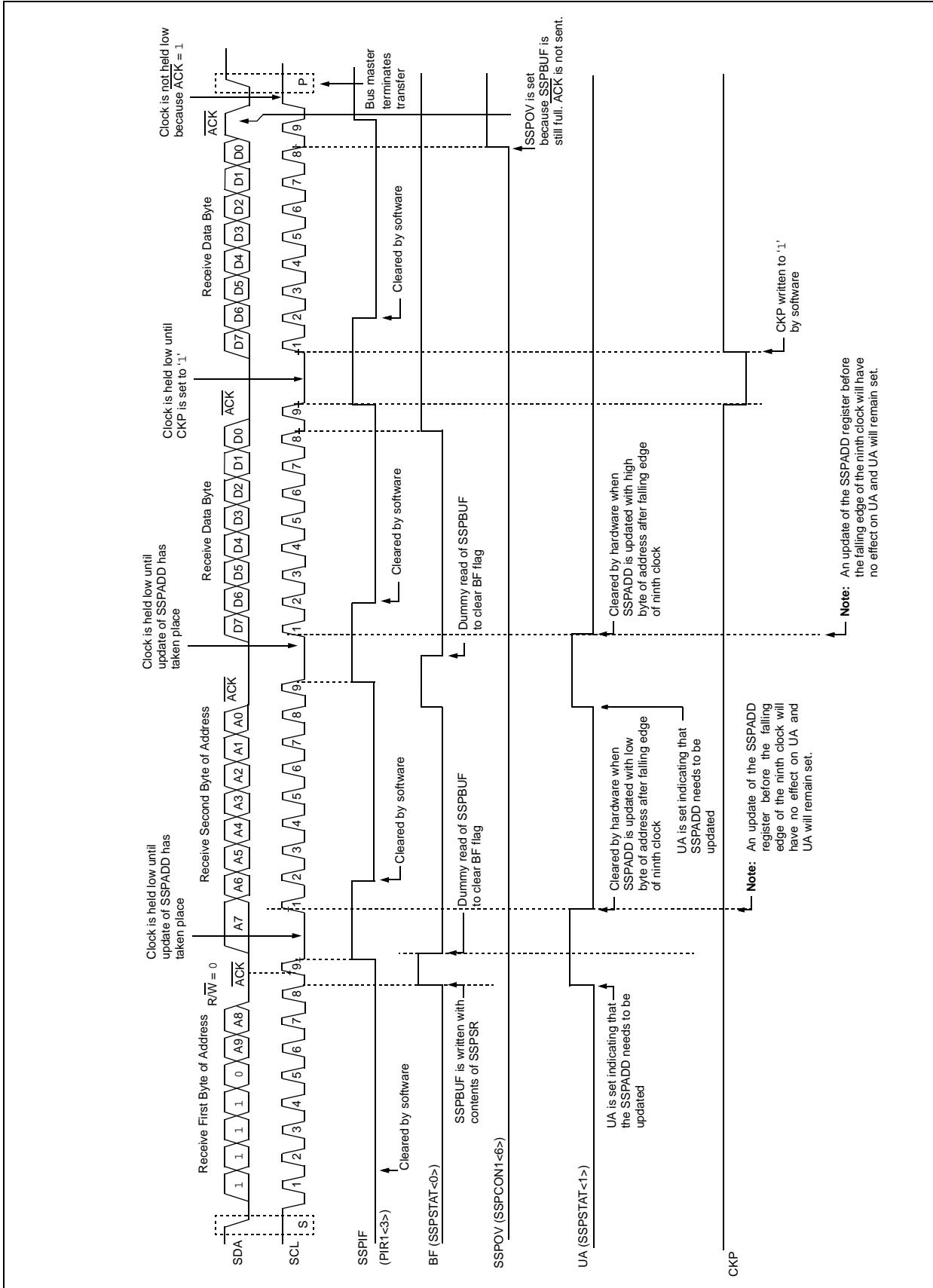
## **PIC18F1XK50/PIC18LF1XK50**

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



## **PIC18F1XK50/PIC18LF1XK50**

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



# PIC18F1XK50/PIC18LF1XK50

## 15.3.5 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.

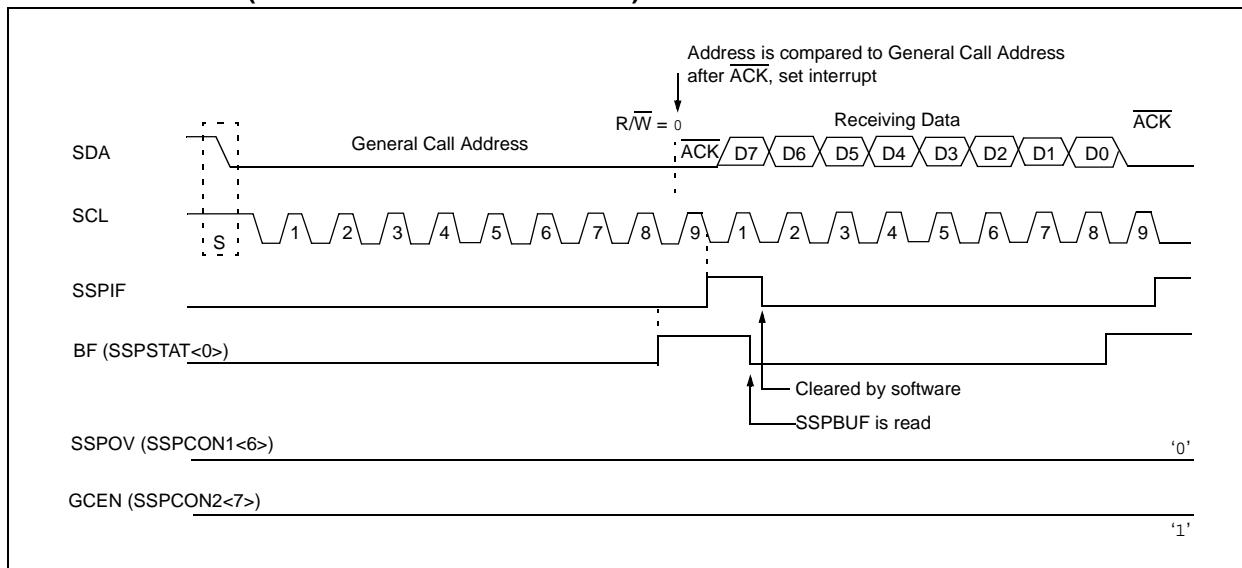
The general call address is recognized when the GCEN bit of the SSPCON2 is set. Following a Start bit detect, 8 bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

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

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

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match and the UA bit of the SSPSTAT register is set. 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)**



### 15.3.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I<sup>2</sup>C bus may be taken when the P bit is set, or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all I<sup>2</sup>C bus operations based on Start and Stop bit conditions.

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

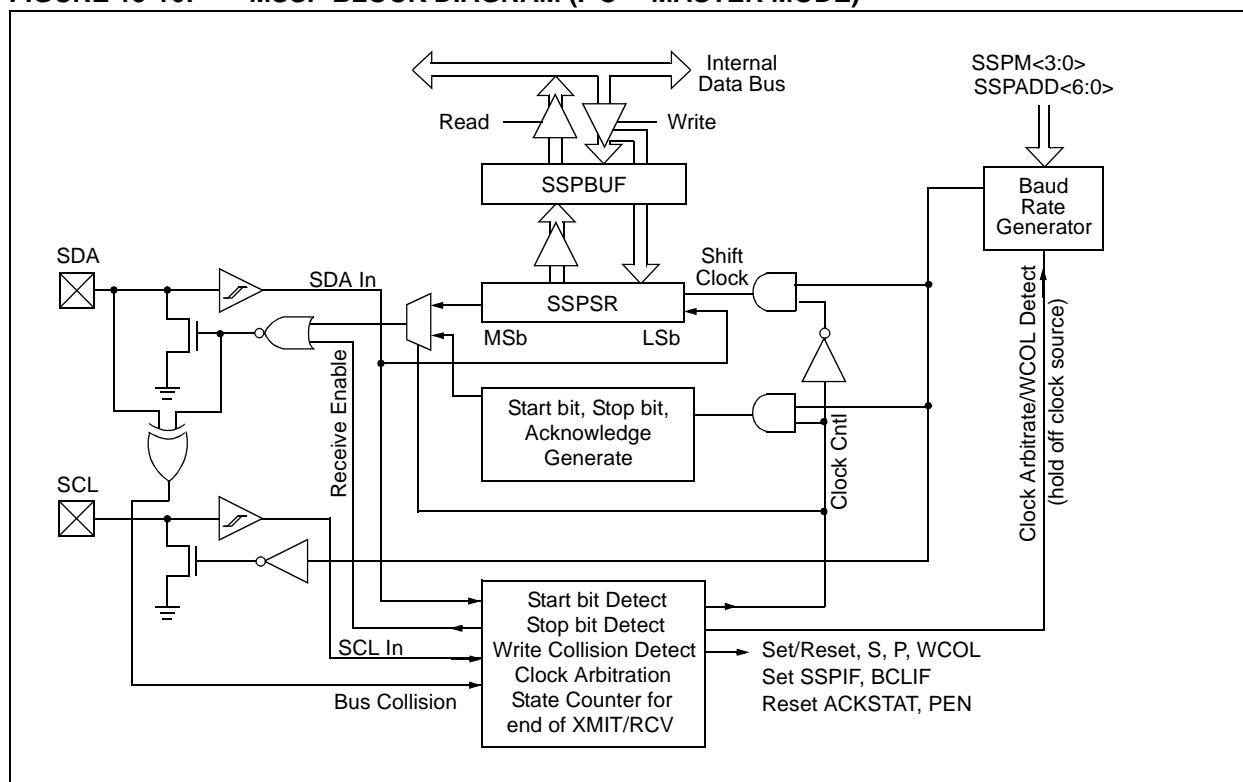
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the I<sup>2</sup>C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6.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

**Note:** The MSSP module, when configured in I<sup>2</sup>C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

The following events will cause the 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)**



## 15.3.6.1 I<sup>2</sup>C Master Mode Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I<sup>2</sup>C bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/W) bit. In this case, the R/W bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/W bit. In this case, the R/W bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

A Baud Rate Generator is used to set the clock frequency output on SCL. See **Section 15.3.7 "Baud Rate"** for more detail.

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the SEN bit of the SSPCON2 register.
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 ACKSTAT bit of the SSPCON2 register.
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
7. The user loads the SSPBUF with eight bits of data.
8. Data is shifted out the SDA pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
11. The user generates a Stop condition by setting the PEN bit of the SSPCON2 register.
12. Interrupt is generated once the Stop condition is complete.

### 15.3.7 BAUD RATE

In I<sup>2</sup>C Master mode, the Baud Rate Generator (BRG) reload value is placed in the SSPADD register (Figure 15-17). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting.

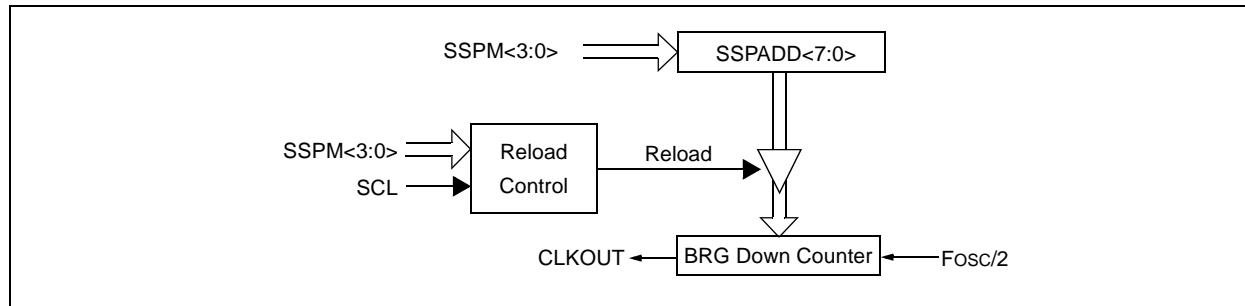
Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

Table 15-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

### EQUATION 15-1:

$$F_{SCL} = \frac{F_{OSC}}{(SSPADD + 1)(4)}$$

**FIGURE 15-17: BAUD RATE GENERATOR BLOCK DIAGRAM**



**TABLE 15-3: I<sup>2</sup>C™ CLOCK RATE W/BRG**

Fosc	F <sub>CY</sub>	BRG Value	F <sub>SCL</sub> (2 Rollovers of BRG)
48 MHz	12 MHz	0Bh	1 MHz <sup>(1)</sup>
48 MHz	12 MHz	1Dh	400 kHz
48 MHz	12 MHz	77h	100 kHz
40 MHz	10 MHz	18h	400 kHz <sup>(1)</sup>
40 MHz	10 MHz	1Fh	312.5 kHz
40 MHz	10 MHz	63h	100 kHz
16 MHz	4 MHz	09h	400 kHz <sup>(1)</sup>
16 MHz	4 MHz	0Ch	308 kHz
16 MHz	4 MHz	27h	100 kHz
4 MHz	1 MHz	02h	333 kHz <sup>(1)</sup>
4 MHz	1 MHz	09h	100 kHz
4 MHz	1 MHz	00h	1 MHz <sup>(1)</sup>

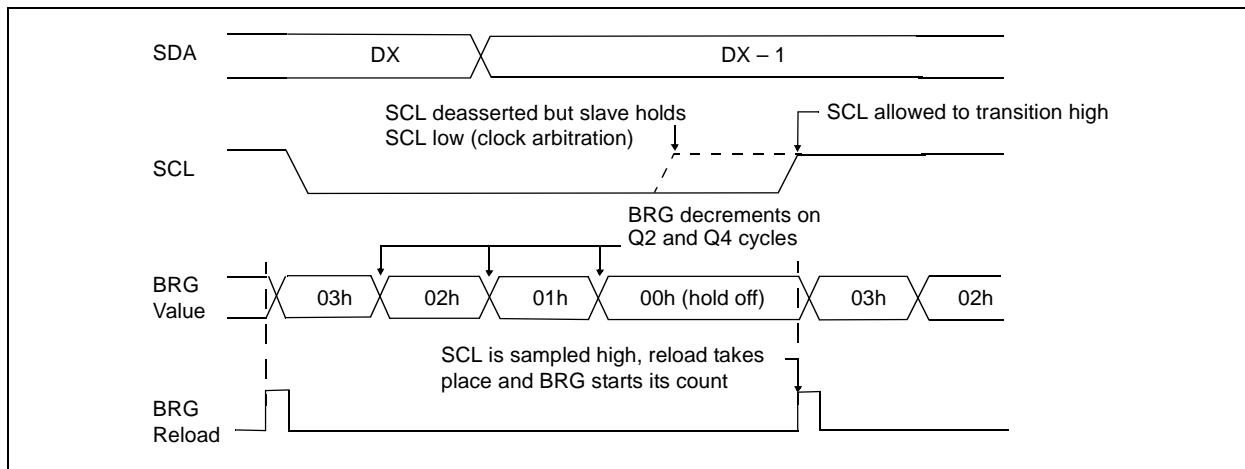
**Note 1:** The I<sup>2</sup>C interface does not conform to the 400 kHz I<sup>2</sup>C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.

# PIC18F1XK50/PIC18LF1XK50

## 15.3.7.1 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 15-18).

**FIGURE 15-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**



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

To initiate a Start condition, the user sets the Start Enable bit, SEN bit of the SSPCON2 register. 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 of the SSPSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit of the SSPCON2 register will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

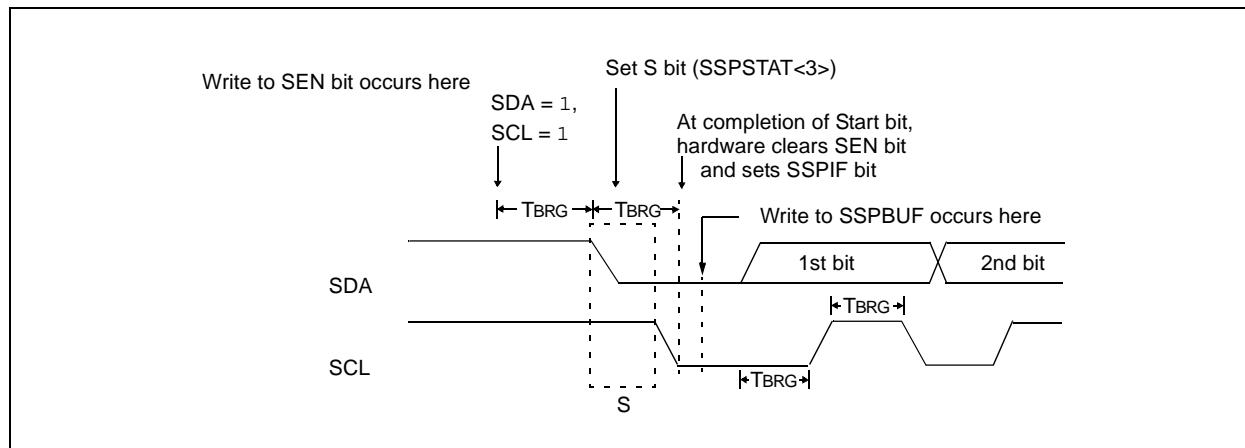
**Note:** If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted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

### 15.3.8.1 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**



# PIC18F1XK50/PIC18LF1XK50

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

A Repeated Start condition occurs when the RSEN bit of the SSPCON2 register 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 and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded 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 of the SSPCON2 register 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 of the SSPSTAT register 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'.

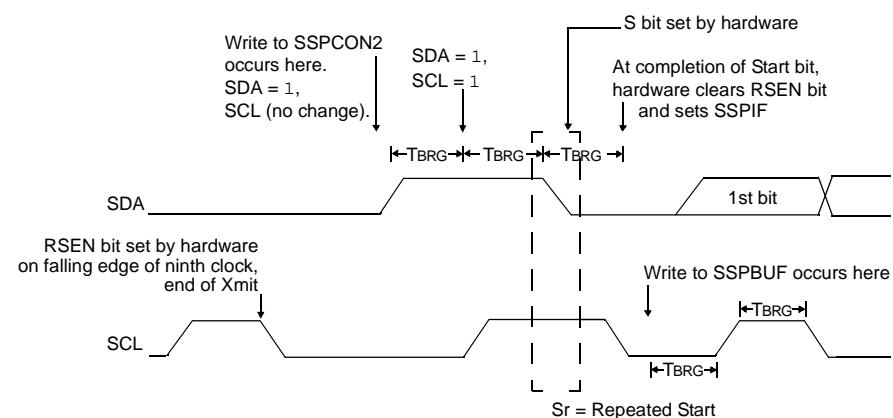
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode)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

### 15.3.9.1 WCOL Status Flag

If the user writes the SSPBUF when a Repeated Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

**Note:** Because queueing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated Start condition is complete.

**FIGURE 15-20: REPEAT START CONDITION WAVEFORM**



### 15.3.10 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 SP106). 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 SP107). 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 the address will be shifted out on the falling edge of SCL until all seven address bits and the R/W bit are completed. On the falling edge of the eighth clock, the master will deassert the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the ACK bit is loaded into the ACKSTAT status bit of the SSPCON2 register. 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.

#### 15.3.10.1 BF Status Flag

In Transmit mode, the BF bit of the SSPSTAT register is set when the CPU writes to SSPBUF and is cleared when all 8 bits are shifted out.

#### 15.3.10.2 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 by software before the next transmission.

#### 15.3.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit of the SSPCON2 register 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.

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

Master mode reception is enabled by programming the Receive Enable bit, RCEN bit of the SSPCON2 register.

**Note:** The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable, ACKEN bit of the SSPCON2 register.

#### 15.3.11.1 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.

#### 15.3.11.2 SSPOV Status Flag

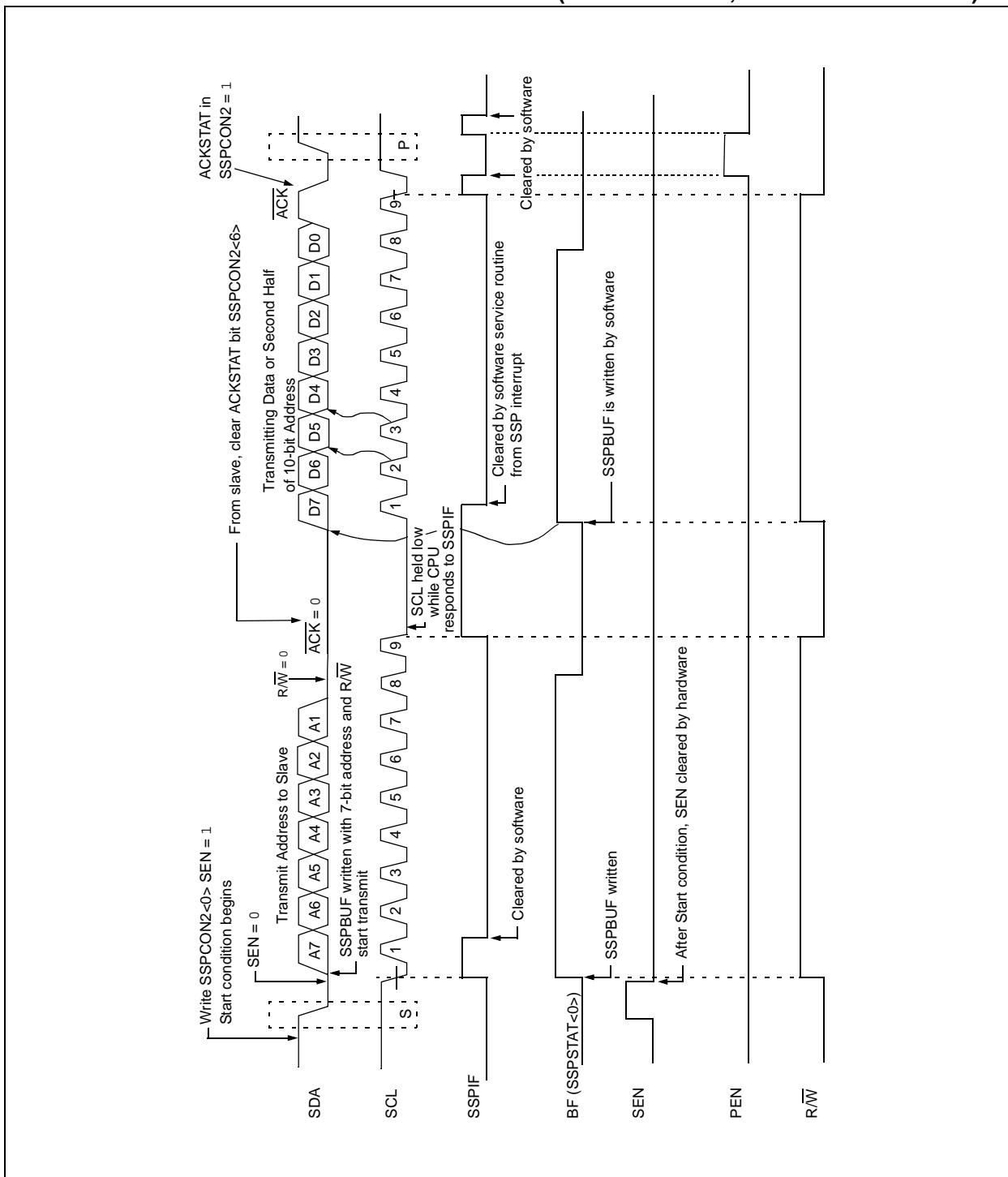
In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

#### 15.3.11.3 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).

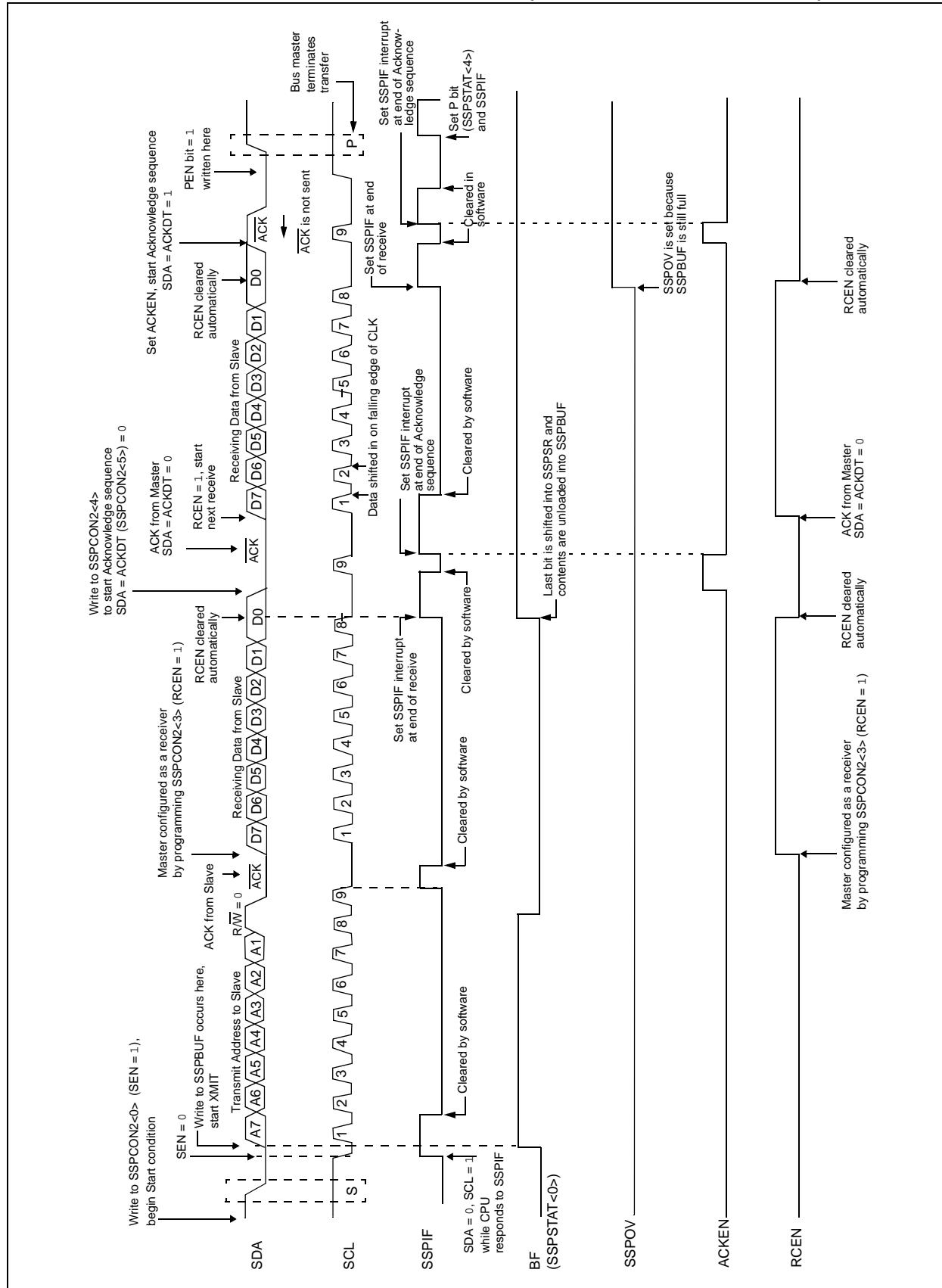
# **PIC18F1XK50/PIC18LF1XK50**

FIGURE 15-21: I<sup>2</sup>C<sup>TM</sup> MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)



## **PIC18F1XK50/PIC18LF1XK50**

FIGURE 15-22: I<sup>2</sup>C<sup>TM</sup> MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



# PIC18F1XK50/PIC18LF1XK50

## 15.3.12 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN bit of the SSPCON2 register. When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into Idle mode (Figure 15-23).

### 15.3.12.1 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).

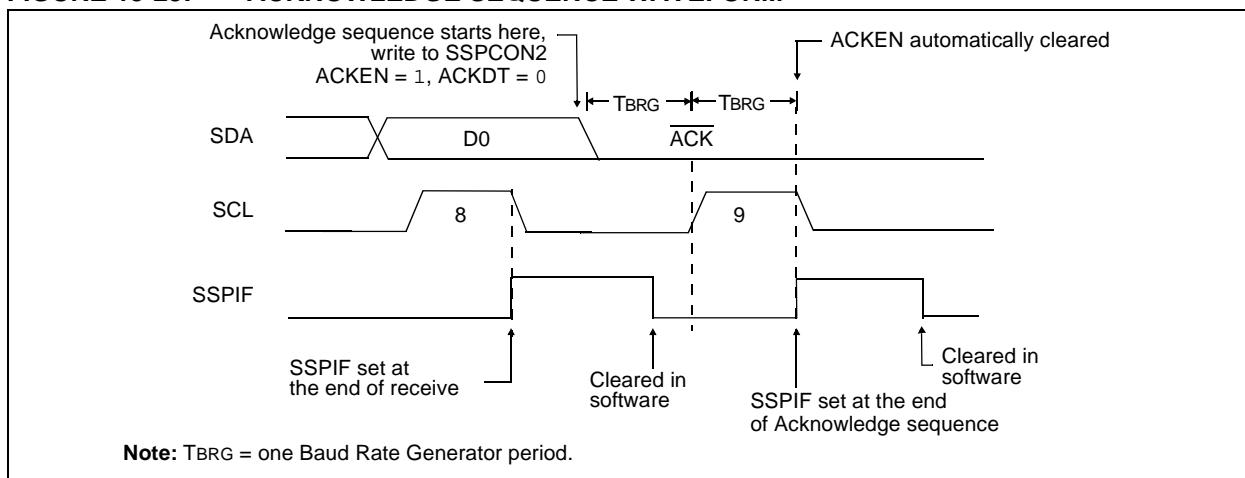
## 15.3.13 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN bit of the SSPCON2 register. At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit of the SSPSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 15-24).

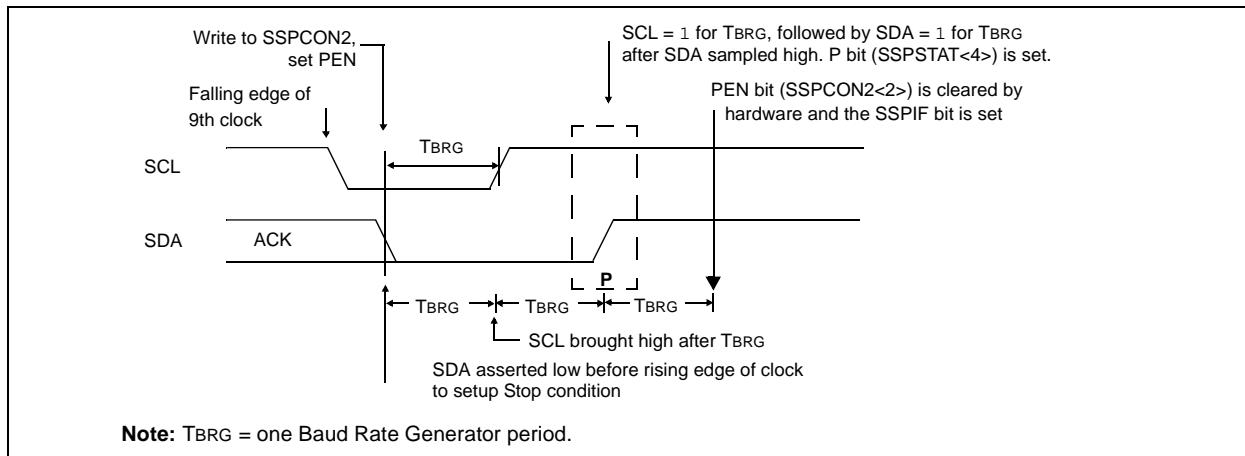
### 15.3.13.1 WCOL Status Flag

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**



### 15.3.14 SLEEP OPERATION

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

### 15.3.15 EFFECTS OF A RESET

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

### 15.3.16 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 of the SSPSTAT register 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 by 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

### 15.3.17 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 deasserted and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I<sup>2</sup>C bus is free, the user can resume communication by asserting a Start condition.

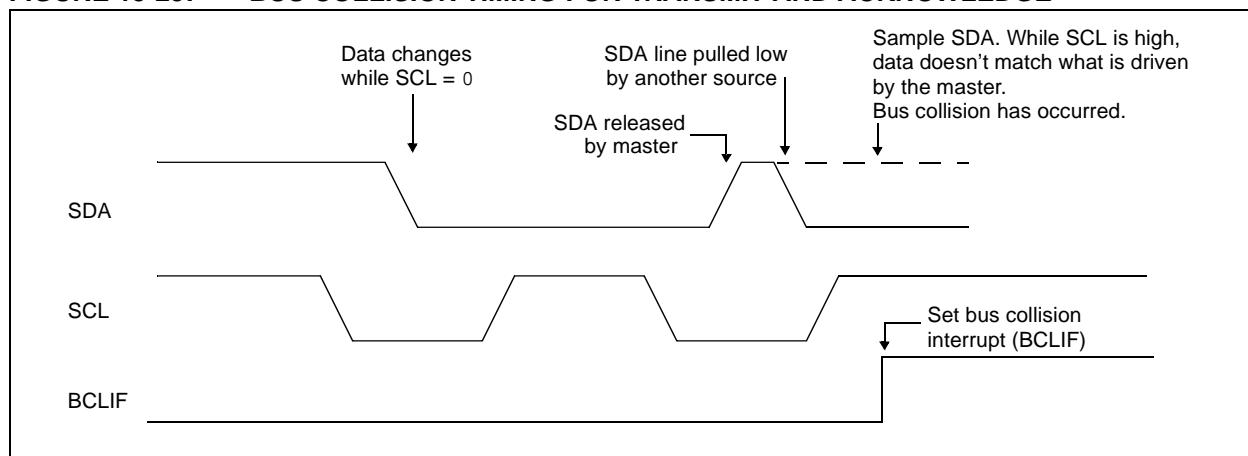
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I<sup>2</sup>C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I<sup>2</sup>C bus can be taken when the P bit is set in the SSPSTAT register, or the bus is Idle and the S and P bits are cleared.

**FIGURE 15-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**



# PIC18F1XK50/PIC18LF1XK50

## 15.3.17.1 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:

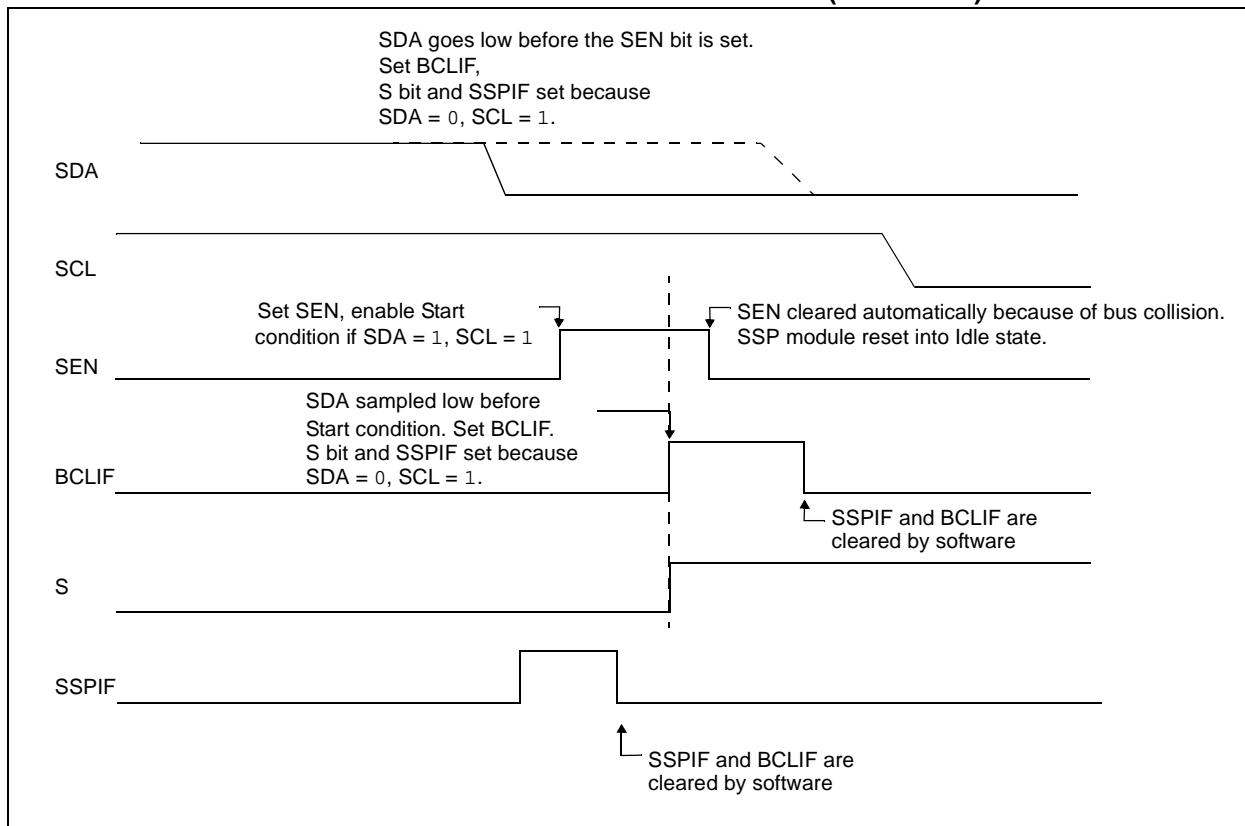
- the Start condition is aborted,
- the BCLIF flag is set and
- the MSSP module is reset to its Idle state (Figure 15-26).

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded and counts down. 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; if the SCL pin is sampled as '0' during this time, a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

**Note:** The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

**FIGURE 15-26: BUS COLLISION DURING START CONDITION (SDA ONLY)**



# PIC18F1XK50/PIC18LF1XK50

FIGURE 15-27: BUS COLLISION DURING START CONDITION (SCL = 0)

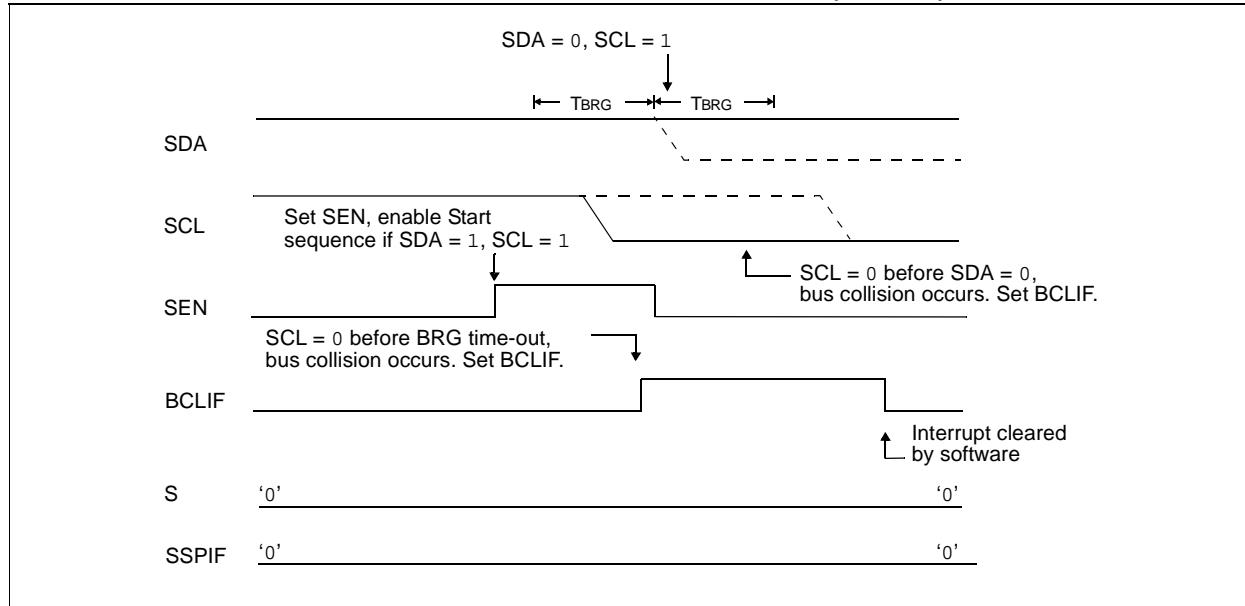
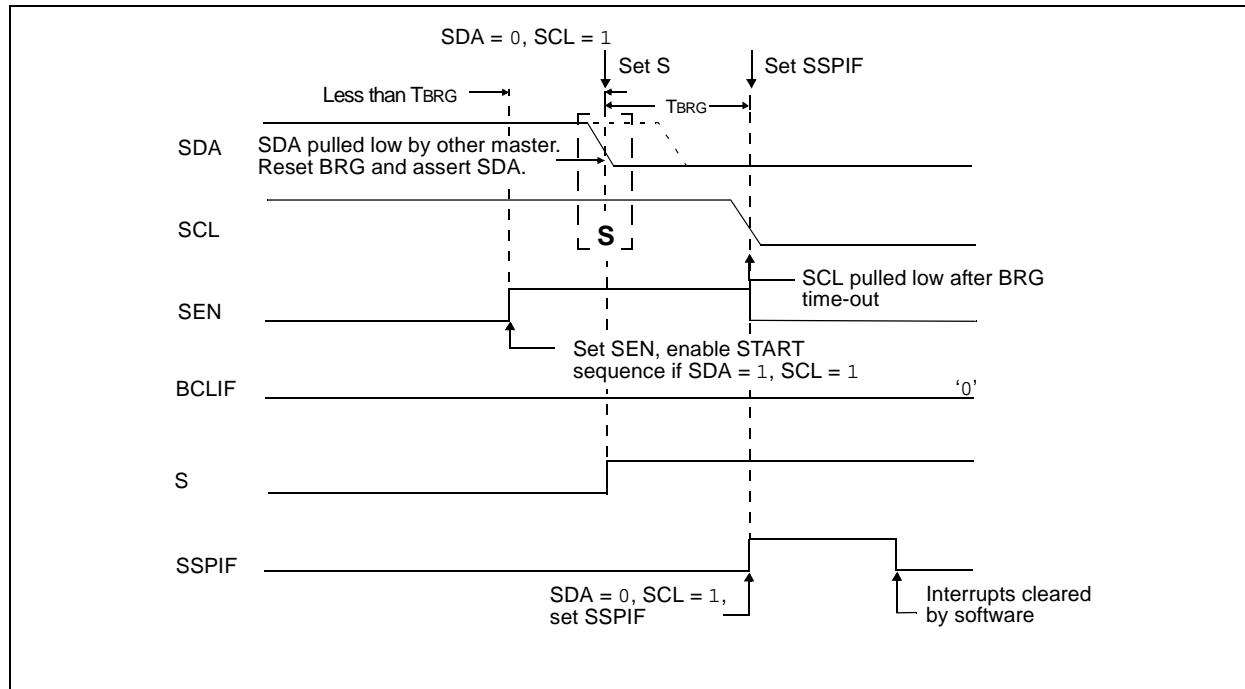


FIGURE 15-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



# PIC18F1XK50/PIC18LF1XK50

## 15.3.17.2 Bus Collision During a Repeated Start Condition

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

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

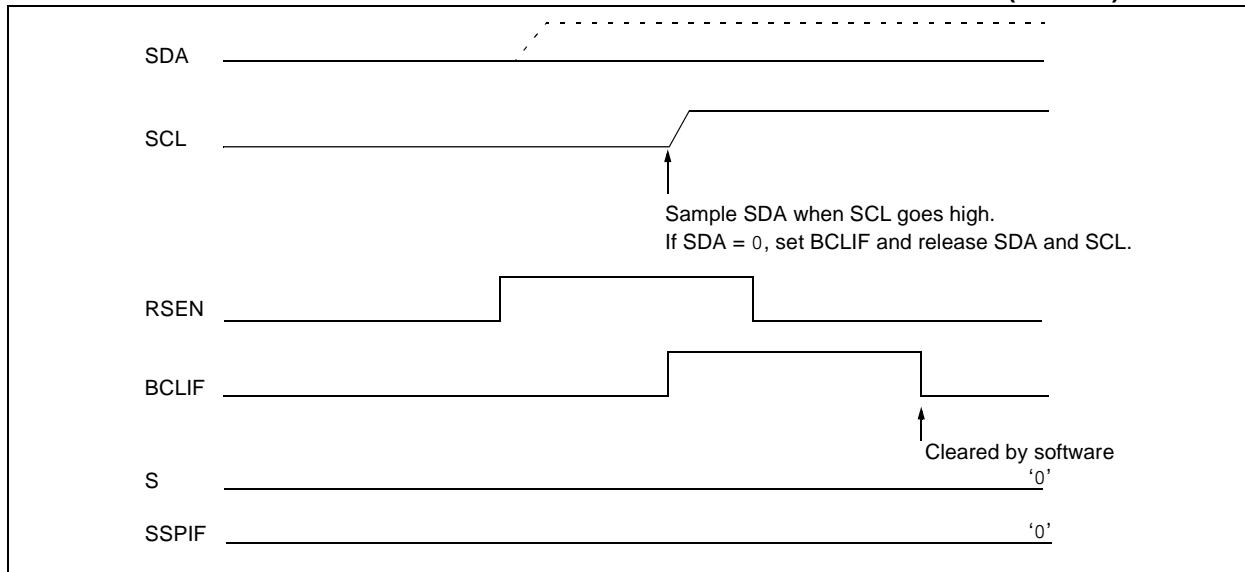
When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD and counts down to 0. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', Figure 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.

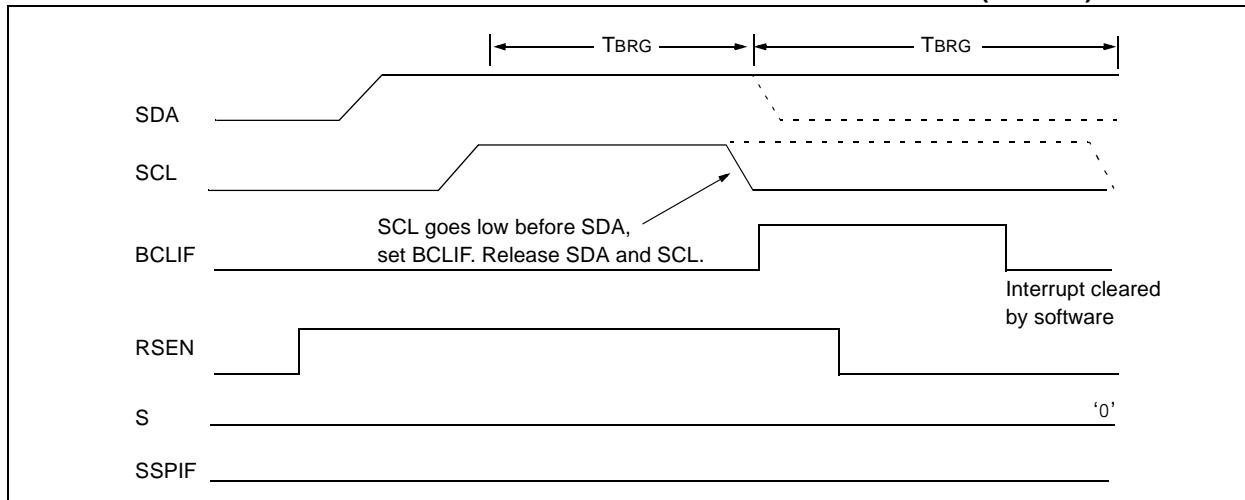
If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition, see Figure 15-30.

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)**



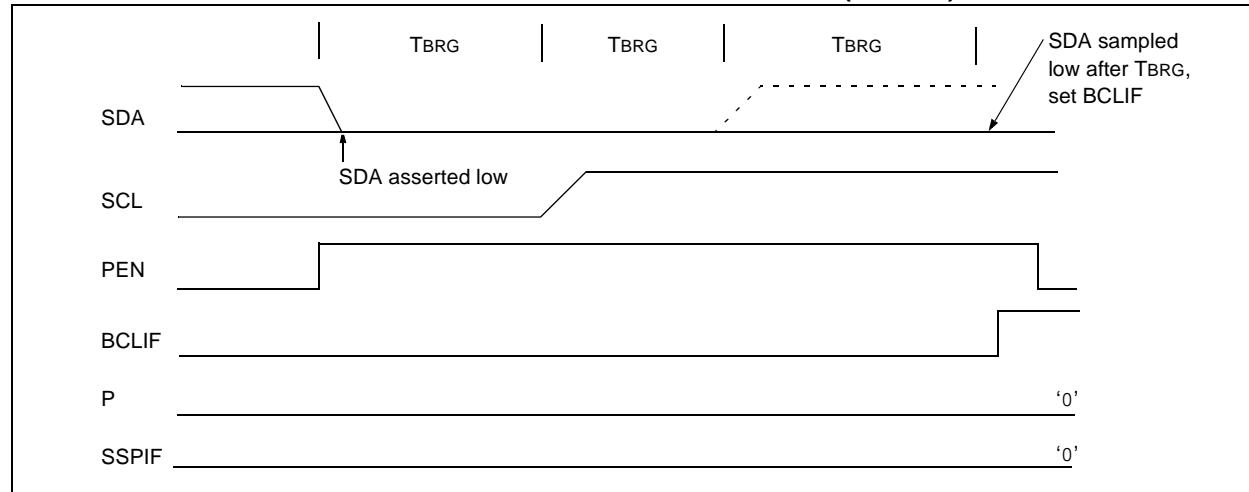
### 15.3.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

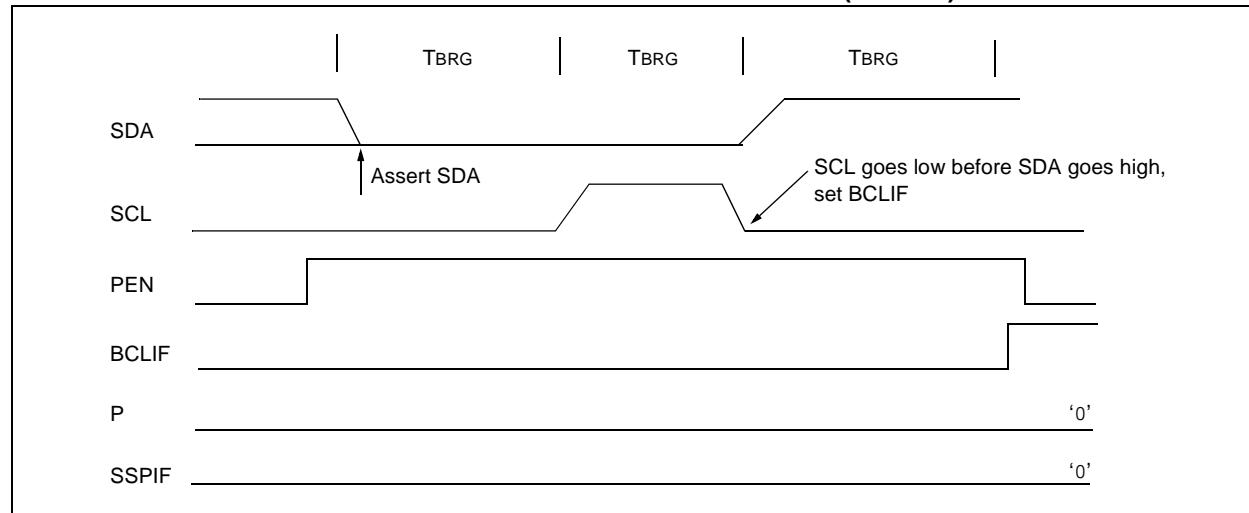
- After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.
- After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD 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)**



# PIC18F1XK50/PIC18LF1XK50

---

TABLE 15-4: SUMMARY OF REGISTERS ASSOCIATED WITH I<sup>2</sup>C™

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	282
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	282
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	282
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.								280
SSPBUF	SSP Receive Buffer/Transmit Register								280
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	280
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	280
SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	282
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	280
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	282

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

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

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer independent of device program execution. The EUSART, also known as a Serial Communications Interface (SCI), can be configured as a full-duplex asynchronous system or half-duplex synchronous system. Full-Duplex mode is useful for communications with peripheral systems, such as CRT terminals and personal computers. Half-Duplex Synchronous mode is intended for communications with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs or other microcontrollers. These devices typically do not have internal clocks for baud rate generation and require the external clock signal provided by a master synchronous device.

The EUSART module includes the following capabilities:

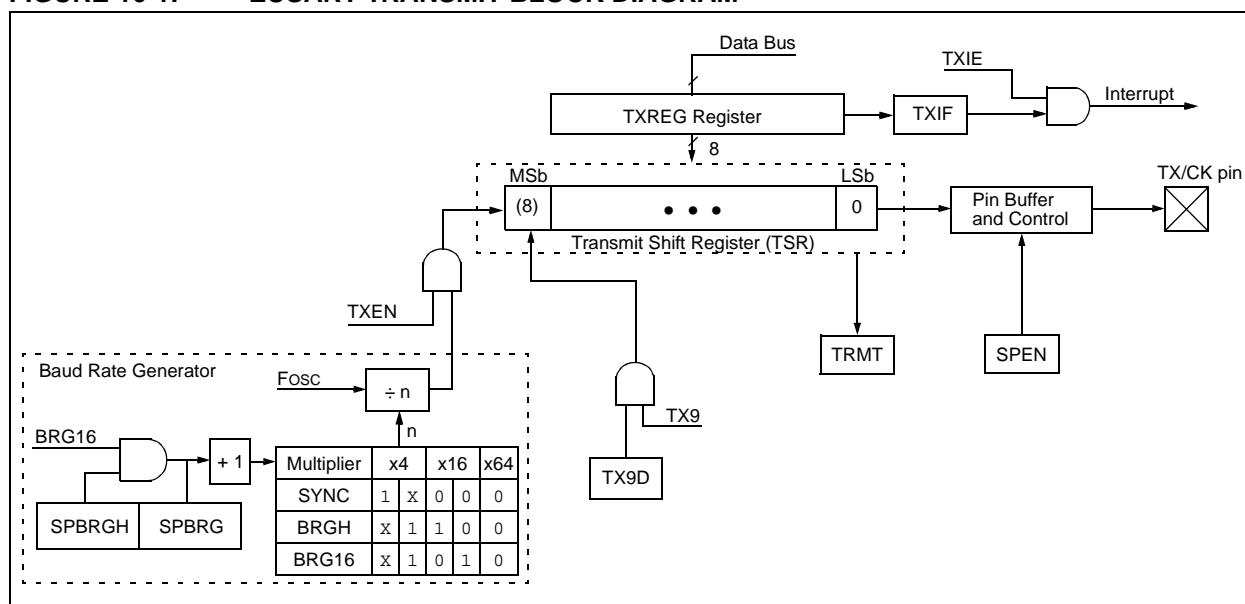
- Full-duplex asynchronous transmit and receive
  - Two-character input buffer
  - One-character output buffer
  - Programmable 8-bit or 9-bit character length
  - Address detection in 9-bit mode
  - Input buffer overrun error detection
  - Received character framing error detection
  - Half-duplex synchronous master
  - Half-duplex synchronous slave
  - Programmable clock and data polarity

The EUSART module implements the following additional features, making it ideally suited for use in Local Interconnect Network (LIN) bus systems:

- Automatic detection and calibration of the baud rate
  - Wake-up on Break reception
  - 13-bit Break character transmit

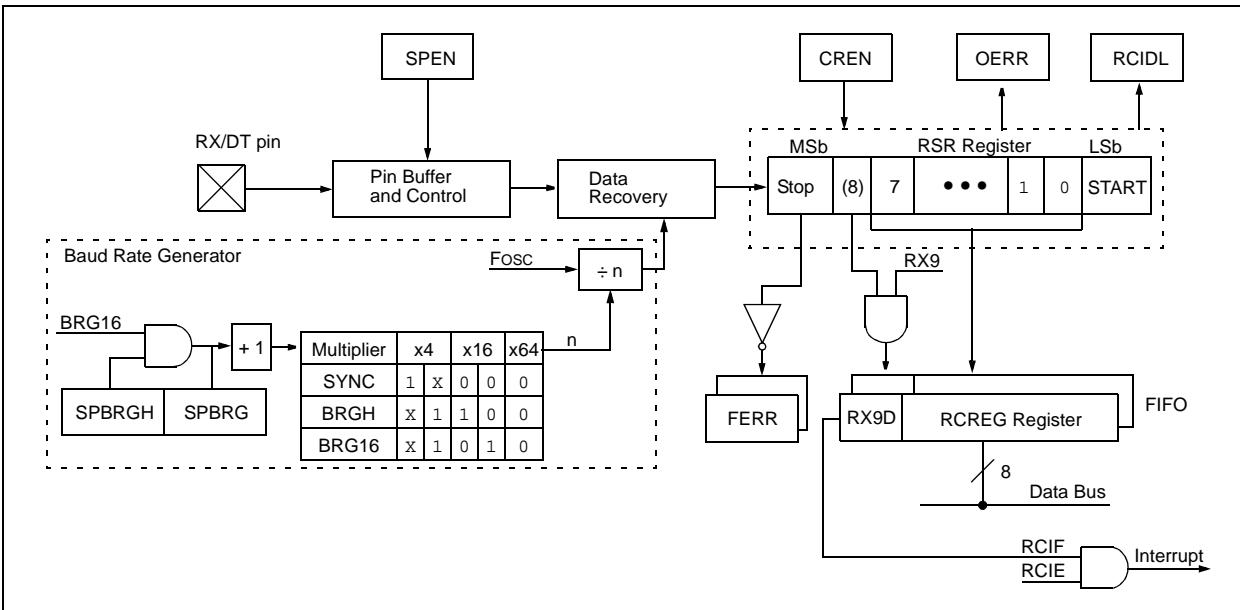
Block diagrams of the EUSART transmitter and receiver are shown in Figure 16-1 and Figure 16-2.

FIGURE 16-1: EUSART TRANSMIT BLOCK DIAGRAM



# PIC18F1XK50/PIC18LF1XK50

FIGURE 16-2: EUSART RECEIVE BLOCK DIAGRAM



The operation of the EUSART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCTL)

These registers are detailed in Register 16-1, Register 16-2 and Register 16-3, respectively.

For all modes of EUSART operation, the TRIS control bits corresponding to the RX/DT and TX/CK pins should be set to '1'. The EUSART control will automatically reconfigure the pin from input to output, as needed.

## 16.1 EUSART Asynchronous Mode

The EUSART transmits and receives data using the standard non-return-to-zero (NRZ) format. NRZ is implemented with two levels: a VOH mark state which represents a '1' data bit, and a VOL space state which represents a '0' data bit. NRZ refers to the fact that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port idles in the mark state. Each character transmission consists of one Start bit followed by eight or nine data bits and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is 8 bits. Each transmitted bit persists for a period of 1/(Baud Rate). An on-chip dedicated 8-bit/16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See Table 16-5 for examples of baud rate configurations.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is not supported by the hardware, but can be implemented in software and stored as the ninth data bit.

### 16.1.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 16-1. The heart of the transmitter is the serial Transmit Shift Register (TSR), which is not directly accessible by software. The TSR obtains its data from the transmit buffer, which is the TXREG register.

#### 16.1.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- TXEN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the TXEN bit of the TXSTA register enables the transmitter circuitry of the EUSART. Clearing the SYNC bit of the TXSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTA register enables the EUSART and automatically configures the TX/CK I/O pin as an output. If the TX/CK pin is shared with an analog peripheral the analog I/O function must be disabled by clearing the corresponding ANSEL bit.

**Note 1:** When the SPEN bit is set the RX/DT I/O pin is automatically configured as an input, regardless of the state of the corresponding TRIS bit and whether or not the EUSART receiver is enabled. The RX/DT pin data can be read via a normal PORT read but PORT latch data output is precluded.

**2:** The TXIF transmitter interrupt flag is set when the TXEN enable bit is set.

#### 16.1.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREG is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the TXREG until the Stop bit of the previous character has been transmitted. The pending character in the TXREG is then transferred to the TSR in one Tcy immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXREG.

#### 16.1.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the CKTXP bit of the BAUDCON register. The default state of this bit is '0' which selects high true transmit idle and data bits. Setting the CKTXP bit to '1' will invert the transmit data resulting in low true idle and data bits. The CKTXP bit controls transmit data polarity only in Asynchronous mode. In Synchronous mode the CKTXP bit has a different function.

#### 16.1.1.4 Transmit Interrupt Flag

The TXIF interrupt flag bit of the PIR1 register is set whenever the EUSART transmitter is enabled and no character is being held for transmission in the TXREG. In other words, the TXIF bit is only clear when the TSR is busy with a character and a new character has been queued for transmission in the TXREG. The TXIF flag bit is not cleared immediately upon writing TXREG. TXIF becomes valid in the second instruction cycle following the write execution. Polling TXIF immediately following the TXREG write will return invalid results. The TXIF bit is read-only, it cannot be set or cleared by software.

The TXIF interrupt can be enabled by setting the TXIE interrupt enable bit of the PIE1 register. However, the TXIF flag bit will be set whenever the TXREG is empty, regardless of the state of TXIE enable bit.

To use interrupts when transmitting data, set the TXIE bit only when there is more data to send. Clear the TXIE interrupt enable bit upon writing the last character of the transmission to the TXREG.

## 16.1.1.5 TSR Status

The TRMT bit of the TXSTA register indicates the status of the TSR register. This is a read-only bit. The TRMT bit is set when the TSR register is empty and is cleared when a character is transferred to the TSR register from the TXREG. The TRMT bit remains clear until all bits have been shifted out of the TSR register. No interrupt logic is tied to this bit, so the user needs to poll this bit to determine the TSR status.

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

## 16.1.1.6 Transmitting 9-Bit Characters

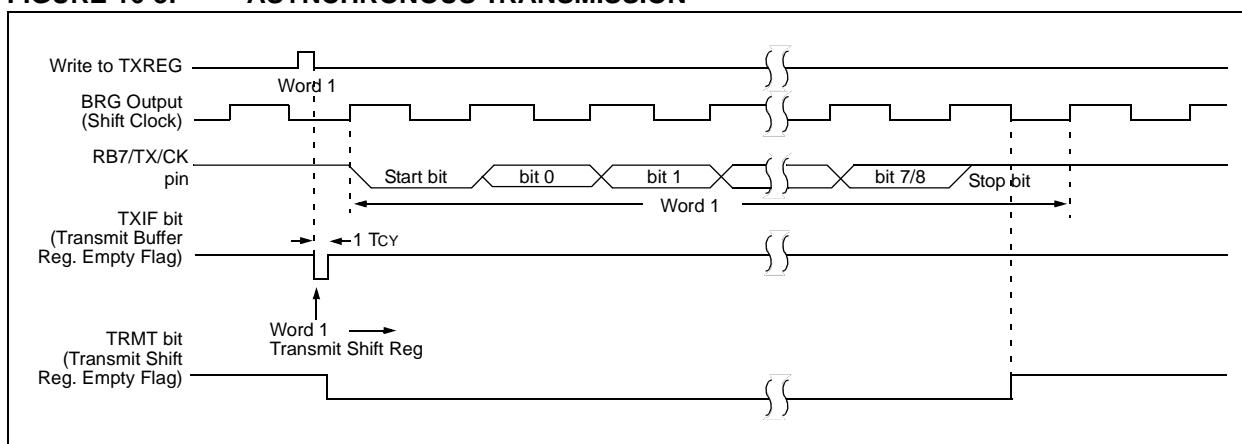
The EUSART supports 9-bit character transmissions. When the TX9 bit of the TXSTA register is set the EUSART will shift 9 bits out for each character transmitted. The TX9D bit of the TXSTA register is the ninth, and Most Significant, data bit. When transmitting 9-bit data, the TX9D data bit must be written before writing the 8 Least Significant bits into the TXREG. All nine bits of data will be transferred to the TSR shift register immediately after the TXREG is written.

A special 9-bit Address mode is available for use with multiple receivers. See **Section 16.1.2.8 “Address Detection”** for more information on the Address mode.

## 16.1.1.7 Asynchronous Transmission Set-up:

1. Initialize the SPBRGH:SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 16.3 “EUSART Baud Rate Generator (BRG)”**).
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If 9-bit transmission is desired, set the TX9 control bit. A set ninth data bit will indicate that the 8 Least Significant data bits are an address when the receiver is set for address detection.
4. Set the CKTXP control bit if inverted transmit data polarity is desired.
5. Enable the transmission by setting the TXEN control bit. This will cause the TXIF interrupt bit to be set.
6. If interrupts are desired, set the TXIE interrupt enable bit. An interrupt will occur immediately provided that the GIE and PEIE bits of the INT-CON register are also set.
7. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
8. Load 8-bit data into the TXREG register. This will start the transmission.

**FIGURE 16-3: ASYNCHRONOUS TRANSMISSION**



# PIC18F1XK50/PIC18LF1XK50

FIGURE 16-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)

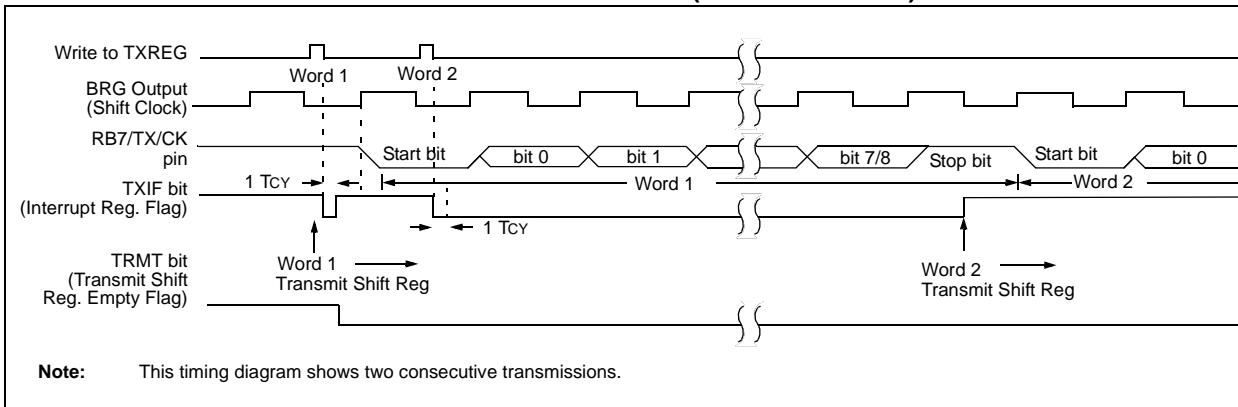


TABLE 16-1: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	281
TXREG	EUSART Transmit Register								281
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	281
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	281
SPBRGH	EUSART Baud Rate Generator Register, High Byte								281
SPBRG	EUSART Baud Rate Generator Register, Low Byte								281

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

## 16.1.2 EUSART ASYNCHRONOUS RECEIVER

The Asynchronous mode would typically be used in RS-232 systems. The receiver block diagram is shown in Figure 16-2. The data is received on the RX/DT pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all 8 or 9 bits of the character have been shifted in, they are immediately transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must start servicing the EUSART receiver. The FIFO and RSR registers are not directly accessible by software. Access to the received data is via the RCREG register.

### 16.1.2.1 Enabling the Receiver

The EUSART receiver is enabled for asynchronous operation by configuring the following three control bits:

- CREN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the CREN bit of the RCSTA register enables the receiver circuitry of the EUSART. Clearing the SYNC bit of the TXSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTA register enables the EUSART. The RX/DT I/O pin must be configured as an input by setting the corresponding TRIS control bit. If the RX/DT pin is shared with an analog peripheral the analog I/O function must be disabled by clearing the corresponding ANSEL bit.

**Note:** When the SPEN bit is set the TX/CK I/O pin is automatically configured as an output, regardless of the state of the corresponding TRIS bit and whether or not the EUSART transmitter is enabled. The PORT latch is disconnected from the output driver so it is not possible to use the TX/CK pin as a general purpose output.

### 16.1.2.2 Receiving Data

The receiver data recovery circuit initiates character reception on the falling edge of the first bit. The first bit, also known as the Start bit, is always a zero. The data recovery circuit counts one-half bit time to the center of the Start bit and verifies that the bit is still a zero. If it is not a zero then the data recovery circuit aborts character reception, without generating an error, and resumes looking for the falling edge of the Start bit. If the Start bit zero verification succeeds then the data recovery circuit counts a full bit time to the center of the next bit. The bit is then sampled by a majority detect circuit and the resulting '0' or '1' is shifted into the RSR. This repeats until all data bits have been sampled and shifted into the RSR. One final bit time is measured and the level sampled. This is the Stop bit, which is always a '1'. If the data recovery circuit samples a '0' in the Stop bit position then a framing error is set for this character, otherwise the framing error is cleared for this character. See **Section 16.1.2.5 "Receive Framing Error"** for more information on framing errors.

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the EUSART receive FIFO and the RCIF interrupt flag bit of the PIR1 register is set. The top character in the FIFO is transferred out of the FIFO by reading the RCREG register.

**Note:** If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See **Section 16.1.2.6 "Receive Overrun Error"** for more information on overrun errors.

### 16.1.2.3 Receive Data Polarity

The polarity of the receive data can be controlled with the DTRXP bit of the BAUDCON register. The default state of this bit is '0' which selects high true receive idle and data bits. Setting the DTRXP bit to '1' will invert the receive data resulting in low true idle and data bits. The DTRXP bit controls receive data polarity only in Asynchronous mode. In synchronous mode the DTRXP bit has a different function.

## 16.1.2.4 Receive Interrupts

The RCIF interrupt flag bit of the PIR1 register is set whenever the EUSART receiver is enabled and there is an unread character in the receive FIFO. The RCIF interrupt flag bit is read-only, it cannot be set or cleared by software.

RCIF interrupts are enabled by setting the following bits:

- RCIE interrupt enable bit of the PIE1 register
- PEIE peripheral interrupt enable bit of the INTCON register
- GIE global interrupt enable bit of the INTCON register

The RCIF interrupt flag bit will be set when there is an unread character in the FIFO, regardless of the state of interrupt enable bits.

## 16.1.2.5 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error status bit. A framing error indicates that a Stop bit was not seen at the expected time. The framing error status is accessed via the FERR bit of the RCSTA register. The FERR bit represents the status of the top unread character in the receive FIFO. Therefore, the FERR bit must be read before reading the RCREG.

The FERR bit is read-only and only applies to the top unread character in the receive FIFO. A framing error (FERR = 1) does not preclude reception of additional characters. It is not necessary to clear the FERR bit. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERR bit can be forced clear by clearing the SPEN bit of the RCSTA register which resets the EUSART. Clearing the CREN bit of the RCSTA register does not affect the FERR bit. A framing error by itself does not generate an interrupt.

**Note:** If all receive characters in the receive FIFO have framing errors, repeated reads of the RCREG will not clear the FERR bit.

## 16.1.2.6 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before the FIFO is accessed. When this happens the OERR bit of the RCSTA register is set. The characters already in the FIFO buffer can be read but no additional characters will be received until the error is cleared. The error must be cleared by either clearing the CREN bit of the RCSTA register or by resetting the EUSART by clearing the SPEN bit of the RCSTA register.

## 16.1.2.7 Receiving 9-bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTA register is set, the EUSART will shift 9 bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth and Most Significant data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the 8 Least Significant bits from the RCREG.

## 16.1.2.8 Address Detection

A special Address Detection mode is available for use when multiple receivers share the same transmission line, such as in RS-485 systems. Address detection is enabled by setting the ADDEN bit of the RCSTA register.

Address detection requires 9-bit character reception. When address detection is enabled, only characters with the ninth data bit set will be transferred to the receive FIFO buffer, thereby setting the RCIF interrupt bit. All other characters will be ignored.

Upon receiving an address character, user software determines if the address matches its own. Upon address match, user software must disable address detection by clearing the ADDEN bit before the next Stop bit occurs. When user software detects the end of the message, determined by the message protocol used, software places the receiver back into the Address Detection mode by setting the ADDEN bit.

# PIC18F1XK50/PIC18LF1XK50

## 16.1.2.9 Asynchronous Reception Set-up:

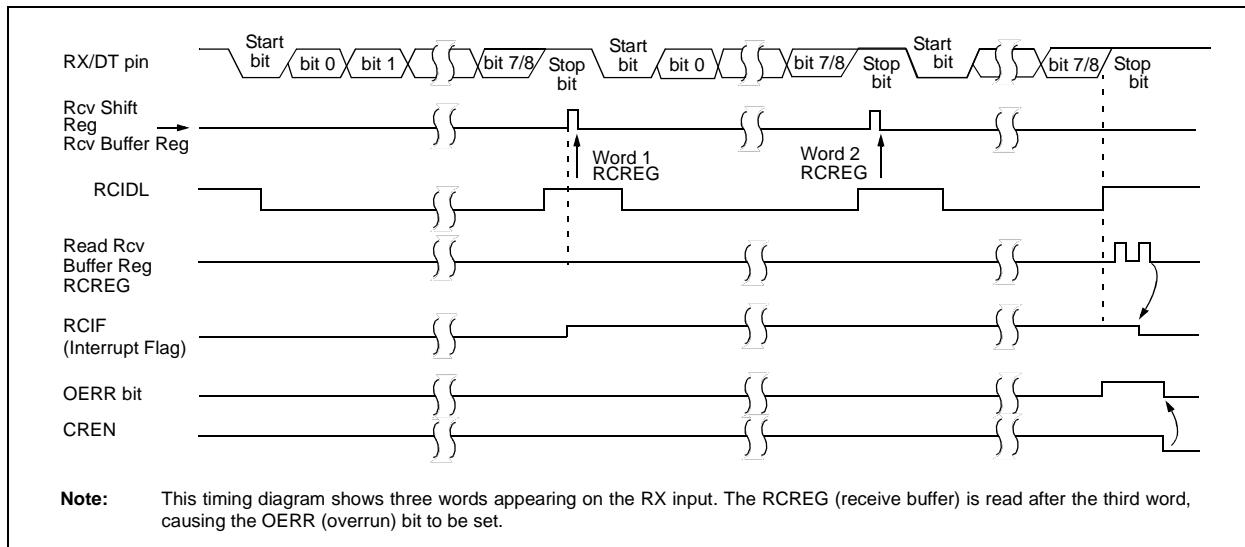
1. Initialize the SPBRGH:SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 16.3 “EUSART Baud Rate Generator (BRG)”**).
2. Enable the serial port by setting the SPEN bit and the RX/DT pin TRIS bit. The SYNC bit must be clear for asynchronous operation.
3. If interrupts are desired, set the RCIE interrupt enable bit and set the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Set the DTRXP if inverted receive polarity is desired.
6. Enable reception by setting the CREN bit.
7. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
8. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
9. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register.
10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

## 16.1.2.10 9-bit Address Detection Mode Set-up

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

1. Initialize the SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 16.3 “EUSART Baud Rate Generator (BRG)”**).
2. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
3. If interrupts are desired, set the RCIE interrupt enable bit and set the GIE and PEIE bits of the INTCON register.
4. Enable 9-bit reception by setting the RX9 bit.
5. Enable address detection by setting the ADDEN bit.
6. Set the DTRXP if inverted receive polarity is desired.
7. Enable reception by setting the CREN bit.
8. The RCIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
9. Read the RCSTA register to get the error flags. The ninth data bit will always be set.
10. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register. Software determines if this is the device's address.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

**FIGURE 16-5: ASYNCHRONOUS RECEPTION**



# PIC18F1XK50/PIC18LF1XK50

TABLE 16-2: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	281
RCREG	EUSART Receive Register								281
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282
TXSTA	CSRC	TX9	TXEN	SYNC	SEND <sub>B</sub>	BRGH	TRMT	TX9D	281
BAUDCON	ABDOVF	RCIDL	DTRXP	CKT <sub>XP</sub>	BRG16	—	WUE	ABDEN	281
SPBRGH	EUSART Baud Rate Generator Register, High Byte								281
SPBRG	EUSART Baud Rate Generator Register, Low Byte								281

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

# PIC18F1XK50/PIC18LF1XK50

## 16.2 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (HFINTOSC). However, the HFINTOSC frequency may drift as V<sub>DD</sub> or temperature changes, and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the HFINTOSC output. Adjusting the value in the OSCTUNE register allows for fine resolution changes to the system clock source. See **Section 2.6.1 “OSCTUNE Register”** for more information.

The other method adjusts the value in the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see **Section 16.3.1 “Auto-Baud Detect”**). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change in the peripheral clock frequency.

## REGISTER 16-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN <sup>(1)</sup>	SYNC	SEND B	BRGH	TRMT	TX9D
bit 7							

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7      **CSRC:** Clock Source Select bit

#### Asynchronous mode:

Don't care

#### Synchronous mode:

1 = Master mode (clock generated internally from BRG)  
0 = Slave mode (clock from external source)

bit 6      **TX9:** 9-bit Transmit Enable bit

1 = Selects 9-bit transmission  
0 = Selects 8-bit transmission

bit 5      **TXEN:** Transmit Enable bit<sup>(1)</sup>

1 = Transmit enabled  
0 = Transmit disabled

bit 4      **SYNC:** EUSART Mode Select bit

1 = Synchronous mode  
0 = Asynchronous mode

bit 3      **SEND B:** Send Break Character bit

#### Asynchronous mode:

1 = Send Sync Break on next transmission (cleared by hardware upon completion)  
0 = Sync Break transmission completed

#### Synchronous mode:

Don't care

bit 2      **BRGH:** High Baud Rate Select bit

#### Asynchronous mode:

1 = High speed  
0 = Low speed

#### Synchronous mode:

Unused in this mode

bit 1      **TRMT:** Transmit Shift Register Status bit

1 = TSR empty  
0 = TSR full

bit 0      **TX9D:** Ninth bit of Transmit Data

Can be address/data bit or a parity bit.

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

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 16-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER<sup>(1)</sup>

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>SPEN:</b> Serial Port Enable bit 1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins) 0 = Serial port disabled (held in Reset)
bit 6	<b>RX9:</b> 9-bit Receive Enable bit 1 = Selects 9-bit reception 0 = Selects 8-bit reception
bit 5	<b>SREN:</b> Single Receive Enable bit <u>Asynchronous mode:</u> Don't care <u>Synchronous mode – Master:</u> 1 = Enables single receive 0 = Disables single receive This bit is cleared after reception is complete. <u>Synchronous mode – Slave</u> Don't care
bit 4	<b>CREN:</b> Continuous Receive Enable bit <u>Asynchronous mode:</u> 1 = Enables receiver 0 = Disables receiver <u>Synchronous mode:</u> 1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN) 0 = Disables continuous receive
bit 3	<b>ADDEN:</b> Address Detect Enable bit <u>Asynchronous mode 9-bit (RX9 = 1):</u> 1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set 0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit <u>Asynchronous mode 8-bit (RX9 = 0):</u> Don't care
bit 2	<b>FERR:</b> Framing Error bit 1 = Framing error (can be updated by reading RCREG register and receive next valid byte) 0 = No framing error
bit 1	<b>OERR:</b> Overrun Error bit 1 = Overrun error (can be cleared by clearing bit CREN) 0 = No overrun error
bit 0	<b>RX9D:</b> Ninth bit of Received Data This can be address/data bit or a parity bit and must be calculated by user firmware.

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 16-3: BAUDCON: BAUD RATE CONTROL REGISTER

R-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **ABDOVF**: Auto-Baud Detect Overflow bit

#### Asynchronous mode:

1 = Auto-baud timer overflowed

0 = Auto-baud timer did not overflow

#### Synchronous mode:

Don't care

bit 6 **RCIDL**: Receive Idle Flag bit

#### Asynchronous mode:

1 = Receiver is Idle

0 = Start bit has been detected and the receiver is active

#### Synchronous mode:

Don't care

bit 5 **DTRXP**: Data/Receive Polarity Select bit

#### Asynchronous mode:

1 = Receive data (RX) is inverted (active-low)

0 = Receive data (RX) is not inverted (active-high)

#### Synchronous mode:

1 = Data (DT) is inverted (active-low)

0 = Data (DT) is not inverted (active-high)

bit 4 **CKTXP**: Clock/Transmit Polarity Select bit

#### Asynchronous mode:

1 = Idle state for transmit (TX) is low

0 = Idle state for transmit (TX) is high

#### Synchronous mode:

1 = Data changes on the falling edge of the clock and is sampled on the rising edge of the clock

0 = Data changes on the rising edge of the clock and is sampled on the falling edge of the clock

bit 3 **BRG16**: 16-bit Baud Rate Generator bit

1 = 16-bit Baud Rate Generator is used (SPBRGH:SPBRG)

0 = 8-bit Baud Rate Generator is used (SPBRG)

bit 2 **Unimplemented**: Read as '0'

**WUE**: Wake-up Enable bit

#### Asynchronous mode:

1 = Receiver is waiting for a falling edge. No character will be received but RCIF will be set on the falling edge. WUE will automatically clear on the rising edge.

0 = Receiver is operating normally

#### Synchronous mode:

Don't care

bit 0 **ABDEN**: Auto-Baud Detect Enable bit

#### Asynchronous mode:

1 = Auto-Baud Detect mode is enabled (clears when auto-baud is complete)

0 = Auto-Baud Detect mode is disabled

#### Synchronous mode:

Don't care

## 16.3 EUSART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is an 8-bit or 16-bit timer that is dedicated to the support of both the asynchronous and synchronous EUSART operation. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit of the BAUDCON register selects 16-bit mode.

The SPBRGH:SPBRG register pair determines the period of the free running baud rate timer. In Asynchronous mode the multiplier of the baud rate period is determined by both the BRGH bit of the TXSTA register and the BRG16 bit of the BAUDCON register. In Synchronous mode, the BRGH bit is ignored.

Table 16-3 contains the formulas for determining the baud rate. Example 16-1 provides a sample calculation for determining the baud rate and baud rate error.

Typical baud rates and error values for various asynchronous modes have been computed for your convenience and are shown in Table 16-5. It may be advantageous to use the high baud rate (BRGH = 1), or the 16-bit BRG (BRG16 = 1) to reduce the baud rate error. The 16-bit BRG mode is used to achieve slow baud rates for fast oscillator frequencies.

Writing a new value to the SPBRGH, SPBRG register pair causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RCIDL bit to make sure that the receive operation is Idle before changing the system clock.

### EXAMPLE 16-1: CALCULATING BAUD RATE ERROR

For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$\text{Desired Baud Rate} = \frac{FOSC}{64(\text{SPBRGH:SPBRG} + 1)}$$

Solving for SPBRGH:SPBRG:

$$\begin{aligned} X &= \left( \frac{FOSC}{64 * (\text{Desired Baud Rate})} \right)^{-1} \\ &= \left( \frac{16,000,000}{64 * 9600} \right)^{-1} \\ &= [25.042] = 25 \\ \text{Calculated Baud Rate} &= \frac{16000000}{64(25 + 1)} \\ &= 9615 \\ \text{Error} &= \frac{\text{Calc. Baud Rate} - \text{Desired Baud Rate}}{\text{Desired Baud Rate}} \\ &= \frac{(9615 - 9600)}{9600} = 0.16\% \end{aligned}$$

TABLE 16-3: BAUD RATE FORMULAS

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

Legend: x = Don't care, n = value of SPBRGH, SPBRG register pair

TABLE 16-4: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TXSTA	CSRC	TX9	TXEN	SYNC	SENDDB	BRGH	TRMT	TX9D	281
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	281
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	281
SPBRGH	EUSART Baud Rate Generator Register, High Byte								281
SPBRG	EUSART Baud Rate Generator Register, Low Byte								281

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

# PIC18F1XK50/PIC18LF1XK50

TABLE 16-5: BAUD RATES FOR ASYNCHRONOUS MODES

BAUD RATE	SYNC = 0, BRGH = 0, BRG16 = 0											
	Fosc = 48.000 MHz			Fosc = 18.432 MHz			Fosc = 12.000 MHz			Fosc = 11.0592 MHz		
	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)
300	—	—	—	—	—	—	—	—	—	—	—	—
1200	—	—	—	1200	0.00	239	1202	0.16	155	1200	0.00	143
2400	—	—	—	2400	0.00	119	2404	0.16	77	2400	0.00	71
9600	9615	0.16	77	9600	0.00	29	9375	-2.34	19	9600	0.00	17
10417	10417	0.00	71	10286	-1.26	27	10417	0.00	17	10165	-2.42	16
19.2k	19.23k	0.16	38	19.20k	0.00	14	18.75k	-2.34	9	19.20k	0.00	8
57.6k	57.69k	0.16	12	57.60k	0.00	7	—	—	—	57.60k	0.00	2
115.2k	—	—	—	—	—	—	—	—	—	—	—	—

BAUD RATE	SYNC = 0, BRGH = 0, BRG16 = 0											
	Fosc = 8.000 MHz			Fosc = 4.000 MHz			Fosc = 3.6864 MHz			Fosc = 1.000 MHz		
	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)
300	—	—	—	300	0.16	207	300	0.00	191	300	0.16	51
1200	1202	0.16	103	1202	0.16	51	1200	0.00	47	1202	0.16	12
2400	2404	0.16	51	2404	0.16	25	2400	0.00	23	—	—	—
9600	9615	0.16	12	—	—	—	9600	0.00	5	—	—	—
10417	10417	0.00	11	10417	0.00	5	—	—	—	—	—	—
19.2k	—	—	—	—	—	—	19.20k	0.00	2	—	—	—
57.6k	—	—	—	—	—	—	57.60k	0.00	0	—	—	—
115.2k	—	—	—	—	—	—	—	—	—	—	—	—

BAUD RATE	SYNC = 0, BRGH = 1, BRG16 = 0											
	Fosc = 48.000 MHz			Fosc = 18.432 MHz			Fosc = 12.000 MHz			Fosc = 11.0592 MHz		
	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)
300	—	—	—	—	—	—	—	—	—	—	—	—
1200	—	—	—	—	—	—	—	—	—	—	—	—
2400	—	—	—	—	—	—	—	—	—	—	—	—
9600	—	—	—	9600	0.00	119	9615	0.16	77	9600	0.00	71
10417	—	—	—	10378	-0.37	110	10417	0.00	71	10473	0.53	65
19.2k	19.23k	0.16	155	19.20k	0.00	59	19.23k	0.16	38	19.20k	0.00	35
57.6k	57.69k	0.16	51	57.60k	0.00	19	57.69k	0.16	12	57.60k	0.00	11
115.2k	115.38k	0.16	25	115.2k	0.00	9	—	—	—	115.2k	0.00	5

# PIC18F1XK50/PIC18LF1XK50

TABLE 16-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE	SYNC = 0, BRGH = 1, BRG16 = 0											
	Fosc = 8.000 MHz			Fosc = 4.000 MHz			Fosc = 3.6864 MHz			Fosc = 1.000 MHz		
	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)	Actual Rate	% Error	SPBRG value (decimal)
300	—	—	—	—	—	—	—	—	—	300	0.16	207
1200	—	—	—	1202	0.16	207	1200	0.00	191	1202	0.16	51
2400	2404	0.16	207	2404	0.16	103	2400	0.00	95	2404	0.16	25
9600	9615	0.16	51	9615	0.16	25	9600	0.00	23	—	—	—
10417	10417	0.00	47	10417	0.00	23	10473	0.53	21	10417	0.00	5
19.2k	19231	0.16	25	19.23k	0.16	12	19.2k	0.00	11	—	—	—
57.6k	55556	-3.55	8	—	—	—	57.60k	0.00	3	—	—	—
115.2k	—	—	—	—	—	—	115.2k	0.00	1	—	—	—

BAUD RATE	SYNC = 0, BRGH = 0, BRG16 = 1											
	Fosc = 48.000 MHz			Fosc = 18.432 MHz			Fosc = 12.000 MHz			Fosc = 11.0592 MHz		
	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)
300	300.0	0.00	9999	300.0	0.00	3839	300	0.00	2499	300.0	0.00	2303
1200	1200.1	0.00	2499	1200	0.00	959	1200	0.00	624	1200	0.00	575
2400	2400	0.00	1249	2400	0.00	479	2404	0.16	311	2400	0.00	287
9600	9615	0.16	311	9600	0.00	119	9615	0.16	77	9600	0.00	71
10417	10417	0.00	287	10378	-0.37	110	10417	0.00	71	10473	0.53	65
19.2k	19.23k	0.16	155	19.20k	0.00	59	19.23k	0.16	38	19.20k	0.00	35
57.6k	57.69k	0.16	51	57.60k	0.00	19	57.69k	0.16	12	57.60k	0.00	11
115.2k	115.38k	0.16	25	115.2k	0.00	9	—	—	—	115.2k	0.00	5

BAUD RATE	SYNC = 0, BRGH = 0, BRG16 = 1											
	Fosc = 8.000 MHz			Fosc = 4.000 MHz			Fosc = 3.6864 MHz			Fosc = 1.000 MHz		
	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)
300	299.9	-0.02	1666	300.1	0.04	832	300.0	0.00	767	300.5	0.16	207
1200	1199	-0.08	416	1202	0.16	207	1200	0.00	191	1202	0.16	51
2400	2404	0.16	207	2404	0.16	103	2400	0.00	95	2404	0.16	25
9600	9615	0.16	51	9615	0.16	25	9600	0.00	23	—	—	—
10417	10417	0.00	47	10417	0.00	23	10473	0.53	21	10417	0.00	5
19.2k	19.23k	0.16	25	19.23k	0.16	12	19.20k	0.00	11	—	—	—
57.6k	55556	-3.55	8	—	—	—	57.60k	0.00	3	—	—	—
115.2k	—	—	—	—	—	—	115.2k	0.00	1	—	—	—

# PIC18F1XK50/PIC18LF1XK50

TABLE 16-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	Fosc = 48.000 MHz			Fosc = 18.432 MHz			Fosc = 12.000 MHz			Fosc = 11.0592 MHz		
	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)
300	300	0.00	39999	300.0	0.00	15359	300	0.00	9999	300.0	0.00	9215
1200	1200	0.00	9999	1200	0.00	3839	1200	0.00	2499	1200	0.00	2303
2400	2400	0.00	4999	2400	0.00	1919	2400	0.00	1249	2400	0.00	1151
9600	9600	0.00	1249	9600	0.00	479	9615	0.16	311	9600	0.00	287
10417	10417	0.00	1151	10425	0.08	441	10417	0.00	287	10433	0.16	264
19.2k	19.20k	0.00	624	19.20k	0.00	239	19.23k	0.16	155	19.20k	0.00	143
57.6k	57.69k	0.16	207	57.60k	0.00	79	57.69k	0.16	51	57.60k	0.00	47
115.2k	115.38k	0.16	103	115.2k	0.00	39	115.38k	0.16	25	115.2k	0.00	23

BAUD RATE	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	Fosc = 8.000 MHz			Fosc = 4.000 MHz			Fosc = 3.6864 MHz			Fosc = 1.000 MHz		
	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)	Actual Rate	% Error	SPBRGH :SPBRG (decimal)
300	300.0	0.00	6666	300.0	0.01	3332	300.0	0.00	3071	300.1	0.04	832
1200	1200	-0.02	1666	1200	0.04	832	1200	0.00	767	1202	0.16	207
2400	2401	0.04	832	2398	0.08	416	2400	0.00	383	2404	0.16	103
9600	9615	0.16	207	9615	0.16	103	9600	0.00	95	9615	0.16	25
10417	10417	0.00	191	10417	0.00	95	10473	0.53	87	10417	0.00	23
19.2k	19.23k	0.16	103	19.23k	0.16	51	19.20k	0.00	47	19.23k	0.16	12
57.6k	57.14k	-0.79	34	58.82k	2.12	16	57.60k	0.00	15	—	—	—
115.2k	117.6k	2.12	16	111.1k	-3.55	8	115.2k	0.00	7	—	—	—

### 16.3.1 AUTO-BAUD DETECT

The EUSART module supports automatic detection and calibration of the baud rate.

In the Auto-Baud Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. The Baud Rate Generator is used to time the period of a received 55h (ASCII "U") which is the Sync character for the LIN bus. The unique feature of this character is that it has five rising edges including the Stop bit edge.

Setting the ABDEN bit of the BAUDCON register starts the auto-baud calibration sequence (Figure 16-6). While the ABD sequence takes place, the EUSART state machine is held in Idle. On the first rising edge of the receive line, after the Start bit, the SPBRG begins counting up using the BRG counter clock as shown in Table 16-6. The fifth rising edge will occur on the RX pin at the end of the eighth bit period. At that time, an accumulated value totaling the proper BRG period is left in the SPBRGH:SPBRG register pair, the ABDEN bit is automatically cleared, and the RCIF interrupt flag is set. A read operation on the RCREG needs to be performed to clear the RCIF interrupt. RCREG content should be discarded. When calibrating for modes that do not use the SPBRGH register the user can verify that the SPBRG register did not overflow by checking for 00h in the SPBRGH register.

The BRG auto-baud clock is determined by the BRG16 and BRGH bits as shown in Table 16-6. During ABD, both the SPBRGH and SPBRG registers are used as a 16-bit counter, independent of the BRG16 bit setting. While calibrating the baud rate period, the SPBRGH

and SPBRG registers are clocked at 1/8th the BRG base clock rate. The resulting byte measurement is the average bit time when clocked at full speed.

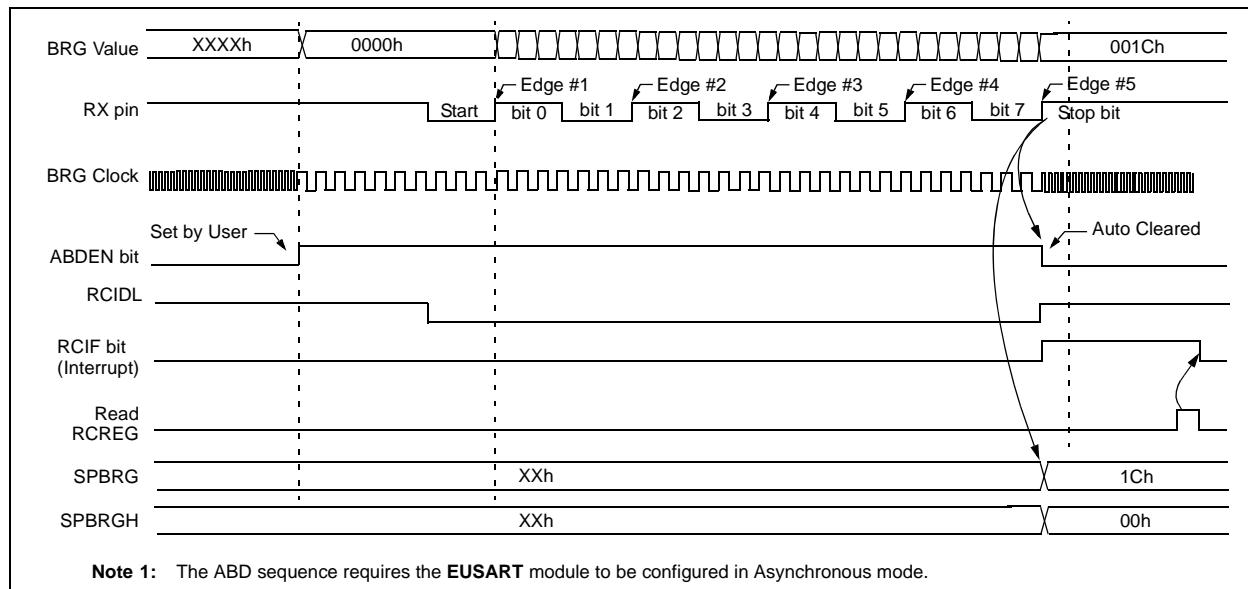
- Note 1:** If the WUE bit is set with the ABDEN bit, auto-baud detection will occur on the byte following the Break character (see [Section 16.3.3 "Auto-Wake-up on Break"](#)).
- 2:** It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible.
- 3:** During the auto-baud process, the auto-baud counter starts counting at 1. Upon completion of the auto-baud sequence, to achieve maximum accuracy, subtract 1 from the SPBRGH:SPBRG register pair.

**TABLE 16-6: BRG COUNTER CLOCK RATES**

BRG16	BRGH	BRG Base Clock	BRG ABD Clock
0	0	Fosc/64	Fosc/512
0	1	Fosc/16	Fosc/128
1	0	Fosc/16	Fosc/128
1	1	Fosc/4	Fosc/32

**Note:** During the ABD sequence, SPBRG and SPBRGH registers are both used as a 16-bit counter, independent of BRG16 setting.

**FIGURE 16-6: AUTOMATIC BAUD RATE CALIBRATION**



# PIC18F1XK50/PIC18LF1XK50

---

## 16.3.2 AUTO-BAUD OVERFLOW

During the course of automatic baud detection, the ABDOVF bit of the BAUDCON register will be set if the baud rate counter overflows before the fifth rising edge is detected on the RX pin. The ABDOVF bit indicates that the counter has exceeded the maximum count that can fit in the 16 bits of the SPBRGH:SPBRG register pair. After the ABDOVF has been set, the counter continues to count until the fifth rising edge is detected on the RX pin. Upon detecting the fifth RX edge, the hardware will set the RCIF Interrupt Flag and clear the ABDEN bit of the BAUDCON register. The RCIF flag can be subsequently cleared by reading the RCREG register. The ABDOVF flag of the BAUDCON register can be cleared by software directly.

To terminate the auto-baud process before the RCIF flag is set, clear the ABDEN bit then clear the ABDOVF bit of the BAUDCON register. The ABDOVF bit will remain set if the ABDEN bit is not cleared first.

## 16.3.3 AUTO-WAKE-UP ON BREAK

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the Baud Rate Generator is inactive and a proper character reception cannot be performed. The Auto-Wake-up feature allows the controller to wake-up due to activity on the RX/DT line. This feature is available only in Asynchronous mode.

The Auto-Wake-up feature is enabled by setting the WUE bit of the BAUDCON register. Once set, the normal receive sequence on RX/DT is disabled, and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a Sync Break or a wake-up signal character for the LIN protocol.)

The EUSART module generates an RCIF interrupt coincident with the wake-up event. The interrupt is generated synchronously to the Q clocks in normal CPU operating modes (Figure 16-7), and asynchronously if the device is in Sleep mode (Figure 16-8). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared by the low-to-high transition on the RX line at the end of the Break. This signals to the user that the Break event is over. At this point, the EUSART module is in Idle mode waiting to receive the next character.

## 16.3.3.1 Special Considerations

### Break Character

To avoid character errors or character fragments during a wake-up event, the wake-up character must be all zeros.

When the wake-up is enabled the function works independent of the low time on the data stream. If the WUE bit is set and a valid non-zero character is received, the low time from the Start bit to the first rising edge will be interpreted as the wake-up event. The remaining bits in the character will be received as a fragmented character and subsequent characters can result in framing or overrun errors.

Therefore, the initial character in the transmission must be all '0's. This must be 10 or more bit times, 13-bit times recommended for LIN bus, or any number of bit times for standard RS-232 devices.

### Oscillator Startup Time

Oscillator start-up time must be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL mode). The Sync Break (or wake-up signal) character must be of sufficient length, and be followed by a sufficient interval, to allow enough time for the selected oscillator to start and provide proper initialization of the EUSART.

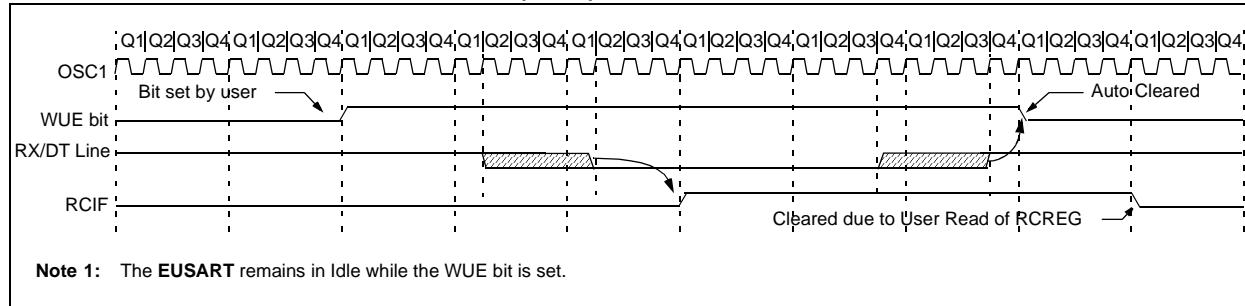
### WUE Bit

The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared by hardware by a rising edge on RX/DT. The interrupt condition is then cleared by software by reading the RCREG register and discarding its contents.

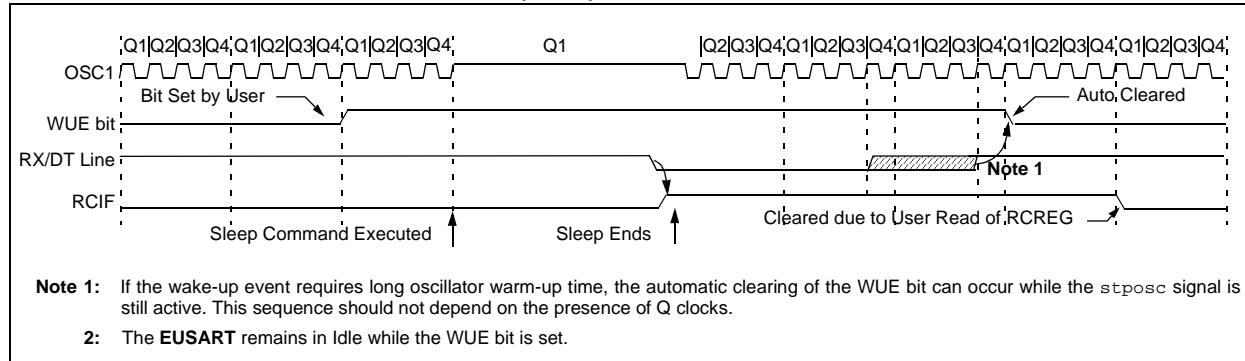
To ensure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process before setting the WUE bit. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

# PIC18F1XK50/PIC18LF1XK50

**FIGURE 16-7: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION**



**FIGURE 16-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**



# PIC18F1XK50/PIC18LF1XK50

## 16.3.4 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. A Break character consists of a Start bit, followed by 12 '0' bits and a Stop bit.

To send a Break character, set the SENDB and TXEN bits of the TXSTA register. The Break character transmission is then initiated by a write to the TXREG. The value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

The TRMT bit of the TXSTA register indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 16-9 for the timing of the Break character sequence.

### 16.3.4.1 Break and Sync Transmit Sequence

The following sequence will start a message frame header made up of a Break, followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to enable the Break sequence.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.

5. After the Break has been sent, the SENDB bit is reset by hardware and the Sync character is then transmitted.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

## 16.3.5 RECEIVING A BREAK CHARACTER

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

The first method to detect a Break character uses the FERR bit of the RCSTA register and the Received data as indicated by RCREG. The Baud Rate Generator is assumed to have been initialized to the expected baud rate.

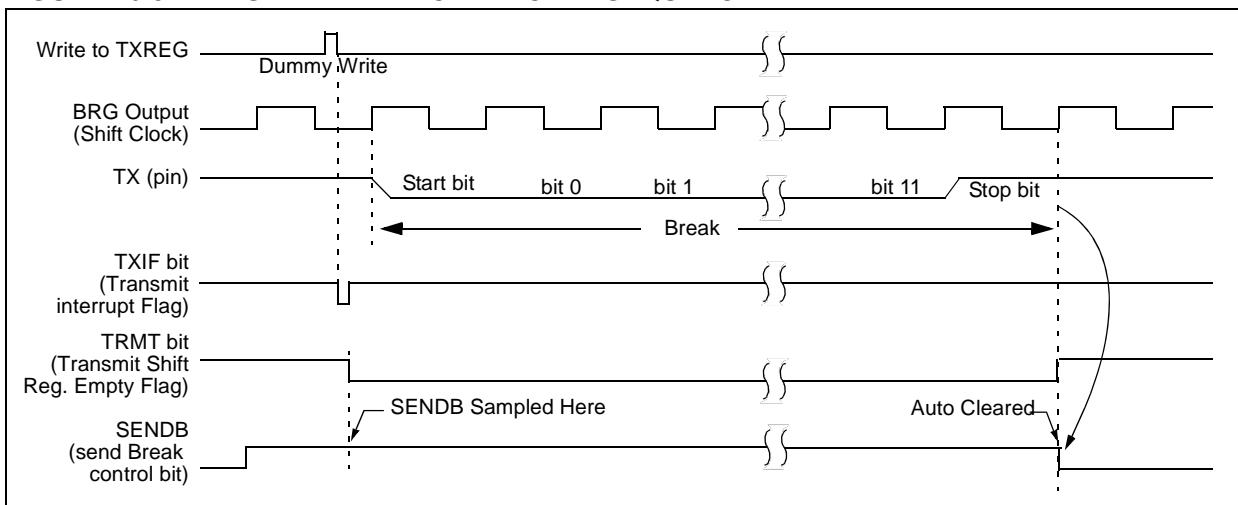
A Break character has been received when;

- RCIF bit is set
- FERR bit is set
- RCREG = 00h

The second method uses the Auto-Wake-up feature described in **Section 16.3.3 “Auto-Wake-up on Break”**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt, and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Detect feature. For both methods, the user can set the ABDEN bit of the BAUDCON register before placing the EUSART in Sleep mode.

**FIGURE 16-9: SEND BREAK CHARACTER SEQUENCE**



## 16.4 EUSART Synchronous Mode

Synchronous serial communications are typically used in systems with a single master and one or more slaves. The master device contains the necessary circuitry for baud rate generation and supplies the clock for all devices in the system. Slave devices can take advantage of the master clock by eliminating the internal clock generation circuitry.

There are two signal lines in Synchronous mode: a bidirectional data line and a clock line. Slaves use the external clock supplied by the master to shift the serial data into and out of their respective receive and transmit shift registers. Since the data line is bidirectional, synchronous operation is half-duplex only. Half-duplex refers to the fact that master and slave devices can receive and transmit data but not both simultaneously. The EUSART can operate as either a master or slave device.

Start and Stop bits are not used in synchronous transmissions.

### 16.4.1 SYNCHRONOUS MASTER MODE

The following bits are used to configure the EUSART for Synchronous Master operation:

- SYNC = 1
- CSRC = 1
- SREN = 0 (for transmit); SREN = 1 (for receive)
- CREN = 0 (for transmit); CREN = 1 (for receive)
- SPEN = 1

Setting the SYNC bit of the TXSTA register configures the device for synchronous operation. Setting the CSRC bit of the TXSTA register configures the device as a master. Clearing the SREN and CREN bits of the RCSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTA register enables the EUSART. If the RX/DT or TX/CK pins are shared with an analog peripheral the analog I/O functions must be disabled by clearing the corresponding ANSEL bits.

The TRIS bits corresponding to the RX/DT and TX/CK pins should be set.

#### 16.4.1.1 Master Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a master transmits the clock on the TX/CK line. The TX/CK pin output driver is automatically enabled when the EUSART is configured for synchronous transmit or receive operation. Serial data bits change on the leading edge to ensure they are valid at the trailing edge of each clock. One clock cycle is generated for each data bit. Only as many clock cycles are generated as there are data bits.

#### 16.4.1.2 Clock Polarity

A clock polarity option is provided for Microwire compatibility. Clock polarity is selected with the CKTXP bit of the BAUDCON register. Setting the CKTXP bit sets the clock Idle state as high. When the CKTXP bit is set, the data changes on the falling edge of each clock and is sampled on the rising edge of each clock. Clearing the CKTXP bit sets the Idle state as low. When the CKTXP bit is cleared, the data changes on the rising edge of each clock and is sampled on the falling edge of each clock.

#### 16.4.1.3 Synchronous Master Transmission

Data is transferred out of the device on the RX/DT pin. The RX/DT and TX/CK pin output drivers are automatically enabled when the EUSART is configured for synchronous master transmit operation.

A transmission is initiated by writing a character to the TXREG register. If the TSR still contains all or part of a previous character the new character data is held in the TXREG until the last bit of the previous character has been transmitted. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREG is immediately transferred to the TSR. The transmission of the character commences immediately following the transfer of the data to the TSR from the TXREG.

Each data bit changes on the leading edge of the master clock and remains valid until the subsequent leading clock edge.

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

#### 16.4.1.4 Data Polarity

The polarity of the transmit and receive data can be controlled with the DTRXP bit of the BAUDCON register. The default state of this bit is '0' which selects high true transmit and receive data. Setting the DTRXP bit to '1' will invert the data resulting in low true transmit and receive data.

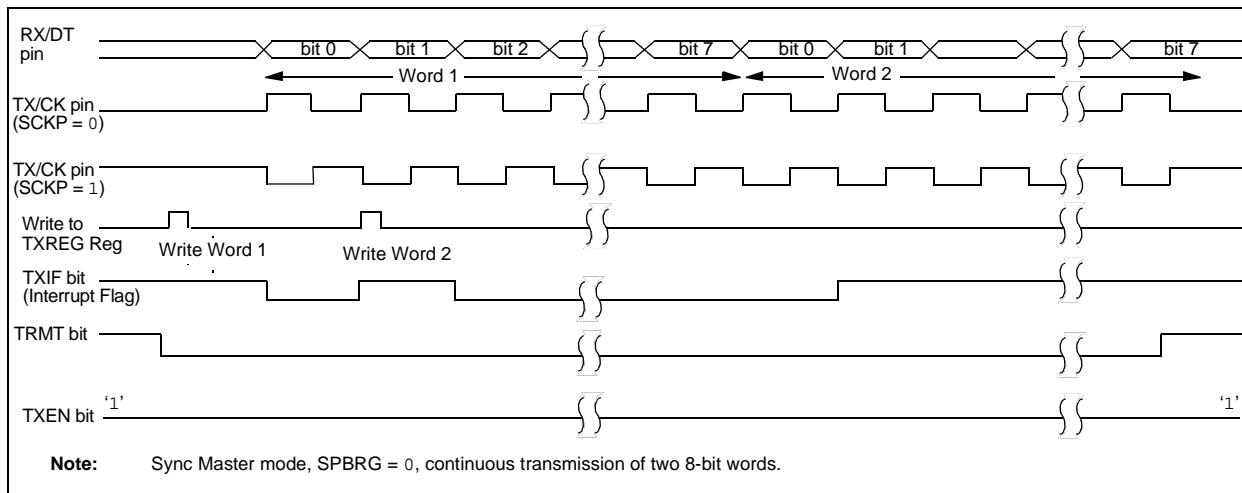
# PIC18F1XK50/PIC18LF1XK50

## 16.4.1.5 Synchronous Master Transmission Set-up:

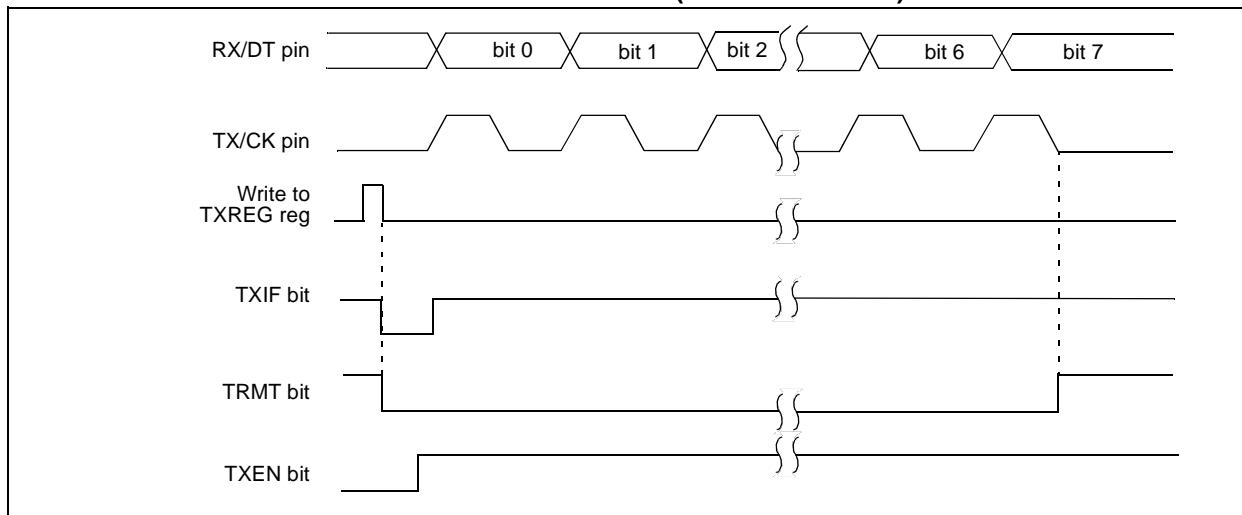
1. Initialize the SPBRGH, SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 16.3 “EUSART Baud Rate Generator (BRG)**”).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC. Set the TRIS bits corresponding to the RX/DT and TX/CK I/O pins.

3. Disable Receive mode by clearing bits SREN and CREN.
4. Enable Transmit mode by setting the TXEN bit.
5. If 9-bit transmission is desired, set the TX9 bit.
6. If interrupts are desired, set the TXIE, GIE and PEIE interrupt enable bits.
7. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
8. Start transmission by loading data to the TXREG register.

**FIGURE 16-10: SYNCHRONOUS TRANSMISSION**



**FIGURE 16-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**



# PIC18F1XK50/PIC18LF1XK50

TABLE 16-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RABIE	TMROIF	INT0IF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	281
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282
TXREG	EUSART Transmit Register								281
TXSTA	CSRC	TX9	TXEN	SYNC	SEND <sub>B</sub>	BRGH	TRMT	TX9D	281
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	281
SPBRGH	EUSART Baud Rate Generator Register, High Byte								281
SPBRG	EUSART Baud Rate Generator Register, Low Byte								281

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

### 16.4.1.6 Synchronous Master Reception

Data is received at the RX/DT pin. The RX/DT pin output driver must be disabled by setting the corresponding TRIS bits when the EUSART is configured for synchronous master receive operation.

In Synchronous mode, reception is enabled by setting either the Single Receive Enable bit (SREN of the RCSTA register) or the Continuous Receive Enable bit (CREN of the RCSTA register).

When SREN is set and CREN is clear, only as many clock cycles are generated as there are data bits in a single character. The SREN bit is automatically cleared at the completion of one character. When CREN is set, clocks are continuously generated until CREN is cleared. If CREN is cleared in the middle of a character the CK clock stops immediately and the partial character is discarded. If SREN and CREN are both set, then SREN is cleared at the completion of the first character and CREN takes precedence.

To initiate reception, set either SREN or CREN. Data is sampled at the RX/DT pin on the trailing edge of the TX/CK clock pin and is shifted into the Receive Shift Register (RSR). When a complete character is received into the RSR, the RCIF bit is set and the character is automatically transferred to the two character receive FIFO. The Least Significant eight bits of the top character in the receive FIFO are available in RCREG. The RCIF bit remains set as long as there are un-read characters in the receive FIFO.

### 16.4.1.7 Slave Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a slave receives the clock on the TX/CK line. The TX/CK pin output driver must be disabled by setting the associated TRIS bit when the device is configured for synchronous slave transmit or receive operation. Serial data bits change on the leading edge to ensure they are valid at the trailing edge of each clock. One data bit is transferred for each clock cycle. Only as many clock cycles should be received as there are data bits.

### 16.4.1.8 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before RCREG is read to access the FIFO. When this happens the OERR bit of the RCSTA register is set. Previous data in the FIFO will not be overwritten. The two characters in the FIFO buffer can be read, however, no additional characters will be received until the error is cleared. The OERR bit can only be cleared by clearing the overrun condition. If the overrun error occurred when the SREN bit is set and CREN is clear then the error is cleared by reading RCREG. If the overrun occurred when the CREN bit is set then the error condition is cleared by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

### 16.4.1.9 Receiving 9-bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTA register is set the EUSART will shift 9-bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth, and Most Significant, data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the 8 Least Significant bits from the RCREG.

### 16.4.1.10 Synchronous Master Reception Set-up:

1. Initialize the SPBRGH, SPBRG register pair for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC. Disable RX/DT and TX/CK output drivers by setting the corresponding TRIS bits.
3. Ensure bits CREN and SREN are clear.
4. If using interrupts, set the GIE and PEIE bits of the INTCON register and set RCIE.
5. If 9-bit reception is desired, set bit RX9.
6. Start reception by setting the SREN bit or for continuous reception, set the CREN bit.
7. Interrupt flag bit RCIF will be set when reception of a character is complete. 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 an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

# PIC18F1XK50/PIC18LF1XK50

FIGURE 16-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)

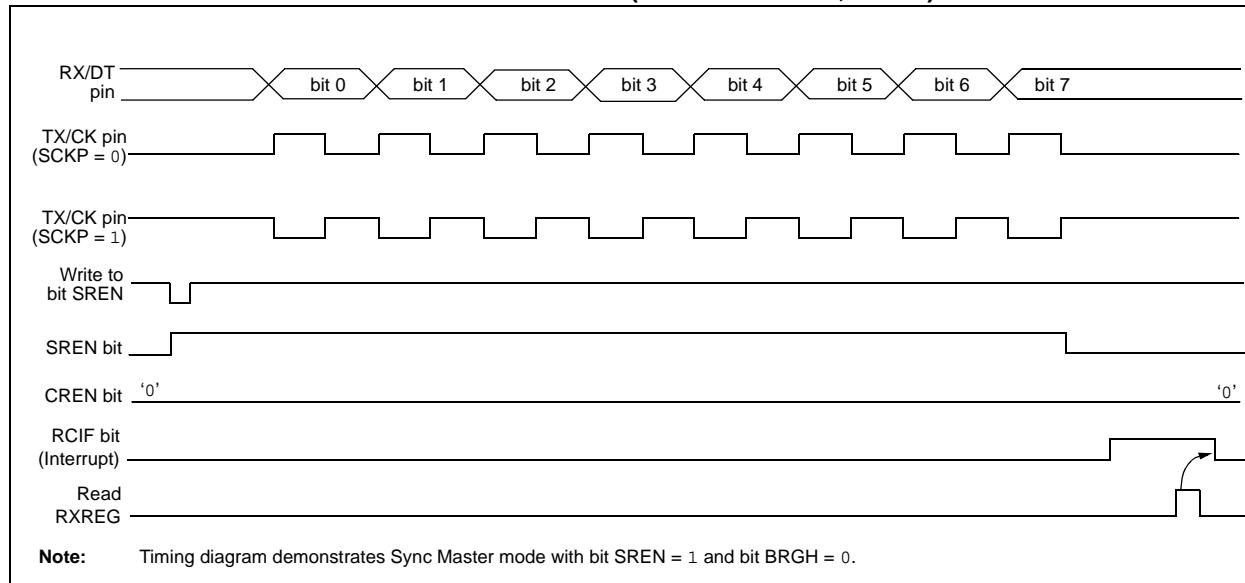


TABLE 16-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INT0IE	RABIE	TMROIF	INT0IF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	281
RCREG	EUSART Receive Register								281
TXSTA	CSRC	TX9	TXEN	SYNC	SEND <sub>B</sub>	BRGH	TRMT	TX9D	281
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	281
SPBRGH	EUSART Baud Rate Generator Register, High Byte								281
SPBRG	EUSART Baud Rate Generator Register, Low Byte								281

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

# PIC18F1XK50/PIC18LF1XK50

## 16.4.2 SYNCHRONOUS SLAVE MODE

The following bits are used to configure the EUSART for Synchronous slave operation:

- SYNC = 1
- CSRC = 0
- SREN = 0 (for transmit); SREN = 1 (for receive)
- CREN = 0 (for transmit); CREN = 1 (for receive)
- SPEN = 1

Setting the SYNC bit of the TXSTA register configures the device for synchronous operation. Clearing the CSRC bit of the TXSTA register configures the device as a slave. Clearing the SREN and CREN bits of the RCSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTA register enables the EUSART. If the RX/DT or TX/CK pins are shared with an analog peripheral the analog I/O functions must be disabled by clearing the corresponding ANSEL bits.

RX/DT and TX/CK pin output drivers must be disabled by setting the corresponding TRIS bits.

### 16.4.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see **Section 16.4.1.3 “Synchronous Master Transmission”**), 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:

1. The first character will immediately transfer to the TSR register and transmit.
2. The second word will remain in TXREG register.
3. The TXIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXREG register will transfer the second character to the TSR and the TXIF bit will now be set.
5. If the PEIE and TXIE bits are set, the interrupt will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will call the Interrupt Service Routine.

### 16.4.2.2 Synchronous Slave Transmission Set-up:

1. Set the SYNC and SPEN bits and clear the CSRC bit. Set the TRIS bits corresponding to the RX/DT and TX/CK I/O pins.
2. Clear the CREN and SREN bits.
3. If using interrupts, ensure that the GIE and PEIE bits of the INTCON register are set and set the TXIE bit.
4. If 9-bit transmission is desired, set the TX9 bit.
5. Enable transmission by setting the TXEN bit.
6. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
7. Start transmission by writing the Least Significant 8 bits to the TXREG register.

TABLE 16-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMROIE	INTOIE	RABIE	TMROIF	INTOIF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	281
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282
TXREG	EUSART Transmit Register								281
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	281
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	281
SPBRGH	EUSART Baud Rate Generator Register, High Byte								281
SPBRG	EUSART Baud Rate Generator Register, Low Byte								281

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

### 16.4.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (**Section 16.4.1.6 “Synchronous Master Reception”**), with the following exceptions:

- Sleep
- CREN bit is always set, therefore the receiver is never Idle
- SREN bit, which is a “don’t care” in Slave mode

A character may be received while in Sleep mode by setting the CREN bit prior to entering Sleep. Once the word is received, the RSR register will transfer the data to the RCREG register. If the RCIE enable bit is set, the interrupt generated will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will branch to the interrupt vector.

### 16.4.2.4 Synchronous Slave Reception Set-up:

1. Set the SYNC and SPEN bits and clear the CSRC bit. Set the TRIS bits corresponding to the RX/DT and TX/CK I/O pins.
2. If using interrupts, ensure that the GIE and PEIE bits of the INTCON register are set and set the RCIE bit.
3. If 9-bit reception is desired, set the RX9 bit.
4. Set the CREN bit to enable reception.
5. The RCIF bit will be set when reception is complete. An interrupt will be generated if the RCIE bit was set.
6. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCSTA register.
7. Retrieve the 8 Least Significant bits from the receive FIFO by reading the RCREG register.
8. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

**TABLE 16-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	281
RCREG	EUSART Receive Register								281
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	281
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	281
SPBRGH	EUSART Baud Rate Generator Register, High Byte								281
SPBRG	EUSART Baud Rate Generator Register, Low Byte								281

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

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 17.0 ANALOG-TO-DIGITAL CONVERTER (ADC) MODULE

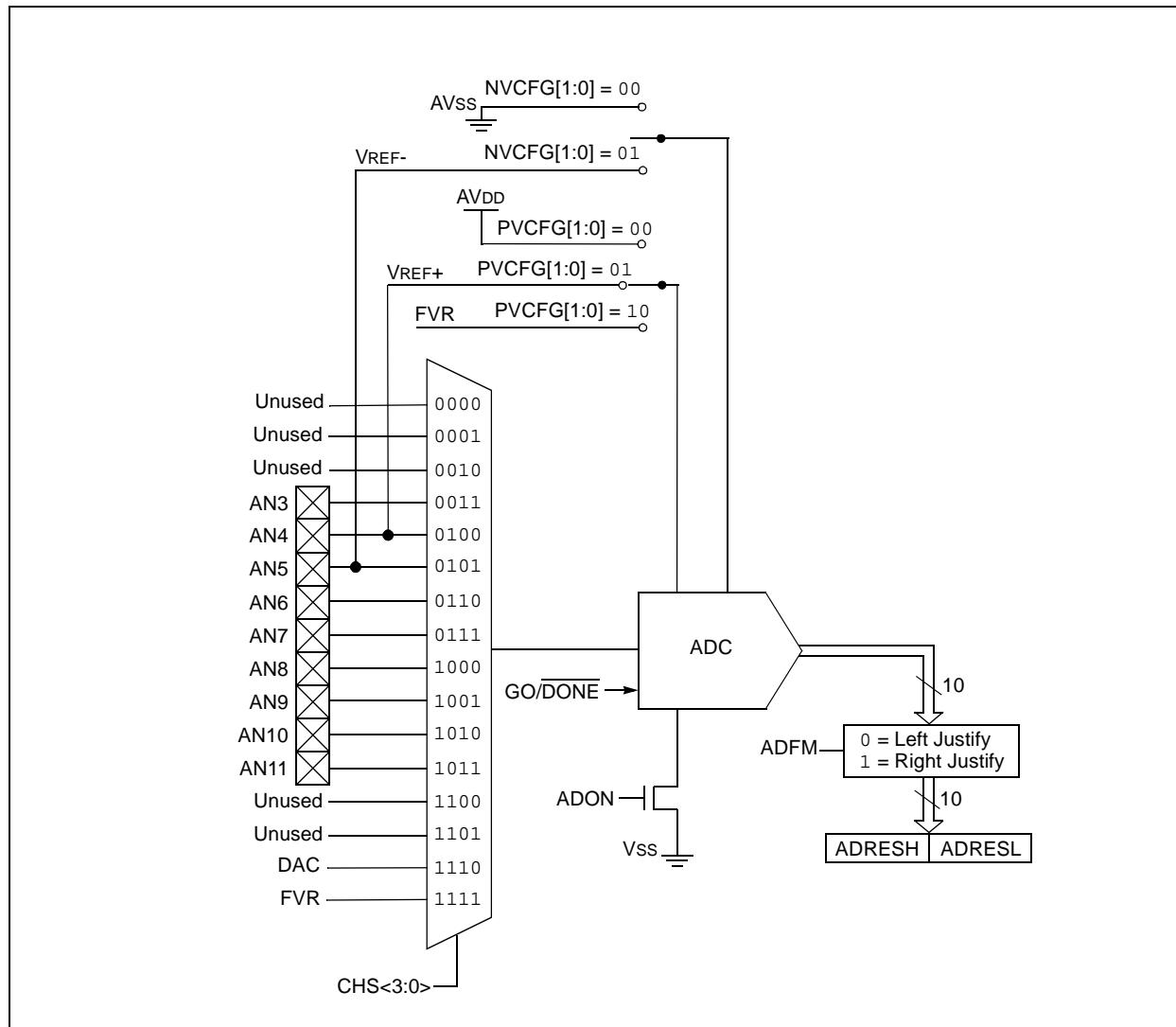
The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESL and ADRESH).

The ADC voltage reference is software selectable to either VDD, or a voltage applied to the external reference pins.

The ADC can generate an interrupt upon completion of a conversion. This interrupt can be used to wake-up the device from Sleep.

Figure 17-1 shows the block diagram of the ADC.

## FIGURE 17-1: ADC BLOCK DIAGRAM



## 17.1 ADC Configuration

When configuring and using the ADC the following functions must be considered:

- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Results formatting

### 17.1.1 PORT CONFIGURATION

The ANSEL, ANSELH, TRISA, TRISB and TRISE registers all configure the A/D port pins. Any port pin needed as an analog input should have its corresponding ANSx bit set to disable the digital input buffer and TRISx bit set to disable the digital output driver. If the TRISx bit is cleared, the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the ANSx bits and the TRIS bits.

- Note 1:** When reading the PORT register, all pins with their corresponding ANSx bit set read as cleared (a low level). However, analog conversion of pins configured as digital inputs (ANSx bit cleared and TRISx bit set) will be accurately converted.
- 2:** Analog levels on any pin with the corresponding ANSx bit cleared may cause the digital input buffer to consume current out of the device's specification limits.

### 17.1.2 CHANNEL SELECTION

The CHS bits of the ADCON0 register determine which channel is connected to the sample and hold circuit.

When changing channels, a delay is required before starting the next conversion. Refer to **Section 17.2 "ADC Operation"** for more information.

### 17.1.3 ADC VOLTAGE REFERENCE

The PVCFG and NVCFG bits of the ADCON1 register provide independent control of the positive and negative voltage references, respectively. The positive voltage reference can be either VDD, FVR or an external voltage source. The negative voltage reference can be either Vss or an external voltage source.

### 17.1.4 SELECTING AND CONFIGURING ACQUISITION TIME

The ADCON2 register allows the user to select an acquisition time that occurs each time the GO/DONE bit is set.

Acquisition time is set with the ACQT<2:0> bits of the ADCON2 register. Acquisition delays cover a range of 2 to 20 TAD. When the GO/DONE bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Since the acquisition time is programmed, there is no need to wait for an acquisition time between selecting a channel and setting the GO/DONE bit.

Manual acquisition is selected when ACQT<2:0> = 000. When the GO/DONE bit is set, sampling is stopped and a conversion begins. The user is responsible for ensuring the required acquisition time has passed between selecting the desired input channel and setting the GO/DONE bit. This option is also the default Reset state of the ACQT<2:0> bits and is compatible with devices that do not offer programmable acquisition times.

In either case, when the conversion is completed, the GO/DONE bit is cleared, the ADIF flag is set and the A/D begins sampling the currently selected channel again. When an acquisition time is programmed, there is no indication of when the acquisition time ends and the conversion begins.

### 17.1.5 CONVERSION CLOCK

The source of the conversion clock is software selectable via the ADCS bits of the ADCON2 register. There are seven possible clock options:

- Fosc/2
- Fosc/4
- Fosc/8
- Fosc/16
- Fosc/32
- Fosc/64
- FRC (dedicated internal oscillator)

The time to complete one bit conversion is defined as TAD. One full 10-bit conversion requires 11 TAD periods as shown in Figure 17-3.

For correct conversion, the appropriate TAD specification must be met. See A/D conversion requirements in Table 27-9 for more information. Table 17-1 gives examples of appropriate ADC clock selections.

**Note:** Unless using the FRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.

## 17.1.6 INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital Conversion. The ADC interrupt flag is the ADIF bit in the PIR1 register. The ADC interrupt enable is the ADIE bit in the PIE1 register. The ADIF bit must be cleared by software.

**Note:** The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled.

This interrupt can be generated while the device is operating or while in Sleep. If the device is in Sleep, the interrupt will wake-up the device. Upon waking from Sleep, the next instruction following the SLEEP instruction is always executed. If the user is attempting to wake-up from Sleep and resume in-line code execution, the global interrupt must be disabled. If the global interrupt is enabled, execution will switch to the Interrupt Service Routine. Please see **Section 17.1.6 “Interrupts”** for more information.

**TABLE 17-1: ADC CLOCK PERIOD (TAD) Vs. DEVICE OPERATING FREQUENCIES**

ADC Clock Period (TAD)		Device Frequency (Fosc)			
ADC Clock Source	ADCS<2:0>	48 MHz	16 MHz	4 MHz	1 MHz
Fosc/2	000	41.67 ns <sup>(2)</sup>	125 ns <sup>(2)</sup>	500 ns <sup>(2)</sup>	2.0 $\mu$ s
Fosc/4	100	83.33 ns <sup>(2)</sup>	250 ns <sup>(2)</sup>	1.0 $\mu$ s	4.0 $\mu$ s
Fosc/8	001	167 ns <sup>(2)</sup>	500 ns <sup>(2)</sup>	2.0 $\mu$ s	8.0 $\mu$ s <sup>(3)</sup>
Fosc/16	101	333 ns <sup>(2)</sup>	1.0 $\mu$ s	4.0 $\mu$ s	16.0 $\mu$ s <sup>(3)</sup>
Fosc/32	010	667 ns <sup>(2)</sup>	2.0 $\mu$ s	8.0 $\mu$ s <sup>(3)</sup>	32.0 $\mu$ s <sup>(3)</sup>
Fosc/64	110	1.33 $\mu$ s	4.0 $\mu$ s	16.0 $\mu$ s <sup>(3)</sup>	64.0 $\mu$ s <sup>(3)</sup>
FRC	x11	1-4 $\mu$ s <sup>(1,4)</sup>			

**Legend:** Shaded cells are outside of recommended range.

**Note 1:** The FRC source has a typical TAD time of 1.7  $\mu$ s.

**2:** These values violate the minimum required TAD time.

**3:** For faster conversion times, the selection of another clock source is recommended.

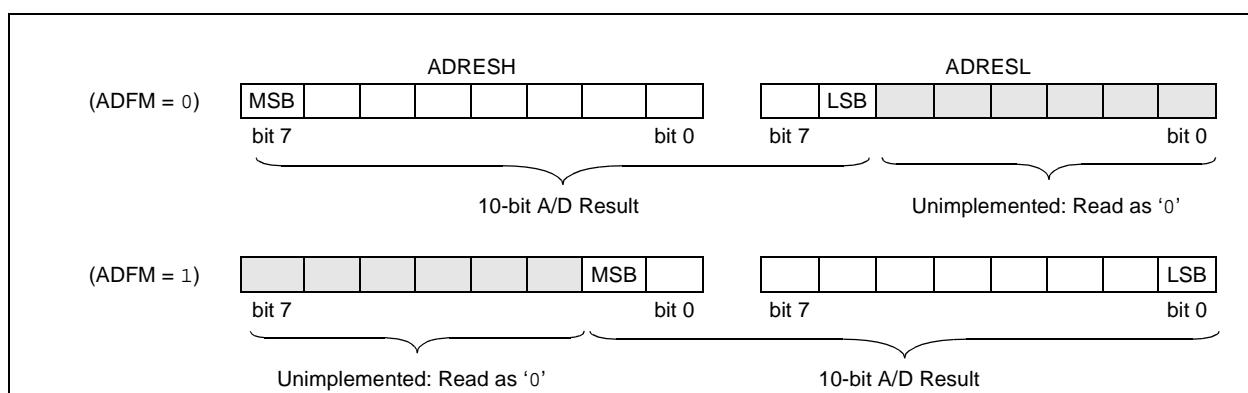
**4:** When the device frequency is greater than 1 MHz, the FRC clock source is only recommended if the conversion will be performed during Sleep.

## 17.1.7 RESULT FORMATTING

The 10-bit A/D conversion result can be supplied in two formats, left justified or right justified. The ADFM bit of the ADCON2 register controls the output format.

Figure 17-2 shows the two output formats.

**FIGURE 17-2: 10-BIT A/D CONVERSION RESULT FORMAT**



# PIC18F1XK50/PIC18LF1XK50

## 17.2 ADC Operation

### 17.2.1 STARTING A CONVERSION

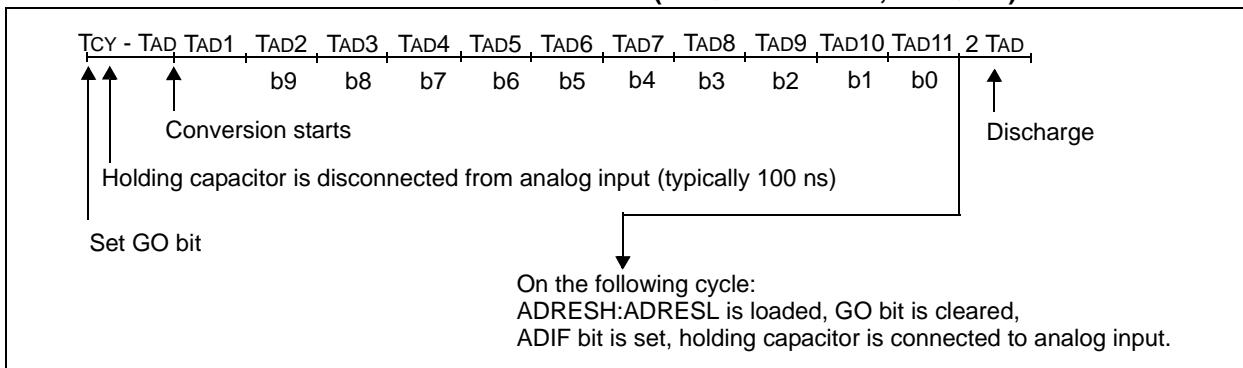
To enable the ADC module, the ADON bit of the ADCON0 register must be set to a '1'. Setting the GO/DONE bit of the ADCON0 register to a '1' will, depending on the ACQT bits of the ADCON2 register, either immediately start the Analog-to-Digital conversion or start an acquisition delay followed by the Analog-to-Digital conversion.

Figure 17-3 shows the operation of the A/D converter after the GO bit has been set and the ACQT<2:0> bits are cleared. A conversion is started after the following instruction to allow entry into SLEEP mode before the conversion begins.

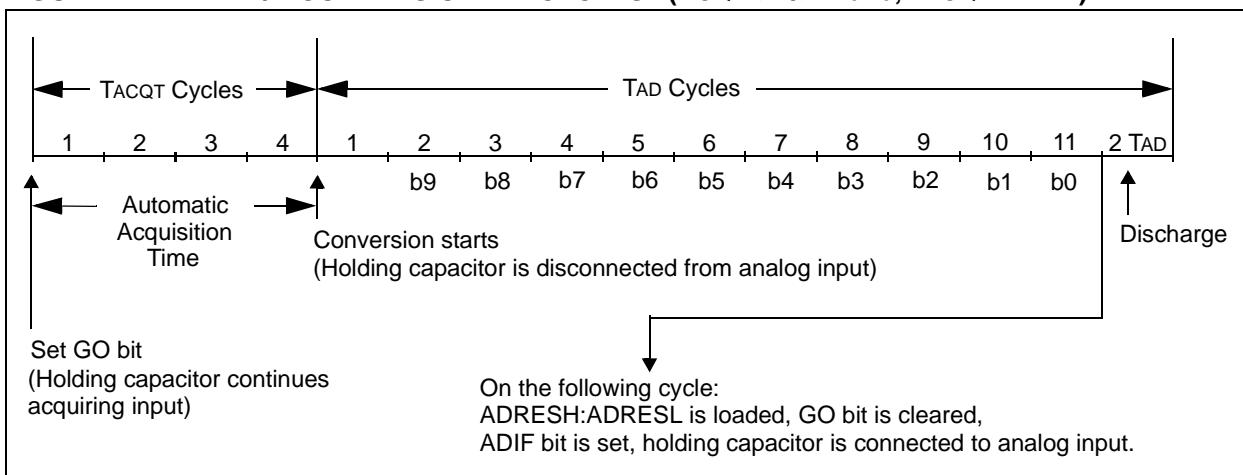
Figure 17-4 shows the operation of the A/D converter after the GO bit has been set and the ACQT<2:0> bits are set to '010' which selects a 4 TAD acquisition time before the conversion starts.

**Note:** The GO/DONE bit should not be set in the same instruction that turns on the ADC. Refer to **Section 17.2.9 "A/D Conversion Procedure"**.

**FIGURE 17-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**



**FIGURE 17-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**



## 17.2.2 COMPLETION OF A CONVERSION

When the conversion is complete, the ADC module will:

- Clear the GO/DONE bit
- Set the ADIF flag bit
- Update the ADRESH:ADRESL registers with new conversion result

## 17.2.3 DISCHARGE

The discharge phase is used to initialize the value of the capacitor array. The array is discharged after every sample. This feature helps to optimize the unity-gain amplifier, as the circuit always needs to charge the capacitor array, rather than charge/discharge based on previous measure values.

## 17.2.4 TERMINATING A CONVERSION

If a conversion must be terminated before completion, the GO/DONE bit can be cleared by software. The ADRESH:ADRESL registers will be updated with the partially complete Analog-to-Digital conversion sample. Unconverted bits will match the last bit converted.

**Note:** A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated.

## 17.2.5 DELAY BETWEEN CONVERSIONS

After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, the currently selected channel is reconnected to the charge holding capacitor commencing the next acquisition.

## 17.2.6 ADC OPERATION IN POWER-MANAGED MODES

The selection of the automatic acquisition time and A/D conversion clock is determined in part by the clock source and frequency while in a power-managed mode.

If the A/D is expected to operate while the device is in a power-managed mode, the ACQT<2:0> and ADCS<2:0> bits in ADCON2 should be updated in accordance with the clock source to be used in that mode. After entering the mode, an A/D acquisition or conversion may be started. Once started, the device should continue to be clocked by the same clock source until the conversion has been completed.

If desired, the device may be placed into the corresponding Idle mode during the conversion. If the device clock frequency is less than 1 MHz, the A/D FRC clock source should be selected.

## 17.2.7 ADC OPERATION DURING SLEEP

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the FRC option. When the FRC clock source is selected, the ADC waits one additional instruction before starting the conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled, the device will wake-up from Sleep when the conversion completes. If the ADC interrupt is disabled, the ADC module is turned off after the conversion completes, although the ADON bit remains set.

When the ADC clock source is something other than FRC, a SLEEP instruction causes the present conversion to be aborted and the ADC module is turned off, although the ADON bit remains set.

## 17.2.8 SPECIAL EVENT TRIGGER

The CCP1 Special Event Trigger allows periodic ADC measurements without software intervention. When this trigger occurs, the GO/DONE bit is set by hardware and the Timer1 or Timer3 counter resets to zero.

Using the Special Event Trigger does not assure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met.

See **Section 14.3.4 “Special Event Trigger”** for more information.

# PIC18F1XK50/PIC18LF1XK50

## 17.2.9 A/D CONVERSION PROCEDURE

This is an example procedure for using the ADC to perform an Analog-to-Digital conversion:

1. Configure Port:
  - Disable pin output driver (See TRIS register)
  - Configure pin as analog
2. Configure the ADC module:
  - Select ADC conversion clock
  - Configure voltage reference
  - Select ADC input channel
  - Select result format
  - Select acquisition delay
  - Turn on ADC module
3. Configure ADC interrupt (optional):
  - Clear ADC interrupt flag
  - Enable ADC interrupt
  - Enable peripheral interrupt
  - Enable global interrupt<sup>(1)</sup>
4. Wait the required acquisition time<sup>(2)</sup>.
5. Start conversion by setting the GO/DONE bit.
6. Wait for ADC conversion to complete by one of the following:
  - Polling the GO/DONE bit
  - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result
8. Clear the ADC interrupt flag (required if interrupt is enabled).

**Note 1:** The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

**2:** Software delay required if ACQT bits are set to zero delay. See **Section 17.3 “A/D Acquisition Requirements”**.

## EXAMPLE 17-1: A/D CONVERSION

```
;This code block configures the ADC
;for polling, Vdd and Vss as reference, Frc
;clock and AN4 input.
;
;Conversion start & polling for completion
;are included.
;
MOVLW B'10101111' ;right justify, Frc,
MOVWF ADCON2 ; & 12 TAD ACQ time
MOVLW B'00000000' ;ADC ref = Vdd,Vss
MOVWF ADCON1 ;
BSF TRISC,0 ;Set RC0 to input
BSF ANSEL,4 ;Set RC0 to analog
MOVLW B'00010001' ;AN4, ADC on
MOVWF ADCON0 ;
BSF ADCON0,GO ;Start conversion
ADCPoll:
BTFSR ADCON0,GO ;Is conversion done?
BRA ADCPoll ;No, test again
; Result is complete - store 2 MSbits in
; RESULTHI and 8 LSbits in RESULTLO
MOVFF ADRESH,RESULTHI
MOVFF ADRESL,RESULTLO
```

## 17.2.10 ADC REGISTER DEFINITIONS

The following registers are used to control the operation of the ADC.

**Note:** Analog pin control is performed by the ANSEL and ANSELH registers. For ANSEL and ANSELH registers, see Register 9-15 and Register 9-16, respectively.

### REGISTER 17-1: ADCON0: A/D CONTROL REGISTER 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
bit 7							

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

bit 5-2      **CHS<3:0>: Analog Channel Select bits**

0000 = Reserved

0001 = Reserved

0010 = Reserved

0011 = AN3

0100 = AN4

0101 = AN5

0110 = AN6

0111 = AN7

1000 = AN8

1001 = AN9

1010 = AN10

1011 = AN11

1100 = Reserved

1101 = Reserved

1110 = DAC

1111 = FVR

bit 1      **GO/DONE:** A/D Conversion Status bit

1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.

This bit is automatically cleared by hardware when the A/D conversion has completed.

0 = A/D conversion completed/not in progress

bit 0      **ADON:** ADC Enable bit

1 = ADC is enabled

0 = ADC is disabled and consumes no operating current

**Note 1:** Selecting reserved channels will yield unpredictable results as unimplemented input channels are left floating.

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 17-2: ADCON1: A/D CONTROL REGISTER 1

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	PVCFG1	PVCFG0	NVCFG1	NVCFG0
bit 7							

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-4      **Unimplemented:** Read as '0'

bit 3-2      **PVCFG<1:0>:** Positive Voltage Reference select bit

00 = Positive voltage reference supplied internally by VDD.

01 = Positive voltage reference supplied externally through VREF+ pin.

10 = Positive voltage reference supplied internally through FVR.

11 = Reserved.

bit 1-0      **NVCFG<1:0>:** Negative Voltage Reference select bit

00 = Positive voltage reference supplied internally by Vss.

01 = Positive voltage reference supplied externally through VREF- pin.

10 = Reserved.

11 = Reserved.

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 17-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>ADFM:</b> A/D Conversion Result Format Select bit 1 = Right justified 0 = Left justified
bit 6	<b>Unimplemented:</b> Read as '0'
bit 5-3	<b>ACQT&lt;2:0&gt;:</b> A/D Acquisition time select bits. Acquisition time is the duration that the A/D charge holding capacitor remains connected to A/D channel from the instant the GO/DONE bit is set until conversions begins. 000 = 0 <sup>(1)</sup> 001 = 2 TAD 010 = 4 TAD 011 = 6 TAD 100 = 8 TAD 101 = 12 TAD 110 = 16 TAD 111 = 20 TAD
bit 2-0	<b>ADCS&lt;2:0&gt;:</b> A/D Conversion Clock Select bits 000 = Fosc/2 001 = Fosc/8 010 = Fosc/32 011 = FRC <sup>(1)</sup> (clock derived from a dedicated internal oscillator = 600 kHz nominal) 100 = Fosc/4 101 = Fosc/16 110 = Fosc/64 111 = FRC <sup>(1)</sup> (clock derived from a dedicated internal oscillator = 600 kHz nominal)

**Note 1:** When the A/D clock source is selected as FRC then the start of conversion is delayed by one instruction cycle after the GO/DONE bit is set to allow the SLEEP instruction to be executed.

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 17-4: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 0

| R/W-x  |
|--------|--------|--------|--------|--------|--------|--------|--------|
| ADRES9 | ADRES8 | ADRES7 | ADRES6 | ADRES5 | ADRES4 | ADRES3 | ADRES2 |
| bit 7  |        |        |        |        |        |        | bit 0  |

**Legend:**

R = Readable bit

-n = Value at POR

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7-0      **ADRES<9:2>**: ADC Result Register bits  
Upper 8 bits of 10-bit conversion result

## REGISTER 17-5: ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 0

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES1	ADRES0	—	—	—	—	—	—
bit 7							bit 0

**Legend:**

R = Readable bit

-n = Value at POR

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7-6      **ADRES<1:0>**: ADC Result Register bits  
Lower 2 bits of 10-bit conversion result

bit 5-0      **Reserved**: Do not use.

## REGISTER 17-6: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 1

| R/W-x  |
|-------|-------|-------|-------|-------|-------|-------|--------|
| —     | —     | —     | —     | —     | —     | —     | ADRES9 |
| bit 7 |       |       |       |       |       |       | bit 0  |

**Legend:**

R = Readable bit

-n = Value at POR

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7-2      **Reserved**: Do not use.

bit 1-0      **ADRES<9:8>**: ADC Result Register bits  
Upper 2 bits of 10-bit conversion result

## REGISTER 17-7: ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 1

| R/W-x  |
|--------|--------|--------|--------|--------|--------|--------|--------|
| ADRES7 | ADRES6 | ADRES5 | ADRES4 | ADRES3 | ADRES2 | ADRES1 | ADRES0 |
| bit 7  |        |        |        |        |        |        | bit 0  |

**Legend:**

R = Readable bit

-n = Value at POR

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7-0      **ADRES<7:0>**: ADC Result Register bits  
Lower 8 bits of 10-bit conversion result

## 17.3 A/D Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 17-5. The source impedance ( $R_s$ ) and the internal sampling switch ( $R_{ss}$ ) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch ( $R_{ss}$ ) impedance varies over the device voltage ( $V_{DD}$ ), see Figure 17-5. **The maximum recommended impedance for analog sources is 10 k $\Omega$ .** As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed),

an A/D acquisition must be done before the conversion can be started. 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 ADC). The 1/2 LSB error is the maximum error allowed for the ADC to meet its specified resolution.

### EQUATION 17-1: ACQUISITION TIME EXAMPLE

*Assumptions: Temperature = 50°C and external impedance of 10k $\Omega$  3.0V VDD*

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \\ &= 5\mu\text{s} + T_C + [(Temperature - 25^\circ\text{C})(0.05\mu\text{s}/^\circ\text{C})] \end{aligned}$$

*The value for  $T_C$  can be approximated with the following equations:*

$$\begin{aligned} V_{APPLIED} \left( 1 - \frac{1}{2047} \right) &= V_{CHOLD} && ;[1] V_{CHOLD} \text{ charged to within 1/2 lsb} \\ V_{APPLIED} \left( 1 - e^{-\frac{T_C}{RC}} \right) &= V_{CHOLD} && ;[2] V_{CHOLD} \text{ charge response to } V_{APPLIED} \\ V_{APPLIED} \left( 1 - e^{-\frac{T_C}{RC}} \right) &= V_{APPLIED} \left( 1 - \frac{1}{2047} \right) && ;\text{combining [1] and [2]} \end{aligned}$$

*Solving for  $T_C$ :*

$$\begin{aligned} T_C &= -CHOLD(R_{IC} + R_{SS} + R_s) \ln(1/2047) \\ &= -13.5pF(1k\Omega + 700\Omega + 10k\Omega) \ln(0.0004885) \\ &= 1.20\mu\text{s} \end{aligned}$$

*Therefore:*

$$\begin{aligned} T_{ACQ} &= 5MS + 1.20MS + [(50^\circ\text{C} - 25^\circ\text{C})(0.05MS/^\circ\text{C})] \\ &= 7.45MS \end{aligned}$$

**Note 1:** The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

- 2:** The charge holding capacitor (CHOLD) is discharged after each conversion.
- 3:** The maximum recommended impedance for analog sources is 10 k $\Omega$ . This is required to meet the pin leakage specification.

# PIC18F1XK50/PIC18LF1XK50

FIGURE 17-5: ANALOG INPUT MODEL

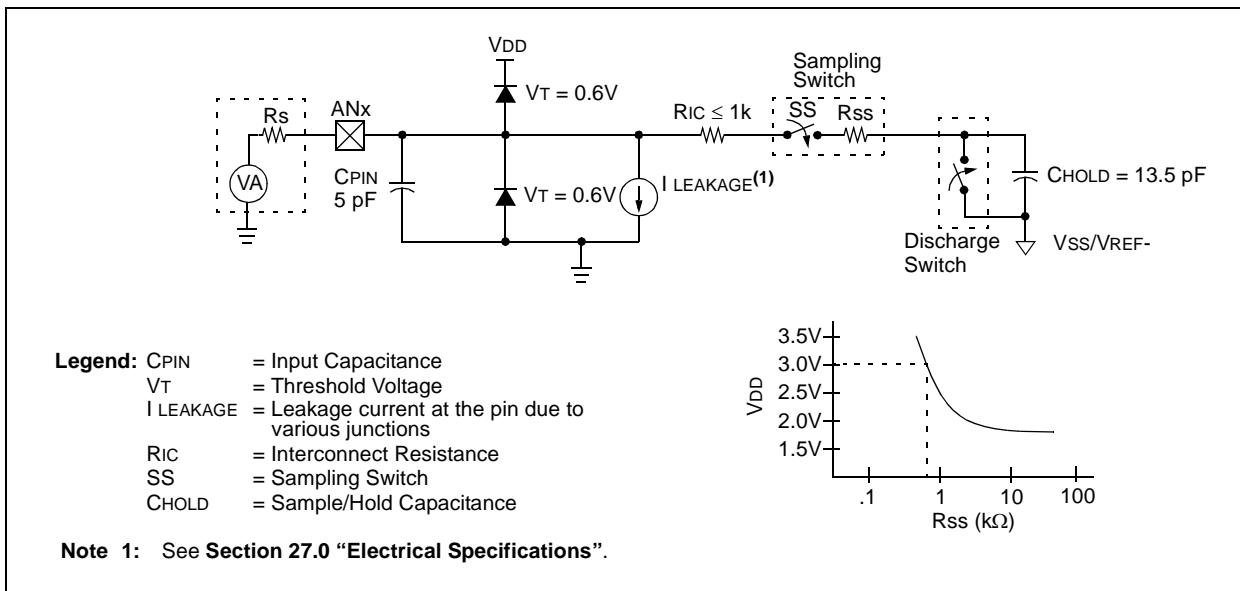
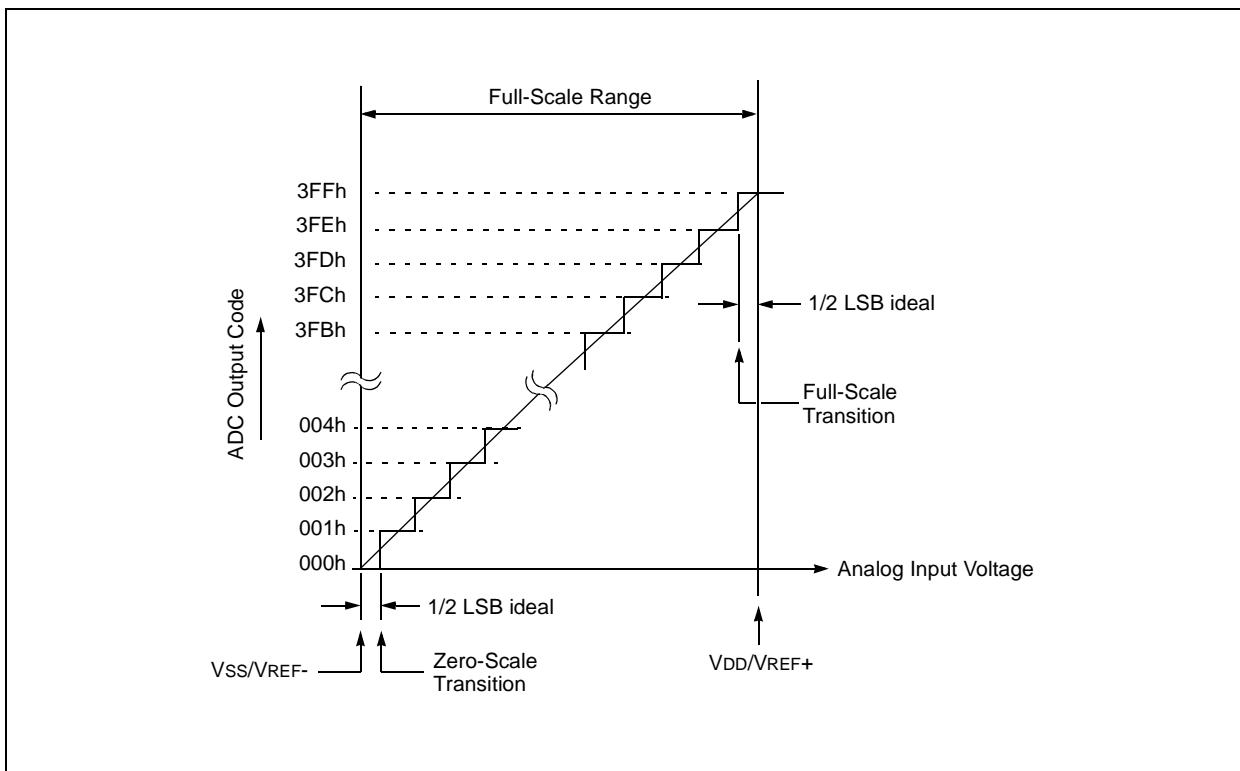


FIGURE 17-6: ADC TRANSFER FUNCTION



# PIC18F1XK50/PIC18LF1XK50

TABLE 17-2: REGISTERS ASSOCIATED WITH A/D OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	282
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	282
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	282
ADRESH	A/D Result Register, High Byte								281
ADRESL	A/D Result Register, Low Byte								281
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	281
ADCON1	—	—	—	—	PVCFG1	PVCFG0	NVCFG1	NVCFG0	281
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	281
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	282
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	282
TRISA	—	—	TRISA5	TRISA4	—	—	—	—	282
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	282
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 18.0 COMPARATOR MODULE

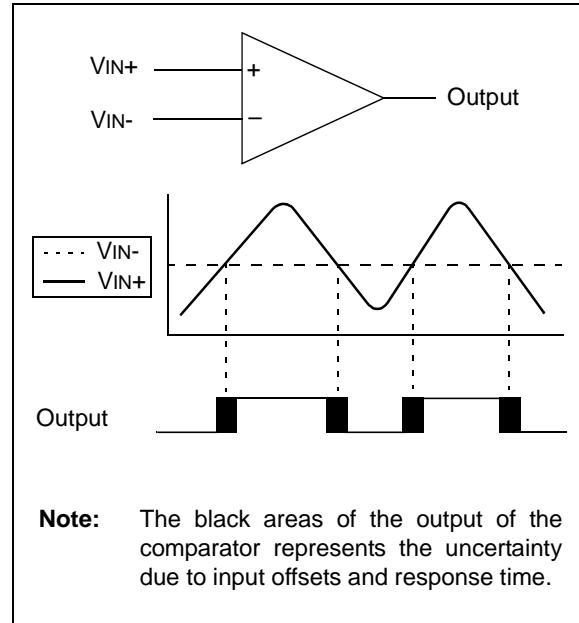
Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. The comparators are very useful mixed signal building blocks because they provide analog functionality independent of the program execution. The Analog Comparator module includes the following features:

- Independent comparator control
- Programmable input selection
- Comparator output is available internally/externally
- Programmable output polarity
- Interrupt-on-change
- Wake-up from Sleep
- Programmable Speed/Power optimization
- PWM shutdown
- Programmable and fixed voltage reference

### 18.1 Comparator Overview

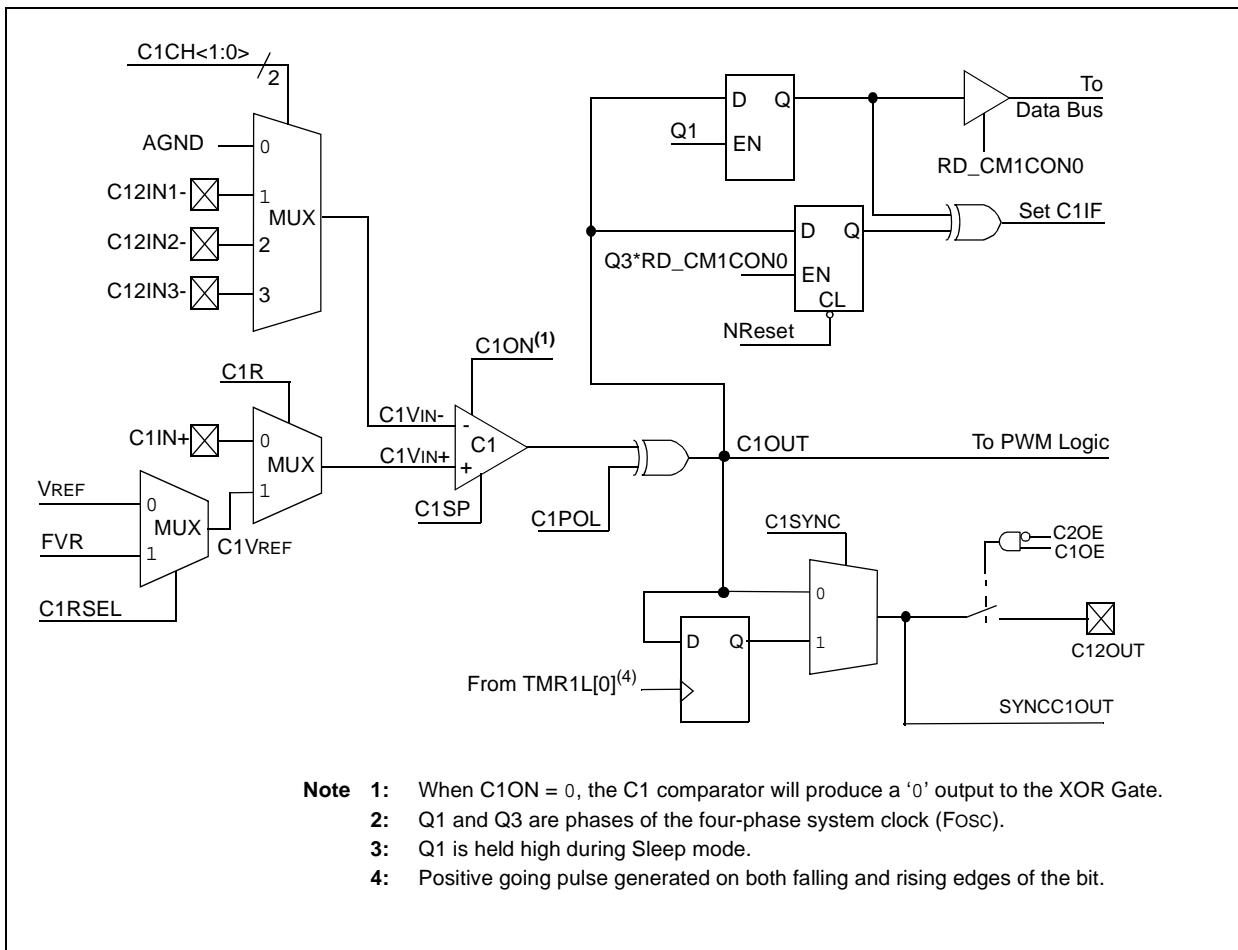
A single comparator is shown in Figure 18-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at  $V_{IN+}$  is less than the analog voltage at  $V_{IN-}$ , the output of the comparator is a digital low level. When the analog voltage at  $V_{IN+}$  is greater than the analog voltage at  $V_{IN-}$ , the output of the comparator is a digital high level.

FIGURE 18-1: SINGLE COMPARATOR



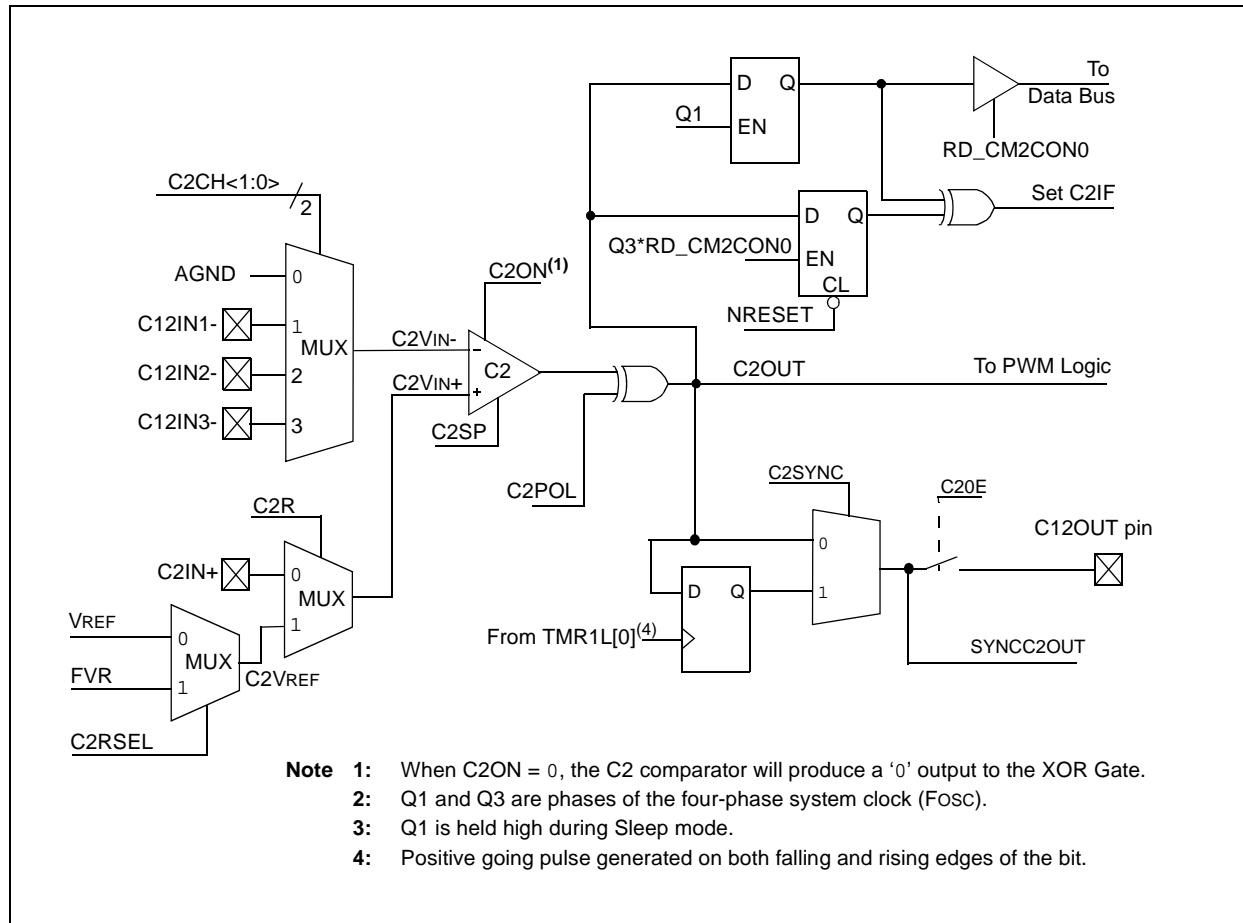
# PIC18F1XK50/PIC18LF1XK50

FIGURE 18-2: COMPARATOR C1 SIMPLIFIED BLOCK DIAGRAM



# PIC18F1XK50/PIC18LF1XK50

FIGURE 18-3: COMPARATOR C2 SIMPLIFIED BLOCK DIAGRAM



## 18.2 Comparator Control

Each comparator has a separate control and Configuration register: CM1CON0 for Comparator C1 and CM2CON0 for Comparator C2. In addition, Comparator C2 has a second control register, CM2CON1, for controlling the interaction with Timer1 and simultaneous reading of both comparator outputs.

The CM1CON0 and CM2CON0 registers (see Registers 18-1 and 18-2, respectively) contain the control and status bits for the following:

- Enable
- Input selection
- Reference selection
- Output selection
- Output polarity
- Speed selection

### 18.2.1 COMPARATOR ENABLE

Setting the CxON bit of the CMxCON0 register enables the comparator for operation. Clearing the CxON bit disables the comparator resulting in minimum current consumption.

### 18.2.2 COMPARATOR INPUT SELECTION

The CxCH<1:0> bits of the CMxCON0 register direct one of four analog input pins to the comparator inverting input.

**Note:** To use CxIN+ and C12INx- pins as analog inputs, the appropriate bits must be set in the ANSEL register and the corresponding TRIS bits must also be set to disable the output drivers.

### 18.2.3 COMPARATOR REFERENCE SELECTION

Setting the CxR bit of the CMxCON0 register directs an internal voltage reference or an analog input pin to the non-inverting input of the comparator. See **Section 21.0 “VOLTAGE REFERENCES”** for more information on the Internal Voltage Reference module.

### 18.2.4 COMPARATOR OUTPUT SELECTION

The output of the comparator can be monitored by reading either the CxOUT bit of the CMxCON0 register or the MCxOUT bit of the CM2CON1 register. In order to make the output available for an external connection, the following conditions must be true:

- CxOE bit of the CMxCON0 register must be set
- Corresponding TRIS bit must be cleared
- CxON bit of the CMxCON0 register must be set

Both comparators share the same output pin (C12OUT). Priority is determined by the states of the C1OE and C2OE bits.

**TABLE 18-1: COMPARATOR OUTPUT PRIORITY**

C10E	C2OE	C12OUT
0	0	I/O
0	1	C2OUT
1	0	C1OUT
1	1	C2OUT

**Note 1:** The CxOE bit overrides the PORT data latch. Setting the CxON has no impact on the port override.

**2:** The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.

### 18.2.5 COMPARATOR OUTPUT POLARITY

Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the CxPOL bit of the CMxCON0 register. Clearing the CxPOL bit results in a non-inverted output.

Table 18-2 shows the output state versus input conditions, including polarity control.

**TABLE 18-2: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS**

Input Condition	CxPOL	CxOUT
CxVIN- > CxVIN+	0	0
CxVIN- < CxVIN+	0	1
CxVIN- > CxVIN+	1	1
CxVIN- < CxVIN+	1	0

### 18.2.6 COMPARATOR SPEED SELECTION

The trade-off between speed or power can be optimized during program execution with the CxSP control bit. The default state for this bit is ‘1’ which selects the normal speed mode. Device power consumption can be optimized at the cost of slower comparator propagation delay by clearing the CxSP bit to ‘0’.

## 18.3 Comparator Response Time

The comparator output is indeterminate for a period of time after the change of an input source or the selection of a new reference voltage. This period is referred to as the response time. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response time to a comparator input change. See the Comparator and Voltage Reference Specifications in **Section 27.0 “Electrical Specifications”** for more details.

## 18.4 Comparator Interrupt Operation

The comparator interrupt flag can be set whenever there is a change in the output value of the comparator. Changes are recognized by means of a mismatch circuit which consists of two latches and an exclusive-or gate (see Figure 18-2 and Figure 18-3). One latch is updated with the comparator output level when the CMxCON0 register is read. This latch retains the value until the next read of the CMxCON0 register or the occurrence of a Reset. The other latch of the mismatch circuit is updated on every Q1 system clock. A mismatch condition will occur when a comparator output change is clocked through the second latch on the Q1 clock cycle. At this point the two mismatch latches have opposite output levels which is detected by the exclusive-or gate and fed to the interrupt circuitry. The mismatch condition persists until either the CMxCON0 register is read or the comparator output returns to the previous state.

- Note 1:** A write operation to the CMxCON0 register will also clear the mismatch condition because all writes include a read operation at the beginning of the write cycle.
- 2:** Comparator interrupts will operate correctly regardless of the state of CxOE.

The comparator interrupt is set by the mismatch edge and not the mismatch level. This means that the interrupt flag can be reset without the additional step of reading or writing the CMxCON0 register to clear the mismatch registers. When the mismatch registers are cleared, an interrupt will occur upon the comparator's return to the previous state, otherwise no interrupt will be generated.

Software will need to maintain information about the status of the comparator output, as read from the CMxCON0 register, or CM2CON1 register, to determine the actual change that has occurred. See Figures 18-4 and 18-5.

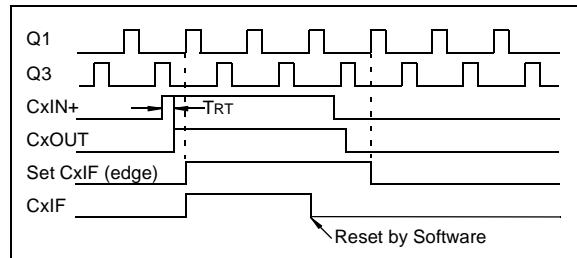
The CxIF bit of the PIR2 register is the comparator interrupt flag. This bit must be reset by software by clearing it to '0'. Since it is also possible to write a '1' to this register, an interrupt can be generated.

In mid-range Compatibility mode the CxIE bit of the PIE2 register and the PEIE and GIE bits of the INTCON register must all be set to enable comparator interrupts. If any of these bits are cleared, the interrupt is not enabled, although the CxIF bit of the PIR2 register will still be set if an interrupt condition occurs.

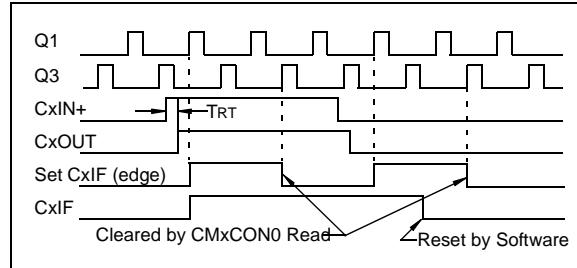
### 18.4.1 PRESETTING THE MISMATCH LATCHES

The comparator mismatch latches can be preset to the desired state before the comparators are enabled. When the comparator is off the CxPOL bit controls the CxOUT level. Set the CxPOL bit to the desired CxOUT non-interrupt level while the CxON bit is cleared. Then, configure the desired CxPOL level in the same instruction that the CxON bit is set. Since all register writes are performed as a Read-Modify-Write, the mismatch latches will be cleared during the instruction Read phase and the actual configuration of the CxON and CxPOL bits will occur in the final Write phase.

**FIGURE 18-4: COMPARATOR INTERRUPT TIMING W/O CMxCON0 READ**



**FIGURE 18-5: COMPARATOR INTERRUPT TIMING WITH CMxCON0 READ**



- Note 1:** If a change in the CMxCON0 register (CxOUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CxIF interrupt flag of the PIR2 register may not get set.

- 2:** When either comparator is first enabled, bias circuitry in the Comparator module may cause an invalid output from the comparator until the bias circuitry is stable. Allow about 1  $\mu$ s for bias settling then clear the mismatch condition and interrupt flags before enabling comparator interrupts.

## 18.5 Operation During Sleep

The comparator, if enabled before entering Sleep mode, remains active during Sleep. The additional current consumed by the comparator is shown separately in the **Section 27.0 “Electrical Specifications”**. If the comparator is not used to wake the device, power consumption can be minimized while in Sleep mode by turning off the comparator. Each comparator is turned off by clearing the CxON bit of the CMxCON0 register.

A change to the comparator output can wake-up the device from Sleep. To enable the comparator to wake the device from Sleep, the CxIE bit of the PIE2 register and the PEIE bit of the INTCON register must be set. The instruction following the `SLEEP` instruction always executes following a wake from Sleep. If the GIE bit of the INTCON register is also set, the device will then execute the Interrupt Service Routine.

## 18.6 Effects of a Reset

A device Reset forces the CMxCON0 and CM2CON1 registers to their Reset states. This forces both comparators and the voltage references to their Off states.

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 18-1: CM1CON0: COMPARATOR 1 CONTROL REGISTER 0

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH1	C1CHO
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>C1ON:</b> Comparator C1 Enable bit 1 = Comparator C1 is enabled 0 = Comparator C1 is disabled
bit 6	<b>C1OUT:</b> Comparator C1 Output bit <u>If C1POL = 1 (inverted polarity):</u> C1OUT = 0 when C1VIN+ > C1VIN- C1OUT = 1 when C1VIN+ < C1VIN- <u>If C1POL = 0 (non-inverted polarity):</u> C1OUT = 1 when C1VIN+ > C1VIN- C1OUT = 0 when C1VIN+ < C1VIN-
bit 5	<b>C1OE:</b> Comparator C1 Output Enable bit <u>If C2OE = 0 (C2 output disable)</u> 0 = C1OUT is internal only 1 = C1OUT is present on the C12OUT pin <sup>(1)</sup> <u>If C2OE = 1 (C2 output enable)</u> 0 = C1OUT is internal only 1 = C1OUT is present on the C12OUT pin <sup>(1)</sup>
bit 4	<b>C1POL:</b> Comparator C1 Output Polarity Select bit 1 = C1OUT logic is inverted 0 = C1OUT logic is not inverted
bit 3	<b>C1SP:</b> Comparator C1 Speed/Power Select bit 1 = C1 operates in normal power, higher speed mode 0 = C1 operates in low-power, low-speed mode
bit 2	<b>C1R:</b> Comparator C1 Reference Select bit (non-inverting input) 1 = C1VIN+ connects to C1VREF output 0 = C1VIN+ connects to C12IN+ pin
bit 1-0	<b>C1CH&lt;1:0&gt;:</b> Comparator C1 Channel Select bit 00 = C1VIN- connects to AGND 01 = C12IN1- pin of C1 connects to C1VIN- 10 = C12IN2- pin of C1 connects to C1VIN- 11 = C12IN3- pin of C1 connects to C1VIN-

**Note 1:** Comparator output requires the following three conditions: C1OE = 1, C1ON = 1 and corresponding port TRIS bit = 0.

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 18-2: CM2CON0: COMPARATOR 2 CONTROL REGISTER 0

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH1	C2CH0
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>C2ON:</b> Comparator C2 Enable bit 1 = Comparator C2 is enabled 0 = Comparator C2 is disabled
bit 6	<b>C2OUT:</b> Comparator C2 Output bit <u>If C2POL = 1 (inverted polarity):</u> C2OUT = 0 when C2VIN+ > C2VIN- C2OUT = 1 when C2VIN+ < C2VIN- <u>If C2POL = 0 (non-inverted polarity):</u> C2OUT = 1 when C2VIN+ > C2VIN- C2OUT = 0 when C2VIN+ < C2VIN-
bit 5	<b>C2OE:</b> Comparator C2 Output Enable bit 1 = C2OUT is present on C12OUT pin <sup>(1)</sup> 0 = C2OUT is internal only
bit 4	<b>C2POL:</b> Comparator C2 Output Polarity Select bit 1 = C2OUT logic is inverted 0 = C2OUT logic is not inverted
bit 3	<b>C2SP:</b> Comparator C2 Speed/Power Select bit 1 = C2 operates in normal power, higher speed mode 0 = C2 operates in low-power, low-speed mode
bit 2	<b>C2R:</b> Comparator C2 Reference Select bits (non-inverting input) 1 = C2VIN+ connects to C2VREF 0 = C2VIN+ connects to C2IN+ pin
bit 1-0	<b>C2CH&lt;1:0&gt;:</b> Comparator C2 Channel Select bits 00 = C1VIN- connects to AGND 01 = C12IN1- pin of C2 connects to C2VIN- 10 = C12IN2- pin of C2 connects to C2VIN- 11 = C12IN3- pin of C2 connects to C2VIN-

**Note 1:** Comparator output requires the following three conditions: C2OE = 1, C2ON = 1 and corresponding port TRIS bit = 0.

## 18.7 Analog Input Connection Considerations

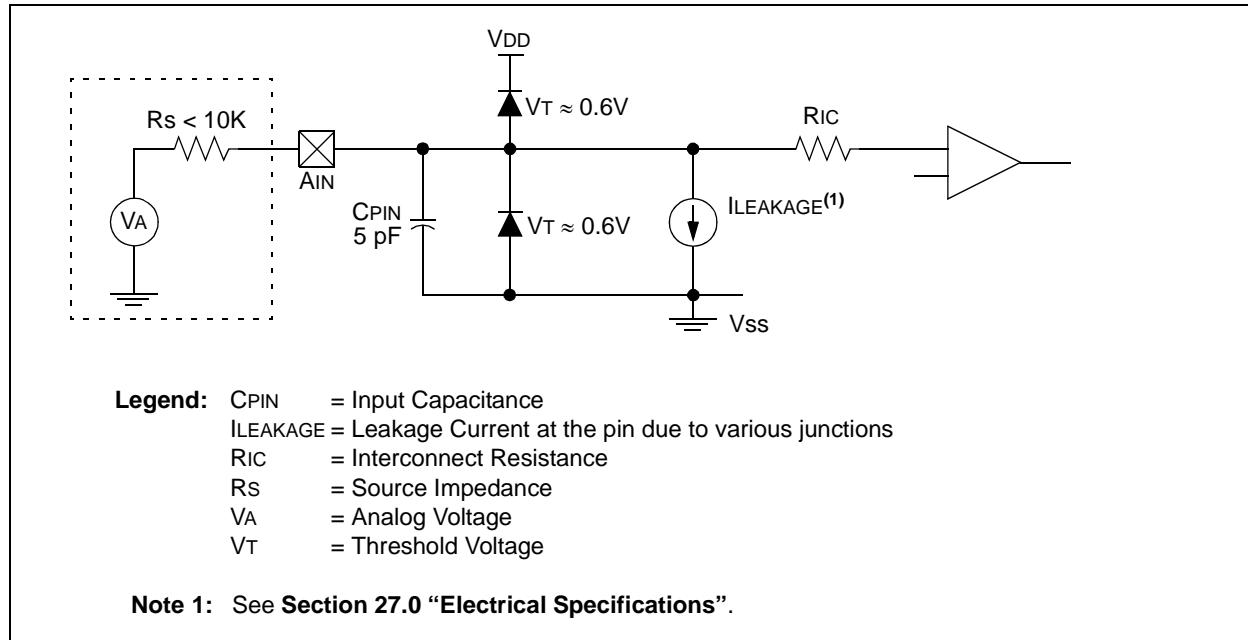
A simplified circuit for an analog input is shown in Figure 18-6. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to VDD and Vss. The analog input, therefore, must be between Vss and VDD. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur.

A maximum source impedance of  $10\text{ k}\Omega$  is recommended for the analog sources. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current to minimize inaccuracies introduced.

**Note 1:** When reading a PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert as an analog input, according to the input specification.

**2:** Analog levels on any pin defined as a digital input, may cause the input buffer to consume more current than is specified.

**FIGURE 18-6: ANALOG INPUT MODEL**



## 18.8 Additional Comparator Features

There are four additional comparator features:

- Simultaneous read of comparator outputs
- Internal reference selection
- Hysteresis selection
- Output Synchronization

### 18.8.1 SIMULTANEOUS COMPARATOR OUTPUT READ

The MC1OUT and MC2OUT bits of the CM2CON1 register are mirror copies of both comparator outputs. The ability to read both outputs simultaneously from a single register eliminates the timing skew of reading separate registers.

**Note 1:** Obtaining the status of C1OUT or C2OUT by reading CM2CON1 does not affect the comparator interrupt mismatch registers.

### 18.8.2 INTERNAL REFERENCE SELECTION

There are two internal voltage references available to the non-inverting input of each comparator. One of these is the Fixed Voltage Reference (FVR) and the other is the variable Comparator Voltage Reference (CVREF). The CxRSEL bit of the CM2CON register determines which of these references is routed to the Comparator Voltage reference output (CXVREF). Further routing to the comparator is accomplished by the CxR bit of the CMxCON0 register. See **Section 21.1 “Voltage Reference”** and Figure 18-2 and Figure 18-3 for more detail.

### 18.8.3 COMPARATOR HYSTERESIS

The Comparator Cx have selectable hysteresis. The hysteresis can be enable by setting the CxHYS bit of the CM2CON1 register. See **Section 27.0 “Electrical Specifications”** for more details.

### 18.8.4 SYNCHRONIZING COMPARATOR OUTPUT TO TIMER 1

The Comparator Cx output can be synchronized with Timer1 by setting the CxSYNC bit of the CM2CON1 register. When enabled, the Cx output is latched on the rising edge of the Timer1 source clock. If a prescaler is used with Timer1, the comparator output is latched after the prescaling function. To prevent a race condition, the comparator output is latched on the rising edge of the Timer1 clock source and Timer1 increments on the rising edge of its clock source. See the Comparator Block Diagram (Figure 18-2 and Figure 18-3) and the Timer1 Block Diagram (Figure 18-2) for more information.

## REGISTER 18-3: CM2CON1: COMPARATOR 2 CONTROL REGISTER 1

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>MC1OUT:</b> Mirror Copy of C1OUT bit
bit 6	<b>MC2OUT:</b> Mirror Copy of C2OUT bit
bit 5	<b>C1RSEL:</b> Comparator C1 Reference Select bit 1 = FVR routed to C1VREF input 0 = CVREF routed to C1VREF input
bit 4	<b>C2RSEL:</b> Comparator C2 Reference Select bit 1 = FVR routed to C2VREF input 0 = CVREF routed to C2VREF input
bit 3	<b>C1HYS:</b> Comparator C1 Hysteresis Enable bit 1 = Comparator C1 hysteresis enabled 0 = Comparator C1 hysteresis disabled
bit 2	<b>C2HYS:</b> Comparator C2 Hysteresis Enable bit 1 = Comparator C2 hysteresis enabled 0 = Comparator C2 hysteresis disabled
bit 1	<b>C1SYNC:</b> C1 Output Synchronous Mode bit 1 = C1 output is synchronous to rising edge to TMR1 clock 0 = C1 output is asynchronous
bit 0	<b>C2SYNC:</b> C2 Output Synchronous Mode bit 1 = C2 output is synchronous to rising edge to TMR1 clock 0 = C2 output is asynchronous

# PIC18F1XK50/PIC18LF1XK50

---

TABLE 18-3: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
CM1CON0	C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH1	C1CH0	282
CM2CON0	C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH1	C2CH0	282
CM2CON1	MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC	282
REFCON0	FVR1EN	FVR1ST	FVR1S1	FVR1S0	TSEN	TSRS	—	—	281
REFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	—	D1NSS	281
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	279
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	282
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	282
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	282
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	282
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	282
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	282

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

## 19.0 POWER-MANAGED MODES

PIC18F1XK50/PIC18LF1XK50 devices offer a total of seven operating modes for more efficient power management. These modes provide a variety of options for selective power conservation in applications where resources may be limited (i.e., battery-powered devices).

There are three categories of power-managed modes:

- Run modes
- Idle modes
- Sleep mode

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

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

### 19.1 Selecting Power-Managed Modes

Selecting a power-managed mode requires two decisions:

- Whether or not the CPU is to be clocked
- The selection of a clock source

The IDLEN bit of the OSCCON register controls CPU clocking, while the SCS<1:0> bits of the OSCCON register select the clock source. The individual modes, bit settings, clock sources and affected modules are summarized in Table 19-1.

#### 19.1.1 CLOCK SOURCES

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

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

#### 19.1.2 ENTERING POWER-MANAGED MODES

Switching from one power-managed mode to another begins by loading the OSCCON register. The SCS<1:0> bits select the clock source and determine which Run or Idle mode is to be used. Changing these bits causes an immediate switch to the new clock source, assuming that it is running. The switch may also be subject to clock transition delays. Refer to **Section 2.8 “Clock Switching”** for more information.

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

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

**TABLE 19-1: POWER-MANAGED MODES**

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

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

**2:** Includes HFINTOSC and HFINTOSC postscaler, as well as the LFINTOSC source.

### 19.1.3 MULTIPLE FUNCTIONS OF THE SLEEP COMMAND

The power-managed mode that is invoked with the *SLEEP* instruction is determined by the setting of the *IDLEN* bit of the *OSCCON* register at the time the instruction is executed. All clocks stop and minimum power is consumed when *SLEEP* is executed with the *IDLEN* bit cleared. The system clock continues to supply a clock to the peripherals but is disconnected from the CPU when *SLEEP* is executed with the *IDLEN* bit set.

## 19.2 Run Modes

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

### 19.2.1 PRI\_RUN MODE

The *PRI\_RUN* mode is the normal, full power execution mode of the microcontroller. This is also the default mode upon a device Reset, unless Two-Speed Start-up is enabled (see **Section 2.12 “Two-Speed Start-up Mode”** for details). In this mode, the device operated off the oscillator defined by the *FOSC* bits of the *CONFIGH* Configuration register.

### 19.2.2 SEC\_RUN MODE

In *SEC\_RUN* mode, the CPU and peripherals are clocked from the secondary external oscillator. This gives users the option of lower power consumption while still using a high accuracy clock source.

*SEC\_RUN* mode is entered by setting the *SCS<1:0>* bits of the *OSCCON* register to '01'. When *SEC\_RUN* mode is active all of the following are true:

- The main clock source is switched to the secondary external oscillator
- Primary external oscillator is shut down
- *T1RUN* bit of the *T1CON* register is set
- *OSTS* bit is cleared.

**Note:** The secondary external oscillator should already be running prior to entering *SEC\_RUN* mode. If the *T1OSCEN* bit is not set when the *SCS<1:0>* bits are set to '01', entry to *SEC\_RUN* mode will not occur until *T1OSCEN* bit is set and secondary external oscillator is ready.

### 19.2.3 RC\_RUN MODE

In *RC\_RUN* mode, the CPU and peripherals are clocked from the internal oscillator. In this mode, the primary external oscillator is shut down. *RC\_RUN* mode provides the best power conservation of all the Run modes when the *LFINTOSC* is the system clock.

*RC\_RUN* mode is entered by setting the *SCS1* bit. When the clock source is switched from the primary oscillator to the internal oscillator, the primary oscillator is shut down and the *OSTS* bit is cleared. The *IRCF* bits may be modified at any time to immediately change the clock speed.

## 19.3 Sleep Mode

The Power-Managed Sleep mode in the PIC18F1XK50/PIC18LF1XK50 devices is identical to the legacy Sleep mode offered in all other PIC® microcontroller devices. It is entered by clearing the IDLEN bit of the OSCCON register and executing the SLEEP instruction. This shuts down the selected oscillator (Figure 19-1) and all clock source status bits are cleared.

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

When a wake event occurs in Sleep mode (by interrupt, Reset or WDT time-out), the device will not be clocked until the clock source selected by the SCS<1:0> bits becomes ready (see Figure 19-2), or it will be clocked from the internal oscillator block if either the Two-Speed Start-up or the Fail-Safe Clock Monitor are enabled (see **Section 24.0 “Special Features of the CPU”**). In either case, the OSTST bit is set when the primary clock is providing the device clocks. The IDLEN and SCS bits are not affected by the wake-up.

## 19.4 Idle Modes

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

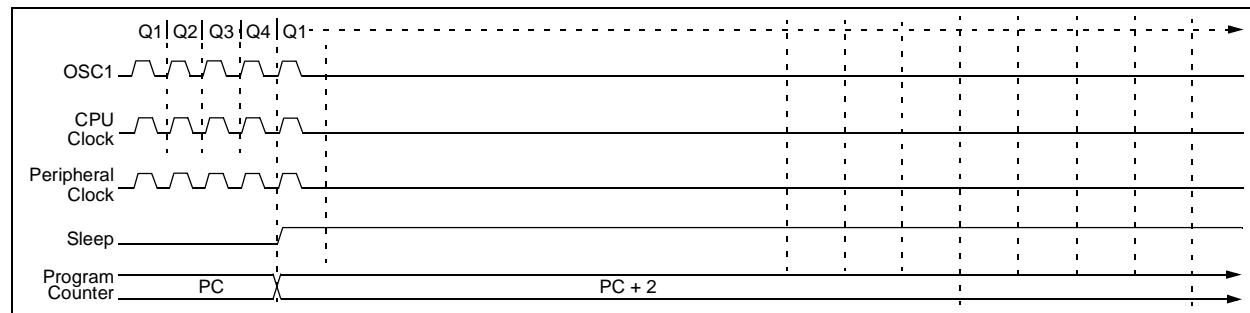
If the IDLEN bit is set to a '1' when a SLEEP instruction is executed, the peripherals will be clocked from the clock source selected by the SCS<1:0> bits; however, the CPU will not be clocked. The clock source status bits are not affected. Setting IDLEN and executing a SLEEP instruction provides a quick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

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

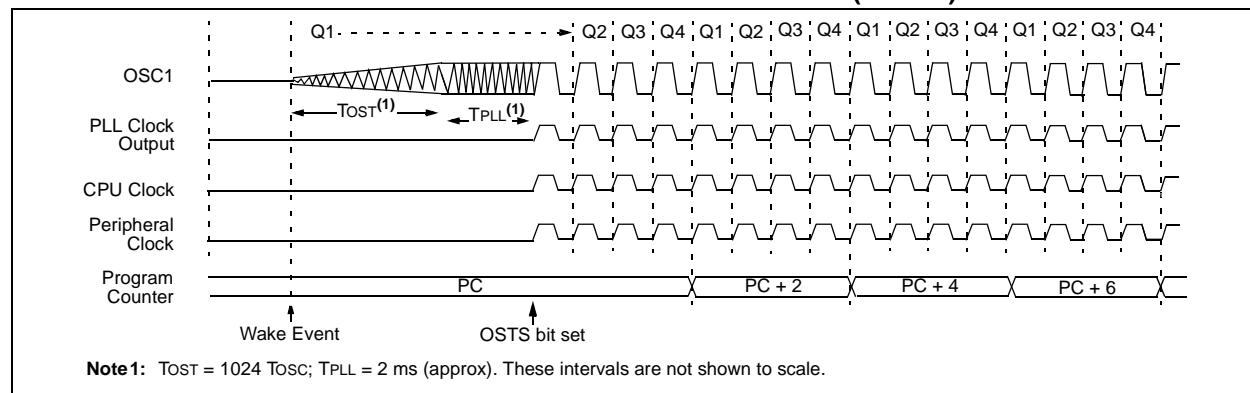
Since the CPU is not executing instructions, the only exits from any of the Idle modes are by interrupt, WDT time-out, or a Reset. When a wake event occurs, CPU execution is delayed by an interval of TcSD while it becomes ready to execute code. When the CPU begins executing code, it resumes with the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

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

**FIGURE 19-1: TRANSITION TIMING FOR ENTRY TO SLEEP MODE**



**FIGURE 19-2: TRANSITION TIMING FOR WAKE FROM SLEEP (HSPLL)**



# PIC18F1XK50/PIC18LF1XK50

## 19.4.1 PRI\_IDLE MODE

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

PRI\_IDLE mode is entered from PRI\_RUN mode by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set IDLEN first, then clear the SCS bits and execute SLEEP. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified by the FOSC<3:0> Configuration bits. The OSTST bit remains set (see Figure 19-3).

When a wake event occurs, the CPU is clocked from the primary clock source. A delay of interval TcSD is required between the wake event and when code execution starts. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTST bit remains set. The IDLEN and SCS bits are not affected by the wake-up (see Figure 19-4).

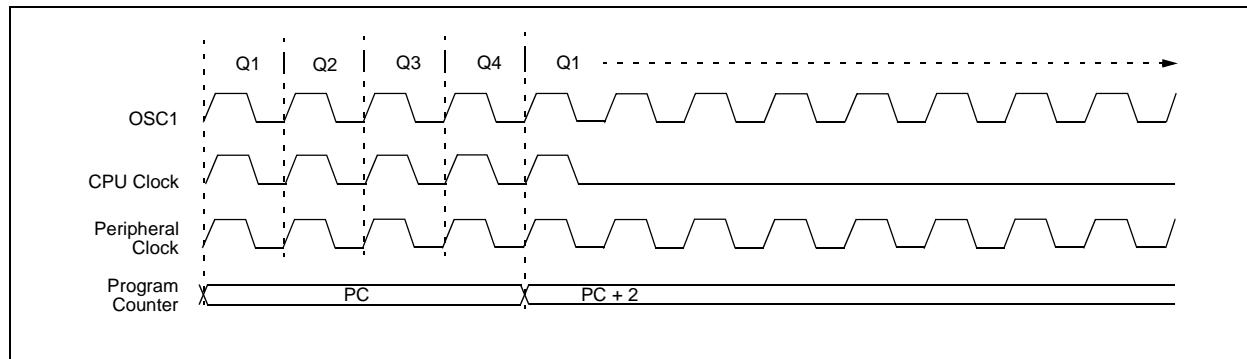
## 19.4.2 SEC\_IDLE MODE

In SEC\_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the Timer1 oscillator. This mode is entered from SEC\_RUN by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set the IDLEN bit first, then set the SCS<1:0> bits to ‘01’ and execute SLEEP. When the clock source is switched to the Timer1 oscillator, the primary oscillator is shut down, the OSTST bit is cleared and the T1RUN bit is set.

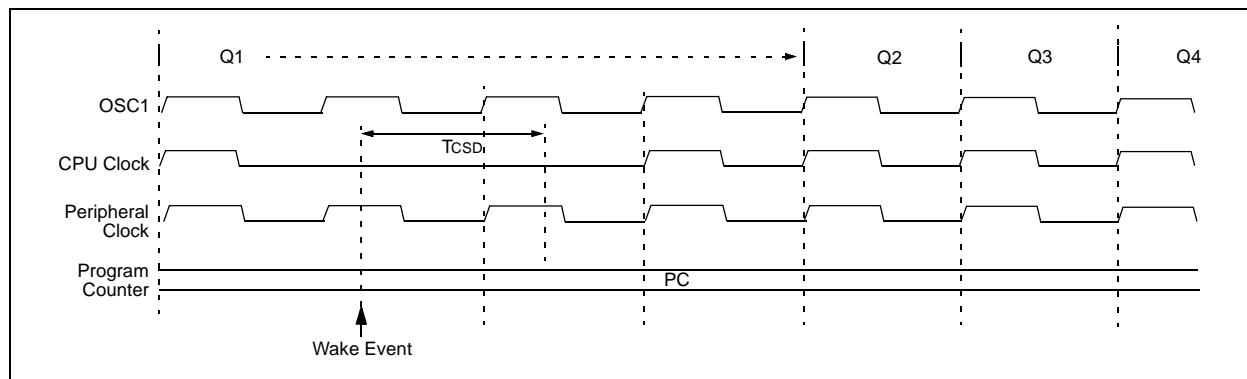
When a wake event occurs, the peripherals continue to be clocked from the Timer1 oscillator. After an interval of TcSD following the wake event, the CPU begins executing code being clocked by the Timer1 oscillator. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run (see Figure 19-4).

**Note:** The Timer1 oscillator should already be running prior to entering SEC\_IDLE mode. If the T1OSCEN bit is not set when the SLEEP instruction is executed, the main system clock will continue to operate in the previously selected mode and the corresponding IDLE mode will be entered (i.e., PRI\_IDLE or RC\_IDLE).

**FIGURE 19-3: TRANSITION TIMING FOR ENTRY TO IDLE MODE**



**FIGURE 19-4: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE**



### 19.4.3 RC\_IDLE MODE

In RC\_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the internal oscillator block from the HFINTOSC multiplexer output. This mode allows for controllable power conservation during Idle periods.

From RC\_RUN, this mode is entered by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, first set IDLEN, then set the SCS1 bit and execute SLEEP. It is recommended that SCS0 also be cleared, although its value is ignored, to maintain software compatibility with future devices. The HFINTOSC multiplexer may be used to select a higher clock frequency by modifying the IRCF bits before executing the SLEEP instruction. When the clock source is switched to the HFINTOSC multiplexer, the primary oscillator is shut down and the OSTS bit is cleared.

If the IRCF bits are set to any non-zero value, or the INTSRC bit is set, the HFINTOSC output is enabled. The IOSF bit becomes set, after the HFINTOSC output becomes stable, after an interval of TIOBST. Clocks to the peripherals continue while the HFINTOSC source stabilizes. If the IRCF bits were previously at a non-zero value, or INTSRC was set before the SLEEP instruction was executed and the HFINTOSC source was already stable, the IOSF bit will remain set. If the IRCF bits and INTSRC are all clear, the HFINTOSC output will not be enabled, the IOSF bit will remain clear and there will be no indication of the current clock source.

When a wake event occurs, the peripherals continue to be clocked from the HFINTOSC multiplexer output. After a delay of Tcsd following the wake event, the CPU begins executing code being clocked by the HFINTOSC multiplexer. The IDLEN and SCS bits are not affected by the wake-up. The LFINTOSC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

## 19.5 Exiting Idle and Sleep Modes

An exit from Sleep mode or any of the Idle modes is triggered by any one of the following:

- an interrupt
- a Reset
- a Watchdog Time-out

This section discusses the triggers that cause exits from power-managed modes. The clocking subsystem actions are discussed in each of the power-managed modes (see **Section 19.2 “Run Modes”**, **Section 19.3 “Sleep Mode”** and **Section 19.4 “Idle Modes”**).

### 19.5.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit from an Idle mode or the Sleep mode to a Run mode. To enable this functionality, an interrupt source must be enabled by setting its enable bit in one of the INTCON or PIE registers. The PEIE bit must also be set if the desired interrupt enable bit is in a PIE register. The exit sequence is initiated when the corresponding interrupt flag bit is set.

The instruction immediately following the SLEEP instruction is executed on all exits by interrupt from Idle or Sleep modes. Code execution then branches to the interrupt vector if the GIE/GIEH bit of the INTCON register is set, otherwise code execution continues without branching (see **Section 7.0 “Interrupts”**).

A fixed delay of interval Tcsd following the wake event is required when leaving Sleep and Idle modes. This delay is required for the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

### 19.5.2 EXIT BY WDT TIME-OUT

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

If the device is not executing code (all Idle modes and Sleep mode), the time-out will result in an exit from the power-managed mode (see **Section 19.2 “Run Modes”** and **Section 19.3 “Sleep Mode”**). If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see **Section 24.2 “Watch-dog Timer (WDT)”**).

The WDT timer and postscaler are cleared by any one of the following:

- executing a SLEEP instruction
- executing a CLRWDAT instruction
- the loss of the currently selected clock source when the Fail-Safe Clock Monitor is enabled
- modifying the IRCF bits in the OSCCON register when the internal oscillator block is the device clock source

# PIC18F1XK50/PIC18LF1XK50

## 19.5.3 EXIT BY RESET

Exiting Sleep and Idle modes by Reset causes code execution to restart at address 0. See **Section 23.0 “Reset”** for more details.

The exit delay time from Reset to the start of code execution depends on both the clock sources before and after the wake-up and the type of oscillator. Exit delays are summarized in Table 19-2.

## 19.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

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

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

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

**TABLE 19-2: EXIT DELAY ON WAKE-UP BY RESET FROM SLEEP MODE OR ANY IDLE MODE (BY CLOCK SOURCES)**

Clock Source before Wake-up	Clock Source after Wake-up	Exit Delay	Clock Ready Status Bit (OSCCON)
Primary Device Clock (PRI_IDLE mode)	LP, XT, HS	Tcsd <sup>(1)</sup>	OSTS
	HSPLL		
	EC, RC		IOSF
	HFINTOSC <sup>(2)</sup>		
T1OSC or LFINTOSC <sup>(1)</sup>	LP, XT, HS	Tost <sup>(3)</sup>	OSTS
	HSPLL	Tost + t <sub>PLL</sub> <sup>(3)</sup>	
	EC, RC	Tcsd <sup>(1)</sup>	
	HFINTOSC <sup>(1)</sup>	TIOBST <sup>(4)</sup>	IOSF
HFINTOSC <sup>(2)</sup>	LP, XT, HS	Tost <sup>(4)</sup>	OSTS
	HSPLL	Tost + t <sub>PLL</sub> <sup>(3)</sup>	
	EC, RC	Tcsd <sup>(1)</sup>	
	HFINTOSC <sup>(1)</sup>	None	IOSF
None (Sleep mode)	LP, XT, HS	Tost <sup>(3)</sup>	OSTS
	HSPLL	Tost + t <sub>PLL</sub> <sup>(3)</sup>	
	EC, RC	Tcsd <sup>(1)</sup>	
	HFINTOSC <sup>(1)</sup>	TIOBST <sup>(4)</sup>	IOSF

**Note 1:** Tcsd is a required delay when waking from Sleep and all Idle modes and runs concurrently with any other required delays (see **Section 19.4 “Idle Modes”**). On Reset, HFINTOSC defaults to 1 MHz.

**2:** Includes both the HFINTOSC 16 MHz source and postscaler derived frequencies.

**3:** Tost is the Oscillator Start-up Timer. t<sub>PLL</sub> is the PLL Lock-out Timer (parameter F12).

**4:** Execution continues during the HFINTOSC stabilization period, TIOBST.

## 20.0 SR LATCH

The module consists of a single SR Latch with multiple Set and Reset inputs as well as selectable latch output. The SR Latch module includes the following features:

- Programmable input selection
- SR Latch output is available internally/externally
- Selectable Q and  $\bar{Q}$  output
- Firmware Set and Reset

## 20.1 Latch Operation

The latch is a Set-Reset latch that does not depend on a clock source. Each of the Set and Reset inputs are active-high. The latch can be Set or Reset by CxOUT, INT1 pin, or variable clock. Additionally the SRPS and the SRPR bits of the SRCON0 register may be used to Set or Reset the SR Latch, respectively. The latch is reset-dominant, therefore, if both Set and Reset inputs are high the latch will go to the Reset state. Both the SRPS and SRPR bits are self resetting which means that a single write to either of the bits is all that is necessary to complete a latch Set or Reset operation.

## 20.2 Latch Output

The SRQEN and SRNQEN bits of the SRCON0 register control the latch output selection. Only one of the SR latch's outputs may be directly output to an I/O pin at a time. Priority is determined by the state of bits SRQEN and SRNQEN in registers SRCON0.

**TABLE 20-1: SR LATCH OUTPUT CONTROL**

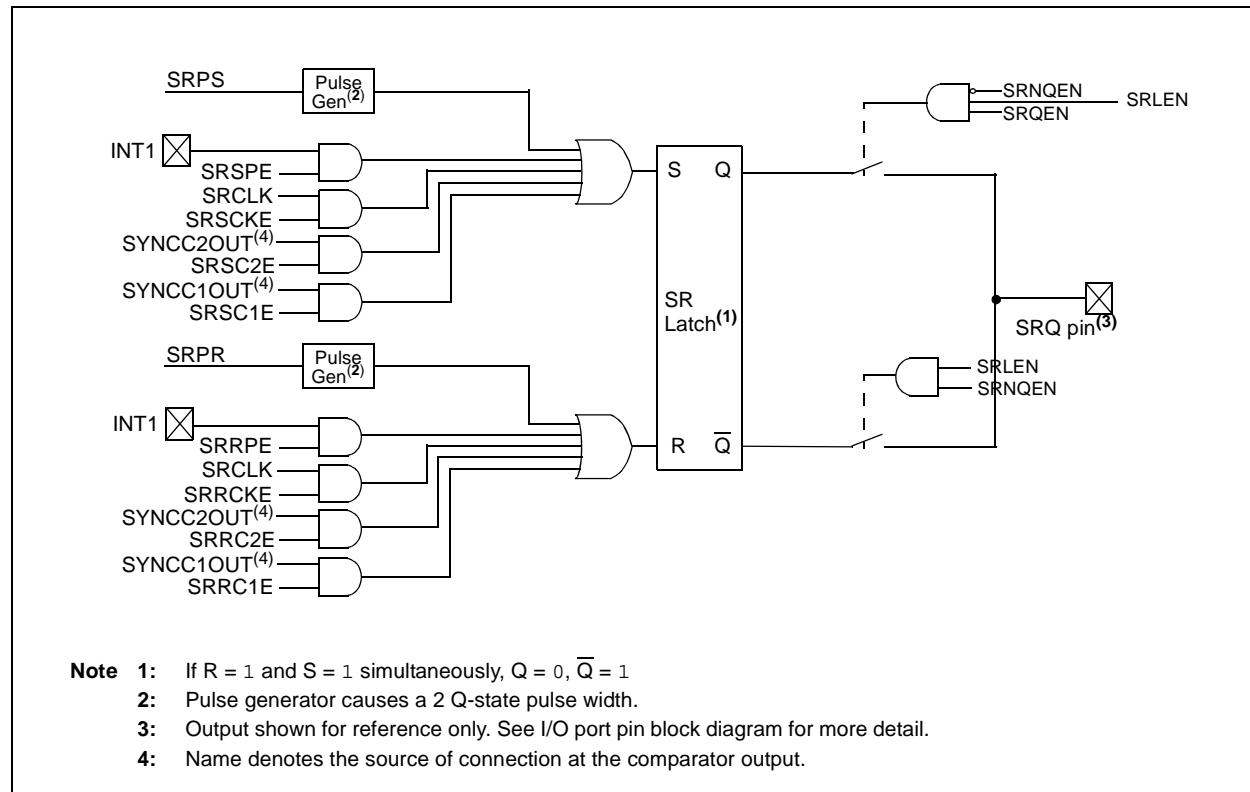
SRLEN	SRQEN	SRNQEN	SR Latch Output to Port I/O
0	X	X	I/O
1	0	0	I/O
1	0	1	$\bar{Q}$
1	1	0	Q
1	1	1	$\bar{Q}$

The applicable TRIS bit of the corresponding port must be cleared to enable the port pin output driver.

## 20.3 Effects of a Reset

Upon any device Reset, the SR latch is not initialized. The user's firmware is responsible to initialize the latch output before enabling it to the output pins.

**FIGURE 20-1: SR LATCH SIMPLIFIED BLOCK DIAGRAM**



# PIC18F1XK50/PIC18LF1XK50

TABLE 20-1: SRCLK FREQUENCY TABLE

SRCLK	Divider	Fosc = 20 MHz	Fosc = 16 MHz	Fosc = 8 MHz	Fosc = 4 MHz	Fosc = 1 MHz
111	512	25.6 $\mu$ s	32 $\mu$ s	64 $\mu$ s	128 $\mu$ s	512 $\mu$ s
110	256	12.8 $\mu$ s	16 $\mu$ s	32 $\mu$ s	64 $\mu$ s	256 $\mu$ s
101	128	6.4 $\mu$ s	8 $\mu$ s	16 $\mu$ s	32 $\mu$ s	128 $\mu$ s
100	64	3.2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	16 $\mu$ s	64 $\mu$ s
011	32	1.6 $\mu$ s	2 $\mu$ s	4 $\mu$ s	8 $\mu$ s	32 $\mu$ s
010	16	0.8 $\mu$ s	1 $\mu$ s	2 $\mu$ s	4 $\mu$ s	16 $\mu$ s
001	8	0.4 $\mu$ s	0.5 $\mu$ s	1 $\mu$ s	2 $\mu$ s	8 $\mu$ s
000	4	0.2 $\mu$ s	0.25 $\mu$ s	0.5 $\mu$ s	1 $\mu$ s	4 $\mu$ s

REGISTER 20-2: SRCON0: SR LATCH CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SRLEN	SRCLK2	SRCLK1	SRCLK0	SRQEN	SRNQEN	SRPS	SRPR
bit 7	bit 0						

**Legend:**

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

U = Unimplemented  
'0' = Bit is cleared

C = Clearable only bit  
x = Bit is unknown

bit 7 **SRLEN:** SR Latch Enable bit<sup>(1)</sup>

1 = SR latch is enabled  
0 = SR latch is disabled

bit 6-4 **SRCLK<2:0><sup>(1)</sup>:** SR Latch Clock divider bits

000 = 1/4 Peripheral cycle clock  
001 = 1/8 Peripheral cycle clock  
010 = 1/16 Peripheral cycle clock  
011 = 1/32 Peripheral cycle clock  
100 = 1/64 Peripheral cycle clock  
101 = 1/128 Peripheral cycle clock  
110 = 1/256 Peripheral cycle clock  
111 = 1/512 Peripheral cycle clock

bit 3 **SRQEN:** SR Latch Q Output Enable bit

If SRNQEN = 0  
1 = Q is present on the RC4 pin  
0 = Q is internal only

bit 2 **SRNQEN:** SR Latch  $\bar{Q}$  Output Enable bit

1 =  $\bar{Q}$  is present on the RC4 pin  
0 =  $\bar{Q}$  is internal only

bit 1 **SRPS:** Pulse Set Input of the SR Latch

1 = Pulse input  
0 = Always reads back '0'

bit 0 **SRPR:** Pulse Reset Input of the SR Latch

1 = Pulse input  
0 = Always reads back '0'

**Note 1:** Changing the SRCLK bits while the SR latch is enabled may cause false triggers to the set and Reset inputs of the latch.

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 20-3: SRCON1: SR LATCH CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SRSPE	SRSCKE	SRSC2E	SRSC1E	SRRPE	SRRCKE	SRRC2E	SRRC1E
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented

C = Clearable only bit

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>SRSPE:</b> SR Latch Peripheral Set Enable bit 1 = INT1 pin status sets SR Latch 0 = INT1 pin status has no effect on SR Latch
bit 6	<b>SRSCKE:</b> SR Latch Set Clock Enable bit 1 = Set input of SR latch is pulsed with SRCLK 0 = Set input of SR latch is not pulsed with SRCLK
bit 5	<b>SRSC2E:</b> SR Latch C2 Set Enable bit 1 = C2 Comparator output sets SR Latch 0 = C2 Comparator output has no effect on SR Latch
bit 4	<b>SRSC1E:</b> SR Latch C1 Set Enable bit 1 = C1 Comparator output sets SR Latch 0 = C1 Comparator output has no effect on SR Latch
bit 3	<b>SRRPE:</b> SR Latch Peripheral Reset Enable bit 1 = INT1 pin resets SR Latch 0 = INT1 pin has no effect on SR Latch
bit 2	<b>SRRCKE:</b> SR Latch Reset Clock Enable bit 1 = Reset input of SR latch is pulsed with SRCLK 0 = Reset input of SR latch is not pulsed with SRCLK
bit 1	<b>SRRC2E:</b> SR Latch C2 Reset Enable bit 1 = C2 Comparator output resets SR Latch 0 = C2 Comparator output has no effect on SR Latch
bit 0	<b>SRRC1E:</b> SR Latch C1 Reset Enable bit 1 = C1 Comparator output resets SR Latch 0 = C1 Comparator output has no effect on SR Latch

## TABLE 20-2: REGISTERS ASSOCIATED WITH THE SR LATCH

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
SRCON0	SRLEN	SRCLK2	SRCLK1	SRCLK0	SRQEN	SRNQEN	SRPS	SRPR	282
SRCON1	SRSPE	SRSCKE	SRSC2E	SRSC1E	SRRPE	SRRCKE	SRRC2E	SRRC1E	282
CM2CON1	MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC	282
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	279
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282

Legend: Shaded cells are not used with the comparator voltage reference.

Note 1: PORTA pins are enabled based on oscillator configuration.

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 21.0 VOLTAGE REFERENCES

There are two independent voltage references available:

- Programmable Voltage Reference
- 1.024V Fixed Voltage Reference

### 21.1 Voltage Reference

The Voltage Reference module provides an internally generated voltage reference for the comparators and the DAC module. The following features are available:

- Independent from Comparator operation
- Single 32-level voltage ranges
- Output clamped to Vss
- Ratiometric with VDD
- 1.024V Fixed Reference Voltage (FVR)

The REFCON1 register (Register 21-2) controls the Voltage Reference module shown in Figure 21-1.

#### 21.1.1 INDEPENDENT OPERATION

The voltage reference is independent of the comparator configuration. Setting the D1EN bit of the REFCON1 register will enable the voltage reference by allowing current to flow in the VREF voltage divider. When the D1EN bit is cleared, current flow in the VREF voltage divider is disabled minimizing the power drain of the voltage reference peripheral.

#### 21.1.2 OUTPUT VOLTAGE SELECTION

The VREF voltage reference has 32 voltage level ranges. The 32 levels are set with the DAC1R<4:0> bits of the REFCON2 register.

The VREF output voltage is determined by the following equations:

#### EQUATION 21-1: VREF OUTPUT VOLTAGE

$$V_{OUT} = (V_{SOURCE+} - V_{SOURCE-}) * (DAC1R<4:0> / (2^5)) + V_{SOURCE-}$$

$V_{SOURCE+}$  = VDD, VREF+, or FVR

$V_{SOURCE-}$  = VSS, or VREF-

#### 21.1.3 OUTPUT CLAMPED TO Vss

The VREF output voltage can be set to Vss with no power consumption by setting the D1EN bit of the REFCON1 register to '0': This allows the comparator to detect a zero-crossing while not consuming additional VREF module current.

#### 21.1.4 OUTPUT RATIO METRIC TO VDD

The comparator voltage reference is VDD derived and therefore, the VREF output changes with fluctuations in VDD. The tested absolute accuracy of the Comparator Voltage Reference can be found in **Section 27.0 “Electrical Specifications”**.

#### 21.1.5 VOLTAGE REFERENCE OUTPUT

The VREF voltage reference can be output to the device CVREF pin by setting the DAC1OE bit of the REFCON1 register to '1'. Selecting the reference voltage for output on the VREF pin automatically overrides the digital output buffer and digital input threshold detector functions of that pin. Reading the CVREF pin when it has been configured for reference voltage output will always return a '0'.

Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to CVREF. Figure 21-2 shows an example buffering technique.

#### 21.1.6 OPERATION DURING SLEEP

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the RECON1 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

#### 21.1.7 EFFECTS OF A RESET

A device Reset affects the following:

- Voltage reference is disabled
- Fixed voltage reference is disabled
- VREF is removed from the CVREF pin
- The DAC1R<4:0> range select bits are cleared

## 21.2 FVR Reference Module

The FVR reference is a stable fixed voltage reference, independent of VDD, with a nominal output voltage of 1.024V. This reference can be enabled by setting the FVR1EN bit of the REFCON0 register to '1'. The FVR voltage reference can be routed to the comparators or an ADC input channel.

#### 21.2.1 FVR STABILIZATION PERIOD

When the Fixed Voltage Reference module is enabled, it will require some time for the reference and its amplifier circuits to stabilize. The user program must include a small delay routine to allow the module to settle. The FVR1ST stable bit of the REFCON0 register also indicates that the FVR reference has been operating long enough to be stable. See **Section 27.0 “Electrical Specifications”** for the minimum delay requirement.

# PIC18F1XK50/PIC18LF1XK50

FIGURE 21-1: VOLTAGE REFERENCE BLOCK DIAGRAM

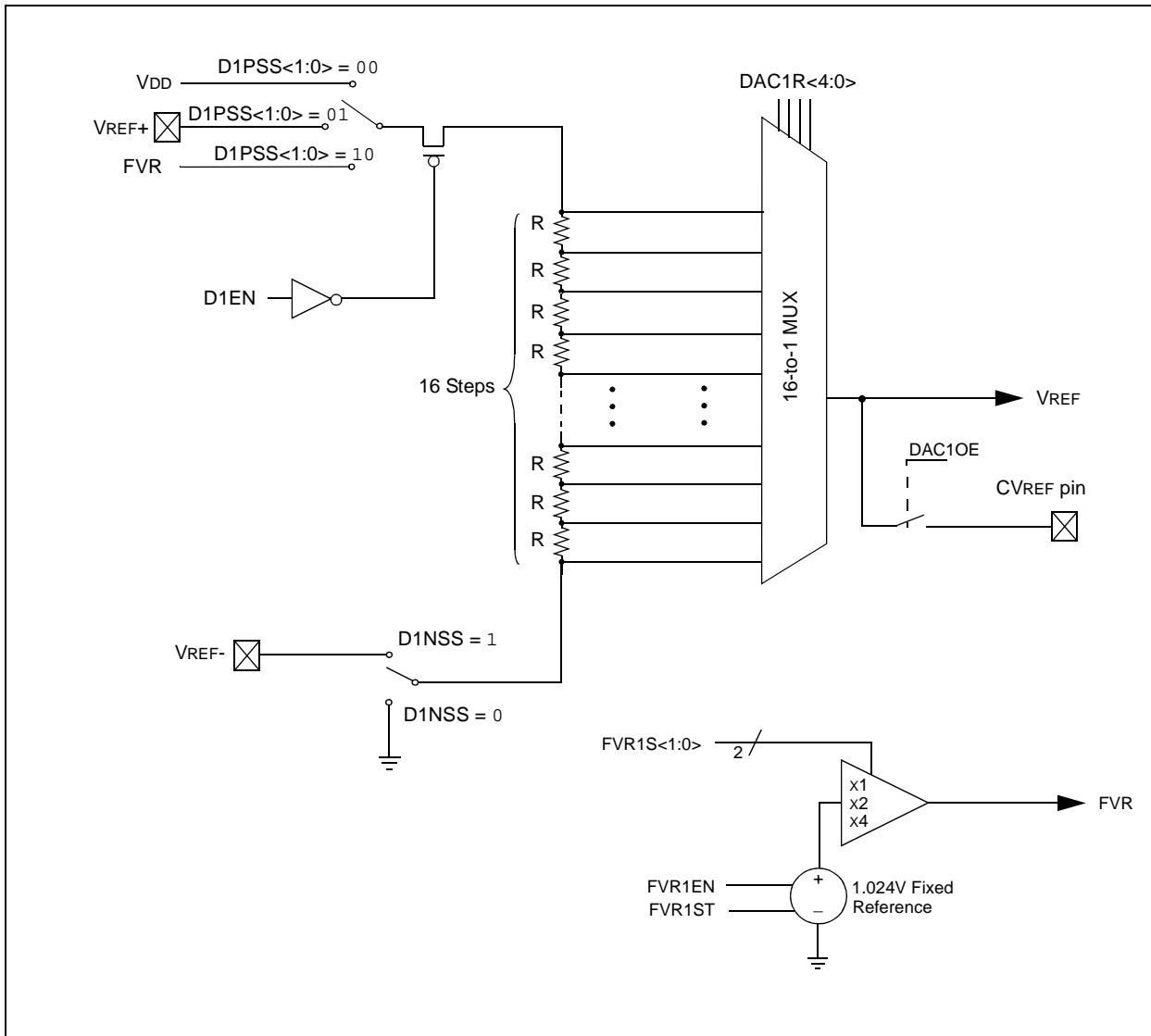
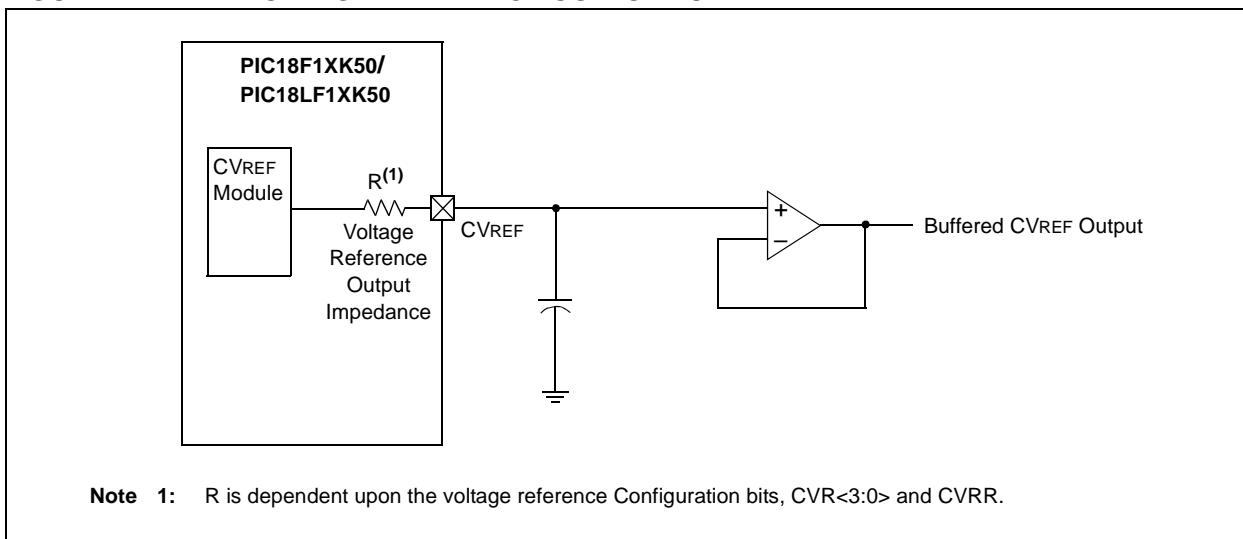


FIGURE 21-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE



# PIC18F1XK50/PIC18LF1XK50

## REGISTER 21-1: REFCONO: REFERENCE CONTROL REGISTER 0

R/W-0	R-0	R/W-0	R/W-1	U-0	U-0	U-0	U-0
FVR1EN	FVR1ST	FVR1S1	FVR1S0	—	—	—	—
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>FVR1EN:</b> Fixed Voltage Reference 1 Enable bit 0 = FVR is disabled 1 = FVR is enabled
bit 6	<b>FVR1ST:</b> Fixed Voltage Reference 1 Stable bit 0 = FVR is not stable 1 = FVR is stable
bit 5-4	<b>FVR1S&lt;1:0&gt;:</b> Fixed Voltage Reference 1 Voltage Select bits 00 = Reserved, do not use 01 = 1.024V (x1) 10 = 2.048V (x2) 11 = 4.096V (x4)
bit 3-0	<b>Unimplemented:</b> Read as '0'

## REGISTER 21-2: REFCON1: REFERENCE CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0
D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	---	D1NSS
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>D1EN:</b> DAC 1 Enable bit 0 = DAC 1 is disabled 1 = DAC 1 is enabled
bit 6	<b>D1LPS:</b> DAC 1 Low-Power Voltage State Select bit 0 = VDAC = DAC1 Negative reference source selected 1 = VDAC = DAC1 Positive reference source selected
bit 5	<b>DAC1OE:</b> DAC 1 Voltage Output Enable bit 1 = DAC 1 voltage level is also outputed on the RC2/AN6/P1D/C12IN2-/CVREF/INT2 pin 0 = DAC 1 voltage level is disconnected from RC2/AN6/P1D/C12IN2-/CVREF/INT2 pin
bit 4	<b>Unimplemented:</b> Read as '0'
bit 3-2	<b>D1PSS&lt;1:0&gt;:</b> DAC 1 Positive Source Select bits 00 = VDD 01 = VREF+ 10 = FVR output 11 = Reserved, do not use
bit 1	<b>Unimplemented:</b> Read as '0'
bit 0	<b>D1NSS:</b> DAC1 Negative Source Select bits 0 = VSS 1 = VREF-

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 21-3: REFCON2: REFERENCE CONTROL REGISTER 2

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
---	---	---	DAC1R4	DAC1R3	DAC1R2	DAC1R1	DAC1R0
bit 7							

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5      **Unimplemented:** Read as '0'bit 4-0      **DAC1R<4:0>:** DAC1 Voltage Output Select bits $V_{OUT} = ((V_{SOURCE+}) - (V_{SOURCE-})) * (DAC1R<4:0>) / (2^5) + V_{SOURCE-}$ **Note 1:** The output select bits are always right justified to ensure that any number of bits can be used without affecting the register layout.**TABLE 21-1: REGISTERS ASSOCIATED WITH VOLTAGE REFERENCE**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
REFCON0	FVR1EN	FVR1ST	FVR1S1	FVR1S0	—	—	—	—	281
REFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	—	D1NSS	281
REFCON2	—	—	—	DAC1R4	DAC1R3	DAC1R2	DAC1R1	DAC1R0	281
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	282

**Legend:** Shaded cells are not used with the comparator voltage reference.**Note 1:** PORTA pins are enabled based on oscillator configuration.

## 22.0 UNIVERSAL SERIAL BUS (USB)

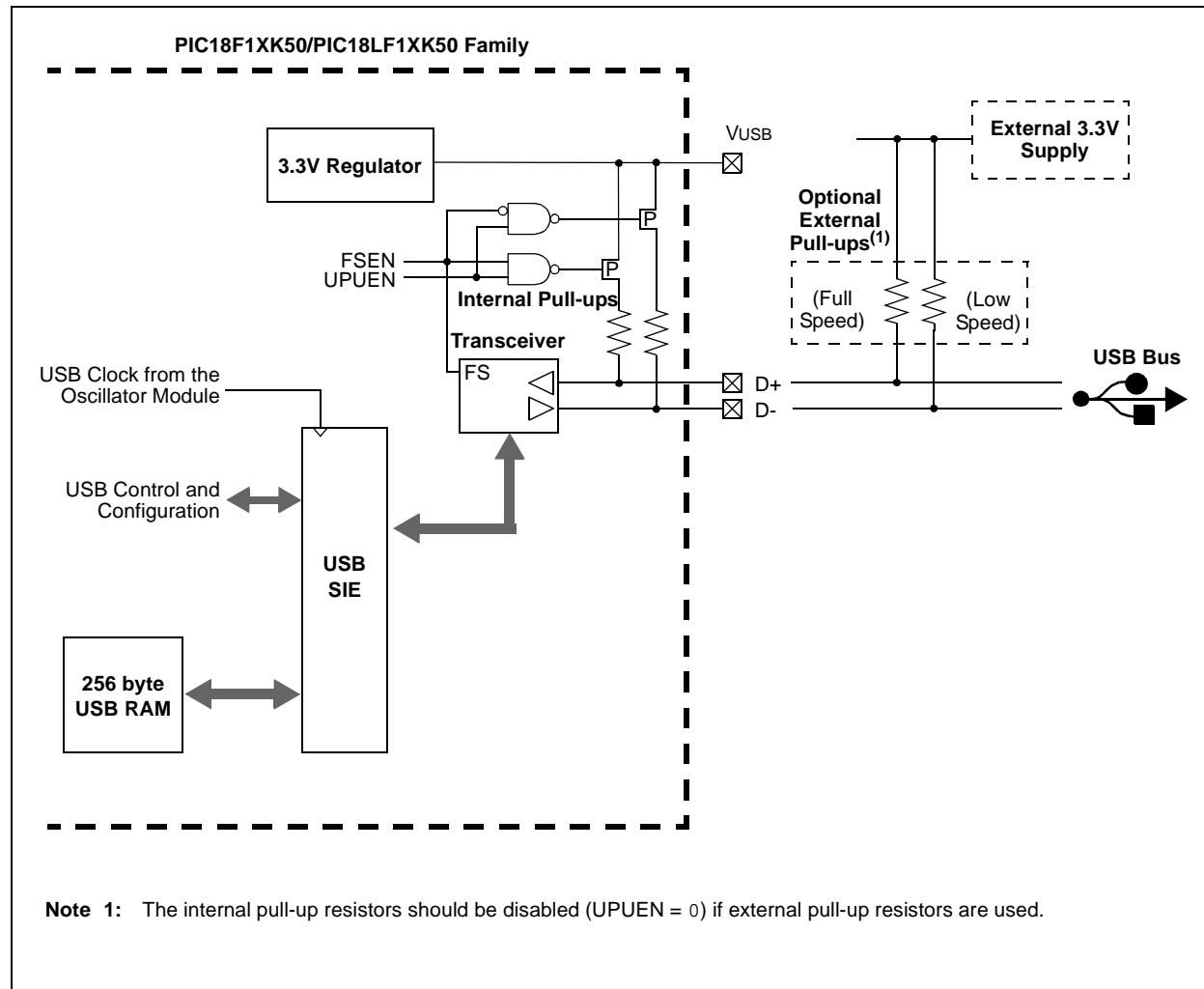
This section describes the details of the USB peripheral. Because of the very specific nature of the module, knowledge of USB is expected. Some high-level USB information is provided in **Section 22.10 “Overview of USB”** only for application design reference. Designers are encouraged to refer to the official specification published by the USB Implementers Forum (USB-IF) for the latest information. USB Specification Revision 2.0 is the most current specification at the time of publication of this document.

## 22.1 Overview of the USB Peripheral

PIC18F1XK50/PIC18LF1XK50 devices contain a full-speed and low-speed, compatible USB Serial Interface Engine (SIE) that allows fast communication between any USB host and the PIC® microcontroller. The SIE can be interfaced directly to the USB by utilizing the internal transceiver.

Some special hardware features have been included to improve performance. Dual access port memory in the device's data memory space (USB RAM) has been supplied to share direct memory access between the microcontroller core and the SIE. Buffer descriptors are also provided, allowing users to freely program endpoint memory usage within the USB RAM space. Figure 22-1 presents a general overview of the USB peripheral and its features.

**FIGURE 22-1: USB PERIPHERAL AND OPTIONS**



# PIC18F1XK50/PIC18LF1XK50

## 22.2 USB Status and Control

The operation of the USB module is configured and managed through three control registers. In addition, a total of 14 registers are used to manage the actual USB transactions. The registers are:

- USB Control register (UCON)
- USB Configuration register (UCFG)
- USB Transfer Status register (USTAT)
- USB Device Address register (UADDR)
- Frame Number registers (UFRMH:UFRML)
- Endpoint Enable registers 0 through 7 (UEPn)

### 22.2.1 USB CONTROL REGISTER (UCON)

The USB Control register (Register 22-1) contains bits needed to control the module behavior during transfers. The register contains bits that control the following:

- Main USB Peripheral Enable
- Ping-Pong Buffer Pointer Reset
- Control of the Suspend mode
- Packet Transfer Disable

In addition, the USB Control register contains a status bit, SE0 (UCON<5>), which is used to indicate the occurrence of a single-ended zero on the bus. When the USB module is enabled, this bit should be monitored to determine whether the differential data lines have come out of a single-ended zero condition. This helps to differentiate the initial power-up state from the USB Reset signal.

The overall operation of the USB module is controlled by the USBEN bit (UCON<3>). Setting this bit activates the module and resets all of the PPBI bits in the Buffer Descriptor Table to '0'. This bit also activates the internal pull-up resistors, if they are enabled. Thus, this bit can be used as a soft attach/detach to the USB. Although all Status and control bits are ignored when this bit is clear, the module needs to be fully preconfigured prior to setting this bit. This bit cannot be set until the USB module is supplied with an active clock source. If the PLL is being used, it should be enabled at least two milliseconds (enough time for the PLL to lock) before attempting to set the USBEN bit.

#### REGISTER 22-1: UCON: USB CONTROL REGISTER

U-0	R/W-0	R-x	R/C-0	R/W-0	R/W-0	R/W-0	U-0
—	PPBRST	SE0	PKTDIS	USBEN <sup>(1)</sup>	RESUME	SUSPND	—
bit 7							

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

bit 7	<b>Unimplemented:</b> Read as '0'
bit 6	<b>PPBRST:</b> Ping-Pong Buffers Reset bit 1 = Reset all Ping-Pong Buffer Pointers to the Even Buffer Descriptor (BD) banks 0 = Ping-Pong Buffer Pointers not being reset
bit 5	<b>SE0:</b> Live Single-Ended Zero Flag bit 1 = Single-ended zero active on the USB bus 0 = No single-ended zero detected
bit 4	<b>PKTDIS:</b> Packet Transfer Disable bit 1 = SIE token and packet processing disabled, automatically set when a SETUP token is received 0 = SIE token and packet processing enabled
bit 3	<b>USBEN:</b> USB Module Enable bit <sup>(1)</sup> 1 = USB module and supporting circuitry enabled (device attached) 0 = USB module and supporting circuitry disabled (device detached)
bit 2	<b>RESUME:</b> Resume Signaling Enable bit 1 = Resume signaling activated 0 = Resume signaling disabled
bit 1	<b>SUSPND:</b> Suspend USB bit 1 = USB module and supporting circuitry in Power Conserve mode, SIE clock inactive 0 = USB module and supporting circuitry in normal operation, SIE clock clocked at the configured rate
bit 0	<b>Unimplemented:</b> Read as '0'

**Note 1:** This bit cannot be set if the USB module does not have an appropriate clock source.

The PPBRST bit (UCON<6>) controls the Reset status when Double-Buffering mode (ping-pong buffering) is used. When the PPBRST bit is set, all Ping-Pong Buffer Pointers are set to the Even buffers. PPBRST has to be cleared by firmware. This bit is ignored in buffering modes not using ping-pong buffering.

The PKTDIS bit (UCON<4>) is a flag indicating that the SIE has disabled packet transmission and reception. This bit is set by the SIE when a SETUP token is received to allow setup processing. This bit cannot be set by the microcontroller, only cleared; clearing it allows the SIE to continue transmission and/or reception. Any pending events within the Buffer Descriptor Table will still be available, indicated within the USTAT register's FIFO buffer.

The RESUME bit (UCON<2>) allows the peripheral to perform a remote wake-up by executing Resume signaling. To generate a valid remote wake-up, firmware must set RESUME for 10 ms and then clear the bit. For more information on "resume signaling", see the **"Universal Serial Bus Specification Revision 2.0"**.

The SUSPND bit (UCON<1>) places the module and supporting circuitry in a Low-Power mode. The input clock to the SIE is also disabled. This bit should be set by the software in response to an IDLEIF interrupt. It should be reset by the microcontroller firmware after an ACTVIF interrupt is observed. When this bit is active, the device remains attached to the bus but the transceiver outputs remain Idle. The voltage on the VUSB pin may vary depending on the value of this bit. Setting this bit before a IDLEIF request will result in unpredictable bus behavior.

**Note:** While in Suspend mode, a typical bus-powered USB device is limited to 500  $\mu$ A of current. This is the complete current which may be drawn by the PIC device and its supporting circuitry. Care should be taken to assure minimum current draw when the device enters Suspend mode.

## 22.2.2 USB CONFIGURATION REGISTER (UCFG)

Prior to communicating over USB, the module's associated internal and/or external hardware must be configured. Most of the configuration is performed with the UCFG register (Register 22-2). The UCFG register contains most of the bits that control the system level behavior of the USB module. These include:

- Bus Speed (full speed versus low speed)
- On-Chip Pull-up Resistor Enable
- Ping-Pong Buffer Usage

The UTEYE bit, UCFG<7>, enables eye pattern generation, which aids in module testing, debugging and USB certifications.

**Note:** The USB speed, transceiver and pull-up should only be configured during the module setup phase. It is not recommended to switch these settings while the module is enabled.

### 22.2.2.1 Internal Transceiver

The USB peripheral has a built-in, USB 2.0, full-speed and low-speed capable transceiver, internally connected to the SIE. This feature is useful for low-cost, single chip applications. Enabling the USB module (USBEN = 1) will also enable the internal transceiver. The FSEN bit (UCFG<2>) controls the transceiver speed; setting the bit enables full-speed operation.

The on-chip USB pull-up resistors are controlled by the UPUEN bit (UCFG<4>). They can only be selected when the on-chip transceiver is enabled.

The internal USB transceiver obtains power from the VUSB pin. In order to meet USB signalling level specifications, VUSB must be supplied with a voltage source between 3.0V and 3.6V. The best electrical signal quality is obtained when a 3.3V supply is used and locally bypassed with a high quality ceramic capacitor. The capacitor should be placed as close as possible to the VUSB and Vss pins found on the same edge of the package (i.e., route ground of the capacitor to Vss pin 20 on 20-lead PDIP, SOIC, and SSOP packaged parts).

The D+ and D- signal lines can be routed directly to their respective pins on the USB connector or cable (for hard-wired applications). No additional resistors, capacitors, or magnetic components are required as the D+ and D- drivers have controlled slew rate and output impedance intended to match with the characteristic impedance of the USB cable.

In order to meet the USB specifications, the traces should be less than 30 cm long. Ideally, these traces should be designed to have a characteristic impedance matching that of the USB cable.

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 22-2: UCFG: USB CONFIGURATION REGISTER

R/W-0	U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
UTEYE	—	—	UPUEN <sup>(1)</sup>	—	FSEN <sup>(1)</sup>	PPB1	PPB0
bit 7					bit 0		

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7                   **UTEYE**: USB Eye Pattern Test Enable bit

1 = Eye pattern test enabled

0 = Eye pattern test disabled

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

bit 4                   **UPUEN**: USB On-Chip Pull-up Enable bit<sup>(1)</sup>

1 = On-chip pull-up enabled (pull-up on D+ with FSEN = 1 or D- with FSEN = 0)

0 = On-chip pull-up disabled

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

bit 2                   **FSEN**: Full-Speed Enable bit<sup>(1)</sup>

1 = Full-speed device: controls transceiver edge rates; requires input clock at 48 MHz

0 = Low-speed device: controls transceiver edge rates; requires input clock at 6 MHz

bit 1-0               **PPB<1:0>**: Ping-Pong Buffers Configuration bits

11 = Even/Odd ping-pong buffers enabled for Endpoints 1 to 15

10 = Even/Odd ping-pong buffers enabled for all endpoints

01 = Even/Odd ping-pong buffer enabled for OUT Endpoint 0

00 = Even/Odd ping-pong buffers disabled

**Note 1:** The UPUEN, and FSEN bits should never be changed while the USB module is enabled. These values must be preconfigured prior to enabling the module.

## 22.2.2.2 Internal Pull-up Resistors

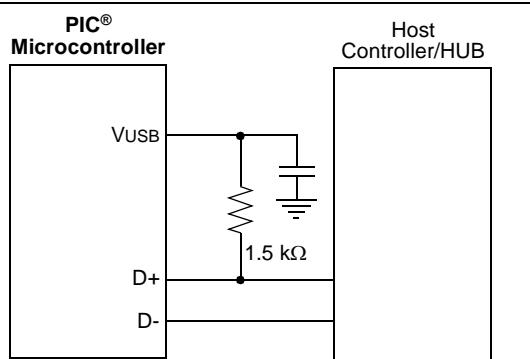
The PIC18F1XK50/PIC18LF1XK50 devices have built-in pull-up resistors designed to meet the requirements for low-speed and full-speed USB. The UPUEN bit (UCFG<4>) enables the internal pull-ups. Figure 22-1 shows the pull-ups and their control.

**Note:** The official USB specifications require that USB devices must never source any current onto the +5V VBUS line of the USB cable. Additionally, USB devices must never source any current on the D+ and D- data lines whenever the +5V VBUS line is less than 1.17V. In order to meet this requirement, applications which are not purely bus powered should monitor the VBUS line and avoid turning on the USB module and the D+ or D- pull-up resistor until VBUS is greater than 1.17V. VBUS can be connected to and monitored by any 5V tolerant I/O pin for this purpose.

## 22.2.2.3 External Pull-up Resistors

External pull-up may also be used. The VUSB pin may be used to pull up D+ or D-. The pull-up resistor must be 1.5 k $\Omega$  ( $\pm 5\%$ ) as required by the USB specifications. Figure 22-2 shows an example.

**FIGURE 22-2: EXTERNAL CIRCUITRY**



**Note:** The above setting shows a typical connection for a full-speed configuration using an on-chip regulator and an external pull-up resistor.

## 22.2.2.4 Ping-Pong Buffer Configuration

The usage of ping-pong buffers is configured using the PPB<1:0> bits. Refer to **Section 22.4.4 “Ping-Pong Buffering”** for a complete explanation of the ping-pong buffers.

## 22.2.2.5 Eye Pattern Test Enable

An automatic eye pattern test can be generated by the module when the UCFG<7> bit is set. The eye pattern output will be observable based on module settings, meaning that the user is first responsible for configuring the SIE clock settings, pull-up resistor and Transceiver mode. In addition, the module has to be enabled.

Once UTEYE is set, the module emulates a switch from a receive to transmit state and will start transmitting a J-K-J-K bit sequence (K-J-K-J for full speed). The sequence will be repeated indefinitely while the Eye Pattern Test mode is enabled.

Note that this bit should never be set while the module is connected to an actual USB system. This Test mode is intended for board verification to aid with USB certification tests. It is intended to show a system developer the noise integrity of the USB signals which can be affected by board traces, impedance mismatches and proximity to other system components. It does not properly test the transition from a receive to a transmit state. Although the eye pattern is not meant to replace the more complex USB certification test, it should aid during first order system debugging.

# PIC18F1XK50/PIC18LF1XK50

## 22.2.3 USB STATUS REGISTER (USTAT)

The USB Status register reports the transaction status within the SIE. When the SIE issues a USB transfer complete interrupt, USTAT should be read to determine the status of the transfer. USTAT contains the transfer endpoint number, direction and Ping-Pong Buffer Pointer value (if used).

**Note:** The data in the USB Status register is valid two SIE clocks after the TRNIF interrupt flag is asserted.

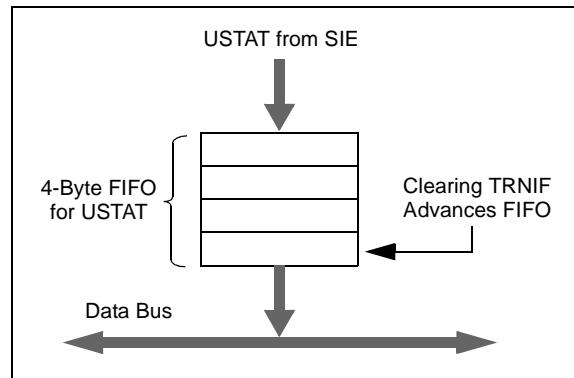
In low-speed operation with the system clock operating at 48 MHz, a delay may be required between receiving the TRNIF interrupt and processing the data in the USTAT register.

The USTAT register is actually a read window into a four-byte status FIFO, maintained by the SIE. It allows the microcontroller to process one transfer while the SIE processes additional endpoints (Figure 22-3). When the SIE completes using a buffer for reading or writing data, it updates the USTAT register. If another USB transfer is performed before a transaction complete interrupt is serviced, the SIE will store the status of the next transfer into the status FIFO.

Clearing the transfer complete flag bit, TRNIF, causes the SIE to advance the FIFO. If the next data in the FIFO holding register is valid, the SIE will reassert the interrupt within 6 TCY of clearing TRNIF. If no additional data is present, TRNIF will remain clear; USTAT data will no longer be reliable.

**Note:** If an endpoint request is received while the USTAT FIFO is full, the SIE will automatically issue a NAK back to the host.

**FIGURE 22-3: USTAT FIFO**



## REGISTER 22-3: USTAT: USB STATUS REGISTER

U-0	U-0	R-x	R-x	R-x	R-x	R-x	U-0
—	—	ENDP2	ENDP1	ENDP0	DIR	PPBI <sup>(1)</sup>	—
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

bit 5-3      **ENDP<2:0>:** Encoded Number of Last Endpoint Activity bits  
(represents the number of the BDT updated by the last USB transfer)

111 = Endpoint 7

110 = Endpoint 6

....

001 = Endpoint 1

000 = Endpoint 0

bit 2      **DIR:** Last BD Direction Indicator bit

1 = The last transaction was an IN token

0 = The last transaction was an OUT or SETUP token

bit 1      **PPBI:** Ping-Pong BD Pointer Indicator bit<sup>(1)</sup>

1 = The last transaction was to the Odd BD bank

0 = The last transaction was to the Even BD bank

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

**Note 1:** This bit is only valid for endpoints with available Even and Odd BD registers.

## 22.2.4 USB ENDPOINT CONTROL

Each of the 8 possible bidirectional endpoints has its own independent control register, UEPn (where 'n' represents the endpoint number). Each register has an identical complement of control bits. The prototype is shown in Register 22-4.

The EPHSHK bit (UEPn<4>) controls handshaking for the endpoint; setting this bit enables USB handshaking. Typically, this bit is always set except when using isochronous endpoints.

The EPCONDIS bit (UEPn<3>) is used to enable or disable USB control operations (SETUP) through the endpoint. Clearing this bit enables SETUP transactions. Note that the corresponding EPINEN and EPOUTEN bits must be set to enable IN and OUT

transactions. For Endpoint 0, this bit should always be cleared since the USB specifications identify Endpoint 0 as the default control endpoint.

The EPOUTEN bit (UEPn<2>) is used to enable or disable USB OUT transactions from the host. Setting this bit enables OUT transactions. Similarly, the EPINEN bit (UEPn<1>) enables or disables USB IN transactions from the host.

The EPSTALL bit (UEPn<0>) is used to indicate a STALL condition for the endpoint. If a STALL is issued on a particular endpoint, the EPSTALL bit for that endpoint pair will be set by the SIE. This bit remains set until it is cleared through firmware, or until the SIE is reset.

## REGISTER 22-4: UEPn: USB ENDPOINT n CONTROL REGISTER (UEP0 THROUGH UEP7)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL <sup>(1)</sup>
bit 7							
bit 0							

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>EPHSHK:</b> Endpoint Handshake Enable bit 1 = Endpoint handshake enabled 0 = Endpoint handshake disabled (typically used for isochronous endpoints)
bit 3	<b>EPCONDIS:</b> Bidirectional Endpoint Control bit <b>If EPOUTEN = 1 and EPINEN = 1:</b> 1 = Disable Endpoint n from control transfers; only IN and OUT transfers allowed 0 = Enable Endpoint n for control (SETUP) transfers; IN and OUT transfers also allowed
bit 2	<b>EPOUTEN:</b> Endpoint Output Enable bit 1 = Endpoint n output enabled 0 = Endpoint n output disabled
bit 1	<b>EPINEN:</b> Endpoint Input Enable bit 1 = Endpoint n input enabled 0 = Endpoint n input disabled
bit 0	<b>EPSTALL:</b> Endpoint STALL Enable bit <sup>(1)</sup> 1 = Endpoint n is stalled 0 = Endpoint n is not stalled

**Note 1:** Valid only if Endpoint n is enabled; otherwise, the bit is ignored.

# PIC18F1XK50/PIC18LF1XK50

## 22.2.5 USB ADDRESS REGISTER (UADDR)

The USB Address register contains the unique USB address that the peripheral will decode when active. UADDR is reset to 00h when a USB Reset is received, indicated by URSTIF, or when a Reset is received from the microcontroller. The USB address must be written by the microcontroller during the USB setup phase (enumeration) as part of the Microchip USB firmware support.

## 22.2.6 USB FRAME NUMBER REGISTERS (UFRMH:UFRML)

The Frame Number registers contain the 11-bit frame number. The low-order byte is contained in UFRML, while the three high-order bits are contained in UFRMH. The register pair is updated with the current frame number whenever a SOF token is received. For the microcontroller, these registers are read-only. The Frame Number registers are primarily used for isochronous transfers. The contents of the UFRMH and UFRML registers are only valid when the 48 MHz SIE clock is active (i.e., contents are inaccurate when SUSPND (UCON<1>) bit = 1).

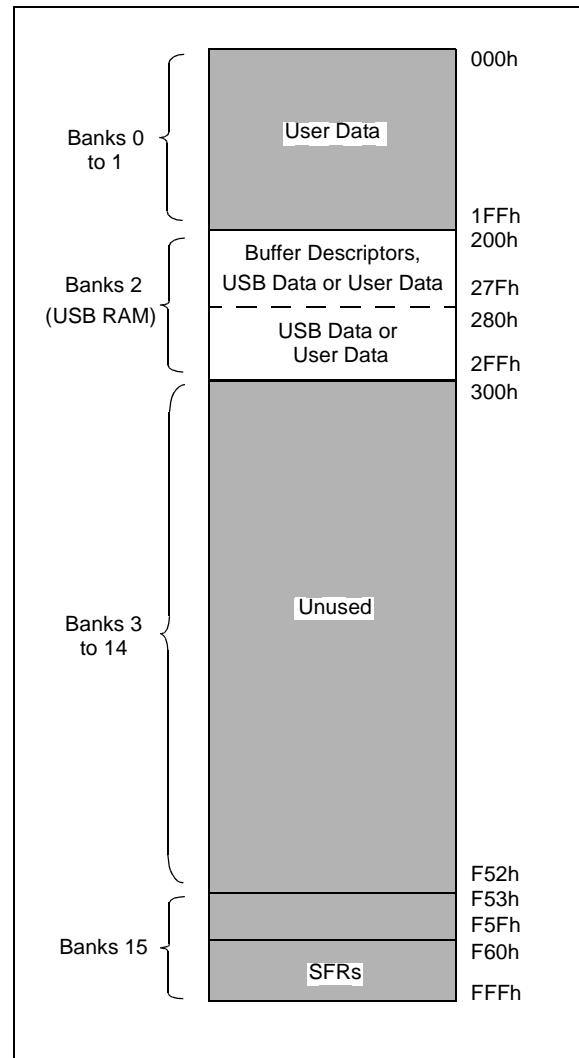
## 22.3 USB RAM

USB data moves between the microcontroller core and the SIE through a memory space known as the USB RAM. This is a special dual access memory that is mapped into the normal data memory space in Bank 2 (200h to 2FFh) for a total of 256 bytes (Figure 22-4).

Bank 2 (200h through 27Fh) is used specifically for endpoint buffer control. Depending on the type of buffering being used, all but 8 bytes of Bank 2 may also be available for use as USB buffer space.

Although USB RAM is available to the microcontroller as data memory, the sections that are being accessed by the SIE should not be accessed by the microcontroller. A semaphore mechanism is used to determine the access to a particular buffer at any given time. This is discussed in **Section 22.4.1.1 “Buffer Ownership”**.

**FIGURE 22-4: IMPLEMENTATION OF USB RAM IN DATA MEMORY SPACE**



## 22.4 Buffer Descriptors and the Buffer Descriptor Table

The registers in Bank 2 are used specifically for endpoint buffer control in a structure known as the Buffer Descriptor Table (BDT). This provides a flexible method for users to construct and control endpoint buffers of various lengths and configuration.

The BDT is composed of Buffer Descriptors (BD) which are used to define and control the actual buffers in the USB RAM space. Each BD, in turn, consists of four registers, where n represents one of the 32 possible BDs (range of 0 to 31):

- BDnSTAT: BD Status register
- BDnCNT: BD Byte Count register
- BDnADRL: BD Address Low register
- BDnADRH: BD Address High register

BDs always occur as a four-byte block in the sequence, BDnSTAT:BDnCNT:BDnADRL:BDnADR. The address of BDnSTAT is always an offset of  $(4n - 1)$  (in hexadecimal) from 200h, with n being the buffer descriptor number.

Depending on the buffering configuration used (**Section 22.4.4 “Ping-Pong Buffering”**), there are up to 16, 17 or 32 sets of buffer descriptors. At a minimum, the BDT must be at least 8 bytes long. This is because the USB specification mandates that every device must have Endpoint 0 with both input and output for initial setup. Depending on the endpoint and buffering configuration, the BDT can be as long as 128 bytes.

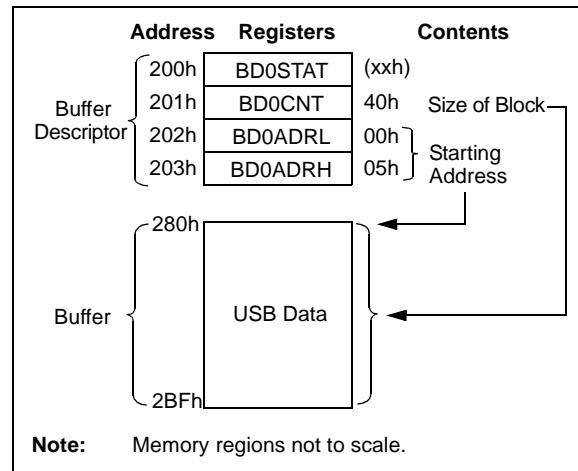
Although they can be thought of as Special Function Registers, the Buffer Descriptor Status and Address registers are not hardware mapped, as conventional microcontroller SFRs in Bank 15 are. If the endpoint corresponding to a particular BD is not enabled, its registers are not used. Instead of appearing as unimplemented addresses, however, they appear as available RAM. Only when an endpoint is enabled by setting the UEPn<1> bit does the memory at those addresses become functional as BD registers. As with any address in the data memory space, the BD registers have an indeterminate value on any device Reset.

An example of a BD for a 64-byte buffer, starting at 280h, is shown in Figure 22-5. A particular set of BD registers is only valid if the corresponding endpoint has been enabled using the UEPn register. All BD registers are available in USB RAM. The BD for each endpoint should be set up prior to enabling the endpoint.

### 22.4.1 BD STATUS AND CONFIGURATION

Buffer descriptors not only define the size of an endpoint buffer, but also determine its configuration and control. Most of the configuration is done with the BD Status register, BDnSTAT. Each BD has its own unique and correspondingly numbered BDnSTAT register.

**FIGURE 22-5: EXAMPLE OF A BUFFER DESCRIPTOR**



Unlike other control registers, the bit configuration for the BDnSTAT register is context sensitive. There are two distinct configurations, depending on whether the microcontroller or the USB module is modifying the BD and buffer at a particular time. Only three bit definitions are shared between the two.

#### 22.4.1.1 Buffer Ownership

Because the buffers and their BDs are shared between the CPU and the USB module, a simple semaphore mechanism is used to distinguish which is allowed to update the BD and associated buffers in memory.

This is done by using the UOWN bit (BDnSTAT<7>) as a semaphore to distinguish which is allowed to update the BD and associated buffers in memory. UOWN is the only bit that is shared between the two configurations of BDnSTAT.

When UOWN is clear, the BD entry is “owned” by the microcontroller core. When the UOWN bit is set, the BD entry and the buffer memory are “owned” by the USB peripheral. The core should not modify the BD or its corresponding data buffer during this time. Note that the microcontroller core can still read BDnSTAT while the SIE owns the buffer and vice versa.

The buffer descriptors have a different meaning based on the source of the register update. Prior to placing ownership with the USB peripheral, the user can configure the basic operation of the peripheral through the BDnSTAT bits. During this time, the byte count and buffer location registers can also be set.

When UOWN is set, the user can no longer depend on the values that were written to the BDs. From this point, the SIE updates the BDs as necessary, overwriting the original BD values. The BDnSTAT register is updated by the SIE with the token PID and the transfer count, BDnCNT, is updated.

# PIC18F1XK50/PIC18LF1XK50

---

The BDnSTAT byte of the BDT should always be the last byte updated when preparing to arm an endpoint. The SIE will clear the UOWN bit when a transaction has completed.

No hardware mechanism exists to block access when the UOWN bit is set. Thus, unexpected behavior can occur if the microcontroller attempts to modify memory when the SIE owns it. Similarly, reading such memory may produce inaccurate data until the USB peripheral returns ownership to the microcontroller.

## 22.4.1.2 BDnSTAT Register (CPU Mode)

When UOWN = 0, the microcontroller core owns the BD. At this point, the other seven bits of the register take on control functions.

The Data Toggle Sync Enable bit, DTSEN (BDnSTAT<3>), controls data toggle parity checking. Setting DTSEN enables data toggle synchronization by the SIE. When enabled, it checks the data packet's parity against the value of DTS (BDnSTAT<6>). If a packet arrives with an incorrect synchronization, the data will essentially be ignored. It will not be written to the USB RAM and the USB transfer complete interrupt flag will not be set. The SIE will send an ACK token back to the host to Acknowledge receipt, however. The effects of the DTSEN bit on the SIE are summarized in Table 22-1.

The Buffer Stall bit, BSTALL (BDnSTAT<2>), provides support for control transfers, usually one-time stalls on Endpoint 0. It also provides support for the SET\_FEATURE/CLEAR\_FEATURE commands specified in Chapter 9 of the USB specification; typically, continuous STALLs to any endpoint other than the default control endpoint.

The BSTALL bit enables buffer stalls. Setting BSTALL causes the SIE to return a STALL token to the host if a received token would use the BD in that location. The EPSTALL bit in the corresponding UEPn control register is set and a STALL interrupt is generated when a STALL is issued to the host. The UOWN bit remains set and the BDs are not changed unless a SETUP token is received. In this case, the STALL condition is cleared and the ownership of the BD is returned to the microcontroller core.

The BD<9:8> bits (BDnSTAT<1:0>) store the two Most Significant digits of the SIE byte count; the lower 8 digits are stored in the corresponding BDnCNT register. See **Section 22.4.2 “BD Byte Count”** for more information.

**TABLE 22-1: EFFECT OF DTSEN BIT ON ODD/EVEN (DATA0/DATA1) PACKET RECEPTION**

OUT Packet from Host	BDnSTAT Settings		Device Response after Receiving Packet			
	DTSEN	DTS	Handshake	UOWN	TRNIF	BDnSTAT and USTAT Status
DATA0	1	0	ACK	0	1	Updated
DATA1	1	0	ACK	1	0	Not Updated
DATA0	1	1	ACK	1	0	Not Updated
DATA1	1	1	ACK	0	1	Updated
Either	0	x	ACK	0	1	Updated
Either, with error	x	x	NAK	1	0	Not Updated

**Legend:** x = don't care

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 22-5: BD<sub>n</sub>STAT: BUFFER DESCRIPTOR <sub>n</sub> STATUS REGISTER (BD0STAT THROUGH BD31STAT), CPU MODE (DATA IS WRITTEN TO THE SIDE)

R/W-x	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
UOWN <sup>(1)</sup>	DTS <sup>(2)</sup>	— <sup>(3)</sup>	— <sup>(3)</sup>	DTSEN	BSTALL	BC9	BC8
bit 7	bit 0						

### Legend:

R = Readable bit  
-n = Value at POR

W = Writable bit  
'1' = Bit is set

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

bit 7	<b>UOWN:</b> USB Own bit <sup>(1)</sup> 0 = The microcontroller core owns the BD and its corresponding buffer
bit 6	<b>DTS:</b> Data Toggle Synchronization bit <sup>(2)</sup> 1 = Data 1 packet 0 = Data 0 packet
bit 5-4	<b>Unimplemented:</b> These bits should always be programmed to '0' <sup>(3)</sup> .
bit 3	<b>DTSEN:</b> Data Toggle Synchronization Enable bit 1 = Data toggle synchronization is enabled; data packets with incorrect Sync value will be ignored except for a SETUP transaction, which is accepted even if the data toggle bits do not match 0 = No data toggle synchronization is performed
bit 2	<b>BSTALL:</b> Buffer Stall Enable bit 1 = Buffer stall enabled; STALL handshake issued if a token is received that would use the BD in the given location (UOWN bit remains set, BD value is unchanged) 0 = Buffer stall disabled
bit 1-0	<b>BC&lt;9:8&gt;:</b> Byte Count 9 and 8 bits The byte count bits represent the number of bytes that will be transmitted for an IN token or received during an OUT token. Together with BC<7:0>, the valid byte counts are 0-1023.

- Note 1:** This bit must be initialized by the user to the desired value prior to enabling the USB module.
- 2:** This bit is ignored unless DTSEN = 1.
- 3:** If these bits are set, USB communication may not work. Hence, these bits should always be maintained as '0'.

# PIC18F1XK50/PIC18LF1XK50

## 22.4.1.3 BDnSTAT Register (SIE Mode)

When the BD and its buffer are owned by the SIE, most of the bits in BDnSTAT take on a different meaning. The configuration is shown in Register 22-6. Once the UOWN bit is set, any data or control settings previously written there by the user will be overwritten with data from the SIE.

The BDnSTAT register is updated by the SIE with the token Packet Identifier (PID) which is stored in BDnSTAT<5:3>. The transfer count in the corresponding BDnCNT register is updated. Values that overflow the 8-bit register carry over to the two Most Significant digits of the count, stored in BDnSTAT<1:0>.

## 22.4.2 BD BYTE COUNT

The byte count represents the total number of bytes that will be transmitted during an IN transfer. After an IN transfer, the SIE will return the number of bytes sent to the host.

For an OUT transfer, the byte count represents the maximum number of bytes that can be received and stored in USB RAM. After an OUT transfer, the SIE will return the actual number of bytes received. If the number of bytes received exceeds the corresponding byte count, the data packet will be rejected and a NAK handshake will be generated. When this happens, the byte count will not be updated.

The 10-bit byte count is distributed over two registers. The lower 8 bits of the count reside in the BDnCNT register. The upper two bits reside in BDnSTAT<1:0>. This represents a valid byte range of 0 to 1023.

## 22.4.3 BD ADDRESS VALIDATION

The BD Address register pair contains the starting RAM address location for the corresponding endpoint buffer. No mechanism is available in hardware to validate the BD address.

If the value of the BD address does not point to an address in the USB RAM, or if it points to an address within another endpoint's buffer, data is likely to be lost or overwritten. Similarly, overlapping a receive buffer (OUT endpoint) with a BD location in use can yield unexpected results. When developing USB applications, the user may want to consider the inclusion of software-based address validation in their code.

## REGISTER 22-6: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD31STAT), SIE MODE (DATA RETURNED BY THE SIE TO THE MCU)

R/W-x	U-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN	—	PID3	PID2	PID1	PID0	BC9	BC8
bit 7							bit 0

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7

**UOWN:** USB Own bit

1 = The SIE owns the BD and its corresponding buffer

bit 6

**Reserved:** Not written by the SIE

bit 5-2

**PID<3:0>:** Packet Identifier bits

The received token PID value of the last transfer (IN, OUT or SETUP transactions only).

bit 1-0

**BC<9:8>:** Byte Count 9 and 8 bits

These bits are updated by the SIE to reflect the actual number of bytes received on an OUT transfer and the actual number of bytes transmitted on an IN transfer.

## 22.4.4 PING-PONG BUFFERING

An endpoint is defined to have a ping-pong buffer when it has two sets of BD entries: one set for an Even transfer and one set for an Odd transfer. This allows the CPU to process one BD while the SIE is processing the other BD. Double-buffering BDs in this way allows for maximum throughput to/from the USB.

The USB module supports four modes of operation:

- No ping-pong support
- Ping-pong buffer support for OUT Endpoint 0 only
- Ping-pong buffer support for all endpoints
- Ping-pong buffer support for all other Endpoints except Endpoint 0

The ping-pong buffer settings are configured using the PPB<1:0> bits in the UCFG register.

The USB module keeps track of the Ping-Pong Pointer individually for each endpoint. All pointers are initially reset to the Even BD when the module is enabled. After

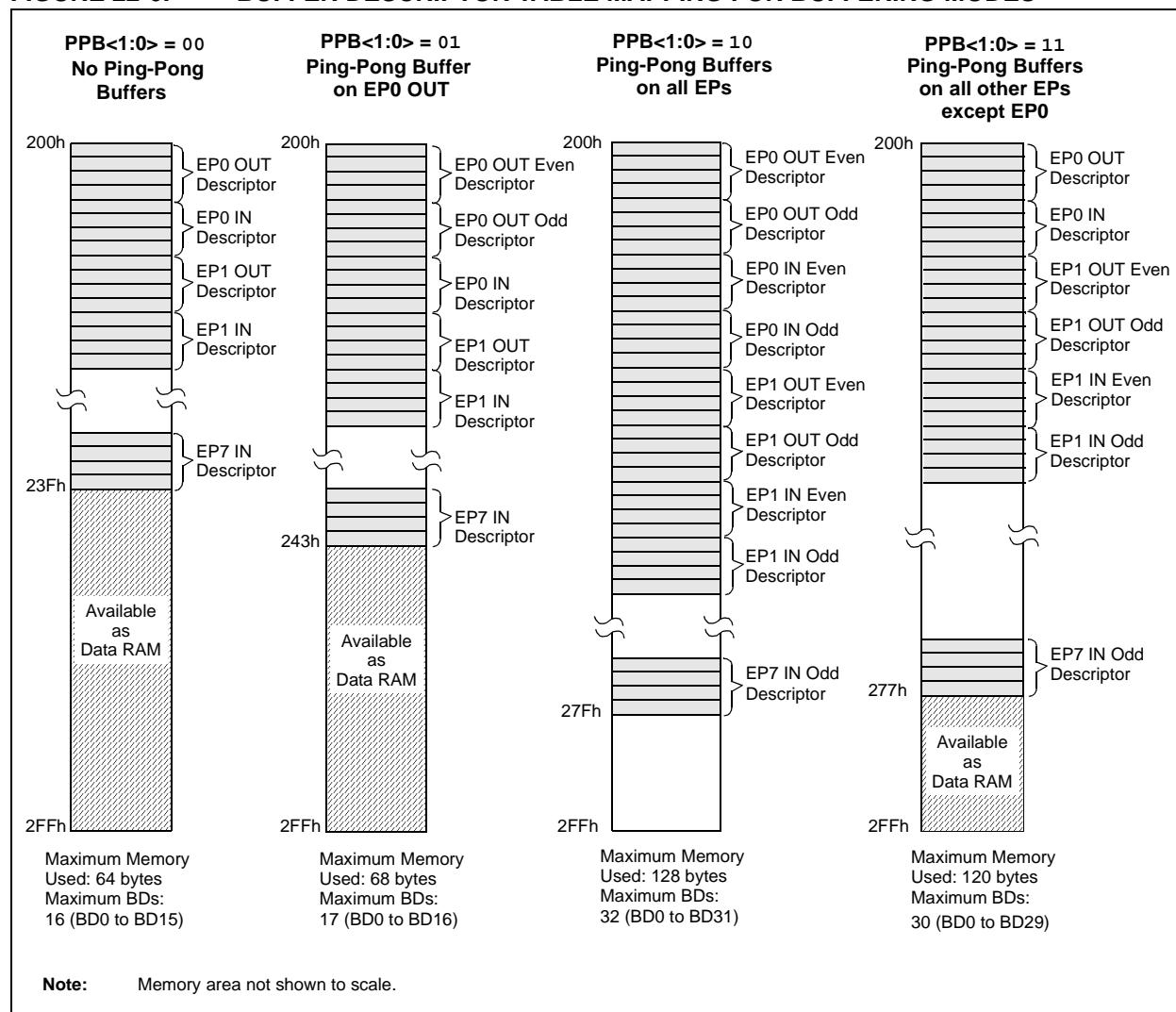
the completion of a transaction (UOWN cleared by the SIE), the pointer is toggled to the Odd BD. After the completion of the next transaction, the pointer is toggled back to the Even BD and so on.

The Even/Odd status of the last transaction is stored in the PPBI bit of the USTAT register. The user can reset all Ping-Pong Pointers to Even using the PPBRST bit.

Figure 22-6 shows the four different modes of operation and how USB RAM is filled with the BDs.

BDs have a fixed relationship to a particular endpoint, depending on the buffering configuration. The mapping of BDs to endpoints is detailed in Table 22-2. This relationship also means that gaps may occur in the BDT if endpoints are not enabled contiguously. This theoretically means that the BDs for disabled endpoints could be used as buffer space. In practice, users should avoid using such spaces in the BDT unless a method of validating BD addresses is implemented.

**FIGURE 22-6: BUFFER DESCRIPTOR TABLE MAPPING FOR BUFFERING MODES**



# PIC18F1XK50/PIC18LF1XK50

**TABLE 22-2: ASSIGNMENT OF BUFFER DESCRIPTORS FOR THE DIFFERENT BUFFERING MODES**

Endpoint	BDs Assigned to Endpoint							
	Mode 0 (No Ping-Pong)		Mode 1 (Ping-Pong on EP0 OUT)		Mode 2 (Ping-Pong on all EPs)		Mode 3 (Ping-Pong on all other EPs, except EP0)	
	Out	In	Out	In	Out	In	Out	In
0	0	1	0 (E), 1 (O)	2	0 (E), 1 (O)	2 (E), 3 (O)	0	1
1	2	3	3	4	4 (E), 5 (O)	6 (E), 7 (O)	2 (E), 3 (O)	4 (E), 5 (O)
2	4	5	5	6	8 (E), 9 (O)	10 (E), 11 (O)	6 (E), 7 (O)	8 (E), 9 (O)
3	6	7	7	8	12 (E), 13 (O)	14 (E), 15 (O)	10 (E), 11 (O)	12 (E), 13 (O)
4	8	9	9	10	16 (E), 17 (O)	18 (E), 19 (O)	14 (E), 15 (O)	16 (E), 17 (O)
5	10	11	11	12	20 (E), 21 (O)	22 (E), 23 (O)	18 (E), 19 (O)	20 (E), 21 (O)
6	12	13	13	14	24 (E), 25 (O)	26 (E), 27 (O)	22 (E), 23 (O)	24 (E), 25 (O)
7	14	15	15	16	28 (E), 29 (O)	30 (E), 31 (O)	26 (E), 27 (O)	28 (E), 29 (O)

**Legend:** (E) = Even transaction buffer, (O) = Odd transaction buffer

**TABLE 22-3: SUMMARY OF USB BUFFER DESCRIPTOR TABLE REGISTERS**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BDnSTAT <sup>(1)</sup>	UOWN	DTS <sup>(4)</sup>	PID3 <sup>(2)</sup>	PID2 <sup>(2)</sup>	PID1 <sup>(2)</sup> DTSEN <sup>(3)</sup>	PID0 <sup>(2)</sup> BSTALL <sup>(3)</sup>	BC9	BC8
BDnCNT <sup>(1)</sup>	Byte Count							
BDnADRL <sup>(1)</sup>	Buffer Address Low							
BDnADRH <sup>(1)</sup>	Buffer Address High							

- Note 1:** For buffer descriptor registers, n may have a value of 0 to 31. For the sake of brevity, all 32 registers are shown as one generic prototype. All registers have indeterminate Reset values (xxxx xxxx).
- 2:** Bits 5 through 2 of the BDnSTAT register are used by the SIE to return PID<3:0> values once the register is turned over to the SIE (UOWN bit is set). Once the registers have been under SIE control, the values written for DTSEN and BSTALL are no longer valid.
- 3:** Prior to turning the buffer descriptor over to the SIE (UOWN bit is cleared), bits 5 through 2 of the BDnSTAT register are used to configure the DTSEN and BSTALL settings.
- 4:** This bit is ignored unless DTSEN = 1.

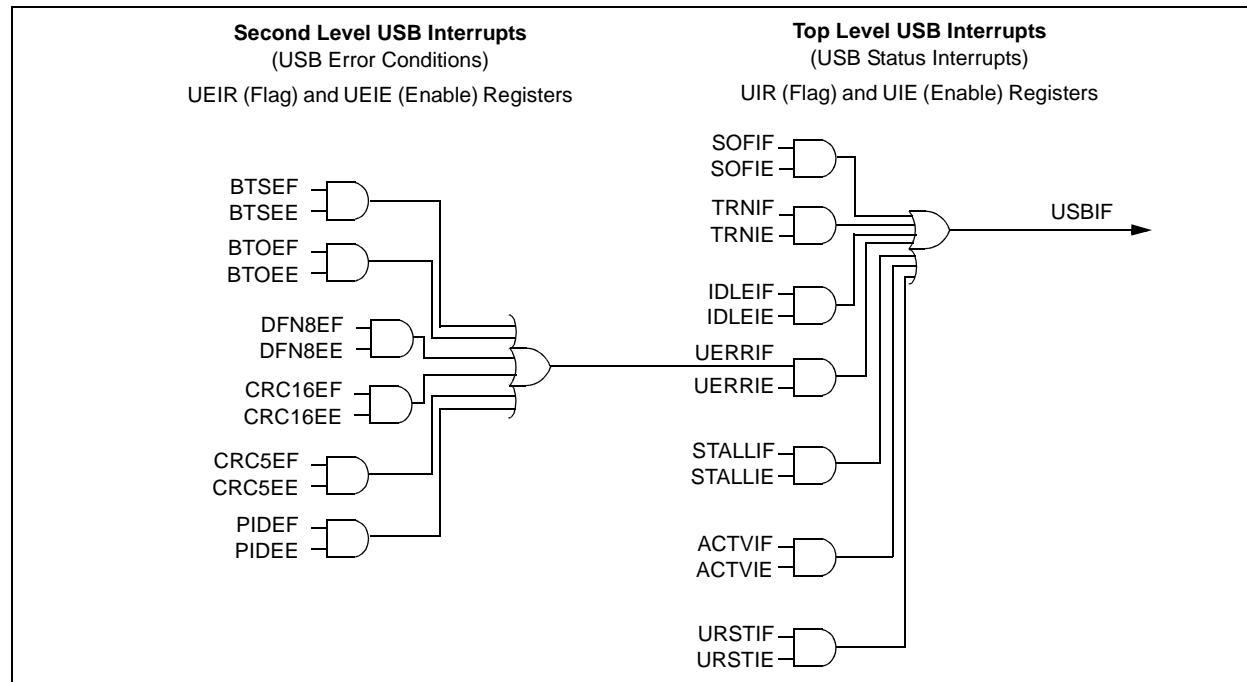
## 22.5 USB Interrupts

The USB module can generate multiple interrupt conditions. To accommodate all of these interrupt sources, the module is provided with its own interrupt logic structure, similar to that of the microcontroller. USB interrupts are enabled with one set of control registers and trapped with a separate set of flag registers. All sources are funneled into a single USB interrupt request, USBIF (PIR2<2>), in the microcontroller's interrupt logic.

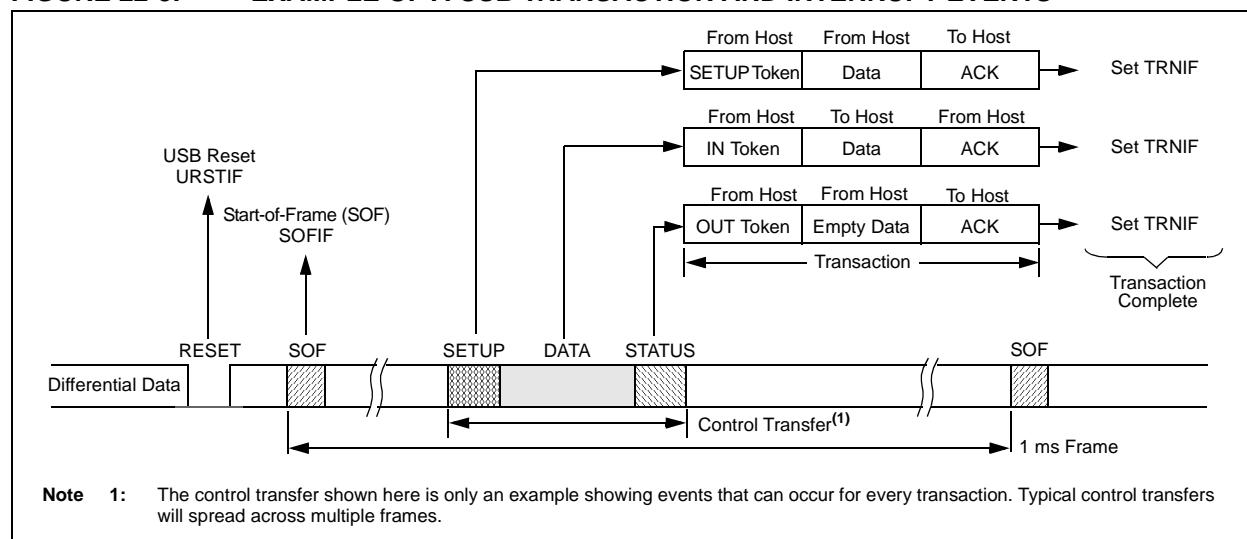
Figure 22-7 shows the interrupt logic for the USB module. There are two layers of interrupt registers in the USB module. The top level consists of overall USB Status interrupts; these are enabled and flagged in the UIR and UIR registers, respectively. The second level consists of USB error conditions, which are enabled and flagged in the UEIR and UEIE registers. An interrupt condition in any of these triggers a USB Error Interrupt Flag (UERRIF) in the top level.

Interrupts may be used to trap routine events in a USB transaction. Figure 22-8 shows some common events within a USB frame and their corresponding interrupts.

**FIGURE 22-7: USB INTERRUPT LOGIC FUNNEL**



**FIGURE 22-8: EXAMPLE OF A USB TRANSACTION AND INTERRUPT EVENTS**



# **PIC18F1XK50/PIC18LF1XK50**

### 22.5.1 USB INTERRUPT STATUS REGISTER (UIR)

The USB Interrupt Status register (Register 22-7) contains the flag bits for each of the USB Status interrupt sources. Each of these sources has a corresponding interrupt enable bit in the UIE register. All of the USB status flags are ORed together to generate the USBIF interrupt flag for the microcontroller's interrupt funnel.

Once an interrupt bit has been set by the SIE, it must be cleared by software by writing a '0'. The flag bits can also be set in software which can aid in firmware debugging.

## REGISTER 22-7: UIR: USB INTERRUPT STATUS REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0
—	SOFIF	STALLIF	IDLEIF <sup>(1)</sup>	TRNIF <sup>(2)</sup>	ACTVIF <sup>(3)</sup>	UERRIF <sup>(4)</sup>	URSTIF
bit 7						bit 0	

**Legend:**

R = Readable bit  
-n = Value at POF

W = Writable bit  
'1' = Bit is set

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

x = Bit is unknown

bit 7	<b>Unimplemented:</b> Read as '0'
bit 6	<b>SOFIF:</b> Start-of-Frame Token Interrupt bit 1 = A Start-of-Frame token received by the SIE 0 = No Start-of-Frame token received by the SIE
bit 5	<b>STALLIF:</b> A STALL Handshake Interrupt bit 1 = A STALL handshake was sent by the SIE 0 = A STALL handshake has not been sent
bit 4	<b>IDLEIF:</b> Idle Detect Interrupt bit <sup>(1)</sup> 1 = Idle condition detected (constant Idle state of 3 ms or more) 0 = No Idle condition detected
bit 3	<b>TRNIF:</b> Transaction Complete Interrupt bit <sup>(2)</sup> 1 = Processing of pending transaction is complete; read USTAT register for endpoint information 0 = Processing of pending transaction is not complete or no transaction is pending
bit 2	<b>ACTVIF:</b> Bus Activity Detect Interrupt bit <sup>(3)</sup> 1 = Activity on the D+/D- lines was detected 0 = No activity detected on the D+/D- lines
bit 1	<b>UERRIF:</b> USB Error Condition Interrupt bit <sup>(4)</sup> 1 = An unmasked error condition has occurred 0 = No unmasked error condition has occurred.
bit 0	<b>URSTIF:</b> USB Reset Interrupt bit 1 = Valid USB Reset occurred; 00h is loaded into UADDR register 0 = No USB Reset has occurred

- Note 1:** Once an Idle state is detected, the user may want to place the USB module in Suspend mode.

**2:** Clearing this bit will cause the USTAT FIFO to advance (valid only for IN, OUT and SETUP tokens).

**3:** This bit is typically unmasked only following the detection of a UIDLE interrupt event.

**4:** Only error conditions enabled through the UEIE register will set this bit. This bit is a status bit only and cannot be set or cleared by the user.

## 22.5.1.1 Bus Activity Detect Interrupt Bit (ACTVIF)

The ACTVIF bit cannot be cleared immediately after the USB module wakes up from Suspend or while the USB module is suspended. A few clock cycles are required to synchronize the internal hardware state machine before the ACTVIF bit can be cleared by firmware. Clearing the ACTVIF bit before the internal hardware is synchronized may not have an effect on the value of ACTVIF. Additionally, if the USB module uses the clock from the 48 MHz PLL source, then after

clearing the SUSPND bit, the USB module may not be immediately operational while waiting for the 48 MHz PLL to lock. The application code should clear the ACTVIF flag as shown in Example 22-1.

Only one ACTVIF interrupt is generated when resuming from the USB bus Idle condition. If user firmware clears the ACTVIF bit, the bit will not immediately become set again, even when there is continuous bus traffic. Bus traffic must cease long enough to generate another IDLEIF condition before another ACTVIF interrupt can be generated.

## EXAMPLE 22-1: CLEARING ACTVIF BIT (UIR<2>)

### Assembly:

```
BCF      UCON, SUSPND
LOOP:
  BTFSS   UIR, ACTVIF
  BRA     DONE
  BCF      UIR, ACTVIF
  BRA     LOOP
DONE:
```

### C:

```
UCONbits.SUSPND = 0;
while (UIRbits.ACTVIF) { UIRbits.ACTVIF = 0; }
```

# PIC18F1XK50/PIC18LF1XK50

## 22.5.2 USB INTERRUPT ENABLE REGISTER (UIE)

The USB Interrupt Enable register (Register 22-8) contains the enable bits for the USB Status interrupt sources. Setting any of these bits will enable the respective interrupt source in the UIR register.

The values in this register only affect the propagation of an interrupt condition to the microcontroller's interrupt logic. The flag bits are still set by their interrupt conditions, allowing them to be polled and serviced without actually generating an interrupt.

### REGISTER 22-8: UIE: USB INTERRUPT ENABLE REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE
bit 7							
bit 0							

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

bit 6                    **SOFIE:** Start-of-Frame Token Interrupt Enable bit

1 = Start-of-Frame token interrupt enabled  
0 = Start-of-Frame token interrupt disabled

bit 5                    **STALLIE:** STALL Handshake Interrupt Enable bit

1 = STALL interrupt enabled  
0 = STALL interrupt disabled

bit 4                    **IDLEIE:** Idle Detect Interrupt Enable bit

1 = Idle detect interrupt enabled  
0 = Idle detect interrupt disabled

bit 3                    **TRNIE:** Transaction Complete Interrupt Enable bit

1 = Transaction interrupt enabled  
0 = Transaction interrupt disabled

bit 2                    **ACTVIE:** Bus Activity Detect Interrupt Enable bit

1 = Bus activity detect interrupt enabled  
0 = Bus activity detect interrupt disabled

bit 1                    **UERRIE:** USB Error Interrupt Enable bit

1 = USB error interrupt enabled  
0 = USB error interrupt disabled

bit 0                    **URSTIE:** USB Reset Interrupt Enable bit

1 = USB Reset interrupt enabled  
0 = USB Reset interrupt disabled

### 22.5.3 USB ERROR INTERRUPT STATUS REGISTER (UEIR)

The USB Error Interrupt Status register (Register 22-9) contains the flag bits for each of the error sources within the USB peripheral. Each of these sources is controlled by a corresponding interrupt enable bit in the UEIE register. All of the USB error flags are ORed together to generate the USB Error Interrupt Flag (UERRIF) at the top level of the interrupt logic.

Each error bit is set as soon as the error condition is detected. Thus, the interrupt will typically not correspond with the end of a token being processed.

Once an interrupt bit has been set by the SIE, it must be cleared by software by writing a '0'.

### REGISTER 22-9: UEIR: USB ERROR INTERRUPT STATUS REGISTER

R/C-0	U-0	U-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF
bit 7	bit 0						

#### Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>BTSEF:</b> Bit Stuff Error Flag bit 1 = A bit stuff error has been detected 0 = No bit stuff error
bit 6-5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>BTOEF:</b> Bus Turnaround Time-out Error Flag bit 1 = Bus turnaround time-out has occurred (more than 16 bit times of Idle from previous EOP elapsed) 0 = No bus turnaround time-out
bit 3	<b>DFN8EF:</b> Data Field Size Error Flag bit 1 = The data field was not an integral number of bytes 0 = The data field was an integral number of bytes
bit 2	<b>CRC16EF:</b> CRC16 Failure Flag bit 1 = The CRC16 failed 0 = The CRC16 passed
bit 1	<b>CRC5EF:</b> CRC5 Host Error Flag bit 1 = The token packet was rejected due to a CRC5 error 0 = The token packet was accepted
bit 0	<b>PIDEF:</b> PID Check Failure Flag bit 1 = PID check failed 0 = PID check passed

# PIC18F1XK50/PIC18LF1XK50

## 22.5.4 USB ERROR INTERRUPT ENABLE REGISTER (UEIE)

The USB Error Interrupt Enable register (Register 22-10) contains the enable bits for each of the USB error interrupt sources. Setting any of these bits will enable the respective error interrupt source in the UEIR register to propagate into the UERR bit at the top level of the interrupt logic.

As with the UIE register, the enable bits only affect the propagation of an interrupt condition to the microcontroller's interrupt logic. The flag bits are still set by their interrupt conditions, allowing them to be polled and serviced without actually generating an interrupt.

### REGISTER 22-10: UEIE: USB ERROR INTERRUPT ENABLE REGISTER

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE
bit 7							

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **BTSEE:** Bit Stuff Error Interrupt Enable bit

1 = Bit stuff error interrupt enabled  
0 = Bit stuff error interrupt disabled

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

bit 4 **BTOEE:** Bus Turnaround Time-out Error Interrupt Enable bit

1 = Bus turnaround time-out error interrupt enabled  
0 = Bus turnaround time-out error interrupt disabled

bit 3 **DFN8EE:** Data Field Size Error Interrupt Enable bit

1 = Data field size error interrupt enabled  
0 = Data field size error interrupt disabled

bit 2 **CRC16EE:** CRC16 Failure Interrupt Enable bit

1 = CRC16 failure interrupt enabled  
0 = CRC16 failure interrupt disabled

bit 1 **CRC5EE:** CRC5 Host Error Interrupt Enable bit

1 = CRC5 host error interrupt enabled  
0 = CRC5 host error interrupt disabled

bit 0 **PIDEE:** PID Check Failure Interrupt Enable bit

1 = PID check failure interrupt enabled  
0 = PID check failure interrupt disabled

## 22.6 USB Power Modes

Many USB applications will likely have several different sets of power requirements and configuration. The most common power modes encountered are Bus Power Only, Self-Power Only and Dual Power with Self-Power Dominance. The most common cases are presented here. Also provided is a means of estimating the current consumption of the USB transceiver.

### 22.6.1 BUS POWER ONLY

In Bus Power Only mode, all power for the application is drawn from the USB (Figure 22-9). This is effectively the simplest power method for the device.

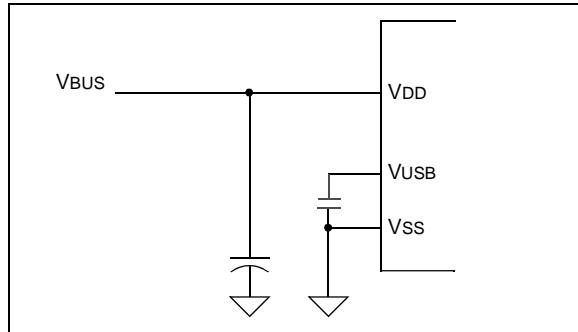
In order to meet the inrush current requirements of the USB 2.0 specifications, the total effective capacitance appearing across VBUS and ground must be no more than 10  $\mu$ F. If not, some kind of inrush liming is required. For more details, see section 7.2.4 of the USB 2.0 specification.

According to the USB 2.0 specification, all USB devices must also support a Low-Power Suspend mode. In the USB Suspend mode, devices must consume no more than 500  $\mu$ A (or 2.5 mA for high powered devices that are remote wake-up capable) from the 5V VBUS line of the USB cable.

The host signals the USB device to enter the Suspend mode by stopping all USB traffic to that device for more than 3 ms. This condition will cause the IDLEIF bit in the UIR register to become set.

During the USB Suspend mode, the D+ or D- pull-up resistor must remain active, which will consume some of the allowed suspend current: 500  $\mu$ A/2.5 mA budget.

**FIGURE 22-9: BUS POWER ONLY**



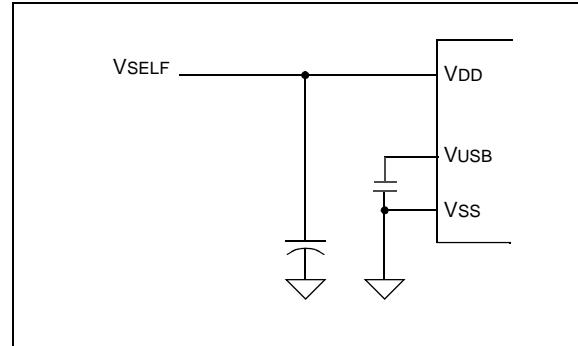
### 22.6.2 SELF-POWER ONLY

In Self-Power Only mode, the USB application provides its own power, with very little power being pulled from the USB. Figure 22-10 shows an example.

In order to meet compliance specifications, the USB module (and the D+ or D- pull-up resistor) should not be enabled until the host actively drives VBUS high.

The application should never source any current onto the 5V VBUS pin of the USB cable.

**FIGURE 22-10: SELF-POWER ONLY**



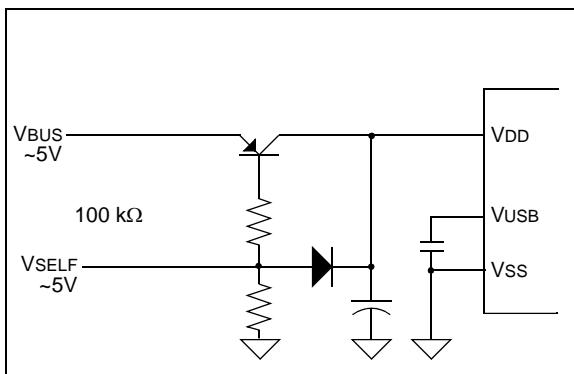
# PIC18F1XK50/PIC18LF1XK50

## 22.6.3 DUAL POWER WITH SELF-POWER DOMINANCE

Some applications may require a dual power option. This allows the application to use internal power primarily, but switch to power from the USB when no internal power is available. Figure 22-11 shows a simple Dual Power with Self-Power Dominance mode example, which automatically switches between Self-Power Only and USB Bus Power Only modes.

Dual power devices must also meet all of the special requirements for inrush current and Suspend mode current and must not enable the USB module until VBUS is driven high. See **Section 22.6.1 “Bus Power Only”** and **Section 22.6.2 “Self-Power Only”** for descriptions of those requirements. Additionally, dual power devices must never source current onto the 5V VBUS pin of the USB cable.

**FIGURE 22-11: DUAL POWER EXAMPLE**



**Note:** Users should keep in mind the limits for devices drawing power from the USB. According to USB Specification 2.0, this cannot exceed 100 mA per low-power device or 500 mA per high-power device.

## 22.6.4 USB TRANSCEIVER CURRENT CONSUMPTION

The USB transceiver consumes a variable amount of current depending on the characteristic impedance of the USB cable, the length of the cable, the VUSB supply voltage and the actual data patterns moving across the USB cable. Longer cables have larger capacitances and consume more total energy when switching output states.

Data patterns that consist of “IN” traffic consume far more current than “OUT” traffic. IN traffic requires the PIC® device to drive the USB cable, whereas OUT traffic requires that the host drive the USB cable.

The data that is sent across the USB cable is NRZI encoded. In the NRZI encoding scheme, ‘0’ bits cause a toggling of the output state of the transceiver (either from a “J” state to a “K” state, or vice versa). With the exception of the effects of bit-stuffing, NRZI encoded ‘1’ bits do not cause the output state of the transceiver to change. Therefore, IN traffic consisting of data bits of value, ‘0’, cause the most current consumption, as the transceiver must charge/discharge the USB cable in order to change states.

More details about NRZI encoding and bit-stuffing can be found in the USB 2.0 specification’s section 7.1, although knowledge of such details is not required to make USB applications using the PIC18F1XK50/PIC18LF1XK50 of microcontrollers. Among other things, the SIE handles bit-stuffing/unstuffing, NRZI encoding/decoding and CRC generation/checking in hardware.

The total transceiver current consumption will be application-specific. However, to help estimate how much current actually may be required in full-speed applications, Equation 22-1 can be used.

Example 22-2 shows how this equation can be used for a theoretical application.

## EQUATION 22-1: ESTIMATING USB TRANSCEIVER CURRENT CONSUMPTION

$$I_{XCVR} = \frac{(60 \text{ mA} \cdot V_{USB} \cdot P_{ZERO} \cdot P_{IN} \cdot L_{CABLE})}{(3.3V \cdot 5m)} + I_{PULLUP}$$

<b>Legend:</b>	<p><b>V<sub>USB</sub>:</b> Voltage applied to the V<sub>USB</sub> pin in volts. (Should be 3.0V to 3.6V.)</p> <p><b>P<sub>ZERO</sub>:</b> Percentage (in decimal) of the IN traffic bits sent by the PIC® device that are a value of '0'.</p> <p><b>P<sub>IN</sub>:</b> Percentage (in decimal) of total bus bandwidth that is used for IN traffic.</p> <p><b>L<sub>CABLE</sub>:</b> Length (in meters) of the USB cable. The USB 2.0 specification requires that full-speed applications use cables no longer than 5m.</p> <p><b>I<sub>PULLUP</sub>:</b> Current which the nominal, 1.5 kΩ pull-up resistor (when enabled) must supply to the USB cable. On the host or hub end of the USB cable, 15 kΩ nominal resistors (14.25 kΩ to 24.8 kΩ) are present which pull both the D+ and D- lines to ground. During bus Idle conditions (such as between packets or during USB Suspend mode), this results in up to 218 μA of quiescent current drawn at 3.3V.</p> <p>I<sub>PULLUP</sub> is also dependant on bus traffic conditions and can be as high as 2.2 mA when the USB bandwidth is fully utilized (either IN or OUT traffic) for data that drives the lines to the "K" state most of the time.</p>
----------------	--

## EXAMPLE 22-2: CALCULATING USB TRANSCEIVER CURRENT<sup>†</sup>

For this example, the following assumptions are made about the application:

- 3.3V will be applied to V<sub>USB</sub> and V<sub>DD</sub>, with the core voltage regulator enabled.
- This is a full-speed application that uses one interrupt IN endpoint that can send one packet of 64 bytes every 1 ms, with no restrictions on the values of the bytes being sent. The application may or may not have additional traffic on OUT endpoints.
- A regular USB "B" or "mini-B" connector will be used on the application circuit board.

In this case, P<sub>ZERO</sub> = 100% = 1, because there should be no restriction on the value of the data moving through the IN endpoint. All 64 kBps of data could potentially be bytes of value, 00h. Since '0' bits cause toggling of the output state of the transceiver, they cause the USB transceiver to consume extra current charging/discharging the cable. In this case, 100% of the data bits sent can be of value '0'. This should be considered the "max" value, as normal data will consist of a fair mix of ones and zeros.

This application uses 64 kBps for IN traffic out of the total bus bandwidth of 1.5 MBps (12 Mbps), therefore:

$$P_{IN} = \frac{64 \text{ kBps}}{1.5 \text{ MBps}} = 4.3\% = 0.043$$

Since a regular "B" or "mini-B" connector is used in this application, the end user may plug in any type of cable up to the maximum allowed 5 m length. Therefore, we use the worst-case length:

$$L_{CABLE} = 5 \text{ meters}$$

Assume I<sub>PULLUP</sub> = 2.2 mA. The actual value of I<sub>PULLUP</sub> will likely be closer to 218 μA, but allow for the worst-case. USB bandwidth is shared between all the devices which are plugged into the root port (via hubs). If the application is plugged into a USB 1.1 hub that has other devices plugged into it, your device may see host to device traffic on the bus, even if it is not addressed to your device. Since any traffic, regardless of source, can increase the I<sub>PULLUP</sub> current above the base 218 μA, it is safest to allow for the worst-case of 2.2 mA.

Therefore:

$$I_{XCVR} = \frac{(60 \text{ mA} \cdot 3.3V \cdot 1 \cdot 0.043 \cdot 5m)}{(3.3V \cdot 5m)} + 2.2 \text{ mA} = 4.8 \text{ mA}$$

The calculated value should be considered an approximation and additional guardband or application-specific product testing is recommended. The transceiver current is "in addition to" the rest of the current consumed by the PIC18F1XK50/PIC18LF1XK50 device that is needed to run the core, drive the other I/O lines, power the various modules, etc.

# PIC18F1XK50/PIC18LF1XK50

## 22.7 Oscillator

The USB module has specific clock requirements. For full-speed operation, the clock source must be 48 MHz. Even so, the microcontroller core and other peripherals are not required to run at that clock speed. Available clocking options are described in detail in **Section 2.11 “USB Operation”**.

## 22.8 Interrupt-On-Change for D+/D- pins

The PIC18F1XK50/PIC18LF1XK50 has interrupt-on-change functionality on both D+ and D- data pins. This feature allows the device to detect voltage level changes when first connected to a USB host/hub.

The USB host/hub has 15K pull-down resistors on the D+ and D- pins. When the PIC18F1XK50/PIC18LF1XK50 attaches to the bus the D+ and D- pins can detect voltage changes. External resistors are needed for each pin to maintain a high state on the pins when detached.

The USB module must be disable (USBEN = 0) for the interrupt-on-change to function. Enabling the USB module (USBEN = 1) will automatically disable the interrupt-on-change for D+ and D- pins. Refer to **Section 7.11 “PORTA and PORTB Interrupt-on-Change”** for mode detail.

## 22.9 USB Firmware and Drivers

Microchip provides a number of application-specific resources, such as USB firmware and driver support. Refer to [www.microchip.com](http://www.microchip.com) for the latest firmware and driver support.

**TABLE 22-4: REGISTERS ASSOCIATED WITH USB MODULE OPERATION<sup>(1)</sup>**

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Details on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	65
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCL1IP	USBIP	TMR3IP	—	73
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCL1IF	USBIF	TMR3IF	—	69
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCL1IE	USBIE	TMR3IE	—	71
UCON	—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—	246
UCFG	UTEYE	—	—	UPUEN	—	FSEN	PPB1	PPB0	248
USTAT	—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI	—	250
UADDR	—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	252
UFRML	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0	246
UFRMH	—	—	—	—	—	FRM10	FRM9	FRM8	246
UIR	—	SOFIF	STALLIF	IDLEIF	TRNIF	ACTVIF	UERRIF	URSTIF	260
UIE	—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE	262
UEIR	BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF	263
UEIE	BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE	264
UEP0	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	251
UEP1	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	251
UEP2	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	251
UEP3	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	251
UEP4	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	251
UEP5	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	251
UEP6	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	251
UEP7	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	251

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

**Note 1:** This table includes only those hardware mapped SFRs located in Bank 15 of the data memory space. The Buffer Descriptor registers, which are mapped into Bank 4 and are not true SFRs, are listed separately in Table 22-3.

## 22.10 Overview of USB

This section presents some of the basic USB concepts and useful information necessary to design a USB device. Although much information is provided in this section, there is a plethora of information provided within the USB specifications and class specifications. Thus, the reader is encouraged to refer to the USB specifications for more information ([www.usb.org](http://www.usb.org)). If you are very familiar with the details of USB, then this section serves as a basic, high-level refresher of USB.

### 22.10.1 LAYERED FRAMEWORK

USB device functionality is structured into a layered framework graphically shown in Figure 22-12. Each level is associated with a functional level within the device. The highest layer, other than the device, is the configuration. A device may have multiple configurations. For example, a particular device may have multiple power requirements based on Self-Power Only or Bus Power Only modes.

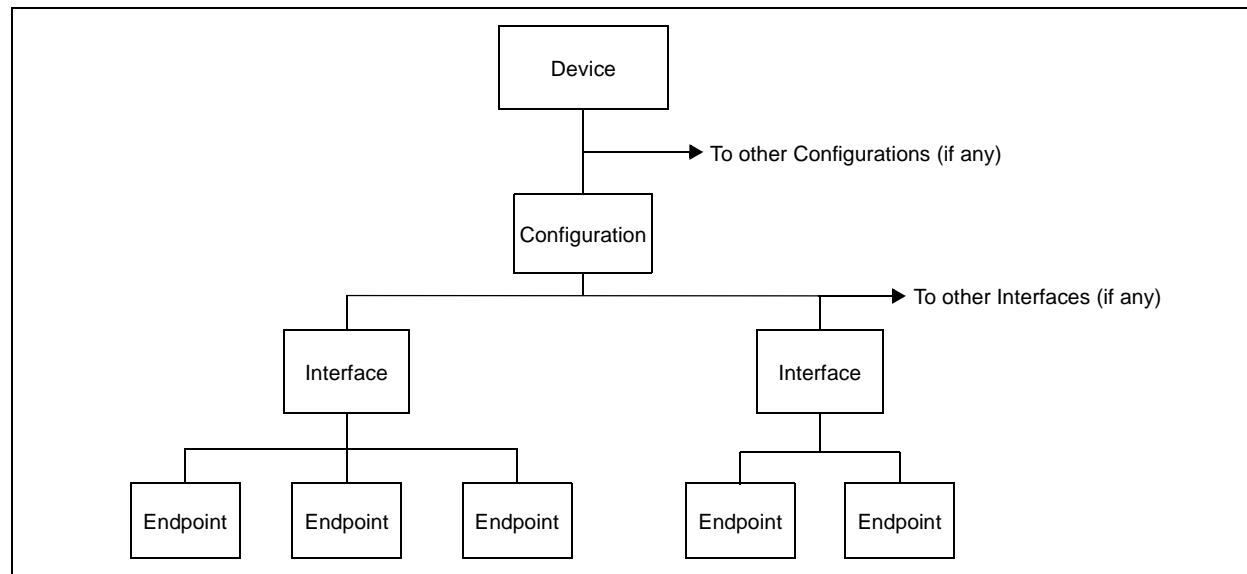
For each configuration, there may be multiple interfaces. Each interface could support a particular mode of that configuration.

Below the interface is the endpoint(s). Data is directly moved at this level. There can be as many as 16 bidirectional endpoints. Endpoint 0 is always a control endpoint and by default, when the device is on the bus, Endpoint 0 must be available to configure the device.

### 22.10.2 FRAMES

Information communicated on the bus is grouped into 1 ms time slots, referred to as frames. Each frame can contain many transactions to various devices and endpoints. Figure 22-8 shows an example of a transaction within a frame.

**FIGURE 22-12: USB LAYERS**



### 22.10.3 TRANSFERS

There are four transfer types defined in the USB specification.

- **Isochronous:** This type provides a transfer method for large amounts of data (up to 1023 bytes) with timely delivery ensured; however, the data integrity is not ensured. This is good for streaming applications where small data loss is not critical, such as audio.
- **Bulk:** This type of transfer method allows for large amounts of data to be transferred with ensured data integrity; however, the delivery timeliness is not ensured.
- **Interrupt:** This type of transfer provides for ensured timely delivery for small blocks of data, plus data integrity is ensured.
- **Control:** This type provides for device setup control.

While full-speed devices support all transfer types, low-speed devices are limited to interrupt and control transfers only.

### 22.10.4 POWER

Power is available from the Universal Serial Bus. The USB specification defines the bus power requirements. Devices may either be self-powered or bus powered. Self-powered devices draw power from an external source, while bus powered devices use power supplied from the bus.

# PIC18F1XK50/PIC18LF1XK50

---

The USB specification limits the power taken from the bus. Each device is ensured 100 mA at approximately 5V (one unit load). Additional power may be requested, up to a maximum of 500 mA. Note that power above one unit load is a request and the host or hub is not obligated to provide the extra current. Thus, a device capable of consuming more than one unit load must be able to maintain a low-power configuration of a one unit load or less, if necessary.

The USB specification also defines a Suspend mode. In this situation, current must be limited to 500  $\mu$ A, averaged over 1 second. A device must enter a Suspend state after 3 ms of inactivity (i.e., no SOF tokens for 3 ms). A device entering Suspend mode must drop current consumption within 10 ms after Suspend. Likewise, when signaling a wake-up, the device must signal a wake-up within 10 ms of drawing current above the Suspend limit.

## 22.10.5 ENUMERATION

When the device is initially attached to the bus, the host enters an enumeration process in an attempt to identify the device. Essentially, the host interrogates the device, gathering information such as power consumption, data rates and sizes, protocol and other descriptive information; descriptors contain this information. A typical enumeration process would be as follows:

1. USB Reset: Reset the device. Thus, the device is not configured and does not have an address (address 0).
2. Get Device Descriptor: The host requests a small portion of the device descriptor.
3. USB Reset: Reset the device again.
4. Set Address: The host assigns an address to the device.
5. Get Device Descriptor: The host retrieves the device descriptor, gathering info such as manufacturer, type of device, maximum control packet size.
6. Get configuration descriptors.
7. Get any other descriptors.
8. Set a configuration.

The exact enumeration process depends on the host.

## 22.10.6 DESCRIPTORS

There are eight different standard descriptor types of which five are most important for this device.

### 22.10.6.1 Device Descriptor

The device descriptor provides general information, such as manufacturer, product number, serial number, the class of the device and the number of configurations. There is only one device descriptor.

### 22.10.6.2 Configuration Descriptor

The configuration descriptor provides information on the power requirements of the device and how many different interfaces are supported when in this configuration. There may be more than one configuration for a device (i.e., low-power and high-power configurations).

### 22.10.6.3 Interface Descriptor

The interface descriptor details the number of endpoints used in this interface, as well as the class of the interface. There may be more than one interface for a configuration.

### 22.10.6.4 Endpoint Descriptor

The endpoint descriptor identifies the transfer type (**Section 22.10.3 “Transfers”**) and direction, as well as some other specifics for the endpoint. There may be many endpoints in a device and endpoints may be shared in different configurations.

### 22.10.6.5 String Descriptor

Many of the previous descriptors reference one or more string descriptors. String descriptors provide human readable information about the layer (**Section 22.10.1 “Layered Framework”**) they describe. Often these strings show up in the host to help the user identify the device. String descriptors are generally optional to save memory and are encoded in a unicode format.

## 22.10.7 BUS SPEED

Each USB device must indicate its bus presence and speed to the host. This is accomplished through a 1.5 k $\Omega$  resistor which is connected to the bus at the time of the attachment event.

Depending on the speed of the device, the resistor either pulls up the D+ or D- line to 3.3V. For a low-speed device, the pull-up resistor is connected to the D- line. For a full-speed device, the pull-up resistor is connected to the D+ line.

## 22.10.8 CLASS SPECIFICATIONS AND DRIVERS

USB specifications include class specifications which operating system vendors optionally support. Examples of classes include Audio, Mass Storage, Communications and Human Interface (HID). In most cases, a driver is required at the host side to ‘talk’ to the USB device. In custom applications, a driver may need to be developed. Fortunately, drivers are available for most common host systems for the most common classes of devices. Thus, these drivers can be reused.

## 23.0 RESET

The PIC18F1XK50/PIC18LF1XK50 devices differentiate between various kinds of Reset:

- Power-on Reset (POR)
- MCLR Reset during normal operation
- MCLR Reset during power-managed modes
- Watchdog Timer (WDT) Reset (during execution)
- Programmable Brown-out Reset (BOR)
- RESET Instruction
- Stack Full Reset
- Stack Underflow Reset

This section discusses Resets generated by MCLR, POR and BOR and covers the operation of the various start-up timers. Stack Reset events are covered in **Section 3.1.2.4 “Stack Full and Underflow Resets”**. WDT Resets are covered in **Section 24.2 “Watchdog Timer (WDT)”**.

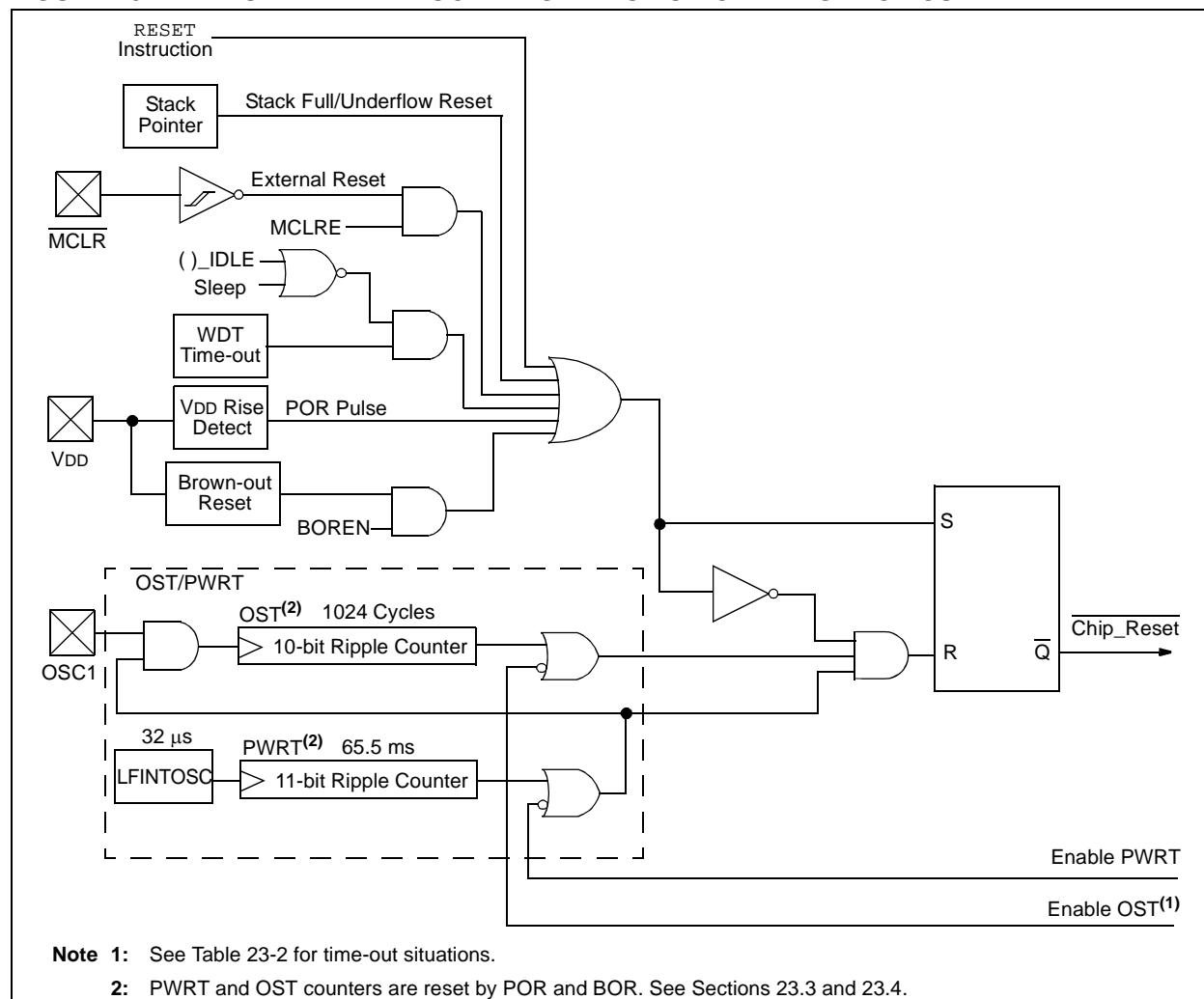
A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 23-1.

## 23.1 RCON Register

Device Reset events are tracked through the RCON register (Register 23-1). The lower five bits of the register indicate that a specific Reset event has occurred. In most cases, these bits can only be cleared by the event and must be set by the application after the event. The state of these flag bits, taken together, can be read to indicate the type of Reset that just occurred. This is described in more detail in **Section 23.6 “Reset State of Registers”**.

The RCON register also has control bits for setting interrupt priority (IPEN) and software control of the BOR (SBOREN). Interrupt priority is discussed in **Section 7.0 “Interrupts”**. BOR is covered in **Section 23.4 “Brown-out Reset (BOR)”**.

**FIGURE 23-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**



# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 23-1: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	SBOREN <sup>(1)</sup>	—	RI	TO	PD	POR <sup>(2)</sup>	BOR
bit 7	bit 0						

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7	<b>IPEN:</b> Interrupt Priority Enable bit 1 = Enable priority levels on interrupts 0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)
bit 6	<b>SBOREN:</b> BOR Software Enable bit <sup>(1)</sup> <u>If BOREN&lt;1:0&gt; = 01:</u> 1 = BOR is enabled 0 = BOR is disabled <u>If BOREN&lt;1:0&gt; = 00, 10 or 11:</u> Bit is disabled and read as '0'.
bit 5	<b>Unimplemented:</b> Read as '0'
bit 4	<b>RI:</b> RESET Instruction Flag bit 1 = The RESET instruction was not executed (set by firmware or Power-on Reset) 0 = The RESET instruction was executed causing a device Reset (must be set in firmware after a code-executed Reset occurs)
bit 3	<b>TO:</b> Watchdog Time-out Flag bit 1 = Set by power-up, CLRWDT instruction or SLEEP instruction 0 = A WDT time-out occurred
bit 2	<b>PD:</b> Power-down Detection Flag bit 1 = Set by power-up or by the CLRWDT instruction 0 = Set by execution of the SLEEP instruction
bit 1	<b>POR:</b> Power-on Reset Status bit <sup>(2)</sup> 1 = No Power-on Reset occurred 0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)
bit 0	<b>BOR:</b> Brown-out Reset Status bit <sup>(3)</sup> 1 = A Brown-out Reset has not occurred (set by firmware only) 0 = A Brown-out Reset occurred (must be set by firmware after a POR or Brown-out Reset occurs)

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

**2:** The actual Reset value of POR is determined by the type of device Reset. See the notes following this register and **Section 23.6 “Reset State of Registers”** for additional information.

**3:** See Table 23-3.

## 23.2 Master Clear (MCLR)

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

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

In PIC18F1XK50/PIC18LF1XK50 devices, the MCLR input can be disabled with the MCLRE Configuration bit. When MCLR is disabled, the pin becomes a digital input. See **Section 9.1 “PORTA, TRISA and LATA Registers”** for more information.

## 23.3 Power-on Reset (POR)

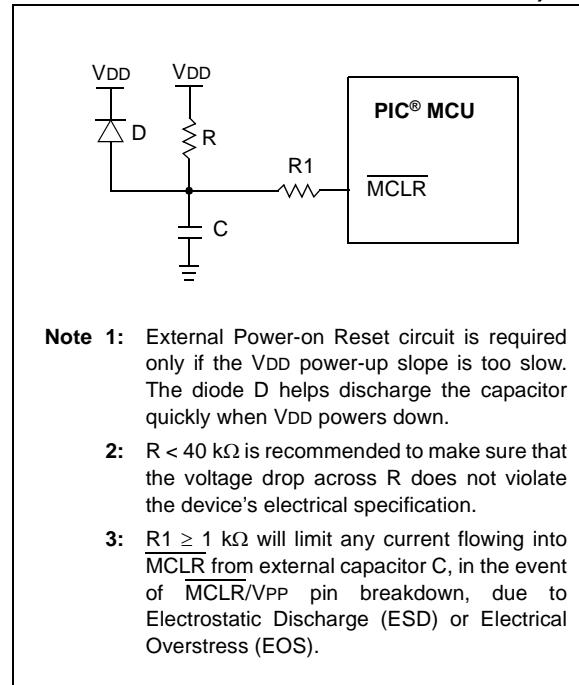
A Power-on Reset pulse is generated on-chip whenever VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.

To take advantage of the POR circuitry, tie the MCLR pin through a resistor (1 k $\Omega$  to 10 k $\Omega$ ) to VDD. This will eliminate external RC components usually needed to create a Power-on Reset delay.

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

POR events are captured by the POR bit of the RCON register. The state of the bit is set to '0' whenever a POR occurs; it does not change for any other Reset event. POR is not reset to '1' by any hardware event. To capture multiple events, the user must manually set the bit to '1' by software following any POR.

**FIGURE 23-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)**



## 23.4 Brown-out Reset (BOR)

PIC18F1XK50/PIC18LF1XK50 devices implement a BOR circuit that provides the user with a number of configuration and power-saving options. The BOR is controlled by the BORV<1:0> and BOREN<1:0> bits of the CONFIG2L Configuration register. There are a total of four BOR configurations which are summarized in Table 23-1.

The BOR threshold is set by the BORV<1:0> bits. If BOR is enabled (any values of BOREN<1:0>, except '00'), any drop of VDD below VBOR for greater than TBOR will reset the device. A Reset may or may not occur if VDD falls below VBOR for less than TBOR. The chip will remain in Brown-out Reset until VDD rises above VBOR.

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

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

### 23.4.1 SOFTWARE ENABLED BOR

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

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

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

### 23.4.2 DETECTING BOR

When BOR is enabled, the BOR bit always resets to '0' on any BOR or POR event. This makes it difficult to determine if a BOR event has occurred just by reading the state of BOR alone. A more reliable method is to simultaneously check the state of both POR and BOR. This assumes that the POR and BOR bits are reset to '1' by software immediately after any POR event. If BOR is '0' while POR is '1', it can be reliably assumed that a BOR event has occurred.

### 23.4.3 DISABLING BOR IN SLEEP MODE

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

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

TABLE 23-1: BOR CONFIGURATIONS

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

## 23.5 Device Reset Timers

PIC18F1XK50/PIC18LF1XK50 devices incorporate three separate on-chip timers that help regulate the Power-on Reset process. Their main function is to ensure that the device clock is stable before code is executed. These timers are:

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

### 23.5.1 POWER-UP TIMER (PWRT)

The Power-up Timer (PWRT) of PIC18F1XK50/PIC18LF1XK50 devices is an 11-bit counter which uses the LFINTOSC source as the clock input. This yields an approximate time interval of  $2048 \times 32 \mu\text{s} = 65.6 \text{ ms}$ . While the PWRT is counting, the device is held in Reset.

The power-up time delay depends on the LFINTOSC clock and will vary from chip-to-chip due to temperature and process variation. See **Section 27.0 “Electrical Specifications”** for details.

The PWRT is enabled by clearing the PWRTEN Configuration bit.

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

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

The OST time-out is invoked only for XT, LP, HS and HSPLL modes and only on Power-on Reset, or on exit from all power-managed modes that stop the external oscillator.

### 23.5.3 PLL LOCK TIME-OUT

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

### 23.5.4 TIME-OUT SEQUENCE

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

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

The total time-out will vary based on oscillator configuration and the status of the PWRT. Figure 23-3, Figure 23-4, Figure 23-5, Figure 23-6 and Figure 23-7 all depict time-out sequences on power-up, with the Power-up Timer enabled and the device operating in HS Oscillator mode. Figures 23-3 through 23-6 also apply to devices operating in XT or LP modes. For devices in RC mode and with the PWRT disabled, on the other hand, there will be no time-out at all.

Since the time-outs occur from the POR pulse, if MCLR is kept low long enough, all time-outs will expire, after which, bringing MCLR high will allow program execution to begin immediately (Figure 23-5). This is useful for testing purposes or to synchronize more than one PIC18F1XK50/PIC18LF1XK50 device operating in parallel.

**TABLE 23-2: TIME-OUT IN VARIOUS SITUATIONS**

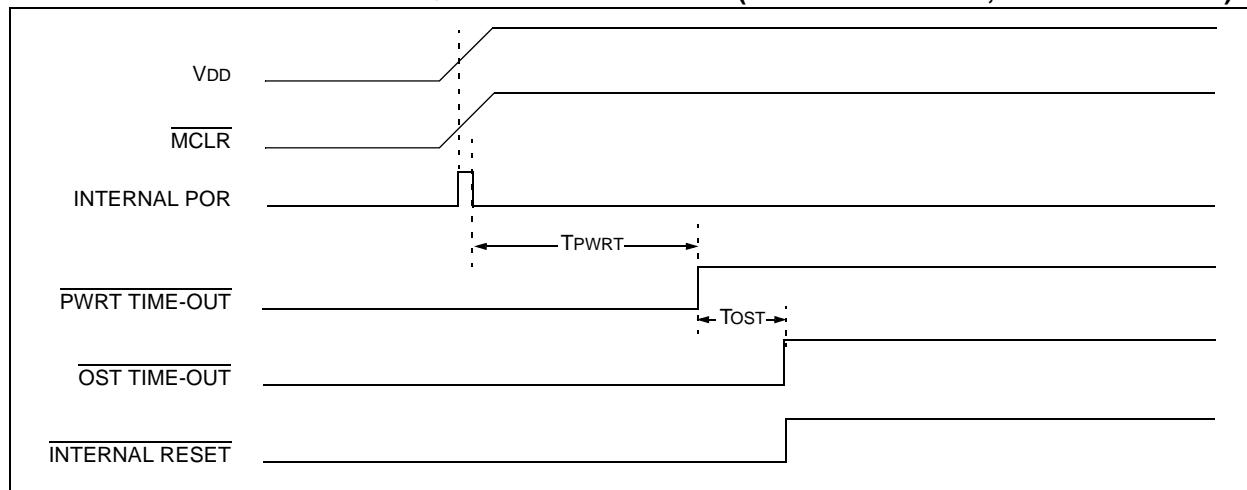
Oscillator Configuration	Power-up <sup>(2)</sup> and Brown-out		Exit from Power-Managed Mode
	<u>PWRTE</u> N = 0	<u>PWRTE</u> N = 1	
HSPLL	66 ms <sup>(1)</sup> + 1024 Tosc + 2 ms <sup>(2)</sup>	1024 Tosc + 2 ms <sup>(2)</sup>	1024 Tosc + 2 ms <sup>(2)</sup>
HS, XT, LP	66 ms <sup>(1)</sup> + 1024 Tosc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

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

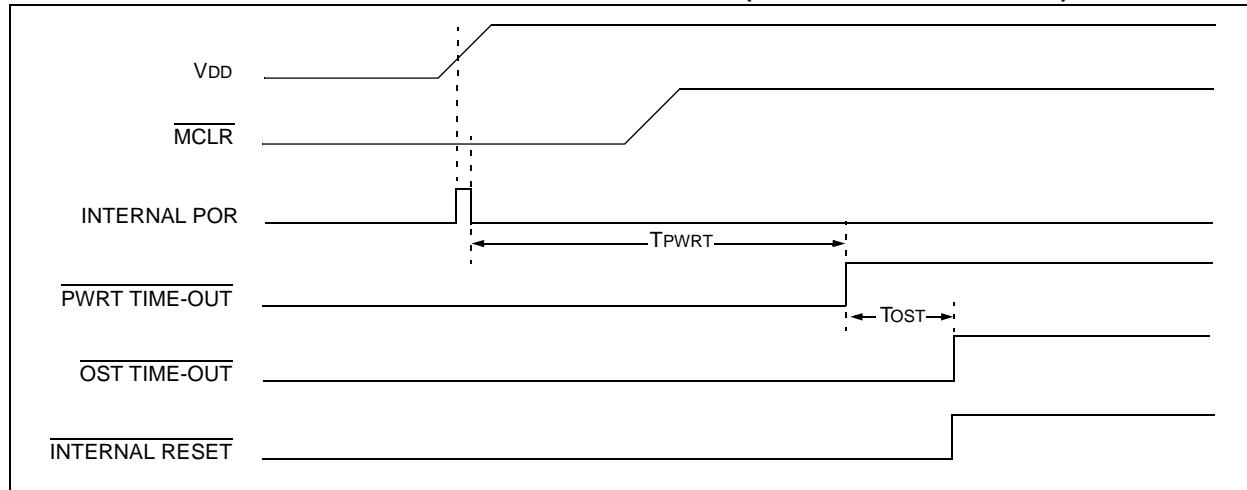
**2:** 2 ms is the nominal time required for the PLL to lock.

# PIC18F1XK50/PIC18LF1XK50

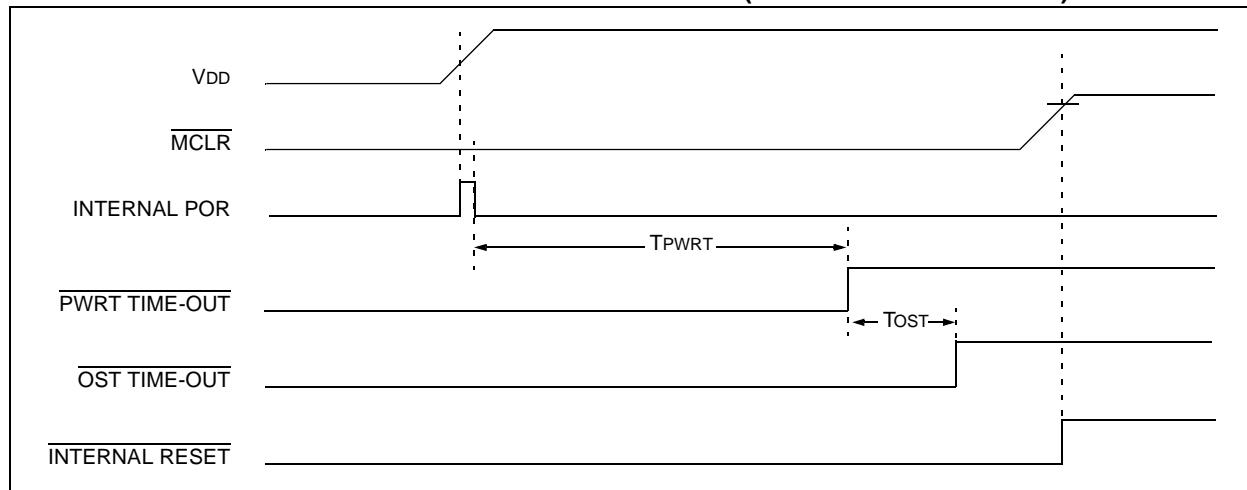
**FIGURE 23-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO V<sub>DD</sub>, V<sub>DD</sub> RISE < T<sub>PWRT</sub>)**



**FIGURE 23-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO V<sub>DD</sub>): CASE 1**



**FIGURE 23-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO V<sub>DD</sub>): CASE 2**



# PIC18F1XK50/PIC18LF1XK50

FIGURE 23-6: SLOW RISE TIME (MCLR TIED TO VDD, VDD RISE > TPWRT)

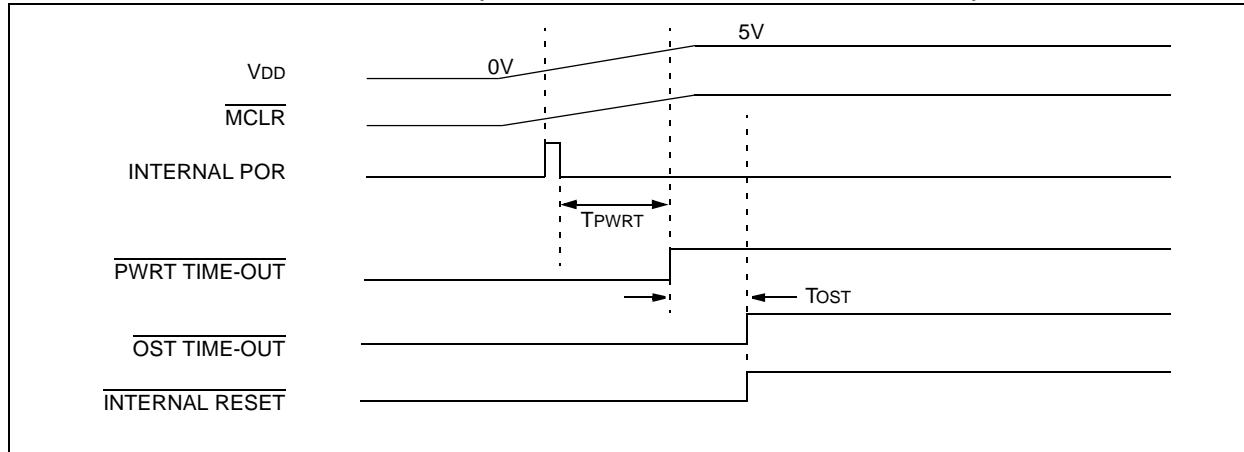
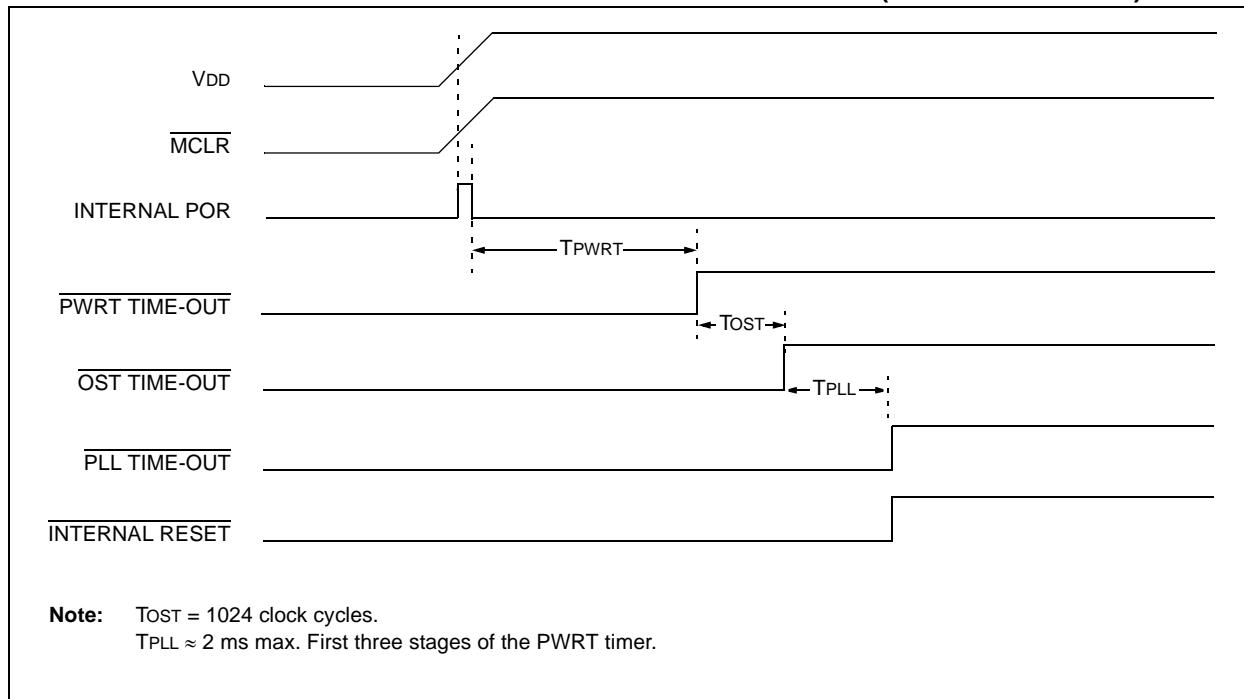


FIGURE 23-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED (MCLR TIED TO VDD)



## 23.6 Reset State of Registers

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

Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register,  $\overline{RI}$ ,  $\overline{TO}$ ,  $\overline{PD}$ ,  $\overline{POR}$  and  $\overline{BOR}$ , are set or cleared differently in different Reset situations, as indicated in Table 23-3. These bits are used by software to determine the nature of the Reset.

Table 23-4 describes the Reset states for all of the Special Function Registers. These are categorized by Power-on and Brown-out Resets, Master Clear and WDT Resets and WDT wake-ups.

**TABLE 23-3: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER**

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

**Legend:** u = unchanged

**Note 1:** When the wake-up is due to an interrupt and the GIEH or GIEL bits are set, the PC is loaded with the interrupt vector (008h or 0018h).

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

# PIC18F1XK50/PIC18LF1XK50

TABLE 23-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Address	Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
TOSU	FFFh	---0 0000	---0 0000	---0 uuuu <sup>(3)</sup>
TOSH	FFEh	0000 0000	0000 0000	uuuu uuuu <sup>(3)</sup>
TOSL	FFDh	0000 0000	0000 0000	uuuu uuuu <sup>(3)</sup>
STKPTR	FFCh	00-0 0000	uu-0 0000	uu-u uuuu <sup>(3)</sup>
PCLATU	FFBh	---0 0000	---0 0000	---u uuuu
PCLATH	FFAh	0000 0000	0000 0000	uuuu uuuu
PCL	FF9h	0000 0000	0000 0000	PC + 2 <sup>(2)</sup>
TBLPTRU	FF8h	---0 0000	---0 0000	---u uuuu
TBLPTRH	FF7h	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	FF6h	0000 0000	0000 0000	uuuu uuuu
TABLAT	FF5h	0000 0000	0000 0000	uuuu uuuu
PRODH	FF4h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	FF3h	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	FF2h	0000 000x	0000 000u	uuuu uuuu <sup>(1)</sup>
INTCON2	FF1h	1111 -1-1	1111 -1-1	uuuu -u-u <sup>(1)</sup>
INTCON3	FF0h	11-0 0-00	11-0 0-00	uu-u u-uu <sup>(1)</sup>
INDF0	FEFh	N/A	N/A	N/A
POSTINC0	FEEh	N/A	N/A	N/A
POSTDEC0	FEDh	N/A	N/A	N/A
PREINC0	FECh	N/A	N/A	N/A
PLUSW0	FEBh	N/A	N/A	N/A
FSR0H	FEAh	---- 0000	---- 0000	---- uuuu
FSR0L	FE9h	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	FE8h	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	FE7h	N/A	N/A	N/A
POSTINC1	FE6h	N/A	N/A	N/A
POSTDEC1	FE5h	N/A	N/A	N/A
PREINC1	FE4h	N/A	N/A	N/A
PLUSW1	FE3h	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 23-3 for Reset value for specific condition.

**5:** All bits of the ANSELH register initialize to '0' if the PBADEN bit of CONFIG3H is '0'.

# PIC18F1XK50/PIC18LF1XK50

TABLE 23-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Address	Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
FSR1H	FE2h	---- 0000	---- 0000	---- uuuu
FSR1L	FE1h	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	FE0h	---- 0000	---- 0000	---- uuuu
INDF2	FDFh	N/A	N/A	N/A
POSTINC2	FDEh	N/A	N/A	N/A
POSTDEC2	FDDh	N/A	N/A	N/A
PREINC2	FDCh	N/A	N/A	N/A
PLUSW2	FDBh	N/A	N/A	N/A
FSR2H	FDAh	---- 0000	---- 0000	---- uuuu
FSR2L	FD9h	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	FD8h	---x xxxx	---u uuuu	---u uuuu
TMR0H	FD7h	0000 0000	0000 0000	uuuu uuuu
TMR0L	FD6h	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	FD5h	1111 1111	1111 1111	uuuu uuuu
OSCCON	FD3h	0011 qq00	0011 qq00	uuuu uuuu
OSCCON2	FD2h	---- -10x	---- -10x	---- -uuu
WDTCON	FD1h	---- ---0	---- ---0	---- ---u
RCON <sup>(4)</sup>	FD0h	0q-1 11q0	0q-q qquu	uq-u qquu
TMR1H	FCFh	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	FCEh	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	FCDh	0000 0000	u0uu uuuu	uuuu uuuu
TMR2	FCCh	0000 0000	0000 0000	uuuu uuuu
PR2	FCBh	1111 1111	1111 1111	1111 1111
T2CON	FCAh	-000 0000	-000 0000	-uuu uuuu
SSPBUF	FC9h	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	FC8h	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	FC7h	0000 0000	0000 0000	uuuu uuuu
SSPCON1	FC6h	0000 0000	0000 0000	uuuu uuuu
SSPCON2	FC5h	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 23-3 for Reset value for specific condition.
- 5:** All bits of the ANSELH register initialize to '0' if the PBADEN bit of CONFIG3H is '0'.

# PIC18F1XK50/PIC18LF1XK50

TABLE 23-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Address	Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
ADRESH	FC4h	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	FC3h	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	FC2h	--00 0000	--00 0000	--uu uuuu
ADCON1	FC1h	---- 0000	---- 0000	---- uuuu
ADCON2	FC0h	0-00 0000	0-00 0000	u-uu uuuu
CCPR1H	FBFh	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	FBEh	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	FBDh	0000 0000	0000 0000	uuuu uuuu
REFCON2	FBCh	---0 0000	---0 0000	---u uuuu
REFCON1	FBBh	000- 00-0	000- 00-0	uuu- uu-u
REFCON0	FBAh	0001 00--	0001 00--	uuuu uu--
PSTRCON	FB9h	---0 0001	---0 0001	---u uuuu
BAUDCON	FB8h	0100 0-00	0100 0-00	uuuu u-uu
PWM1CON	FB7h	0000 0000	0000 0000	uuuu uuuu
ECCP1AS	FB6h	0000 0000	0000 0000	uuuu uuuu
TMR3H	FB3h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	FB2h	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	FB1h	0000 0000	uuuu uuuu	uuuu uuuu
SPBRGH	FB0h	0000 0000	0000 0000	uuuu uuuu
SPBRG	FAFh	0000 0000	0000 0000	uuuu uuuu
RCREG	FAEh	0000 0000	0000 0000	uuuu uuuu
TXREG	FADh	0000 0000	0000 0000	uuuu uuuu
TXSTA	FACh	0000 0010	0000 0010	uuuu uuuu
RCSTA	FABh	0000 000x	0000 000x	uuuu uuuu
EEADR	FAAh	0000 0000	0000 0000	uuuu uuuu
EEADRH	FA9h	---- --00	---- --00	---- --uu
EEDATA	FA8h	0000 0000	0000 0000	uuuu uuuu
EECON2	FA7h	0000 0000	0000 0000	0000 0000
EECON1	FA6h	xx-0 x000	uu-0 u000	uu-0 u000

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

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

- Note 1:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).
- 2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
- 3:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.
- 4:** See Table 23-3 for Reset value for specific condition.
- 5:** All bits of the ANSELH register initialize to '0' if the PBADEN bit of CONFIG3H is '0'.

# PIC18F1XK50/PIC18LF1XK50

TABLE 23-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Address	Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
IPR2	FA2h	1111 111-	1111 111-	uuuu uuu-
PIR2	FA1h	0000 000-	0000 000-	uuuu uuu-(1)
PIE2	FA0h	0000 000-	0000 000-	uuuu uuu-
IPR1	F9Fh	-111 1111	-111 1111	-uuu uuuu
PIR1	F9Eh	-000 0000	-000 0000	-uuu uuuu(1)
PIE1	F9Dh	-000 0000	-000 0000	-uuu uuuu
OSCTUNE	F9Bh	0000 0000	0000 0000	uuuu uuuu
TRISC	F95h	1111 1111	1111 1111	uuuu uuuu
TRISB	F94h	1111 ----	1111 ----	uuuu ----
TRISA	F93h	--11 ----	--11 ----	--uu ----
LATC	F8Bh	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	F8Ah	xxxx ----	uuuu ----	uuuu ----
LATA	F89h	--xx ----	--uu ----	--uu ----
PORTC	F82h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	F81h	xxxx ----	uuuu ----	uuuu ----
PORTA	F80h	--xx x-xx	--xx x-xx	--uu u-uu
ANSELH <sup>(5)</sup>	F7Fh	---- 1111	---- 1111	---- uuuu
ANSEL	F7Eh	1111 1---	1111 1---	uuuu u---
IOCB	F7Ah	0000 ----	0000 ----	uuuu ----
IOCA	F79h	--00 0-00	--00 0-00	--uu u-uu
WPUB	F78h	1111 ----	1111 ----	uuuu ----
WPUA	F77h	--11 1---	--11 1---	--uu u---
SLRCON	F76h	---- -111	---- -111	---- -uuu
SSPMSK	F6Fh	1111 1111	1111 1111	uuuu uuuu
CM1CON0	F6Dh	0000 0000	0000 0000	uuuu uuuu
CM2CON1	F6Ch	0000 0000	0000 0000	uuuu uuuu
CM2CON0	F6Bh	0000 0000	0000 0000	uuuu uuuu
SRCON1	F69h	0000 0000	0000 0000	uuuu uuuu
SRCON0	F68h	0000 0000	0000 0000	uuuu uuuu
UCON	F64h	-0x0 000-	-0x0 000-	-uuu uuu-

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

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

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

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

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

**4:** See Table 23-3 for Reset value for specific condition.

**5:** All bits of the ANSELH register initialize to '0' if the PBADEN bit of CONFIG3H is '0'.

# PIC18F1XK50/PIC18LF1XK50

TABLE 23-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Address	Power-on Reset, Brown-out Reset	MCLR Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
USTAT	F63h	-xxx xxxx-	-xxx xxxx-	-uuu uuu-
UIR	F62h	-000 0000	-000 0000	-uuu uuuu
UCFG	F61h	0--0 -000	0--0 -000	u--u -uuu
UIE	F60h	-000 0000	-000 0000	-uuu uuuu
UEIR	F5Fh	0--0 0000	0--0 0000	u--u uuuu
UFRMH	F5Eh	---- -xxxx	---- -xxxx	---- -uuu
UFRML	F5Dh	xxxxx xxxx	xxxxx xxxx	uuuu uuuu
UADDR	F5Ch	-000 0000	-000 0000	-uuu uuuu
UEIE	F5Bh	0--0 0000	0--0 0000	u--u uuuu
UEP7	F5Ah	----0 0000	----0 0000	----u uuuu
UEP6	F59h	----0 0000	----0 0000	----u uuuu
UEP5	F58h	----0 0000	----0 0000	----u uuuu
UEP4	F57h	----0 0000	----0 0000	----u uuuu
UEP3	F56h	----0 0000	----0 0000	----u uuuu
UEP2	F55h	----0 0000	----0 0000	----u uuuu
UEP1	F54h	----0 0000	----0 0000	----u uuuu
UEP0	F53h	----0 0000	----0 0000	----u 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 23-3 for Reset value for specific condition.
- 5:** All bits of the ANSELH register initialize to '0' if the PBADEN bit of CONFIG3H is '0'.

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 24.0 SPECIAL FEATURES OF THE CPU

PIC18F1XK50/PIC18LF1XK50 devices include several features intended to maximize reliability and minimize cost through elimination of external components. These are:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Code Protection
- ID Locations
- In-Circuit Serial Programming™

The oscillator can be configured for the application depending on frequency, power, accuracy and cost. All of the options are discussed in detail in **Section 2.0 “Oscillator Module”**.

A complete discussion of device Resets and interrupts is available in previous sections of this data sheet.

In addition to their Power-up and Oscillator Start-up Timers provided for Resets, PIC18F1XK50/PIC18LF1XK50 devices have a Watchdog Timer, which is either permanently enabled via the Configuration bits or software controlled (if configured as disabled).

The inclusion of an internal RC oscillator also provides the additional benefits of a Fail-Safe Clock Monitor (FSCM) and Two-Speed Start-up. FSCM provides for background monitoring of the peripheral clock and automatic switchover in the event of its failure. Two-Speed Start-up enables code to be executed almost immediately on start-up, while the primary clock source completes its start-up delays.

All of these features are enabled and configured by setting the appropriate Configuration register bits.

# PIC18F1XK50/PIC18LF1XK50

---

## 24.1 Configuration Bits

The Configuration bits can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFFh), which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The WR bit in the EECON1 register starts a self-timed write to the Configuration register. In normal operation mode, a TBLWT instruction with the TBLPTR pointing to the Configuration register sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a TBLWT instruction can write a '1' or a '0' into the cell. For additional details on Flash programming, refer to **Section 4.5 "Writing to Flash Program Memory"**.

TABLE 24-1: CONFIGURATION BITS AND DEVICE IDs

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300000h	CONFIG1L	—	—	USBDIV	CPUDIV1	CPUDIV0	—	—	—	--00 0---
300001h	CONFIG1H	IESO	FCMEN	PCLKEN	PLLEN	FOSC3	FOSC2	FOSC1	FOSC0	0010 0111
300002h	CONFIG2L	—	—	—	BORV1	BORV0	BOREN1	BOREN0	PWRREN	---1 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	---1 1111
300005h	CONFIG3H	MCLRE	—	—	—	HFOFST	—	—	—	1--- 1---
300006h	CONFIG4L	—	XINST	—	—	BBSIZ	LVP	—	STVREN	-0-- 01-1
300008h	CONFIG5L	—	—	—	—	—	CP1	CP0	-----11	
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	—	—	WRT1	WRT0	-----11
30000Bh	CONFIG6H	WRTD	WRTB	WRTE	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	—	—	EBTR1	EBTR0	-----11
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1 <sup>(1)</sup>	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	qqqq qqqq <sup>(1)</sup>
3FFFFFh	DEVID2 <sup>(1)</sup>	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 1100

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

Shaded cells are unimplemented, read as '0'

**Note 1:** See Register 24-13 for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 24-1: CONFIG1L: CONFIGURATION REGISTER 1 LOW

U-0	U-0	R/P-0	R/P-0	R/P-0	U-0	U-0	U-0
—	—	USBDIV	CPUDIV1	CPUDIV0	—	—	—
bit 7	bit 0						

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

x = Bit is unknown

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

bit 5      **USBDIV:** USB Clock Selection bit

Selects the clock source for Low-speed USB operation

1 = USB clock comes from the OSC1/OSC2 divided by 2

0 = USB clock comes directly from the OSC1/OSC2 Oscillator block; no divide

bit 4-3      **CPUDIV<1:0>:** CPU System Clock Selection bits

11 = CPU system clock divided by 4

10 = CPU system clock divided by 3

01 = CPU system clock divided by 2

00 = No CPU system clock divide

bit 2-0      **Unimplemented:** Read as '0'

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 24-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH

R/P-0	R/P-0	R/P-1	R/P-0	R/P-0	R/P-1	R/P-1	R/P-1
IESO	FCMEN	PCLKEN	PLLEN	FOSC3	FOSC2	FOSC1	FOSC0
bit 7	bit 0						

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

x = Bit is unknown

bit 7	<b>IESO:</b> Internal/External Oscillator Switchover bit 1 = Oscillator Switchover mode enabled 0 = Oscillator Switchover mode disabled
bit 6	<b>FCMEN:</b> Fail-Safe Clock Monitor Enable bit 1 = Fail-Safe Clock Monitor enabled 0 = Fail-Safe Clock Monitor disabled
bit 5	<b>PCLKEN:</b> Primary Clock Enable bit 1 = Primary Clock enabled 0 = Primary Clock is under software control
bit 4	<b>PLLEN:</b> 4 X PLL Enable bit 1 = Oscillator multiplied by 4 0 = PLL is under software control
bit 3-0	<b>FOSC&lt;3:0&gt;:</b> Oscillator Selection bits 1111 = External RC oscillator, CLKOUT function on OSC2 1110 = External RC oscillator, CLKOUT function on OSC2 1101 = EC (low) 1100 = EC, CLKOUT function on OSC2 (low) 1011 = EC (medium) 1010 = EC, CLKOUT function on OSC2 (medium) 1001 = Internal RC oscillator, CLKOUT function on OSC2 1000 = Internal RC oscillator 0111 = External RC oscillator 0110 = External RC oscillator, CLKOUT function on OSC2 0101 = EC (high) 0100 = EC, CLKOUT function on OSC2 (high) 0011 = External RC oscillator, CLKOUT function on OSC2 0010 = HS oscillator 0001 = XT oscillator 0000 = LP oscillator

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 24-3: CONFIG2L: CONFIGURATION REGISTER 2 LOW

U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	BORV1 <sup>(1)</sup>	BORV0 <sup>(1)</sup>	BOREN1 <sup>(2)</sup>	BOREN0 <sup>(2)</sup>	PWRTE <sup>(2)</sup>
bit 7							bit 0

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

x = Bit is unknown

bit 7-5	<b>Unimplemented:</b> Read as '0'
bit 4-3	<b>BORV&lt;1:0&gt;:</b> Brown-out Reset Voltage bits <sup>(1)</sup> 11 = VBOR set to 1.9V nominal 10 = VBOR set to 2.2V nominal 01 = VBOR set to 2.7V nominal 00 = VBOR set to 3.0V nominal
bit 2-1	<b>BOREN&lt;1:0&gt;:</b> Brown-out Reset Enable bits <sup>(2)</sup> 11 = Brown-out Reset enabled in hardware only (SBOREN is disabled) 10 = Brown-out Reset enabled in hardware only and disabled in Sleep mode (SBOREN is disabled) 01 = Brown-out Reset enabled and controlled by software (SBOREN is enabled) 00 = Brown-out Reset disabled in hardware and software
bit 0	<b>PWRTE:</b> Power-up Timer Enable bit <sup>(2)</sup> 1 = PWRT disabled 0 = PWRT enabled

**Note 1:** See **Section 26.1 “DC Characteristics: Supply Voltage”** for specifications.

**2:** The Power-up Timer is decoupled from Brown-out Reset, allowing these features to be independently controlled.

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 24-4: CONFIG2H: CONFIGURATION REGISTER 2 HIGH

U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN
bit 7							bit 0

**Legend:**

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

x = Bit is unknown

bit 7-5      **Unimplemented:** Read as '0'bit 4-1      **WDTPS<3:0>:** Watchdog Timer Postscale Select bits

1111 = 1:32,768

1110 = 1:16,384

1101 = 1:8,192

1100 = 1:4,096

1011 = 1:2,048

1010 = 1:1,024

1001 = 1:512

1000 = 1:256

0111 = 1:128

0110 = 1:64

0101 = 1:32

0100 = 1:16

0011 = 1:8

0010 = 1:4

0001 = 1:2

0000 = 1:1

bit 0      **WDTEN:** Watchdog Timer Enable bit

1 = WDT is always enabled. SWDTEN bit has no effect

0 = WDT is controlled by SWDTEN bit of the WDTCON register

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 24-5: CONFIG3H: CONFIGURATION REGISTER 3 HIGH

R/P-1	U-0	U-0	U-0	R/P-1	U-0	U-0	U-0
MCLRE	—	—	—	HFOFST	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

x = Bit is unknown

- bit 7            **MCLRE:** MCLR Pin Enable bit  
                   1 = MCLR pin enabled; RA3 input pin disabled  
                   0 = RA3 input pin enabled; MCLR disabled
- bit 6-4          **Unimplemented:** Read as '0'
- bit 3            **HFOFST:** HFINTOSC Fast Start-up bit  
                   1 = HFINTOSC starts clocking the CPU without waiting for the oscillator to stabilize.  
                   0 = The system clock is held off until the HFINTOSC is stable.
- bit 2-0          **Unimplemented:** Read as '0'

## REGISTER 24-6: CONFIG4L: CONFIGURATION REGISTER 4 LOW

U-0	R/P-0	U-0	U-0	R/P-0	R/P-1	U-0	R/P-1
—	XINST	—	—	BBSIZ	LVP	—	STVREN
bit 7							bit 0

### Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

x = Bit is unknown

- bit 7            **Unimplemented:** Read as '0'
- bit 6            **XINST:** Extended Instruction Set Enable bit  
                   1 = Instruction set extension and Indexed Addressing mode enabled  
                   0 = Instruction set extension and Indexed Addressing mode disabled (Legacy mode)
- bit 5-4          **Unimplemented:** Read as '0'
- bit 3            **BBSIZ:** Boot Block Size Select bit  
                   1 = 2 kW boot block size for PIC18F14K50/PIC18LF14K50 (1 kW boot block size for PIC18F13K50/PIC18LF13K50)  
                   0 = 1 kW boot block size for PIC18F14K50/PIC18LF14K50 (512 W boot block size for PIC18F13K50/PIC18LF13K50)
- bit 2            **LVP:** Single-Supply ICSP™ Enable bit  
                   1 = Single-Supply ICSP enabled  
                   0 = Single-Supply ICSP disabled
- bit 1            **Unimplemented:** Read as '0'
- bit 0            **STVREN:** Stack Full/Underflow Reset Enable bit  
                   1 = Stack full/underflow will cause Reset  
                   0 = Stack full/underflow will not cause Reset

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 24-7: CONFIG5L: CONFIGURATION REGISTER 5 LOW

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	CP1	CP0
bit 7							bit 0

### Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

bit 7-2      **Unimplemented:** Read as '0'

bit 1      **CP1:** Code Protection bit

1 = Block 1 not code-protected

0 = Block 1 code-protected

bit 0      **CP0:** Code Protection bit

1 = Block 0 not code-protected

0 = Block 0 code-protected

## REGISTER 24-8: CONFIG5H: CONFIGURATION REGISTER 5 HIGH

R/C-1	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
CPD	CPB	—	—	—	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

bit 7      **CPD:** Data EEPROM Code Protection bit

1 = Data EEPROM not code-protected

0 = Data EEPROM code-protected

bit 6      **CPB:** Boot Block Code Protection bit

1 = Boot block not code-protected

0 = Boot block code-protected

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

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 24-9: CONFIG6L: CONFIGURATION REGISTER 6 LOW

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	WRT1	WRT0
bit 7							bit 0

### Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

bit 7-2      **Unimplemented:** Read as '0'

bit 1      **WRT1:** Write Protection bit  
1 = Block 1 not write-protected  
0 = Block 1 write-protected

bit 0      **WRT0:** Write Protection bit  
1 = Block 0 not write-protected  
0 = Block 0 write-protected

## REGISTER 24-10: CONFIG6H: CONFIGURATION REGISTER 6 HIGH

R/C-1	R/C-1	R-1	U-0	U-0	U-0	U-0	U-0
WRTD	WRTB	WR <sup>(1)</sup> TC	—	—	—	—	—
bit 7							bit 0

### Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

bit 7      **WRTD:** Data EEPROM Write Protection bit

1 = Data EEPROM not write-protected  
0 = Data EEPROM write-protected

bit 6      **WRTB:** Boot Block Write Protection bit  
1 = Boot block not write-protected  
0 = Boot block write-protected

bit 5      **WR<sup>(1)</sup>TC:** Configuration Register Write Protection bit<sup>(1)</sup>  
1 = Configuration registers not write-protected  
0 = Configuration registers write-protected

bit 4-0      **Unimplemented:** Read as '0'

**Note 1:** This bit is read-only in normal execution mode; it can be written only in Program mode.

# PIC18F1XK50/PIC18LF1XK50

---

---

## REGISTER 24-11: CONFIG7L: CONFIGURATION REGISTER 7 LOW

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	EBTR1	EBTR0
bit 7						bit 0	

### Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

bit 7-2      **Unimplemented:** Read as '0'

bit 1      **EBTR1:** Table Read Protection bit

1 = Block 1 not protected from table reads executed in other blocks

0 = Block 1 protected from table reads executed in other blocks

bit 0      **EBTR0:** Table Read Protection bit

1 = Block 0 not protected from table reads executed in other blocks

0 = Block 0 protected from table reads executed in other blocks

## REGISTER 24-12: CONFIG7H: CONFIGURATION REGISTER 7 HIGH

U-0	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
—	EBTRB	—	—	—	—	—	—
bit 7						bit 0	

### Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

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

bit 6      **EBTRB:** Boot Block Table Read Protection bit

1 = Boot block not protected from table reads executed in other blocks

0 = Boot block protected from table reads executed in other blocks

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

# PIC18F1XK50/PIC18LF1XK50

## REGISTER 24-13: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F1XK50/PIC18LF1XK50

R	R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	
bit 7								bit 0

### Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

bit 7-5      **DEV<2:0>**: Device ID bits

010 = PIC18F13K50

011 = PIC18F14K50

bit 4-0      **REV<4:0>**: Revision ID bits

These bits are used to indicate the device revision.

## REGISTER 24-14: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F1XK50/PIC18LF1XK50

R	R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	
bit 7								bit 0

### Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

bit 7-0      **DEV<10:3>**: Device ID bits

These bits are used with the DEV<2:0> bits in the Device ID Register 1 to identify the part number.

0010 0000 = PIC18F2XK20/4XK20 devices

**Note 1:** These values for DEV<10:3> may be shared with other devices. The specific device is always identified by using the entire DEV<10:0> bit sequence.

# PIC18F1XK50/PIC18LF1XK50

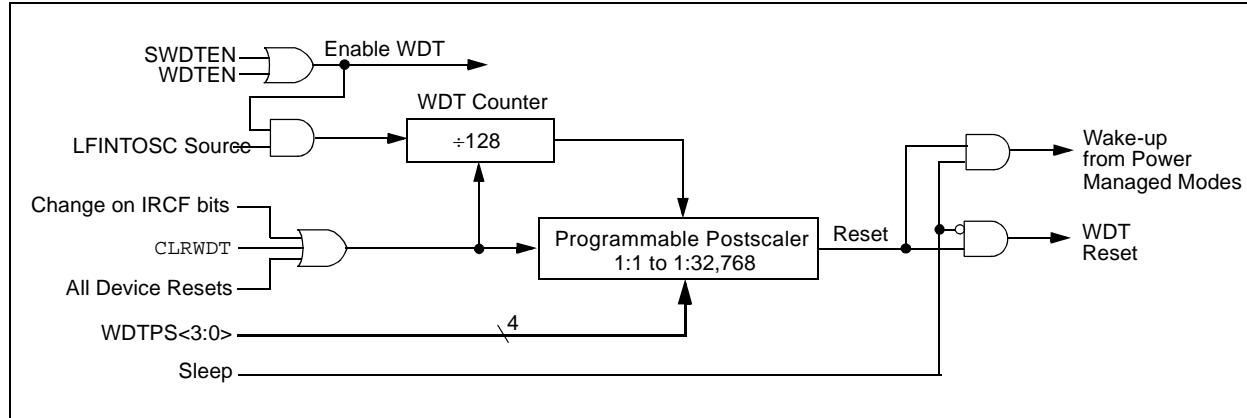
## 24.2 Watchdog Timer (WDT)

For PIC18F1XK50/PIC18LF1XK50 devices, the WDT is driven by the LFINTOSC source. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the LFINTOSC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by bits in Configuration Register 2H. Available periods range from 4 ms to 131.072 seconds (2.18 minutes). The WDT and postscaler are cleared when any of the following events occur: a SLEEP or CLRWDT instruction is executed, the IRCF bits of the OSCCON register are changed or a clock failure has occurred.

- Note 1:** The CLRWDT and SLEEP instructions clear the WDT and postscaler counts when executed.
- 2:** Changing the setting of the IRCF bits of the OSCCON register clears the WDT and postscaler counts.
- 3:** When a CLRWDT instruction is executed, the postscaler count will be cleared.

**FIGURE 24-1: WDT BLOCK DIAGRAM**



## 24.2.1 CONTROL REGISTER

Register 24-15 shows the WDTCON register. This is a readable and writable register which contains a control bit that allows software to override the WDT enable Configuration bit, but only if the Configuration bit has disabled the WDT.

### REGISTER 24-15: WDTCON: WATCHDOG TIMER CONTROL REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN <sup>(1)</sup>
bit 7							bit 0

#### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

bit 0      **SWDTEN:** Software Enable or Disable the Watchdog Timer bit<sup>(1)</sup>

1 = WDT is turned on

0 = WDT is turned off (Reset value)

**Note 1:** This bit has no effect if the Configuration bit, WDTEN, is enabled.

TABLE 24-2: SUMMARY OF WATCHDOG TIMER REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
RCON	IPEN	SBOREN	—	RI	TO	PD	POR	BOR	272
WDTCON	—	—	—	—	—	—	—	SWDTEN	280
CONFIG2H				WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	290

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

## 24.3 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 Flash devices differs significantly from other PIC® microcontroller devices.

The user program memory is divided into five blocks. One of these is a boot block of 0.5K or 2K bytes, depending on the device. The remainder of the memory is divided into individual 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 24-2 shows the program memory organization for 8, 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 24-3.

# PIC18F1XK50/PIC18LF1XK50

FIGURE 24-2: CODE-PROTECTED PROGRAM MEMORY FOR PIC18F1XK50/PIC18LF1XK50

Address (from/to)	Device			
	14K50		13K50	
	BBSIZ = 1	BBSIZ = 0	BBSIZ = 1	BBSIZ = 0
0000h 01FFh	Boot Block, 2 KW CPB, WRTB, EBTRB	Boot Block, 1 KW CPB, WRTB, EBTRB	Boot Block, 1 KW CPB, WRTB, EBTRB	Boot Block, 0.512 KW CPB, WRTB, EBTRB
0200h 03FFh				Block 0 1.512 KW CP0, WRT0, EBTR0
0400h 05FFh		Block 0 3 KW CP0, WRT0, EBTR0	Block 0 1 KW CP0, WRT0, EBTR0	
0600h 07FFh				
0800h 0FFFh	Block 0 2 KW CP0, WRT0, EBTR0		Block 1 2 KW CP1, WRT1, EBTR1	Block 1 2 KW CP1, WRT1, EBTR1
1000h 1FFFh	Block 1 4 KW CP1, WRT1, EBTR1	Block 1 4 KW CP1, WRT1, EBTR1	Reads all '0's	Reads all '0's
2000h 27FFh	Reads all '0's	Reads all '0's		
2800h 2FFFh				
3000h 37FFh				
3800h 3FFFh				
4000h 47FFh				
4800h 4FFFh				
5000h 57FFh				
5800h 5FFFh				
6000h 67FFh				
6800h 6FFFh				
7000h 77FFh				
7800h 7FFFh				
8000h FFFFh				

**Note:** Refer to the test section for requirements on test memory mapping.

TABLE 24-3: SUMMARY OF CODE PROTECTION REGISTERS

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h	CONFIG5L	—	—	—	—	—	CP1	CP0
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—
30000Ah	CONFIG6L	—	—	—	—	—	WRT1	WRT0
30000Bh	CONFIG6H	WRTD	WRTB	WRTE	—	—	—	—
30000Ch	CONFIG7L	—	—	—	—	—	EBTR1	EBTR0
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—

**Legend:** Shaded cells are unimplemented.

**Note 1:** Unimplemented in PIC18FX3K20 and PIC18FX4K20 devices; maintain this bit set.

### 24.3.1 PROGRAM MEMORY CODE PROTECTION

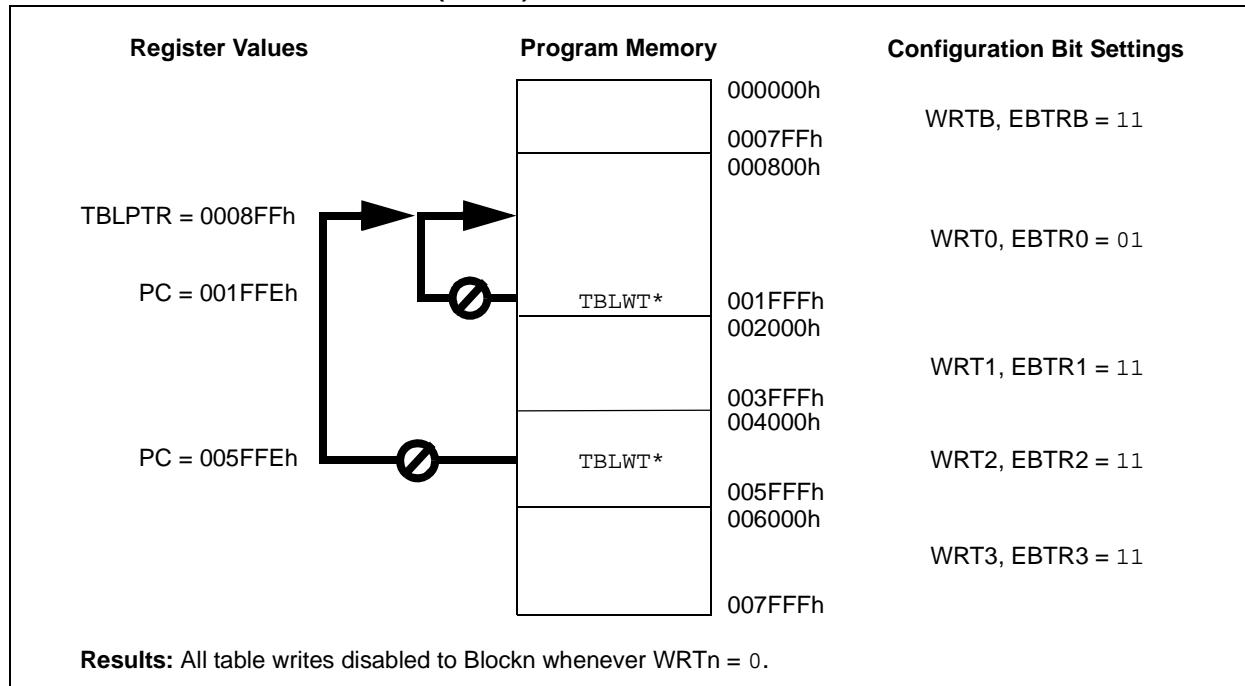
The program memory may be read to or written from any location using the table read and table write instructions. The device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In normal execution mode, the CPn bits have no direct effect. CPn bits inhibit external reads and writes. A block of user memory may be protected from table writes if the WRTn Configuration bit is '0'. The EBTRn bits control table reads. For a block of user memory with the EBTRn bit cleared 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 24-3 through 24-5 illustrate table write and table read protection.

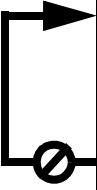
**Note:** Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full chip erase or block erase function. The full chip erase and block erase functions can only be initiated via ICSP or an external programmer.

FIGURE 24-3: TABLE WRITE (WRTn) DISALLOWED



# PIC18F1XK50/PIC18LF1XK50

FIGURE 24-4: EXTERNAL BLOCK TABLE READ (EBTRn) DISALLOWED

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 0008FFh	000000h 0007FFh 000800h	WRTB, EBTRB = 11
PC = 003FFEh		WRT0, EBTR0 = 10
	001FFFh 002000h TBLRD*	WRT1, EBTR1 = 11
	003FFFh 004000h	WRT2, EBTR2 = 11
	005FFFh 006000h	WRT3, EBTR3 = 11
	007FFFh	

**Results:** All table reads from external blocks to Blockn are disabled whenever EBTRn = 0.  
TABLAT register returns a value of '0'.

FIGURE 24-5: EXTERNAL BLOCK TABLE READ (EBTRn) ALLOWED

Register Values	Program Memory	Configuration Bit Settings
TBLPTR = 0008FFh	000000h 0007FFh 000800h	WRTB, EBTRB = 11
PC = 001FFEh		WRT0, EBTR0 = 10
	001FFFh 002000h TBLRD*	WRT1, EBTR1 = 11
	003FFFh 004000h	WRT2, EBTR2 = 11
	005FFFh 006000h	WRT3, EBTR3 = 11
	007FFFh	

**Results:** Table reads permitted within Blockn, even when EBTRBn = 0.  
TABLAT register returns the value of the data at the location TBLPTR.

### 24.3.2 DATA EEPROM CODE PROTECTION

The entire data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of data EEPROM. WRTD inhibits internal and external writes to data EEPROM. The CPU can always read data EEPROM under normal operation, regardless of the protection bit settings.

### 24.3.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In normal execution mode, the WRTC bit is readable only. WRTC can only be written via ICSP or an external programmer.

## 24.4 ID Locations

Eight memory locations (200000h-200007h) are designated as ID locations, where the user can store checksum or other code identification numbers. These locations are both readable and writable during normal execution through the TBLRD and TBLWT instructions or during program/verify. The ID locations can be read when the device is code-protected.

## 24.5 In-Circuit Serial Programming

PIC18F1XK50/PIC18LF1XK50 devices can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

## 24.6 In-Circuit Debugger

When the DEBUG Configuration bit is programmed to a '0', the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some resources are not available for general use. Table 24-4 shows which resources are required by the background debugger.

**TABLE 24-4: DEBUGGER RESOURCES**

I/O pins:	RB6, RB7
Stack:	2 levels
Program Memory:	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 the following pins:

- MCLR/VPP/RE3
- VDD
- Vss
- RB7
- RB6

This will interface to the In-Circuit Debugger module available from Microchip or one of the third party development tool companies.

## 24.7 Single-Supply ICSP Programming

The LVP Configuration bit enables Single-Supply ICSP Programming (formerly known as Low-Voltage ICSP Programming or LVP). When Single-Supply Programming is enabled, the microcontroller can be programmed without requiring high voltage being applied to the MCLR/VPP/RE3 pin, but the RB5/KBI1/PGM pin is then dedicated to controlling Program mode entry and is not available as a general purpose I/O pin.

While programming, using Single-Supply Programming mode, VDD is applied to the MCLR/VPP/RE3 pin as in normal execution mode. To enter Programming mode, VDD is applied to the PGM pin.

**Note 1:** High-voltage programming is always available, regardless of the state of the LVP bit or the PGM pin, by applying VIHH to the MCLR pin.

- 2: By default, Single-Supply ICSP is enabled in unprogrammed devices (as supplied from Microchip) and erased devices.
- 3: When Single-Supply Programming is enabled, the RB5 pin can no longer be used as a general purpose I/O pin.
- 4: When LVP is enabled, externally pull the PGM pin to Vss to allow normal program execution.

If Single-Supply ICSP Programming mode will not be used, the LVP bit can be cleared. RB5/KBI1/PGM then becomes available as the digital I/O pin, RB5. The LVP bit may be set or cleared only when using standard high-voltage programming (VIHH applied to the MCLR/VPP/RE3 pin). Once LVP has been disabled, only the standard high-voltage programming is available and must be used to program the device.

Memory that is not code-protected can be erased using either a block erase, or erased row by row, then written at any specified VDD. If code-protected memory is to be erased, a block erase is required.

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 25.0 INSTRUCTION SET SUMMARY

PIC18F1XK50/PIC18LF1XK50 devices incorporate the standard set of 75 PIC18 core instructions, as well as an extended set of 8 new instructions, for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

### 25.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC® MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are four instructions that require two program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18 instruction set summary in Table 25-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 25-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The destination of the result (specified by 'd')
3. The accessed memory (specified by 'a')

The file register designator 'f' specifies which file register is to be used by the instruction. The destination designator 'd' specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the WREG register. If 'd' is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The bit in the file register (specified by 'b')
3. The accessed memory (specified by 'a')

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by 'k')
- The desired FSR register to load the literal value into (specified by 'f')
- No operand required (specified by '—')

The **control** instructions may use some of the following operands:

- A program memory address (specified by 'n')
- The mode of the CALL or RETURN instructions (specified by 's')
- The mode of the table read and table write instructions (specified by 'm')
- No operand required (specified by '—')

All instructions are a single word, except for four double-word instructions. These instructions were made double-word to contain the required information in 32 bits. In the second word, the 4 MSbs are '1's. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1  $\mu$ s. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2  $\mu$ s. Two-word branch instructions (if true) would take 3  $\mu$ s.

Figure 25-1 shows the general formats that the instructions can have. All examples use the convention 'nnh' to represent a hexadecimal number.

The Instruction Set Summary, shown in Table 25-2, lists the standard instructions recognized by the Microchip Assembler (MPASM™).

**Section 25.1.1 “Standard Instruction Set”** provides a description of each instruction.

# PIC18F1XK50/PIC18LF1XK50

---

TABLE 25-1: OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7).
BSR	Bank Select Register. Used to select the current RAM bank.
C, DC, Z, OV, N	ALU Status bits: <b>C</b> arry, <b>D</b> igit <b>C</b> arry, <b>Z</b> ero, <b>O</b> verflow, <b>N</b> egative.
d	Destination select bit d = 0: store result in WREG d = 1: store result in file register f
dest	Destination: either the WREG register or the specified register file location.
f	8-bit Register file address (00h to FFh) or 2-bit FSR designator (0h to 3h).
f <sub>s</sub>	12-bit Register file address (000h to FFFh). This is the source address.
f <sub>d</sub>	12-bit Register file address (000h to FFFh). This is the destination address.
GIE	Global Interrupt Enable bit.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
label	Label name.
mm	The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions: * No change to register (such as TBLPTR with table reads and writes) *+ Post-Increment register (such as TBLPTR with table reads and writes) *- Post-Decrement register (such as TBLPTR with table reads and writes) +* Pre-Increment register (such as TBLPTR with table reads and writes)
n	The relative address (2's complement number) for relative branch instructions or the direct address for CALL/BRANCH and RETURN instructions.
PC	Program Counter.
PCL	Program Counter Low Byte.
PCH	Program Counter High Byte.
PCLATH	Program Counter High Byte Latch.
PCLATU	Program Counter Upper Byte Latch.
PD	Power-down bit.
PRODH	Product of Multiply High Byte.
PRODL	Product of Multiply Low Byte.
s	Fast Call/Return mode select bit s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
TBLPTR	21-bit Table Pointer (points to a Program Memory location).
TABLAT	8-bit Table Latch.
TO	Time-out bit.
TOS	Top-of-Stack.
u	Unused or unchanged.
WDT	Watchdog Timer.
WREG	Working register (accumulator).
x	Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
z <sub>s</sub>	7-bit offset value for indirect addressing of register files (source).
z <sub>d</sub>	7-bit offset value for indirect addressing of register files (destination).
{ }	Optional argument.
[text]	Indicates an indexed address.
(text)	The contents of text.
[expr]<n>	Specifies bit n of the register indicated by the pointer expr.
→	Assigned to.
< >	Register bit field.
ε	In the set of.
italics	User defined term (font is Courier).

# PIC18F1XK50/PIC18LF1XK50

**FIGURE 25-1: GENERAL FORMAT FOR INSTRUCTIONS**

Byte-oriented file register operations	Example Instruction											
<p>15 10 9 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td>d</td> <td>a</td> <td>f (FILE #)</td> <td>0</td> </tr> </table> <p>d = 0 for result destination to be WREG register  d = 1 for result destination to be file register (f)  a = 0 to force Access Bank  a = 1 for BSR to select bank  f = 8-bit file register address</p>	OPCODE	d	a	f (FILE #)	0	ADDWF MYREG, W, B						
OPCODE	d	a	f (FILE #)	0								
<b>Byte to Byte move operations (2-word)</b>												
<p>15 12 11 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">f (Source FILE #)</td> <td>0</td> </tr> </table> <p>15 12 11 0</p> <table border="1"> <tr> <td>1111</td> <td colspan="3">f (Destination FILE #)</td> <td>0</td> </tr> </table> <p>f = 12-bit file register address</p>	OPCODE	f (Source FILE #)			0	1111	f (Destination FILE #)			0	MOVFF MYREG1, MYREG2	
OPCODE	f (Source FILE #)			0								
1111	f (Destination FILE #)			0								
<b>Bit-oriented file register operations</b>												
<p>15 12 11 9 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td>b (BIT #)</td> <td>a</td> <td>f (FILE #)</td> <td>0</td> </tr> </table> <p>b = 3-bit position of bit in file register (f)  a = 0 to force Access Bank  a = 1 for BSR to select bank  f = 8-bit file register address</p>	OPCODE	b (BIT #)	a	f (FILE #)	0	BSF MYREG, bit, B						
OPCODE	b (BIT #)	a	f (FILE #)	0								
<b>Literal operations</b>												
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">k (literal)</td> <td>0</td> </tr> </table> <p>k = 8-bit immediate value</p>	OPCODE	k (literal)			0	MOVLW 7Fh						
OPCODE	k (literal)			0								
<b>Control operations</b>												
<b>CALL, GOTO and Branch operations</b>												
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="3">n&lt;7:0&gt; (literal)</td> <td>0</td> </tr> </table> <p>15 12 11 0</p> <table border="1"> <tr> <td>1111</td> <td colspan="3">n&lt;19:8&gt; (literal)</td> <td>0</td> </tr> </table> <p>n = 20-bit immediate value</p>	OPCODE	n<7:0> (literal)			0	1111	n<19:8> (literal)			0	GOTO Label	
OPCODE	n<7:0> (literal)			0								
1111	n<19:8> (literal)			0								
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td>S</td> <td colspan="3">n&lt;7:0&gt; (literal)</td> <td>0</td> </tr> </table> <p>15 12 11 0</p> <table border="1"> <tr> <td>1111</td> <td colspan="3">n&lt;19:8&gt; (literal)</td> <td>0</td> </tr> </table> <p>S = Fast bit</p>	OPCODE	S	n<7:0> (literal)			0	1111	n<19:8> (literal)			0	CALL MYFUNC
OPCODE	S	n<7:0> (literal)			0							
1111	n<19:8> (literal)			0								
<p>15 11 10 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="4">n&lt;10:0&gt; (literal)</td> <td>0</td> </tr> </table>	OPCODE	n<10:0> (literal)				0	BRA MYFUNC					
OPCODE	n<10:0> (literal)				0							
<p>15 8 7 0</p> <table border="1"> <tr> <td>OPCODE</td> <td colspan="4">n&lt;7:0&gt; (literal)</td> <td>0</td> </tr> </table>	OPCODE	n<7:0> (literal)				0	BC MYFUNC					
OPCODE	n<7:0> (literal)				0							

# PIC18F1XK50/PIC18LF1XK50

TABLE 25-2: PIC18FXXXX INSTRUCTION SET

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
			MSb	Lsb				
<b>BYTE-ORIENTED OPERATIONS</b>								
ADDWF f, d, a	Add WREG and f	1	0010 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		None	
			1111 ffff	ffff	ffff			
MOVWF f, a	Move WREG to f	1	0110 111a	ffff	ffff		None	
MULWF f, a	Multiply WREG with f	1	0000 001a	ffff	ffff		None	1, 2
NEGF f, a	Negate f	1	0110 110a	ffff	ffff		C, DC, Z, OV, N	
RLCF f, d, a	Rotate Left f through Carry	1	0011 01da	ffff	ffff		C, Z, N	1, 2
RLNCF f, d, a	Rotate Left f (No Carry)	1	0100 01da	ffff	ffff		Z, N	
RRCF f, d, a	Rotate Right f through Carry	1	0011 00da	ffff	ffff		C, Z, N	
RRNCF f, d, a	Rotate Right f (No Carry)	1	0100 00da	ffff	ffff		Z, N	
SETf f, a	Set f	1	0110 100a	ffff	ffff		None	1, 2
SUBFWB f, d, a	Subtract f from WREG with borrow	1	0101 01da	ffff	ffff		C, DC, Z, OV, N	
SUBWF f, d, a	Subtract WREG from f	1	0101 11da	ffff	ffff		C, DC, Z, OV, N	1, 2
SUBWFB f, d, a	Subtract WREG from f with borrow	1	0101 10da	ffff	ffff		C, DC, Z, OV, N	
SWAPF f, d, a	Swap nibbles in f	1	0011 10da	ffff	ffff		None	4
TSTFSZ f, a	Test f, skip if 0	1 (2 or 3)	0110 011a	ffff	ffff		None	1, 2
XORWF f, d, a	Exclusive OR WREG with f	1	0001 10da	ffff	ffff		Z, N	

**Note 1:** When a PORT register is modified as a function of itself (e.g., `MOVf PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

- 2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.
- 4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

# PIC18F1XK50/PIC18LF1XK50

TABLE 25-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes	
			MSb	LSb					
<b>BIT-ORIENTED OPERATIONS</b>									
BCF f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2	
BSF f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2	
BTFSC f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4	
BTFSS f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4	
BTG f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2	
<b>CONTROL OPERATIONS</b>									
BC n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None		
BN n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None		
BNC n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None		
BNN n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None		
BNOV n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None		
BNZ n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None		
BOV n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None		
BRA n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None		
BZ n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None		
CALL n, s	Call subroutine 1st word	2	1110	110s	kkkk	kkkk	None		
	2nd word		1111	kkkk	kkkk	kkkk			
CLRWDT —	Clear Watchdog Timer	1	0000	0000	0000	0100	TO, PD		
DAW —	Decimal Adjust WREG	1	0000	0000	0000	0111	C		
GOTO n	Go to address 1st word	2	1110	1111	kkkk	kkkk	None		
	2nd word		1111	kkkk	kkkk	kkkk			
NOP —	No Operation	1	0000	0000	0000	0000	None		
NOP —	No Operation	1	1111	xxxx	xxxx	xxxx	None	4	
POP —	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None		
PUSH —	Push top of return stack (TOS)	1	0000	0000	0000	0101	None		
RCALL n	Relative Call	2	1101	1nnn	nnnn	nnnn	None		
RESET	Software device Reset	1	0000	0000	1111	1111	All		
RETFIE s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL		
RETLW k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None		
RETURN s	Return from Subroutine	2	0000	0000	0001	001s	None		
SLEEP —	Go into Standby mode	1	0000	0000	0000	0011	TO, PD		

- Note 1:** When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

# PIC18F1XK50/PIC18LF1XK50

TABLE 25-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
			MSb	LSb				
<b>LITERAL OPERATIONS</b>								
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N
LFSR	f, k	Move literal (12-bit) 2nd word to FSR(f) 1st word	2	1110	1110	0fff	kkkk	None
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N
<b>DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS</b>								
TBLRD*		Table Read	2	0000	0000	0000	1000	None
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None
TBLWT*		Table Write		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 two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

# PIC18F1XK50/PIC18LF1XK50

## 25.1.1 STANDARD INSTRUCTION SET

ADDLW	ADD literal to W			
Syntax:	ADDLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	(W) + k → W			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0000	1111	kkkk	kkkk
Description:	The contents of W are added to the 8-bit literal 'k' and the result is placed in W.			
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'	Process Data	Write to W

Example: ADDLW 15h

Before Instruction

W = 10h

After Instruction

W = 25h

ADDWF	ADD W to f			
Syntax:	ADDWF f {,d {,a}}			
Operands:	0 ≤ f ≤ 255			
	d ∈ [0,1]			
	a ∈ [0,1]			
Operation:	(W) + (f) → 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 is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.			

Words: 1  
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: ADDWF REG, 0, 0

Before Instruction

W = 17h  
REG = 0C2h

After Instruction

W = 0D9h  
REG = 0C2h

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

# PIC18F1XK50/PIC18LF1XK50

---



---

ADDWFC	ADD W and CARRY bit to f								
Syntax:	ADDWFC f {,d {,a}}								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(W) + (f) + (C) \rightarrow \text{dest}$								
Status Affected:	N,OV, C, DC, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0010</td> <td>00da</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0010	00da	ffff	ffff				
0010	00da	ffff	ffff						
Description:	<p>Add W, the CARRY flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to W</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to W						

Example: ADDWFC REG, 0, 1

Before Instruction

CARRY bit = 1  
 REG = 02h  
 W = 4Dh

After Instruction

CARRY bit = 0  
 REG = 02h  
 W = 50h

ANDLW	AND literal with W								
Syntax:	ANDLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	$(W) .AND. k \rightarrow W$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>1011</td> <td>kkkk</td> <td>kkkk</td> </tr> </table>	0000	1011	kkkk	kkkk				
0000	1011	kkkk	kkkk						
Description:	The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'k'</td> <td>Process Data</td> <td>Write to W</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read literal 'k'	Process Data	Write to W						

Example: ANDLW 05Fh

Before Instruction

W = A3h

After Instruction

W = 03h

# PIC18F1XK50/PIC18LF1XK50

ANDWF	AND W with f								
Syntax:	ANDWF f {,d {,a}}								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(W) .AND. (f) \rightarrow \text{dest}$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0001</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>	0001	01da	ffff	ffff				
0001	01da	ffff	ffff						
Description:	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 is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><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: ANDWF REG, 0, 0

Before Instruction

W = 17h  
REG = C2h

After Instruction

W = 02h  
REG = C2h

BC	Branch if Carry												
Syntax:	BC n												
Operands:	$-128 \leq n \leq 127$												
Operation:	if CARRY bit is '1' $(PC) + 2 + 2n \rightarrow PC$												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>0010</td><td>nnnn</td><td>nnnn</td></tr></table>	1110	0010	nnnn	nnnn								
1110	0010	nnnn	nnnn										
Description:	If the CARRY bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC + 2 + 2n$ . This instruction is then a two-cycle instruction.												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:	If Jump: <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	No operation										

Example: HERE BC 5

Before Instruction

PC = address (HERE)

After Instruction

If CARRY = 1;  
PC = address (HERE + 12)  
If CARRY = 0;  
PC = address (HERE + 2)

# PIC18F1XK50/PIC18LF1XK50

---



---

BCF		Bit Clear f					
Syntax:	BCF f, b {,a}						
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]						
Operation:	0 → f<b>						
Status Affected:	None						
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1001</td><td>bbba</td><td>ffff</td><td>ffff</td></tr></table>	1001	bbba	ffff	ffff		
1001	bbba	ffff	ffff				
Description:	Bit 'b' in register 'f' is cleared. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.						
Words:	1						
Cycles:	1						
Q Cycle Activity:							
	Q1	Q2	Q3	Q4			
	Decode	Read register 'f'	Process Data	Write register 'f'			

Example: BCF FLAG\_REG, 7, 0

Before Instruction  
FLAG\_REG = C7h  
After Instruction  
FLAG\_REG = 47h

BN		Branch if Negative					
Syntax:	BN n						
Operands:	-128 ≤ n ≤ 127						
Operation:	if NEGATIVE bit is '1' (PC) + 2 + 2n → PC						
Status Affected:	None						
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>0110</td><td>nnnn</td><td>nnnn</td></tr></table>	1110	0110	nnnn	nnnn		
1110	0110	nnnn	nnnn				
Description:	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)

# PIC18F1XK50/PIC18LF1XK50

BNC	Branch if Not Carry															
Syntax:	BNC n															
Operands:	-128 ≤ n ≤ 127															
Operation:	if CARRY bit is '0' (PC) + 2 + 2n → PC															
Status Affected:	None															
Encoding:	1110	0011	nnnn	nnnn												
Description:	If the CARRY bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.															
Words:	1															
Cycles:	1(2)															
Q Cycle Activity:																
If Jump:	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	Write to PC													
No operation	No operation	No operation	No operation													
If No Jump:	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	No operation													

Example: HERE BNC Jump

Before Instruction  
 PC = address (HERE)

After Instruction  
 If CARRY = 0;  
 PC = address (Jump)  
 If CARRY = 1;  
 PC = address (HERE + 2)

BNN	Branch if Not Negative															
Syntax:	BNN n															
Operands:	-128 ≤ n ≤ 127															
Operation:	if NEGATIVE bit is '0' (PC) + 2 + 2n → PC															
Status Affected:	None															
Encoding:	1110	0111	nnnn	nnnn												
Description:	If the NEGATIVE bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.															
Words:	1															
Cycles:	1(2)															
Q Cycle Activity:																
If Jump:	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	Write to PC													
No operation	No operation	No operation	No operation													
If No Jump:	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4													
Decode	Read literal 'n'	Process Data	No operation													

Example: HERE BNN Jump

Before Instruction  
 PC = address (HERE)

After Instruction  
 If NEGATIVE = 0;  
 PC = address (Jump)  
 If NEGATIVE = 1;  
 PC = address (HERE + 2)

# PIC18F1XK50/PIC18LF1XK50

---



---

<b>BNOV</b> <b>Branch if Not Overflow</b>													
Syntax:	BNOV n												
Operands:	$-128 \leq n \leq 127$												
Operation:	if OVERFLOW bit is '0' (PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>0101</td><td>nnnn</td><td>nnnn</td></tr></table>	1110	0101	nnnn	nnnn								
1110	0101	nnnn	nnnn										
Description:	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:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	No operation										

Example:      HERE      BNOV Jump

Before Instruction  
 $PC = \text{address (HERE)}$   
 After Instruction  
 If OVERFLOW = 0;  
 $PC = \text{address (Jump)}$   
 If OVERFLOW = 1;  
 $PC = \text{address (HERE + 2)}$

<b>BNZ</b> <b>Branch if Not Zero</b>													
Syntax:	BNZ n												
Operands:	$-128 \leq n \leq 127$												
Operation:	if ZERO bit is '0' (PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>0001</td><td>nnnn</td><td>nnnn</td></tr></table>	1110	0001	nnnn	nnnn								
1110	0001	nnnn	nnnn										
Description:	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:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	No operation										

Example:      HERE      BNZ Jump

Before Instruction  
 $PC = \text{address (HERE)}$   
 After Instruction  
 If ZERO = 0;  
 $PC = \text{address (Jump)}$   
 If ZERO = 1;  
 $PC = \text{address (HERE + 2)}$

# PIC18F1XK50/PIC18LF1XK50

BRA	Unconditional Branch															
Syntax:	BRA n															
Operands:	-1024 ≤ n ≤ 1023															
Operation:	(PC) + 2 + 2n → PC															
Status Affected:	None															
Encoding:	1101	0nnn	nnnn	nnnn												
Description:	Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction.															
Words:	1															
Cycles:	2															
Q Cycle Activity:	<table border="1" style="margin-left: auto; margin-right: auto;"> <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													

Example:      HERE      BRA      Jump

Before Instruction  
 PC                    =      address (HERE)

After Instruction  
 PC                    =      address (Jump)

BSF	Bit Set f											
Syntax:	BSF f, b {,a}											
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]											
Operation:	1 → f<b>											
Status Affected:	None											
Encoding:	1000	bbba	ffff	ffff								
Description:	<p>Bit 'b' in register 'f' is set.            If 'a' is '0', the Access Bank is selected.            If 'a' is '1', the BSR is used to select the GPR bank (default).            If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="margin-left: auto; margin-right: auto;"> <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:      BSF      FLAG\_REG, 7, 1

Before Instruction  
 FLAG\_REG            =      0Ah

After Instruction  
 FLAG\_REG            =      8Ah

# PIC18F1XK50/PIC18LF1XK50

BTFS <sub>C</sub>		Bit Test File, Skip if Clear	
Syntax:	BTFS <sub>C</sub> f, b {,a}		
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]		
Operation:	skip if (f<b>) = 0		
Status Affected:	None		
Encoding:	1011 bbba ffff ffff		
Description:	If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.		
Words:	1		
Cycles:	1(2)		
<b>Note:</b>	3 cycles if skip and followed by a 2-word instruction.		

## Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE BTFSC FLAG, 1, 0  
FALSE :  
TRUE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;  
PC = address (TRUE)  
If FLAG<1> = 1;  
PC = address (FALSE)

BTFS <sub>S</sub>		Bit Test File, Skip if Set	
Syntax:	BTFS <sub>S</sub> f, b {,a}		
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]		
Operation:	skip if (f<b>) = 1		
Status Affected:	None		
Encoding:	1010 bbba ffff ffff		
Description:	If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.		

Words:

1

Cycles:

1(2)

**Note:** 3 cycles if skip and followed by a 2-word instruction.

## Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE BTFS<sub>S</sub> FLAG, 1, 0  
FALSE :  
TRUE :

Before Instruction

PC = address (HERE)

After Instruction

If FLAG<1> = 0;  
PC = address (FALSE)  
If FLAG<1> = 1;  
PC = address (TRUE)

BTG	Bit Toggle f								
Syntax:	BTG f, b {,a}								
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]								
Operation:	$(\overline{f<b>}) \rightarrow f<b>$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0111</td> <td>bbba</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0111	bbba	ffff	ffff				
0111	bbba	ffff	ffff						
Description:	<p>Bit 'b' in data memory location 'f' is inverted.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write register 'f'</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write register 'f'						

Example: BTG PORTC, 4, 0

Before Instruction:

PORTC = 0111 0101 [75h]

After Instruction:

PORTC = 0110 0101 [65h]

BOV	Branch if Overflow												
Syntax:	BOV n												
Operands:	-128 ≤ n ≤ 127												
Operation:	if OVERFLOW bit is '1' (PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1110</td> <td>0100</td> <td>nnnn</td> <td>nnnn</td> </tr> </table>	1110	0100	nnnn	nnnn								
1110	0100	nnnn	nnnn										
Description:	<p>If the OVERFLOW bit is '1', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.</p>												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>Write to PC</td> </tr> <tr> <td>No operation</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>No operation</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	No operation										

Example: HERE BOV Jump

Before Instruction

PC = address (HERE)

After Instruction

If OVERFLOW = 1;  
PC = address (Jump)  
If OVERFLOW = 0;  
PC = address (HERE + 2)

# PIC18F1XK50/PIC18LF1XK50

<b>BZ</b> Branch if Zero													
Syntax:	BZ n												
Operands:	$-128 \leq n \leq 127$												
Operation:	if ZERO bit is '1' (PC) + 2 + 2n → PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0000</td><td>nnnn</td><td>nnnn</td></tr> </table>	1110	0000	nnnn	nnnn								
1110	0000	nnnn	nnnn										
Description:	<p>If the ZERO bit is '1', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.</p>												
Words:	1												
Cycles:	1(2)												
Q Cycle Activity:													
If Jump:													
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	Write to PC										
No operation	No operation	No operation	No operation										
If No Jump:													
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n'	Process Data	No operation				
Q1	Q2	Q3	Q4										
Decode	Read literal 'n'	Process Data	No operation										

Example: HERE BZ Jump

Before Instruction  
 $PC = \text{address (HERE)}$

After Instruction  
 $\begin{array}{ll} \text{If ZERO} & = 1; \\ \text{PC} & = \text{address (Jump)} \\ \text{If ZERO} & = 0; \\ \text{PC} & = \text{address (HERE + 2)} \end{array}$

<b>CALL</b> Subroutine Call													
Syntax:	CALL k {s}												
Operands:	$0 \leq k \leq 1048575$ $s \in [0,1]$												
Operation:	$(PC) + 4 \rightarrow TOS$ , $k \rightarrow PC<20:1>$ , $\text{if } s = 1$ $(W) \rightarrow WS$ , $(\text{Status}) \rightarrow \text{STATUS}$ , $(\text{BSR}) \rightarrow \text{BSRS}$												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>110s</td><td>k<sub>7</sub>kkk</td><td>kkkk<sub>0</sub></td></tr> <tr><td>1111</td><td>k<sub>19</sub>kkk</td><td>kkkk</td><td>kkkk<sub>8</sub></td></tr> </table>	1110	110s	k <sub>7</sub> kkk	kkkk <sub>0</sub>	1111	k <sub>19</sub> kkk	kkkk	kkkk <sub>8</sub>				
1110	110s	k <sub>7</sub> kkk	kkkk <sub>0</sub>										
1111	k <sub>19</sub> kkk	kkkk	kkkk <sub>8</sub>										
Description:	<p>Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC&lt;20:1&gt;. CALL is a two-cycle instruction.</p>												
Words:	2												
Cycles:	2												
Q Cycle Activity:													
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr><td>Decode</td><td>Read literal 'k'&lt;7:0&gt;,</td><td>PUSH PC to stack</td><td>Read literal 'k'&lt;19:8&gt;, Write to PC</td></tr> <tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'<7:0>,	PUSH PC to stack	Read literal 'k'<19:8>, Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'k'<7:0>,	PUSH PC to stack	Read literal 'k'<19:8>, Write to PC										
No operation	No operation	No operation	No operation										

Example: HERE CALL THERE, 1

Before Instruction  
 $PC = \text{address (HERE)}$

After Instruction  
 $\begin{array}{ll} \text{PC} & = \text{address (THERE)} \\ \text{TOS} & = \text{address (HERE + 4)} \\ \text{WS} & = \text{W} \\ \text{BSRS} & = \text{BSR} \\ \text{STATUS} & = \text{Status} \end{array}$

CLRF	Clear f								
Syntax:	CLRF f {,a}								
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$								
Operation:	$000h \rightarrow f$ $1 \rightarrow Z$								
Status Affected:	Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0110</td><td>101a</td><td>ffff</td><td>ffff</td></tr></table>	0110	101a	ffff	ffff				
0110	101a	ffff	ffff						
Description:	Clears the contents of the specified register. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write register 'f'</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write register 'f'						

Example: CLRF FLAG\_REG, 1

Before Instruction  
FLAG\_REG = 5Ah  
After Instruction  
FLAG\_REG = 00h

CLRWDT	Clear Watchdog Timer								
Syntax:	CLRWDT								
Operands:	None								
Operation:	$000h \rightarrow \text{WDT}$ , $000h \rightarrow \text{WDT postscaler}$ , $1 \rightarrow \overline{\text{TO}}$ , $1 \rightarrow \overline{\text{PD}}$								
Status Affected:	$\overline{\text{TO}}, \overline{\text{PD}}$								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>0000</td><td>0000</td><td>0100</td></tr></table>	0000	0000	0000	0100				
0000	0000	0000	0100						
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{\text{TO}}$ and $\overline{\text{PD}}$ , are set.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>No operation</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	No operation	Process Data	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	Process Data	No operation						

Example: CLRWDT

Before Instruction	
WDT Counter	= ?
After Instruction	
WDT Counter	= 00h
WDT Postscaler	= 0
$\overline{\text{TO}}$	= 1
PD	= 1

# PIC18F1XK50/PIC18LF1XK50

COMF	Complement f								
Syntax:	COMF f {,d {,a}}								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(\bar{f}) \rightarrow \text{dest}$								
Status Affected:	N, Z								
Encoding:	0001 11da ffff ffff								
Description:	The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: COMF REG, 0, 0

Before Instruction  
REG = 13h  
After Instruction  
REG = 13h  
W = ECh

CPFSEQ	Compare f with W, skip if f = W
Syntax:	CPFSEQ f {,a}
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$
Operation:	$(f) - (W)$ , skip if $(f) = (W)$ (unsigned comparison)
Status Affected:	None
Encoding:	0110 001a ffff ffff
Description:	Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.
Words:	1
Cycles:	1(2)
<b>Note:</b>	3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE CPFSEQ REG, 0  
NEQUAL :  
EQUAL :

Before Instruction

PC Address = HERE  
W = ?  
REG = ?

After Instruction

If REG = W;  
PC = Address (EQUAL)  
If REG ≠ W;  
PC = Address (NEQUAL)

CPFSGT	Compare f with W, skip if f > W	CPFSLT	Compare f with W, skip if f < W																
Syntax:	CPFSGT f {,a}	Syntax:	CPFSLT f {,a}																
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$	Operands:	$0 \leq f \leq 255$ $a \in [0,1]$																
Operation:	$(f) - (W)$ , skip if $(f) > (W)$ (unsigned comparison)	Operation:	$(f) - (W)$ , skip if $(f) < (W)$ (unsigned comparison)																
Status Affected:	None	Status Affected:	None																
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0110</td><td>010a</td><td>ffff</td><td>ffff</td></tr></table>	0110	010a	ffff	ffff	Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0110</td><td>000a</td><td>ffff</td><td>ffff</td></tr></table>	0110	000a	ffff	ffff								
0110	010a	ffff	ffff																
0110	000a	ffff	ffff																
Description:	Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction. If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.	Description:	Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).																
Words:	1	Words:	1																
Cycles:	1(2)	Cycles:	1(2)																
	<b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.		<b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.																
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	No operation	Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	No operation
Q1	Q2	Q3	Q4																
Decode	Read register 'f'	Process Data	No operation																
Q1	Q2	Q3	Q4																
Decode	Read register 'f'	Process Data	No operation																
If skip:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	If skip:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4																
No operation	No operation	No operation	No operation																
Q1	Q2	Q3	Q4																
No operation	No operation	No operation	No operation																
If skip and followed by 2-word instruction:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation	If skip and followed by 2-word instruction:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4																
No operation	No operation	No operation	No operation																
Q1	Q2	Q3	Q4																
No operation	No operation	No operation	No operation																
Example:	HERE CPFSGT REG, 0 NGREATER : GREATER :	Example:	HERE CPFSLT REG, 1 NLESS : LESS :																
Before Instruction	PC = Address (HERE) W = ?	Before Instruction	PC = Address (HERE) W = ?																
After Instruction	If REG > W; PC = Address (GREATER) If REG $\leq$ W; PC = Address (NGREATER)	After Instruction	If REG < W; PC = Address (LESS) If REG $\geq$ W; PC = Address (NLESS)																

# PIC18F1XK50/PIC18LF1XK50

DAW	Decimal Adjust W Register								
Syntax:	DAW								
Operands:	None								
Operation:	<p>If <math>[W&lt;3:0&gt; &gt; 9]</math> or <math>[DC = 1]</math> then  <math>(W&lt;3:0&gt;) + 6 \rightarrow W&lt;3:0&gt;;</math>  else  <math>(W&lt;3:0&gt;) \rightarrow W&lt;3:0&gt;;</math></p> <p>If <math>[W&lt;7:4&gt; + DC &gt; 9]</math> or <math>[C = 1]</math> then  <math>(W&lt;7:4&gt;) + 6 + DC \rightarrow W&lt;7:4&gt;;</math>  else  <math>(W&lt;7:4&gt;) + DC \rightarrow W&lt;7:4&gt;;</math></p>								
Status Affected:	C								
Encoding:	<table border="1"> <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"> <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	=	A5h
C	=	0
DC	=	0

##### After Instruction

W	=	05h
C	=	1
DC	=	0

#### Example 2:

##### Before Instruction

W	=	CEh
C	=	0
DC	=	0

##### After Instruction

W	=	34h
C	=	1
DC	=	0

DECF	Decrement f				
Syntax:	DECF f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(f) - 1 \rightarrow \text{dest}$				
Status Affected:	C, DC, N, OV, Z				
Encoding:	<table border="1"> <tr> <td>0000</td> <td>01da</td> <td>ffff</td> <td>ffff</td> </tr> </table>	0000	01da	ffff	ffff
0000	01da	ffff	ffff		
Description:	<p>Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>				

Words: 1  
Cycles: 1

##### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

#### Example:

DECF CNT, 1, 0

##### Before Instruction

CNT	=	01h
Z	=	0

##### After Instruction

CNT	=	00h
Z	=	1

# PIC18F1XK50/PIC18LF1XK50

DECFSZ	Decrement f, skip if 0				
Syntax:	DECFSZ f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(f) - 1 \rightarrow \text{dest}$ , skip if result = 0				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0010</td><td>11da</td><td>ffff</td><td>ffff</td></tr></table>	0010	11da	ffff	ffff
0010	11da	ffff	ffff		
Description:	The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.				
Words:	1				
Cycles:	1(2)				
<b>Note:</b>	3 cycles if skip and followed by a 2-word instruction.				

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE DECFSZ CNT, 1, 1  
GOTO LOOP  
CONTINUE

#### Before Instruction

PC = Address (HERE)

#### After Instruction

CNT = CNT - 1

If CNT = 0;

PC = Address (CONTINUE)

If CNT ≠ 0;

PC = Address (HERE + 2)

DCFSNZ	Decrement f, skip if not 0				
Syntax:	DCFSNZ f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(f) - 1 \rightarrow \text{dest}$ , skip if result ≠ 0				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0100</td><td>11da</td><td>ffff</td><td>ffff</td></tr></table>	0100	11da	ffff	ffff
0100	11da	ffff	ffff		
Description:	The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.				
Words:	1				
Cycles:	1(2)				
<b>Note:</b>	3 cycles if skip and followed by a 2-word instruction.				

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	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)

# PIC18F1XK50/PIC18LF1XK50

GOTO	Unconditional Branch	INCF	Increment f
Syntax:	GOTO k	Syntax:	INCF f {,d {,a}}
Operands:	0 ≤ k ≤ 1048575	Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	k → PC<20:1>	Operation:	(f) + 1 → dest
Status Affected:	None	Status Affected:	C, DC, N, OV, Z
Encoding:		Encoding:	0010 10da ffff ffff
1st word (k<7:0>)	1110 1111 k <sub>7</sub> kkk kkkk <sub>0</sub>	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).
2nd word(k<19:8>)	1111 k <sub>19</sub> kkk kkkk kkkk <sub>8</sub>		If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
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.		If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.
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

Example: GOTO THERE  
 After Instruction  
 PC = Address (THERE)

<u>Example:</u>	INCF	CNT, 1, 0
Before Instruction		
CNT	=	FFh
Z	=	0
C	=	?
DC	=	?
After Instruction		
CNT	=	00h
Z	=	1
C	=	1
DC	=	1

INCFSZ	Increment f, skip if 0				
Syntax:	INCFSZ f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(f) + 1 \rightarrow \text{dest}$ , skip if result = 0				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0011</td><td>11da</td><td>ffff</td><td>ffff</td></tr></table>	0011	11da	ffff	ffff
0011	11da	ffff	ffff		
Description:	<p>The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>				
Words:	1				
Cycles:	1(2)				
<b>Note:</b>	3 cycles if skip and followed by a 2-word instruction.				

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE INCFSZ CNT, 1, 0  
NZERO :  
ZERO :

Before Instruction

PC = Address (HERE)

After Instruction

CNT = CNT + 1  
If CNT = 0;  
PC = Address (ZERO)  
If CNT ≠ 0;  
PC = Address (NZERO)

INFSNZ	Increment f, skip if not 0				
Syntax:	INFSNZ f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(f) + 1 \rightarrow \text{dest}$ , skip if result ≠ 0				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table;"><tr><td>0100</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0100	10da	ffff	ffff
0100	10da	ffff	ffff		
Description:	<p>The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>				
Words:	1				
Cycles:	1(2)				
<b>Note:</b>	3 cycles if skip and followed by a 2-word instruction.				

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE INFSNZ REG, 1, 0  
ZERO :  
NZERO :

Before Instruction

PC = Address (HERE)

After Instruction

REG = REG + 1  
If REG ≠ 0;  
PC = Address (NZERO)  
If REG = 0;  
PC = Address (ZERO)

# PIC18F1XK50/PIC18LF1XK50

---



---

IORLW      Inclusive OR literal with W									
Syntax:	IORLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	$(W) .OR. k \rightarrow W$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td><td>1001</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1001	kkkk	kkkk				
0000	1001	kkkk	kkkk						
Description:	The contents of W are ORed with the eight-bit literal 'k'. The result is placed in W.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="text-align: center; width: 25%;">Q1</th><th style="text-align: center; width: 25%;">Q2</th><th style="text-align: center; width: 25%;">Q3</th><th style="text-align: center; width: 25%;">Q4</th></tr> <tr> <td style="text-align: center;">Decode</td><td style="text-align: center;">Read literal 'k'</td><td style="text-align: center;">Process Data</td><td style="text-align: center;">Write to W</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read literal 'k'	Process Data	Write to W						

Example:      IORLW      35h

Before Instruction

W = 9Ah

After Instruction

W = BFh

IORWF      Inclusive OR W with f									
Syntax:	IORWF f {,d {,a}}								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(W) .OR. (f) \rightarrow \text{dest}$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0001</td><td>00da</td><td>ffff</td><td>ffff</td></tr> </table>	0001	00da	ffff	ffff				
0001	00da	ffff	ffff						
Description:	<p>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).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="text-align: center; width: 25%;">Q1</th><th style="text-align: center; width: 25%;">Q2</th><th style="text-align: center; width: 25%;">Q3</th><th style="text-align: center; width: 25%;">Q4</th></tr> <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> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example:      IORWF      RESULT, 0, 1

Before Instruction

RESULT = 13h

W = 91h

After Instruction

RESULT = 13h

W = 93h

# PIC18F1XK50/PIC18LF1XK50

LFSR	Load FSR															
Syntax:	LFSR f, k															
Operands:	0 ≤ f ≤ 2 0 ≤ k ≤ 4095															
Operation:	k → FSRf															
Status Affected:	None															
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>1110</td><td>00ff</td><td>k<sub>11</sub>kkk</td></tr> <tr><td>1111</td><td>0000</td><td>k<sub>7</sub>kkk</td><td>kkkk</td></tr> </table>				1110	1110	00ff	k <sub>11</sub> kkk	1111	0000	k <sub>7</sub> kkk	kkkk				
1110	1110	00ff	k <sub>11</sub> kkk													
1111	0000	k <sub>7</sub> kkk	kkkk													
Description:	The 12-bit literal 'k' is loaded into the File Select Register pointed to by 'f'.															
Words:	2															
Cycles:	2															
Q Cycle Activity:	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'k' MSB</td><td>Process Data</td><td>Write literal 'k' MSB to FSRfH</td></tr> <tr> <td>Decode</td><td>Read literal 'k' LSB</td><td>Process Data</td><td>Write literal 'k' to FSRfL</td></tr> </tbody> </table>				Q1	Q2	Q3	Q4	Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH	Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL
Q1	Q2	Q3	Q4													
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH													
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL													

Example: LFSR 2, 3ABh

After Instruction

FSR2H	=	03h
FSR2L	=	ABh

MOVF	Move f				
Syntax:	MOVF f {,d {,a}}				
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]				
Operation:	f → dest				
Status Affected:	N, Z				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0101</td><td>00da</td><td>ffff</td><td>ffff</td></tr> </table>	0101	00da	ffff	ffff
0101	00da	ffff	ffff		
Description:	The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256-byte bank.				
	If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).				
	If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.				

Words: 1  
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write W

Example: MOVF REG, 0, 0

Before Instruction

REG	=	22h
W	=	FFh

After Instruction

REG	=	22h
W	=	22h

# PIC18F1XK50/PIC18LF1XK50

---

## MOVFF

### Move f to f

Syntax:	MOVFF f <sub>s</sub> , f <sub>d</sub>
Operands:	0 ≤ f <sub>s</sub> ≤ 4095 0 ≤ f <sub>d</sub> ≤ 4095
Operation:	(f <sub>s</sub> ) → f <sub>d</sub>
Status Affected:	None
Encoding:	
1st word (source)	1100 ffff ffff ffff <sub>s</sub>
2nd word (destin.)	1111 ffff ffff ffff <sub>d</sub>
Description:	<p>The contents of source register 'f<sub>s</sub>' are moved to destination register 'f<sub>d</sub>'. Location of source 'f<sub>s</sub>' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f<sub>d</sub>' can also be anywhere from 000h to FFFh.</p> <p>Either source or destination can be W (a useful special situation).</p> <p>MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).</p> <p>The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.</p>
Words:	2
Cycles:	2 (3)

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: MOVFF REG1, REG2

#### Before Instruction

REG1 = 33h  
REG2 = 11h

#### After Instruction

REG1 = 33h  
REG2 = 33h

## MOVLB

### Move literal to low nibble in BSR

Syntax:	MOVLW k		
Operands:	0 ≤ k ≤ 255		
Operation:	k → BSR		
Status Affected:	None		
Encoding:	0000 0001 kkkk kkkk		
Description:	<p>The eight-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR&lt;7:4&gt; always remains '0', regardless of the value of k<sub>7</sub>:k<sub>4</sub>.</p>		
Words:	1		
Cycles:	1		
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR

Example: MOVLB 5

Before Instruction  
BSR Register = 02h  
After Instruction  
BSR Register = 05h

# PIC18F1XK50/PIC18LF1XK50

## MOVLW

### Move literal to W

Syntax:	MOVLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	k → W			
Status Affected:	None			
Encoding:	0000	1110	kkkk	kkkk
Description:	The eight-bit literal 'k' is loaded into W.			
Words:	1			
Cycles:	1			

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: MOVLW 5Ah

#### After Instruction

W = 5Ah

## MOVWF

### Move W to f

Syntax:	MOVWF f {,a}			
Operands:	0 ≤ f ≤ 255			
	a ∈ [0,1]			
Operation:	(W) → f			
Status Affected:	None			
Encoding:	0110	111a	ffff	ffff

Description: Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: MOVWF REG, 0

#### Before Instruction

W = 4Fh  
REG = FFh

#### After Instruction

W = 4Fh  
REG = 4Fh

# PIC18F1XK50/PIC18LF1XK50

MULLW	Multiply literal with W								
Syntax:	MULLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	$(W) \times k \rightarrow \text{PRODH:PRODL}$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>1101</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1101	kkkk	kkkk				
0000	1101	kkkk	kkkk						
Description:	<p>An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.</p> <p>None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write registers PRODH: PRODL</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL
Q1	Q2	Q3	Q4						
Decode	Read literal 'k'	Process Data	Write registers PRODH: PRODL						

Example: MULLW 0C4h

Before Instruction

W	=	E2h
PRODH	=	?
PRODL	=	?

After Instruction

W	=	E2h
PRODH	=	ADh
PRODL	=	08h

MULWF	Multiply W with f								
Syntax:	MULWF f {,a}								
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$								
Operation:	$(W) \times (f) \rightarrow \text{PRODH:PRODL}$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>001a</td><td>ffff</td><td>ffff</td></tr> </table>	0000	001a	ffff	ffff				
0000	001a	ffff	ffff						
Description:	<p>An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged.</p> <p>None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write registers PRODH: PRODL</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write registers PRODH: PRODL						

Example: MULWF REG, 1

Before Instruction

W	=	C4h
REG	=	B5h
PRODH	=	?
PRODL	=	?

After Instruction

W	=	C4h
REG	=	B5h
PRODH	=	8Ah
PRODL	=	94h

NEGF	Negate f								
Syntax:	NEGF f {,a}								
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$								
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 is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write register 'f'</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write register 'f'
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write register 'f'						

Example:      NEGF      REG, 1

Before Instruction  
REG = 0011 1010 [3Ah]  
After Instruction  
REG = 1100 0110 [C6h]

NOP	No Operation								
Syntax:	NOP								
Operands:	None								
Operation:	No operation								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>0000</td> </tr> <tr> <td>1111</td> <td>xxxx</td> <td>xxxx</td> <td>xxxx</td> </tr> </table>	0000	0000	0000	0000	1111	xxxx	xxxx	xxxx
0000	0000	0000	0000						
1111	xxxx	xxxx	xxxx						
Description:	No operation.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	No operation	No operation						

Example:

None.

# PIC18F1XK50/PIC18LF1XK50

---

POP	Pop Top of Return Stack								
Syntax:	POP								
Operands:	None								
Operation:	(TOS) → bit bucket								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0110</td></tr> </table>	0000	0000	0000	0110				
0000	0000	0000	0110						
Description:	<p>The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>POP TOS value</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	POP TOS value	No operation
Q1	Q2	Q3	Q4						
Decode	No operation	POP TOS value	No operation						

<u>Example:</u>	POP		
	GOTO	NEW	
<b>Before Instruction</b>			
TOS	=	0031A2h	
Stack (1 level down)	=	014332h	
<b>After Instruction</b>			
TOS	=	014332h	
PC	=	NEW	

PUSH	Push Top of Return Stack								
Syntax:	PUSH								
Operands:	None								
Operation:	(PC + 2) → TOS								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0101</td></tr> </table>	0000	0000	0000	0101				
0000	0000	0000	0101						
Description:	<p>The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>PUSH PC + 2 onto return stack</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	PUSH PC + 2 onto return stack	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	PUSH PC + 2 onto return stack	No operation	No operation						

<u>Example:</u>	PUSH		
<b>Before Instruction</b>			
TOS	=	345Ah	
PC	=	0124h	
<b>After Instruction</b>			
PC	=	0126h	
TOS	=	0126h	
Stack (1 level down)	=	345Ah	

# PIC18F1XK50/PIC18LF1XK50

RCALL	Relative Call												
Syntax:	RCALL n												
Operands:	$-1024 \leq n \leq 1023$												
Operation:	$(PC) + 2 \rightarrow TOS$ , $(PC) + 2 + 2n \rightarrow PC$												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1101</td> <td>1nnn</td> <td>nnnn</td> <td>nnnn</td> </tr> </table>	1101	1nnn	nnnn	nnnn								
1101	1nnn	nnnn	nnnn										
Description:	Subroutine call with a jump up to 1K from the current location. First, return address ( $PC + 2$ ) is pushed onto the stack. Then, add the 2's complement number ' $2n$ ' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC + 2 + 2n$ . This instruction is a two-cycle instruction.												
Words:	1												
Cycles:	2												
Q Cycle Activity:													
	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read literal 'n' PUSH PC to stack</td> <td>Process Data</td> <td>Write to PC</td> </tr> <tr> <td>No operation</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'n' PUSH PC to stack	Process Data	Write to PC	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'n' PUSH PC to stack	Process Data	Write to PC										
No operation	No operation	No operation	No operation										

Example: HERE RCALL Jump

Before Instruction

PC = Address (HERE)

After Instruction

PC = Address (Jump)

TOS = Address (HERE + 2)

RESET	Reset								
Syntax:	RESET								
Operands:	None								
Operation:	Reset all registers and flags that are affected by a MCLR Reset.								
Status Affected:	All								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>1111</td> <td>1111</td> </tr> </table>	0000	0000	1111	1111				
0000	0000	1111	1111						
Description:	This instruction provides a way to execute a MCLR Reset by software.								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Start Reset</td> <td>No operation</td> <td>No operation</td> </tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Start Reset	No operation	No operation
Q1	Q2	Q3	Q4						
Decode	Start Reset	No operation	No operation						

Example: RESET

After Instruction

Registers = Reset Value  
Flags\* = Reset Value

# PIC18F1XK50/PIC18LF1XK50

RETFIE	Return from Interrupt												
Syntax:	RETFIE {s}												
Operands:	$s \in [0,1]$												
Operation:	$(TOS) \rightarrow PC$ , $1 \rightarrow GIE/GIEH$ or $PEIE/GIEL$ , if $s = 1$ $(WS) \rightarrow W$ , $(STATUS) \rightarrow Status$ , $(BSRS) \rightarrow BSR$ , PCLATU, PCLATH are unchanged.												
Status Affected:	GIE/GIEH, PEIE/GIEL.												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0001</td><td>000s</td></tr> </table>	0000	0000	0001	000s								
0000	0000	0001	000s										
Description:	Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers, WS, STATUS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).												
Words:	1												
Cycles:	2												
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>No operation</td><td>POP PC from stack, Set GIEH or GIEL</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	No operation	No operation	POP PC from stack, Set GIEH or GIEL	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	No operation	No operation	POP PC from stack, Set GIEH or GIEL										
No operation	No operation	No operation	No operation										

Example:      RETFIE 1

After Interrupt

PC	=	TOS
W	=	WS
BSR	=	BSRS
Status	=	STATUS
GIE/GIEH, PEIE/GIEL	=	1

RETLW	Return literal to W												
Syntax:	RETLW k												
Operands:	$0 \leq k \leq 255$												
Operation:	$k \rightarrow W$ , $(TOS) \rightarrow PC$ , PCLATU, PCLATH are unchanged												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>1100</td><td>kkkk</td><td>kkkk</td></tr> </table>	0000	1100	kkkk	kkkk								
0000	1100	kkkk	kkkk										
Description:	<p>W is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.</p> <p>Words: 1</p> <p>Cycles: 2</p> <p>Q Cycle Activity:</p> <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>POP PC from stack, Write to W</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	POP PC from stack, Write to W	No operation	No operation	No operation	No operation
Q1	Q2	Q3	Q4										
Decode	Read literal 'k'	Process Data	POP PC from stack, Write to W										
No operation	No operation	No operation	No operation										

Example:

```

CALL TABLE ; W contains table
; offset value
; W now has
; table value
:
TABLE
  ADDWF PCL ; W = offset
  RETLW k0 ; Begin table
  RETLW k1 ;
  :
  RETLW kn ; End of table

Before Instruction
  W      = 07h
After Instruction
  W      = value of kn

```

# PIC18F1XK50/PIC18LF1XK50

RETURN	Return from Subroutine	RLCF	Rotate Left f through Carry								
Syntax:	RETURN {s}	Syntax:	RLCF f {,d {,a}}								
Operands:	$s \in [0,1]$	Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(TOS) \rightarrow PC$ , if $s = 1$ $(WS) \rightarrow W$ , $(STATUS) \rightarrow Status$ , $(BSRS) \rightarrow BSR$ , PCLATU, PCLATH are unchanged	Operation:	$(f<n>) \rightarrow dest<n+1>$ , $(f<7>) \rightarrow C$ , $(C) \rightarrow dest<0>$								
Status Affected:	None	Status Affected:	C, N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>0000</td><td>0001</td><td>001s</td></tr></table>	0000	0000	0001	001s	Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0011</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>	0011	01da	ffff	ffff
0000	0000	0001	001s								
0011	01da	ffff	ffff								
Description:	Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's' = 1, the contents of the shadow registers, WS, STATUS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).	Description:	The contents of register 'f' are rotated one bit to the left through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode</b> ” for details.								
Words:	1	Words:	1								
Cycles:	2	Cycles:	1								
Q Cycle Activity:		Q Cycle Activity:									
	Q1      Q2      Q3      Q4		Q1      Q2      Q3      Q4								
	Decode      No operation      Process Data      POP PC from stack		C      register f								
	No operation      No operation      No operation      No operation										

Example: RETURN

After Instruction:  
PC = TOS

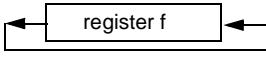
Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLCF REG, 0, 0

Before Instruction  
REG = 1110 0110  
C = 0

After Instruction  
REG = 1110 0110  
W = 1100 1100  
C = 1

# PIC18F1XK50/PIC18LF1XK50

RLNCF	Rotate Left f (No Carry)								
Syntax:	RLNCF f {,d {,a}}								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(f<8>) \rightarrow \text{dest}<8>$ , $(f<0>) \rightarrow \text{dest}<0>$								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0100</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>	0100	01da	ffff	ffff				
0100	01da	ffff	ffff						
Description:	<p>The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>								
									
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

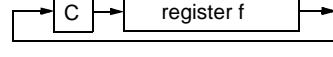
Example: RLNCF REG, 1, 0

Before Instruction

REG = 1010 1011

After Instruction

REG = 0101 0111

RRCF	Rotate Right f through Carry								
Syntax:	RRCF f {,d {,a}}								
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$								
Operation:	$(f<8>) \rightarrow \text{dest}<8>$ , $(f<0>) \rightarrow C$ , $(C) \rightarrow \text{dest}<0>$								
Status Affected:	C, N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0011</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>	0011	00da	ffff	ffff				
0011	00da	ffff	ffff						
Description:	<p>The contents of register 'f' are rotated one bit to the right through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>								
									
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4						
Decode	Read register 'f'	Process Data	Write to destination						

Example: 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)	SETF	Set f
Syntax:	RRNCF f {,d {,a}}	Syntax:	SETF f {,a}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]	Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(f<n>) → dest<n – 1>, (f<0>) → dest<7>	Operation:	FFh → f
Status Affected:	N, Z	Status Affected:	None
Encoding:	0100 00da ffff ffff	Encoding:	0110 100a ffff ffff
Description:	<p>The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p> 	Description:	<p>The contents of the specified register are set to FFh. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>
Words:	1	Words:	1
Cycles:	1	Cycles:	1
Q Cycle Activity:		Q Cycle Activity:	
	Q1      Q2      Q3      Q4		Q1      Q2      Q3      Q4
	Decode      Read register 'f'      Process Data      Write to destination		Decode      Read register 'f'      Process Data      Write register 'f'

Example 1:      RRNCF      REG, 1, 0

Before Instruction  
REG = 1101 0111  
After Instruction  
REG = 1110 1011

Example 2:      RRNCF      REG, 0, 0

Before Instruction  
W = ?  
REG = 1101 0111  
After Instruction  
W = 1110 1011  
REG = 1101 0111

Example:      SETF      REG, 1

Before Instruction  
REG = 5Ah  
After Instruction  
REG = FFh

# PIC18F1XK50/PIC18LF1XK50

SLEEP	Enter Sleep mode								
Syntax:	SLEEP								
Operands:	None								
Operation:	00h → WDT, 0 → WDT postscaler, 1 → <u>TO</u> , 0 → PD								
Status Affected:	<u>TO</u> , PD								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><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" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>No operation</td><td>Process Data</td><td>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:	SUBFWB 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:	<table border="1" style="display: inline-table; vertical-align: middle;"><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 is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.				

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: 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

SUBLW	Subtract W from literal			
Syntax:	SUBLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	k - (W) → W			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0000	1000	kkkk	kkkk
Description	W is subtracted from the eight-bit literal 'k'. The result is placed in W.			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'	Process Data	Write to W

Example 1: SUBLW 02h

Before Instruction

W = 01h  
C = ?

After Instruction

W = 01h  
C = 1 ; result is positive  
Z = 0  
N = 0

Example 2: SUBLW 02h

Before Instruction

W = 02h  
C = ?

After Instruction

W = 00h  
C = 1 ; result is zero  
Z = 1  
N = 0

Example 3: SUBLW 02h

Before Instruction

W = 03h  
C = ?

After Instruction

W = FFh ; (2's complement)  
C = 0 ; result is negative  
Z = 0  
N = 1

SUBWF	Subtract W from f			
Syntax:	SUBWF f {d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) - (W) → dest			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0101	11da	ffff	ffff
Description:	Subtract W from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process Data	Write to destination

Example 1: SUBWF REG, 1, 0

Before Instruction

REG = 3  
W = 2  
C = ?

After Instruction

REG = 1  
W = 2  
C = 1 ; result is positive  
Z = 0  
N = 0

Example 2: SUBWF REG, 0, 0

Before Instruction

REG = 2  
W = 2  
C = ?

After Instruction

REG = 2  
W = 0  
C = 1 ; result is zero  
Z = 1  
N = 0

Example 3: SUBWF REG, 1, 0

Before Instruction

REG = 1  
W = 2  
C = ?

After Instruction

REG = FFh ;(2's complement)  
W = 2  
C = 0 ; result is negative  
Z = 0  
N = 1

# PIC18F1XK50/PIC18LF1XK50

SUBWFB	Subtract W from f with Borrow	SWAPF	Swap f																
Syntax:	SUBWFB f {,d {,a}}	Syntax:	SWAPF f {,d {,a}}																
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]	Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]																
Operation:	(f) – (W) – (C) → dest	Operation:	(f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0>																
Status Affected:	N, OV, C, DC, Z	Status Affected:	None																
Encoding:	0101 10da ffff ffff	Encoding:	0011 10da ffff ffff																
Description:	<p>Subtract W and the CARRY flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>	Description:	<p>The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>																
Words:	1	Words:	1																
Cycles:	1	Cycles:	1																
Q Cycle Activity:		Q Cycle Activity:																	
	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination		<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4																
Decode	Read register 'f'	Process Data	Write to destination																
Q1	Q2	Q3	Q4																
Decode	Read register 'f'	Process Data	Write to destination																
<u>Example 1:</u>	SUBWFB REG, 1, 0	<u>Example:</u>	SWAPF REG, 1, 0																
Before Instruction		Before Instruction																	
REG = 19h (0001 1001)		REG = 53h																	
W = 0Dh (0000 1101)		After Instruction																	
C = 1		REG = 35h																	
After Instruction																			
REG = 0Ch (0000 1011)																			
W = 0Dh (0000 1101)																			
C = 1																			
Z = 0																			
N = 0 ; result is positive																			
<u>Example 2:</u>	SUBWFB REG, 0, 0																		
Before Instruction		Before Instruction																	
REG = 1Bh (0001 1011)		REG = 53h																	
W = 1Ah (0001 1010)		After Instruction																	
C = 0		REG = 35h																	
After Instruction																			
REG = 1Bh (0001 1011)																			
W = 00h																			
C = 1																			
Z = 1 ; result is zero																			
N = 0																			
<u>Example 3:</u>	SUBWFB REG, 1, 0																		
Before Instruction		Before Instruction																	
REG = 03h (0000 0011)		REG = 53h																	
W = 0Eh (0000 1101)		After Instruction																	
C = 1		REG = 35h																	
After Instruction																			
REG = F5h (1111 0100) ; [2's comp]																			
W = 0Eh (0000 1101)																			
C = 0																			
Z = 0																			
N = 1 ; result is negative																			

# PIC18F1XK50/PIC18LF1XK50

TBLRD	Table Read				
Syntax:	TBLRD ( *; *+; *-; +* )				
Operands:	None				
Operation:	<p>if TBLRD *,          (Prog Mem (TBLPTR)) → TABLAT;          TBLPTR – No Change;</p> <p>if TBLRD *+,          (Prog Mem (TBLPTR)) → TABLAT;          (TBLPTR) + 1 → TBLPTR;</p> <p>if TBLRD *-,          (Prog Mem (TBLPTR)) → TABLAT;          (TBLPTR) – 1 → TBLPTR;</p> <p>if TBLRD +*,          (TBLPTR) + 1 → TBLPTR;          (Prog Mem (TBLPTR)) → TABLAT;</p>				
Status Affected:	None				
Encoding:	<table border="1"> <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.</p> <p>The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.</p> <p>TBLPTR[0] = 0: Least Significant Byte of Program Memory Word</p> <p>TBLPTR[0] = 1: Most Significant Byte of Program Memory Word</p> <p>The TBLRD instruction can modify the value of TBLPTR as follows:</p> <ul style="list-style-type: none"> <li>• no change</li> <li>• post-increment</li> <li>• post-decrement</li> <li>• pre-increment</li> </ul>				
Words:	1				
Cycles:	2				

## Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)

TBLRD	Table Read (Continued)																		
Example1:	<u>TBLRD *+ ;</u> <p>Before Instruction</p> <table> <tr><td>TABLAT</td><td>=</td><td>55h</td></tr> <tr><td>TBLPTR</td><td>=</td><td>00A356h</td></tr> <tr><td>MEMORY (00A356h)</td><td>=</td><td>34h</td></tr> </table> <p>After Instruction</p> <table> <tr><td>TABLAT</td><td>=</td><td>34h</td></tr> <tr><td>TBLPTR</td><td>=</td><td>00A357h</td></tr> </table>	TABLAT	=	55h	TBLPTR	=	00A356h	MEMORY (00A356h)	=	34h	TABLAT	=	34h	TBLPTR	=	00A357h			
TABLAT	=	55h																	
TBLPTR	=	00A356h																	
MEMORY (00A356h)	=	34h																	
TABLAT	=	34h																	
TBLPTR	=	00A357h																	
Example2:	<u>TBLRD +* ;</u> <p>Before Instruction</p> <table> <tr><td>TABLAT</td><td>=</td><td>AAh</td></tr> <tr><td>TBLPTR</td><td>=</td><td>01A357h</td></tr> <tr><td>MEMORY (01A357h)</td><td>=</td><td>12h</td></tr> <tr><td>MEMORY (01A358h)</td><td>=</td><td>34h</td></tr> </table> <p>After Instruction</p> <table> <tr><td>TABLAT</td><td>=</td><td>34h</td></tr> <tr><td>TBLPTR</td><td>=</td><td>01A358h</td></tr> </table>	TABLAT	=	AAh	TBLPTR	=	01A357h	MEMORY (01A357h)	=	12h	MEMORY (01A358h)	=	34h	TABLAT	=	34h	TBLPTR	=	01A358h
TABLAT	=	AAh																	
TBLPTR	=	01A357h																	
MEMORY (01A357h)	=	12h																	
MEMORY (01A358h)	=	34h																	
TABLAT	=	34h																	
TBLPTR	=	01A358h																	

# PIC18F1XK50/PIC18LF1XK50

TBLWT	Table Write				
Syntax:	TBLWT ( *, *+; *-; +* )				
Operands:	None				
Operation:	<pre> if TBLWT*,  (TABLAT) → Holding Register;  TBLPTR – No Change;  if TBLWT*+,  (TABLAT) → Holding Register;  (TBLPTR) + 1 → TBLPTR;  if TBLWT*-,  (TABLAT) → Holding Register;  (TBLPTR) – 1 → TBLPTR;  if TBLWT+*,  (TBLPTR) + 1 → TBLPTR;  (TABLAT) → Holding Register; </pre>				
Status Affected:	None				
Encoding:	<table border="1"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>11nn nn=0 * =1 *+ =2 *- =3 +*</td> </tr> </table>	0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*
0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*		
Description:	<p>This instruction uses the 3 LSBs of TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to <b>Section 4.0 “Flash Program Memory”</b> for additional details on programming Flash memory.) The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-MByte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.</p> <p>TBLPTR[0] = 0: Least Significant Byte of Program Memory Word  TBLPTR[0] = 1: Most Significant Byte of Program Memory Word</p> <p>The TBLWT instruction can modify the value of TBLPTR as follows:</p> <ul style="list-style-type: none"> <li>• no change</li> <li>• post-increment</li> <li>• post-decrement</li> <li>• pre-increment</li> </ul>				
Words:	1				
Cycles:	2				
Q Cycle Activity:					

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register )

TBLWT	Table Write (Continued)
Example1:	TBLWT *+;
Before Instruction	
TABLAT	= 55h
TBLPTR	= 00A356h
HOLDING REGISTER (00A356h)	= FFh
After Instructions (table write completion)	
TABLAT	= 55h
TBLPTR	= 00A357h
HOLDING REGISTER (00A356h)	= 55h
Example 2:	TBLWT +*;
Before Instruction	
TABLAT	= 34h
TBLPTR	= 01389Ah
HOLDING REGISTER (01389Ah)	= FFh
HOLDING REGISTER (01389Bh)	= FFh
After Instruction (table write completion)	
TABLAT	= 34h
TBLPTR	= 01389Bh
HOLDING REGISTER (01389Ah)	= FFh
HOLDING REGISTER (01389Bh)	= 34h

# PIC18F1XK50/PIC18LF1XK50

TSTFSZ	Test f, skip if 0				
Syntax:	TSTFSZ f {,a}				
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$				
Operation:	skip if $f = 0$				
Status Affected:	None				
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0110</td><td>011a</td><td>ffff</td><td>ffff</td></tr></table>	0110	011a	ffff	ffff
0110	011a	ffff	ffff		
Description:	If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a NOP is executed, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.				
Words:	1				
Cycles:	1(2) <b>Note:</b> 3 cycles if skip and followed by a 2-word instruction.				

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example: HERE TSTFSZ CNT, 1  
NZERO :  
ZERO :

Before Instruction  
PC = Address (HERE)  
After Instruction  
If CNT = 00h,  
PC = Address (ZERO)  
If CNT  $\neq$  00h,  
PC = Address (NZERO)

XORLW	Exclusive OR literal with W								
Syntax:	XORLW k								
Operands:	$0 \leq k \leq 255$								
Operation:	(W) .XOR. k $\rightarrow$ W								
Status Affected:	N, Z								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>1010</td><td>kkkk</td><td>kkkk</td></tr></table>	0000	1010	kkkk	kkkk				
0000	1010	kkkk	kkkk						
Description:	The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to W</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to W
Q1	Q2	Q3	Q4						
Decode	Read literal 'k'	Process Data	Write to W						

Example: XORLW 0AFh

Before Instruction  
W = B5h  
After Instruction  
W = 1Ah

# PIC18F1XK50/PIC18LF1XK50

---

---

## XORWF      Exclusive OR W with f

---

Syntax:	XORWF    f {,d {,a}}											
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$											
Operation:	$(W) .XOR. (f) \rightarrow \text{dest}$											
Status Affected:	N, Z											
Encoding:	0001	10da	ffff	ffff								
Description:	Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See <b>Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table><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    =    AFh  
W       =    B5h

After Instruction

REG    =    1Ah  
W       =    B5h

## 25.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18F1XK50/PIC18LF1XK50 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for indexed addressing. Two of the instructions, ADDFSR and SUBFSR, each have an additional special instantiation for using FSR2. These versions (ADDULNK and SUBULNK) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 25-3. Detailed descriptions are provided in **Section 25.2.2 “Extended Instruction Set”**. The opcode field descriptions in Table 25-1 (page 304) apply to both the standard and extended PIC18 instruction sets.

**Note:** The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

### 25.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets ("[ ]"). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 25.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”**.

**Note:** In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{ }").

TABLE 25-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected
			MSb	LSb			
ADDFSR f, k	Add literal to FSR	1	1110	1000	ffkk	kkkk	None
ADDULNK k	Add literal to FSR2 and return	2	1110	1000	11kk	kkkk	None
CALLW	Call subroutine using WREG	2	0000	0000	0001	0100	None
MOVSF z <sub>s</sub> , f <sub>d</sub>	Move z <sub>s</sub> (source) to 1st word f <sub>d</sub> (destination) 2nd word	2	1110	1011	0zzz	zzzz	None
MOVSS z <sub>s</sub> , z <sub>d</sub>	Move z <sub>s</sub> (source) to 1st word z <sub>d</sub> (destination) 2nd word	2	1111	ffff	ffff	ffff	None
PUSHL k	Store literal at FSR2, decrement FSR2	1	1110	1011	1zzz	zzzz	None
SUBFSR f, k	Subtract literal from FSR	1	1110	1001	ffkk	kkkk	None
SUBULNK k	Subtract literal from FSR2 and return	2	1110	1001	11kk	kkkk	None

# PIC18F1XK50/PIC18LF1XK50

## 25.2.2 EXTENDED INSTRUCTION SET

ADDFSR	Add Literal to FSR	ADDULNK	Add Literal to FSR2 and Return								
Syntax:	ADDFSR f, k	Syntax:	ADDULNK k								
Operands:	$0 \leq k \leq 63$ $f \in [0, 1, 2]$	Operands:	$0 \leq k \leq 63$								
Operation:	$FSR(f) + k \rightarrow FSR(f)$	Operation:	$FSR2 + k \rightarrow FSR2$ , (TOS) $\rightarrow$ PC								
Status Affected:	None	Status Affected:	None								
Encoding:	1110 1000 ffk <sub>k</sub> kkkk	Encoding:	1110 1000 11k <sub>k</sub> kkkk								
Description:	The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.	Description:	The 6-bit literal 'k' is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS.								
Words:	1	Words:	1								
Cycles:	1	Cycles:	2								
Q Cycle Activity:		Q Cycle Activity:	The instruction takes two cycles to execute; a NOP is performed during the second cycle.								
	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to FSR</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read literal 'k'	Process Data	Write to FSR		<p>This may be thought of as a special case of the ADDFSR instruction, where <math>f = 3</math> (binary '11'); it operates only on FSR2.</p>
Q1	Q2	Q3	Q4								
Decode	Read literal 'k'	Process Data	Write to FSR								

Example: ADDFSR 2, 23h

Before Instruction  
FSR2 = 03FFh  
After Instruction  
FSR2 = 0422h

Words: 1  
Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to FSR
No Operation	No Operation	No Operation	No Operation

Example: ADDULNK 23h

Before Instruction  
FSR2 = 03FFh  
PC = 0100h  
After Instruction  
FSR2 = 0422h  
PC = (TOS)

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).

CALLW	Subroutine Call Using WREG	MOVSF	Move Indexed to f																								
Syntax:	CALLW	Syntax:	MOVSF $[z_s], f_d$																								
Operands:	None	Operands:	$0 \leq z_s \leq 127$ $0 \leq f_d \leq 4095$																								
Operation:	$(PC + 2) \rightarrow TOS,$ $(W) \rightarrow PCL,$ $(PCLATH) \rightarrow PCH,$ $(PCLATU) \rightarrow PCU$	Operation:	$((FSR2) + z_s) \rightarrow f_d$																								
Status Affected:	None	Status Affected:	None																								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>0000</td><td>0001</td><td>0100</td></tr></table>	0000	0000	0001	0100	Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>1011</td><td>0zzz</td><td>zzzz<sub>s</sub></td></tr><tr><td>1111</td><td>ffff</td><td>ffff</td><td>ffff<sub>d</sub></td></tr></table>	1110	1011	0zzz	zzzz <sub>s</sub>	1111	ffff	ffff	ffff <sub>d</sub>												
0000	0000	0001	0100																								
1110	1011	0zzz	zzzz <sub>s</sub>																								
1111	ffff	ffff	ffff <sub>d</sub>																								
Description	First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched into PCH and PCU, respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched. Unlike CALL, there is no option to update W, Status or BSR.	Description:	The contents of the source register are moved to destination register ' $f_d$ '. The actual address of the source register is determined by adding the 7-bit literal offset ' $z_s$ ' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal ' $f_d$ ' in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh). The MOVSF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register. If the resultant source address points to an indirect addressing register, the value returned will be 00h.																								
Words:	1	Words:	2																								
Cycles:	2	Cycles:	2																								
Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read WREG</td><td>PUSH PC to stack</td><td>No operation</td></tr><tr><td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Read WREG	PUSH PC to stack	No operation	No operation	No operation	No operation	No operation	Q Cycle Activity:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Determine source addr</td><td>Determine source addr</td><td>Read source reg</td></tr><tr><td>Decode</td><td>No operation No dummy read</td><td>No operation</td><td>Write register '<math>f</math>' (dest)</td></tr></table>	Q1	Q2	Q3	Q4	Decode	Determine source addr	Determine source addr	Read source reg	Decode	No operation No dummy read	No operation	Write register ' $f$ ' (dest)
Q1	Q2	Q3	Q4																								
Decode	Read WREG	PUSH PC to stack	No operation																								
No operation	No operation	No operation	No operation																								
Q1	Q2	Q3	Q4																								
Decode	Determine source addr	Determine source addr	Read source reg																								
Decode	No operation No dummy read	No operation	Write register ' $f$ ' (dest)																								
Example:	HERE CALLW	Example:	MOVSF [05h], REG2																								
Before Instruction		Before Instruction																									
PC = address (HERE)		FSR2 = 80h																									
PCLATH = 10h		Contents of 85h = 33h																									
PCLATU = 00h		REG2 = 11h																									
W = 06h																											
After Instruction		After Instruction																									
PC = 001006h		FSR2 = 80h																									
TOS = address (HERE + 2)		Contents of 85h = 33h																									
PCLATH = 10h		REG2 = 33h																									
PCLATU = 00h																											
W = 06h																											

# PIC18F1XK50/PIC18LF1XK50

MOVSS	Move Indexed to Indexed	PUSHL	Store Literal at FSR2, Decrement FSR2																				
Syntax:	MOVSS [z <sub>s</sub> ], [z <sub>d</sub> ]	Syntax:	PUSHL k																				
Operands:	0 ≤ z <sub>s</sub> ≤ 127 0 ≤ z <sub>d</sub> ≤ 127	Operands:	0 ≤ k ≤ 255																				
Operation:	((FSR2) + z <sub>s</sub> ) → ((FSR2) + z <sub>d</sub> )	Operation:	k → (FSR2), FSR2 – 1 → FSR2																				
Status Affected:	None	Status Affected:	None																				
Encoding:		Encoding:																					
1st word (source)	1110 1011 1zzz zzzz <sub>s</sub>	1111 1010 kkkk kkkk																					
2nd word (dest.)	1111 xxxx xzzz zzzz <sub>d</sub>																						
Description	<p>The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets 'z<sub>s</sub>' or 'z<sub>d</sub>', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).</p> <p>The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.</p> <p>If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP.</p>	<p>The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation. This instruction allows users to push values onto a software stack.</p>																					
Words:	2	Words:	1																				
Cycles:	2	Cycles:	1																				
Q Cycle Activity:		Q Cycle Activity:																					
	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Determine source addr</td><td>Determine source addr</td><td>Read source reg</td></tr> <tr> <td>Decode</td><td>Determine dest addr</td><td>Determine dest addr</td><td>Write to dest reg</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Determine source addr	Determine source addr	Read source reg	Decode	Determine dest addr	Determine dest addr	Write to dest reg	<table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read 'k'</td><td>Process data</td><td>Write to destination</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read 'k'	Process data	Write to destination	
Q1	Q2	Q3	Q4																				
Decode	Determine source addr	Determine source addr	Read source reg																				
Decode	Determine dest addr	Determine dest addr	Write to dest reg																				
Q1	Q2	Q3	Q4																				
Decode	Read 'k'	Process data	Write to destination																				

Example: MOVSS [05h], [06h]

Before Instruction

FSR2	=	80h
Contents of 85h	=	33h
Contents of 86h	=	11h

After Instruction

FSR2	=	80h
Contents of 85h	=	33h
Contents of 86h	=	33h

Example: PUSHL 08h

Before Instruction

FSR2H:FSR2L	=	01ECh
Memory (01ECh)	=	00h

After Instruction

FSR2H:FSR2L	=	01EBh
Memory (01ECh)	=	08h

# PIC18F1XK50/PIC18LF1XK50

<b>SUBFSR</b>	<b>Subtract Literal from FSR</b>								
Syntax:	SUBFSR f, k								
Operands:	$0 \leq k \leq 63$ $f \in [0, 1, 2]$								
Operation:	$FSR(f) - k \rightarrow FSRf$								
Status Affected:	None								
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>1001</td><td>ffkk</td><td>kkkk</td></tr></table>	1110	1001	ffkk	kkkk				
1110	1001	ffkk	kkkk						
Description:	The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.								
Words:	1								
Cycles:	1								
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 2px;">Q1</th><th style="text-align: center; padding: 2px;">Q2</th><th style="text-align: center; padding: 2px;">Q3</th><th style="text-align: center; padding: 2px;">Q4</th></tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 2px;">Decode</td><td style="text-align: center; padding: 2px;">Read register 'f'</td><td style="text-align: center; padding: 2px;">Process Data</td><td style="text-align: center; padding: 2px;">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: SUBFSR 2, 23h

Before Instruction

FSR2 = 03FFh

After Instruction

FSR2 = 03DCh

<b>SUBULNK</b>	<b>Subtract Literal from FSR2 and Return</b>												
Syntax:	SUBULNK k												
Operands:	$0 \leq k \leq 63$												
Operation:	$FSR2 - k \rightarrow FSR2$ (TOS) $\rightarrow$ PC												
Status Affected:	None												
Encoding:	<table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>1110</td><td>1001</td><td>11kk</td><td>kkkk</td></tr></table>	1110	1001	11kk	kkkk								
1110	1001	11kk	kkkk										
Description:	The 6-bit literal 'k' is subtracted from the contents of the FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a NOP is performed during the second cycle. This may be thought of as a special case of the SUBFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.												
Words:	1												
Cycles:	2												
Q Cycle Activity:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 2px;">Q1</th><th style="text-align: center; padding: 2px;">Q2</th><th style="text-align: center; padding: 2px;">Q3</th><th style="text-align: center; padding: 2px;">Q4</th></tr> </thead> <tbody> <tr> <td style="text-align: center; padding: 2px;">Decode</td><td style="text-align: center; padding: 2px;">Read register 'f'</td><td style="text-align: center; padding: 2px;">Process Data</td><td style="text-align: center; padding: 2px;">Write to destination</td></tr> <tr> <td style="text-align: center; padding: 2px;">No Operation</td><td style="text-align: center; padding: 2px;">No Operation</td><td style="text-align: center; padding: 2px;">No Operation</td><td style="text-align: center; padding: 2px;">No Operation</td></tr> </tbody> </table>	Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination	No Operation	No Operation	No Operation	No Operation
Q1	Q2	Q3	Q4										
Decode	Read register 'f'	Process Data	Write to destination										
No Operation	No Operation	No Operation	No Operation										

Example: SUBULNK 23h

Before Instruction

FSR2 = 03FFh

PC = 0100h

After Instruction

FSR2 = 03DCh

PC = (TOS)

## 25.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

**Note:** Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing mode (**Section 3.5.1 “Indexed Addressing with Literal Offset”**). This has a significant impact on the way that many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: either as a location in the Access Bank ('a' = 0), or in a GPR bank designated by the BSR ('a' = 1). When the extended instruction set is enabled and 'a' = 0, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions that use the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between C and assembly routines in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see **Section 25.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”**).

Although the Indexed Literal Offset Addressing mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset Addressing mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

### 25.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument, 'f', in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value, 'k'. As already noted, this occurs only when 'f' is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets ("[ ]"). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within brackets, will generate an error in the MPASM™ assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be '0'. This is in contrast to standard operation (extended instruction set disabled) when 'a' is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM assembler.

The destination argument, 'd', functions as before.

In the latest versions of the MPASM assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, /y, or the PE directive in the source listing.

## 25.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18F1XK50/PIC18LF1XK50, it is very important to consider the type of code. A large, re-entrant application that is written in 'C' and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.

# PIC18F1XK50/PIC18LF1XK50

ADDWF	ADD W to Indexed (Indexed Literal Offset mode)											
Syntax:	ADDWF [k] {,d}											
Operands:	0 ≤ k ≤ 95 d ∈ {0,1}											
Operation:	(W) + ((FSR2) + k) → dest											
Status Affected:	N, OV, C, DC, Z											
Encoding:	0010	01d0	kkkk	kkkk								
Description:	The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="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 'k'</td><td>Process Data</td><td>Write to destination</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read 'k'	Process Data	Write to destination
Q1	Q2	Q3	Q4									
Decode	Read 'k'	Process Data	Write to destination									

Example: ADDWF [OFST], 0

Before Instruction

W	=	17h
OFST	=	2Ch
FSR2	=	0A00h
Contents of 0A2Ch	=	20h

After Instruction

W	=	37h
Contents of 0A2Ch	=	20h

BSF	Bit Set Indexed (Indexed Literal Offset mode)											
Syntax:	BSF [k], b											
Operands:	0 ≤ f ≤ 95 0 ≤ b ≤ 7											
Operation:	1 → ((FSR2) + k)<b>											
Status Affected:	None											
Encoding:	1000	bbb0	kkkk	kkkk								
Description:	Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4									
Decode	Read register 'f'	Process Data	Write to destination									

Example: BSF [FLAG\_OFST], 7

Before Instruction	
FLAG_OFST	= 0Ah
FSR2	= 0A00h
Contents of 0A0Ah	= 55h
After Instruction	
Contents of 0A0Ah	= D5h

SETF	Set Indexed (Indexed Literal Offset mode)											
Syntax:	SETF [k]											
Operands:	0 ≤ k ≤ 95											
Operation:	FFh → ((FSR2) + k)											
Status Affected:	None											
Encoding:	0110	1000	kkkk	kkkk								
Description:	The contents of the register indicated by FSR2, offset by 'k', are set to FFh.											
Words:	1											
Cycles:	1											
Q Cycle Activity:	<table border="1" style="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 'k'</td><td>Process Data</td><td>Write register</td></tr> </table>				Q1	Q2	Q3	Q4	Decode	Read 'k'	Process Data	Write register
Q1	Q2	Q3	Q4									
Decode	Read 'k'	Process Data	Write register									

Example: SETF [OFST]

Before Instruction	
OFST	= 2Ch
FSR2	= 0A00h
Contents of 0A2Ch	= 00h
After Instruction	
Contents of 0A2Ch	= FFh

# PIC18F1XK50/PIC18LF1XK50

---

---

## 25.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18F1XK50/PIC18LF1XK50 family of devices. This includes the MPLAB® C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

## 26.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C18 and MPLAB C30 C Compilers
  - MPLINK™ Object Linker/  
MPLIB™ Object Librarian
  - MPLAB ASM30 Assembler/Linker/Library
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD 2
- Device Programmers
  - PICSTART® Plus Development Programmer
  - MPLAB PM3 Device Programmer
  - PICkit™ 2 Development Programmer
- Low-Cost Demonstration and Development Boards and Evaluation Kits

## 26.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Visual device initializer for easy register initialization
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as HI-TECH Software C Compilers and IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either assembly or C)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (assembly or C)
  - Mixed assembly and C
  - Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

## 26.2 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for all PIC MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

## 26.3 MPLAB C18 and MPLAB C30 C Compilers

The MPLAB C18 and MPLAB C30 Code Development Systems are complete ANSI C compilers for Microchip's PIC18 and PIC24 families of microcontrollers and the dsPIC30 and dsPIC33 family of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

## 26.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

## 26.5 MPLAB ASM30 Assembler, Linker and Librarian

MPLAB ASM30 Assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

## 26.6 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C18 and MPLAB C30 C Compilers, and the MPASM and MPLAB ASM30 Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 26.7 MPLAB ICE 2000 High-Performance In-Circuit Emulator

The MPLAB ICE 2000 In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers. Software control of the MPLAB ICE 2000 In-Circuit Emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The architecture of the MPLAB ICE 2000 In-Circuit Emulator allows expansion to support new PIC microcontrollers.

The MPLAB ICE 2000 In-Circuit Emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft® Windows® 32-bit operating system were chosen to best make these features available in a simple, unified application.

## 26.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs PIC® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The MPLAB REAL ICE probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with the popular MPLAB ICD 2 system (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

MPLAB REAL ICE is field upgradeable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added, such as software breakpoints and assembly code trace. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, real-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 26.9 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low-cost, run-time development tool, connecting to the host PC via an RS-232 or high-speed USB interface. This tool is based on the Flash PIC MCUs and can be used to develop for these and other PIC MCUs and dsPIC DSCs. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the Flash devices. This feature, along with Microchip's In-Circuit Serial Programming™ (ICSP™) protocol, offers cost-effective, in-circuit Flash debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single stepping and watching variables, and CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real time. MPLAB ICD 2 also serves as a development programmer for selected PIC devices.

## 26.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages and a modular, detachable socket assembly to support various package types. The ICSP™ cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices and incorporates an SD/MMC card for file storage and secure data applications.

# PIC18F1XK50/PIC18LF1XK50

---

## 26.11 PICSTART Plus Development Programmer

The PICSTART Plus Development Programmer is an easy-to-use, low-cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. The PICSTART Plus Development Programmer supports most PIC devices in DIP packages up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus Development Programmer is CE compliant.

## 26.12 PICkit 2 Development Programmer

The PICkit™ 2 Development Programmer is a low-cost programmer and selected Flash device debugger with an easy-to-use interface for programming many of Microchip's baseline, mid-range and PIC18F families of Flash memory microcontrollers. The PICkit 2 Starter Kit includes a prototyping development board, twelve sequential lessons, software and HI-TECH's PICC™ Lite C compiler, and is designed to help get up to speed quickly using PIC® microcontrollers. The kit provides everything needed to program, evaluate and develop applications using Microchip's powerful, mid-range Flash memory family of microcontrollers.

## 26.13 Demonstration, Development and Evaluation Boards

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Check the Microchip web page ([www.microchip.com](http://www.microchip.com)) for the complete list of demonstration, development and evaluation kits.

## 27.0 ELECTRICAL SPECIFICATIONS

### Absolute Maximum Ratings<sup>(†)</sup>

Ambient temperature under bias.....	-40°C to +125°C
Storage temperature .....	-65°C to +150°C
Voltage on VDD with respect to Vss, PIC18F1XK50 .....	-0.3V to +6.0V
Voltage on VDD with respect to Vss, PIC18LF1XK50 .....	-0.3V to +4.0V
Voltage on MCLR with respect to Vss .....	-0.3V to +9.0V
Voltage on VUSB pin with respect to Vss .....	-0.3V to +4.0V
Voltage on D+ and D- pins with respect to Vss .....	-0.3V to (VUSB + 0.3V)
Voltage on all other pins with respect to Vss .....	-0.3V to (VDD + 0.3V)
Total power dissipation <sup>(1)</sup> .....	800 mW
Maximum current out of VSS pin .....	95 mA
Maximum current into VDD pin .....	95 mA
Clamp current, IK (VPIN < 0 or VPIN > VDD).....	± 20 mA
Maximum output current sunk by any I/O pin.....	25 mA
Maximum output current sourced by any I/O pin .....	25 mA
Maximum current sunk by all ports .....	90 mA
Maximum current sourced by all ports .....	90 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:** Vusb must always be  $\leq V_{DD} + 0.3V$

**† 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 above maximum rating conditions for extended periods may affect device reliability.

ADVANCE

# PIC18F1XK50/PIC18LF1XK50

## 27.1 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (Industrial, Extended)

PIC18LF1XK50			Standard Operating Conditions (unless otherwise stated)							
PIC18F1XK50			Standard Operating Conditions (unless otherwise stated)							
Param. No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions			
D001	VDD	<b>Supply Voltage</b>		PIC18LF1XK50	1.8	—	3.6	V	FOSC < = 20 MHz FOSC < = 48 MHz	
					2.7	—	3.6	V		
D001		PIC18F1XK50	1.8	—	5.5	V	FOSC < = 20 MHz FOSC < = 48 MHz			
D002*	VDR	<b>RAM Data Retention Voltage<sup>(1)</sup></b>		PIC18LF1XK50	1.5	—	—	V	Device in Sleep mode	
					1.7	—	—	V	Device in Sleep mode	
	VPOR*	<b>Power-on Reset Release Voltage</b>	—	1.6	—	V				
	VPORR*	<b>Power-on Reset Rearm Voltage</b>	—	0.8	—	V				
	VFVR	<b>Fixed Voltage Reference Voltage (calibrated)</b>	0.974 1.968 3.736	1.024 2.048 4.096	1.064 2.158 4.226	V	FVR1S<1:0> = 00 (1x) FVR1S<1:0> = 01 (2x) FVR1S<1:0> = 10 (4x), VDD > = 4.75V			
D004*	SVDD	<b>VDD Rise Rate</b> to ensure internal Power-on Reset signal	0.05	—	—	V/ms				

\* These parameters are characterized but not tested.

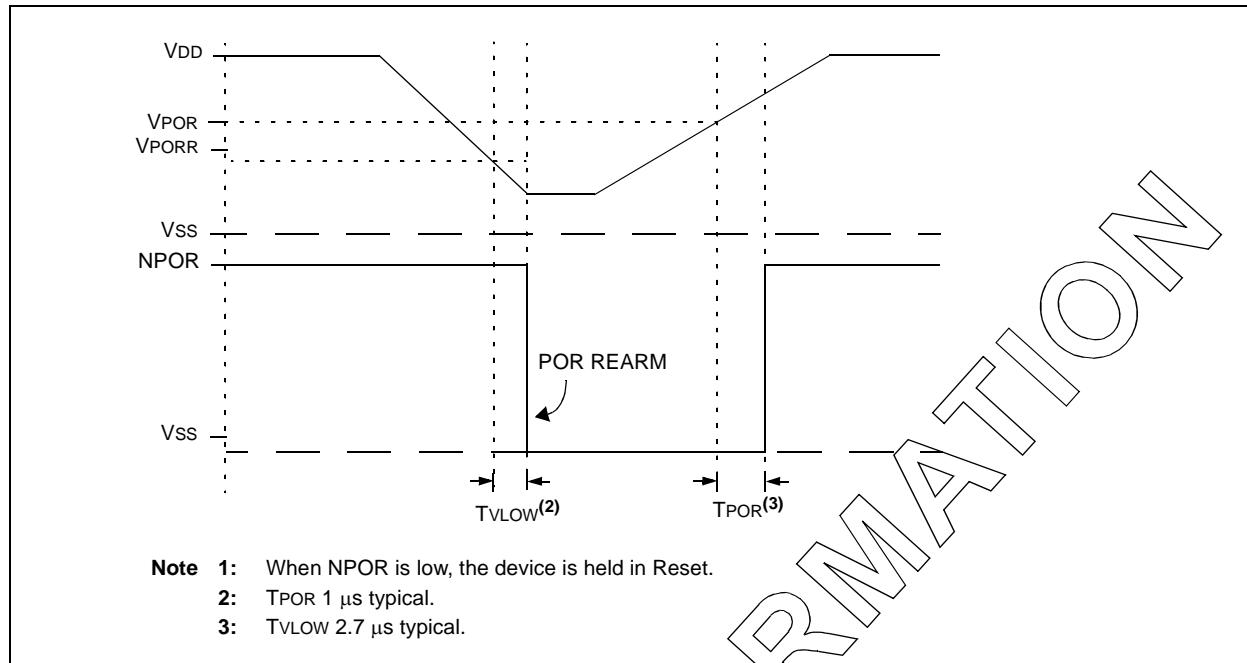
† Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.

ADVANCE

# PIC18F1XK50/PIC18LF1XK50

FIGURE 27-1: POR AND POR REARM WITH SLOW RISING VDD



ADVANCE INFORMATION

# PIC18F1XK50/PIC18LF1XK50

## 27.2 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (Industrial, Extended)

PIC18LF1XK50			Standard Operating Conditions (unless otherwise stated)				
PIC18F1XK50			Standard Operating Conditions (unless otherwise stated)				
Param No.	Device Characteristics	Min	Typ†	Max	Units	Conditions	
						VDD	Note
<b>Supply Current (IDD)<sup>(1, 2)</sup></b>							
D009	LDO Regulator	—	30	—	µA	—	
		—	5	—	µA	—	LP Clock mode and Sleep (requires FVR and BOR to be disabled)
D010		—	6.0	9	µA	1.8	Fosc = 32 kHz
		—	7	12	µA	3.0	LP Oscillator <sup>(4)</sup> , -40°C ≤ TA ≤ +85°C
D010		—	6	11	µA	1.8	Fosc = 32 kHz
		—	7	17	µA	3.0	LP Oscillator <sup>(4)</sup> , -40°C ≤ TA ≤ +85°C
		—	12	20	µA	5.0	
D011*		—	6.0	12	µA	1.8	Fosc = 32 kHz
		—	9.0	16	µA	3.0	LP Oscillator -40°C ≤ TA ≤ +125°C
D011*		—	8.0	15	µA	1.8	Fosc = 32 kHz
		—	11	25	µA	3.0	LP Oscillator <sup>(4)</sup>
		—	12	35	µA	5.0	-40°C ≤ TA ≤ +125°C
D011*		—	170	220	µA	1.8	Fosc = 1 MHz
		—	280	370	µA	3.0	XT Oscillator
D011*		—	200	250	µA	1.8	Fosc = 1 MHz
		—	310	400	µA	3.0	XT Oscillator
		—	380	490	µA	5.0	
D011*		—	75	110	µA	1.8	Fosc = 1 MHz
		—	130	190	µA	3.0	XT Oscillator CPU Idle
D011*		—	90	130	µA	1.8	Fosc = 1 MHz
		—	140	210	µA	3.0	XT Oscillator CPU Idle
		—	160	250	µA	5.0	

\* These parameters are characterized but not tested.

Legend: TBD = To Be Determined

Note 1: 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 disabled.

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.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be extended by the formula  $IR = VDD/2REXT$  (mA) with REXT in kΩ.

4: FVR and BOR are disabled.

5: 330 nF capacitor on VUSB pin.

# PIC18F1XK50/PIC18LF1XK50

## 27.2 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (Industrial, Extended)

PIC18LF1XK50			Standard Operating Conditions (unless otherwise stated)				
Param No.	Device Characteristics	Min	Typ†	Max	Units	Conditions	
						VDD	Note
Supply Current (IDD) <sup>(1, 2)</sup>							
D012		—	300	700	µA	1.8	FOSC = 4 MHz XT Oscillator
		—	500	1200	µA	3.0	
D012		—	330	700	µA	1.8	FOSC = 4 MHz XT Oscillator
		—	530	1200	µA	3.0	
D012A		—	240	300	µA	1.8	FOSC = 4 MHz, XT Oscillator CPU Idle
		—	440	550	µA	3.0	
D012A		—	230	300	µA	1.8	FOSC = 4 MHz XT Oscillator CPU Idle
		—	400	550	µA	3.0	
D013		—	470	640	µA	5.0	
		—	140	180	µA	1.8	FOSC = 1 MHz EC Oscillator (medium power)
D013		—	230	300	µA	3.0	
		—	160	210	µA	1.8	FOSC = 1 MHz EC Oscillator (medium power) <sup>(5)</sup>
D013A		—	250	310	µA	3.0	
		—	290	380	µA	5.0	
D013A		—	50	64	µA	1.8	FOSC = 1 MHz EC Oscillator (medium power) CPU Idle
		—	86	110	µA	3.0	
D013A		—	70	100	µA	1.8	FOSC = 1 MHz EC Oscillator (medium power) CPU Idle <sup>(5)</sup>
		—	100	150	µA	3.0	
D014		—	120	170	µA	5.0	
		—	500	640	µA	1.8	FOSC = 4 MHz EC Oscillator (medium power)
D014		—	830	1100	µA	3.0	
		—	520	660	µA	1.8	FOSC = 4 MHz EC Oscillator (medium power) <sup>(5)</sup>
		—	860	1100	µA	3.0	
		—	1000	1300	µA	5.0	

\* These parameters are characterized but not tested.

Legend: TBD = To Be Determined

Note 1: 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 disabled.

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.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be extended by the formula  $IR = VDD/2REXT$  (mA) with REXT in kΩ.

4: FVR and BOR are disabled.

5: 330 nF capacitor on VUSB pin.

# PIC18F1XK50/PIC18LF1XK50

## 27.2 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (Industrial, Extended)

PIC18LF1XK50			Standard Operating Conditions (unless otherwise stated)				
PIC18F1XK50			Standard Operating Conditions (unless otherwise stated)				
Param No.	Device Characteristics	Min	Typ†	Max	Units	Conditions	
						V <sub>DD</sub>	Note
<b>Supply Current (I<sub>DD</sub>)<sup>(1, 2)</sup></b>							
D014A		—	200	250	µA	1.8	Fosc = 4 MHz EC Oscillator (medium power) CPU Idle
		—	340	440	µA	3.0	
D014A		—	210	280	µA	1.8	Fosc = 4 MHz EC Oscillator (medium power) CPU Idle <sup>(5)</sup>
		—	360	470	µA	3.0	
		—	430	570	µA	5.0	
D015		—	820	1000	µA	1.8	Fosc = 6 MHz EC Oscillator (high power)
		—	1500	1900	µA	3.0	
D015		—	830	1100	µA	1.8	Fosc = 6 MHz EC Oscillator (high power) <sup>(5)</sup>
		—	1500	1900	µA	3.0	
		—	1700	2300	µA	5.0	
D015A		—	300	370	µA	1.8 3.0	Fosc = 6 MHz EC Oscillator (high power) CPU Idle
		—	510	660	µA		
D015A		—	320	430	µA	1.8	Fosc = 6 MHz EC Oscillator (high power) CPU Idle <sup>(5)</sup>
		—	530	690	µA	3.0	
		—	640	840	µA	5.0	
D015B		—	4.7	6.0	mA	3.0	Fosc = 24 MHz 6 MHz EC Oscillator (high power) PLL enabled
D015B		—	4.7	6.1	mA	3.0	Fosc = 24 MHz
		—	5.6	7.4	mA	5.0	6 MHz EC Oscillator (high power) PLL enabled <sup>(5)</sup>
D015C			2.0	2.5	mA	3.0	Fosc = 24 MHz 6 MHz EC Oscillator (high power) PLL enabled, CPU Idle
D015C			2.0	2.5	mA	3.0	Fosc = 24 MHz 6 MHz EC Oscillator (high power) PLL enabled, CPU Idle <sup>(5)</sup>
			2.3	3.0	mA	5.0	

\* These parameters are characterized but not tested.

Legend: TBD = To Be Determined

Note 1: The test conditions for all I<sub>DD</sub> measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V<sub>DD</sub>; MCLR = V<sub>DD</sub>; WDT disabled.

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.

3: For RC oscillator configurations, current through R<sub>EXT</sub> is not included. The current through the resistor can be extended by the formula  $I_R = V_{DD}/2R_{EXT}$  (mA) with R<sub>EXT</sub> in kΩ.

4: FVR and BOR are disabled.

5: 330 nF capacitor on V<sub>USB</sub> pin.

## 27.2 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (Industrial, Extended) (Continued)

PIC18LF1XK50			Standard Operating Conditions (unless otherwise stated)				
Param No.	Device Characteristics	Min	Typ†	Max	Units	Conditions	
						VDD	Note
<b>Supply Current (IDD)<sup>(1, 2)</sup></b>							
D016		—	2.6	3.3	mA	3.0	FOSC = 12 MHz EC Oscillator (high power)
D016		—	2.6	3.3	mA	3.0	FOSC = 12 MHz EC Oscillator (high power) <sup>(5)</sup>
D017		—	1.0	1.3	mA	3.0	FOSC = 12 MHz EC Oscillator (high power) CPU Idle
D017		—	1.0	1.3	mA	3.0	FOSC = 12 MHz EC Oscillator (high power) CPU Idle <sup>(5)</sup>
D017A		—	9	12	mA	3.0	FOSC = 48 MHz 12 MHz EC Oscillator (high power) PLL enabled
D017A		—	8.9	12	mA	3.0	FOSC = 48 MHz 12 MHz EC Oscillator (high power) PLL enabled
D017A		—	11	14	mA	5.0	FOSC = 48 MHz 12 MHz EC Oscillator (high power) PLL enabled <sup>(5)</sup>
D017B		—	3.9	5.0	mA	3.0	FOSC = 48 MHz 12 MHz EC Oscillator (high power) PLL enabled, CPU Idle
D017B		—	3.9	5.0	mA	3.0	FOSC = 48 MHz 12 MHz EC Oscillator (high power) PLL enabled, CPU Idle <sup>(5)</sup>
D017B		—	4.7	6.0	mA	5.0	FOSC = 48 MHz 12 MHz EC Oscillator (high power) PLL enabled, CPU Idle <sup>(5)</sup>
D018		—	19	38	µA	1.8	FOSC = 32 kHz LFINTOSC Oscillator mode <sup>(3, 5)</sup>
D018		—	23	44	µA	3.0	FOSC = 32 kHz LFINTOSC Oscillator mode <sup>(3, 5)</sup>
D018		—	21	40	µA	1.8	FOSC = 32 kHz LFINTOSC Oscillator mode <sup>(3, 5)</sup>
D018		—	25	46	µA	3.0	FOSC = 32 kHz LFINTOSC Oscillator mode <sup>(3, 5)</sup>
D018		—	26	48	µA	5.0	FOSC = 32 kHz LFINTOSC Oscillator mode <sup>(3, 5)</sup>
D019		—	16	33	µA	1.8	FOSC = 32 kHz LFINTOSC Oscillator CPU Idle
D019		—	18	38	µA	3.0	FOSC = 32 kHz LFINTOSC Oscillator CPU Idle
D019		—	18	35	µA	1.8	FOSC = 32 kHz LFINTOSC Oscillator CPU Idle
D019		—	20	40	µA	3.0	FOSC = 32 kHz LFINTOSC Oscillator CPU Idle <sup>(5)</sup>
D019		—	21	42	µA	5.0	FOSC = 32 kHz LFINTOSC Oscillator CPU Idle <sup>(5)</sup>

\* These parameters are characterized but not tested.

Legend: TBD = To Be Determined

Note 1: 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 disabled.

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.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be extended by the formula  $IR = VDD/2REXT$  (mA) with REXT in kΩ.

4: FVR and BOR are disabled.

5: 330 nF capacitor on VUSB pin.

# PIC18F1XK50/PIC18LF1XK50

## 27.2 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (Industrial, Extended) (Continued)

PIC18LF1XK50			Standard Operating Conditions (unless otherwise stated)				
PIC18F1XK50			Standard Operating Conditions (unless otherwise stated)				
Param No.	Device Characteristics	Min	Typ†	Max	Units	Conditions	
						VDD	Note
<b>Supply Current (IDD)<sup>(1, 2)</sup></b>							
D020		—	320	430	µA	1.8	Fosc = 500 kHz LFINTOSC Oscillator
		—	460	600	µA	3.0	
D020		—	350	460	µA	1.8	Fosc = 500 kHz LFINTOSC Oscillator <sup>(5)</sup>
		—	490	630	µA	3.0	
		—	540	710	µA	5.0	
D021		—	380	530	µA	1.8	Fosc = 1 MHz HFINTOSC Oscillator
		—	550	770	µA	3.0	
D021		—	410	530	µA	1.8	Fosc = 1 MHz HFINTOSC Oscillator <sup>(5)</sup>
		—	580	770	µA	3.0	
		—	650	900	µA	5.0	
D021A		—	290	400	µA	1.8	Fosc = 1 MHz HFINTOSC Oscillator CPU Idle
		—	410	560	µA	3.0	
D021A		—	320	420	µA	1.8	Fosc = 1 MHz HFINTOSC Oscillator CPU Idle <sup>(5)</sup>
		—	440	570	µA	3.0	
		—	490	680	µA	5.0	
D022		—	1.2	1.6	mA	1.8	Fosc = 8 MHz HFINTOSC Oscillator
		—	2.1	2.9	mA	3.0	
D022		—	1.2	1.6	mA	1.8	Fosc = 8 MHz HFINTOSC Oscillator <sup>(5)</sup>
		—	2.1	2.9	mA	3.0	
		—	2.4	3.5	mA	5.0	
D023		—	2.0	2.7	mA	1.8	Fosc = 16 MHz HFINTOSC Oscillator
		—	3.5	4.8	mA	3.0	
D023		—	2.0	2.7	mA	1.8	Fosc = 16 MHz HFINTOSC Oscillator <sup>(5)</sup>
		—	3.5	4.8	mA	3.0	
		—	4.0	6.0	mA	5.0	
D023A		—	0.9	1.3	mA	1.8	Fosc = 16 MHz HFINTOSC Oscillator CPU Idle
		—	1.5	2.1	mA	3.0	
D023A		—	0.9	1.3	mA	1.8	Fosc = 16 MHz HFINTOSC Oscillator CPU Idle <sup>(5)</sup>
		—	1.5	2.1	mA	3.0	
		—	1.7	2.6	mA	5.0	

\* These parameters are characterized but not tested.

Legend: TBD = To Be Determined

Note 1: 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 disabled.

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.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be extended by the formula  $IR = VDD/2REXT$  (mA) with REXT in kΩ.

4: FVR and BOR are disabled.

5: 330 nF capacitor on VUSB pin.

# PIC18F1XK50/PIC18LF1XK50

## 27.2 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (Industrial, Extended) (Continued)

PIC18LF1XK50			Standard Operating Conditions (unless otherwise stated)						
PIC18F1XK50			Standard Operating Conditions (unless otherwise stated)						
Param No.	Device Characteristics	Min	Typ†	Max	Units	Conditions		Note	
						V <sub>DD</sub>			
<b>Supply Current (I<sub>DD</sub>)<sup>(1, 2)</sup></b>									
D024		—	0.5	0.7	mA	1.8	FOSC = 4 MHz EXT RC Oscillator mode		
		—	0.9	1.1	mA	3.0			
D024		—	0.5	0.7	mA	1.8	FOSC = 4 MHz EXT RC Oscillator mode <sup>(5)</sup>		
		—	0.9	1.1	mA	3.0			
		—	1.0	1.4	mA	5.0			
D025		—	1.0	1.1	mA	1.8	FOSC = 6 MHz HS Oscillator		
		—	2.1	2.0	mA	3.0			
D025		—	1.0	1.1	mA	1.8	FOSC = 6 MHz HS Oscillator <sup>(5)</sup>		
		—	2.1	2.0	mA	3.0			
		—	3.5	2.5	mA	5.0			
D025A		—	5.4	6.0	mA	3.0	FOSC = 24 MHz 6 MHz HS Oscillator PLL enabled		
D025A		—	5.4	6.0	mA	3.0	FOSC = 24 MHz 6 MHz HS Oscillator PLL enabled <sup>(5)</sup>		
		—	7.4	7.6	mA	5.0			
D026		—	3.2	3.3	mA	3.0	FOSC = 12 MHz HS Oscillator		
D026		—	3.2	3.3	mA	3.0	FOSC = 12 MHz HS Oscillator <sup>(5)</sup>		
		—	4.8	4.2	mA	5.0			
D026A		—	10	12	mA	3.0	FOSC = 48 MHz, 12 MHz HS Oscillator PLL enabled		
D026A		—	10	12	mA	3.0	FOSC = 48 MHz, 12 MHz HS Oscillator PLL enabled <sup>(5)</sup>		
		—	13	15	mA	5.0			

\* These parameters are characterized but not tested.

Legend: TBD = To Be Determined

Note 1: The test conditions for all I<sub>DD</sub> measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V<sub>DD</sub>; MCLR = V<sub>DD</sub>; WDT disabled.

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.

3: For RC oscillator configurations, current through R<sub>EXT</sub> is not included. The current through the resistor can be extended by the formula  $I_R = V_{DD}/2R_{EXT}$  (mA) with R<sub>EXT</sub> in k $\Omega$ .

4: FVR and BOR are disabled.

5: 330 nF capacitor on V<sub>USB</sub> pin.

# PIC18F1XK50/PIC18LF1XK50

## 27.3 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (Power-Down)

PIC18LF1XK50			Standard Operating Conditions (unless otherwise stated)					
PIC18F1XK50			Standard Operating Conditions (unless otherwise stated)					
Param No.	Device Characteristics	Min	Typ†	Max +85°C	Max +125°C	Units	Conditions	
							VDD	Note
<b>Power-down Base Current (IPD)<sup>(2)</sup></b>								
D027		—	0.024	0.7	6.7	μA	1.8	WDT, BOR, FVR, Voltage Regulator and T1OSC disabled, all Peripherals Inactive
		—	0.078	1.9	8.5	μA	3.0	
D027		—	6.0	7.0	13	μA	1.8	WDT, BOR, FVR and T1OSC disabled, all Peripherals Inactive
		—	7.0	10	15	μA	3.0	
		—	8.0	12	19	μA	5.0	
<b>Power-down Module Current</b>								
D028		—	0.45	1.3	4.4	μA	1.8	LPWDT Current <sup>(1)</sup>
		—	0.75	2.0	6.0	μA	3.0	
D028		—	6.5	7.0	10.5	μA	1.8	LPWDT Current <sup>(1)</sup>
		—	9.6	10.6	17.6	μA	3.0	
		—	10.5	16.5	20	μA	5.0	
D029		—	12	17	23	μA	1.8	FVR current <sup>(3)</sup>
		—	22	19	25	μA	3.0	
D029		—	28	42	50	μA	1.8	FVR current <sup>(3, 5)</sup>
		—	35.6	45.6	55	μA	3.0	
		—	38.5	49	60	mA	5.0	
D030		—	—	—	—	μA	1.8	BOR Current <sup>(1, 3)</sup>
		—	—	21	27	μA	3.0	
D030		—	—	—	—	μA	1.8	BOR Current <sup>(1, 3, 5)</sup>
		—	27	48	51	μA	3.0	
		—	36.5	51	55	μA	5.0	
D031		—	0.79	3.6	5.3	μA	1.8	T1OSC Current <sup>(1)</sup>
		—	1.8	2.9	6.9	μA	3.0	
D031		—	8.0	7.5	10	μA	1.8	T1OSC Current <sup>(1)</sup>
		—	8.5	10.5	15	μA	3.0	
		—	10.5	12.5	24	μA	5.0	

**Legend:** TBD = To Be Determined

\* These parameters are characterized but not tested.

† 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:** The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral  $\Delta$  current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.

**2:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD.

**3:** Fixed Voltage Reference is automatically enabled whenever the BOR is enabled

**4:** A/D oscillator source is FRC

**5:** 0.1  $\mu$ s capacitor on VCAP (RA0).

# PIC18F1XK50/PIC18LF1XK50

## 27.3 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (Power-Down) (Continued)

PIC18LF1XK50			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.	Device Characteristics	Min	Typ	Max +85°C	Max +125°C	Units	Conditions	
							VDD	Note
<b>Power-down Module Current</b>								
D032		—	—	1.8	8	μA	1.8	A/D Current <sup>(1,4)</sup> , no conversion in progress
		—	—	3	10	μA	3.0	
D032		—	—	6	12	μA	1.8	A/D Current <sup>(1,3)</sup> , no conversion in progress
		—	—	10	17	μA	3.0	
		—	—	11.5	22	μA	5.0	
D033		—	—	38	44	μA	1.8	Comparator Current, low power
		—	—	40	47	μA	3.0	
D033		—	30	40	49	μA	2.0	Comparator Current, low power
		—	34	44	53	μA	3.0	
		—	36	50	60	μA	5.0	
D033A		—	—	239	244	μA	1.8	Comparator Current, high power
		—	—	242	249	μA	3.0	
D033A		—	144	243	250	μA	2.0	Comparator Current, high power
		—	146	247	256	μA	3.0	
		—	151	253	264	μA	5.0	
D034		—	—	18	23	μA	1.8	Voltage Reference Current
		—	—	30	35	μA	3.0	
D034		—	35	36	44	μA	2.0	Voltage Reference Current
		—	43	44	60	μA	3.0	
		—	55	65	74	μA	5.0	

**Legend:** TBD = To Be Determined

\* These parameters are characterized but not tested.

† 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:** The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.

**2:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD.

**3:** Fixed Voltage Reference is automatically enabled whenever the BOR is enabled

**4:** A/D oscillator source is FRC

**5:** 0.1 μs capacitor on VCAP (RA0).

# PIC18F1XK50/PIC18LF1XK50

## 27.4 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E

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.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D036 D036A D037 D038 D039A	V <sub>IL</sub>	<b>Input Low Voltage</b>					
		I/O PORT: with TTL buffer	—	—	0.8	V	$4.5\text{V} \leq \text{VDD} \leq 5.5\text{V}$
			—	—	0.15 V <sub>DD</sub>	V	$1.8\text{V} \leq \text{VDD} \leq 4.5\text{V}$
		with Schmitt Trigger buffer with I <sup>2</sup> C levels	—	—	0.2 V <sub>DD</sub>	V	$1.8\text{V} \leq \text{VDD} \leq 5.5\text{V}$
		MCLR, OSC1 (RC mode) <sup>(1)</sup>	—	—	0.3 V <sub>DD</sub>	V	
		OSC1 (HS mode)	—	—	0.2 V <sub>DD</sub>	V	
			—	—	0.3 V <sub>DD</sub>	V	
D040 D040A D041 D042 D043A D043B	V <sub>IH</sub>	<b>Input High Voltage</b>					
		I/O ports: with TTL buffer	2.0	—	—	V	$4.5\text{V} \leq \text{VDD} \leq 5.5\text{V}$
			0.25 V <sub>DD</sub> + 0.8	—	—	V	$1.8\text{V} \leq \text{VDD} \leq 4.5\text{V}$
		with Schmitt Trigger buffer with I <sup>2</sup> C levels	0.8 V <sub>DD</sub>	—	—	V	$1.8\text{V} \leq \text{VDD} \leq 5.5\text{V}$
		MCLR	0.7 V <sub>DD</sub>	—	—	V	
		OSC1 (HS mode)	0.8 V <sub>DD</sub>	—	—	V	
		OSC1 (RC mode)	0.7 V <sub>DD</sub>	—	—	V	
D060 D061 D063	I <sub>IL</sub>	<b>Input Leakage Current<sup>(2)</sup></b>					
		I/O ports	—	$\pm 5$	$\pm 100$	nA	$\text{Vss} \leq \text{VPIN} \leq \text{VDD}$ , Pin at high-impedance
		MCLR <sup>(3)</sup>	—	$\pm 50$	$\pm 200$	nA	$\text{Vss} \leq \text{VPIN} \leq \text{VDD}$
D070*	I <sub>PUR</sub>	<b>PORTB Weak Pull-up Current</b>					
			50	250	400	μA	$\text{VDD} = 5.0\text{V}$ , $\text{VPIN} = \text{Vss}$
D080	V <sub>OL</sub>	<b>Output Low Voltage<sup>(4)</sup></b>					
		I/O ports	—	—	V <sub>ss</sub> +0.6	V	$\text{IOH} = 8\text{mA}$ , $\text{VDD} = 5\text{V}$ $\text{IOH} = 6\text{mA}$ , $\text{VDD} = 3.3\text{V}$ $\text{IOH} = 3\text{mA}$ , $\text{VDD} = 1.8\text{V}$
D090	V <sub>OH</sub>	<b>Output High Voltage<sup>(4)</sup></b>					
		I/O ports	V <sub>DD</sub> -0.7	—	—	V	$\text{IOL} = 3.5\text{mA}$ , $\text{VDD} = 5\text{V}$ $\text{IOL} = 3\text{mA}$ , $\text{VDD} = 3.3\text{V}$ $\text{IOL} = 2\text{mA}$ , $\text{VDD} = 1.8\text{V}$

**Legend:** TBD = To Be Determined

\* These parameters are characterized but not tested.

† 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:** In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended to use an external clock in RC mode.

**2:** Negative current is defined as current sourced by the pin.

**3:** 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.

**4:** Including OSC2 in CLKOUT mode.

# PIC18F1XK50/PIC18LF1XK50

## 27.4 DC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E (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.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
D101*	COSC2	Capacitive Loading Specs on Output Pins					
		OSC2 pin	—	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
D101A*	CIO	All I/O pins	—	—	50	pF	
D130	EP	Flash Memory					
		Cell Endurance	10K 100K	—	—	E/W	Program Flash Memory Data Flash Memory
D131		VDD for Read	V <sub>MIN</sub>	—	—	V	
		Voltage on MCLR/VPP during Erase/Program	V <sub>DD</sub> + 1.5	—	9.0	V	Temperature during programming: $40^{\circ}\text{C} \leq \text{TA} \leq 85^{\circ}\text{C}$
		VDD for Bulk Erase	TBD	2.1	—	V	Temperature during programming: $10^{\circ}\text{C} \leq \text{TA} \leq 40^{\circ}\text{C}$
D132	VPEW	VDD for Write or Row Erase	V <sub>MIN</sub>	—	—		V <sub>MIN</sub> = Minimum operating voltage V <sub>MAX</sub> = Maximum operating voltage
	IPPPGM	Current on MCLR/VPP during Erase/Write	—	—	5.0	mA	
	IDDPGM	Current on VDD during Erase/Write	—	—	5.0	mA	
D133	TPEW	Erase/Write cycle time	—	—	4.0	ms	
D134	TRETD	Characteristic Retention	40°	—	—	Year	Provided no other specifications are violated
D135	D135A	VCAP Capacitor Charging					
		Charging current	—	200	—	μA	
		Source/sink capability when charging complete	—	0.0	—	mA	

**Legend:** TBD = To Be Determined

\* These parameters are characterized but not tested.

† 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:** In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended to use an external clock in RC mode.

**2:** Negative current is defined as current sourced by the pin.

**3:** 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.

**4:** Including OSC2 in CLKOUT mode.

ADV

# PIC18F1XK50/PIC18LF1XK50

## 27.5 USB Module Specifications

Operating Conditions-40°C ≤ TA ≤ +85°C (unless otherwise state)

Param No.	Sym	Characteristic	Min	Typ	Max	Units	Conditions
D313	VUSB	USB Voltage	3.0	—	3.6	V	Voltage on VUSB pin must be in this range for proper USB operation
D314	IIL	Input Leakage on pin	—	—	± 1	µA	Vss ≤ VPIN ≤ VDD pin at high impedance
D315	VILUSB	Input Low Voltage for USB Buffer	—	—	0.8	V	For VUSB range
D316	VIHUSB	Input High Voltage for USB Buffer	2.0	—	—	V	For VUSB range
D318	VDIFS	Differential Input Sensitivity	—	—	0.2	V	The difference between D+ and D- must exceed this value while VCM is met
D319	VCM	Differential Common Mode Range	0.8	—	2.5	V	
D320	ZOUT	Driver Output Impedance <sup>(1)</sup>	28	—	44	Ω	
D321	VOL	Voltage Output Low	0.0	—	0.3	V	1.5 kΩ load connected to 3.6V
D322	VOH	Voltage Output High	2.8	—	3.6	V	1.5 kΩ load connected to ground

**Note 1:** The D+ and D- signal lines have been built-in impedance matching resistors. No external resistors, capacitors or magnetic components are necessary on the D+/D- signal paths between the PIC18F1XK50/PIC18LF1XK50 family device and USB cable.

ADVANCE INFORMATION

## 27.6 Thermal Considerations

Standard Operating Conditions (unless otherwise stated)					
Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$					
Param No.	Sym	Characteristic	Typ	Units	Conditions
TH01	$\theta_{JA}$	Thermal Resistance Junction to Ambient	62.4	$^{\circ}\text{C/W}$	20-pin PDIP package
			85.2	$^{\circ}\text{C/W}$	20-pin SOIC package
			108.1	$^{\circ}\text{C/W}$	20-pin SSOP package
			TBD	$^{\circ}\text{C/W}$	20-pin QFN 5x5mm package
TH02	$\theta_{JC}$	Thermal Resistance Junction to Case	31.4	$^{\circ}\text{C/W}$	20-pin PDIP package
			24	$^{\circ}\text{C/W}$	20-pin SOIC package
			24	$^{\circ}\text{C/W}$	20-pin SSOP package
			24	$^{\circ}\text{C/W}$	20-pin QFN 6x6mm package
TH03	TJMAX	Maximum Junction Temperature	150	$^{\circ}\text{C}$	
TH04	PD	Power Dissipation	—	W	$PD = P_{INTERNAL} + P_{I/O}$
TH05	PINTERNAL	Internal Power Dissipation	—	W	$P_{INTERNAL} = IDD \times V_{DD}^{(1)}$
TH06	Pi/O	I/O Power Dissipation	—	W	$P_{I/O} = \sum (I_{OL} \times V_{OL}) + \sum (I_{OH} \times (V_{DD} - V_{OH}))$
TH07	PDER	Derated Power	—	W	$P_{DER} = PD_{MAX} (TJ - TA) / \theta_{JA}^{(2)}$

Legend: TBD = To Be Determined

Note 1: IDD is current to run the chip alone without driving any load on the output pins.

2: TA = Ambient Temperature

3: TJ = Junction Temperature

ADVANCE INFORMATION

# PIC18F1XK50/PIC18LF1XK50

## 27.7 Timing Parameter Symbology

The timing parameter symbols have been created with one of the following formats:

1.  $T_{ppS2}T_{ppS}$

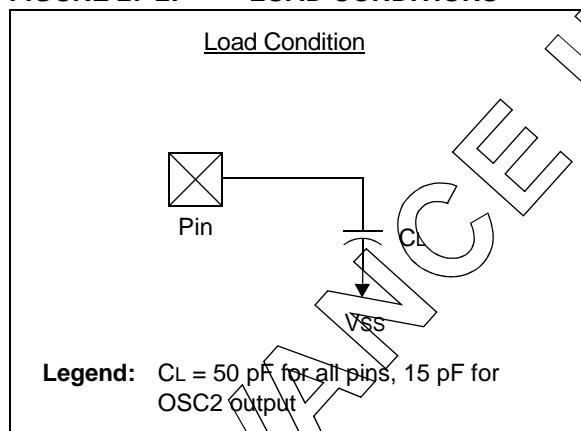
2.  $T_{ppS}$

T		T	Time
F	Frequency		
Lowercase letters (pp) and their meanings:			
<b>pp</b>			
cc	CCP1	osc	OSC1
ck	CLKOUT	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O PORT	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

Uppercase letters and their meanings:

S		P	Period
F	Fall	R	Rise
H	High	V	Valid
I	Invalid (High-impedance)	Z	High-impedance
L	Low		

FIGURE 27-2: LOAD CONDITIONS



# PIC18F1XK50/PIC18LF1XK50

## 27.8 AC Characteristics: PIC18F1XK50/PIC18LF1XK50-I/E

FIGURE 27-3: CLOCK TIMING

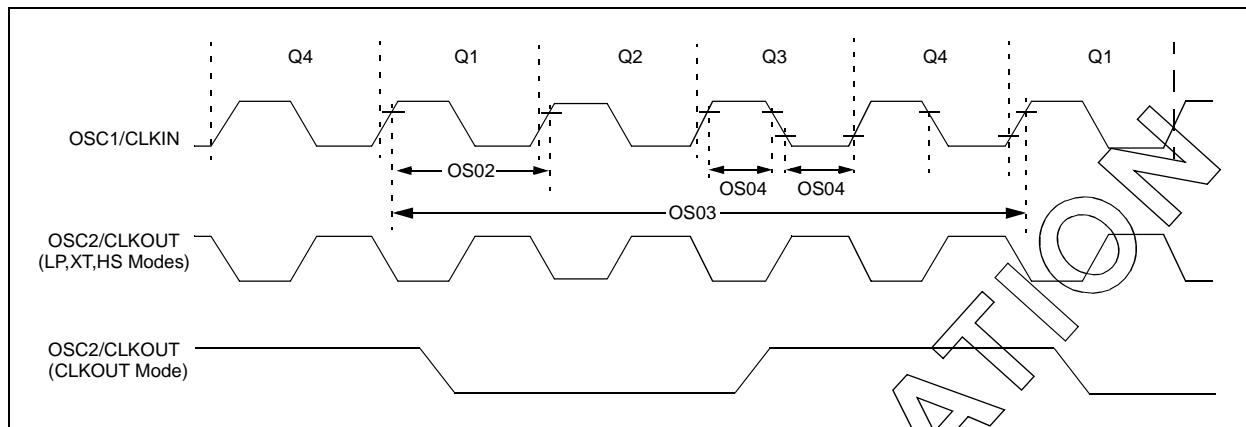
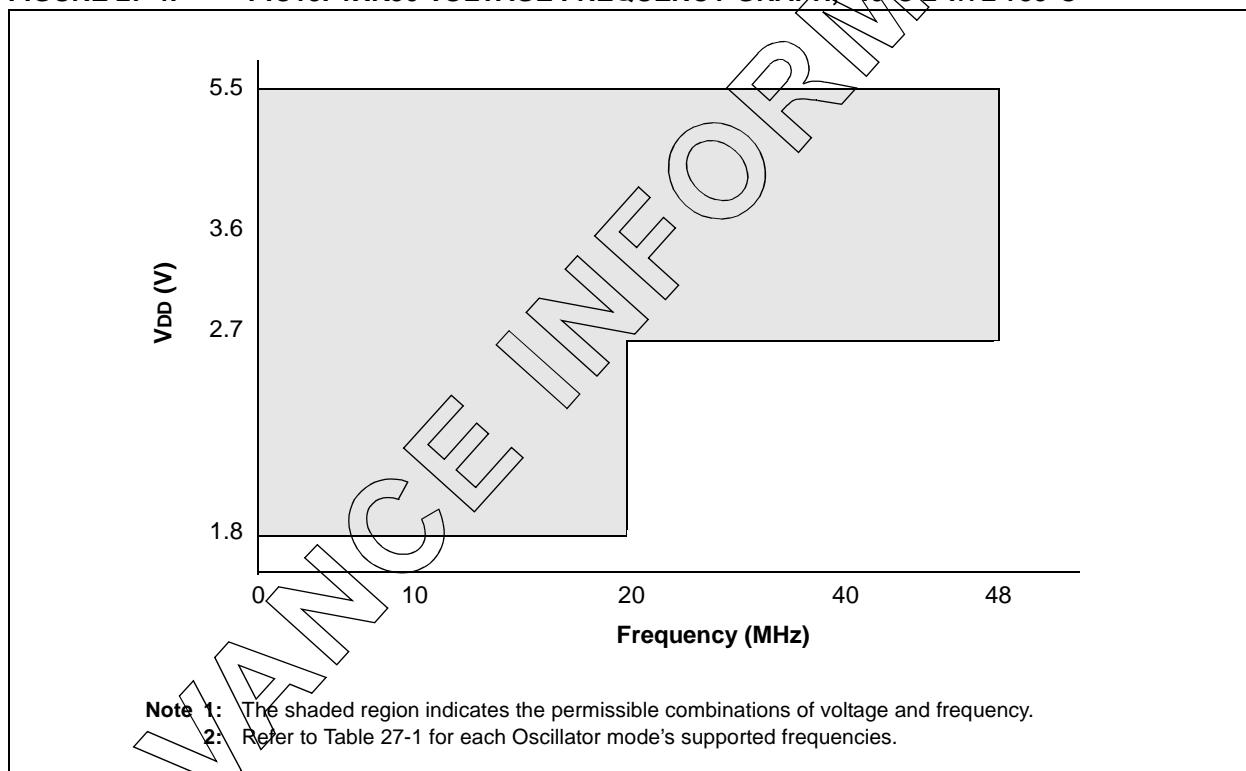


FIGURE 27-4: PIC18F1XK50 VOLTAGE FREQUENCY GRAPH,  $-40^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$



# PIC18F1XK50/PIC18LF1XK50

FIGURE 27-5: PIC18LF1XK50 VOLTAGE FREQUENCY GRAPH,  $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$

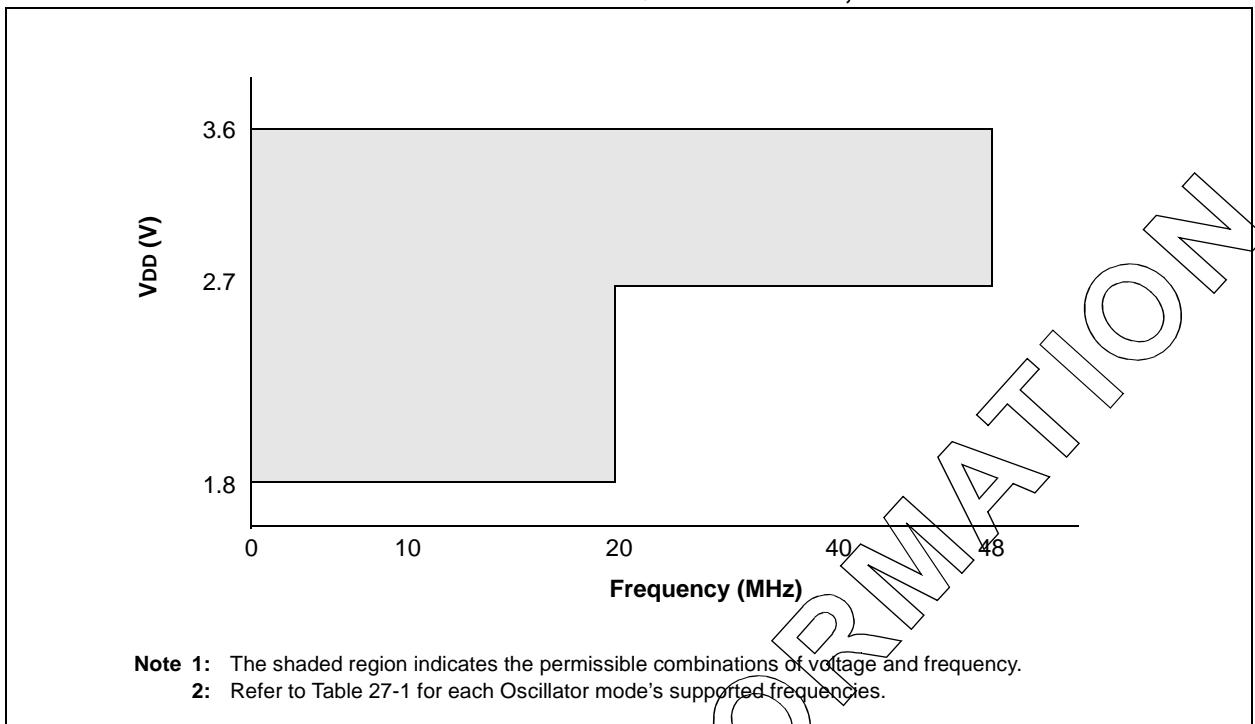
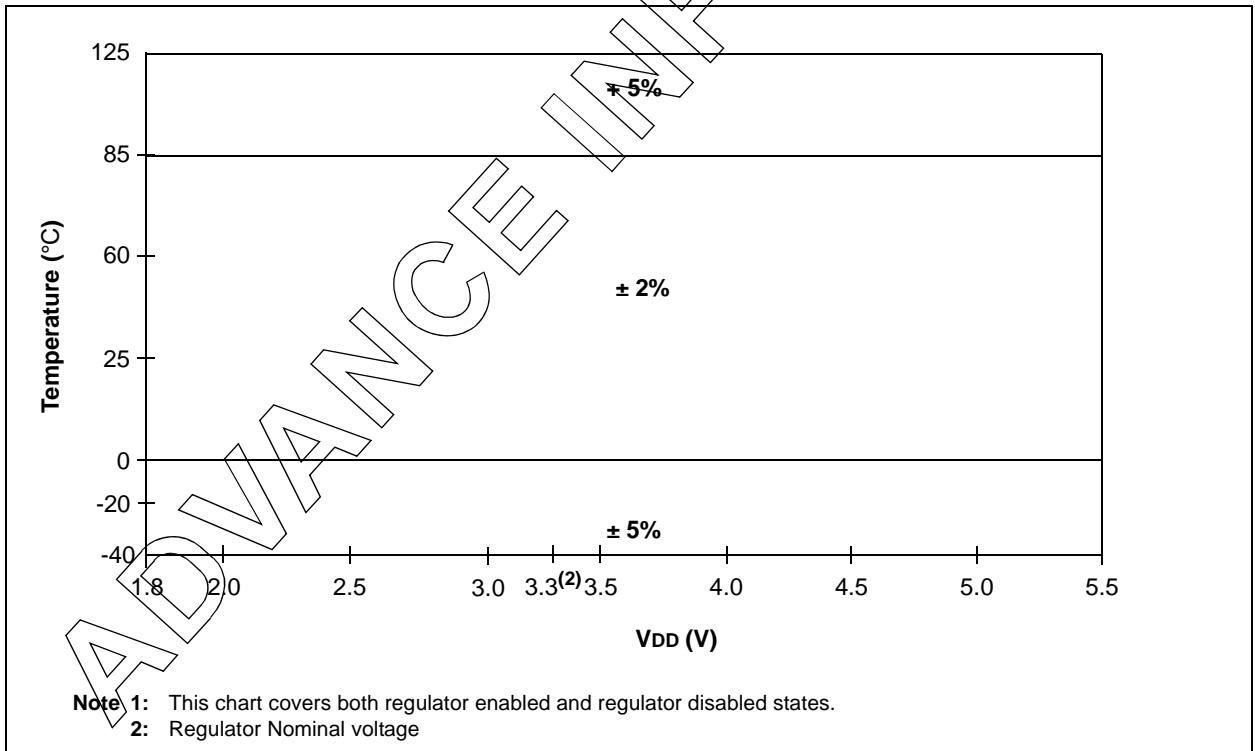


FIGURE 27-6: HFINTOSC FREQUENCY ACCURACY OVER DEVICE  $\text{V}_{\text{DD}}$  AND TEMPERATURE



# PIC18F1XK50/PIC18LF1XK50

TABLE 27-1: CLOCK OSCILLATOR TIMING REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)

Operating temperature  $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
OS01	FOSC	External CLKIN Frequency <sup>(1)</sup>	DC	—	37	kHz	EC Oscillator mode (low)
			DC	—	4	MHz	EC Oscillator mode (medium)
			DC	—	48	MHz	EC Oscillator mode (high)
		Oscillator Frequency <sup>(1)</sup>	—	32.768	33	kHz	LP Oscillator mode
			0.1	—	4	MHz	XT Oscillator mode
			1	—	20	MHz	HS Oscillator mode
OS02	TOSC	External CLKIN Period <sup>(1)</sup>	27	—	$\infty$	$\mu\text{s}$	LP Oscillator mode
			250	—	$\infty$	ns	XT Oscillator mode
			50	—	$\infty$	ns	HS Oscillator mode
			20.80	—	$\infty$	ns	EC Oscillator mode
		Oscillator Period <sup>(1)</sup>	—	30.5	—	$\mu\text{s}$	LP Oscillator mode
			250	—	10,000	ns	XT Oscillator mode
			50	—	1,000	ns	HS Oscillator mode
			250	—	—	ns	RC Oscillator mode
OS03	T <sub>CY</sub>	Instruction Cycle Time <sup>(1)</sup>	83	T <sub>CY</sub>	DC	ns	T <sub>CY</sub> = 4/FOSC
OS04*	TosH, TosL	External CLKIN High, External CLKIN Low	2	—	—	$\mu\text{s}$	LP oscillator
			100	—	—	ns	XT oscillator
			20	—	—	ns	HS oscillator
OS05*	TosR, TosF	External CLKIN Rise, External CLKIN Fall	0	—	$\infty$	ns	LP oscillator
			0	—	$\infty$	ns	XT oscillator
			0	—	$\infty$	ns	HS oscillator

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Instruction cycle period (T<sub>CY</sub>) equals four times the input oscillator time base period. 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 OSC1 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

ADVANCED

# PIC18F1XK50/PIC18LF1XK50

TABLE 27-2: OSCILLATOR PARAMETERS

Standard Operating Conditions (unless otherwise stated) Operating Temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$								
Param No.	Sym	Characteristic	Freq. Tolerance	Min	Typ†	Max	Units	Conditions
OS08	HFosc	Internal Calibrated HFINTOSC Frequency <sup>(2)</sup>	$\pm 2\%$ $\pm 5\%$	— —	16.0 16.0	— —	MHz MHz	$0^{\circ}\text{C} \leq \text{TA} \leq +85^{\circ}\text{C}$ $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$
OS10*	Tiosc ST	HFINTOSC Wake-up from Sleep Start-up Time	—	—	5	7	μs	$\text{VDD} = 2.0\text{V}, -40^{\circ}\text{C} \text{ to } +85^{\circ}\text{C}$
			—	—	5	7	μs	$\text{VDD} = 3.0\text{V}, -40^{\circ}\text{C} \text{ to } +85^{\circ}\text{C}$
			—	—	5	7	μs	$\text{VDD} = 5.0\text{V}, -40^{\circ}\text{C} \text{ to } +85^{\circ}\text{C}$

\* These parameters are characterized but not tested.

† 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:** Instruction cycle period (TCY) equals four times the input oscillator time base period. 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 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

**2:** To ensure these oscillator frequency tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1  $\mu\text{F}$  and 0.01  $\mu\text{F}$  values in parallel are recommended.

**3:** By design.

TABLE 27-3: PLL CLOCK TIMING SPECIFICATIONS (VDD = 4.27V TO 5.5V)

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
F10	FOSC	Oscillator Frequency Range	4	—	12	MHz	
F11	FSYS	On-Chip VCO System Frequency	16	—	48	MHz	
F12	t <sub>rc</sub>	PLL Start-up Time (Lock Time)	—	—	2	ms	
F13*	$\Delta\text{CLK}$	CLKOUT Stability (Jitter)	—	-0.25%	—	+0.25%	%

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 27-7: CLKOUT AND I/O TIMING

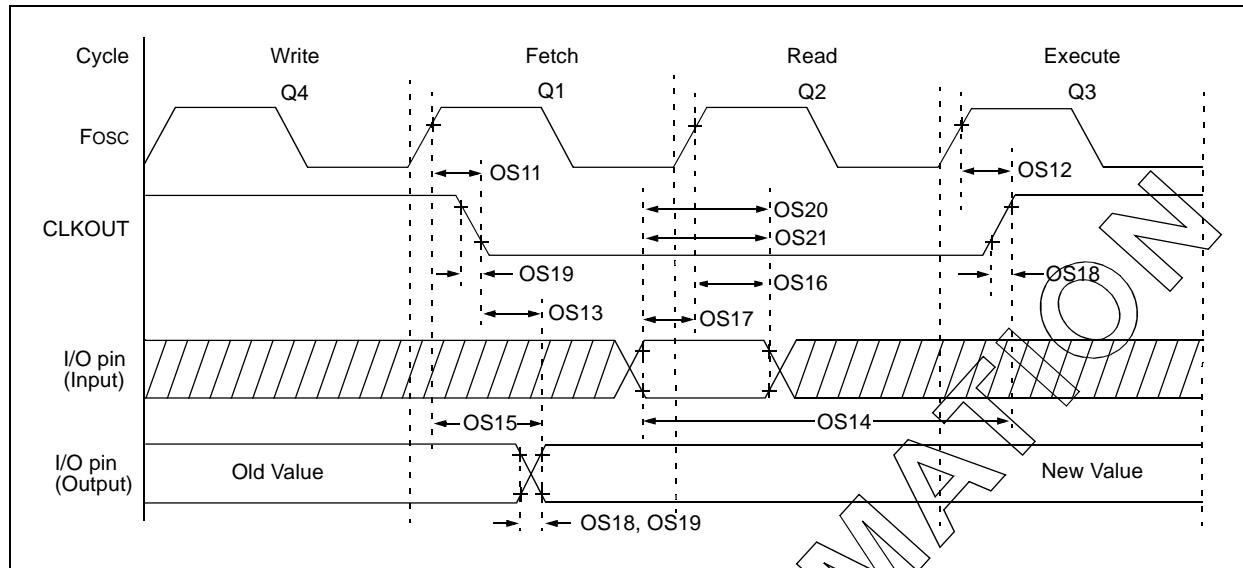


TABLE 27-4: CLKOUT AND I/O TIMING PARAMETERS

Param No.	Sym	Characteristic	Min	Typt	Max	Units	Conditions
OS11	TosH2ckL	Fosc $\uparrow$ to CLKOUT $\downarrow$ <sup>(1)</sup>	—	—	70	ns	VDD = 3.3-5.0V
OS12	TosH2ckH	Fosc $\uparrow$ to CLKOUT $\uparrow$ <sup>(1)</sup>	—	—	72	ns	VDD = 3.3-5.0V
OS13	TckL2ioV	CLKOUT $\downarrow$ to Port out valid <sup>(1)</sup>	—	—	20	ns	
OS14	TioV2ckH	Port input valid before CLKOUT $\uparrow$ <sup>(1)</sup>	Tosc + 200 ns	—	—	ns	
OS15	TosH2ioV	Fosc $\uparrow$ (Q1 cycle) to Port out valid	—	50	70*	ns	VDD = 3.3-5.0V
OS16	TosH2iol	Fosc $\uparrow$ (Q2 cycle) to Port input invalid (I/O in hold time)	50	—	—	ns	VDD = 3.3-5.0V
OS17	TioV2osH	Port input valid to Fosc $\uparrow$ (Q2 cycle) (I/O in setup time)	20	—	—	ns	
OS18	TioR	Port output rise time <sup>(2)</sup>	—	40	72	ns	VDD = 2.0V VDD = 3.3-5.0V
OS19	TioF	Port output fall time <sup>(2)</sup>	—	15	32	ns	VDD = 2.0V VDD = 3.3-5.0V
OS20*	Tinp	INT pin input high or low time	25	—	—	ns	
OS21*	Trbp	PORTB interrupt-on-change new input level time	Tcy	—	—	ns	

\* These parameters are characterized but not tested.

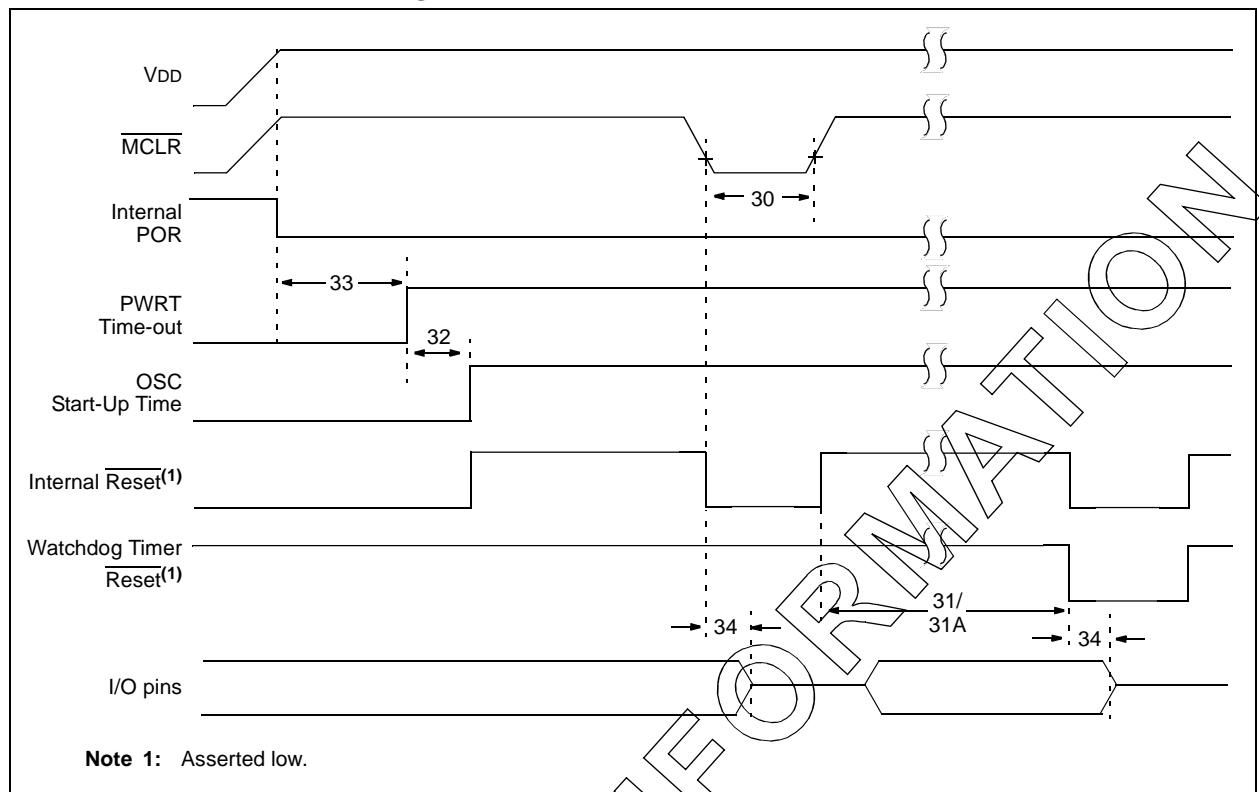
† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated.

Note 1: Measurements are taken in RC mode where CLKOUT output is 4 x Tosc.

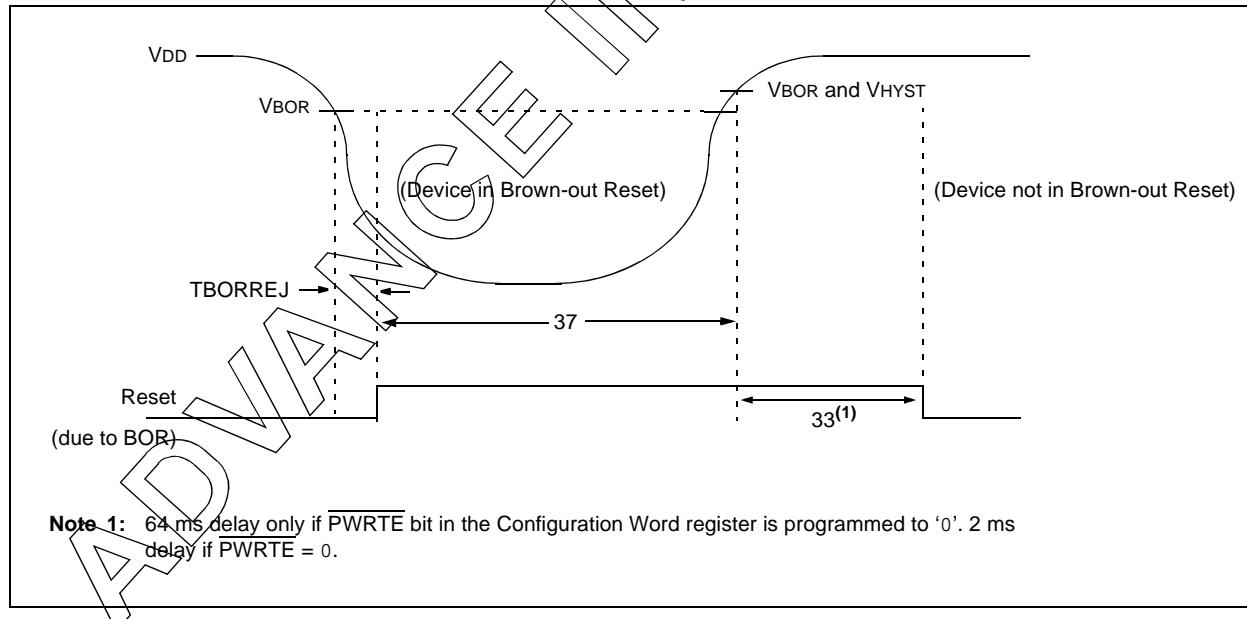
2: Includes OSC2 in CLKOUT mode.

# PIC18F1XK50/PIC18LF1XK50

**FIGURE 27-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**



**FIGURE 27-9: BROWN-OUT RESET TIMING AND CHARACTERISTICS**



# PIC18F1XK50/PIC18LF1XK50

**TABLE 27-5: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, AND BROWN-OUT RESET PARAMETERS**

Standard Operating Conditions (unless otherwise stated) Operating Temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$							
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
30	TMC <sub>L</sub>	MCLR Pulse Width (low)	2 5	— —	— —	μs μs	VDD = 3.3-5V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ VDD = 3.3-5V
31	TWDT	Standard Watchdog Timer Time-out Period (No Prescaler) <sup>(5)</sup>	10 10	17 17	27 30	ms ms	VDD = 3.3V-5V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ VDD = 3.3V-5V
31A	TWDTLP	Low Power Watchdog Timer Time-out Period (No Prescaler)	10 10	18 18	27 33	ms ms	VDD = 3.3V-5V, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ VDD $\geq$ 3.3V-5V
32	TOST	Oscillator Start-up Timer Period <sup>(1), (2)</sup>	—	1024	—	Tosc	(Note 3)
33*	TPWRT	Power-up Timer Period, PWRTE = 0	40	65	140	ms	
34*	TIOZ	I/O high-impedance from MCLR Low or Watchdog Timer Reset	—	—	2.0	μs	
35	VBOR	Brown-out Reset Voltage	TBD TBD TBD TBD	1.9 2.2 2.7 3.0	TBD TBD TBD TBD	V V V V	BORV = 1.9V BORV = 2.2V BORV = 2.7V BORV = 3.6V
36*	VHYST	Brown-out Reset Hysteresis	25	50	75	mV	$-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$
37*	TBORDC	Brown-out Reset DC Response Time	1	3	5 10	μs	VDD $\leq$ VBOR, $-40^{\circ}\text{C}$ to $+85^{\circ}\text{C}$ VDD $\leq$ VBOR

**Legend:** TBD = To Be Determined

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Instruction cycle period (TCY) equals four times the input oscillator time base period. 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 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

2: By design.

3: Period of the slower clock.

4: To ensure these voltage tolerances, VDD and Vss must be capacitively decoupled as close to the device as possible. 0.1  $\mu\text{F}$  and 0.01  $\mu\text{F}$  values in parallel are recommended.

5: Design Target. If unable to meet this target, the maximum can be increased, but the minimum cannot be changed.

ADV  
C

# PIC18F1XK50/PIC18LF1XK50

FIGURE 27-10: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

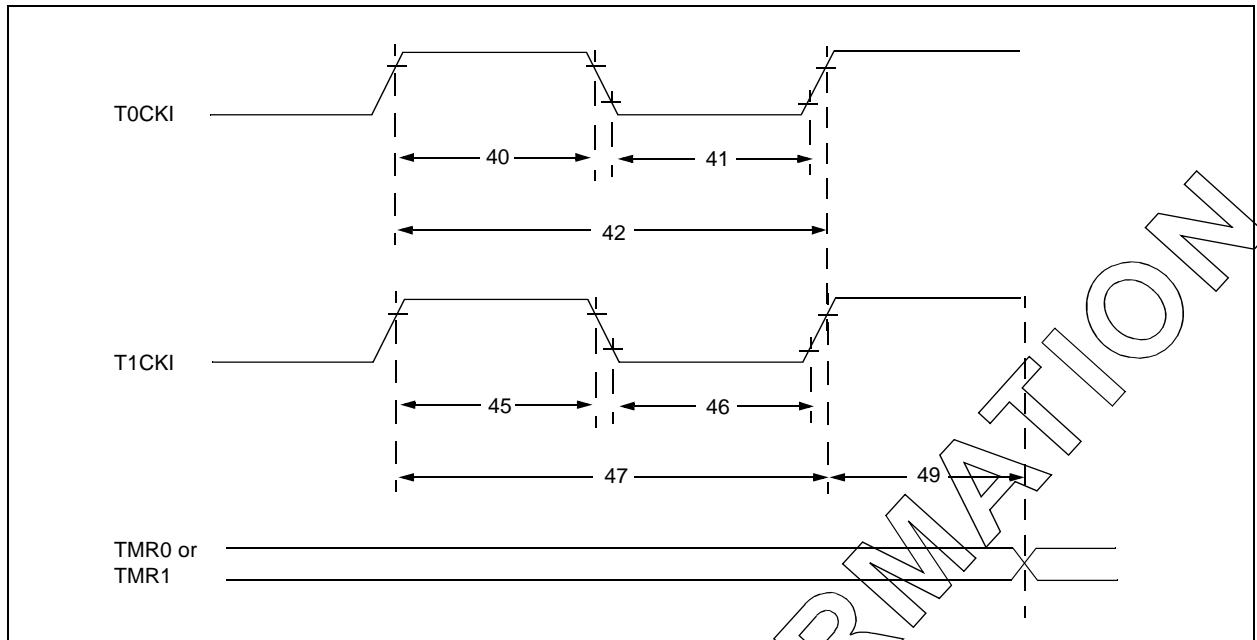


TABLE 27-6: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Standard Operating Conditions (unless otherwise stated) Operating Temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$								
Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
40*	TT0H	T0CKI High Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
41*	TT0L	T0CKI Low Pulse Width	No Prescaler	0.5 TCY + 20	—	—	ns	
			With Prescaler	10	—	—	ns	
42*	TT0P	T0CKI Period		Greater of: 20 or $\frac{\text{TCY} + 40}{\text{N}}$	—	—	ns	N = prescale value (2, 4, ..., 256)
45*	TT1H	T1CKI High Time	Synchronous, No Prescaler	0.5 TCY + 20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
46*	TT1L	T1CKI Low Time	Synchronous, No Prescaler	0.5 TCY + 20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
47*	TT1P	T1CKI Input Period	Synchronous	Greater of: 30 or $\frac{\text{TCY} + 40}{\text{N}}$	—	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous	60	—	—	ns	
48	Fr1	Timer1 Oscillator Input Frequency Range (oscillator enabled by setting bit T1OSCEN)		32.4	32.768	33.1	kHz	
49*	TCKEZTMR1	Delay from External Clock Edge to Timer Increment		2 Tosc	—	7 Tosc	—	Timers in Sync mode

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

# PIC18F1XK50/PIC18LF1XK50

FIGURE 27-11: CAPTURE/COMPARE/PWM TIMINGS (CCP)

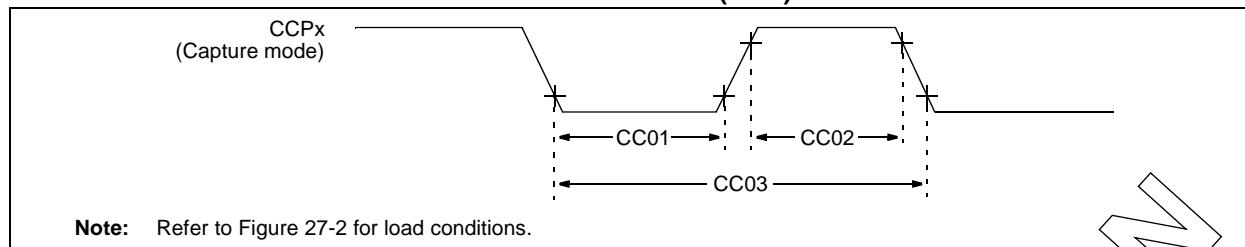


TABLE 27-7: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)

Standard Operating Conditions (unless otherwise stated)								Operating Temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$
Param No.	Sym	Characteristic		Min	Typ†	Max	Units	Conditions
CC01*	Tccl	CCPx Input Low Time	No Prescaler	0.5TCY + 20	—	—	ns	
			With Prescaler	20	—	—	ns	
CC02*	TccH	CCPx Input High Time	No Prescaler	0.5TCY + 20	—	—	ns	N = prescale value (1, 4 or 16)
			With Prescaler	20	—	—	ns	
CC03*	TccP	CCPx Input Period		$3\text{TCY} + 40$ N	—	—	ns	N = prescale value (1, 4 or 16)

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

TABLE 27-8: PIC18F1XK50/PIC18LF1XK50 A/D CONVERTER (ADC) CHARACTERISTICS:

Standard Operating Conditions (unless otherwise stated)							
Operating temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$							
Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
AD01	NR	Resolution	—	—	10	bit	
AD02	EIL	Integral Error	—	—	$\pm 1$	LSb	VREF = 5.0V
AD03	EDL	Differential Error	—	—	$\pm 1$	LSb	No missing codes VREF = 5.0V
AD04	EOFF	Offset Error	—	—	$\pm 3$	LSb	VREF = 5.0V
AD05	EGN	Gain Error	—	—	$\pm 3$	LSb	VREF = 5.0V
AD06	VREF	Reference Voltage <sup>(3)</sup>	1.8	—	VDD	V	VDD < 3.0V
AD06A			3.0	—	VDD	V	VDD < 3.0V
AD07	VAIN	Full-Scale Range	Vss	—	VREF	V	
AD08	ZAIN	Recommended Impedance of Analog Voltage Source	—	—	2.5	kΩ	Can go higher if external 0.01μF capacitor is present on input pin.
AD09*	IREF	VREF Input Current <sup>(3)</sup>	10	—	1000	μA	During VAIN acquisition. Based on differential of VHOLD to VAIN.
			—	—	10	μA	During A/D conversion cycle.

\* These parameters are characterized but not tested.

† 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: Total Absolute Error includes integral, differential, offset and gain errors.

2: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

3: ADC VREF is from external VREF, VDD pin or FVR, whichever is selected as reference input.

4: When ADC is off, it will not consume any current other than leakage current. The power-down current specification includes any such leakage from the ADC module.

# PIC18F1XK50/PIC18LF1XK50

FIGURE 27-12: A/D CONVERSION TIMING

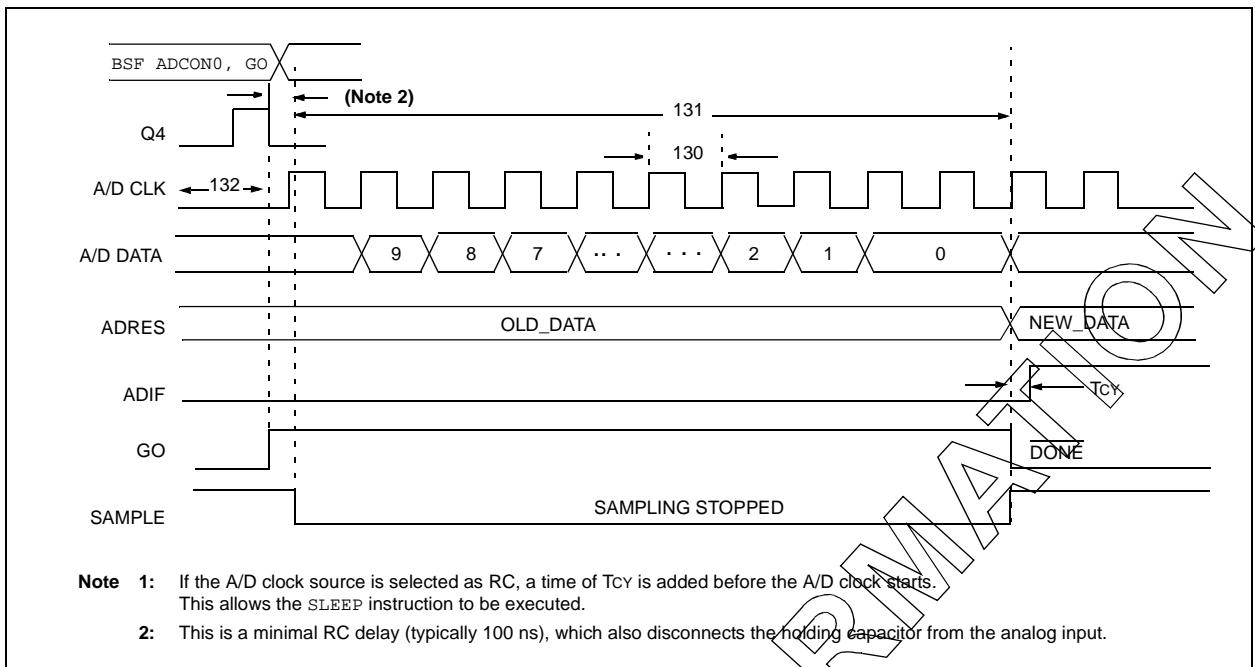


TABLE 27-9: A/D CONVERSION REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
130	TAD	A/D Clock Period	0.7	25.0 <sup>(1)</sup>	μs	Tosc based, VREF $\geq$ 3.0V
			TBD	1	μs	A/D RC mode
131	TCNV	Conversion Time (not including acquisition time) <sup>(2)</sup>	11	12	TAD	
132	TACQ	Acquisition Time <sup>(3)</sup>	1.4	—	μs	-40°C to +85°C
135	Tswc	Switching Time from Convert to Sample	—	(Note 4)	μs	0°C $\leq$ to $\leq$ +85°C
TBD	TDIS	Discharge Time	0.2	—	μs	

**Legend:** TBD = To Be Determined

**Note 1:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

**2:** ADRES register may be read on the following  $T_{CY}$  cycle.

**3:** The time for the holding capacitor to acquire the "New" input voltage when the voltage changes full scale after the conversion (VDD to Vss or Vss to VDD). The source impedance ( $R_s$ ) on the input channels is 50  $\Omega$ .

**4:** On the following cycle of the device clock.

# PIC18F1XK50/PIC18LF1XK50

**TABLE 27-10: COMPARATOR SPECIFICATIONS**

**Operating Conditions:**  $1.8V < VDD < 3.6V$ ,  $-40^\circ C < TA < +125^\circ C$  (unless otherwise stated).

Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
CM01	VIOFF	Input Offset Voltage	—	$\pm 7.5$	$\pm 15$	mV	
CM02	VICM	Input Common Mode Voltage	0	—	VDD	V	
CM03	CMRR	Common Mode Rejection Ratio	55	—	—	dB	
CM04	TRESP	Response Time	—	150	400	ns	<b>Note 1</b>
CM05	TMC2OV	Comparator Mode Change to Output Valid*	—	—	10	$\mu s$	

\* These parameters are characterized but not tested.

**Note 1:** Response time measured with one comparator input at  $VDD/2$ , while the other input transitions from  $Vss$  to  $VDD$ .

**TABLE 27-11: CVREF VOLTAGE REFERENCE SPECIFICATIONS**

**Operating Conditions:**  $1.8V < VDD < 3.6V$ ,  $-40^\circ C < TA < +125^\circ C$  (unless otherwise stated).

Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
CV01*	CLSB	Step Size <sup>(2)</sup>	—	$VDD/24$ $VDD/32$	—	V	Low Range (VRR = 1) High Range (VRR = 0)
CV02*	CACC	Absolute Accuracy	—	—	$\pm 1/4$ $\pm 1/2$	Lsb Lsb	Low Range (VRR = 1) High Range (VRR = 0)
CV03*	CR	Unit Resistor Value (R)	—	2k	—	$\Omega$	
CV04*	CST	Settling Time <sup>(1)</sup>	—	10	—	$\mu s$	

\* These parameters are characterized but not tested.

**Note 1:** Settling time measured while  $CVRR = 1$  and  $CVR3:CVR0$  transitions from '0000' to '1111'.

**TABLE 27-12: FIXED VOLTAGE REFERENCE (FVR) SPECIFICATIONS**

**Operating Conditions:**  $1.8V < VDD < 3.6V$ ,  $-40^\circ C < TA < +125^\circ C$  (unless otherwise stated).

VR Voltage Reference Specifications		Standard Operating Conditions (unless otherwise stated)					
Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
VR01	VROUT	VR voltage output	TBD	1.2	TBD	V	
VR02	TCVOUT	Voltage drift temperature coefficient	—	TBD	TBD	ppm/ $^\circ C$	
VR03	$\Delta VROUT/\Delta VDD$	Voltage drift with respect to VDD regulation	—	TBD	—	$\mu V/V$	
VR04	TSTABLE	Settling Time	—	TBD	TBD	$\mu s$	

**Legend:** TBD = To Be Determined

\* These parameters are characterized but not tested.

# PIC18F1XK50/PIC18LF1XK50

FIGURE 27-13: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

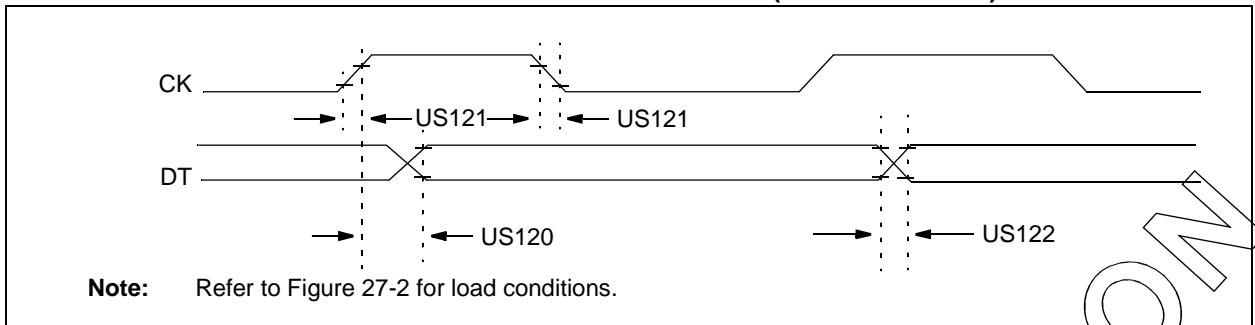


TABLE 27-13: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Standard Operating Conditions (unless otherwise stated) Operating Temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$						
Param. No.	Symbol	Characteristic	Min.	Max.	Units	Conditions
US120	TckH2DTV	SYNC XMIT (Master and Slave)	3.0-5.5V	—	80	ns
		Clock high to data-out valid	1.8-5.5V	—	100	ns
US121	TCKRF	Clock out rise time and fall time (Master mode)	3.0-5.5V	—	45	ns
			1.8-5.5V	—	50	ns
US122	TDTRF	Data-out rise time and fall time	3.0-5.5V	—	45	ns
			1.8-5.5V	—	50	ns

FIGURE 27-14: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

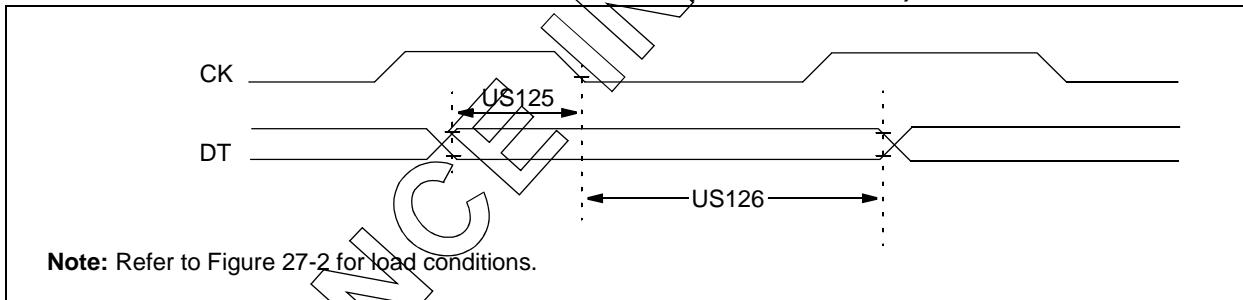


TABLE 27-14: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Standard Operating Conditions (unless otherwise stated) Operating Temperature $-40^{\circ}\text{C} \leq \text{TA} \leq +125^{\circ}\text{C}$						
Param. No.	Symbol	Characteristic	Min.	Max.	Units	Conditions
US125	TDTV2CKL	SYNC RCV (Master and Slave) Data-hold before CK $\downarrow$ (DT hold time)	10	—	ns	
US126	TCKL2DTL	Data-hold after CK $\downarrow$ (DT hold time)	15	—	ns	

# PIC18F1XK50/PIC18LF1XK50

FIGURE 27-15: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)

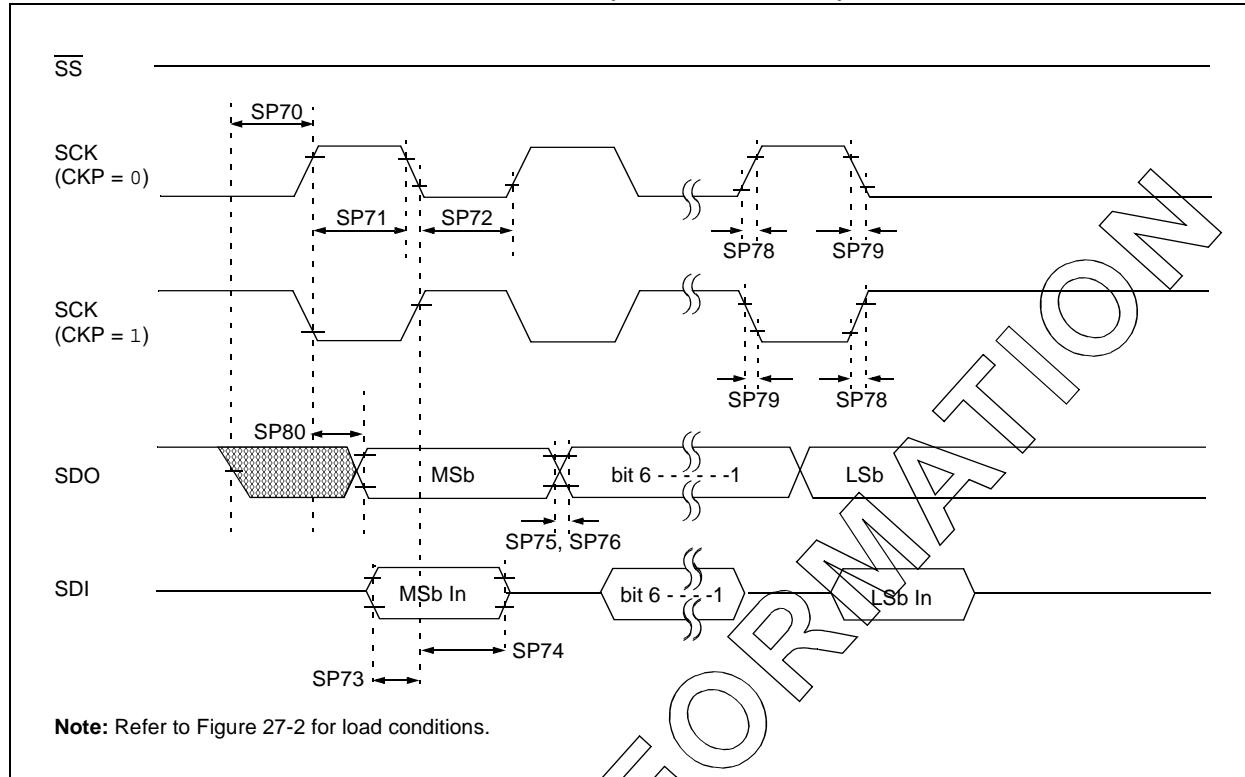
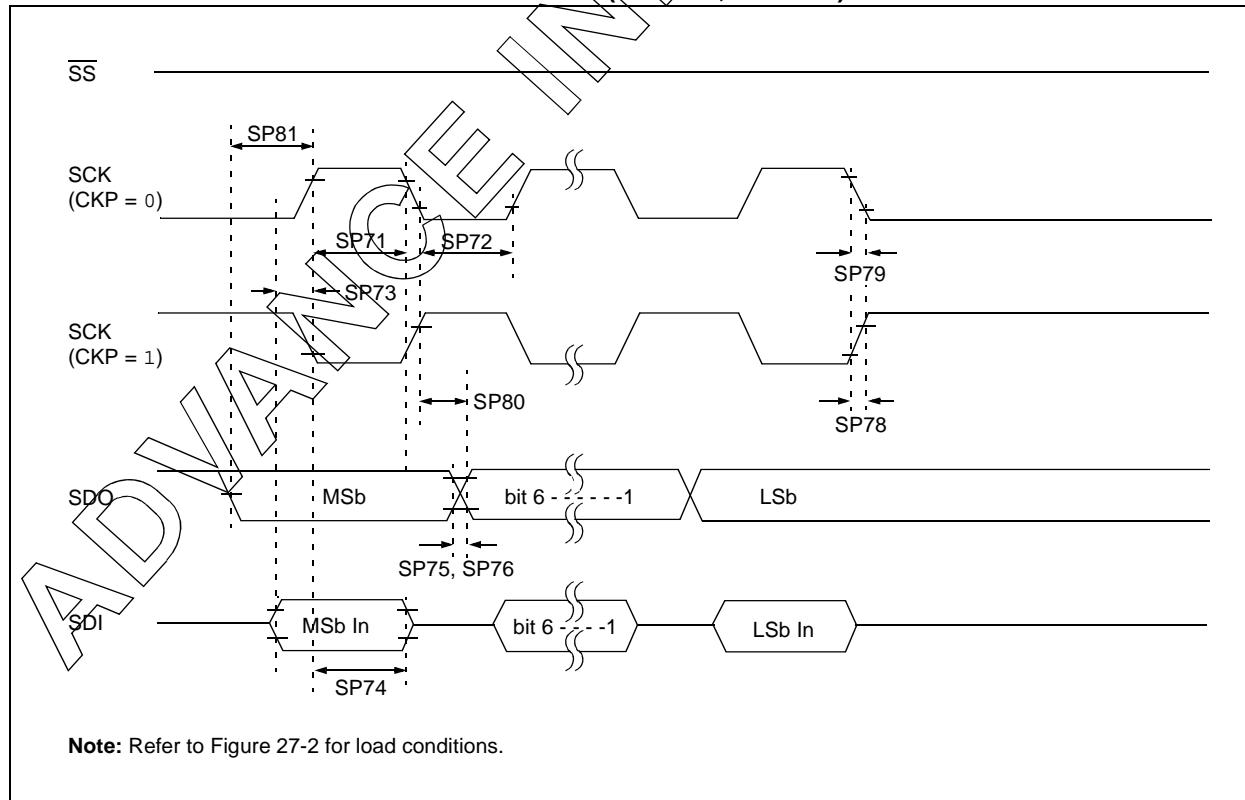


FIGURE 27-16: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)



# PIC18F1XK50/PIC18LF1XK50

FIGURE 27-17: SPI SLAVE MODE TIMING (CKE = 0)

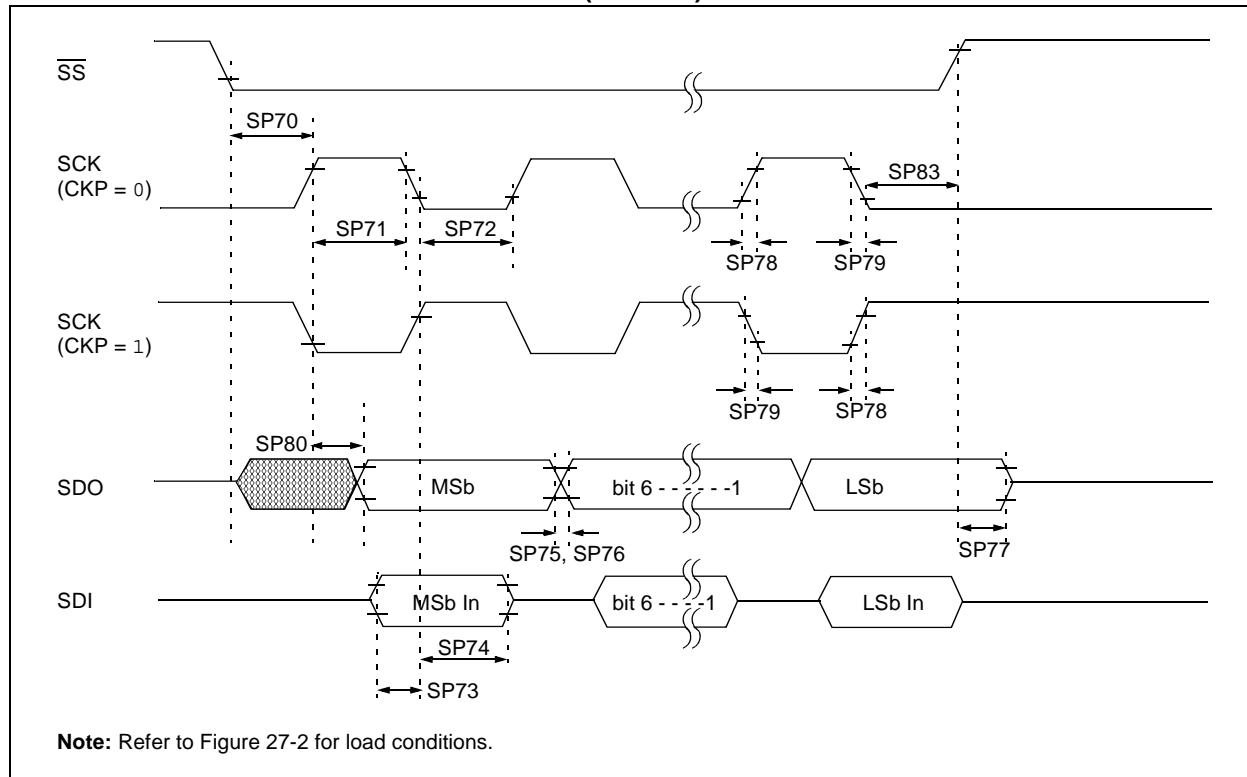


FIGURE 27-18: SPI SLAVE MODE TIMING (CKE = 1)

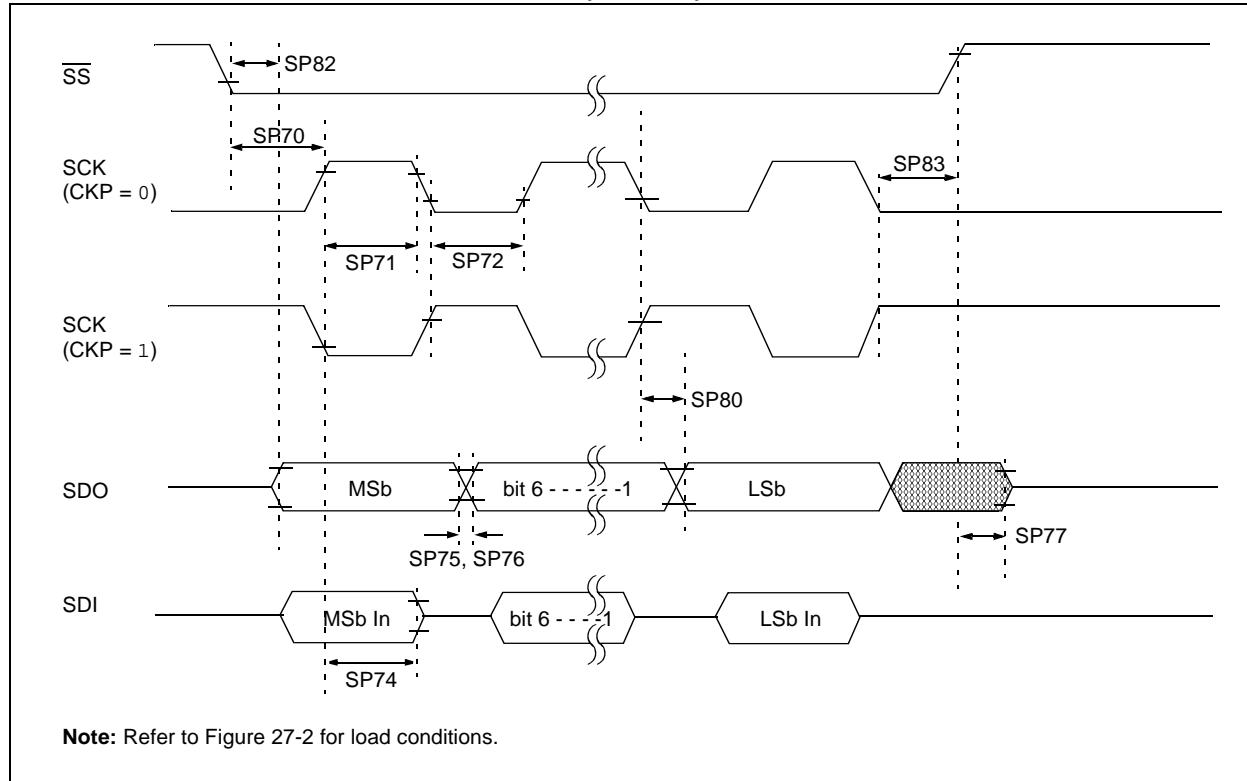


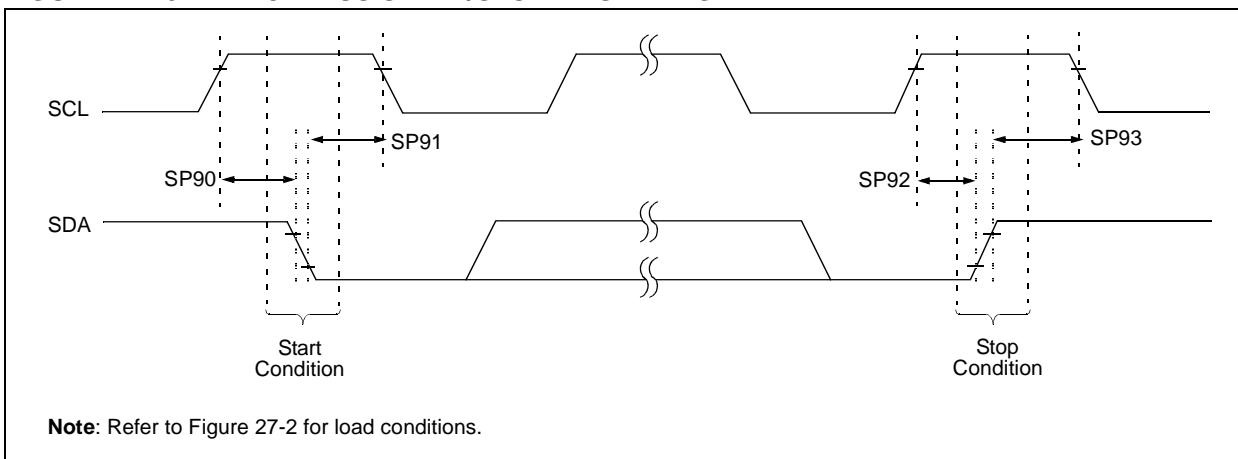
TABLE 27-15: SPI MODE REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ†	Max	Units	Conditions
SP70*	TssL2scH, TssL2scL	$\overline{SS}$ ↓ to SCK↓ or SCK↑ input		TCY	—	—	ns	
SP71*	TscH	SCK input high time (Slave mode)		TCY + 20	—	—	ns	
SP72*	TscL	SCK input low time (Slave mode)		TCY + 20	—	—	ns	
SP73*	TdIV2scH, TdIV2scL	Setup time of SDI data input to SCK edge		100	—	—	ns	
SP74*	TsCH2dIL, TsCL2dIL	Hold time of SDI data input to SCK edge		100	—	—	ns	
SP75*	TdoR	SDO data output rise time	3.0-5.5V	—	10	25	ns	
			1.8-5.5V	—	25	50	ns	
SP76*	TdoF	SDO data output fall time		—	10	25	ns	
SP77*	TssH2doZ	$\overline{SS}$ ↑ to SDO output high-impedance		10	—	50	ns	
SP78*	TscR	SCK output rise time (Master mode)	3.0-5.5V	—	10	25	ns	
			1.8-5.5V	—	25	50	ns	
SP79*	TscF	SCK output fall time (Master mode)		—	10	25	ns	
SP80*	TsCH2dOV, TsCL2dOV	SDO data output valid after SCK edge	3.0-5.5V	—	—	50	ns	
			1.8-5.5V	—	—	145	ns	
SP81*	TdoV2scH, TdoV2scL	SDO data output setup to SCK edge		Tcy	—	—	ns	
SP82*	TssL2dOV	SDO data output valid after $\overline{SS}$ ↓ edge		—	—	50	ns	
SP83*	TscH2ssH, TscL2ssH	$\overline{SS}$ ↑ after SCK edge		1.5TCY + 40	—	—	ns	

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 27-19: I<sup>2</sup>C<sup>TM</sup> BUS START/STOP BITS TIMING



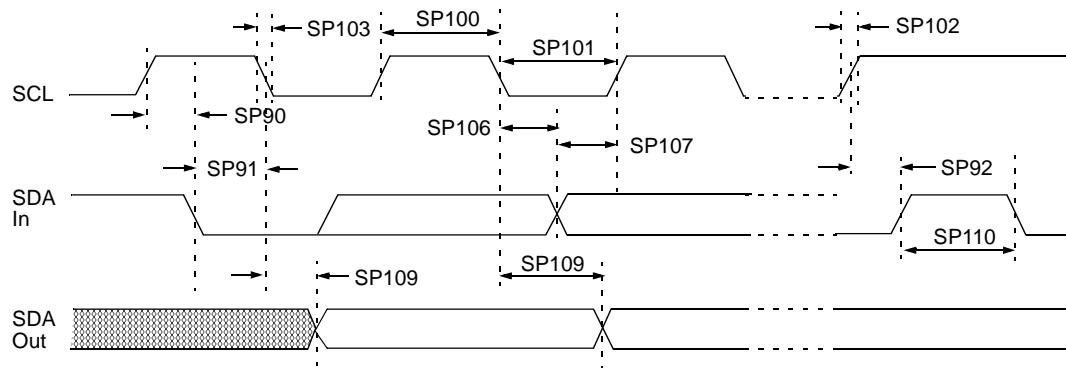
# PIC18F1XK50/PIC18LF1XK50

TABLE 27-16: I<sup>2</sup>C™ BUS START/STOP BITS REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
SP90*	Ts <sub>U</sub> :STA	Start condition	100 kHz mode	4700	—	—	ns	Only relevant for Repeated Start condition
		Setup time	400 kHz mode	600	—	—		
SP91*	THD:STA	Start condition	100 kHz mode	4000	—	—	ns	After this period, the first clock pulse is generated
		Hold time	400 kHz mode	600	—	—		
SP92*	Ts <sub>U</sub> :STO	Stop condition	100 kHz mode	4700	—	—	ns	
		Setup time	400 kHz mode	600	—	—		
SP93	THD:STO	Stop condition	100 kHz mode	4000	—	—	ns	
		Hold time	400 kHz mode	600	—	—		

\* These parameters are characterized but not tested.

FIGURE 27-20: I<sup>2</sup>C™ BUS DATA TIMING



Note: Refer to Figure 27-2 for load conditions.

# PIC18F1XK50/PIC18LF1XK50

TABLE 27-17: I<sup>2</sup>C™ BUS DATA REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
SP100*	THIGH	Clock high time	100 kHz mode	4.0	—	μs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	μs	Device must operate at a minimum of 10 MHz
			SSP Module	1.5TCY	—		
SP101*	TLOW	Clock low time	100 kHz mode	4.7	—	μs	Device must operate at a minimum of 1.5 MHz
			400 kHz mode	1.3	—	μs	Device must operate at a minimum of 10 MHz
			SSP Module	1.5TCY	—		
SP102*	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1CB	300	ns	CB is specified to be from 10-400 pF
SP103*	TF	SDA and SCL fall time	100 kHz mode	—	250	ns	
			400 kHz mode	20 + 0.1CB	250	ns	CB is specified to be from 10-400 pF
SP90*	TSU:STA	Start condition setup time	100 kHz mode	4.7	—	μs	Only relevant for Repeated Start condition
			400 kHz mode	0.6	—	μs	
SP91*	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	
SP106*	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	μs	
SP107*	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	(Note 2)
			400 kHz mode	100	—	ns	
SP92*	TSU:STO	Stop condition setup time	100 kHz mode	4.7	—	μs	
			400 kHz mode	0.6	—	μs	
SP109*	TAA	Output valid from clock	100 kHz mode	—	3500	ns	(Note 1)
			400 kHz mode	—	—	ns	
SP110*	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	
SP	CB	Bus capacitive loading		—	400	pF	

\* These parameters are characterized but not tested.

**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 (400 kHz) I<sup>2</sup>C bus device can be used in a Standard mode (100 kHz) I<sup>2</sup>C bus system, but the requirement  $TSU:DAT \geq 250$  ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line TR max. +  $TSU:DAT = 1000 + 250 = 1250$  ns (according to the Standard mode I<sup>2</sup>C bus specification), before the SCL line is released.

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## **28.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES**

Graphs and tables are not available at this time.

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## 29.0 PACKAGING INFORMATION

### 29.1 Package Marking Information

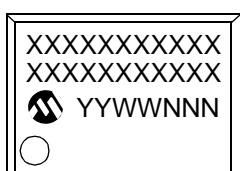
20-Lead PDIP



Example



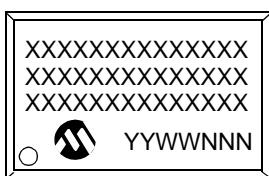
20-Lead SSOP



Example



20-Lead SOIC (.300")



Example



<b>Legend:</b>	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code
	(e3)	Pb-free JEDEC designator for Matte Tin (Sn)
*		This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

**Note:** In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

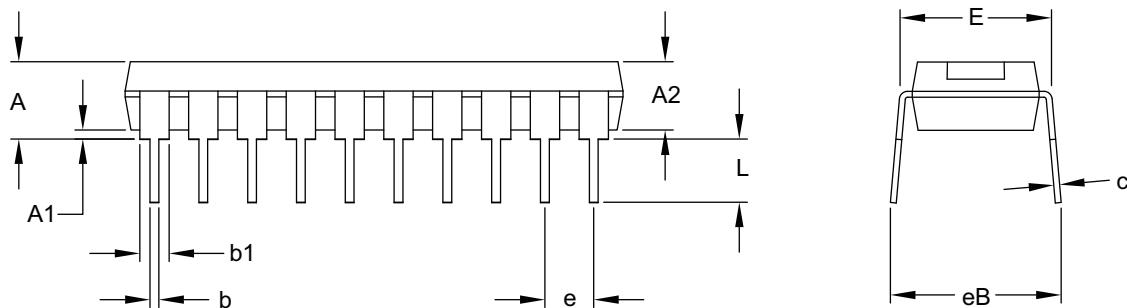
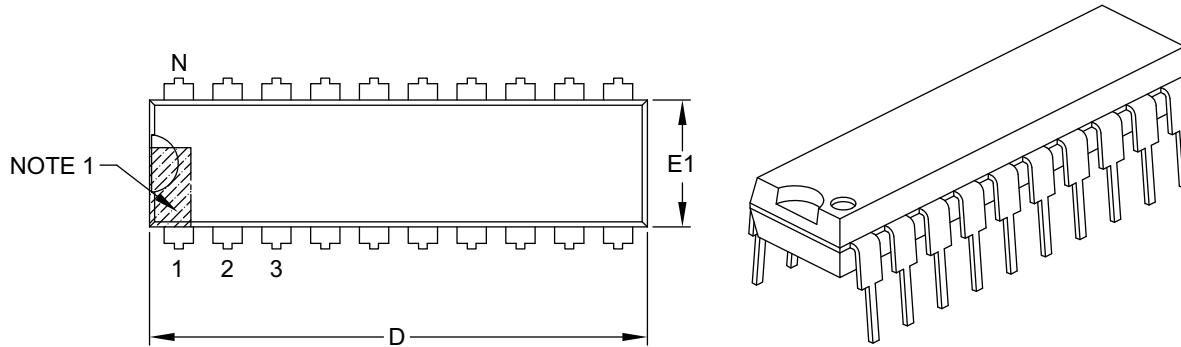
# PIC18F1XK50/PIC18LF1XK50

## 29.2 Package Details

The following sections give the technical details of the packages.

### 20-Lead Plastic Dual In-Line (P) – 300 mil Body [PDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Dimension Limits	Units	INCHES		
		MIN	NOM	MAX
Number of Pins	N		20	
Pitch	e		.100 BSC	
Top to Seating Plane	A	—	—	.210
Molded Package Thickness	A2	.115	.130	.195
Base to Seating Plane	A1	.015	—	—
Shoulder to Shoulder Width	E	.300	.310	.325
Molded Package Width	E1	.240	.250	.280
Overall Length	D	.980	1.030	1.060
Tip to Seating Plane	L	.115	.130	.150
Lead Thickness	c	.008	.010	.015
Upper Lead Width	b1	.045	.060	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	—	—	.430

#### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

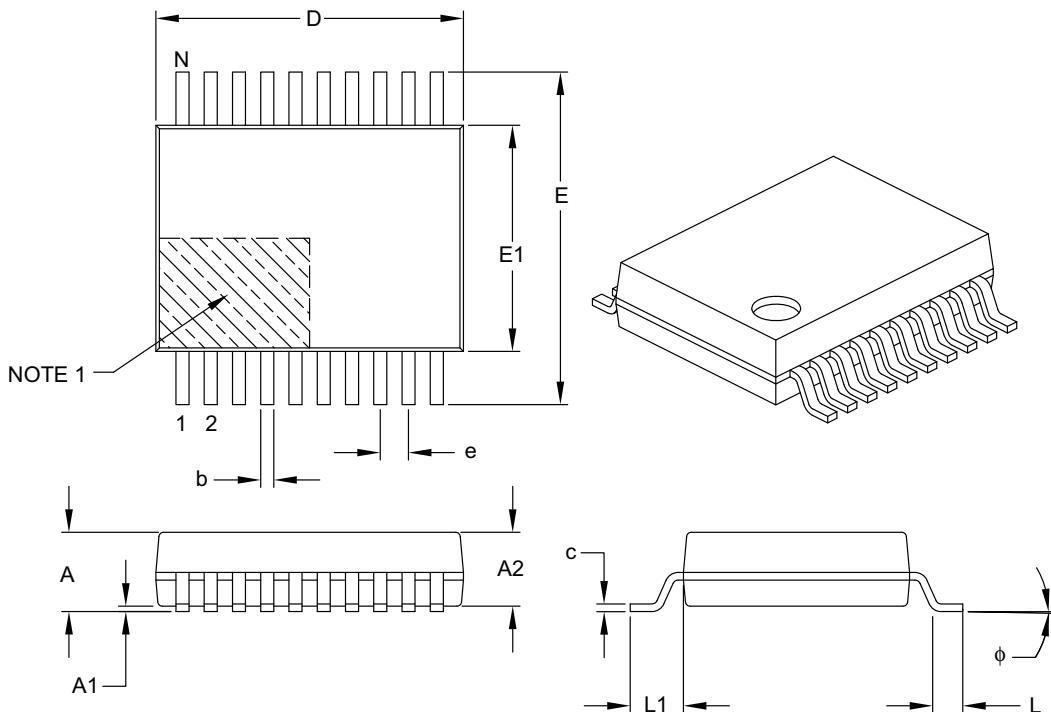
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-019B

# PIC18F1XK50/PIC18LF1XK50

## 20-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins		N		
Pitch		e		
Overall Height		A		
Molded Package Thickness		A2		
Standoff		A1		
Overall Width		E		
Molded Package Width		E1		
Overall Length		D		
Foot Length		L		
Footprint		L1		
Lead Thickness		c		
Foot Angle		phi		
Lead Width		b		
		1.25 REF		

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

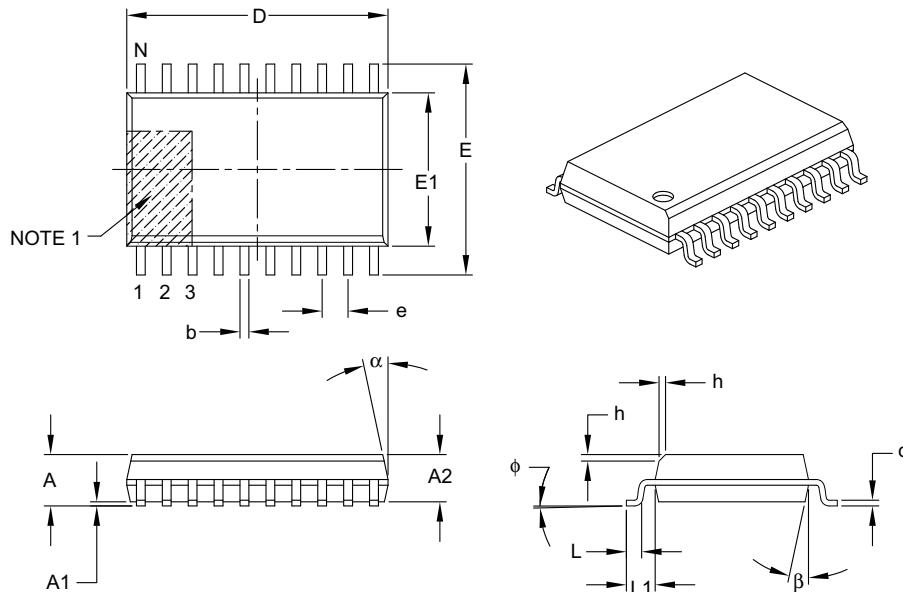
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-072B

# PIC18F1XK50/PIC18LF1XK50

## 20-Lead Plastic Small Outline (SO) – Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	20		
Pitch	e	1.27 BSC		
Overall Height	A	–	–	2.65
Molded Package Thickness	A2	2.05	–	–
Standoff §	A1	0.10	–	0.30
Overall Width	E	10.30 BSC		
Molded Package Width	E1	7.50 BSC		
Overall Length	D	12.80 BSC		
Chamfer (optional)	h	0.25	–	0.75
Foot Length	L	0.40	–	1.27
Footprint	L1	1.40 REF		
Foot Angle	phi	0°	–	8°
Lead Thickness	c	0.20	–	0.33
Lead Width	b	0.31	–	0.51
Mold Draft Angle Top	alpha	5°	–	15°
Mold Draft Angle Bottom	beta	5°	–	15°

### Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-094B

## APPENDIX A: REVISION HISTORY

### Revision A (May 2008)

Original data sheet for PIC18F1XK50/PIC18LF1XK50 devices.

### Revision B (June 2008)

Revised 27.4 DC Characteristics table.

### Revision C (04/2009)

Revised data sheet title; Revised Features section; Revised Table 1-2; Revised Table 3-1, Table 3-2; Added Note 3 in Section 9.1; Revised Register 14-1; Revised Example 16-1; Revised Section 18.8.4; Revised Register 18-3; Revised Table 20-2; Revised Sections 22.2.1, 22.2.2, 22.5.1.1, 22.7; Revised Tables 23-4, 27-1, 27-2, 27-3 27-4, 27-8.

## APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table B-1.

**TABLE B-1: DEVICE DIFFERENCES**

Features	PIC18F13K50	PIC18F14K50	PIC18LF13K50	PIC18F26K20	PIC18LF14K50	PIC18F44K20	PIC18F45K20	PIC18F46K20
Program Memory (Bytes)	8192	16384	32768	65536	8192	16384	32768	65536
Program Memory (Instructions)	4096	8192	16384	32768	4096	8192	16384	32768
Interrupt Sources	19	19	19	19	20	20	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E	Ports A, B, C, D, E	Ports A, B, C, D, E			
Capture/Compare/ PWM Modules	1	1	1	1	1	1	1	1
Enhanced Capture/Compare/ PWM Modules	1	1	1	1	1	1	1	1
Parallel Communications (PSP)	No	No	No	No	Yes	Yes	Yes	Yes
10-bit Analog-to-Digital Module	11 input channels	11 input channels	11 input channels	11 input channels	14 input channels	14 input channels	14 input channels	14 input channels
Packages	20-pin PDIP 20-pin SOIC 20-pin SSOP	20-pin PDIP 20-pin SOIC 20-pin SSOP	20-pin PDIP 20-pin SOIC 20-pin SSOP	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	20-pin PDIP 20-pin SOIC 20-pin SSOP	40-pin PDIP 44-pin TQFP 44-pin QFN	40-pin PDIP 44-pin TQFP 44-pin QFN	40-pin PDIP 44-pin TQFP 44-pin QFN

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## INDEX

### A

#### A/D

Analog Port Pins, Configuring .....	217
Associated Registers .....	217
Conversions .....	208
Discharge .....	209
Selecting and Configuring Acquisition Time .....	206
Specifications .....	381
Absolute Maximum Ratings .....	357
AC Characteristics	
Industrial and Extended .....	373
Load Conditions .....	372
Access Bank	
Mapping with Indexed Literal Offset Mode .....	46
ACKSTAT .....	167
ACKSTAT Status Flag .....	167
ADC .....	205
Acquisition Requirements .....	215
Block Diagram .....	205
Calculating Acquisition Time .....	215
Channel Selection .....	206
Configuration .....	206
Conversion Clock .....	206
Conversion Procedure .....	210
Internal Sampling Switch (Rss) IMPEDANCE .....	215
Interrupts .....	207
Operation .....	208
Operation During Sleep .....	209
Port Configuration .....	206
Power Management .....	209
Reference Voltage (VREF) .....	206
Result Formatting .....	207
Source Impedance .....	215
Special Event Trigger .....	209
Starting an A/D Conversion .....	207
ADCON0 Register .....	211
ADCON1 Register .....	212, 213
ADDFSR .....	346
ADDLW .....	309
ADDULNK .....	346
ADDWF .....	309
ADDWFC .....	310
ADRESH Register (ADFM = 0) .....	214
ADRESH Register (ADFM = 1) .....	214
ADRESL Register (ADFM = 0) .....	214
ADRESL Register (ADFM = 1) .....	214
Analog Input Connection Considerations .....	227
Analog-to-Digital Converter. See ADC	
ANDLW .....	310
ANDWF .....	311
ANSEL (PORT Analog Control) .....	94
ANSEL Register .....	94
ANSELH Register .....	95
Assembler	
MPASM Assembler .....	354

### B

Bank Select Register (BSR) .....	31
Baud Rate Generator .....	163
BAUDCON Register .....	188
BC .....	311
BCF .....	312
BF .....	167

BF Status Flag .....	167
Block Diagrams	
ADC .....	205
ADC Transfer Function .....	216
Analog Input Model .....	216, 227
Baud Rate Generator .....	163
Capture Mode Operation .....	115
Clock Source .....	12
Comparator 1 .....	220
Comparator 2 .....	221
Crystal Operation .....	13
EUSART Receive .....	178
EUSART Transmit .....	177
External POR Circuit (Slow VDD Power-up) .....	273
External RC Mode .....	14
Fail-Safe Clock Monitor (FSCM) .....	22
Generic I/O Port .....	79
Interrupt Logic .....	64
MSSP (I <sup>2</sup> C Master Mode) .....	161
MSSP (I <sup>2</sup> C Mode) .....	144
MSSP (SPI Mode) .....	135
On-Chip Reset Circuit .....	271
PIC18F1XK50/PIC18LF1XK50 .....	8
PWM (Enhanced) .....	117
Reads from Flash Program Memory .....	51
Resonator Operation .....	14
Table Read Operation .....	47
Table Write Operation .....	48
Table Writes to Flash Program Memory .....	53
Timer0 in 16-Bit Mode .....	99
Timer0 in 8-Bit Mode .....	98
Timer1 .....	102
Timer1 (16-Bit Read/Write Mode) .....	102
Timer2 .....	108
Timer3 .....	110
Timer3 (16-Bit Read/Write Mode) .....	111
USB Interrupt Logic .....	259
USB Peripheral and Options .....	245
Voltage Reference .....	242
Voltage Reference Output Buffer Example .....	242
Watchdog Timer .....	296
BN .....	312
BNC .....	313
BNN .....	313
BNOV .....	314
BNZ .....	314
BOR. See Brown-out Reset.	
BOV .....	317
BRA .....	315
Break Character (12-bit) Transmit and Receive .....	196
BRG. See Baud Rate Generator.	
Brown-out Reset (BOR) .....	274
Detecting .....	274
Disabling in Sleep Mode .....	274
Software Enabled .....	274
Specifications .....	379
Timing and Characteristics .....	378
BSF .....	315
BTFSC .....	316
BTFSS .....	316
BTG .....	317
BZ .....	318

# PIC18F1XK50/PIC18LF1XK50

---

## C

C Compilers	
MPLAB C18	354
MPLAB C30	354
CALL	318
CALLW	347
Capture (CCP Module)	115
CCP Pin Configuration	115
CCPRxH:CCPRxL Registers	115
Prescaler	115
Software Interrupt	115
Timer1/Timer3 Mode Selection	115
Capture/Compare/PWM (CCP)	
Capture Mode. See Capture.	
CCP Mode and Timer Resources	114
Compare Mode. See Compare.	
CCP1CON Register	113
Clock Accuracy with Asynchronous Operation	186
Clock Sources	
Associated registers	23
External Modes	
HS	13
LP	13
XT	13
CLRF	319
CLRWDT	319
CM1CON0 Register	225
CM2CON0 Register	226
CM2CON1 Register	229
Code Examples	
16 x 16 Signed Multiply Routine	62
16 x 16 Unsigned Multiply Routine	62
8 x 8 Signed Multiply Routine	61
8 x 8 Unsigned Multiply Routine	61
A/D Conversion	210
Changing Between Capture Prescalers	115
Clearing RAM Using Indirect Addressing	42
Computed GOTO Using an Offset Value	28
Data EEPROM Read	59
Data EEPROM Refresh Routine	60
Data EEPROM Write	59
Erasing a Flash Program Memory Row	52
Fast Register Stack	28
Implementing a Timer1 Real-Time Clock	105
Initializing PORTA	80
Initializing PORTB	85
Initializing PORTC	90
Loading the SSPBUF (SSPSR) Register	138
Reading a Flash Program Memory Word	51
Saving Status, WREG and BSR Registers in RAM	75
Writing to Flash Program Memory	54-55
Code Protection	285
COMF	320
Comparator	
Associated Registers	230
Operation	219
Operation During Sleep	224
Response Time	222
Comparator Module	219
C1 Output State Versus Input Conditions	222
Comparator Specifications	383
Comparator Voltage Reference (CVREF)	
Response Time	222
Comparator Voltage Reference (CVREF)	
Associated Registers	244

Effects of a Reset	224, 241
Operation During Sleep	241
Overview	241
Comparators	
Effects of a Reset	224
Compare (CCP Module)	116
CCPRx Register	116
Pin Configuration	116
Software Interrupt	116
Special Event Trigger	112, 116
Timer1/Timer3 Mode Selection	116
Computed GOTO	28
CONFIG1H Register	287, 288
CONFIG1L Register	287
CONFIG2H Register	290
CONFIG2L Register	289
CONFIG3H Register	291
CONFIG4L Register	291
CONFIG5H Register	292
CONFIG5L Register	292
CONFIG6H Register	293
CONFIG6L Register	293
CONFIG7H Register	294
CONFIG7L Register	294
Configuration Bits	286
Configuration Register Protection	301
Context Saving During Interrupts	75
CPFSEQ	320
CPFSGT	321
CPFSLT	321
Customer Change Notification Service	409
Customer Notification Service	409
Customer Support	409
CVREF Voltage Reference Specifications	383
<b>D</b>	
Data Addressing Modes	42
Comparing Addressing Modes with the Extended Instruction Set Enabled	45
Direct	42
Indexed Literal Offset	44
Instructions Affected	44
Indirect	42
Inherent and Literal	42
Data EEPROM	
Code Protection	301
Data EEPROM Memory	57
Associated Registers	60
EEADR and EEADRH Registers	57
EECON1 and EECON2 Registers	57
Operation During Code-Protect	60
Protection Against Spurious Write	60
Reading	59
Using	60
Write Verify	59
Writing	59
Data Memory	31
Access Bank	35
and the Extended Instruction Set	44
Bank Select Register (BSR)	31
General Purpose Registers	35
Map for PIC18F13K50/PIC18LF13K50	32
Map for PIC18F14K50/PIC18LF14K50	33
Special Function Registers	35
USB RAM	31
DAW	322

DC and AC Characteristics	
Graphs and Tables	391
DC Characteristics	
Extended and Industrial	368
Industrial and Extended	358
DCFSNZ	323
DECFSZ	322
DECF	323
Development Support	353
Device Differences	397
Device Overview	5
Details on Individual Family Members	6
Features (28-Pin Devices)	7
New Core Features	5
Other Special Features	6
Device Reset Timers	275
Oscillator Start-up Timer (OST)	275
PLL Lock Time-out	275
Power-up Timer (PWRT)	275
Time-out Sequence	275
DEVID1 Register	295
DEVID2 Register	295
Direct Addressing	43
<b>E</b>	
ECCPAS Register	125
EECON1 Register	49, 58
Effect on Standard PIC Instructions	350
Electrical Specifications	357
Enhanced Capture/Compare/PWM (ECCP)	113
Associated Registers	133
Enhanced PWM Mode	117
Auto-Restart	126
Auto-shutdown	125
Direction Change in Full-Bridge Output Mode	123
Full-Bridge Application	121
Full-Bridge Mode	121
Half-Bridge Application	120
Half-Bridge Application Examples	127
Half-Bridge Mode	120
Output Relationships (Active-High and Active-Low)	118
Output Relationships Diagram	119
Programmable Dead Band Delay	127
Shoot-through Current	127
Start-up Considerations	124
Outputs and Configuration	114
Specifications	381
Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART)	177
Equations	
Estimating USB Transceiver Current Consumption	267
Errata	4
EUSART	177
Asynchronous Mode	179
12-bit Break Transmit and Receive	196
Associated Registers, Receive	185
Associated Registers, Transmit	181
Auto-Wake-up on Break	194
Baud Rate Generator (BRG)	189
Clock Accuracy	186
Receiver	182
Setting up 9-bit Mode with Address Detect	184
Transmitter	179
Baud Rate Generator (BRG)	
Associated Registers	189
Clock polarity	
Synchronous Mode	197
Data polarity	
Asynchronous Receive	182
Asynchronous Transmit	179
Synchronous Mode	197
Interrupts	
Asynchronous Receive	183
Asynchronous Transmit	179
Synchronous Master Mode	197, 202
Associated Registers, Receive	201
Associated Registers, Transmit	199, 202
Reception	200
Transmission	197
Synchronous Slave Mode	
Associated Registers, Receive	203
Reception	203
Transmission	202
Extended Instruction Set	
ADDFSR	346
ADDULNK	346
and Using MPLAB Tools	352
CALLW	347
Considerations for Use	350
MOVSF	347
MOVSS	348
PUSHL	348
SUBFSR	349
SUBULNK	349
Syntax	345
<b>F</b>	
Fail-Safe Clock Monitor	22, 285
Fail-Safe Condition Clearing	23
Fail-Safe Detection	22
Fail-Safe Operation	22
Reset or Wake-up from Sleep	23
Fast Register Stack	28
Firmware Instructions	303
Flash Program Memory	47
Associated Registers	55
Control Registers	48
EECON1 and EECON2	48
TABLAT (Table Latch) Register	50
TBLPTR (Table Pointer) Register	50
Erase Sequence	52
Erasing	52
Operation During Code-Protect	55
Reading	51
Table Pointer	
Boundaries Based on Operation	50
Table Pointer Boundaries	50
Table Reads and Table Writes	47
Write Sequence	53
Writing To	53
Protection Against Spurious Writes	55
Unexpected Termination	55
Write Verify	55

# PIC18F1XK50/PIC18LF1XK50

---

## G

General Call Address Support .....	160
GOTO .....	324

## H

Hardware Multiplier .....	61
Introduction .....	61
Operation .....	61
Performance Comparison .....	61

## I

I/O Ports .....	79
I <sup>2</sup> C .....	

Associated Registers .....	176
I <sup>2</sup> C Mode (MSSP) .....	
Acknowledge Sequence Timing .....	170
Baud Rate Generator .....	163
Bus Collision .....	
During a Repeated Start Condition .....	174
During a Stop Condition .....	175
Clock Arbitration .....	164
Clock Stretching .....	156
10-Bit Slave Receive Mode (SEN = 1) .....	156
10-Bit Slave Transmit Mode .....	156
7-Bit Slave Receive Mode (SEN = 1) .....	156
7-Bit Slave Transmit Mode .....	156
Clock Synchronization and the CKP bit (SEN = 1) .....	157
Effects of a Reset .....	171
General Call Address Support .....	160
I <sup>2</sup> C Clock Rate w/BRG .....	163
Master Mode .....	161
Operation .....	162
Reception .....	167
Repeated Start Condition Timing .....	166
Start Condition Timing .....	165
Transmission .....	167
Multi-Master Communication, Bus Collision and Arbitration .....	171
Multi-Master Mode .....	171
Operation .....	148
Read/Write Bit Information (R/W Bit) .....	148, 149
Registers .....	144
Serial Clock (RC3/SCK/SCL) .....	149
Slave Mode .....	148

Addressing .....	148
Reception .....	149
Transmission .....	149
Sleep Operation .....	171
Stop Condition Timing .....	170
ID Locations .....	285, 301
INCFSZ .....	324
INCFSZ .....	325

In-Circuit Debugger .....	301
In-Circuit Serial Programming (ICSP) .....	285, 301
Indexed Literal Offset Addressing .....	

and Standard PIC18 Instructions .....	350
Indexed Literal Offset Mode .....	350
Indirect Addressing .....	43
INFSNZ .....	325
Initialization Conditions for all Registers .....	279–283
Instruction Cycle .....	29
Clocking Scheme .....	29
Instruction Flow/Pipelining .....	29
Instruction Set .....	303
ADDLW .....	309

ADDWF .....	309
ADDWF (Indexed Literal Offset Mode) .....	351
ADDWFC .....	310
ANDLW .....	310
ANDWF .....	311
BC .....	311
BCF .....	312
BN .....	312
BNC .....	313
BNN .....	313
BNOV .....	314
BNZ .....	314
BOV .....	317
BRA .....	315
BSF .....	315
BSF (Indexed Literal Offset Mode) .....	351
BTFSZ .....	316
BTFSZ .....	316
BTG .....	317
BZ .....	318
CALL .....	318
CLRF .....	319
CLRWDT .....	319
COMF .....	320
CPFSEQ .....	320
CPFSGT .....	321
CPFSLT .....	321
DAW .....	322
DCFSNZ .....	323
DEC F .....	322
DECFSZ .....	323
Extended Instruction Set .....	345
General Format .....	305
GOTO .....	324
INC F .....	324
INCFSZ .....	325
INFSNZ .....	325
IORLW .....	326
IORWF .....	326
LFSR .....	327
MOV F .....	327
MOVFF .....	328
MOVLB .....	328
MOV LW .....	329
MOVWF .....	329
MULLW .....	330
MULWF .....	330
NEGF .....	331
NOP .....	331
Opcode Field Descriptions .....	304
POP .....	332
PUSH .....	332
RCALL .....	333
RESET .....	333
RETFIE .....	334
RETLW .....	334
RETURN .....	335
RLCF .....	335
RLNCF .....	336
RRCF .....	336
RRNCF .....	337
SET F .....	337
SETF (Indexed Literal Offset Mode) .....	351
SLEEP .....	338
SUBFWB .....	338

# PIC18F1XK50/PIC18LF1XK50

---

SUBLW .....	339	MPLAB ICD 2 In-Circuit Debugger .....	355
SUBWF .....	339	MPLAB ICE 2000 High-Performance Universal .....	355
SUBWFB .....	340	In-Circuit Emulator .....	355
SWAPF .....	340	MPLAB Integrated Development Environment Software ..	353
TBLRD .....	341	MPLAB PM3 Device Programmer .....	355
TBLWT .....	342	MPLAB REAL ICE In-Circuit Emulator System .....	355
TSTFSZ .....	343	MPLINK Object Linker/MPLIB Object Librarian .....	354
XORLW .....	343	<b>MSSP</b>	
XORWF .....	344	ACK Pulse .....	148, 149
INTCON Register .....	65	I <sup>2</sup> C Mode. See I <sup>2</sup> C Mode.	
INTCON Registers .....	65–67	Module Overview .....	135
INTCON2 Register .....	66	SPI Mode. See SPI Mode.	
INTCON3 Register .....	67	SSPBUF Register .....	140
Inter-Integrated Circuit. See I <sup>2</sup> C.		SSPSR Register .....	140
Internal Oscillator Block		MULLW .....	330
INTOSC		MULWF .....	330
Specifications .....	376, 377	<b>N</b>	
Internal RC Oscillator		NEGF .....	331
Use with WDT .....	296	NOP .....	331
Internal Sampling Switch (RSS) IMPEDANCE .....	215	<b>O</b>	
Internet Address .....	409	OSCCON Register .....	16, 17
Interrupt Sources .....	285	Oscillator Module .....	11
ADC .....	207	Oscillator Parameters .....	376
Capture Complete (CCP) .....	115	Oscillator Selection .....	285
Compare Complete (CCP) .....	116	Oscillator Specifications .....	375
Interrupt-on-Change (RB7:RB4) .....	79, 85	Oscillator Start-up Timer (OST) .....	275
INTn Pin .....	75	Specifications .....	379
PORTB, Interrupt-on-Change .....	75	Oscillator Switching	
TMR0 .....	75	Fail-Safe Clock Monitor .....	22
TMR0 Overflow .....	99	Oscillator, Timer1 .....	101, 111
TMR1 Overflow .....	101	Oscillator, Timer3 .....	109
TMR3 Overflow .....	109, 111	OSCTUNE Register .....	18
Interrupts .....	63	<b>P</b>	
INTOSC Specifications .....	376, 377	P1A/P1B/P1C/P1D. See Enhanced Capture/Compare/PWM (ECCP) .....	117
IOCA Register .....	82	Packaging Information .....	393
IOCB Register .....	87	Marking .....	393
IORLW .....	326	PICSTART Plus Development Programmer .....	356
IORWF .....	326	PIE Registers .....	70
IPR Registers .....	72	PIE1 Register .....	70
IPR1 Register .....	72	PIE2 Register .....	71
IPR2 Register .....	73	Pinout Descriptions	
<b>L</b>		PIC18F1XK50/PIC18LF1XK50 .....	9
LATA Register .....	82	PIR Registers .....	68
LATB Register .....	87	PIR1 Register .....	68
LATC Register .....	91	PIR2 Register .....	69
LFSR .....	327	POP .....	332
Load Conditions .....	372	POR. See Power-on Reset.	
Low-Voltage ICSP Programming. See Single-Supply ICSP Programming		PORTA	
<b>M</b>		Associated Registers .....	84
Master Clear (MCLR) .....	273	LATA Register .....	79
Master Synchronous Serial Port (MSSP). See MSSP.		PORTA Register .....	79
Memory Organization .....	25	Specifications .....	377
Data Memory .....	31	TRISA Register .....	79
Program Memory .....	25	PORTA Register .....	81
Microchip Internet Web Site .....	409	<b>PORTB</b>	
MOVF .....	327	Associated Registers .....	89
MOVFF .....	328	LATB Register .....	85
MOVLB .....	328	PORTB Register .....	85
MOVlw .....	329	TRISB Register .....	85
MOVsf .....	347	PORTB Register .....	86, 90
MOVss .....	348	<b>PORTC</b>	
MOVwf .....	329	Associated Registers .....	93
MPLAB ASM30 Assembler, Linker, Librarian .....	354		

# PIC18F1XK50/PIC18LF1XK50

---

LATC Register .....	90
PORTC Register .....	90
RC3/SCK/SCL Pin .....	149
Specifications .....	377
TRISC Register .....	90
Power Managed Modes .....	231
and A/D Operation .....	209
and PWM Operation .....	132
and SPI Operation .....	143
Entering .....	231
Exiting Idle and Sleep Modes .....	235
by Interrupt .....	235
by Reset .....	236
by WDT Time-out .....	235
Without a Start-up Delay .....	236
Idle Modes .....	233
PRI_IDLE .....	234
RC_IDLE .....	235
SEC_IDLE .....	234
Multiple Sleep Functions .....	232
Run Modes .....	232
PRI_RUN .....	232
RC_RUN .....	232
SEC_RUN .....	232
Selecting .....	231
Sleep Mode .....	233
Summary (table) .....	231
Power-on Reset (POR) .....	273
Power-up Timer (PWRT) .....	275
Time-out Sequence .....	275
Power-up Timer (PWRT)	
Specifications .....	379
Precision Internal Oscillator Parameters .....	377
Prescaler, Timer0 .....	99
PRI_IDLE Mode .....	234
PRI_RUN Mode .....	232
Program Counter .....	26
PCL, PCH and PCU Registers .....	26
PCLATH and PCLATU Registers .....	26
Program Memory	
and Extended Instruction Set .....	46
Code Protection .....	299
Instructions .....	30
Two-Word .....	30
Interrupt Vector .....	25
Look-up Tables .....	28
Map and Stack (diagram) .....	25
Reset Vector .....	25
Program Verification and Code Protection .....	298
Associated Registers .....	298
Programming, Device Instructions .....	303
PSTRCON Register .....	129
Pulse Steering .....	129
PUSH .....	332
PUSH and POP Instructions .....	27
PUSHL .....	348
PWM (ECCP Module)	
Effects of a Reset .....	132
Operation in Power Managed Modes .....	132
Operation with Fail-Safe Clock Monitor .....	132
Pulse Steering .....	129
Steering Synchronization .....	131
PWM Mode. See Enhanced Capture/Compare/PWM .....	117
PWM1CON Register .....	128
<b>R</b>	
RAM. See Data Memory.	
RC_IDLE Mode .....	235
RC_RUN Mode .....	232
RCALL .....	333
RCON Register .....	74, 272
Bit Status During Initialization .....	278
RCREG .....	184
RCSTA Register .....	187
Reader Response .....	410
RECON0 (Reference Control 0) Register .....	243
RECON1 (Reference Control 1) Register .....	243
RECON2 (Reference Control 2) Register .....	244
Registers	
ADCON0 (ADC Control 0) .....	211
ADCON1 (ADC Control 1) .....	212, 213
ADRESH (ADC Result High) with ADFM = 0) .....	214
ADRESH (ADC Result High) with ADFM = 1) .....	214
ADRESL (ADC Result Low) with ADFM = 0) .....	214
ADRESL (ADC Result Low) with ADFM = 1) .....	214
ANSEL (Analog Select 1) .....	94
ANSEL (PORT Analog Control) .....	94
ANSELH (Analog Select 2) .....	95
ANSELH (PORT Analog Control) .....	95
BAUDCON (EUSART Baud Rate Control) .....	188
BDnSTAT (Buffer Descriptor n Status, CPU Mode) .....	255
BDnSTAT (Buffer Descriptor n Status, SIE Mode) .....	256
CCP1CON (Enhanced Capture/Compare/PWM Control) .....	113
CM1CON0 (C1 Control) .....	225
CM2CON0 (C2 Control) .....	226
CM2CON1 (C2 Control) .....	229
CONFIG1H (Configuration 1 High) .....	287, 288
CONFIG1L (Configuration 1 Low) .....	287
CONFIG2H (Configuration 2 High) .....	290
CONFIG2L (Configuration 2 Low) .....	289
CONFIG3H (Configuration 3 High) .....	291
CONFIG4L (Configuration 4 Low) .....	291
CONFIG5H (Configuration 5 High) .....	292
CONFIG5L (Configuration 5 Low) .....	292
CONFIG6H (Configuration 6 High) .....	293
CONFIG6L (Configuration 6 Low) .....	293
CONFIG7H (Configuration 7 High) .....	294
CONFIG7L (Configuration 7 Low) .....	294
DEVID1 (Device ID 1) .....	295
DEVID2 (Device ID 2) .....	295
ECCPAS (Enhanced CCP Auto-shutdown Control) .....	125
EECON1 (Data EEPROM Control 1) .....	49, 58
INTCON (Interrupt Control) .....	65
INTCON2 (Interrupt Control 2) .....	66
INTCON3 (Interrupt Control 3) .....	67
IOCA (Interrupt-on-Change PORTA) .....	82
IOCB (Interrupt-on-Change PORTB) .....	87
IPR1 (Peripheral Interrupt Priority 1) .....	72
IPR2 (Peripheral Interrupt Priority 2) .....	73
LATA (PORTA Data Latch) .....	82
LATB (PORTB Data Latch) .....	87
LATC (PORTC Data Latch) .....	91
OSCCON (Oscillator Control) .....	16, 17
OSCTUNE (Oscillator Tuning) .....	18
PIE1 (Peripheral Interrupt Enable 1) .....	70

PIE2 (Peripheral Interrupt Enable 2) .....	71	Serial Clock, SCK .....	135
PIR1 (Peripheral Interrupt Request 1) .....	68	Serial Data In (SDI) .....	135
PIR2 (Peripheral Interrupt Request 2) .....	69	Serial Data Out (SDO) .....	135
PORTA .....	81	Serial Peripheral Interface. See SPI Mode.	
PORTB .....	86, 90	SETF .....	337
PSTRCON (Pulse Steering Control) .....	129	Shoot-through Current .....	127
PWM1CON (Enhanced PWM Control) .....	128	Single-Supply ICSP Programming.	
RCON (Reset Control) .....	74, 272	Slave Select (SS) .....	135
RCSTA (Receive Status and Control) .....	187	Slave Select Synchronization .....	141
REFCON0 .....	243	SLEEP .....	338
REFCON1 .....	243	Sleep Mode .....	233
REFCON2 .....	244	SLRCON Register .....	96
SLRCON (PORT Slew Rate Control) .....	96	Software Simulator (MPLAB SIM) .....	354
SRCON0 (SR Latch Control 0) .....	238	SPBRG .....	189
SRCON1 (SR Latch Control 1) .....	239	SPBRGH .....	189
SSPADD (MSSP Address and Baud Rate, SPI Mode) .....	155	Special Event Trigger .....	209
SSPCON1 (MSSP Control 1, I <sup>2</sup> C Mode) .....	146	Special Event Trigger. See Compare (ECCP Mode).	
SSPCON1 (MSSP Control 1, SPI Mode) .....	137	Special Features of the CPU .....	285
SSPCON2 (MSSP Control 2, I <sup>2</sup> C Mode) .....	147	Special Function Registers .....	35
SSPMASK (SSP Mask) .....	154	Map .....	36
SSPSTAT (MSSP Status, SPI Mode) .....	136, 145	SPI Mode	
STATUS .....	41	Typical Master/Slave Connection .....	139
STKPTR (Stack Pointer) .....	27	SPI Mode (MSSP)	
T0CON (Timer0 Control) .....	97	Associated Registers .....	143
T1CON (Timer1 Control) .....	101	Bus Mode Compatibility .....	143
T2CON (Timer2 Control) .....	107	Effects of a Reset .....	143
T3CON (Timer3 Control) .....	109	Enabling SPI I/O .....	139
TRISA (Tri-State PORTA) .....	81	Master Mode .....	140
TRISB (Tri-State PORTB) .....	86, 90	Operation .....	138
TXSTA (Transmit Status and Control) .....	186	Operation in Power Managed Modes .....	143
UCFG (USB Configuration) .....	248	Serial Clock .....	135
UCON (USB Control) .....	246	Serial Data In .....	135
UEIE (USB Error Interrupt Enable) .....	264	Serial Data Out .....	135
UEIR (USB Error Interrupt Status) .....	263	Slave Mode .....	141
UEPn (USB Endpoint n Control) .....	251	Slave Select .....	135
UIE (USB Interrupt Enable) .....	262	Slave Select Synchronization .....	141
UIR (USB Interrupt Status) .....	260	SPI Clock .....	140
USTAT (USB Status) .....	250	Typical Connection .....	139
WDTCON (Watchdog Timer Control) .....	297	SR Latch .....	237
WPUA (Weak Pull-up PORTA) .....	82	Associated Registers .....	239
WPUB (Weak Pull-up PORTB) .....	87	SRCON0 Register .....	238
RESET .....	333	SRCON1 Register .....	239
Reset State of Registers .....	278	SS .....	135
Resets .....	271, 285	SSP	
Brown-out Reset (BOR) .....	285	Typical SPI Master/Slave Connection .....	139
Oscillator Start-up Timer (OST) .....	285	SSPADD Register .....	155
Power-on Reset (POR) .....	285	SSPCON1 Register .....	137, 146
Power-up Timer (PWRT) .....	285	SSPCON2 Register .....	147
RETFIE .....	334	SSPMASK Register .....	154
RETLW .....	334	SSPOV .....	167
RETURN .....	335	SSPOV Status Flag .....	167
Return Address Stack .....	26	SSPSTAT Register .....	136, 145
Return Stack Pointer (STKPTR) .....	27	R/W Bit .....	148, 149
Revision History .....	397	Stack Full/Underflow Resets .....	28
RLCF .....	335	Standard Instructions .....	303
RLNCF .....	336	STATUS Register .....	41
RRCF .....	336	STKPTR Register .....	27
RRNCF .....	337	SUBFSR .....	349
<b>S</b>		SUBFWB .....	338
SCK .....	135	SUBLW .....	339
SDI .....	135	SUBULNK .....	349
SDO .....	135	SUBWF .....	339
SEC_IDLE Mode .....	234	SUBWFB .....	340
SEC_RUN Mode .....	232	SWAPF .....	340

# PIC18F1XK50/PIC18LF1XK50

---

## T

T0CON Register .....	97
T1CON Register .....	101
T2CON Register .....	107
T3CON Register .....	109
Table Pointer Operations (table) .....	50
Table Reads/Table Writes .....	28
TBLRD .....	341
TBLWT .....	342
Thermal Considerations .....	371
Time-out in Various Situations (table) .....	275
Timer0 .....	97
Associated Registers .....	99
Operation .....	98
Overflow Interrupt .....	99
Prescaler .....	99
Prescaler Assignment (PSA Bit) .....	99
Prescaler Select (T0PS2:T0PS0 Bits) .....	99
Prescaler. See Prescaler, Timer0.	
Reads and Writes in 16-Bit Mode .....	98
Source Edge Select (T0SE Bit) .....	98
Source Select (T0CS Bit) .....	98
Specifications .....	380
Switching Prescaler Assignment .....	99
Timer1 .....	101
16-Bit Read/Write Mode .....	103
Associated Registers .....	106
Interrupt .....	104
Operation .....	102
Oscillator .....	101, 103
Oscillator Layout Considerations .....	104
Overflow Interrupt .....	101
Resetting, Using the CCP Special Event Trigger .....	104
Specifications .....	380
TMR1H Register .....	101
TMR1L Register .....	101
Use as a Real-Time Clock .....	105
Timer2 .....	107
Associated Registers .....	108
Interrupt .....	108
Operation .....	107
Output .....	108
Timer3 .....	109
16-Bit Read/Write Mode .....	111
Associated Registers .....	112
Operation .....	110
Oscillator .....	109, 111
Overflow Interrupt .....	109, 111
Special Event Trigger (CCP) .....	112
TMR3H Register .....	109
TMR3L Register .....	109
Timing Diagrams .....	
A/D Conversion .....	382
Acknowledge Sequence .....	170
Asynchronous Reception .....	184
Asynchronous Transmission .....	180
Asynchronous Transmission (Back to Back) .....	181
Auto Wake-up Bit (WUE) During Normal Operation .....	195
Auto Wake-up Bit (WUE) During Sleep .....	195
Automatic Baud Rate Calculator .....	193
Baud Rate Generator with Clock Arbitration .....	164
BRG Reset Due to SDA Arbitration During Start Condition .....	173
Brown-out Reset (BOR) .....	378

Bus Collision During a Repeated Start Condition (Case 1) .....	174
Bus Collision During a Repeated Start Condition (Case 2) .....	174
Bus Collision During a Start Condition (SCL = 0) .....	173
Bus Collision During a Stop Condition (Case 1) .....	175
Bus Collision During a Stop Condition (Case 2) .....	175
Bus Collision During Start Condition (SDA only) .....	172
Bus Collision for Transmit and Acknowledge .....	171
CLKOUT and I/O .....	377
Clock Synchronization .....	157
Clock Timing .....	373
Clock/Instruction Cycle .....	29
Comparator Output .....	219
Enhanced Capture/Compare/PWM (ECCP) .....	381
Fail-Safe Clock Monitor (FSCM) .....	23
First Start Bit Timing .....	165
Full-Bridge PWM Output .....	122
Half-Bridge PWM Output .....	120, 127
I <sup>2</sup> C Bus Data .....	388
I <sup>2</sup> C Bus Start/Stop Bits .....	387
I <sup>2</sup> C Master Mode (7 or 10-Bit Transmission) .....	168
I <sup>2</sup> C Master Mode (7-Bit Reception) .....	169
I <sup>2</sup> C Slave Mode (10-Bit Reception, SEN = 0) .....	152
I <sup>2</sup> C Slave Mode (10-Bit Reception, SEN = 1) .....	159
I <sup>2</sup> C Slave Mode (10-Bit Transmission) .....	153
I <sup>2</sup> C Slave Mode (7-bit Reception, SEN = 0) .....	150
I <sup>2</sup> C Slave Mode (7-Bit Reception, SEN = 1) .....	158
I <sup>2</sup> C Slave Mode (7-Bit Transmission) .....	151
I <sup>2</sup> C Slave Mode General Call Address Sequence (7 or 10-Bit Address Mode) .....	160
I <sup>2</sup> C Stop Condition Receive or Transmit Mode .....	170
Internal Oscillator Switch Timing .....	19
PWM Auto-shutdown	
Auto-restart Enabled .....	126
Firmware Restart .....	126
PWM Direction Change .....	123
PWM Direction Change at Near 100% Duty Cycle ..	124
PWM Output (Active-High) .....	118
PWM Output (Active-Low) .....	119
Repeat Start Condition .....	166
Reset, WDT, OST and Power-up Timer .....	378
Send Break Character Sequence .....	196
Slave Synchronization .....	141
Slow Rise Time (MCLR Tied to VDD, VDD Rise > TPWRT) .....	277
SPI Master Mode (CKE = 1, SMP = 1) .....	385
SPI Mode (Master Mode) .....	140
SPI Mode (Slave Mode, CKE = 0) .....	142
SPI Mode (Slave Mode, CKE = 1) .....	142
SPI Slave Mode (CKE = 0) .....	386
SPI Slave Mode (CKE = 1) .....	386
Synchronous Reception (Master Mode, SREN) .....	201
Synchronous Transmission .....	198
Synchronous Transmission (Through TXEN) .....	198
Time-out Sequence on POR w/PLL Enabled	
(MCLR Tied to VDD) .....	277
Time-out Sequence on Power-up (MCLR Not Tied to VDD, Case 1) .....	276
Time-out Sequence on Power-up (MCLR Not Tied to VDD, Case 2) .....	276
Time-out Sequence on Power-up (MCLR Tied to VDD, VDD Rise < TPWRT) .....	276
Timer0 and Timer1 External Clock .....	380
Transition for Entry to Sleep Mode .....	233

Transition for Wake from Sleep (HSPLL) .....	233
Transition Timing for Entry to Idle Mode .....	234
Transition Timing for Wake from Idle to Run Mode .....	234
USART Synchronous Receive (Master/Slave) .....	384
USART Synchronous Transmission (Master/Slave) .....	384
<b>Timing Diagrams and Specifications</b>	
A/D Conversion Requirements .....	382
PLL Clock .....	376
<b>Timing Parameter Symbology</b> .....	372
<b>Timing Requirements</b>	
I <sup>2</sup> C Bus Data .....	389
I <sup>2</sup> C Bus Start/Stop Bits .....	388
SPI Mode .....	387
<b>Top-of-Stack Access</b> .....	26
TRISA Register .....	81
TRISB Register .....	86, 90
TSTFSZ .....	343
Two-Speed Start-up .....	285
Two-Word Instructions	
Example Cases .....	30
TXREG .....	179
TXSTA Register .....	186
BRGH Bit .....	189
<b>U</b>	
<b>Universal Serial Bus</b>	
Address Register (UADDR) .....	252
Associated Registers .....	268
Buffer Descriptor Table .....	253
Buffer Descriptors .....	253
Address Validation .....	256
Assignment in Different Buffering Modes .....	258
BDnSTAT Register (CPU Mode) .....	254
BDnSTAT Register (SIE Mode) .....	256
Byte Count .....	256
Example .....	253
Memory Map .....	257
Ownership .....	253
Ping-Pong Buffering .....	257
Register Summary .....	258
Status and Configuration .....	253
Class Specifications and Drivers .....	270
Descriptors .....	270
Endpoint Control .....	251
Enumeration .....	270
External Pull-up Resistors .....	249
Eye Pattern Test Enable .....	249
Firmware and Drivers .....	268
Frame Number Registers .....	252
Frames .....	269
Internal Pull-up Resistors .....	249
Internal Transceiver .....	247
Interrupts .....	259
and USB Transactions .....	259
Layered Framework .....	269
Oscillator Requirements .....	268
Overview .....	245, 269
Ping-Pong Buffer Configuration .....	249
Power .....	269
Power Modes .....	265
Bus Power Only .....	265
Dual Power with Self-Power Dominance .....	266
Self-Power Only .....	265
RAM .....	252
Memory Map .....	252
Speed .....	270
Status and Control .....	246
Transfer Types .....	269
UFRMH:UFRML Registers .....	252
<b>USART</b>	
Synchronous Master Mode	
Requirements, Synchronous Receive .....	384
Requirements, Synchronous Transmission .....	384
Timing Diagram, Synchronous Receive .....	384
Timing Diagram, Synchronous Transmission .....	384
USB Module Electrical Specifications .....	370
<b>USB RAM</b>	
Serial Interface Engine (SIE) .....	31
<b>USB.</b> See <b>Universal Serial Bus.</b>	
<b>V</b>	
<b>Voltage Reference (VR)</b>	
Specifications .....	383
<b>Voltage Reference.</b> See <b>Comparator Voltage Reference (CVREF)</b>	
<b>Voltage References</b>	
Fixed Voltage Reference (FVR) .....	241
VR Stabilization .....	241
<b>VREF.</b> SEE <b>ADC Reference Voltage</b>	
<b>W</b>	
<b>Wake-up on Break</b> .....	194
<b>Watchdog Timer (WDT)</b> .....	285, 296
Associated Registers .....	297
Control Register .....	297
Programming Considerations .....	296
Specifications .....	379
WCOL .....	165, 166, 167, 170
WCOL Status Flag .....	165, 166, 167, 170
WDTCON Register .....	297
WPUA Register .....	82
WPUB Register .....	87
WWW Address .....	409
WWW, On-Line Support .....	4
<b>X</b>	
<b>XORLW</b> .....	343
<b>XORWF</b> .....	344

# **PIC18F1XK50/PIC18LF1XK50**

---

---

## **NOTES:**

## THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at [www.microchip.com](http://www.microchip.com). This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at [www.microchip.com](http://www.microchip.com), click on Customer Change Notification and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support
- Development Systems Information Line

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the web site at: <http://support.microchip.com>**

# PIC18F1XK50/PIC18LF1XK50

---

---

## 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: PIC18F1XK50/PIC18LF1XK50 Literature Number: DS41350C

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?

---

# PIC18F1XK50/PIC18LF1XK50

## 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	<u>XX</u>	<u>XXX</u>	
Device	Temperature Range	Package	Pattern		
Device:	PIC18F13K50 <sup>(1)</sup> , PIC18F14K50 <sup>(1)</sup> , PIC18LF13K50 <sup>(1)</sup> , PIC18LF14K50				
Temperature Range:	E	= -40°C to +125°C	(Extended)		
Package:	P	= PDIP			
	SO	= SOIC			
	SS	= SSOP			
Pattern:	QTP, SQTP, Code or Special Requirements (blank otherwise)				

**Examples:**

- a) PIC18F14K50-E/P 301 = Extended temp., PDIP package, Extended VDD limits, QTP pattern #301.
- b) PIC18LF14K50-E/SO = Extended temp., SOIC package.
- c) PIC18LF14K50-E/P = Extended temp., PDIP package.

**Note 1:** T = in tape and reel PLCC, and TQFP packages only.



# MICROCHIP

## WORLDWIDE SALES AND SERVICE

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://support.microchip.com>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**

Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Boston**

Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**

Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Cleveland**

Independence, OH  
Tel: 216-447-0464  
Fax: 216-447-0643

**Dallas**

Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**

Farmington Hills, MI  
Tel: 248-538-2250  
Fax: 248-538-2260

**Kokomo**

Kokomo, IN  
Tel: 765-864-8360  
Fax: 765-864-8387

**Los Angeles**

Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**Santa Clara**

Santa Clara, CA  
Tel: 408-961-6444  
Fax: 408-961-6445

**Toronto**

Mississauga, Ontario,  
Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8528-2100  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Hong Kong SAR**  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Nanjing**  
Tel: 86-25-8473-2460  
Fax: 86-25-8473-2470

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8203-2660  
Fax: 86-755-8203-1760

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4080

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**  
Tel: 91-20-2566-1512  
Fax: 91-20-2566-1513

**Japan - Yokohama**  
Tel: 81-45-471-6166  
Fax: 81-45-471-6122

**Korea - Daegu**  
Tel: 82-53-744-4301  
Fax: 82-53-744-4302

**Korea - Seoul**  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Malaysia - Kuala Lumpur**  
Tel: 60-3-6201-9857  
Fax: 60-3-6201-9859

**Malaysia - Penang**  
Tel: 60-4-227-8870  
Fax: 60-4-227-4068

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

**Singapore**  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan - Hsin Chu**  
Tel: 886-3-6578-300  
Fax: 886-3-6578-370

**Taiwan - Kaohsiung**  
Tel: 886-7-536-4818  
Fax: 886-7-536-4803

**Taiwan - Taipei**  
Tel: 886-2-2500-6610  
Fax: 886-2-2508-0102

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820