# PIC18(L)F1XK22

## 20-Pin Flash Microcontrollers with XLP Technology

### 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 512 bytes Linear Data Memory Addressing
- Up to 16 MIPS Operation
- 16-bit Wide Instructions, 8-bit Wide Data Path
- Priority Levels for Interrupts
- 31-Level, Software Accessible Hardware Stack
- 8 x 8 Single-Cycle Hardware Multiplier

### Flexible Oscillator Structure

- Precision 16 MHz Internal Oscillator Block:
  - Factory calibrated to ± 1%
  - Software selectable frequencies range of 31 kHz to 16 MHz
  - 64 MHz performance available using PLL – no external components required
- Four Crystal modes up to 64 MHz
- Two External Clock modes up to 64 MHz
- 4X Phase Lock Loop (PLL)
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if peripheral clock stops
- Two-Speed Oscillator Start-up

### Special Microcontroller Features

- 2.3V - 5.5V Operation – PIC18F1XK22
- 1.8V-3.6V Operation – PIC18LF1XK22
- Self-reprogrammable under Software Control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Programmable Brown-out Reset (BOR)
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Programmable Code Protection
- In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug via Two Pins

### Extreme Low-Power Management PIC18LF1XK22 with XLP Technology

- Sleep mode: 34 nA
- Watchdog Timer: 460 nA
- Timer1 Oscillator: 650 nA @ 32 kHz

### Analog Features

- Analog-to-Digital Converter (ADC) module
  - 10-bit resolution, 12 channels
  - Auto-acquisition capability
  - Conversion available during Sleep
- Analog Comparator module:
  - Two rail-to-rail analog comparators
  - Independent input multiplexing
  - Inputs and outputs externally accessible
- Voltage Reference module:
  - Fixed Voltage Reference (FVR) with 1.024V, 2.048V and 4.096V output levels
  - 5-bit rail-to-rail resistive Digital-to-Analog Converter (DAC) with positive and negative reference selection

### Peripheral Highlights

- 17 I/O Pins and 1 Input-only Pin:
  - High current sink/source 25 mA/25 mA
  - Programmable weak pull-ups
  - Programmable interrupt-on- change
  - Three external interrupt pins
- Four Timer modules:
  - Three 16-bit timers/counters with prescaler
  - One 8-bit timer/counter with 8-bit period register, prescaler and postscaler
  - Dedicated, low-power Timer1 oscillator
- Enhanced Capture/Compare/PWM (ECCP) module:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and Auto-restart
  - PWM output steering control
- Master Synchronous Serial Port (MSSP) module
  - 3-wire SPI (supports all four SPI modes)
  - $I^2C$ Master and Slave modes (Slave mode address masking)
- Enhanced Universal Synchronous Asynchronous Receiver Transmitter module (EUSART)
  - Supports RS-232, RS-485 and LIN 2.0
  - Auto-Baud Detect
  - Auto Wake-up on Break
- SR Latch (555 Timer) module with:
  - Configurable inputs and outputs
  - Supports mTouch® capacitive sensing applications

© 2009-2016 Microchip Technology Inc.

## PIC18(L)F1XK22 Family Types

| Device | Data Sheet Index | Program Memory | | Data Memory | | Pins | I/O(1) | 10-bit A/D Channels | Comparators | Timers 8-bit/16-bit | ECCP | MSSP | EUSART | SR Latch |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Bytes | Words | SRAM (bytes) | Data EEPROM (bytes) | | | | | | | | | |
| PIC18(L)F13K22 | (1) | 8K | 4K | 256 | 256 | 20 | 18 | 12-ch | 2 | 1 / 3 | 1 | 1 | 1 | Yes |
| PIC18(L)F14K22 | (1) | 16K | 8K | 512 | 256 | 20 | 18 | 12-ch | 2 | 1 / 3 | 1 | 1 | 1 | Yes |

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

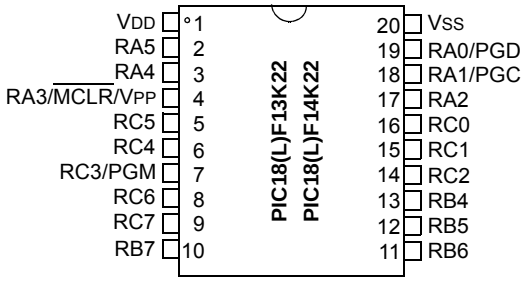
**Data Sheet Index**: (Unshaded devices are described in this document)

**1.** DS40001365 PIC18(L)F1XK22 20-Pin Flash Microcontrollers with XLP Technology

**Note:** For other small form-factor package availability and marking information, please visit **http://www.microchip.com/packaging** or contact your local sales office.

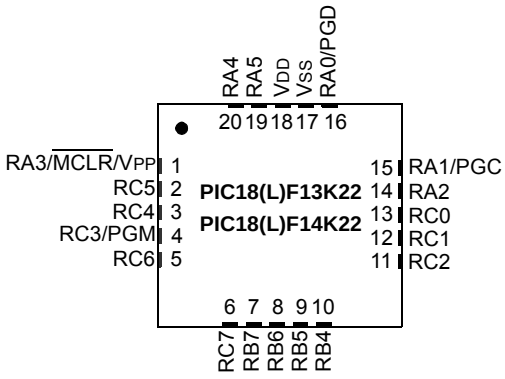
## Pin Diagrams

**FIGURE 1: 20-PIN PDIP, SSOP, SOIC**




**Note:** See Table 1 for location of all peripheral functions.

**FIGURE 2: 20-PIN QFN (4x4)**




**Note:** See Table 1 for location of all peripheral functions.


© 2009-2016 Microchip Technology Inc.

**TABLE 1: 20-PIN ALLOCATION TABLE (PIC18(L)F1XK22)**

| I/O | 20-Pin PDIP/SSOP/SOIC | 20-Pin QFN | Analog | Comparator | Reference | ECCP | EUSART | MSSP | SR Latch | Timers | Interrupts | Pull-up | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 19 | 16 | AN0 | C1IN+ | VREF-/ CVREF(DAC1OUT) | — | — | — | — | — | IOC/INT0 | Y | PGD |
| RA1 | 18 | 15 | AN1 | C12IN0- | VREF+ | — | — | — | — | — | IOC/INT1 | Y | PGC |
| RA2 | 17 | 14 | AN2 | C1OUT | — | — | — | — | SRQ | T0CKI | IOC/INT2 | Y | — |
| RA3 | 4 | 1 | — | — | — | — | — | — | — | — | IOC | Y | $\overline{MCLR}$/VPP |
| RA4 | 3 | 20 | AN3 | — | — | — | — | — | — | — | IOC | Y | OSC2/CLKOUT |
| RA5 | 2 | 19 | — | — | — | — | — | — | — | T13CKI | IOC | Y | OSC1/CLKIN |
| RB4 | 13 | 10 | AN10 | — | — | — | — | SDI/SDA | — | — | IOC | Y | — |
| RB5 | 12 | 9 | AN11 | — | — | — | RX/DT | — | — | — | IOC | Y | — |
| RB6 | 11 | 8 | — | — | — | — | — | SCL/SCK | — | — | IOC | Y | — |
| RB7 | 10 | 7 | — | — | — | — | TX/CK | — | — | — | IOC | Y | — |
| RC0 | 16 | 13 | AN4 | C2IN+ | — | — | — | — | — | — | — | — | — |
| RC1 | 15 | 12 | AN5 | C12IN1- | — | — | — | — | — | — | — | — | — |
| RC2 | 14 | 11 | AN6 | C12IN2- | — | P1D | — | — | — | — | — | — | — |
| RC3 | 7 | 4 | AN7 | C12IN3- | — | P1C | — | — | — | — | — | — | PGM |
| RC4 | 6 | 3 | — | C2OUT | — | P1B | — | — | SRNQ | — | — | — | — |
| RC5 | 5 | 2 | — | — | — | CCP1/P1A | — | — | — | — | — | — | — |
| RC6 | 8 | 5 | AN8 | — | — | — | — | $\overline{SS}$ | — | — | — | — | — |
| RC7 | 9 | 6 | AN9 | — | — | — | — | SDO | — | — | — | — | — |
| — | 1 | 18 | — | — | — | — | — | — | — | — | — | — | VDD |
| — | 20 | 17 | — | — | — | — | — | — | — | — | — | — | VSS |

## Table of Contents

1.0 Device Overview ..... 6
2.0 Oscillator Module ..... 12
3.0 Memory Organization ..... 24
4.0 Flash Program Memory ..... 45
5.0 Data EEPROM Memory ..... 54
6.0 8 x 8 Hardware Multiplier ..... 58
7.0 Interrupts ..... 60
8.0 I/O Ports ..... 73
9.0 Timer0 Module ..... 91
10.0 Timer1 Module ..... 94
11.0 Timer2 Module ..... 100
12.0 Timer3 Module ..... 102
13.0 Enhanced Capture/Compare/PWM (ECCP) Module ..... 106
14.0 Master Synchronous Serial Port (MSSP) Module ..... 127
15.0 Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) ..... 170
16.0 Analog-to-Digital Converter (ADC) Module ..... 197
17.0 Comparator Module ..... 210
18.0 Power-Managed Modes ..... 222
19.0 SR Latch ..... 228
20.0 Fixed Voltage Reference (FVR) ..... 231
21.0 Digital-to-Analog Converter (DAC) Module ..... 233
22.0 Reset ..... 237
23.0 Special Features of the CPU ..... 249
24.0 Instruction Set Summary ..... 265
25.0 Development Support ..... 315
26.0 Electrical Specifications ..... 319
27.0 DC and AC Characteristics Graphs and Charts ..... 356
28.0 Packaging Information ..... 372
Appendix A: Revision History ..... 382
Appendix B: Device Differences ..... 383
The Microchip WebSite ..... 384
Customer Change Notification Service ..... 384
Customer Support ..... 384
Product Identification System ..... 385
Worldwide Sales and Service ..... 387


© 2009-2016 Microchip Technology Inc.

## TO OUR VALUED CUSTOMERS

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

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com**. 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 Website 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., DS30000000A is version A of document DS30000000).

### 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 Website; **http://www.microchip.com**
- Your local Microchip sales office (see last page)

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

### Customer Notification System

Register on our website at **www.microchip.com** to receive the most current information on all of our products.

# 1.0 DEVICE OVERVIEW

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance with the addition of high-endurance, Flash program memory. On top of these features, the PIC18(L)F1XK22 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 XLP TECHNOLOGY

All of the devices in the PIC18(L)F1XK22 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 26.0 "Electrical Specifications"** for values.

### 1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18(L)F1XK22 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 eight 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 64 MHz. Used with the internal oscillator, the PLL gives users a complete selection of clock speeds, from 31 kHz to 64 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.


© 2009-2016 Microchip Technology Inc.

## 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 10K 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 PIC18(L)F1XK22 family introduces an optional extension to the PIC18 instruction set, which adds eight 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 one, two or four 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 four 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 26.0 "Electrical Specifications"** for time-out periods.

## 1.3 Details on Individual Family Members

Devices in the PIC18(L)F1XK22 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 PIC18(L)F13K22
   - 16 Kbytes for PIC18(L)F14K22

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

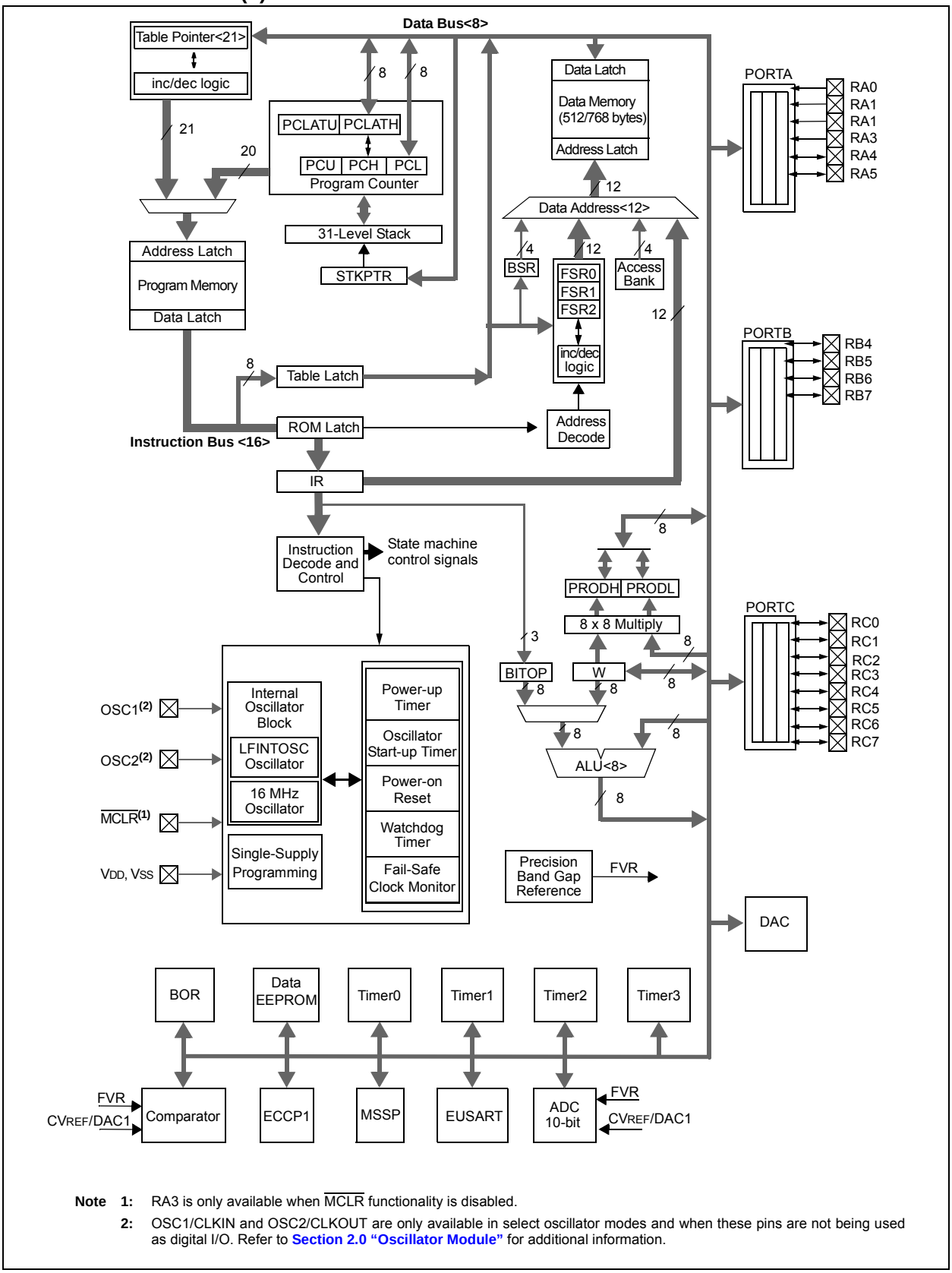
The pinouts for all devices are listed in Table 1 and I/O description are in Table 1-2.

**TABLE 1-1: DEVICE FEATURES FOR THE PIC18(L)F1XK22 (20-PIN DEVICES)**

| Features | PIC18F13K22 | PIC18LF13K22 | PIC18F14K22 | PIC18LF14K22 |
|---|---|---|---|---|
| Voltage Range (1.8 - 5.5V) | 2.3-5.5V | 1.8V-3.6V | 2.3-5.5V | 1.8V-3.6V |
| Program Memory (Bytes) | 8K | | 16K | |
| Program Memory (Instructions) | 4096 | | 8192 | |
| Data Memory (Bytes) | 256 | | 512 | |
| Operating Frequency | DC – 64 MHz | | | |
| Interrupt Sources | 30 | | | |
| I/O Ports | Ports A, B, C | | | |
| Timers | 4 | | | |
| Enhanced Capture/ Compare/PWM Modules | 1 | | | |
| Serial Communications | MSSP, Enhanced USART | | | |
| 10-Bit Analog-to-Digital Module | 12 Input Channels | | | |
| Resets (and Delays) | POR, BOR, RESET Instruction, Stack Full, Stack Underflow, $\overline{\text{MCLR}}$, WDT (PWRT, OST) | | | |
| Instruction Set | 75 Instructions, 83 with Extended Instruction Set Enabled | | | |
| Packages | 20-Pin PDIP, SSOP, SOIC<br>QFN (4x4x0.9mm) | | | |


© 2009-2016 Microchip Technology Inc.

**FIGURE 1-1: PIC18(L)F1XK22 BLOCK DIAGRAM**




**Note 1:** RA3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

**2:** OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Module"** for additional information.

**TABLE 1-2: PIC18(L)F1XK22 PIN SUMMARY**

| Pin Name | Pin Number | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | PDIP/SSOP/SOIC | QFN | | | |
| RA0/AN0/CVREF/VREF-/C1IN+/INT0/PGD | 19 | 16 | | | |
| RA0 | | | I/O | TTL | Digital I/O |
| AN0 | | | I | Analog | ADC channel 0 |
| CVREF/DAC1OUT | | | O | Analog | DAC reference voltage output |
| VREF- | | | I | Analog | ADC and DAC reference voltage (low) input |
| C1IN+ | | | I | Analog | Comparator C1 noninverting input |
| INT0 | | | I | ST | External interrupt 0 |
| PGD | | | I/O | ST | ICSP™ programming data pin |
| RA1/AN1/C12IN0-/VREF+/INT1/PGC | 18 | 15 | | | |
| RA1 | | | I/O | TTL | Digital I/O |
| AN1 | | | I | Analog | ADC channel 1 |
| C12IN0- | | | 1 | Analog | Comparator C1 and C2 inverting input |
| VREF+ | | | I | Analog | ADC and DAC reference voltage (high) input |
| INT1 | | | I | ST | External interrupt 1 |
| PGC | | | I/O | ST | ICSP programming clock pin |
| RA2/AN2/C1OUT/T0CKI/INT2/SRQ | 17 | 14 | | | |
| RA2 | | | I/O | ST | Digital I/O |
| AN2 | | | I | Analog | ADC channel 2 |
| C1OUT | | | — | CMOS | Comparator C1 output |
| T0CKI | | | I | ST | Timer0 external clock input |
| INT2 | | | I | ST | External interrupt 2 |
| SRQ | | | O | CMOS | SR latch output |
| RA3/$\overline{\text{MCLR}}$/VPP | 4 | 1 | | | |
| RA3 | | | I | ST | Digital input |
| $\overline{\text{MCLR}}$ | | | I | ST | Active-low Master Clear with internal pull-up |
| VPP | | | P | — | High voltage programming input |
| RA4/AN3/OSC2/CLKOUT | 3 | 20 | | | |
| RA4 | | | I/O | TTL | Digital I/O |
| AN3 | | | I | Analog | ADC channel 3 |
| OSC2 | | | O | XTAL | Oscillator crystal output. Connect to crystal or resonator in Crystal Oscillator mode |
| CLKOUT | | | O | CMOS | In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate |
| RA5/OSC1/CLKIN/T13CKI | 2 | 19 | | | |
| RA5 | | | I/O | TTL | Digital I/O |
| OSC1 | | | I | XTAL | Oscillator crystal input or external clock input ST buffer when configured in RC mode; analog other wise |
| CLKIN | | | I | CMOS | External clock source input. Always associated with the pin function OSC1 (See related OSC1/CLKIN, OSC2, CLKOUT pins |
| T13CKI | | | I | ST | Timer0 and Timer3 external clock input |
| RB4/AN10/SDI/SDA | 13 | 10 | | | |
| RB4 | | | I/O | TTL | Digital I/O |
| AN10 | | | I | Analog | ADC channel 10 |
| SDI | | | I | ST | SPI data in |
| SDA | | | I/O | ST | $I^2C$ data I/O |

**Legend:** TTL = TTL compatible input
ST = Schmitt Trigger input
O = Output
XTAL = Crystal Oscillator
CMOS = CMOS compatible input or output
I = Input
P = Power


© 2009-2016 Microchip Technology Inc.

**TABLE 1-2: PIC18(L)F1XK22 PIN SUMMARY (CONTINUED)**

| Pin Name | Pin Number | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|
| | PDIP/SSOP/SOIC | QFN | | | |
| RB5/AN11/RX/DT | 12 | 9 | | | |
| RB5 | | | I/O | TLL | Digital I/O |
| AN11 | | | I | Analog | ADC channel 11 |
| RX | | | I | ST | EUSART asynchronous receive |
| DT | | | I/O | ST | EUSART synchronous data (see related RX/TX) |
| RB6/SCK/SCL | 11 | 8 | | | |
| RB6 | | | I/O | TLL | Digital I/O |
| SCK | | | I/O | ST | Synchronous serial clock input/output for SPI mode |
| SCL | | | I/O | ST | Synchronous serial clock input/output for $I^2C$ mode |
| RB7/TX/CK | 10 | 7 | | | |
| RB7 | | | I/O | TLL | Digital I/O |
| TX | | | O | CMOS | EUSART asynchronous transmit |
| CK | | | I/O | ST | EUSART synchronous clock (see related RX/DT) |
| RC0/AN4/C2IN+ | 16 | 13 | | | |
| RC0 | | | I/O | ST | Digital I/O |
| AN4 | | | I | Analog | ADC channel 4 |
| C2IN+ | | | I | Analog | Comparator C2 noninverting input |
| RC1/AN5/C12IN- | 15 | 12 | | | |
| RC1 | | | I/O | ST | Digital I/O |
| AN5 | | | I | Analog | ADC channel 5 |
| C12IN- | | | I | Analog | Comparator C1 and C2 inverting input |
| RC2/AN6/C12IN2-/P1D | 14 | 11 | | | |
| RC2 | | | I/O | ST | Digital I/O |
| AN6 | | | I | Analog | ADC channel 6 |
| C12IN2- | | | I | Analog | Comparator C1 and C2 inverting input |
| P1D | | | O | CMOS | Enhanced CCP1 PWM output |
| RC3/AN7/C12IN3-/P1C/PGM | 7 | 4 | | | |
| RC3 | | | I/O | ST | Digital I/O |
| AN7 | | | I | Analog | ADC channel 7 |
| C12IN3- | | | I | Analog | Comparator C1 and C2 inverting input |
| P1C | | | O | CMOS | Enhanced CCP1 PWM output |
| PGM | | | I/O | ST | Low-Voltage ICSP Programming enable pin |
| RC4/C2OUT/P1B/SRNQ | 6 | 3 | | | |
| RC4 | | | I/O | ST | Digital I/O |
| C2OUT | | | O | CMOS | Comparator C2 output |
| P1B | | | O | CMOS | Enhanced CCP1 PWM output |
| SRNQ | | | O | CMOS | SR latch inverted output |
| RC5/CCP1/P1A | 5 | 2 | | | |
| RC5 | | | I/O | ST | Digital I/O |
| CCP1 | | | I/O | ST | Capture 1 input/Compare 1 output/PWM 1 output |
| P1A | | | O | CMOS | Enhanced CCP1 PWM output |
| RC6/AN8/$\overline{SS}$ | 8 | 5 | | | |
| RC6 | | | I/O | ST | Digital I/O |
| AN8 | | | I | Analog | ADC channel 8 |
| $\overline{SS}$ | | | I | TTL | SPI slave select input |
| RC7/AN9/SDO | 9 | 6 | | | |
| RC7 | | | I/O | ST | Digital I/O |
| AN9 | | | I | Analog | ADC channel 9 |
| SDO | | | O | CMOS | SPI data out |
| VSS | 20 | 17 | P | — | Ground reference for logic and I/O pins |
| VDD | 1 | 18 | P | — | Positive supply for logic and I/O pins |

**Legend:** TTL = TTL compatible input
ST = Schmitt Trigger input
O = Output
XTAL = Crystal Oscillator
CMOS = CMOS compatible input or output
I = Input
P = Power

# 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 Clocks
- 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
- Two-Speed Start-up Mode
- Fail-Safe Clock Monitoring

## 2.2 System Clocks

The PIC18(L)F1XK22 can be operated in 13 different oscillator modes. The user can program these using the available Configuration bits. In addition, clock support functions such as Fail-Safe and two Start-up can also be configured.

The available Primary oscillator options include:

- External Clock, low power (ECL)
- External Clock, medium power (ECM)
- External Clock, high power (ECH)
- External Clock, low power, CLKOUT function on RA4/OSC2 (ECCLKOUTL)
- External Clock, medium power, CLKOUT function on RA4/OSC2 (ECCLKOUTM)
- External Clock, high power, CLKOUT function on RA4/OSC2 (ECCLKOUTH)
- External Crystal (XT)
- High-speed Crystal (HS)
- Low-power crystal (LP)
- External Resistor/Capacitor (EXTRC)
- External RC, CLKOUT function on RA4/OSC2
- 31.25 kHz – 16 MHz internal oscillator (INTOSC)
- 31.25 kHz – 16 MHz internal oscillator, CLKOUT function on RA4/OSC2

Additionally, the 4x PLL may be enabled in hardware or software (under certain conditions) for increased oscillator speed.

## 2.3 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 IRCF 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 IRCF<2:0> = 000 and the INTSRC bit is clear. All other combinations of the IRCF bits and the INTSRC bit will select the HFINTOSC as the system clock.

## 2.4 Primary External Oscillator

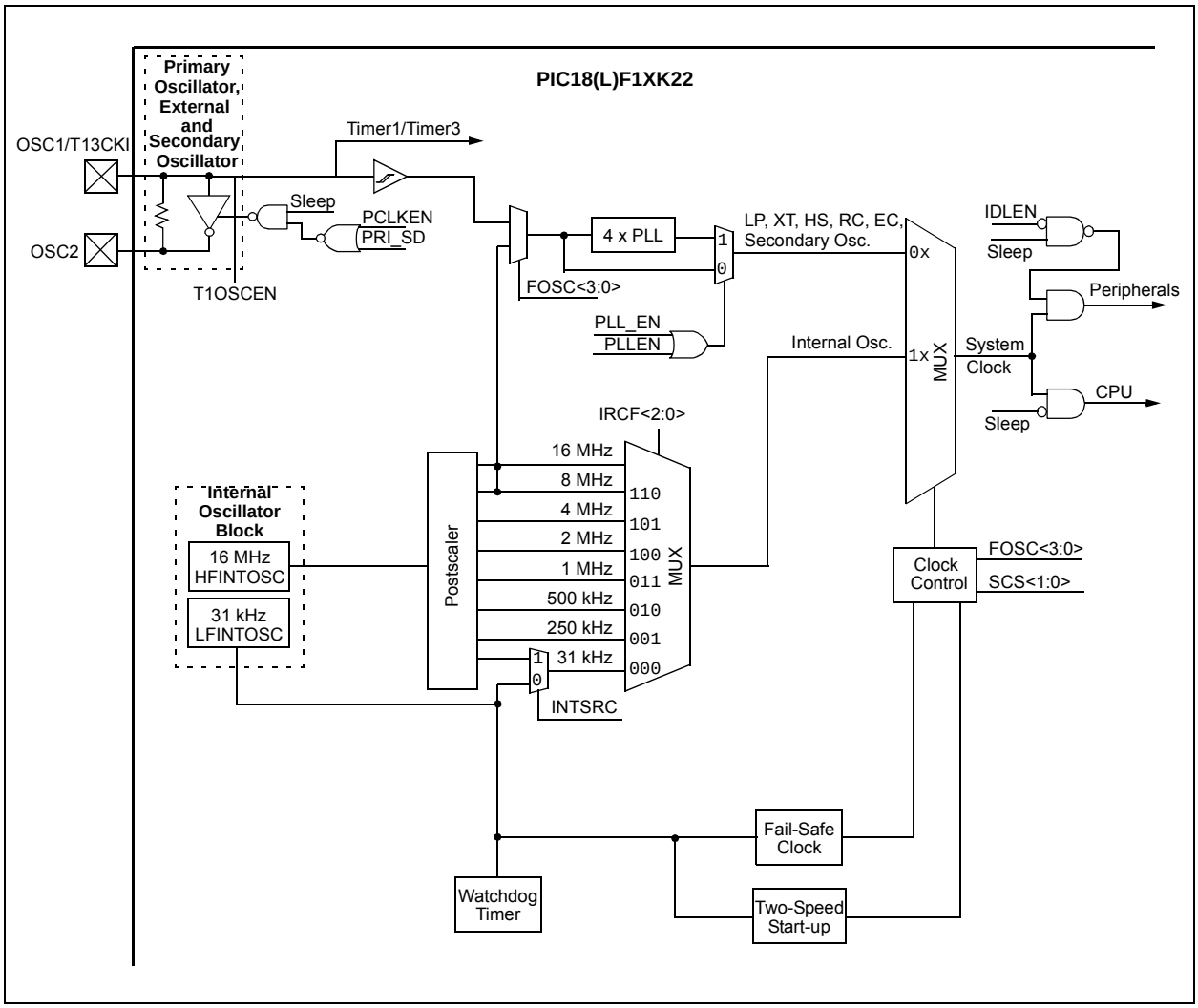
The Primary External Oscillator's mode of operation is selected by setting the FOSC<3:0> bits of the CONFIG1H Configuration register. The oscillator can be set to the following modes:

- 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.


© 2009-2016 Microchip Technology Inc.

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




| **Note:** | If using a low-frequency external oscillator and want to multiple it by 4 via PLL, the ideal input frequency is from 4 MHz to 16 MHz. |
|---|---|

### 2.4.1 PRIMARY EXTERNAL OSCILLATOR SHUTDOWN

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. |

### 2.4.2 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.

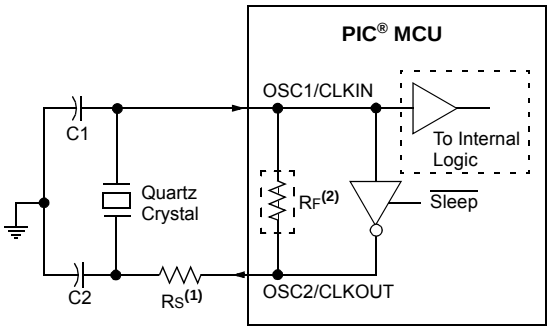
**LP** Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is best suited to drive resonators with a low drive level specification, for example, tuning fork type crystals.

**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)**



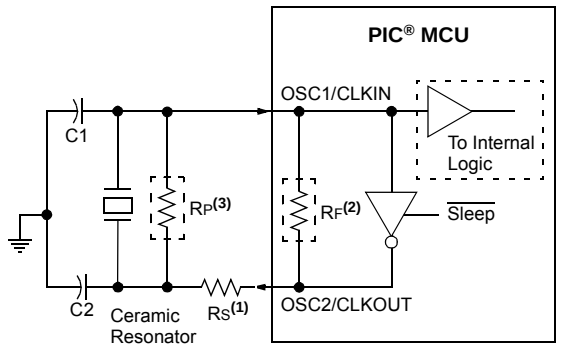

**Note 1:** A series resistor (RS) may be required for quartz crystals with low drive level.

**2:** The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

| | |
|---|---|
| **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:<br>• *AN826, Crystal Oscillator Basics and Crystal Selection for rfPIC® and PICmicro® Devices* (DS00826)<br>• *AN849, Basic PICmicro® Oscillator Design* (DS00849)<br>• *AN943, Practical PICmicro® Oscillator Analysis and Design* (DS00943)<br>• *AN949, Making Your Oscillator Work* (DS00949) |


© 2009-2016 Microchip Technology Inc.

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




**Note 1:** A series resistor (RS) may be required for ceramic resonators with low drive level.

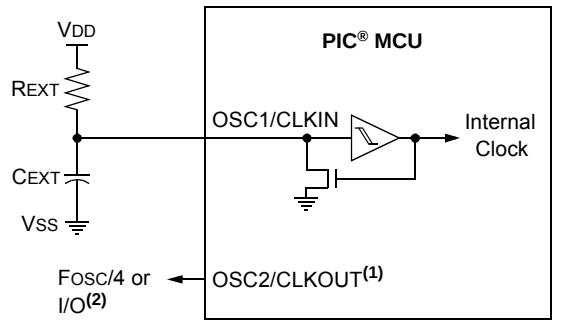
**2:** The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

**3:** An additional parallel feedback resistor (RP) may be required for proper ceramic resonator operation.

### 2.4.3 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 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 RC oscillator divided by 4. Figure 2-4 shows the external RC mode connections.

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




Recommended values: 10 kΩ ≤ REXT ≤ 100 kΩ
CEXT > 20 pF

**Note 1:** Alternate pin functions are listed in **Section 1.0 "Device Overview"**.

**2:** Output depends upon RC or RCIO clock mode.

The RC oscillator frequency is a function of the supply voltage, the resistor REXT, the capacitor CEXT and the operating temperature. Other factors affecting the oscillator frequency are:

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

### 2.4.4 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 kHz |
| EC | Medium power | 250 kHz – 4 MHz |
| EC | High power | 4 – 64 MHz |

## 2.5 Secondary External Oscillator

The Secondary External Oscillator is designed to drive an external 32.768 kHz crystal. This oscillator is enabled or disabled by the T1OSCEN bit of the T1CON register. See **Section 10.0 "Timer1 Module"** for more information.

## 2.6 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.

### 2.6.1 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 (PWRTEN = 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

### 2.6.2 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 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


© 2009-2016 Microchip Technology Inc.

## 2.7 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(1) | 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 **IDLEN:** Idle Enable bit
1 = Device enters Idle mode on SLEEP instruction
0 = Device enters Sleep mode on SLEEP instruction

bit 6-4 **IRCF<2:0>:** Internal Oscillator Frequency Select bits
111 = 16 MHz
110 = 8 MHz
101 = 4 MHz
100 = 2 MHz
011 = 1 MHz(3)
010 = 500 kHz
001 = 250 kHz
000 = 31 kHz(2)

bit 3 **OSTS:** Oscillator Start-up Time-out Status bit(1)
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 **HFIOFS:** HFINTOSC Frequency Stable bit
1 = HFINTOSC frequency is stable
0 = HFINTOSC frequency is not stable

bit 1-0 **SCS<1:0>:** 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.

**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 | | | | | | | 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 7-3 **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


© 2009-2016 Microchip Technology Inc.

### 2.7.1 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 PLLEN bits, which control certain features of the internal oscillator block.

The INTSRC bit allows users to select which internal oscillator provides the clock source when the 31 kHz frequency option is selected. This is covered in greater detail in **Section 2.6.1 "LFINTOSC"**.

The PLLEN bit controls the operation of the frequency multiplier. For more details about the function of the PLLEN bit see **Section 2.10 "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 | PLLEN | 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 **PLLEN:** Frequency Multiplier PLL bit
1 = PLL enabled (for HFINTOSC 8 MHz and 16 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

## 2.8 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 2.11 "Two-Speed Start-up Mode"** for more information.

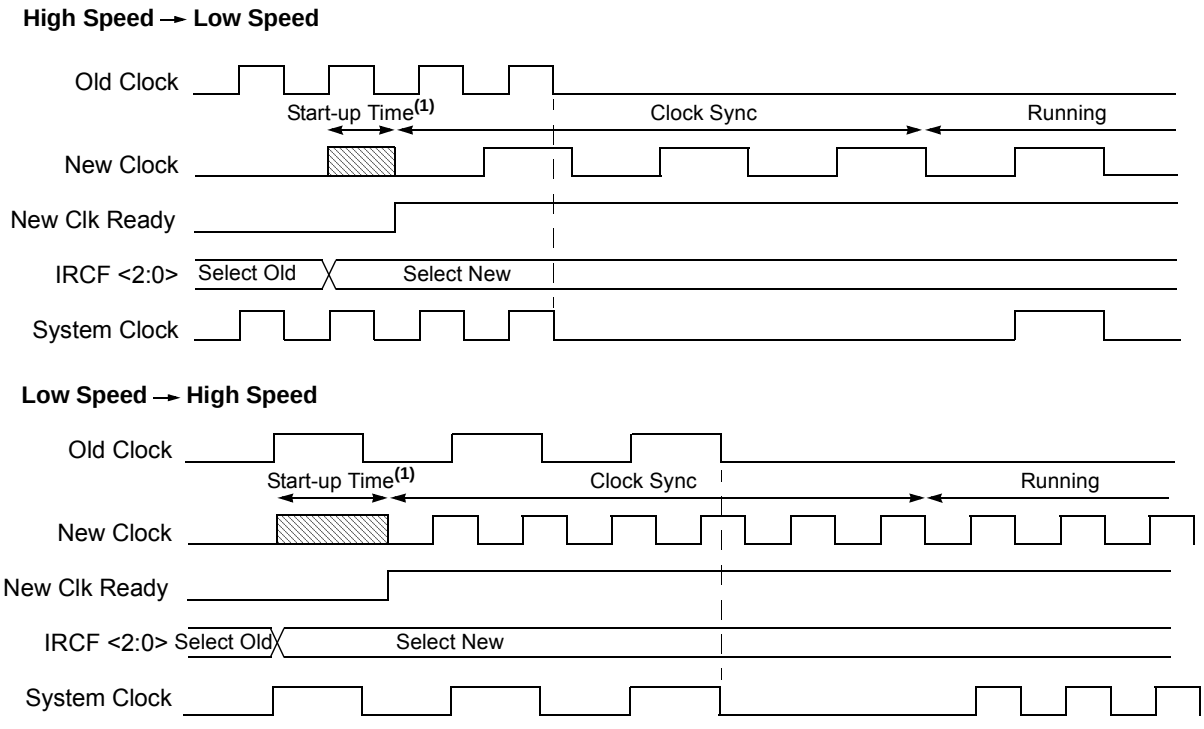
## 2.9 Clock Switching

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**




**Note 1:** Start-up time includes TOST (1024 TOSC) for external clocks, plus TPLL (approx. 2 ms) for HSPLL mode.


© 2009-2016 Microchip Technology Inc.

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

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

## 2.10 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 or 64 MHz with the HFINTOSC. The PLL is designed for an input frequency from 4 MHz to 16 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 PLL_EN bit of the CONFIG1H Configuration register and the PLLEN bit of the OSCTUNE register. The PLL is enabled when the PLL_EN bit is set and it is under software control when the PLL_EN bit is cleared. Refer to Table 2-3 and Table 2-4 for more information.

**TABLE 2-3: PLL CONFIGURATION**

| PLL_EN | PLLEN | PLL Status |
|---|---|---|
| 1 | x | PLL enabled |
| 0 | 1 | PLL enabled |
| 0 | 0 | PLL disabled |

**TABLE 2-4: PLL CONFIG1H/SOFTWARE ENABLE CLOCK SOURCE RESTRICTIONS**

| Mode | PLL CONFIG1H Enable (PLL_EN) | PLL Software Enable (PLLEN) |
|---|---|---|
| LP | Yes | No |
| XT | Yes | No |
| HS | Yes | No |
| EC | Yes | No |
| EXTRC | Yes | No |
| LF INTOSC | No | No |
| HF INTOSC | 8/16 MHz | 8/16 MHz |

## 2.11 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

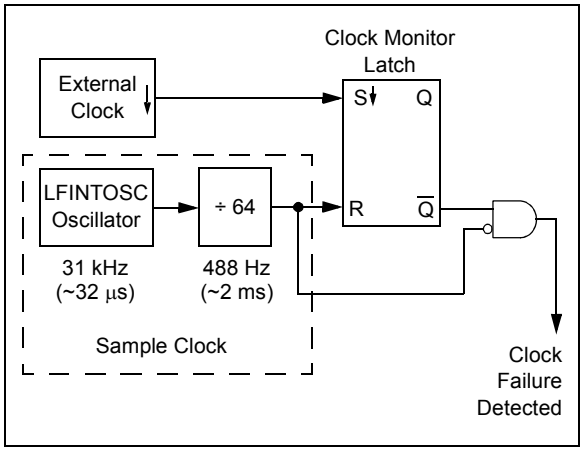
The OSTS bit of the OSCCON register reports which oscillator the device is currently using for operation. The device is running from the oscillator defined by the FOSC bits of the CONFIG1H Configuration register when the OSTS bit is set. The device is running from the internal oscillator when the OSTS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

© 2009-2016 Microchip Technology Inc.

## 2.12 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**




### 2.12.1 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.

### 2.12.2 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 IRCF<2:0> bits of the OSCCON register. This allows the internal oscillator to be configured before a failure occurs.

### 2.12.3 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.

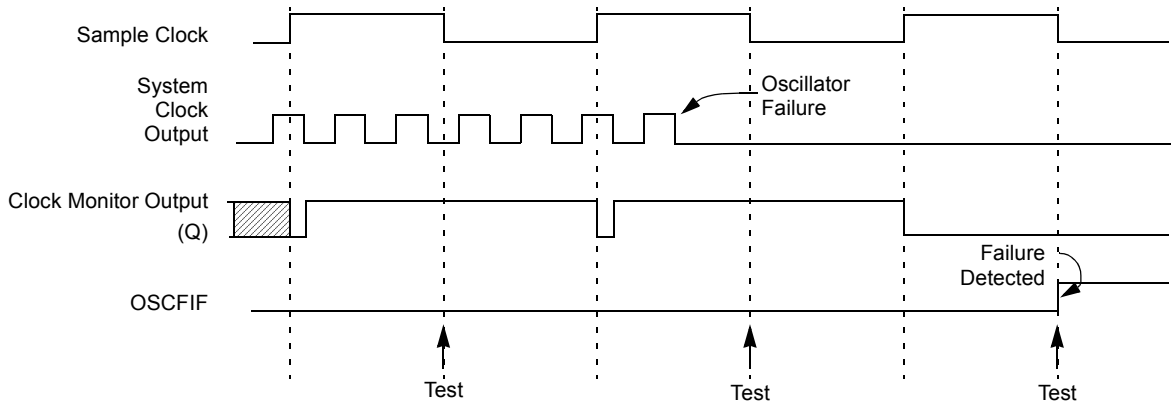
### 2.12.4 RESET OR WAKE-UP FROM SLEEP

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. |


© 2009-2016 Microchip Technology Inc.

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




**Note:** The system clock is normally at a much higher frequency than the sample clock. The relative frequencies in this example have been chosen for clarity.

**TABLE 2-5: 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 | PLL_EN | FOSC3 | FOSC2 | FOSC1 | FOSC0 | 251 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| OSCCON | IDLEN | IRCF2 | IRCF1 | IRCF0 | OSTS | HFIOFS | SCS1 | SCS0 | 246 |
| OSCCON2 | — | — | — | — | — | PRI_SD | HFIOFL | LFIOFS | 246 |
| OSCTUNE | INTSRC | PLLEN | TUN5 | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 | 248 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | — | TMR3IP | — | 248 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | — | TMR3IE | — | 248 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | — | TMR3IF | — | 248 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 246 |

**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 $\overline{\text{MCLR}}$ Reset and Watchdog Timer Reset during normal operation.

## 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 (PC) 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).

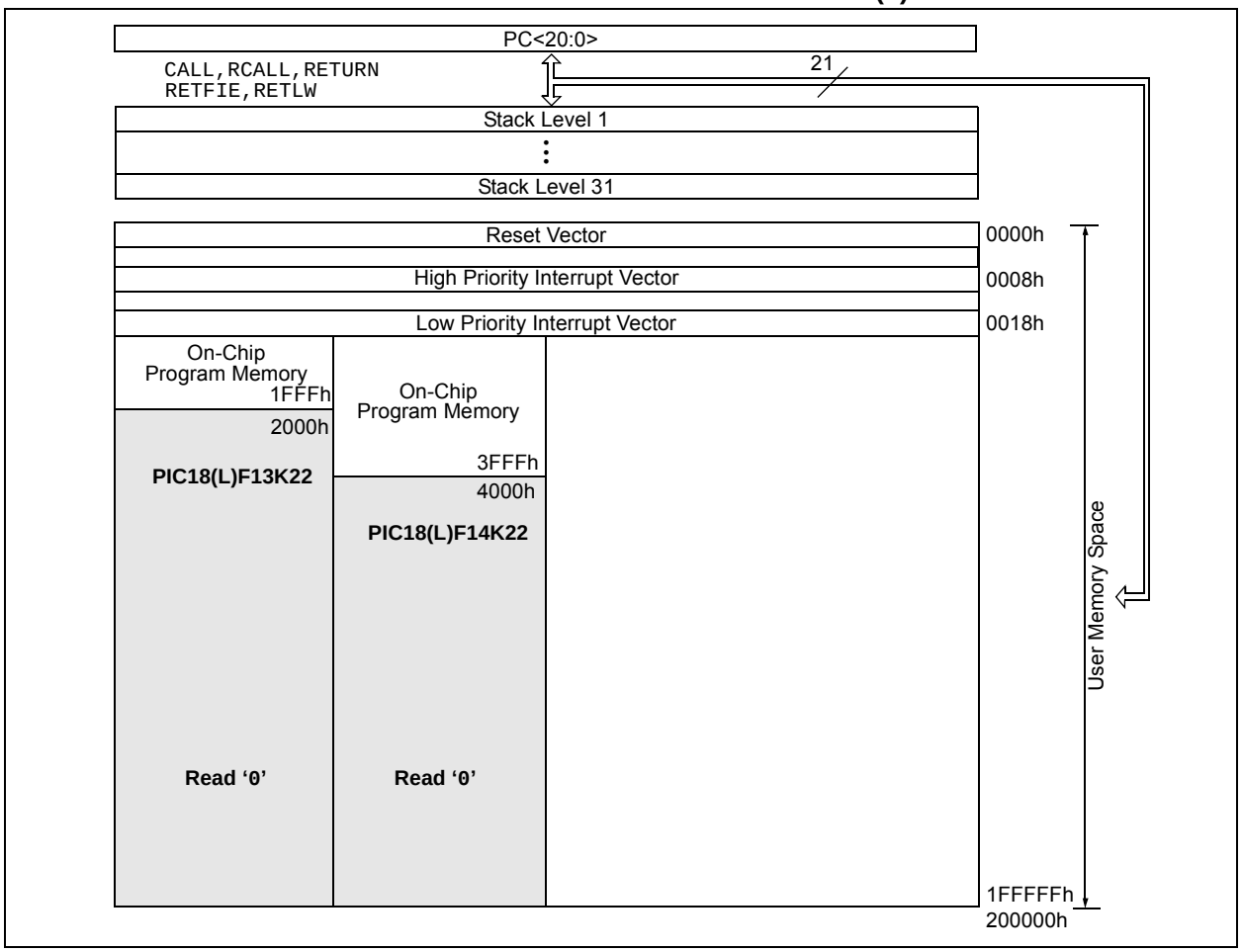
This family of devices contain the following:

- PIC18(L)F13K22: 8 Kbytes of Flash Memory, up to 4,096 single-word instructions
- PIC18(L)F14K22: 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 PIC18(L)F1XK22 devices is shown in Figure 3-1. Memory block details are shown in Figure 3-2.

**FIGURE 3-1: PROGRAM MEMORY MAP AND STACK FOR PIC18(L)F1XK22 DEVICES**





© 2009-2016 Microchip Technology Inc.

### 3.1.1 PROGRAM COUNTER

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

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

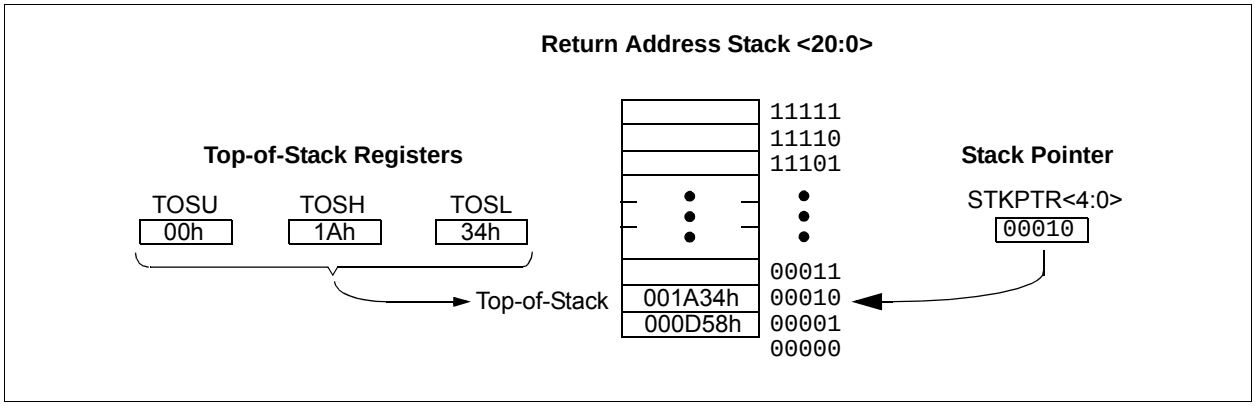
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits indicate if the stack is full or has overflowed or has underflowed.

#### 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 (Figure 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 STKOVF bit is cleared by software or by a POR.

The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 23.1 "Configuration Bits"** for a description of the device Configuration bits.) If STVREN is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKOVF bit and reset the device. The STKOVF bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKOVF 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 |
|---|---|---|---|---|---|---|---|
| STKOVF(1) | STKUNF(1) | — | 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 **STKOVF:** Stack Overflow Flag bit(1)
1 = Stack became full or overflowed
0 = Stack has not become full or overflowed

bit 6 **STKUNF:** Stack Underflow Flag bit(1)
1 = Stack underflow occurred
0 = Stack underflow did not occur

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

bit 4-0 **SP<4:0>:** Stack Pointer Location bits

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


© 2009-2016 Microchip Technology Inc.

#### 3.1.2.4 Stack Overflow 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 STKOVF or STKUNF bit and then cause a device Reset. When STVREN is cleared, a full or underflow condition will set the appropriate STKOVF or STKUNF bit but not cause a device Reset. The STKOVF 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.

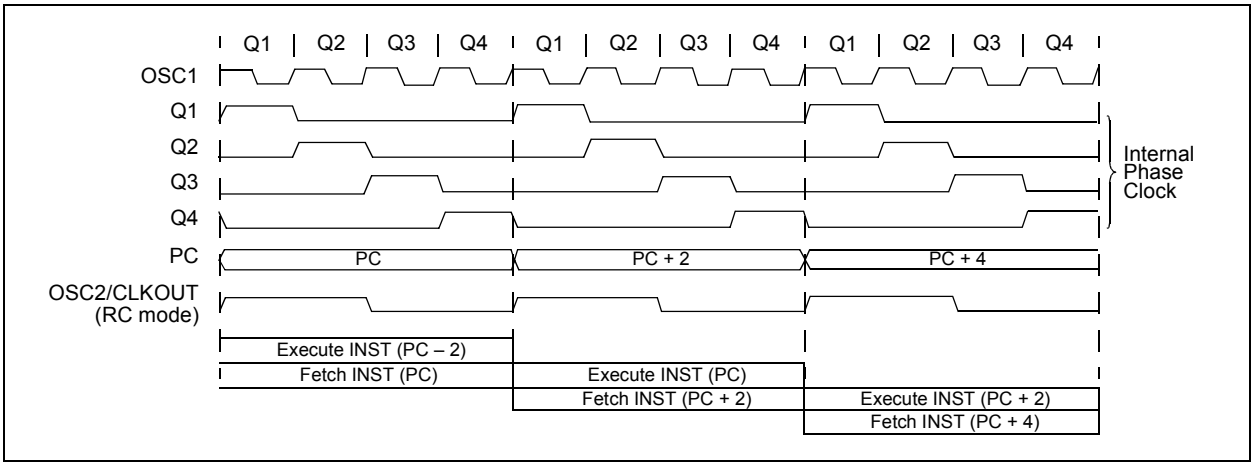
### 3.2.2 INSTRUCTION FLOW/PIPELINING

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.


© 2009-2016 Microchip Technology Inc.

### 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 24.0 "Instruction Set Summary"** provides further details of the instruction set.

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

Program Memory Byte Locations →

| | | | LSB = 1 | LSB = 0 | Word Address ↓ |
|---|---|---|---|---|---|
| | | | | | 000000h |
| | | | | | 000002h |
| | | | | | 000004h |
| | | | | | 000006h |
| Instruction 1: | MOVLW | 055h | 0Fh | 55h | 000008h |
| Instruction 2: | GOTO | 0006h | EFh | 03h | 00000Ah |
| | | | F0h | 00h | 00000Ch |
| Instruction 3: | MOVFF | 123h, 456h | C1h | 23h | 00000Eh |
| | | | F4h | 56h | 000010h |
| | | | | | 000012h |
| | | | | | 000014h |

### 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 PIC18(L)F1XK22 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 3.3.2 "Access Bank"** provides a detailed description of the Access RAM.

### 3.3.1 BANK SELECT REGISTER (BSR)

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

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

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

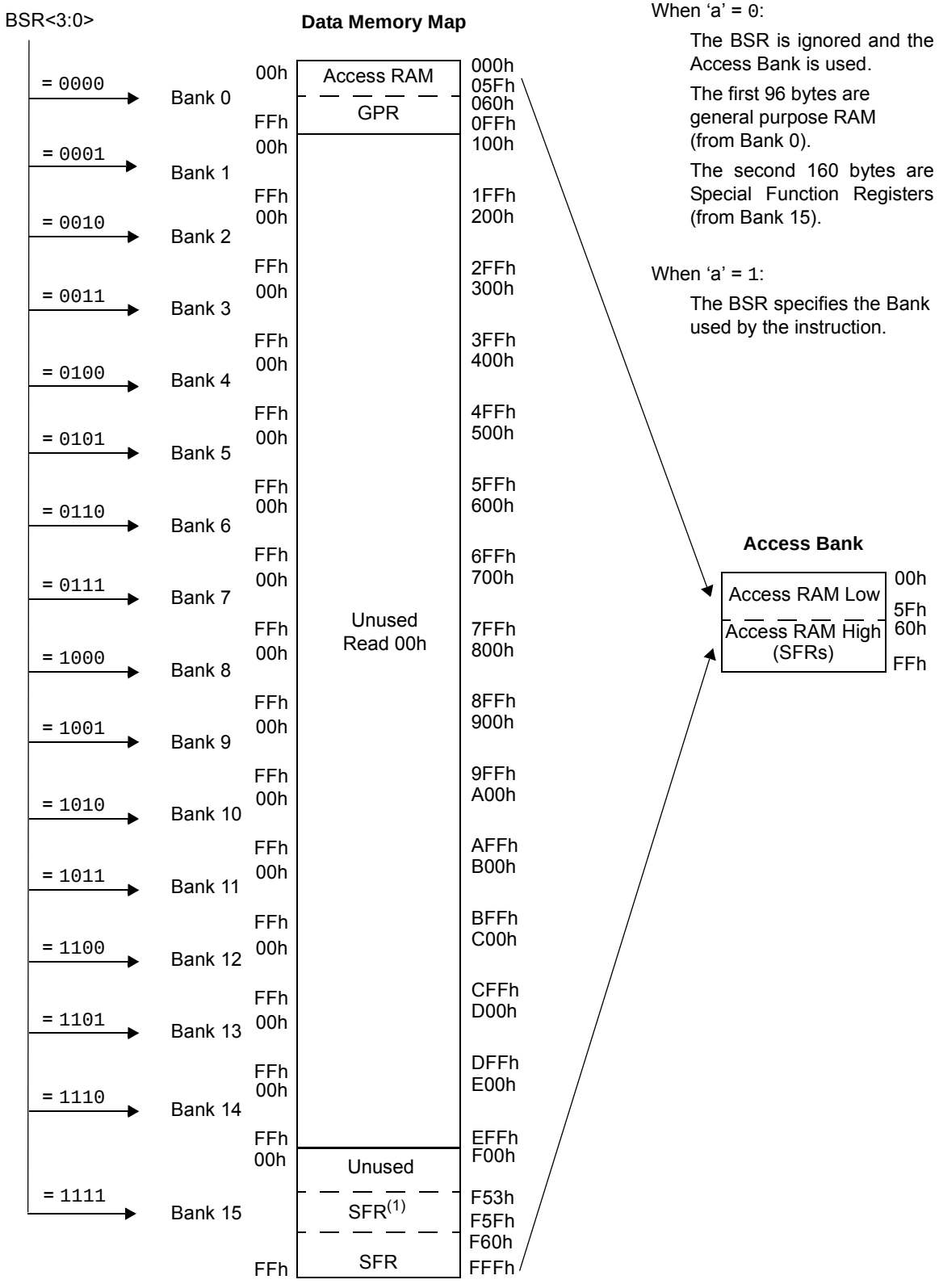
Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 0Fh will end up resetting the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

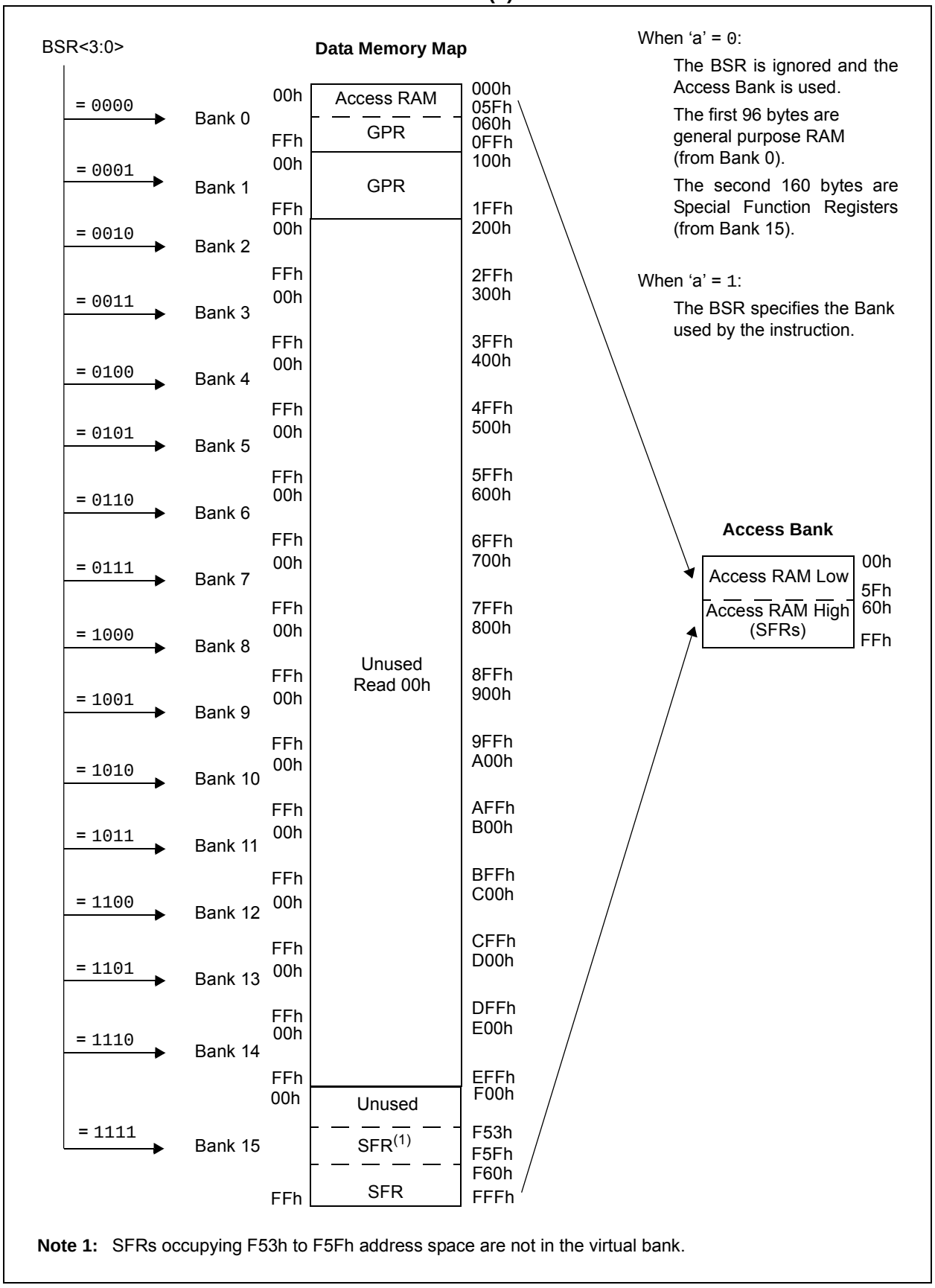

© 2009-2016 Microchip Technology Inc.

**FIGURE 3-5: DATA MEMORY MAP FOR PIC18(L)F13K22 DEVICES**




**Note 1:** SFRs occupying F53h to F5Fh address space are not in the virtual bank.

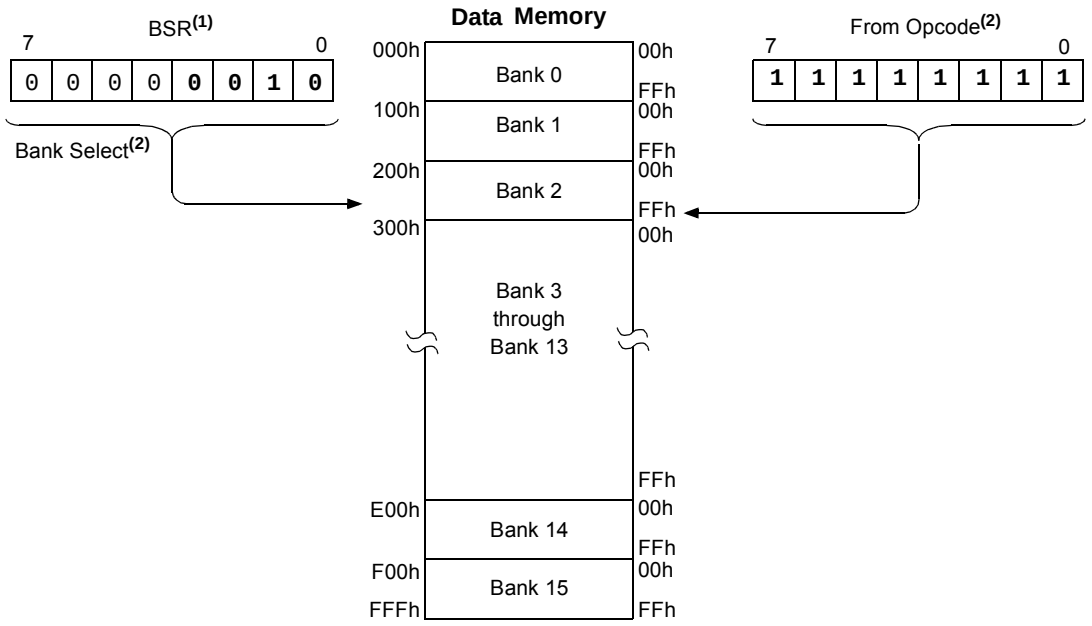
**FIGURE 3-6: DATA MEMORY MAP FOR PIC18(L)F14K22 DEVICES**




**Note 1:** SFRs occupying F53h to F5Fh address space are not in the virtual bank.


© 2009-2016 Microchip Technology Inc.

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




**Note 1:** The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.

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

### 3.3.2 ACCESS BANK

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

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Block 15. The lower half is known as the “Access RAM” and is composed of GPRs. This upper half is 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”**.

### 3.3.3 GENERAL PURPOSE REGISTER FILE

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

### 3.3.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top 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.


© 2009-2016 Microchip Technology Inc.

**TABLE 3-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F1XK22 DEVICES**

| Address | Name | Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|---|---|
| FFFh | TOSU | FD7h | TMR0H | FAFh | SPBRG | F87h | —(2) | F5Fh | —(2) |
| FFEh | TOSH | FD6h | TMR0L | FAEh | RCREG | F86h | —(2) | F5Eh | —(2) |
| FFDh | TOSL | FD5h | T0CON | FADh | TXREG | F85h | —(2) | F5Dh | —(2) |
| FFCh | STKPTR | FD4h | —(2) | FACh | TXSTA | F84h | —(2) | F5Ch | —(2) |
| FFBh | PCLATU | FD3h | OSCCON | FABh | RCSTA | F83h | —(2) | F5Bh | —(2) |
| FFAh | PCLATH | FD2h | OSCCON2 | FAAh | —(2) | F82h | PORTC | F5Ah | —(2) |
| FF9h | PCL | FD1h | WDTCON | FA9h | EEADR | F81h | PORTB | F59h | —(2) |
| FF8h | TBLPTRU | FD0h | RCON | FA8h | EEDATA | F80h | PORTA | F58h | —(2) |
| FF7h | TBLPTRH | FCFh | TMR1H | FA7h | EECON2(1) | F7Fh | ANSELH | F57h | —(2) |
| FF6h | TBLPTRL | FCEh | TMR1L | FA6h | EECON1 | F7Eh | ANSEL | F56h | —(2) |
| FF5h | TABLAT | FCDh | T1CON | FA5h | —(2) | F7Dh | —(2) | F55h | —(2) |
| FF4h | PRODH | FCCh | TMR2 | FA4h | —(2) | F7Ch | —(2) | F54h | —(2) |
| FF3h | PRODL | FCBh | PR2 | FA3h | —(2) | F7Bh | —(2) | F53h | —(2) |
| 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(1) | FC7h | SSPSTAT | F9Fh | IPR1 | F77h | WPUA | | |
| FEEh | POSTINC0(1) | FC6h | SSPCON1 | F9Eh | PIR1 | F76h | SLRCON | | |
| FEDh | POSTDEC0(1) | FC5h | SSPCON2 | F9Dh | PIE1 | F75h | —(2) | | |
| FECh | PREINC0(1) | FC4h | ADRESH | F9Ch | —(2) | F74h | —(2) | | |
| FEBh | PLUSW0(1) | 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(1) | FBFh | CCPR1H | F97h | —(2) | F6Fh | SSPMASK | | |
| FE6h | POSTINC1(1) | FBEh | CCPR1L | F96h | —(2) | F6Eh | —(2) | | |
| FE5h | POSTDEC1(1) | FBDh | CCP1CON | F95h | —(2) | F6Dh | CM1CON0 | | |
| FE4h | PREINC1(1) | FBCh | VREFCON2 | F94h | TRISC | F6Ch | CM2CON1 | | |
| FE3h | PLUSW1(1) | FBBh | VREFCON1 | F93h | TRISB | F6Bh | CM2CON0 | | |
| FE2h | FSR1H | FBAh | VREFCON0 | F92h | TRISA | F6Ah | —(2) | | |
| FE1h | FSR1L | FB9h | PSTRCON | F91h | —(2) | F69h | SRCON1 | | |
| FE0h | BSR | FB8h | BAUDCON | F90h | —(2) | F68h | SRCON0 | | |
| FDFh | INDF2(1) | FB7h | PWM1CON | F8Fh | —(2) | F67h | —(2) | | |
| FDEh | POSTINC2(1) | FB6h | ECCP1AS | F8Eh | —(2) | F66h | —(2) | | |
| FDDh | POSTDEC2(1) | FB5h | —(2) | F8Dh | —(2) | F65h | —(2) | | |
| FDCh | PREINC2(1) | FB4h | —(2) | F8Ch | —(2) | F64h | —(2) | | |
| FDBh | PLUSW2(1) | FB3h | TMR3H | F8Bh | LATC | F63h | —(2) | | |
| FDAh | FSR2H | FB2h | TMR3L | F8Ah | LATB | F62h | —(2) | | |
| FD9h | FSR2L | FB1h | T3CON | F89h | LATA | F61h | —(2) | | |
| FD8h | STATUS | FB0h | SPBRGH | F88h | —(2) | F60h | —(2) | | |

**Legend:** = Unimplemented data memory locations, read as '0',

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

**2:** Unimplemented registers are read as '0'.

**TABLE 3-2: REGISTER FILE SUMMARY (PIC18(L)F1XK22)**

| 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 | 245, 25 |
| TOSH | Top-of-Stack, High Byte (TOS<15:8>) | | | | | | | | 0000 0000 | 245, 25 |
| TOSL | Top-of-Stack, Low Byte (TOS<7:0>) | | | | | | | | 0000 0000 | 245, 25 |
| STKPTR | STKOVF | STKUNF | — | SP4 | SP3 | SP2 | SP1 | SP0 | 00-0 0000 | 245, 26 |
| PCLATU | — | — | — | Holding Register for PC<20:16> | | | | | ---0 0000 | 245, 25 |
| PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000 0000 | 245, 25 |
| PCL | PC, Low Byte (PC<7:0>) | | | | | | | | 0000 0000 | 245, 25 |
| TBLPTRU | — | — | — | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | ---0 0000 | 245, 48 |
| TBLPTRH | Program Memory Table Pointer, High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 | 245, 48 |
| TBLPTRL | Program Memory Table Pointer, Low Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 | 245, 48 |
| TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 | 245, 48 |
| PRODH | Product Register, High Byte | | | | | | | | xxxx xxxx | 245, 58 |
| PRODL | Product Register, Low Byte | | | | | | | | xxxx xxxx | 245, 58 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 0000 000x | 245, 62 |
| INTCON2 | $\overline{\text{RABPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RABIP | 1111 -1-1 | 245, 63 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 11-0 0-00 | 245, 64 |
| INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) | | | | | | | | N/A | 245, 41 |
| POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | N/A | 245, 41 |
| POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) | | | | | | | | N/A | 245, 41 |
| PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) | | | | | | | | N/A | 245, 41 |
| 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 | 245, 41 |
| FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0, High Byte | | | | ---- 0000 | 245, 41 |
| FSR0L | Indirect Data Memory Address Pointer 0, Low Byte | | | | | | | | xxxx xxxx | 245, 41 |
| WREG | Working Register | | | | | | | | xxxx xxxx | 245 |
| INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) | | | | | | | | N/A | 245, 41 |
| POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | N/A | 245, 41 |
| POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) | | | | | | | | N/A | 245, 41 |
| PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) | | | | | | | | N/A | 245, 41 |
| 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 | 245, 41 |
| FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1, High Byte | | | | ---- 0000 | 246, 41 |
| FSR1L | Indirect Data Memory Address Pointer 1, Low Byte | | | | | | | | xxxx xxxx | 246, 41 |
| BSR | — | — | — | — | Bank Select Register | | | | ---- 0000 | 246, 30 |
| INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) | | | | | | | | N/A | 246, 41 |
| POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | N/A | 246, 41 |
| POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) | | | | | | | | N/A | 246, 41 |
| PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) | | | | | | | | N/A | 246, 41 |
| 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 | 246, 41 |
| FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2, High Byte | | | | ---- 0000 | 246, 41 |
| FSR2L | Indirect Data Memory Address Pointer 2, Low Byte | | | | | | | | xxxx xxxx | 246, 41 |
| STATUS | — | — | — | N | OV | Z | DC | C | ---x xxxx | 246, 39 |

**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 22.4 "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:** Unimplemented, read as '1'.


© 2009-2016 Microchip Technology Inc.

**TABLE 3-2: REGISTER FILE SUMMARY (PIC18(L)F1XK22) (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 | 246, 92 |
| TMR0L | Timer0 Register, Low Byte | | | | | | | | xxxx xxxx | 246, 92 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 1111 1111 | 246, 91 |
| OSCCON | IDLEN | IRCF2 | IRCF1 | IRCF0 | OSTS | HFIOFS | SCS1 | SCS0 | 0011 qq00 | 246, 17 |
| OSCCON2 | — | — | — | — | — | PRI_SD | HFIOFL | LFIOFS | ---- -10x | 246, 18 |
| WDTCON | — | — | — | — | — | — | — | SWDTEN | --- ---0 | 246, 260 |
| RCON | IPEN | SBOREN(1) | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 0q-1 11q0 | 237, 246, 71 |
| TMR1H | Timer1 Register, High Byte | | | | | | | | xxxx xxxx | 246, 94 |
| TMR1L | Timer1 Register, Low Bytes | | | | | | | | xxxx xxxx | 246, 94 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{T1SYNC}$ | TMR1CS | TMR1ON | 0000 0000 | 246, 94 |
| TMR2 | Timer2 Register | | | | | | | | 0000 0000 | 246, 100 |
| PR2 | Timer2 Period Register | | | | | | | | 1111 1111 | 246, 100 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | 246, 100 |
| SSPBUF | SSP Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | 246, 128, 130 |
| SSPADD | SSP Address Register in $I^2C$ Slave Mode. SSP Baud Rate Reload Register in $I^2C$ Master Mode. | | | | | | | | 0000 0000 | 246, 147 |
| SSPSTAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 0000 0000 | 246, 128, 137 |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 246, 128, 138 |
| SSPCON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | 246, 139 |
| ADRESH | A/D Result Register, High Byte | | | | | | | | xxxx xxxx | 247, 197 |
| ADRESL | A/D Result Register, Low Byte | | | | | | | | xxxx xxxx | 247, 197 |
| ADCON0 | — | — | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{DONE}$ | ADON | --00 0000 | 247, 203 |
| ADCON1 | — | — | — | — | PVCFG1 | PVCFG0 | NVCFG1 | NVCFG0 | ---- 0000 | 247, 204 |
| ADCON2 | ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 0-00 0000 | 247, 205 |
| CCPR1H | Capture/Compare/PWM Register 1, High Byte | | | | | | | | xxxx xxxx | 247, 126 |
| CCPR1L | Capture/Compare/PWM Register 1, Low Byte | | | | | | | | xxxx xxxx | 247, 126 |
| CCP1CON | P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 0000 0000 | 247, 106 |
| VREFCON2 | — | — | — | DAC1R<4:0> | | | | | ---0 0000 | 247, 236 |
| VREFCON1 | D1EN | D1LPS | DAC1OE | --- | D1PSS<1:0> | | — | D1NSS | 000- 00-0 | 247, 235 |
| VREFCON0 | FVR1EN | FVR1ST | FVR1S<1:0> | | — | — | — | — | 0001 ---- | 247, 232 |
| PSTRCON | — | — | — | STRSYNC | STRD | STRC | STRB | STRA | ---0 0001 | 247, 123 |
| BAUDCON | ABDOVF | RCIDL | DTRXP | CKTXP | BRG16 | — | WUE | ABDEN | 0100 0-00 | 247, 181 |
| PWM1CON | PRSEN | PDC6 | PDC5 | PDC4 | PDC3 | PDC2 | PDC1 | PDC0 | 0000 0000 | 247, 122 |
| ECCP1AS | ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 | 0000 0000 | 247, 118 |
| TMR3H | Timer3 Register, High Byte | | | | | | | | xxxx xxxx | 247, 102 |
| TMR3L | Timer3 Register, Low Byte | | | | | | | | xxxx xxxx | 247, 102 |
| T3CON | RD16 | — | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{T3SYNC}$ | TMR3CS | TMR3ON | 0-00 0000 | 247, 102 |

**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 22.4 "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:** Unimplemented, read as '1'.

**TABLE 3-2: REGISTER FILE SUMMARY (PIC18(L)F1XK22) (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 | 247, 182 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 0000 0000 | 247, 182 |
| RCREG | EUSART Receive Register | | | | | | | | 0000 0000 | 247, 175 |
| TXREG | EUSART Transmit Register | | | | | | | | 0000 0000 | 247, 172 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 247, 179 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 247, 180 |
| EEADR | EEADR7 | EEADR6 | EEADR5 | EEADR4 | EEADR3 | EEADR2 | EEADR1 | EEADR0 | 0000 0000 | 247, 45, 54 |
| EEDATA | EEPROM Data Register | | | | | | | | 0000 0000 | 247, 45, 54 |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | 0000 0000 | 247, 45, 54 |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | xx-0 x000 | 247, 45, 54 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | — | TMR3IP | — | 1111 1-1- | 248, 70 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | — | TMR3IF | — | 0000 0-0- | 248, 66 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | — | TMR3IE | — | 0000 0-0- | 248, 68 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | -111 1111 | 248, 69 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | -000 0000 | 248, 65 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | -000 0000 | 248, 67 |
| OSCTUNE | INTSRC | PLLEN | TUN5 | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 | 0000 0000 | 248, 19 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 1111 1111 | 248, 84 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | — | — | — | — | 1111 ---- | 248, 80 |
| TRISA | — | — | TRISA5 | TRISA4 | —(3) | TRISA2 | TRISA1 | TRISA0 | --11 1111 | 248, 75 |
| LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | xxxx xxxx | 248, 85 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | — | — | — | — | xxxx ---- | 248, 80 |
| LATA | — | — | LATA5 | LATA4 | — | LATA2 | LATA1 | LATA0 | --xx -xxx | 248, 76 |
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | xxxx xxxx | 248, 84 |
| PORTB | RB7 | RB6 | RB5 | RB4 | — | — | — | — | xxxx ---- | 248, 80 |
| PORTA | — | — | RA5 | RA4 | RA3(2) | RA2 | RA1 | RA0 | --xx xxxx | 248, 75 |
| ANSELH | — | — | — | — | ANS11 | ANS10 | ANS9 | ANS8 | ---- 1111 | 248, 89 |
| ANSEL | ANS7 | ANS6 | ANS5 | ANS4 | ANS3 | ANS2 | ANS1 | ANS0 | 1111 1111 | 248, 88 |
| IOCB | IOCB7 | IOCB6 | IOCB5 | IOCB4 | — | — | — | — | 0000 ---- | 248, 81 |
| IOCA | — | — | IOCA5 | IOCA4 | IOCA3 | IOCA2 | IOCA1 | IOCA0 | --00 0000 | 248, 76 |
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | — | — | — | — | 1111 ---- | 248, 81 |
| WPUA | — | — | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 | --11 1111 | 245, 76 |
| SLRCON | — | — | — | — | — | SLRC | SLRB | SLRA | ---- -111 | 248, 90 |
| SSPMSK | MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0 | 1111 1111 | 248, 146 |
| CM1CON0 | C1ON | C1OUT | C1OE | C1POL | C1SP | C1R | C1CH1 | C1CH0 | 0000 0000 | 248, 216 |
| CM2CON1 | MC1OUT | MC2OUT | C1RSEL | C2RSEL | C1HYS | C2HYS | C1SYNC | C2SYNC | 0000 0000 | 248, 220 |
| CM2CON0 | C2ON | C2OUT | C2OE | C2POL | C2SP | C2R | C2CH1 | C2CH0 | 0000 0000 | 248, 217 |
| SRCON1 | SRSPE | SRSCKE | SRSC2E | SRSC1E | SRRPE | SRRCKE | SRRC2E | SRRC1E | 0000 0000 | 248, 230 |
| SRCON0 | SRLEN | SRCLK2 | SRCLK1 | SRCLK0 | SRQEN | SRNQEN | SRPS | SRPR | 0000 0000 | 248, 229 |

**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 22.4 "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:** Unimplemented, read as '1'.


© 2009-2016 Microchip Technology Inc.

### 3.3.5 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 u1uu').

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

For other instructions that do not affect Status bits, see the instruction set summaries in Table 24-2 and Table 24-3.

| **Note:** | The C and DC bits operate as the $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ 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(1) | C(1) |
| bit 7 | | | | | | | bit 0 |

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

bit 7-5 **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/$\overline{\text{Borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
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/$\overline{\text{Borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
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 $\overline{\text{Borrow}}$, 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 3.3.3 "General Purpose Register File"**) or a location in the Access Bank (**Section 3.3.2 "Access Bank"**) as the data source for the instruction.

The Access RAM bit 'a' determines how the address is interpreted. When 'a' is '1', the contents of the BSR (**Section 3.3.1 "Bank Select Register (BSR)"**) are used with the address to determine the complete 12-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

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

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

### 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                   ; YES, continue
```


© 2009-2016 Microchip Technology Inc.

#### 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.

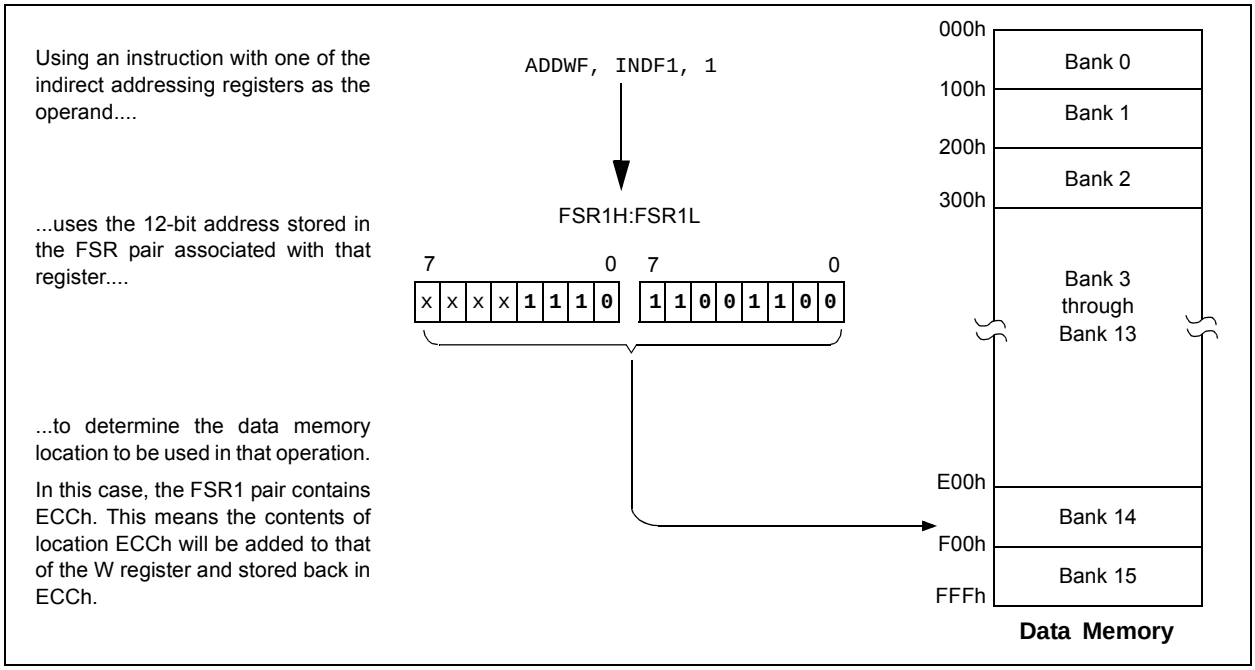
#### 3.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

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, rollovers of the FSRnL register from FFh to 00h carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (e.g., Z, N, OV, etc.).

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 24.2.1 "Extended Instruction Syntax"**.


© 2009-2016 Microchip Technology Inc.

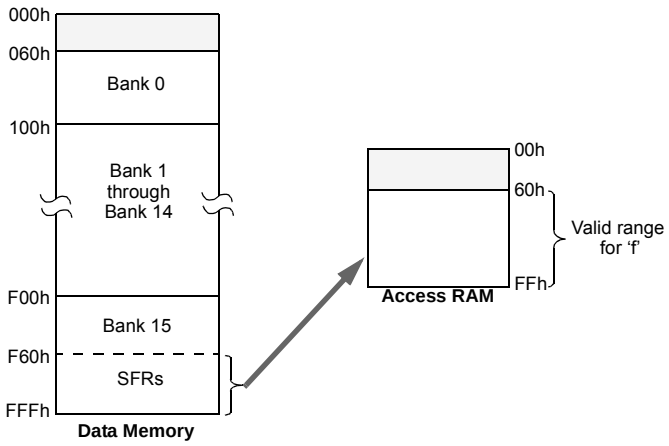
**FIGURE 3-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

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

**When 'a' = 0 and f ≥ 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 FFFh (Bank 15) of data memory.

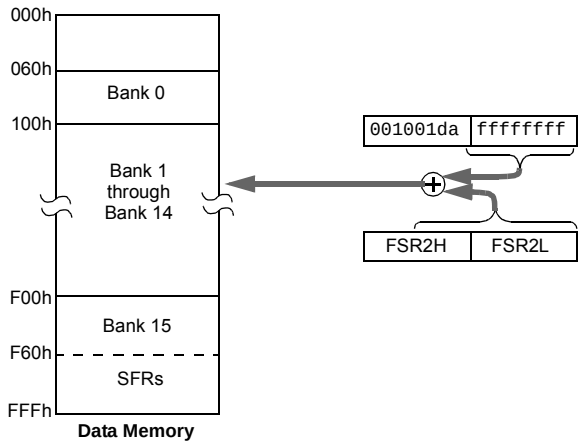
Locations below 060h are not available in this addressing mode.




**When 'a' = 0 and f ≤ 5Fh:**

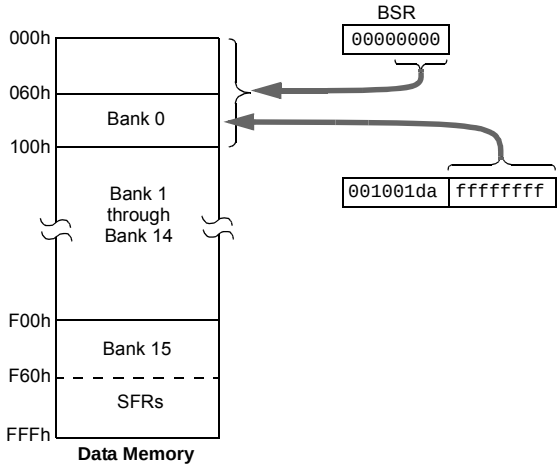
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

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




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

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



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

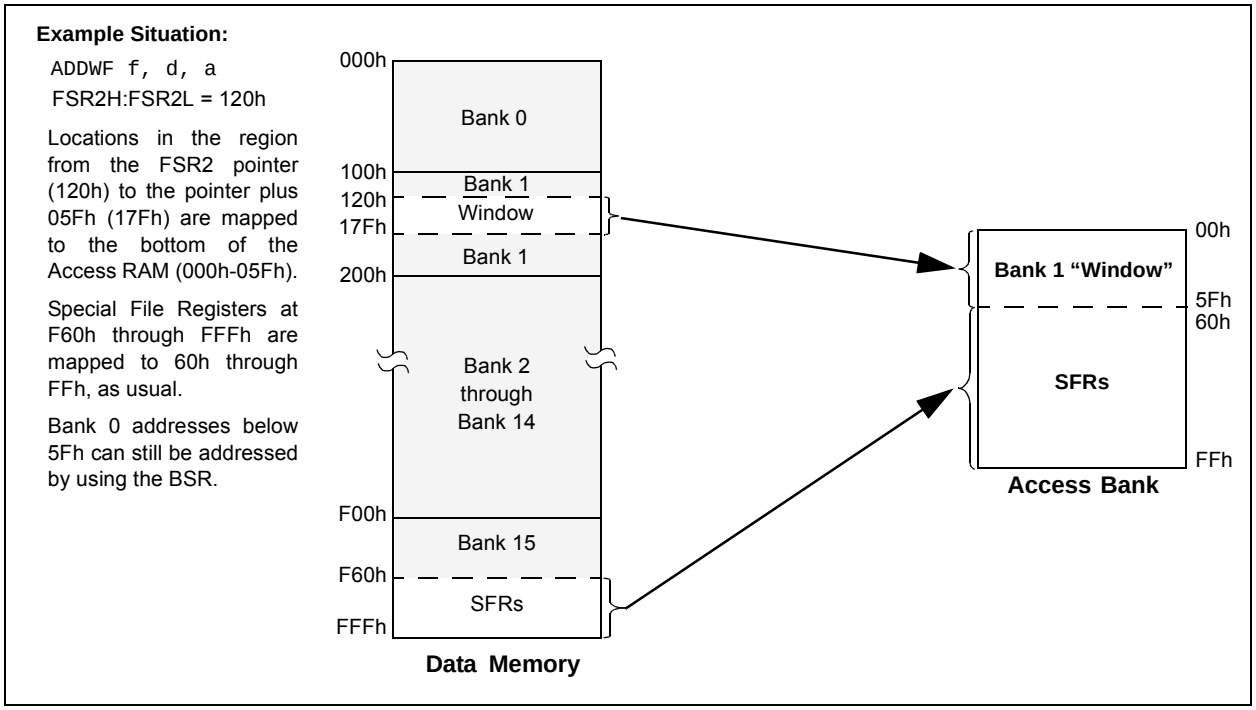
The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the bottom section of Bank 0, this mode maps the contents from a user defined “window” that can be located anywhere in the 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 3.3.2 “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 24.2 “Extended Instruction Set”**.

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





© 2009-2016 Microchip Technology Inc.

## 4.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed one byte at a time. A write to program memory is executed on blocks of 16 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 4 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) |
|---|---|---|
| PIC18(L)F13K22 | 8 | 64 |
| PIC18(L)F14K22 | 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)

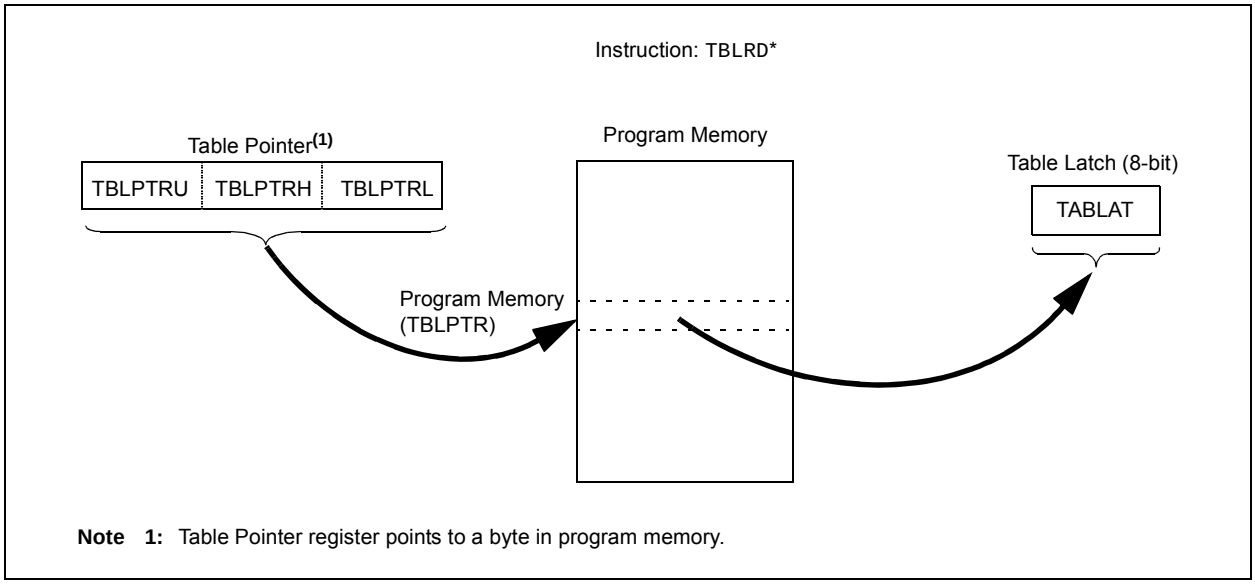
The program memory space is 16-bit wide, while the data RAM space is 8-bit wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

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.

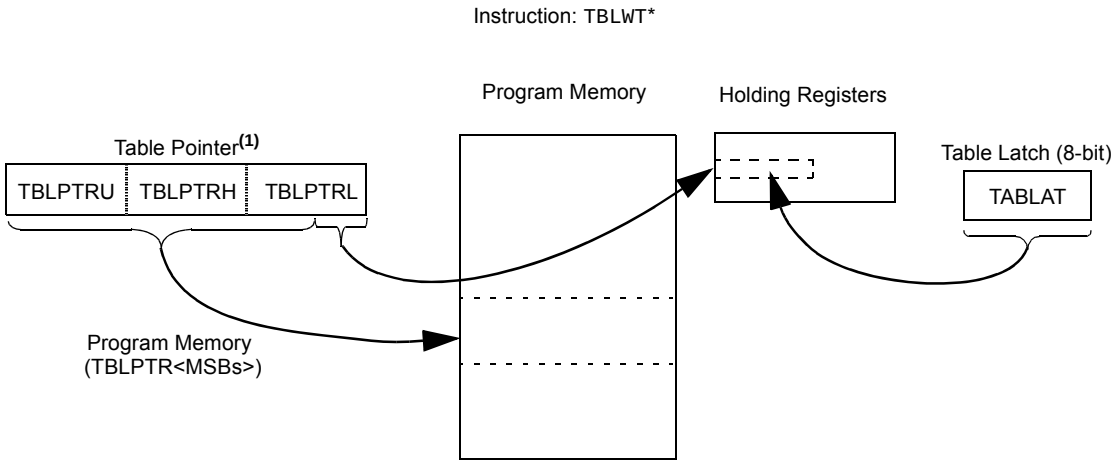
Table operations work with byte entities. Tables containing data, rather than program instructions, are not required to be word-aligned. Therefore, a table can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need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

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




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

**FIGURE 4-2: TABLE WRITE OPERATION**




**Note 1:** During table writes the Table Pointer does not point directly to Program Memory. The LSBs of TBLPRTL actually point to an address within the write block holding registers. The MSBs of the Table Pointer determine where the write block will eventually be written. The process for writing the holding registers to the program memory array is discussed in **Section 4.5 "Writing to Flash Program Memory"**.

## 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 23.0 "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. |
|---|---|


© 2009-2016 Microchip Technology Inc.

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

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
- 1 = Access Configuration registers
- 0 = Access Flash program or data EEPROM memory

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

bit 4 **FREE:** Flash Row (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 **WRERR:** Flash Program/Data EEPROM Error Flag bit(1)
- 1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation, or an improper write attempt)
- 0 = The write operation completed

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

bit 1 **WR:** Write Control bit
- 1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) by software.)
- 0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
- 1 = Initiates an EEPROM read (Read takes one cycle. RD is cleared 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.

### 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”**.

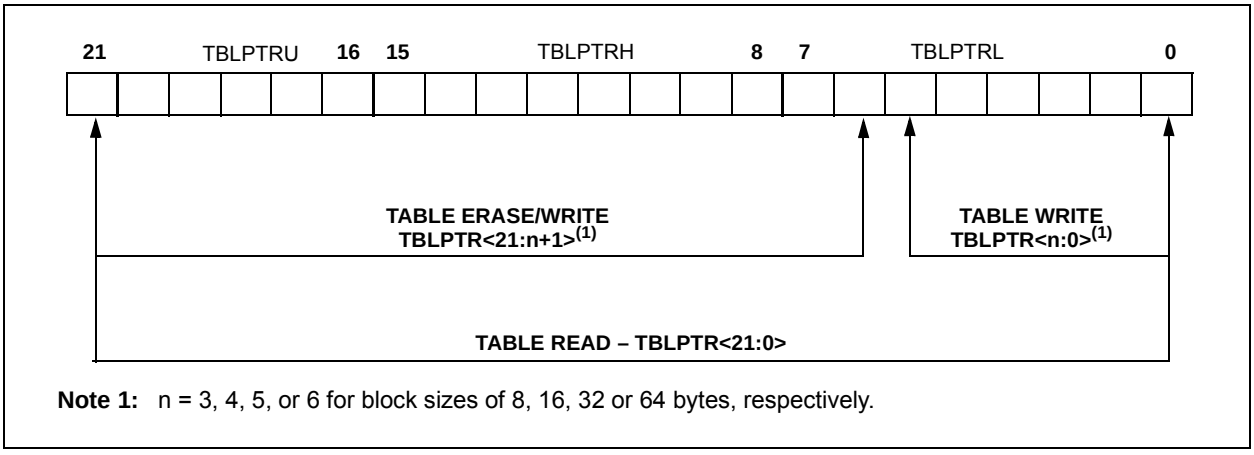
When an erase of program memory is executed, the 16 MSbs of the Table Pointer register (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

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*<br>TBLWT* | TBLPTR is not modified |
| TBLRD*+<br>TBLWT*+ | TBLPTR is incremented after the read/write |
| TBLRD*-<br>TBLWT*- | TBLPTR is decremented after the read/write |
| TBLRD+*<br>TBLWT+* | TBLPTR is incremented before the read/write |

**FIGURE 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.


© 2009-2016 Microchip Technology Inc.

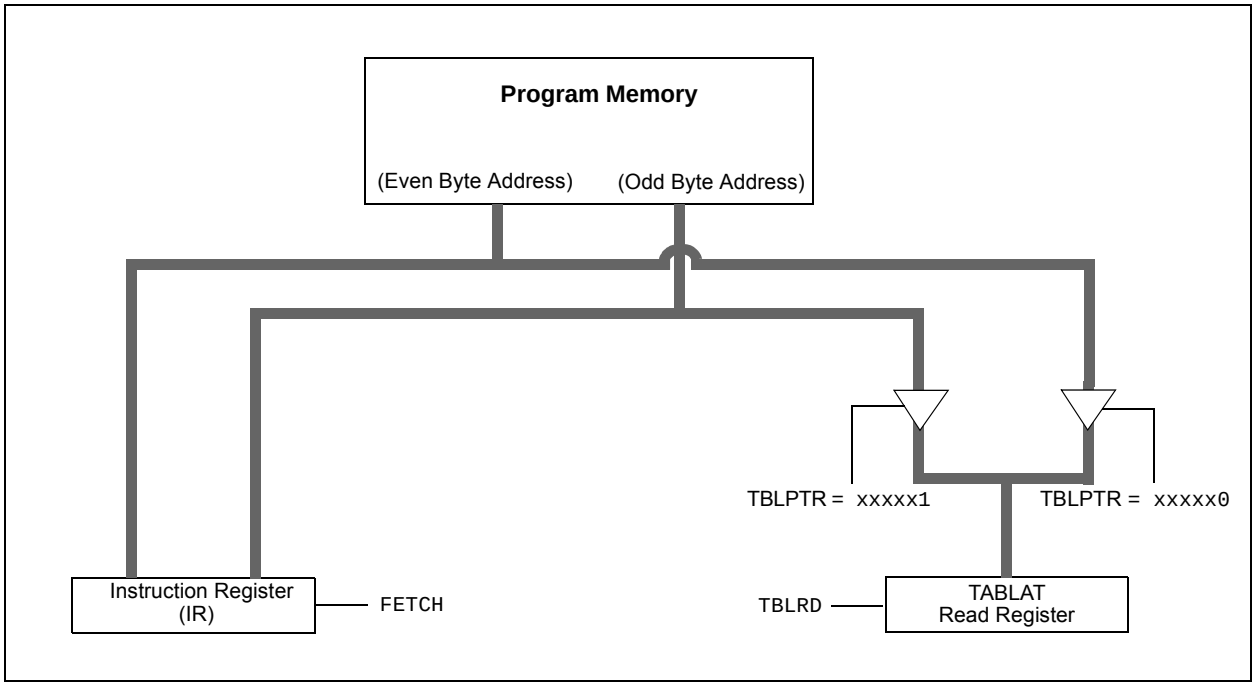
## 4.3 Reading the Flash Program Memory

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*+                           ; read into TABLAT and increment
            MOVF      TABLAT, W               ; get data
            MOVWF     WORD_EVEN
            TBLRD*+                           ; read into TABLAT and increment
            MOVFW     TABLAT, W               ; get data
            MOVF      WORD_ODD
```

## 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 ICSP control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

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

### 4.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

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

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

**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       MOVLW   55h
     Sequence       MOVWF   EECON2                  ; write 55h
                    MOVLW   0AAh
                    MOVWF   EECON2                  ; write 0AAh
                    BSF     EECON1, WR              ; start erase (CPU stall)
                    BSF     INTCON, GIE             ; re-enable interrupts
```


© 2009-2016 Microchip Technology Inc.

## 4.5 Writing to Flash Program Memory

The programming block size is 8 or 16 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).

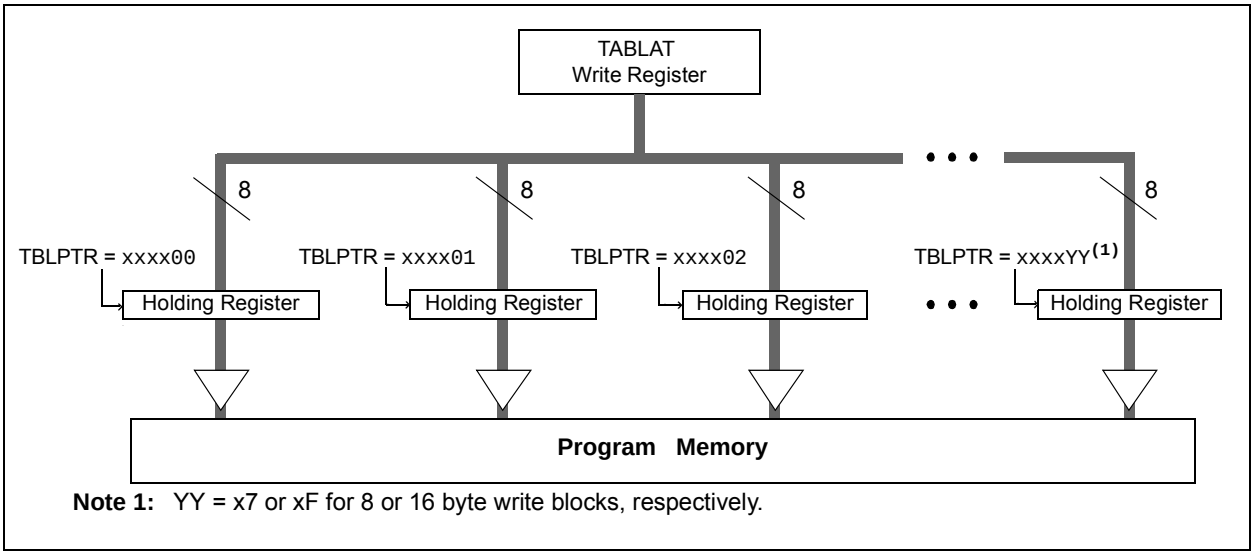
Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction may need to be executed 8, or 16 times, depending on the device, for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. After all the 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**




**Note 1:** YY = x7 or xF for 8 or 16 byte write blocks, respectively.

### 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 16 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. |

**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     FSR0H
                    MOVLW     BUFFER_ADDR_LOW
                    MOVWF     FSR0L
                    MOVLW     CODE_ADDR_UPPER         ; Load TBLPTR with the base
                    MOVWF     TBLPTRU                 ; address of the memory block
                    MOVLW     CODE_ADDR_HIGH
                    MOVWF     TBLPTRH
                    MOVLW     CODE_ADDR_LOW
                    MOVWF     TBLPTRL
READ_BLOCK
                    TBLRD*+                           ; read into TABLAT, and inc
                    MOVF      TABLAT, W               ; get data
                    MOVWF     POSTINC0                ; store data
                    DECFSZ    COUNTER                 ; done?
                    BRA       READ_BLOCK              ; repeat
MODIFY_WORD
                    MOVLW     BUFFER_ADDR_HIGH        ; point to buffer
                    MOVWF     FSR0H
                    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                 ; address of the memory block
                    MOVLW     CODE_ADDR_HIGH
                    MOVWF     TBLPTRH
                    MOVLW     CODE_ADDR_LOW
                    MOVWF     TBLPTRL
                    BSF       EECON1, EEPGD           ; point to Flash program memory
                    BCF       EECON1, CFGS            ; access Flash program memory
                    BSF       EECON1, WREN            ; enable write to memory
                    BSF       EECON1, FREE            ; enable Erase operation
                    BCF       INTCON, GIE             ; disable interrupts
                    MOVLW     55h
Required            MOVWF     EECON2                  ; write 55h
Sequence            MOVLW     0AAh
                    MOVWF     EECON2                  ; write 0AAh
                    BSF       EECON1, WR              ; start erase (CPU stall)
                    BSF       INTCON, GIE             ; re-enable interrupts
                    TBLRD*-                           ; dummy read decrement
                    MOVLW     BUFFER_ADDR_HIGH        ; point to buffer
                    MOVWF     FSR0H
                    MOVLW     BUFFER_ADDR_LOW
                    MOVWF     FSR0L
WRITE_BUFFER_BACK
                    MOVLW     BlockSize               ; number of bytes in holding register
                    MOVWF     COUNTER
                    MOVLW     D'64'/BlockSize         ; number of write blocks in 64 bytes
                    MOVWF     COUNTER2
WRITE_BYTE_TO_HREGS
                    MOVF      POSTINC0, W             ; get low byte of buffer data
                    MOVWF     TABLAT                  ; present data to table latch
                    TBLWT+*                           ; write data, perform a short write
                                                      ; to internal TBLWT holding register.
```


© 2009-2016 Microchip Technology Inc.

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

```
                DECFSZ  COUNTER                 ; loop until holding registers are full
                BRA     WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
                BSF     EECON1, EEPGD           ; point to Flash program memory
                BCF     EECON1, CFGS            ; access Flash program memory
                BSF     EECON1, WREN            ; enable write to memory
                BCF     INTCON, GIE             ; disable interrupts
                MOVLW   55h
    Required    MOVWF   EECON2                  ; write 55h
    Sequence    MOVLW   0AAh
                MOVWF   EECON2                  ; write 0AAh
                BSF     EECON1, WR              ; start program (CPU stall)
                DCFSZ   COUNTER2                ; repeat for remaining write blocks
                BRA     WRITE_BYTE_TO_HREGS     ;
                BSF     INTCON, GIE             ; re-enable interrupts
                BCF     EECON1, WREN            ; 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 $\overline{\text{MCLR}}$ 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 23.0 "Special Features of the CPU"** for more detail.

## 4.6 Flash Program Operation During Code Protection

See **Section 23.3 "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 |
|---|---|---|---|---|---|---|---|---|---|
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | 247 |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | — | TMR3IP | — | 248 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | — | TMR3IE | — | 248 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | — | TMR3IF | — | 248 |
| TABLAT | Program Memory Table Latch | | | | | | | | 245 |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 245 |
| TBLPTRU | — | — | bit 21 | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | 245 |
| TBPLTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 245 |

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

## 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

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

The EEPROM data memory is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer; it will vary with voltage and temperature as well as from chip-to-chip. Please refer to parameter US122 (Table 26-24) for exact limits.

### 5.1 EEADR Register

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

### 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.


© 2009-2016 Microchip Technology Inc.

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

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
- 1 = Access Configuration registers
- 0 = Access Flash program or data EEPROM memory

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

bit 4 **FREE:** Flash Row (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 **WRERR:** Flash Program/Data EEPROM Error Flag bit(1)
- 1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation, or an improper write attempt)
- 0 = The write operation completed

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

bit 1 **WR:** Write Control bit
- 1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) by software.)
- 0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
- 1 = Initiates an EEPROM read (Read takes one cycle. RD is cleared 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.

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.

**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_DATA        ;
                MOVWF   EEDATA              ; Data Memory Value to write
                BCF     EECON1, EEPGD       ; Point to DATA memory
                BCF     EECON1, CFGS        ; Access EEPROM
                BSF     EECON1, WREN        ; Enable writes
                BCF     INTCON, GIE         ; Disable Interrupts
                MOVLW   55h                 ;
    Required    MOVWF   EECON2              ; Write 55h
    Sequence    MOVLW   0AAh                ;
                MOVWF   EECON2              ; Write 0AAh
                BSF     EECON1, WR          ; Set WR bit to begin write
                BSF     INTCON, GIE         ; Enable Interrupts

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


© 2009-2016 Microchip Technology Inc.

## 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 23.0 "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.

A simple data EEPROM refresh routine is shown in Example 5-3.

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

**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                                ; Loop to refresh array
        BSF     EECON1, RD          ; Read current address
        MOVLW   55h                 ;
        MOVWF   EECON2              ; Write 55h
        MOVLW   0AAh                ;
        MOVWF   EECON2              ; Write 0AAh
        BSF     EECON1, WR          ; Set WR bit to begin write
        BTFSC   EECON1, WR          ; Wait for write to complete
        BRA     $-2
        INCFSZ  EEADR, F            ; Increment address
        BRA     LOOP                ; Not zero, do it again

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

**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 |
|---|---|---|---|---|---|---|---|---|---|
| EEADR | EEADR7 | EEADR6 | EEADR5 | EEADR4 | EEADR3 | EEADR2 | EEADR1 | EEADR0 | 247 |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | 247 |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | 247 |
| EEDATA | EEPROM Data Register | | | | | | | | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | — | TMR3IP | — | 248 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | — | TMR3IE | — | 248 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | — | TMR3IF | — | 248 |

**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.

**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
```

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


© 2009-2016 Microchip Technology Inc.

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**

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

**EXAMPLE 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**

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

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

## 7.0 INTERRUPTS

The PIC18(L)F1XK22 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 twelve 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.

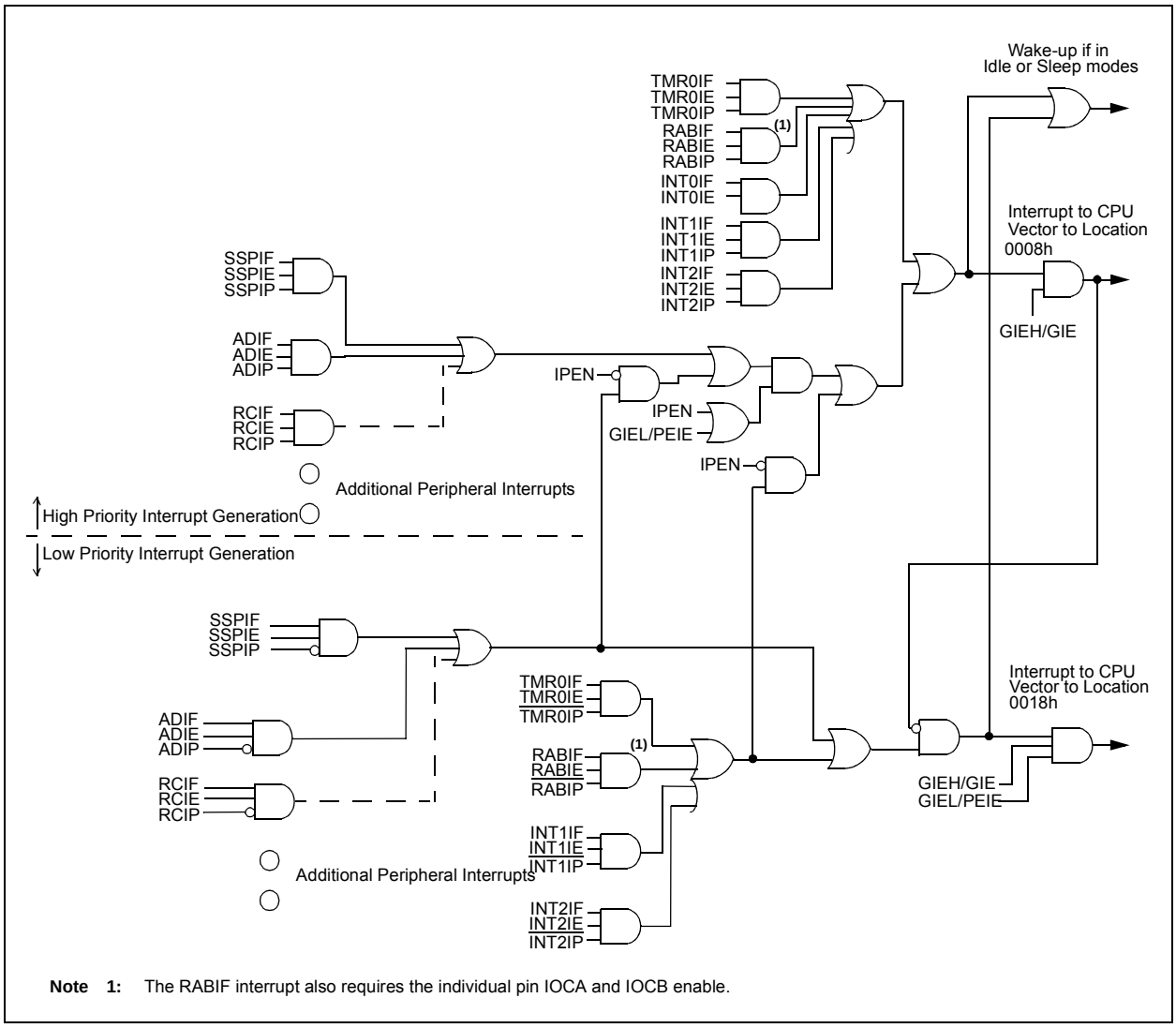
The "return-from-interrupt" instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

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.


© 2009-2016 Microchip Technology Inc.

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

**FIGURE 7-1: PIC18 INTERRUPT LOGIC**




**Note 1:** The RABIF interrupt also requires the individual pin IOCA and IOCB enable.

## 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 | TMR0IE | 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 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts including peripherals
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts including low priority

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority interrupts
0 = Disables all low priority interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RABIE:** RA and RB Port Change Interrupt Enable bit(2)
1 = Enables the RA and RB port change interrupt
0 = Disables the RA and RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared by software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** 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 **RABIF:** RA and RB Port Change Interrupt Flag bit(1)
1 = At least one of the RA <5:0> or RB<7:4> pins changed state (must be cleared by software)
0 = None of the RA<5:0> 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.


© 2009-2016 Microchip Technology Inc.

**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 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{RABPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | 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 **$\overline{\text{RABPU}}$:** PORTA and PORTB Pull-up Enable bit
- 1 = 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 **INTEDG0:** External Interrupt 0 Edge Select bit
- 1 = Interrupt on rising edge
- 0 = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit
- 1 = Interrupt on rising edge
- 0 = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit
- 1 = Interrupt on rising edge
- 0 = Interrupt on falling edge

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

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
- 1 = High priority
- 0 = Low priority

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

bit 0 **RABIP:** 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 might ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

**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 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

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

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

bit 2 **Unimplemented:** Read as '0'

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared by software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared 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 might ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.


© 2009-2016 Microchip Technology Inc.

## 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 might 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'.

**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 | U-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|
| OSCFIF | C1IF | C2IF | EEIF | BCLIF | — | 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 **OSCFIF:** 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 **C1IF:** 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 **C2IF:** 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 **EEIF:** 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 **BCLIF:** Bus Collision Interrupt Flag bit
1 = A bus collision occurred (must be cleared by software)
0 = No bus collision occurred

bit 2 **Unimplemented:** Read as '0'

bit 1 **TMR3IF:** TMR3 Overflow Interrupt Flag bit
1 = TMR3 register overflowed (must be cleared by software)
0 = TMR3 register did not overflow

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


© 2009-2016 Microchip Technology Inc.

## 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 | | | | | | | 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 **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 | U-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|
| OSCFIE | C1IE | C2IE | EEIE | BCLIE | — | 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 **OSCFIE:** Oscillator Fail Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **C1IE:** Comparator C1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **C2IE:** Comparator C2 Interrupt Enable bit
1 = Enabled
0 = Disabled

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

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

bit 2 **Unimplemented:** Read as '0'

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

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


© 2009-2016 Microchip Technology Inc.

## 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 | | | | | | | 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 **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

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

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | U-0 | R/W-1 | U-0 |
|---|---|---|---|---|---|---|---|
| OSCFIP | C1IP | C2IP | EEIP | BCLIP | — | 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 **OSCFIP:** Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **C1IP:** Comparator C1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **C2IP:** Comparator C2 Interrupt Priority bit
1 = High priority
0 = Low priority

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

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

bit 2 **Unimplemented:** Read as '0'

bit 1 **TMR3IP:** TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

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


© 2009-2016 Microchip Technology Inc.

## 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 22.1 "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(1) | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$(2) | $\overline{BOR}$(3) |
| bit 7 | | | | | | | bit 0 |

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

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

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

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

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
1 = The RESET instruction was not executed (set by firmware 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 **$\overline{TO}$:** Watchdog Time-out Flag bit
1 = Set by power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT Time-out occurred

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
1 = Set by power-up or by the CLRWDT instruction
0 = Set by execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit(2)
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit(3)
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 $\overline{POR}$ is determined by the type of device Reset. See the notes following this register and **Section 22.6 "Reset State of Registers"** for additional information.
**3:** See Table 22-3.

## 7.9 INTx Pin Interrupts

External interrupts on the INT0, INT1 and INT2 pins are edge-triggered. If the corresponding INTEDGx bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. When a valid edge appears on the INTx pin, the corresponding flag bit, INTxF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxE. Flag bit, INTxF, must be cleared 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 9.0 "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_TMEP located anywhere
;
; USER ISR CODE
;
MOVFF     BSR_TEMP, BSR                   ; Restore BSR
MOVF      W_TEMP, W                       ; Restore WREG
MOVFF     STATUS_TEMP, STATUS             ; Restore STATUS
```


© 2009-2016 Microchip Technology Inc.

## 8.0 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.

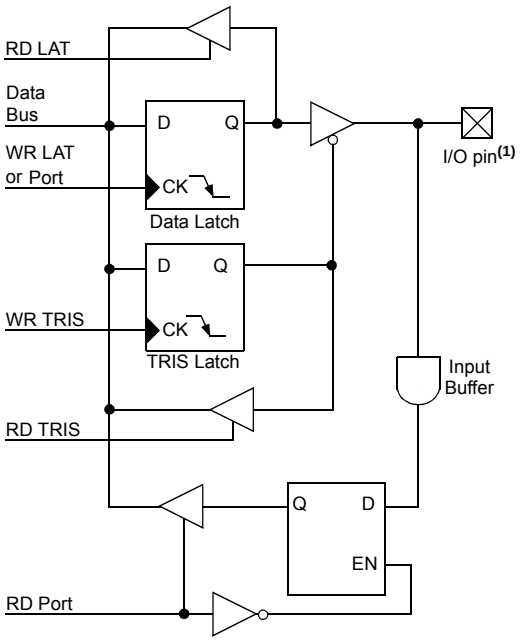
Each port has three registers for its operation. These registers are:

- 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 8-1.

**FIGURE 8-1: GENERIC I/O PORT OPERATION**




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

## 8.1 PORTA, TRISA and LATA Registers

PORTA is a 6-bit wide, bidirectional port, with the exception of RA3, which is input-only and its TRIS bit will always read as '1'. 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.

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.

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 23.1 "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'.

RA<4,2:0> are pins multiplexed with analog inputs. The operation of pins RA<4,2:0> as analog are selected by setting the ANS<3:0> bits in the ANSEL register, which is the default setting after a Power-on Reset.

**EXAMPLE 8-1: 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
```


© 2009-2016 Microchip Technology Inc.

**REGISTER 8-1: PORTA: PORTA REGISTER**

| U-0 | U-0 | R/W-x | R/W-x | R-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | RA5 | RA4 | RA3 | RA2 | 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 **Unimplemented**: Read as '0'

bit 5-0 **RA<5:0>**: PORTA I/O Pin bit(1)
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 8-2: TRISA: PORTA TRI-STATE REGISTER**

| U-0 | U-0 | R/W-1 | R/W-1 | U-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| — | — | TRISA5 | TRISA4 | — | TRISA2 | TRISA1 | TRISA0 |
| bit 7 | | | | | | | bit 0 |

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

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

bit 5-4 **TRISA<5:4>:** PORTA Tri-State Control bit(1)
1 = PORTA pin configured as an input (tri-stated)
0 = PORTA pin configured as an output

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

bit 2-0 **TRISA<2:0>:** PORTA Tri-State Control bit(1)
1 = PORTA pin configured as an input (tri-stated)
0 = PORTA pin configured as an output

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

**REGISTER 8-3: LATA: PORTA DATA LATCH REGISTER**

| U-0 | U-0 | R/W-x | R/W-x | U-0 | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | LATA5 | LATA4 | — | LATA2 | LATA1 | LATA0 |
| bit 7 | | | | | | | bit 0 |

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

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

bit 5-4 **LATA<5:4>**: RA<5:4> Port I/O Output Latch Register bits

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

bit 2-0 **LATA<2:0>**: RA<2:0> Port I/O Output Latch Register bits

**REGISTER 8-4: WPUA: WEAK PULL-UP PORTA REGISTER**

| U-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| — | — | WPUA5 | WPUA4 | WPUA3(1) | WPUA2 | WPUA1 | WPUA0 |
| 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 5-0 **WPUA<5:0>:** Weak Pull-up Enable bit
1 = Pull-up enabled
0 = Pull-up disabled

**Note 1:** For the WPUA3 bit, when MCLRE = 1, weak pull-up is internally enabled, but not reported here.

**REGISTER 8-5: IOCA: INTERRUPT-ON-CHANGE PORTA REGISTER**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | IOCA5 | IOCA4 | IOCA3 | IOCA2 | IOCA1 | IOCA0 |
| 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 5-0 **IOCA<5:0>**: PORTA I/O Pin bit
1 = Interrupt-on-change enabled
0 = Interrupt-on-change disabled


© 2009-2016 Microchip Technology Inc.

**TABLE 8-1: PORTA I/O SUMMARY**

| Pin | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RA0/AN0/CVREF/ DAC1OUT/VREF-/ C1IN+/INT0/PGD | RA0 | 0 | O | DIG | LATA<0> data output. |
| | | 1 | I | TTL | PORTA<0> data input; Programmable weak pull-up. |
| | AN0 | 1 | I | ANA | ADC channel 0 input. |
| | CVREF/ DAC1OUT | x | O | ANA | DAC reference voltage output. |
| | VREF- | 1 | I | ANA | ADC and DAC reference voltage (low) input. |
| | C1IN+ | 1 | I | DIG | Comparator C1 noninverting input. |
| | INT0 | 1 | I | ST | External interrupt 0. |
| | PGD | x | O | DIG | Serial execution data output for ICSP™. |
| | | x | I | ST | Serial execution data input for ICSP. |
| RA1/AN1/C12IN0-/ VREF+/INT1/PGC | RA1 | 0 | O | DIG | LATA<1> data output. |
| | | 1 | I | TTL | PORTA<1> data input; Programmable weak pull-up. |
| | AN1 | 1 | I | ANA | ADC channel 1. |
| | C12IN0- | 1 | I | ANA | Comparator C1 and C2 inverting input channel 0. |
| | VREF+ | 1 | I | ANA | ADC and DAC reference voltage (high) input |
| | INT1 | 1 | | ST | External interrupt 1. |
| | PGC | x | O | DIG | Serial execution clock output for ICSP™. |
| | | x | I | ST | Serial execution clock input for ICSP. |
| RA2/AN2/C1OUT/ T0CKI/INT2/SRQ | RA2 | 0 | O | DIG | LATA<2> data output. |
| | | 1 | I | TTL | PORTA<2> data input; Programmable weak pull-up. |
| | AN2 | 1 | I | ANA | ADC channel 2. |
| | C1OUT | 0 | O | DIG | Comparator C1 output. |
| | T0CKI | 1 | I | ST | Timer0 external clock input. |
| | INT2 | 1 | I | ST | External interrupt 2. |
| | SRQ | 0 | O | DIG | SR latch output. |
| RA3/$\overline{\text{MCLR}}$/VPP | RA3 | —(1) | I | ST | PORTA<37> data input; Programmable weak pull-up. |
| | $\overline{\text{MCLR}}$ | — | I | ST | Active-low Master Clear with internal pull-up. |
| | VPP | — | I | ANA | High-voltage programming input. |
| RA4/AN3/OSC2/ CLKOUT | RA4 | 0 | O | DIG | LATA<4> data output. |
| | | 1 | I | TTL | PORTA<4> data input; Programmable weak pull-up. |
| | AN3 | 1 | I | ANA | A/D input channel 3. |
| | OSC2 | x | O | ANA | Main oscillator feedback output connection (XT, HS and LP modes). |
| | CLKOUT | x | O | DIG | System instruction cycle clock output. |
| RA5/OSC1/CLKIN/ T13CKI | RA5 | 0 | O | DIG | LATA<5> data output. |
| | | 1 | I | TTL | PORTA<5> data input; Programmable weak pull-up. |
| | OSC1 | x | I | ANA | Main oscillator input connection. |
| | CLKIN | x | I | ANA | Main clock input connection. |
| | T13CKI | 1 | I | ST | Timer1 and Timer3 external clock input. |

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

**TABLE 8-2: 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 |
|---|---|---|---|---|---|---|---|---|---|
| ANSEL | ANS7 | ANS6 | ANS5 | ANS4 | ANS3 | ANS2 | ANS1 | ANS0 | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 244 |
| INTCON2 | $\overline{\text{RABPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RABIP | 244 |
| IOCA | — | — | IOCA5 | IOCA4 | IOCA3(2) | IOCA2 | IOCA1 | IOCA0 | 247 |
| LATA | — | — | LATA5(1) | LATA4(1) | — | LATA2 | LATA1 | LATA0 | 247 |
| PORTA | — | — | RA5(1) | RA4(1) | RA3(2) | RA2 | RA1 | RA0 | 247 |
| SLRCON | — | — | — | — | — | SLRC | SLRB | SLRA | 247 |
| TRISA | — | — | TRISA5(1) | TRISA4(1) | —(3) | TRISA2 | TRISA1 | TRISA0 | 247 |
| WPUA | — | — | WPUA5 | WPUA4 | WPUA3(2) | WPUA2 | WPUA1 | WPUA0 | 244 |

**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:** Unimplemented, read as '1'.


© 2009-2016 Microchip Technology Inc.

## 8.2 PORTB, TRISB and LATB Registers

PORTB is an 4-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 8-2: 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:

a) Any read or write of PORTB to clear the mismatch condition (except when PORTB 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 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'. |
|---|---|

**REGISTER 8-6: 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 | 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 **RB<7:4>**: PORTB I/O Pin bit
1 = Port pin is >VIH
0 = Port pin is <VIL

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

**REGISTER 8-7: 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 | 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 **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'

**REGISTER 8-8: 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'


© 2009-2016 Microchip Technology Inc.

**REGISTER 8-9: 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 8-10: 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’

**TABLE 8-3: PORTB I/O SUMMARY**

| Pin | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RB4/AN10/SDI/SDA | RB4 | 0 | O | DIG | LATB<4> data output. |
| | | 1 | I | TTL | PORTB<4> data input; Programmable weak pull-up. |
| | AN10 | 1 | I | ANA | ADC input channel 10. |
| | SDI | 1 | I | ST | SPI data input (MSSP module). |
| | SDA | 1 | O | DIG | $I^2C$ data output (MSSP module). |
| | | 1 | I | $I^2C$ | $I^2C$ data input (MSSP module); input type depends on module setting. |
| RB5/AN11/RX/DT | RB5 | 0 | O | DIG | LATB<5> data output. |
| | | 1 | I | TTL | PORTB<5> data input; Programmable weak pull-up. |
| | 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/SCK/SCL | RB6 | 0 | O | DIG | LATB<6> data output. |
| | | 1 | I | TTL | PORTB<6> data input; Programmable weak pull-up. |
| | SCK | 0 | O | DIG | SPI clock output (MSSP module). |
| | | 1 | I | ST | SPI clock input (MSSP module). |
| | SCL | 0 | O | DIG | $I^2C$ clock output (MSSP module). |
| | | 1 | I | $I^2C$ | $I^2C$ clock input (MSSP module); input type depends on module setting. |
| RB7/TX/CK | RB7 | 0 | O | DIG | LATB<7> data output. |
| | | 1 | I | TTL | PORTB<7> data input; Programmable weak pull-up. |
| | TX | 1 | O | DIG | Asynchronous serial transmit data output (USART module). |
| | CK | 1 | O | DIG | Synchronous serial clock output (USART module). |
| | | 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; $I^2C$ = Schmitt Trigger input with $I^2C$; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).


© 2009-2016 Microchip Technology Inc.

**TABLE 8-4: 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 |
|---|---|---|---|---|---|---|---|---|---|
| ANSELH | — | — | — | — | ANS11 | ANS10 | ANS9 | ANS8 | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 244 |
| INTCON2 | $\overline{\text{RABPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RABIP | 244 |
| IOCB | IOCB7 | IOCB6 | IOCB5 | IOCB4 | | | | | 247 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | — | — | — | — | 247 |
| PORTB | RB7 | RB6 | RB5 | RB4 | — | — | — | — | 247 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 246 |
| SLRCON | — | — | — | — | — | SLRC | SLRB | SLRA | 247 |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 245 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | — | — | — | — | 247 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 246 |
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | — | — | — | — | 247 |

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

## 8.3 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 8-3: 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 8-11: PORTC: PORTC REGISTER**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 |
| 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 **RC<7:0>**: PORTC I/O Pin bits
1 = Port pin is > $V_{IH}$
0 = Port pin is < $V_{IL}$

**REGISTER 8-12: TRISC: PORTC TRI-STATE 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 |
|---|---|---|---|---|---|---|---|
| TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| 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 **TRISC<7:0>:** PORTC Tri-State Control bits
1 = PORTC pin configured as an input (tri-stated)
0 = PORTC pin configured as an output


© 2009-2016 Microchip Technology Inc.

**REGISTER 8-13: LATC: PORTC DATA LATCH REGISTER**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | 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

**TABLE 8-5: PORTC I/O SUMMARY**

| Pin | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RC0/AN4/C2IN+ | RC0 | 0 | O | DIG | LATC<0> data output. |
| | | 1 | I | ST | PORTC<0> data input. |
| | AN4 | 1 | I | ANA | A/D input channel 4. |
| | C2IN+ | 1 | I | ANA | Comparators C2 noninverting input. |
| RC1/AN5/ C12IN1- | 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, channel 1. |
| RC2/AN6/ C12IN2-/P1D | 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. |
| | P1D | 0 | O | DIG | ECCP1 Enhanced PWM output, channel D. |
| 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. |
| | P1C | 0 | O | DIG | ECCP1 Enhanced PWM output, channel C. |
| | PGM | x | I | ST | Single-Supply Programming mode entry (ICSP™). Enabled by LVP Configuration bit; all other pin functions disabled. |
| RC4/C2OUT/P1B/ SRNQ | RC4 | 0 | O | DIG | LATC<4> data output. |
| | | 1 | I | ST | PORTC<4> data input. |
| | C2OUT | 0 | O | DIG | Comparator 2 output. |
| | P1B | 0 | O | DIG | ECCP1 Enhanced PWM output, channel B. |
| | SRNQ | 0 | O | DIG | SR Latch inverted output |
| RC5/CCP1/P1A | RC5 | 0 | O | DIG | LATC<5> data output. |
| | | 1 | I | ST | PORTC<5> data input. |
| | CCP1 | 0 | O | DIG | ECCP1 compare or PWM output. |
| | | 1 | I | ST | ECCP1 capture input. |
| | P1A | 0 | 0 | DIG | ECCP1 Enhanced PWM output, channel A. |
| RC6/AN8/$\overline{SS}$ | RC6 | 0 | O | DIG | LATC<6> data output. |
| | | 1 | I | ST | PORTC<6> data input. |
| | AN8 | 1 | I | ANA | A/D input channel 8. |
| | $\overline{SS}$ | 1 | I | TTL | Slave select input for SSP (MSSP module) |
| RC7/AN9/SDO | 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 | O | DIG | SPI data output (MSSP 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).


© 2009-2016 Microchip Technology Inc.

**TABLE 8-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on page |
|---|---|---|---|---|---|---|---|---|---|
| ANSEL | ANS7 | ANS6 | ANS5 | ANS4 | ANS3 | ANS2 | ANS1 | ANS0 | 247 |
| ANSELH | — | — | — | — | ANS11 | ANS10 | ANS9 | ANS8 | 247 |
| CCP1CON | P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 246 |
| ECCP1AS | ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 | 246 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 244 |
| INTCON2 | $\overline{\text{RABPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | RABIP | 244 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 244 |
| LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | 247 |
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | 247 |
| PSTRCON | — | — | — | STRSYNC | STRD | STRC | STRB | STRA | 246 |
| VREFCON1 | D1EN | D1LPS | DAC1OE | --- | D1PSS1 | D1PSS0 | --- | D1NSS | 246 |
| SLRCON | — | — | — | — | — | SLRC | SLRB | SLRA | 247 |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 245 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 247 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 245 |
| T3CON | RD16 | — | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 246 |

## 8.4 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 ANSx 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 ANSx bits has no affect on digital output functions. A pin with the associated TRISx bit clear and ANSx bit set will still operate as a digital output but the Input mode will be analog.

**REGISTER 8-14: ANSEL: ANALOG SELECT 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 |
|---|---|---|---|---|---|---|---|
| ANS7 | ANS6 | ANS5 | ANS4 | ANS3 | ANS2 | ANS1 | ANS0 |
| 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 **ANS2:** RA2 Analog Select Control bit
1 = Digital input buffer of RA2 is disabled
0 = Digital input buffer of RA2 is enabled

bit 1 **ANS1:** RA1 Analog Select Control bit
1 = Digital input buffer of RA1 is disabled
0 = Digital input buffer of RA1 is enabled

bit 0 **ANS0:** RA0 Analog Select Control bit
1 = Digital input buffer of RA0 is disabled
0 = Digital input buffer of RA0 is enabled


© 2009-2016 Microchip Technology Inc.

**REGISTER 8-15: ANSELH: ANALOG SELECT HIGH REGISTER**

| 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 | | | | | | | 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 **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

## 8.5 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 8-16: 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(1)
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.


© 2009-2016 Microchip Technology Inc.

## 9.0 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 9-1) 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 9-1. Figure 9-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

**REGISTER 9-1: T0CON: TIMER0 CONTROL REGISTER**

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

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

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

© 2009-2016 Microchip Technology Inc.

## 9.1 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 9.3 "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.

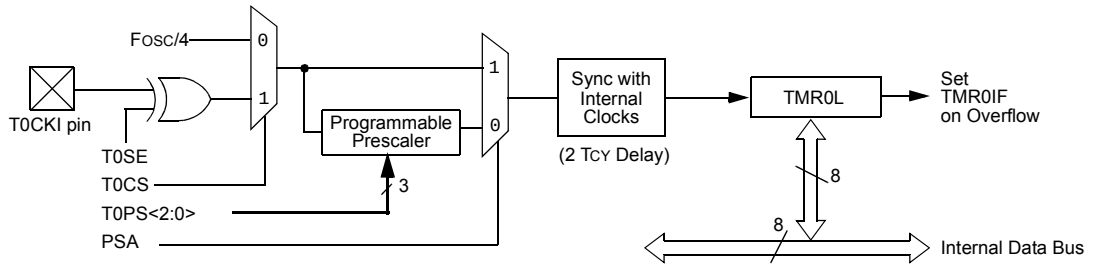
An external clock source can be used to drive Timer0; however, it must meet certain requirements (see Table 26-17) to ensure that the external clock can be synchronized with the internal phase clock (TOSC). There is a delay between synchronization and the onset of incrementing the timer/counter.

## 9.2 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 9-2). 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.

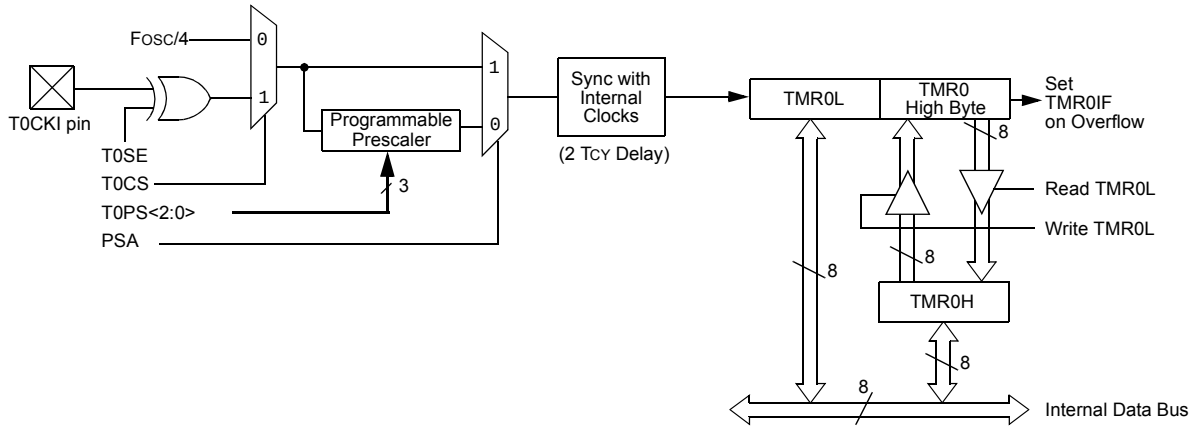
**FIGURE 9-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)**




**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI. TMR0 Prescaler is set to maximum (1:256), but on Reset is not assigned to the timer.


© 2009-2016 Microchip Technology Inc.

**FIGURE 9-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)**




**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI. TMR0 Prescaler is set to maximum (1:256), but on Reset is not assigned to the timer.

## 9.3 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. |
|---|---|

### 9.3.1 SWITCHING PRESCALER ASSIGNMENT

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

## 9.4 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 9-1: 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 |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| PORTA | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 248 |
| TMR0H | Timer0 Register, High Byte | | | | | | | | 246 |
| TMR0L | Timer0 Register, Low Byte | | | | | | | | 246 |
| TRISA | — | — | TRISA5 | TRISA4 | —(1) | TRISA2 | TRISA1 | TRISA0 | 248 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 246 |

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

**Note:** Unimplemented, read as’1’.

## 10.0 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 10-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 10-2.

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 10-1). 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 10-1: T1CON: TIMER1 CONTROL REGISTER**

| R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of TImer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **T1RUN:** Timer1 System Clock Status bit
1 = Main system clock is derived from Timer1 oscillator
0 = Main system clock is derived from another source

bit 5-4 **T1CKPS<1:0>:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from the T13CKI pin (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1


© 2009-2016 Microchip Technology Inc.

## 10.1 Timer1 Operation

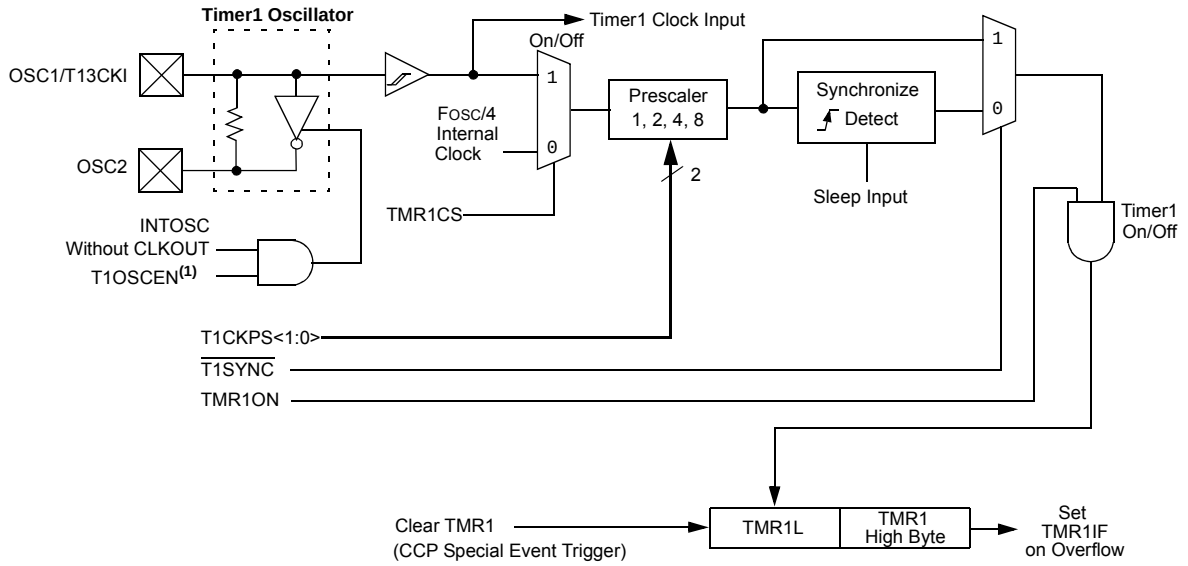
Timer1 can operate in one of the following modes:

- 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.

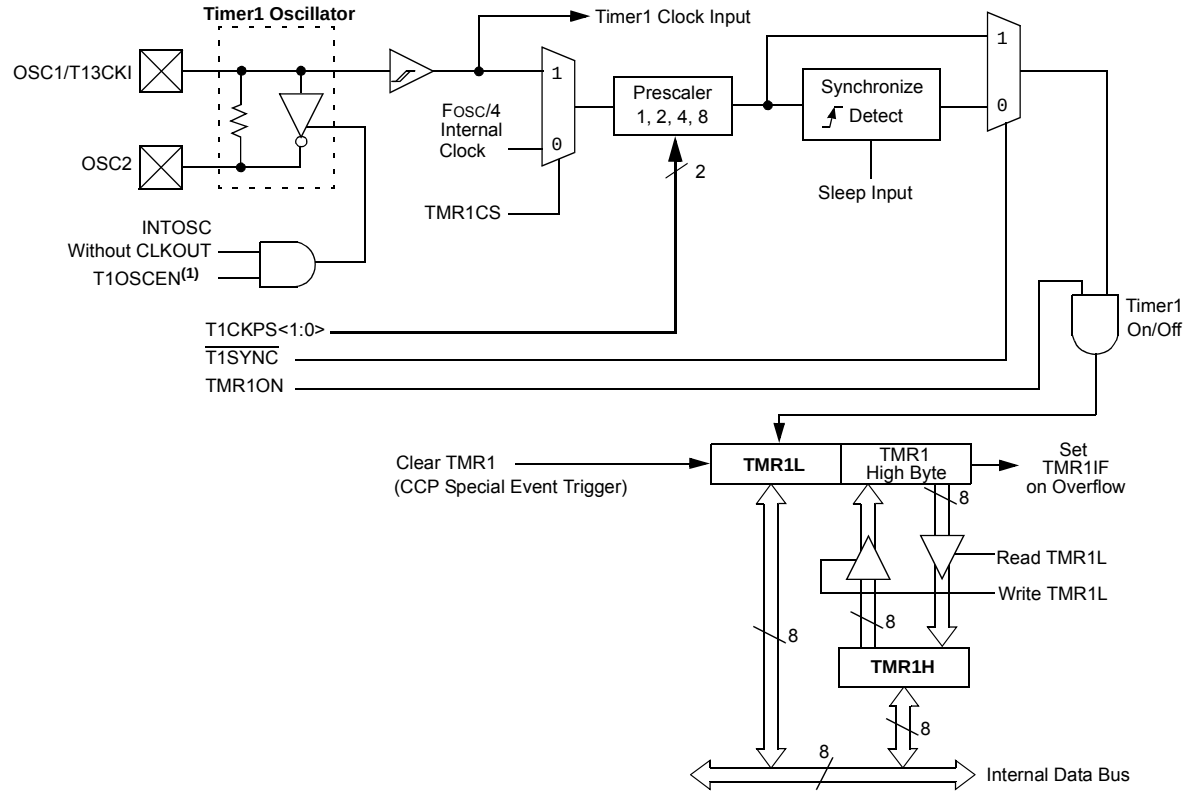
When the Timer1 oscillator is enabled, the digital circuitry associated with the OSC1 and OSC2 pins is disabled. This means the values of TRISA<5:4> are ignored and the pins are read as '0'.

**FIGURE 10-1: TIMER1 BLOCK DIAGRAM**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

**FIGURE 10-2: TIMER1 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.


© 2009-2016 Microchip Technology Inc.

## 10.2 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 10-2). 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.

## 10.3 Clock Source Selection

The TMR1CS bit of the T1CON register is used to select the clock source. When TMR1CS = 0, the clock source is FOSC/4. When TMR1CS = 1, the clock source is supplied externally.

| Clock Source | T1OSCEN | FOSC Mode | TMR1CS |
|---|---|---|---|
| FOSC/4 | x | xxx | 0 |
| T1CKI pin | 0 | xxx | 1 |
| T1LPOSC | 1 | LP or INTOSCIO | 1 |

### 10.3.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected the TMR1H:TMR1L register pair will increment on multiples of FOSC as determined by the Timer1 prescaler.

### 10.3.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1 module may work as a timer or a counter.

When counting, Timer1 is incremented on the rising edge of the external clock input T1CKI. In addition, the Counter mode clock can be synchronized to the microcontroller system clock or run asynchronously.

If an external clock oscillator is needed (and the microcontroller is using the INTOSC without CLKOUT), Timer1 can use the LP oscillator as a clock source.

**Note:** In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:

- Timer1 enabled after POR
- Write to TMR1H or TMR1L
- Timer1 is disabled
- Timer1 is disabled (TMR1ON 0) when T1CKI is high then Timer1 is enabled (TMR1ON= 1) when T1CKI is low.

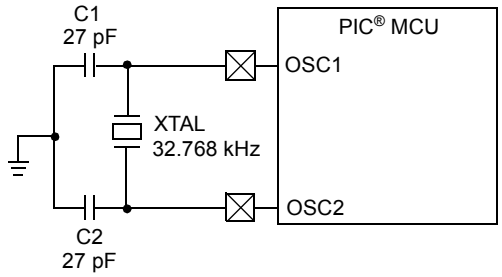
**Note:** See Figure 9-2.

## 10.4 Timer1 Oscillator

An on-chip crystal oscillator circuit is incorporated between pins OSC1 (input) and OSC2 (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 10-3. Table 10-1 shows the capacitor selection for the Timer1 oscillator.

The Timer1 oscillator is shared with the system LP oscillator. Thus, Timer1 can use this mode only when the primary system clock is derived from the internal oscillator or when the oscillator is in the LP mode. The user must provide a software time delay to ensure proper oscillator start-up.

**FIGURE 10-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR**




**Note:** See the Notes with Table 10-1 for additional information about capacitor selection.

**TABLE 10-1: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR**

| Osc Type | Freq. | C1 | C2 |
|---|---|---|---|
| LP | 32 kHz | 27 pF(1) | 27 pF(1) |

**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.

## 10.5 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.

## 10.6 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 13.3.4 "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.

## 10.7 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 10.4 "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 10-1, 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.

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

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


© 2009-2016 Microchip Technology Inc.

**EXAMPLE 10-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE**

```
RTCinit
            MOVLW   80h             ; Preload TMR1 register pair
            MOVWF   TMR1H           ; for 1 second overflow
            CLRF    TMR1L
            MOVLW   b'00001111'     ; Configure for external clock,
            MOVWF   T1CON           ; Asynchronous operation, external oscillator
            CLRF    secs            ; Initialize timekeeping registers
            CLRF    mins            ;
            MOVLW   .12
            MOVWF   hours
            BSF     PIE1, TMR1IE    ; Enable Timer1 interrupt
            RETURN
RTCisr
            BSF     TMR1H, 7        ; Preload for 1 sec overflow
            BCF     PIR1, TMR1IF    ; Clear interrupt flag
            INCF    secs, F         ; Increment seconds
            MOVLW   .59             ; 60 seconds elapsed?
            CPFSGT  secs
            RETURN                  ; No, done
            CLRF    secs            ; Clear seconds
            INCF    mins, F         ; Increment minutes
            MOVLW   .59             ; 60 minutes elapsed?
            CPFSGT  mins
            RETURN                  ; No, done
            CLRF    mins            ; clear minutes
            INCF    hours, F        ; Increment hours
            MOVLW   .23             ; 24 hours elapsed?
            CPFSGT  hours
            RETURN                  ; No, done
            CLRF    hours           ; Reset hours
            RETURN                  ; Done
```

**TABLE 10-2: 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 | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| TMR1H | Timer1 Register, High Byte | | | | | | | | 246 |
| TMR1L | Timer1 Register, Low Byte | | | | | | | | 246 |
| TRISA | — | — | TRISA5 | TRISA4 | —(1) | TRISA2 | TRISA1 | TRISA0 | 248 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 246 |

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

**Note 1:** Unimplemented, read as'1'.

## 11.0 TIMER2 MODULE

The Timer2 module timer incorporates the following features:

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

The module is controlled through the T2CON register (Register 11-1), which enables or disables the timer and configures the prescaler and postscaler. 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 11-1.

### 11.1 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/postscaler (see **Section 11.2 "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 postscaler counters are cleared on the following events:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (Power-on Reset, $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

**REGISTER 11-1: T2CON: TIMER2 CONTROL REGISTER**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| bit 7 | | | | | | | bit 0 |

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

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

bit 6-3 **T2OUTPS<3:0>**: Timer2 Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS<1:0>**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16


© 2009-2016 Microchip Technology Inc.

## 11.2 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 postscaler control bits, T2OUTPS<3:0> of the T2CON register.

## 11.3 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 11-1: TIMER2 BLOCK DIAGRAM**

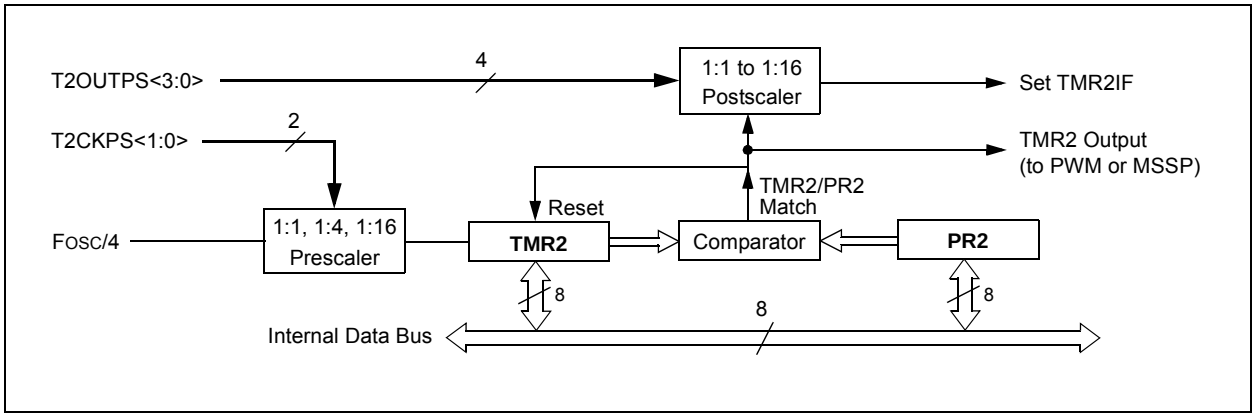



**TABLE 11-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

| 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 | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| PR2 | Timer2 Period Register | | | | | | | | 246 |
| TMR2 | Timer2 Register | | | | | | | | 246 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | 246 |

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

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

The Timer3 module is controlled through the T3CON register (Register 12-1). It also selects the clock source options for the CCP modules (see **Section 13.1.1 "CCP Module and Timer Resources"** for more information).

**REGISTER 12-1: 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 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

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

bit 5-4 **T3CKPS<1:0>**: Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T3CCP1:** 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 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T13CKI (on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3


© 2009-2016 Microchip Technology Inc.

## 12.1 Timer3 Operation

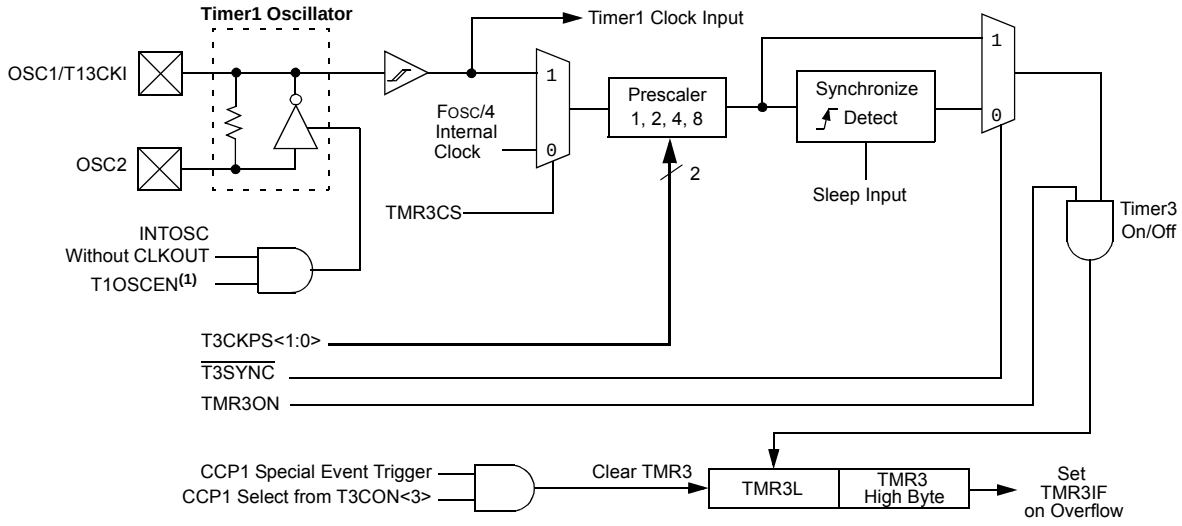
Timer3 can operate in one of three modes:

- 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.

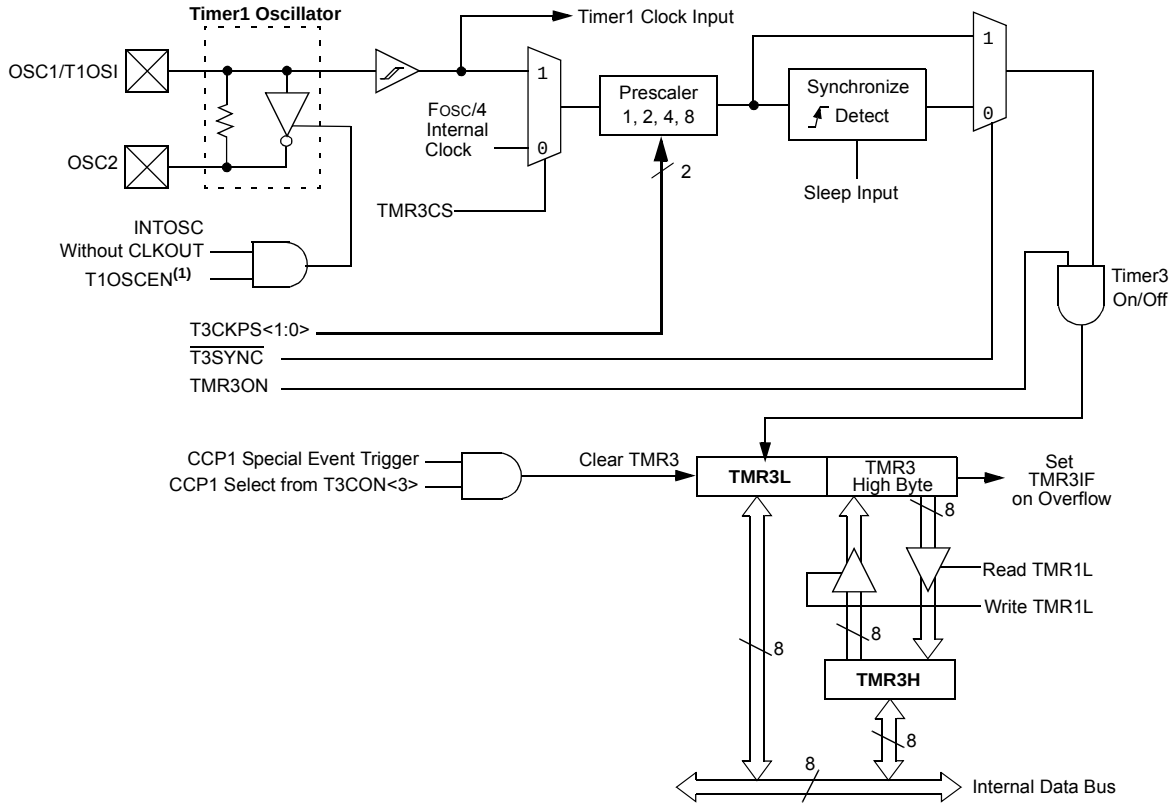
As with Timer1, the digital circuitry associated with the OSC1 and OSC2 pins is disabled when the Timer1 oscillator is enabled. This means the values of TRISA<5:4> are ignored and the pins are read as '0'.

**FIGURE 12-1: TIMER3 BLOCK DIAGRAM**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

**FIGURE 12-2: TIMER3 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.


© 2009-2016 Microchip Technology Inc.

## 12.2 Timer3 16-Bit Read/Write Mode

Timer3 can be configured for 16-bit reads and writes (see Figure 12-2). 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.

## 12.3 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 10.0 "Timer1 Module"**.

## 12.4 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.

## 12.5 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<3:0>), this signal will reset Timer3. It will also start an A/D conversion if the A/D module is enabled (see **Section 16.2.8 "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 CCPR1H:CCPR1L 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 12-1: 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 | 245 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | — | TMR3IP | — | 248 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | — | TMR3IE | — | 248 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | — | TMR3IF | — | 248 |
| TMR3H | Timer3 Register, High Byte | | | | | | | | 247 |
| TMR3L | Timer3 Register, Low Byte | | | | | | | | 247 |
| TRISA | — | — | TRISA5 | TRISA4 | —(1) | TRISA2 | TRISA1 | TRISA0 | 248 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 246 |
| T3CON | RD16 | — | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 247 |

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

**Note 1:** Unimplemented, read as '1'.

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

PIC18(L)F1XK22 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 two or four output channels
- Output steering
- Programmable polarity
- Programmable dead-band control
- Automatic shutdown and restart

The enhanced features are discussed in detail in **Section 13.4 "PWM (Enhanced Mode)"**.

**REGISTER 13-1: 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 **P1M<1:0>:** Enhanced PWM Output Configuration bits
If CCP1M<3:2> = 00, 01, 10:
xx = P1A assigned as Capture/Compare input/output; P1B, P1C, P1D assigned as port pins
If CCP1M<3:2> = 11:
00 = Single output: P1A, P1B, P1C and P1D controlled by steering (See **Section 13.4.7 "Pulse Steering Mode"**).
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 **DC1B<1:0>**: PWM Duty Cycle bit 1 and bit 0
Capture mode:
Unused.
Compare mode:
Unused.
PWM mode:
These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.

bit 3-0 **CCP1M<3:0>**: Enhanced CCP Mode Select bits
0000 = Capture/Compare/PWM off (resets ECCP 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


© 2009-2016 Microchip Technology Inc.

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)

## 13.1 ECCP Outputs and Configuration

The enhanced CCP module may have up to four PWM outputs, depending on the selected operating mode. These outputs, designated P1A through P1D, are multiplexed with I/O pins on PORTC. The outputs that are active depend on the CCP operating mode selected. The pin assignments are summarized in Table 13-2.

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 TRISC direction bits for the port pins must also be set as outputs.

### 13.1.1 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 13-1: 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 12-1). The interactions between the two modules are summarized in Figure 13-1. In Asynchronous Counter mode, the capture operation will not work reliably.

## 13.2 Capture Mode

In Capture mode, the CCPR1H:CCPR1L register pair captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on the 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.

### 13.2.1 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. |

### 13.2.2 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 13.1.1 "CCP Module and Timer Resources"**).

### 13.2.3 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.

### 13.2.4 CCP PRESCALER

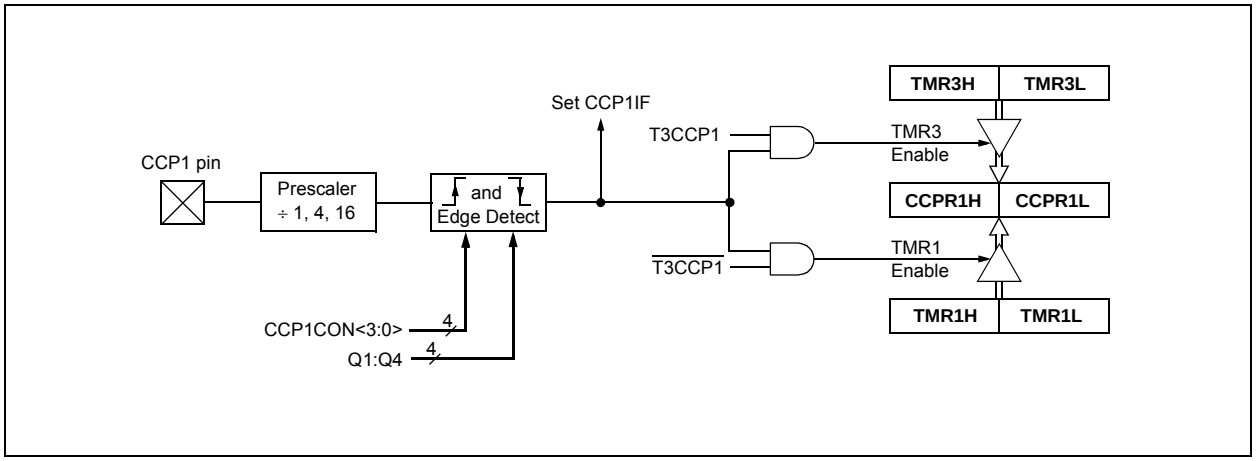
There are four prescaler settings in Capture mode; they are specified as part of the operating mode selected by the mode select bits (CCP1M<3:0>). Whenever the CCP module is turned off or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

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 13-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

**EXAMPLE 13-1: 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 13-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**





© 2009-2016 Microchip Technology Inc.

## 13.3 Compare Mode

In Compare mode, the 16-bit CCPR1 register value is constantly compared against either the TMR1 or TMR3 register pair value. When a match occurs, the CCP1 pin can be:

- Driven high
- Driven low
- Toggled (high-to-low or low-to-high)
- Remain unchanged (that is, reflects the state of the I/O latch)

The action on the pin is based on the value of the mode select bits (CCP1M<3:0>). At the same time, the interrupt flag bit, CCP1IF, is set.

### 13.3.1 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. |

### 13.3.2 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.

### 13.3.3 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.

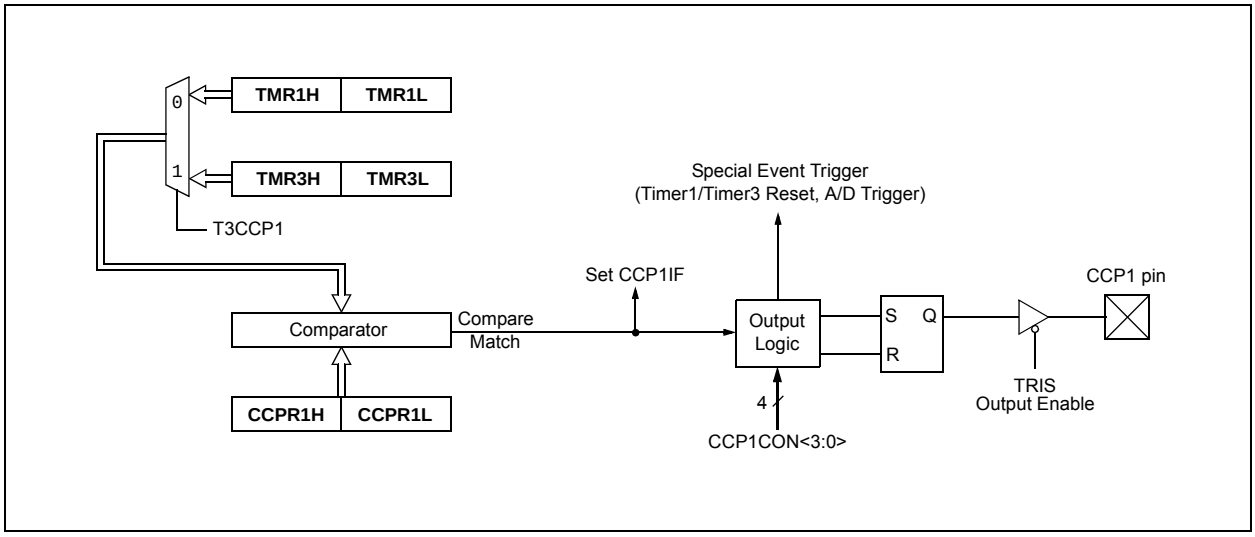
### 13.3.4 SPECIAL EVENT TRIGGER

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 CCPR1 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 13-2: COMPARE MODE OPERATION BLOCK DIAGRAM**



## 13.4 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.

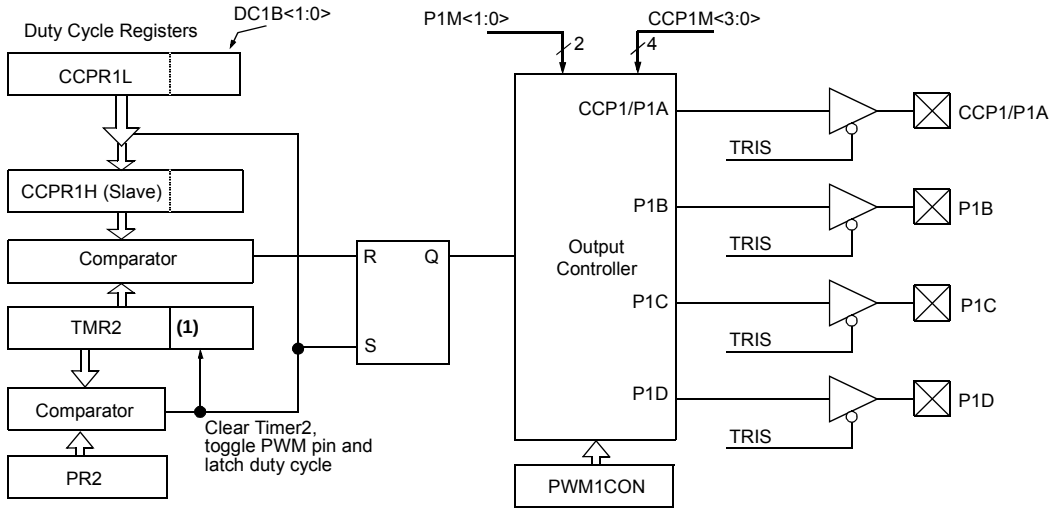
The PWM outputs are multiplexed with I/O pins and are designated P1A, P1B, P1C and P1D. The polarity of the PWM pins is configurable and is selected by setting the CCP1M bits in the CCP1CON register appropriately.

Table 13-1 shows the pin assignments for each Enhanced PWM mode.

Figure 13-3 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 13-3: EXAMPLE SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODE**




**Note 1:** The 8-bit timer TMR2 register is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler to create the 10-bit time base.

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

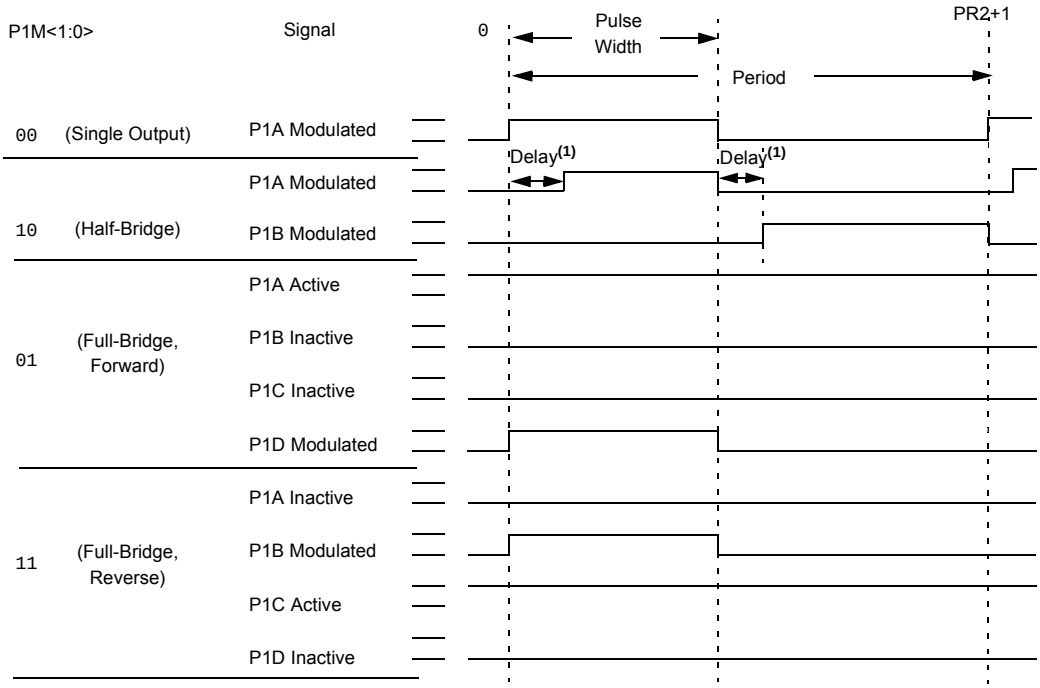
**TABLE 13-2: EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES**

| ECCP Mode | P1M<1:0> | CCP1/P1A | P1B | P1C | P1D |
|---|---|---|---|---|---|
| Single | 00 | Yes(1) | Yes(1) | Yes(1) | Yes(1) |
| 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 13-4.


© 2009-2016 Microchip Technology Inc.

**FIGURE 13-4: EXAMPLE PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)**

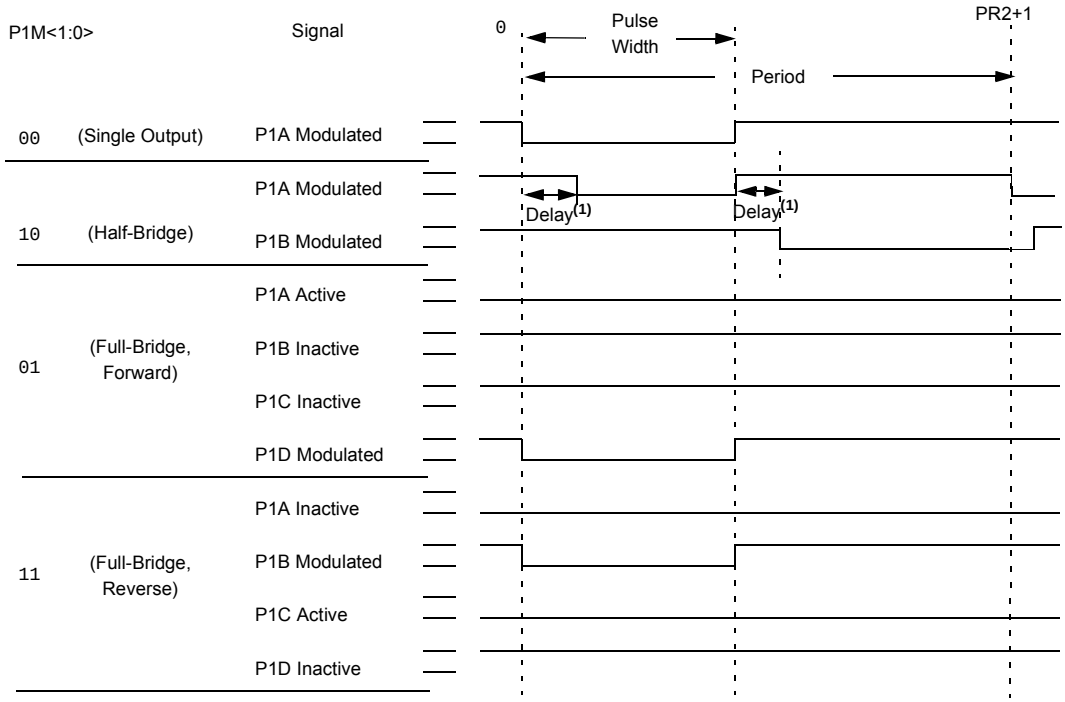



Relationships:

- Period = 4 * Tosc * (PR2 + 1) * (TMR2 Prescale Value)
- Pulse Width = Tosc * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * Tosc * (PWM1CON<6:0>)

**Note 1:** Dead-band delay is programmed using the PWM1CON register (**Section 13.4.6 "Programmable Dead-Band Delay Mode"**).

**FIGURE 13-5: EXAMPLE ENHANCED PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)**




Relationships:

- Period = 4 * Tosc * (PR2 + 1) * (TMR2 Prescale Value)
- Pulse Width = Tosc * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * Tosc * (PWM1CON<6:0>)

**Note 1:** Dead-band delay is programmed using the PWM1CON register (**Section 13.4.6 "Programmable Dead-Band Delay Mode"**).


© 2009-2016 Microchip Technology Inc.

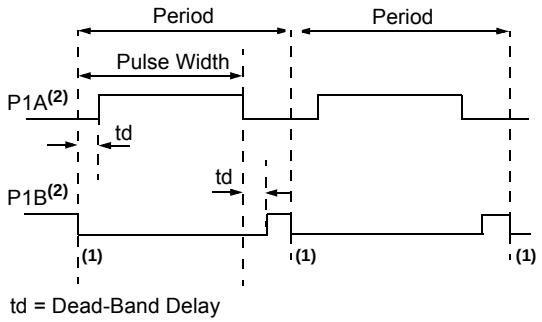
### 13.4.1 HALF-BRIDGE MODE

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 13-6). This mode can be used for half-bridge applications, as shown in Figure 13-7, 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 13.4.6 "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 13-6: EXAMPLE OF HALF-BRIDGE PWM OUTPUT**

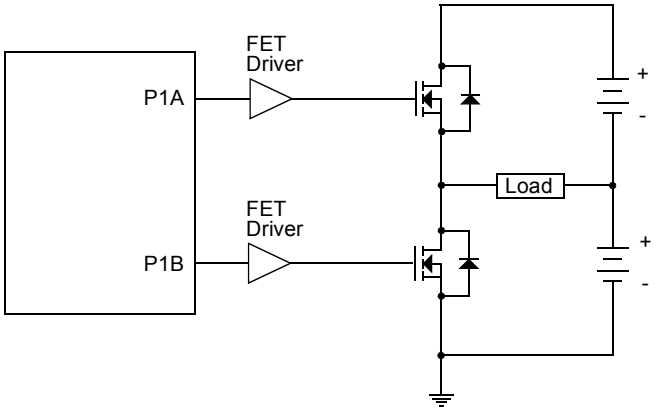



**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

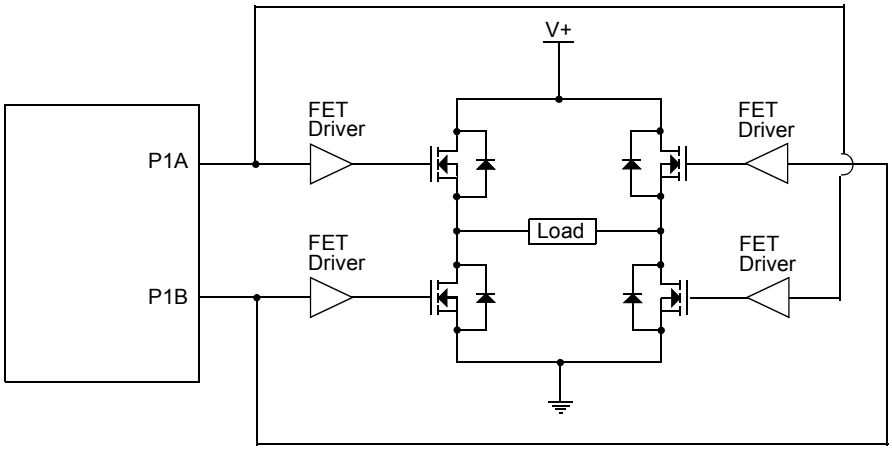
**2:** Output signals are shown as active-high.

**FIGURE 13-7: EXAMPLE OF HALF-BRIDGE APPLICATIONS**

**Standard Half-Bridge Circuit ("Push-Pull")**




**Half-Bridge Output Driving a Full-Bridge Circuit**



### 13.4.2 FULL-BRIDGE MODE

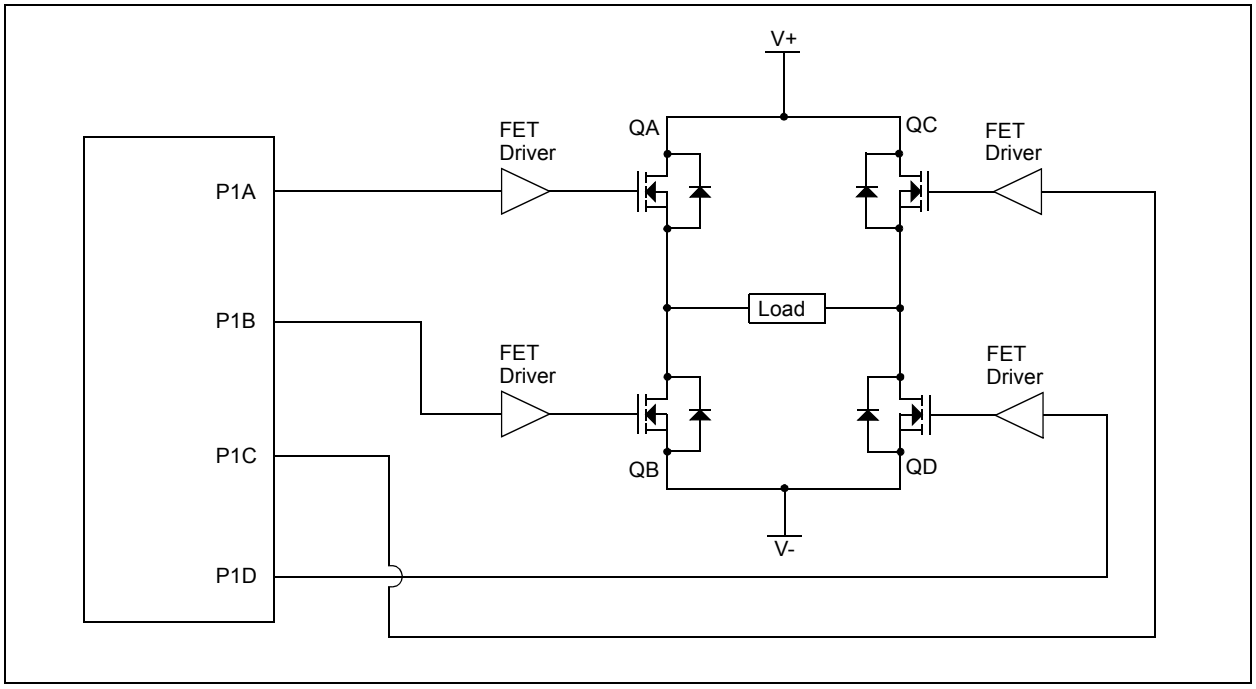
In Full-Bridge mode, all four pins are used as outputs. An example of Full-Bridge application is shown in Figure 13-8.

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 13-9.

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 13-9.

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.

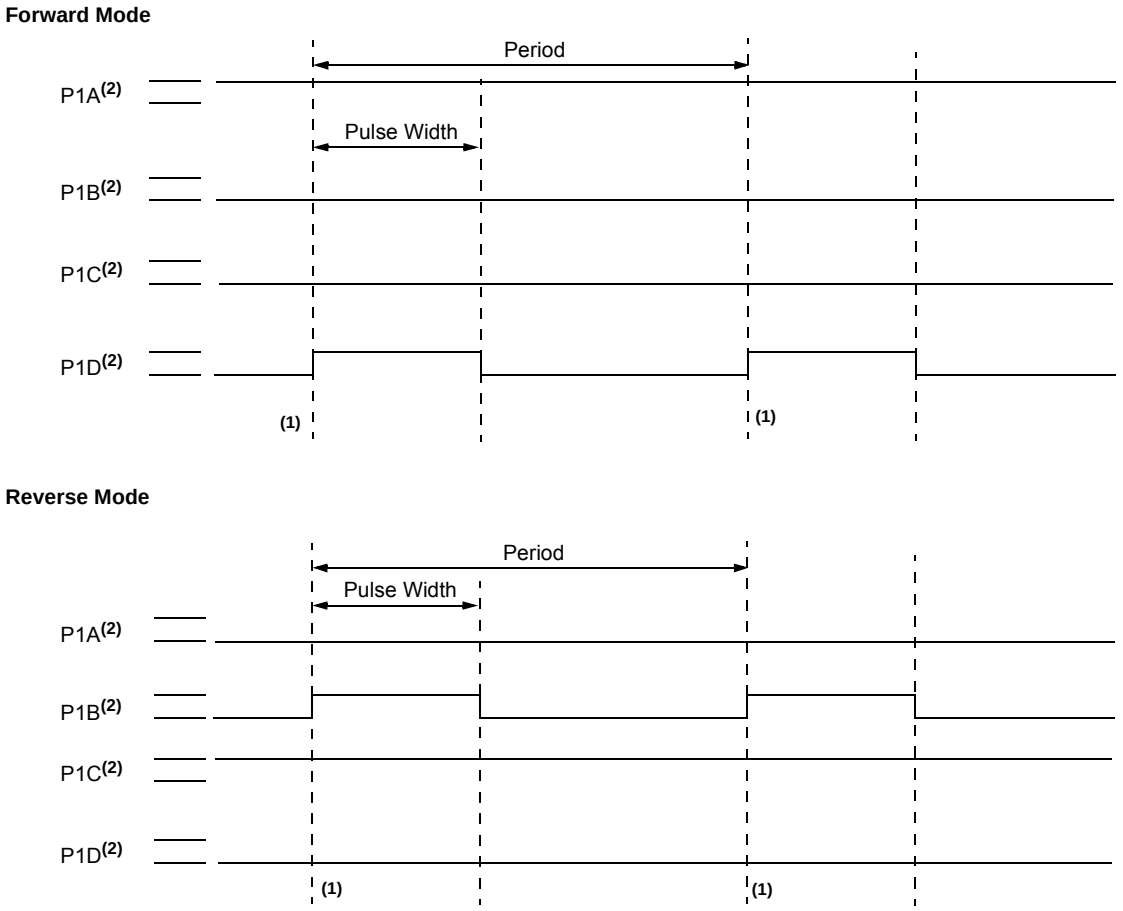
**FIGURE 13-8: EXAMPLE OF FULL-BRIDGE APPLICATION**





© 2009-2016 Microchip Technology Inc.

**FIGURE 13-9: EXAMPLE OF FULL-BRIDGE PWM OUTPUT**




**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

**2:** Output signal is shown as active-high.

#### 13.4.2.1 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 13-10 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.

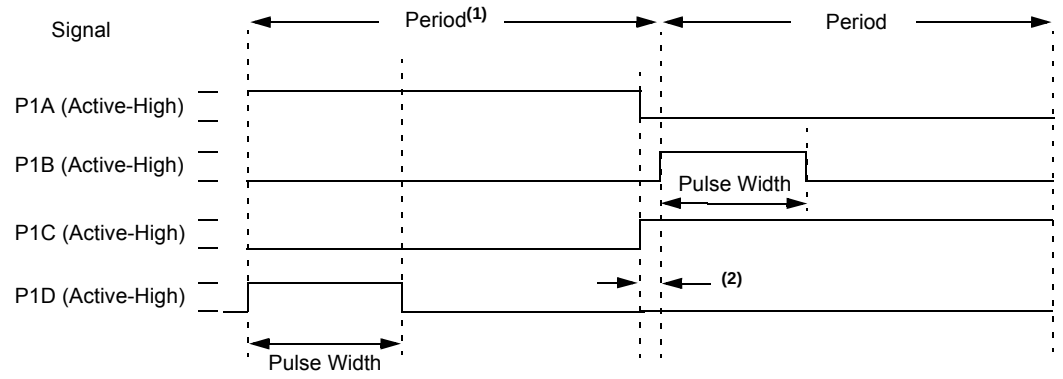
Figure 13-11 shows an example of the PWM direction changing from forward to reverse, at a near 100% duty cycle. In this example, at time t1, the output P1A and P1D become inactive, while output P1C becomes active. Since the turn off time of the power devices is longer than the turn on time, a shoot-through current will flow through power devices QC and QD (see Figure 13-8) 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 13-10: EXAMPLE OF PWM DIRECTION CHANGE**

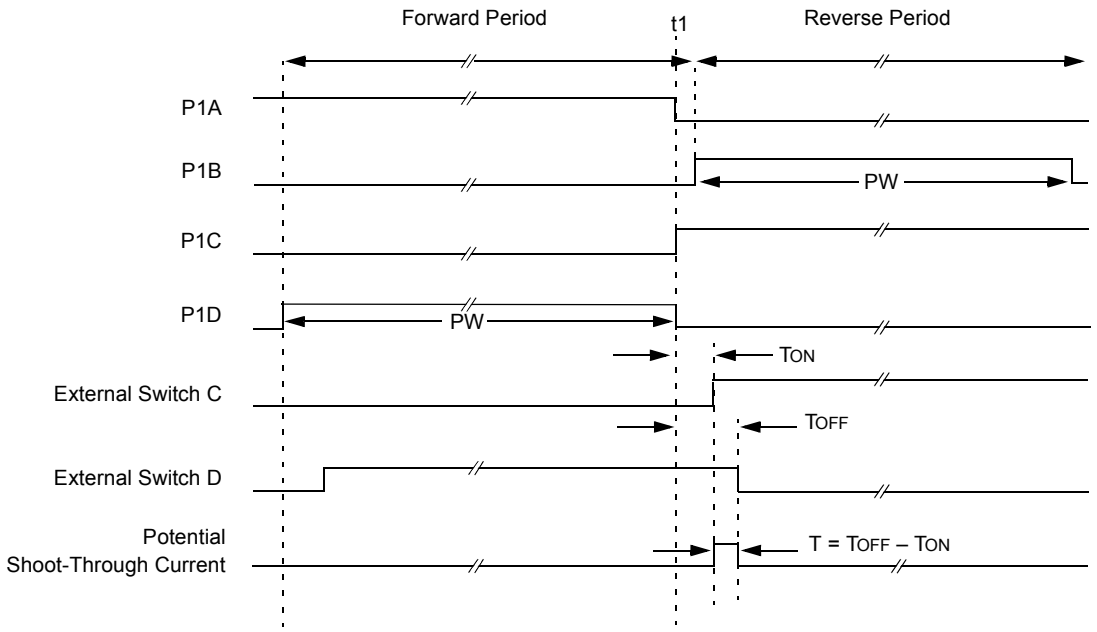



**Note 1:** The direction bit P1M1 of the CCP1CON register is written any time during the PWM cycle.

**2:** When changing directions, the P1A and P1C signals switch before the end of the current PWM cycle. The modulated P1B and P1D signals are inactive at this time. The length of this time is (1/FOSC) • TMR2 prescale value.


© 2009-2016 Microchip Technology Inc.

**FIGURE 13-11: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE**




**Note 1:** All signals are shown as active-high.

**2:** TON is the turn on delay of power switch QC and its driver.

**3:** TOFF is the turn off delay of power switch QD and its driver.

### 13.4.3 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.

### 13.4.4 ENHANCED PWM AUTO-SHUTDOWN MODE

The PWM mode supports an Auto-Shutdown mode that will disable the PWM outputs when an external shutdown event occurs. Auto-Shutdown mode places the PWM output pins into a predetermined state. This mode is used to help prevent the PWM from damaging the application.

The auto-shutdown sources are selected using the ECCPAS<2:0> bits of the ECCPAS register. A shutdown event may be generated by:

- A logic '0' on the INT0 pin
- A logic '1' on a comparator (Cx) output

A shutdown condition is indicated by the ECCPASE (Auto-Shutdown Event Status) bit of the ECCPAS register. If the bit is a '0', the PWM pins are operating normally. If the bit is a '1', the PWM outputs are in the shutdown state.

When a shutdown event occurs, two things happen:

The ECCPASE bit is set to '1'. The ECCPASE will remain set until cleared in firmware or an auto-restart occurs (see **Section 13.4.5 "Auto-Restart Mode"**).

The enabled PWM pins are asynchronously placed in their shutdown states. The PWM output pins are grouped into pairs [P1A/P1C] and [P1B/P1D]. The state of each pin pair is determined by the PSSAC and PSSBD bits of the ECCPAS register. Each pin pair may be placed into one of three states:

- Drive logic '1'
- Drive logic '0'
- Tri-state (high-impedance)

**REGISTER 13-2: ECCP1AS: ENHANCED CAPTURE/COMPARE/PWM AUTO-SHUTDOWN CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **ECCPASE:** ECCP Auto-Shutdown Event Status bit
1 = A shutdown event has occurred; ECCP outputs are in shutdown state
0 = ECCP outputs are operating

bit 6-4 **ECCPAS<2:0>:** ECCP Auto-shutdown Source Select bits
000 = Auto-Shutdown is disabled
001 = Comparator C1OUT output is high
010 = Comparator C2OUT output is high
011 = Either Comparator C1OUT or C2OUT is high
100 = VIL on INT0 pin
101 = VIL on INT0 pin or Comparator C1OUT output is high
110 = VIL on INT0 pin or Comparator C2OUT output is high
111 = VIL on INT0 pin or Comparator C1OUT or Comparator C2OUT is high

bit 3-2 **PSSACn:** Pins P1A and P1C Shutdown State Control bits
00 = Drive pins P1A and P1C to '0'
01 = Drive pins P1A and P1C to '1'
10 = Pins 1A and P1C tri-state
11 = Reserved, do not use

bit 1-0 **PSSBDn:** Pins P1B and P1D Shutdown State Control bits
00 = Drive pins P1B and P1D to '0'
01 = Drive pins P1B and P1D to '1'
10 = Pins P1B and P1D tri-state
11 = Reserved, do not use


© 2009-2016 Microchip Technology Inc.

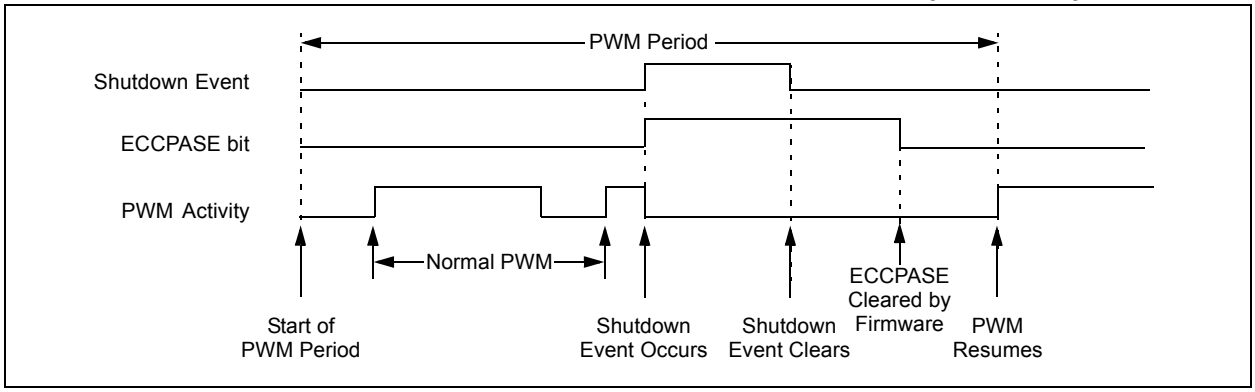
**Note 1:** The auto-shutdown condition is a level-based signal, not an edge-based signal. As long as the level is present, the auto-shutdown will persist.

**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.

**4:** Prior to an auto-shutdown event caused by a comparator output or INT pin event, a software shutdown can be triggered in firmware by setting the CCPxASE bit to a '1'. The Auto-Restart feature tracks the active status of a shutdown caused by a comparator output or INT pin event only, so if it is enabled at this time, it will immediately clear this bit and restart the ECCP module at the beginning of the next PWM period.

**FIGURE 13-12: PWM AUTO-SHUTDOWN WITH FIRMWARE RESTART (PRSEN = 0)**

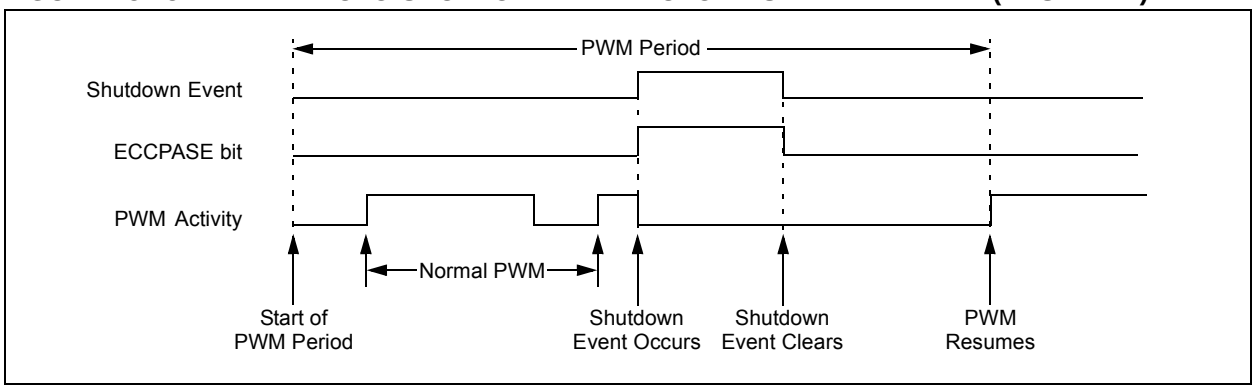


### 13.4.5 AUTO-RESTART MODE

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 13-13: PWM AUTO-SHUTDOWN WITH AUTO-RESTART ENABLED (PRSEN = 1)**





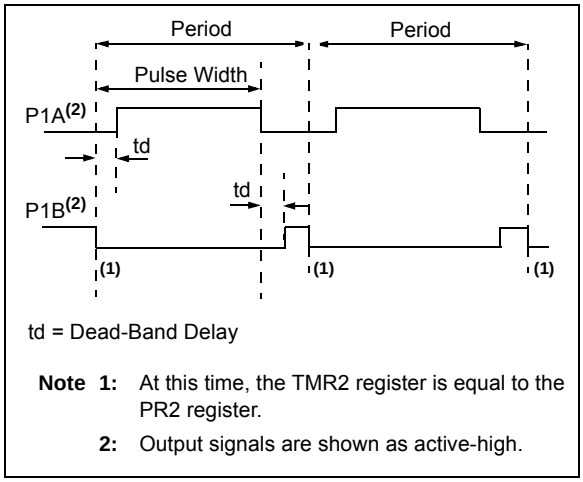
© 2009-2016 Microchip Technology Inc.

### 13.4.6 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 13-14 for illustration. The lower seven bits of the associated PWM1CON register (Register 13-3) sets the delay period in terms of microcontroller instruction cycles (TCY or 4 TOSC).

**FIGURE 13-14: EXAMPLE OF HALF-BRIDGE PWM OUTPUT**

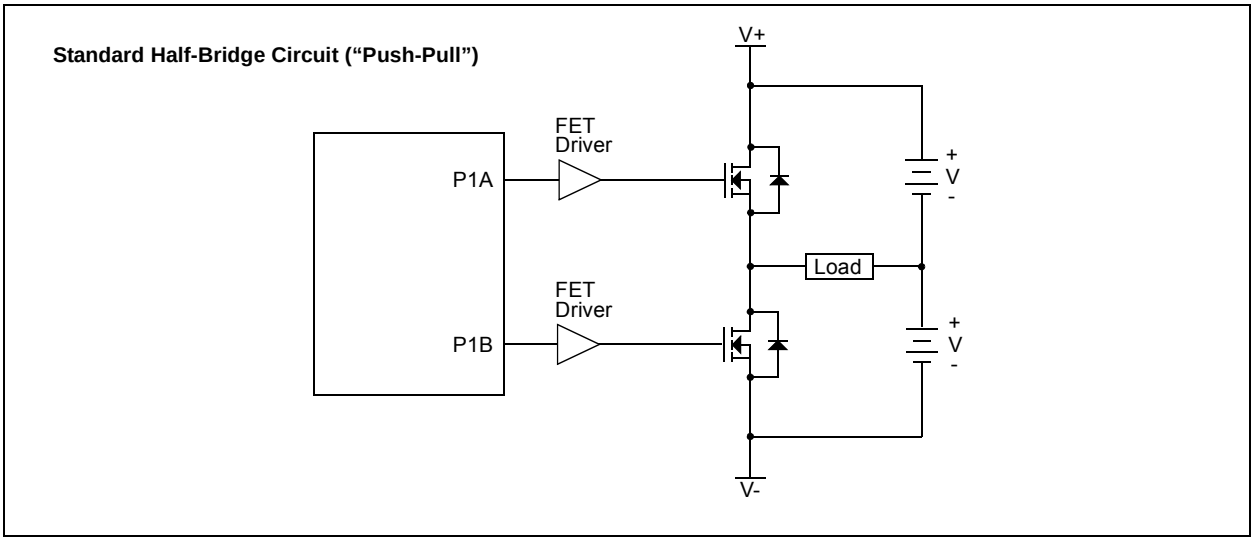



td = Dead-Band Delay

**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

**2:** Output signals are shown as active-high.

**FIGURE 13-15: EXAMPLE OF HALF-BRIDGE APPLICATIONS**



**REGISTER 13-3: PWM1CON: ENHANCED 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 |
|---|---|---|---|---|---|---|---|
| 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


© 2009-2016 Microchip Technology Inc.

### 13.4.7 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 13-2.

**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 13.4.4 "Enhanced PWM Auto-shutdown mode"**. An auto-shutdown event will only affect pins that have PWM outputs enabled.

**REGISTER 13-4: PSTRCON: PULSE STEERING CONTROL REGISTER(1)**

| 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

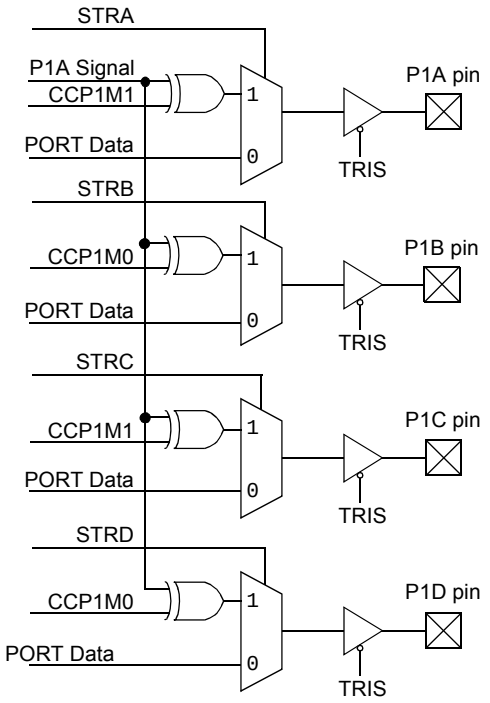
bit 2 **STRC:** Steering Enable bit C
1 = P1C pin has the PWM waveform with polarity control from CCP1M<1:0>
0 = P1C pin is assigned to port pin

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.

**FIGURE 13-16: SIMPLIFIED STEERING BLOCK DIAGRAM**




**Note 1:** Port outputs are configured as shown when the CCP1CON register bits P1M<1:0> = 00 and CCP1M<3:2> = 11.

**2:** Single PWM output requires setting at least one of the STRx bits.


© 2009-2016 Microchip Technology Inc.

#### 13.4.7.1 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 13-17 and 13-18 illustrate the timing diagrams of the PWM steering depending on the STRSYNC setting.

**FIGURE 13-17: EXAMPLE OF STEERING EVENT AT END OF INSTRUCTION (STRSYNC = 0)**

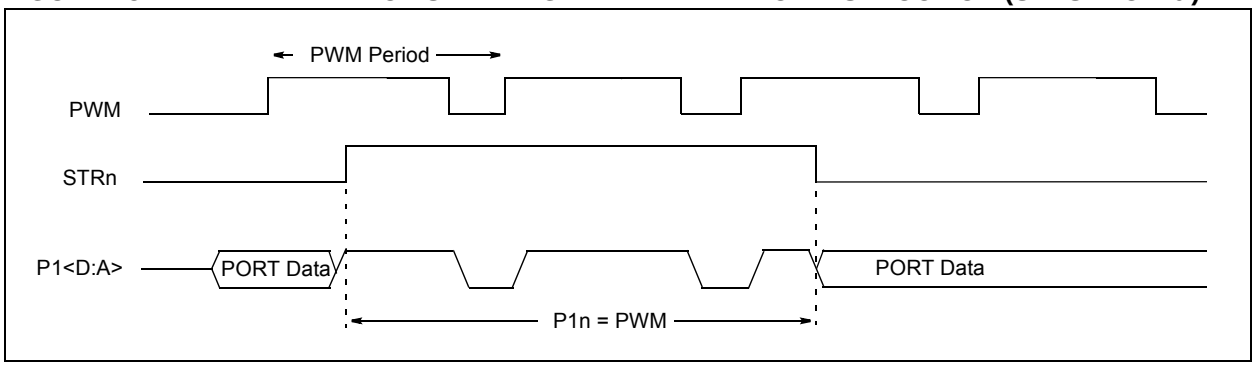
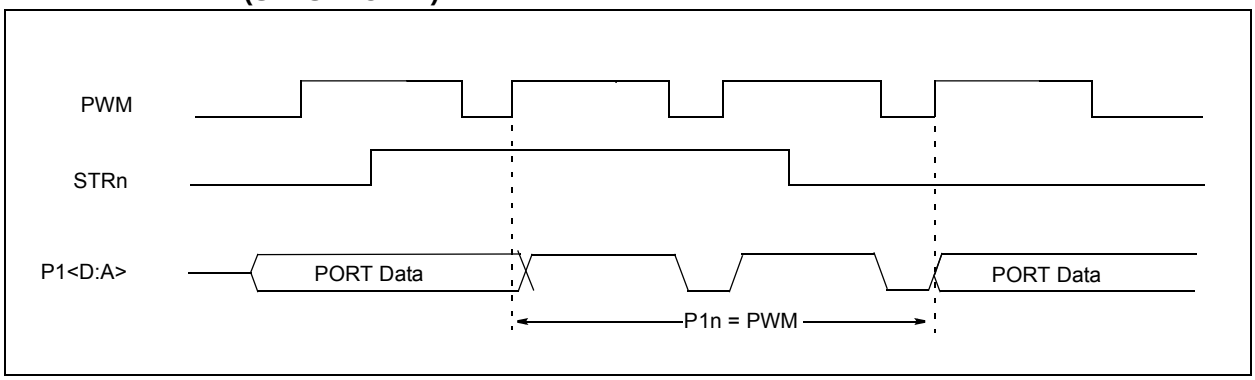



**FIGURE 13-18: EXAMPLE OF STEERING EVENT AT BEGINNING OF INSTRUCTION (STRSYNC = 1)**



### 13.4.8 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.

#### 13.4.8.1 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.

### 13.4.9 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.

**TABLE 13-3: 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 |
|---|---|---|---|---|---|---|---|---|---|
| CCPR1H | Capture/Compare/PWM Register 1, High Byte | | | | | | | | 247 |
| CCPR1L | Capture/Compare/PWM Register 1, Low Byte | | | | | | | | 247 |
| CCP1CON | P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 247 |
| ECCP1AS | ECCPASE | ECCPAS2 | ECCPAS1 | ECCPAS0 | PSSAC1 | PSSAC0 | PSSBD1 | PSSBD0 | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | — | TMR3IP | — | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | — | TMR3IE | — | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | — | TMR3IF | — | 248 |
| PR2 | Timer2 Period Register | | | | | | | | 246 |
| PWM1CON | PRSEN | PDC6 | PDC5 | PDC4 | PDC3 | PDC2 | PDC1 | PDC0 | 247 |
| RCON | IPEN | SBOREN | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 246 |
| TMR1H | Timer1 Register, High Byte | | | | | | | | 246 |
| TMR1L | Timer1 Register, Low Byte | | | | | | | | 246 |
| TMR2 | Timer2 Register | | | | | | | | 246 |
| TMR3H | Timer3 Register, High Byte | | | | | | | | 247 |
| TMR3L | Timer3 Register, Low Byte | | | | | | | | 247 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 248 |
| T1CON | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{T1SYNC}$ | TMR1CS | TMR1ON | 246 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | 246 |
| T3CON | RD16 | — | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{T3SYNC}$ | TMR3CS | TMR3ON | 247 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during ECCP operation.


© 2009-2016 Microchip Technology Inc.

# 14.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

## 14.1 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

## 14.2 SPI Mode

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

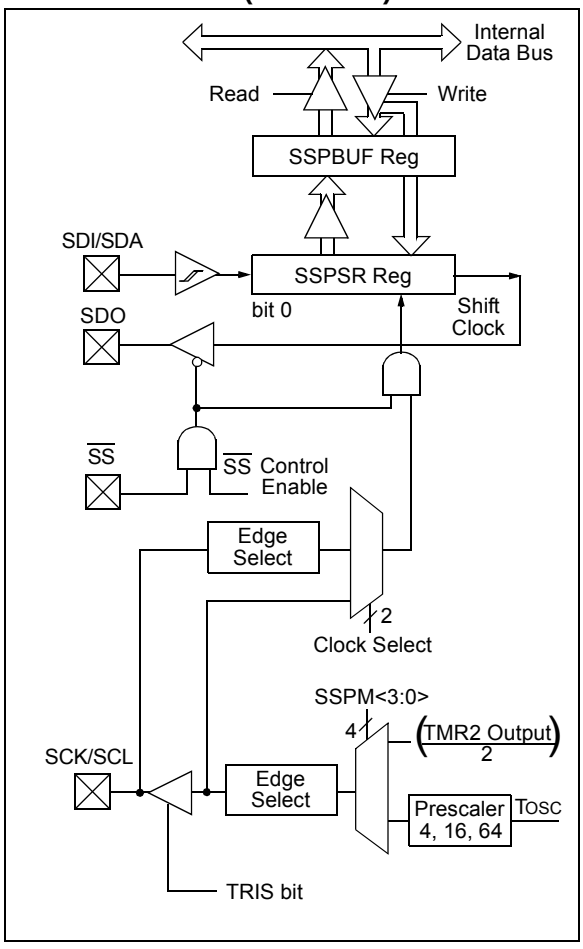
- Serial Data Out – SDO
- Serial Data In – SDI
- Serial Clock – SCK

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

- Slave Select – $\overline{SS}$

Figure 14-1 shows the block diagram of the MSSP module when operating in SPI mode.

**FIGURE 14-1: MSSP BLOCK DIAGRAM (SPI MODE)**



### 14.2.1 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 six 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 14-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)**

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Select bit(1)
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit
Used in $I^2C$ mode only.

bit 4 **P:** Stop bit
Used in $I^2C$ mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** Start bit
Used in $I^2C$ mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{Write}$ Information bit
Used in $I^2C$ mode only.

bit 1 **UA:** Update Address bit
Used in $I^2C$ mode only.

bit 0 **BF:** Buffer Full Status bit (Receive mode only)
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty

**Note 1:** Polarity of clock state is set by the CKP bit of the SSPCON1 register.


© 2009-2016 Microchip Technology Inc.

**REGISTER 14-2: SSPCON1: MSSP CONTROL 1 REGISTER (SPI MODE)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| WCOL | SSPOV | 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 **WCOL:** Write Collision Detect bit (Transmit mode only)
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared by software)
0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit(1)
SPI Slave mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared by software).
0 = No overflow

bit 5 **SSPEN:** Synchronous Serial Port Enable bit(2)
1 = Enables serial port and configures SCK, SDO, SDI and $\overline{SS}$ as serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

bit 3-0 **SSPM<3:0>:** Synchronous Serial Port Mode Select bits(3)
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
0011 = SPI Master mode, clock = TMR2 output/2
0010 = SPI Master mode, clock = FOSC/64
0001 = SPI Master mode, clock = FOSC/16
0000 = SPI Master mode, clock = FOSC/4

**Note 1:** In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.
**2:** When enabled, these pins must be properly configured as input or output.
**3:** Bit combinations not specifically listed here are either reserved or implemented in $I^2C$ mode only.

### 14.2.2 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 14-1 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 14-1: LOADING THE SSPBUF (SSPSR) REGISTER**

```
LOOP    BTFSS   SSPSTAT, BF     ;Has data been received (transmit complete)?
        BRA     LOOP            ;No
        MOVF    SSPBUF, W       ;WREG reg = contents of SSPBUF
        MOVWF   RXDATA          ;Save in user RAM, if data is meaningful
        MOVF    TXDATA, W       ;W reg = contents of TXDATA
        MOVWF   SSPBUF          ;New data to xmit
```


© 2009-2016 Microchip Technology Inc.

### 14.2.3 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 $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI is automatically controlled by the SPI module
- SDO must have corresponding TRIS bit cleared
- SCK (Master mode) must have corresponding TRIS bit cleared
- SCK (Slave mode) must have corresponding TRIS bit set
- $\overline{SS}$ must have corresponding TRIS bit set

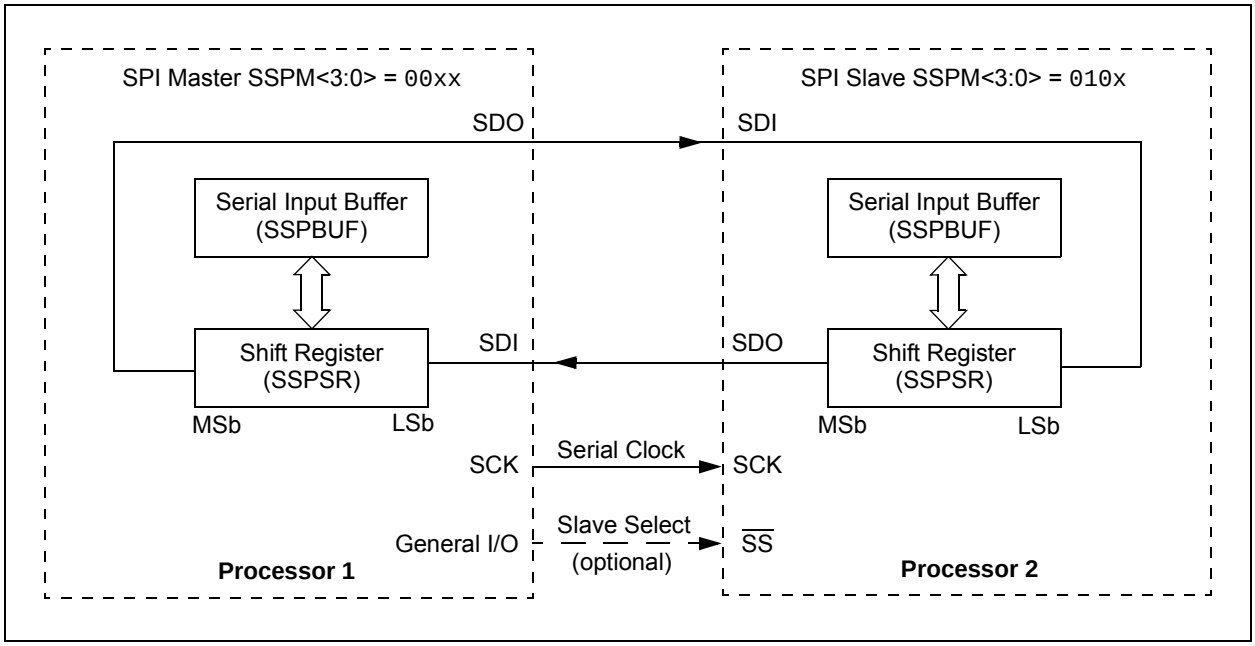
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

### 14.2.4 TYPICAL CONNECTION

Figure 14-2 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 14-2: TYPICAL SPI MASTER/SLAVE CONNECTION**



### 14.2.5 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 14-2) 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).

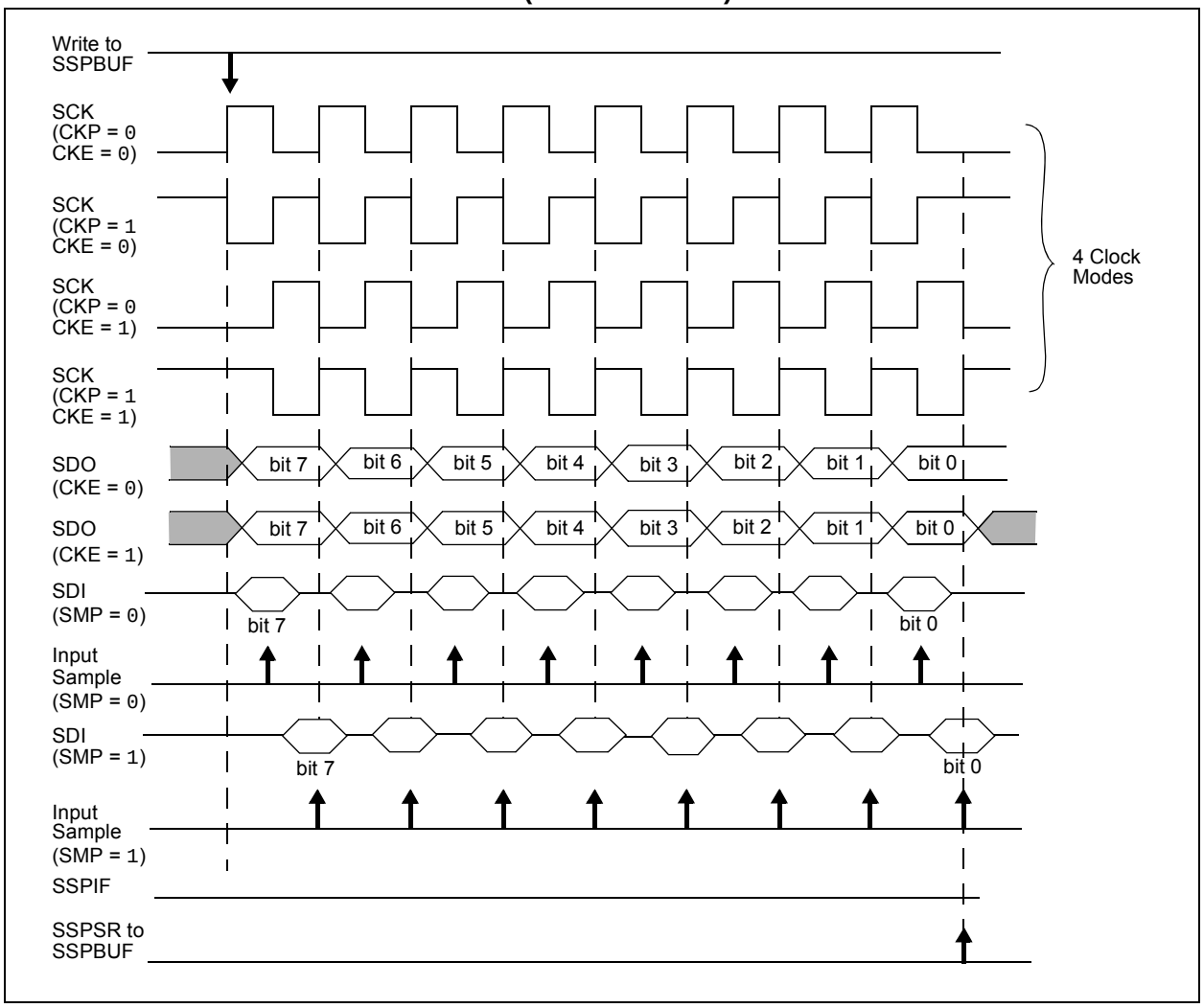
The clock polarity is selected by appropriately programming the CKP bit of the SSPCON1 register. This then, would give waveforms for SPI communication as shown in Figure 14-3, Figure 14-5 and Figure 14-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable 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 14-3 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 14-3: SPI MODE WAVEFORM (MASTER MODE)**





© 2009-2016 Microchip Technology Inc.

### 14.2.6 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.

### 14.2.7 SLAVE SELECT SYNCHRONIZATION

The $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 0100). When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven. When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

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

**2:** When the SPI is used in Slave mode with CKE set the $\overline{SS}$ 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 $\overline{SS}$ pin to a high level or clearing the SSPEN bit.

**FIGURE 14-4: SLAVE SYNCHRONIZATION WAVEFORM**

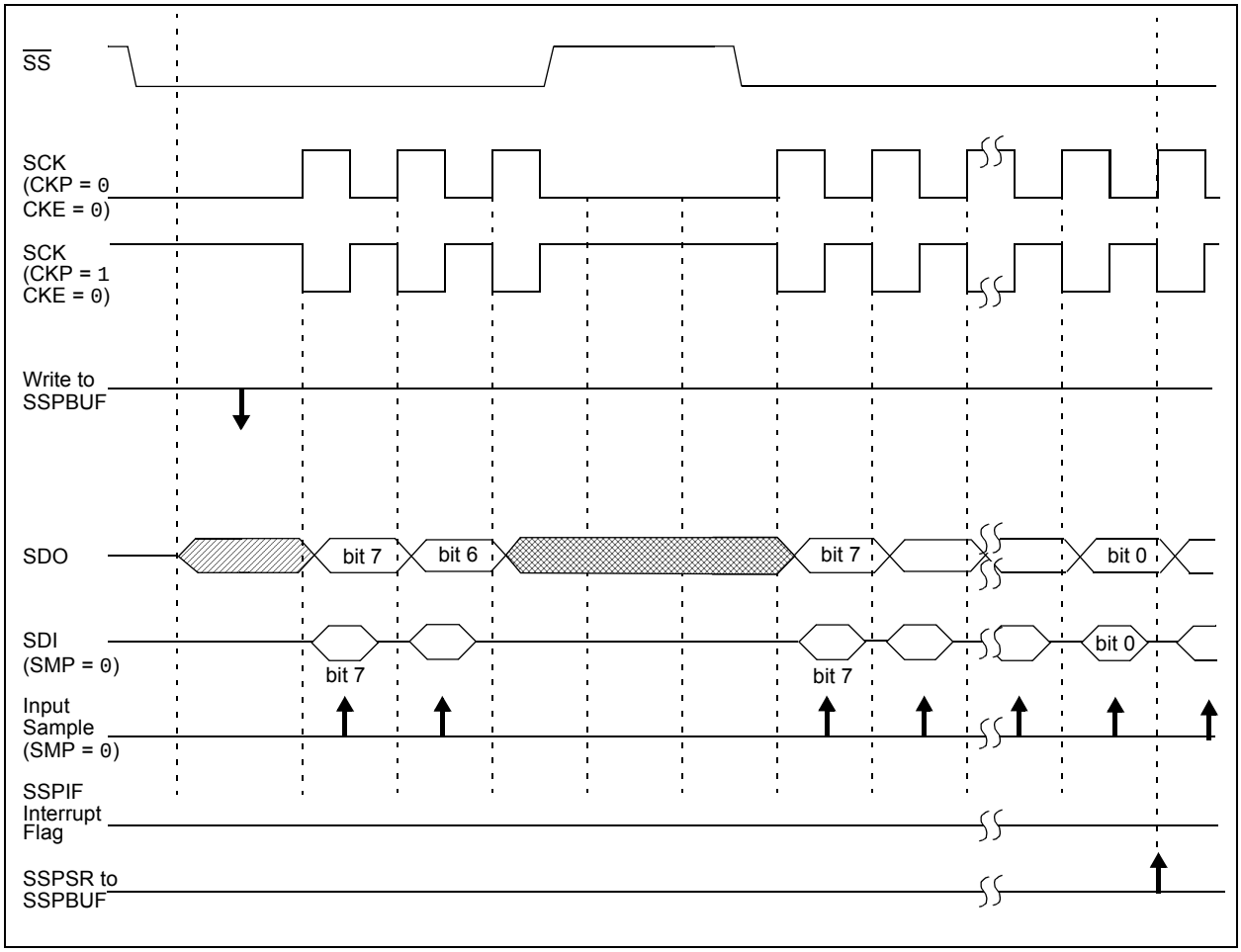


**FIGURE 14-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**

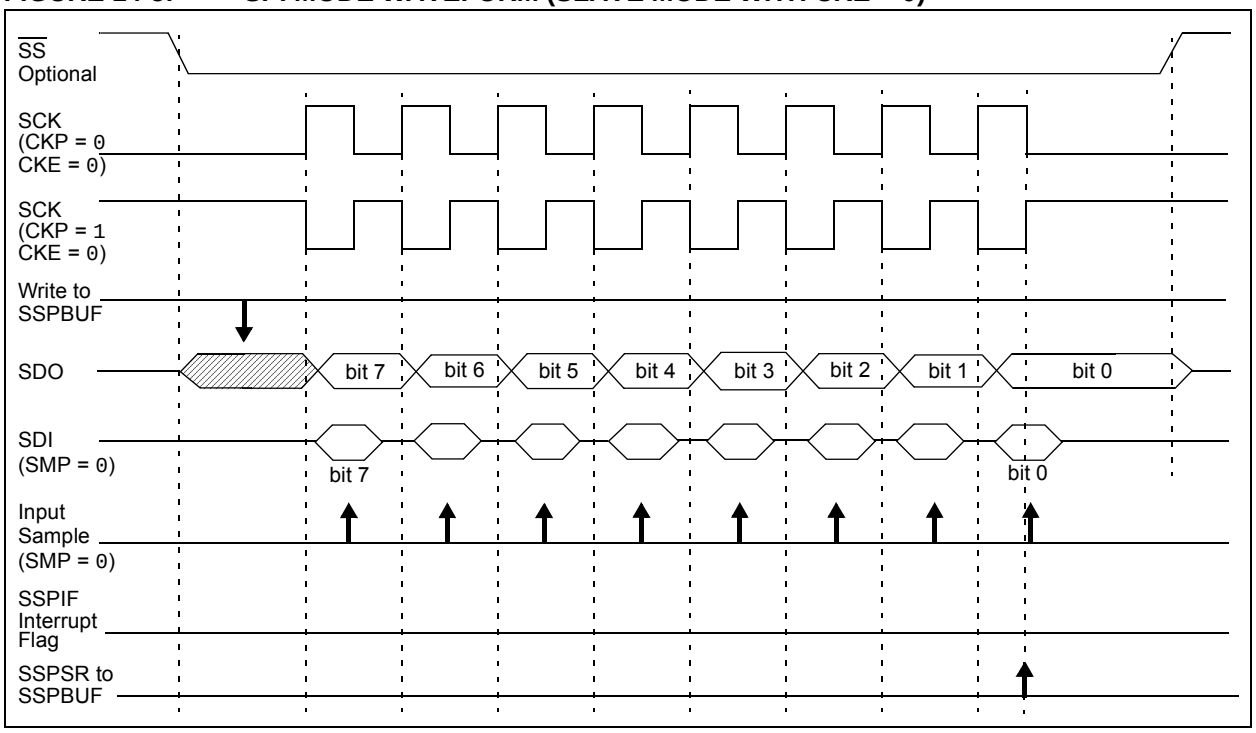
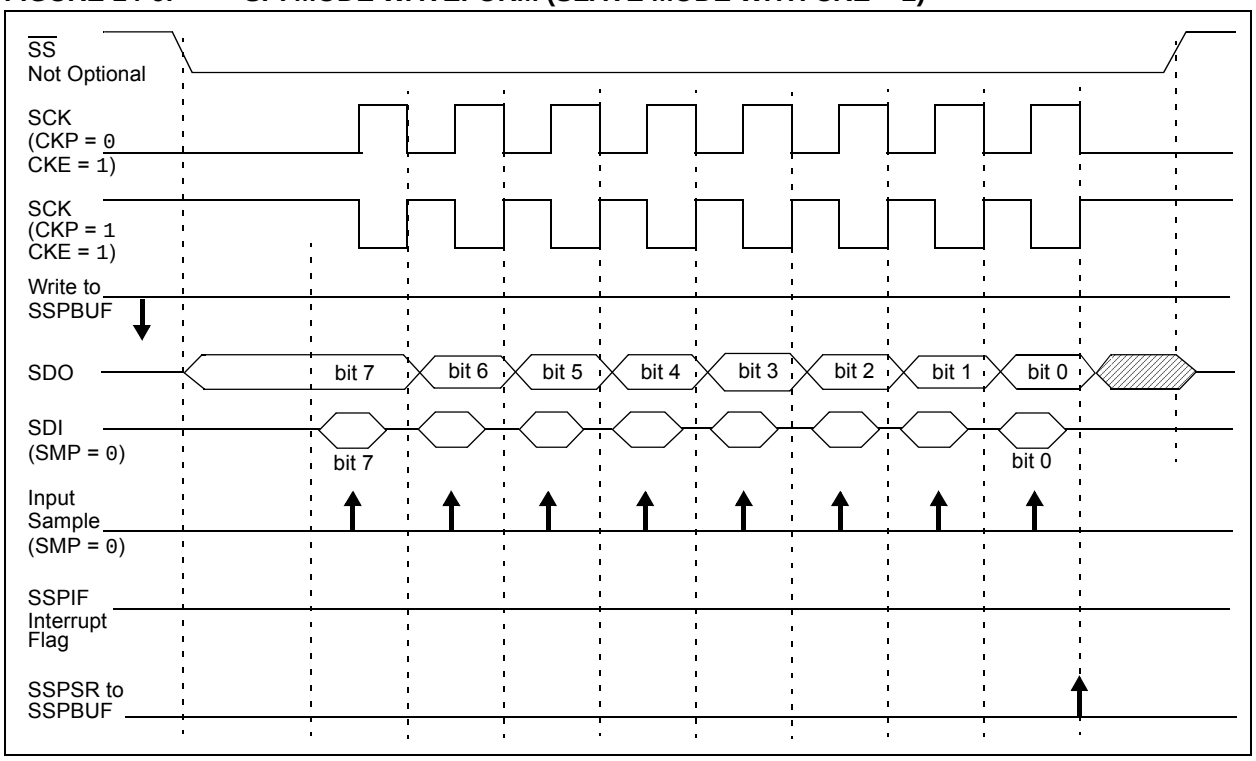



**FIGURE 14-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**





© 2009-2016 Microchip Technology Inc.

### 14.2.8 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 18.0 "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 eight bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

### 14.2.9 EFFECTS OF A RESET

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

### 14.2.10 BUS MODE COMPATIBILITY

Table 14-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

**TABLE 14-1: 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 14-2: REGISTERS ASSOCIATED WITH SPI OPERATION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on page |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | — | — | — | — | 248 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 248 |
| SSPBUF | SSP Receive Buffer/Transmit Register | | | | | | | | 246 |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 246 |
| SSPSTAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 246 |

**Legend:** Shaded cells are not used by the MSSP in SPI mode.

## 14.3 I²C Mode

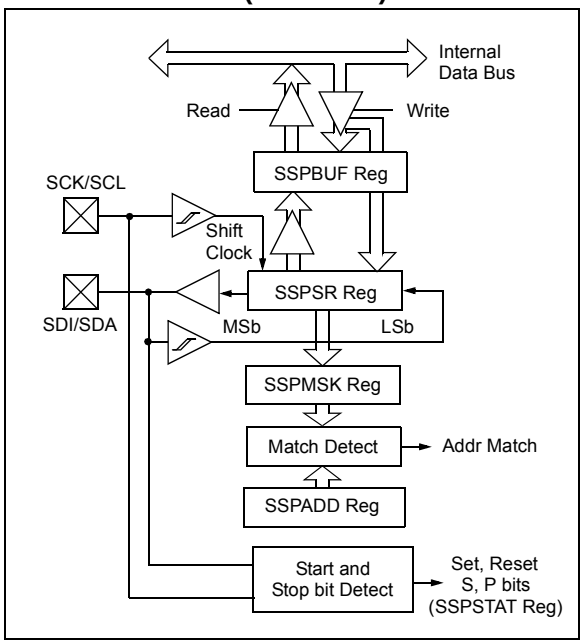
The MSSP module in I²C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial clock – SCL
- Serial data – SDA

**Note:** The user must configure these pins as inputs with the corresponding TRIS bits.

**FIGURE 14-7: MSSP BLOCK DIAGRAM (I²C MODE)**




### 14.3.1 REGISTERS

The MSSP module has seven registers for I²C operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Control Register 2 (SSPCON2)
- MSSP Status register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible
- MSSP Address Register (SSPADD)
- MSSP Address Mask (SSPMSK)

SSPCON1, SSPCON2 and SSPSTAT are the control and STATUS registers in I²C mode operation. The SSPCON1 and SSPCON2 registers are readable and writable. The lower six 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 I²C 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.


© 2009-2016 Microchip Technology Inc.

**REGISTER 14-3: SSPSTAT: MSSP STATUS REGISTER (I²C MODE)**

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SMP | CKE | D/$\overline{A}$ | P(1) | S(1) | R/$\overline{W}$(2, 3) | UA | BF |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **SMP:** Slew Rate Control bit
In Master or Slave mode:
1 = Slew rate control disabled for standard Speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for High-Speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit
In Master or Slave mode:
1 = Enable SMBus specific inputs
0 = Disable SMBus specific inputs

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit
In Master mode:
Reserved.
In Slave mode:
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received was an address

bit 4 **P:** Stop bit(1)
1 = Indicates that a Stop bit has been detected last
0 = Stop bit was not detected last

bit 3 **S:** Start bit(1)
1 = Indicates that a Start bit has been detected last
0 = Start bit was not detected last

bit 2 **R/$\overline{W}$:** Read/$\overline{\text{Write}}$ Information bit (I²C mode only)(2, 3)
In Slave mode:
1 = Read
0 = Write
In Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress

bit 1 **UA:** Update Address bit (10-bit Slave mode only)
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit
In Transmit mode:
1 = SSPBUF is full
0 = SSPBUF is empty
In Receive mode:
1 = SSPBUF is full (does not include the $\overline{\text{ACK}}$ and Stop bits)
0 = SSPBUF is empty (does not include the $\overline{\text{ACK}}$ and Stop bits)

**Note 1:** This bit is cleared on Reset and when SSPEN is cleared.
**2:** This bit holds the R/$\overline{W}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{\text{ACK}}$ bit.
**3:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the Master mode is active.

**REGISTER 14-4: SSPCON1: MSSP CONTROL 1 REGISTER (I²C MODE)**

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

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

bit 7 **WCOL:** Write Collision Detect bit

In Master Transmit mode:
1 = A write to the SSPBUF register was attempted while the I²C conditions were not valid for a transmission to be started (must be cleared by software)
0 = No collision

In Slave Transmit mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared by software)
0 = No collision

In Receive mode (Master or Slave modes):
This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared by software)
0 = No overflow

In Transmit mode:
This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins
0 = Disables serial port and configures these pins as I/O port pins
When enabled, the SDA and SCL pins must be properly configured as inputs.

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

In Slave mode:
1 = Release clock
0 = Holds clock low (clock stretch), used to ensure data setup time

In Master mode:
Unused in this mode.

bit 3-0 **SSPM<3:0>:** Synchronous Serial Port Mode Select bits

1111 = I²C Slave mode, 10-bit address with Start and Stop bit interrupts enabled
1110 = I²C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1011 = I²C Firmware Controlled Master mode (Slave Idle)
1000 = I²C Master mode, clock = FOSC/(4 * (SSPADD + 1))
0111 = I²C Slave mode, 10-bit address
0110 = I²C Slave mode, 7-bit address
Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.


© 2009-2016 Microchip Technology Inc.

**REGISTER 14-5: SSPCON2: MSSP CONTROL REGISTER (I²C MODE)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| GCEN | ACKSTAT | ACKDT(2) | ACKEN(1) | RCEN(1) | PEN(1) | RSEN(1) | SEN(1) |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **GCEN:** 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 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)(2)
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)(1)
1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence Idle

bit 3 **RCEN:** Receive Enable bit (Master mode only)(1)
1 = Enables Receive mode for I²C
0 = Receive Idle

bit 2 **PEN:** Stop Condition Enable bit (Master mode only)(1)
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enable bit (Master mode only)(1)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enable/Stretch Enable bit(1)
In Master mode:
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle
In Slave mode:
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is disabled

**Note 1:** For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I²C 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.

### 14.3.2 OPERATION

The MSSP module functions are enabled by setting SSPEN bit of the SSPCON1 register.

The SSPCON1 register allows control of the $I^2C$ operation. Four mode selection bits of the SSPCON1 register allow one of the following $I^2C$ modes to be selected:

- $I^2C$ Master mode, clock = (FOSC/(4*(SSPADD + 1))
- $I^2C$ Slave mode (7-bit address)
- $I^2C$ Slave mode (10-bit address)
- $I^2C$ Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- $I^2C$ Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- $I^2C$ Firmware Controlled Master mode, slave is Idle

Selection of any $I^2C$ mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain, provided these pins are programmed to inputs by setting the appropriate TRIS bits

| **Note:** | To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins. |
|---|---|

### 14.3.3 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 $I^2C$ Slave mode hardware will always generate an interrupt on an address match. Through the mode select bits, the user can also choose to interrupt on Start and Stop bits

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge ($\overline{ACK}$) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

Any combination of the following conditions will cause the MSSP module not to give this $\overline{ACK}$ pulse:

- The Buffer Full bit, BF 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 $I^2C$ specification, as well as the requirement of the MSSP module, are shown in **Section 26.0 "Electrical Specifications"**.

#### 14.3.3.1 Addressing

Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the eight bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register SSPSR<7:1> is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

1. The SSPSR register value is loaded into the SSPBUF register.
2. The Buffer Full bit, BF, is set.
3. An $\overline{ACK}$ pulse is generated.
4. MSSP Interrupt Flag bit, SSPIF 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/$\overline{W}$ 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/$\overline{W}$ bit set (bits SSPIF, BF, R/$\overline{W}$ 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.


© 2009-2016 Microchip Technology Inc.

#### 14.3.3.2 Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and the SDA line is held low ($\overline{ACK}$).

When the address byte overflow condition exists, then the no Acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit BF 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 14.3.4 "Clock Stretching"** for more detail.

#### 14.3.3.3 Transmission

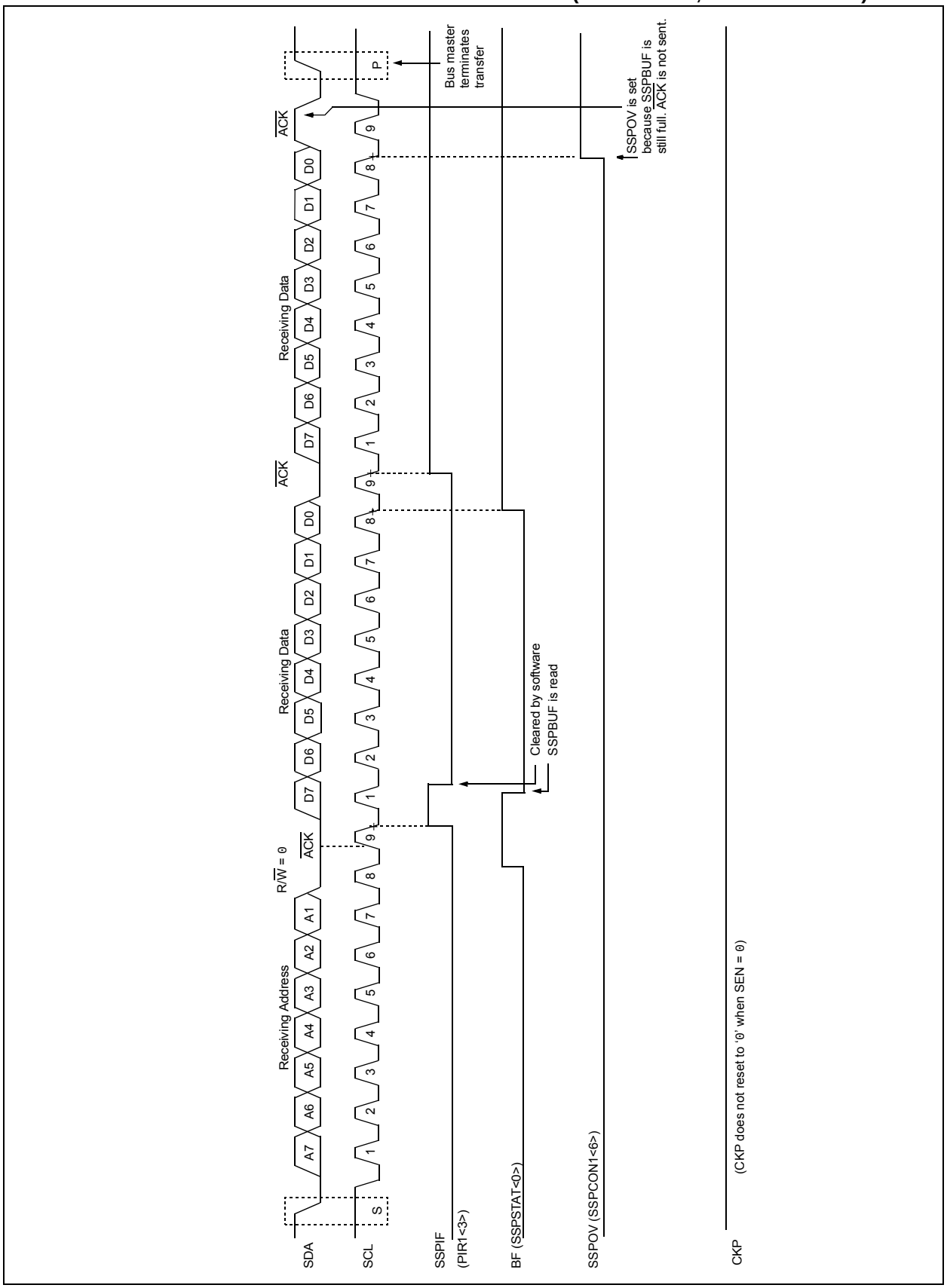
When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{ACK}$ pulse will be sent on the ninth bit and pin SCK/SCL is held low regardless of SEN (see **Section 14.3.4 "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 14-9).

The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not $\overline{ACK}$), then the data transfer is complete. In this case, when the $\overline{ACK}$ is latched by the slave, the slave logic is reset (resets SSPSTAT register) and the slave monitors for another occurrence of the Start bit. If the SDA line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPBUF register. Again, pin 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.

© 2009-2016 Microchip Technology Inc.

**FIGURE 14-8: I²C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)**





© 2009-2016 Microchip Technology Inc.

**FIGURE 14-9: I²C SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)**

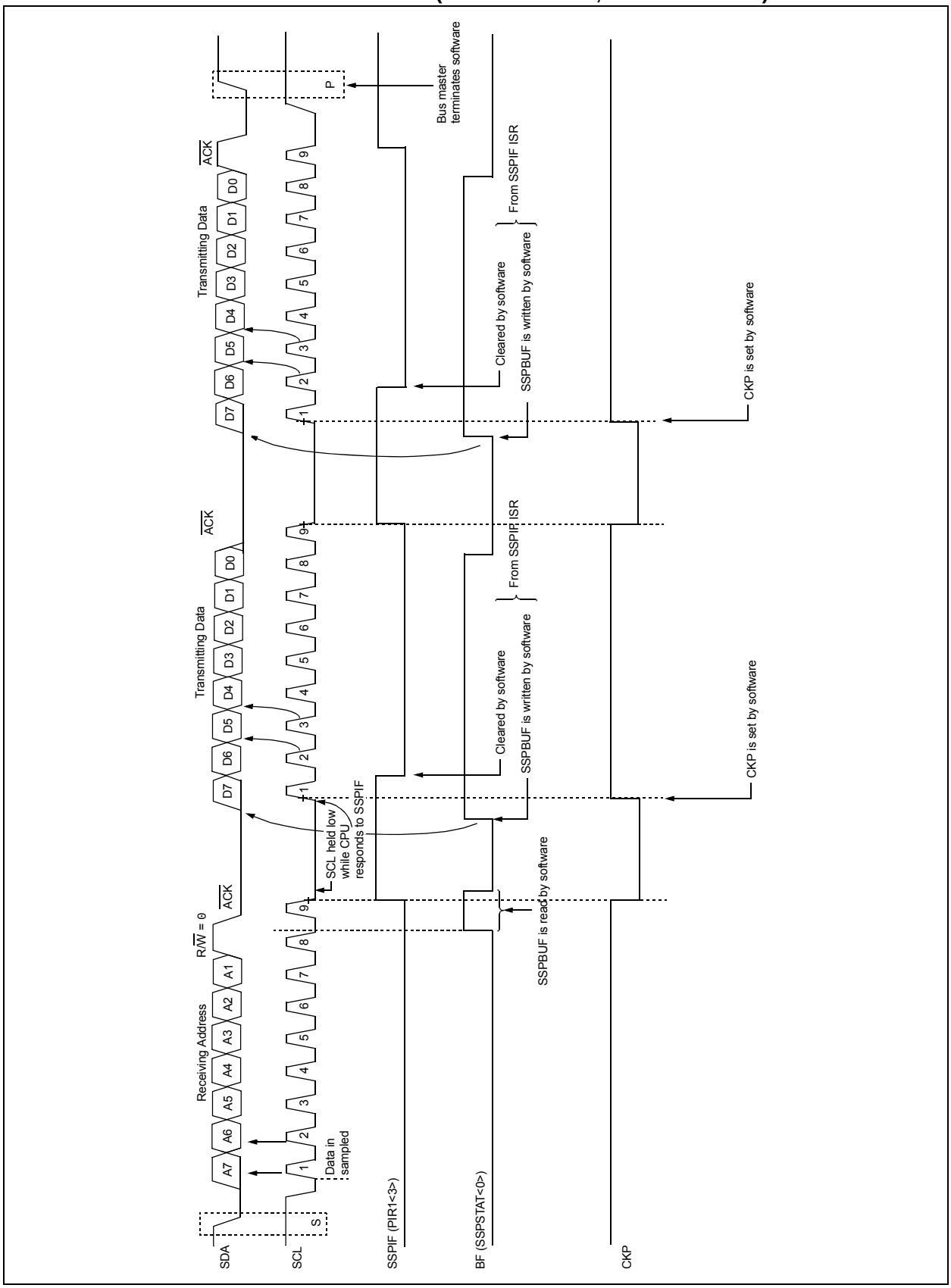
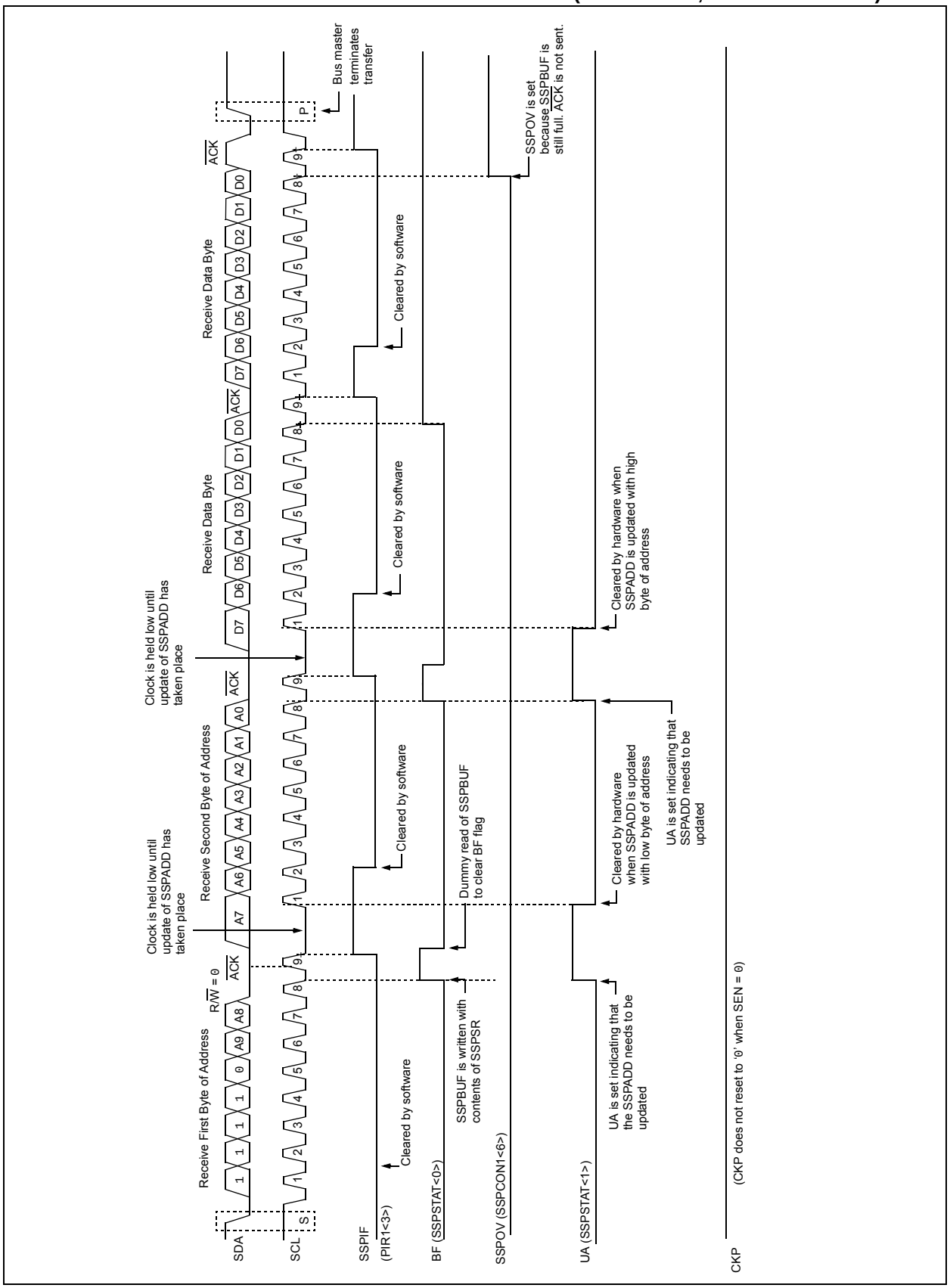


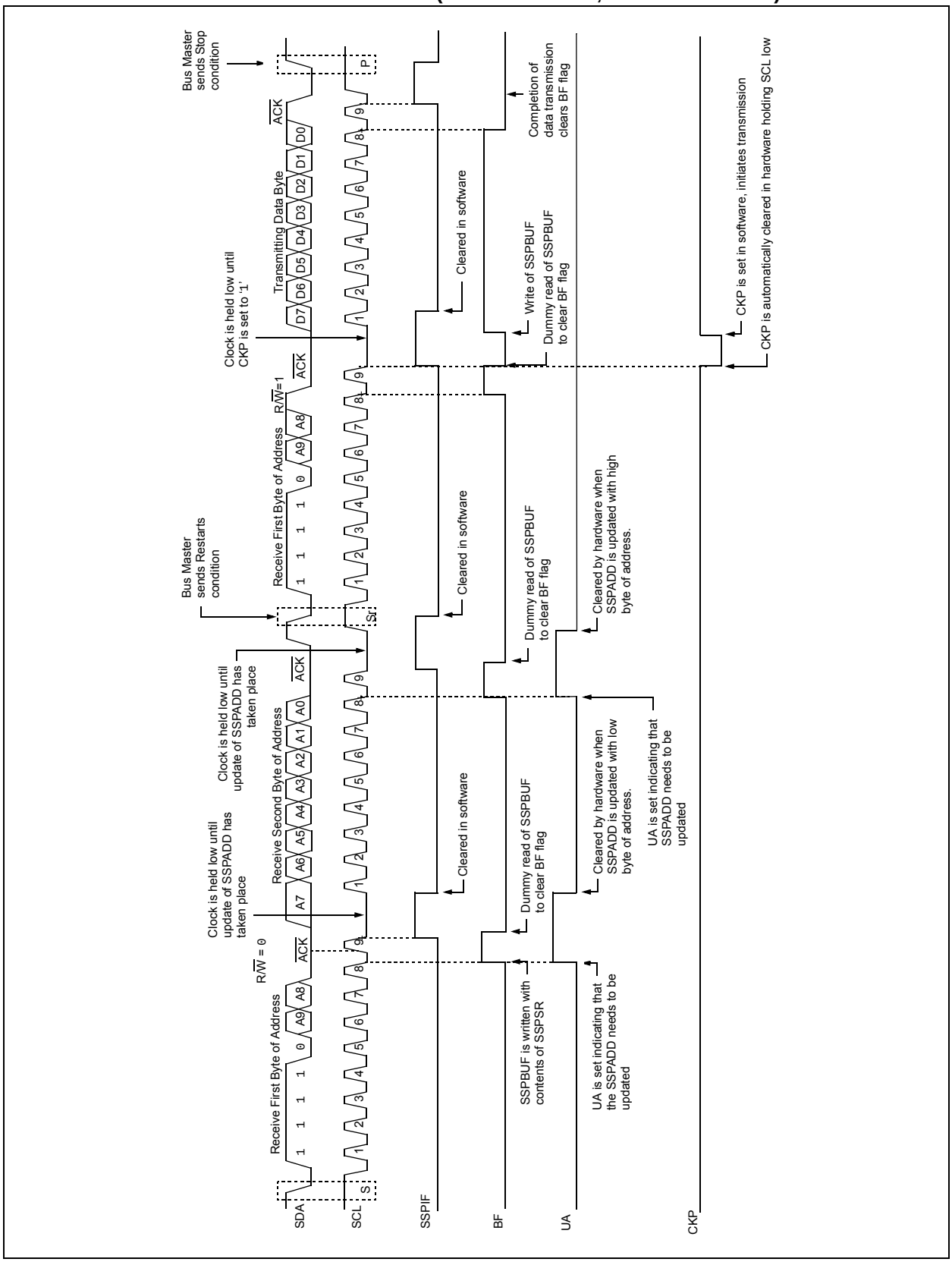
**FIGURE 14-10: I²C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)**





© 2009-2016 Microchip Technology Inc.

**FIGURE 14-11: I²C SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)**



#### 14.3.3.4 SSP Mask Register

An SSP Mask (SSPMSK) register is available in $I^2C$ 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 $I^2C$ 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 14-6: SSPMSK: SSP MASK 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 |
|---|---|---|---|---|---|---|---|
| MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0(1) |
| bit 7 | | | | | | | bit 0 |

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

bit 7-1 **MSK<7:1>:** Mask bits
1 = The received address bit n is compared to SSPADD<n> to detect $I^2C$ address match
0 = The received address bit n is not used to detect $I^2C$ address match

bit 0 **MSK<0>:** Mask bit for $I^2C$ Slave mode, 10-bit Address(1)
$I^2C$ Slave mode, 10-bit Address (SSPM<3:0> = 0111):
1 = The received address bit 0 is compared to SSPADD<0> to detect $I^2C$ address match
0 = The received address bit 0 is not used to detect $I^2C$ address match

**Note 1:** The MSK0 bit is used only in 10-bit Slave mode. In all other modes, this bit has no effect.


© 2009-2016 Microchip Technology Inc.

**REGISTER 14-7: SSPADD: MSSP ADDRESS AND BAUD RATE REGISTER (I²C MODE)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| 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 I²C 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."

### 14.3.4 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.

#### 14.3.4.1 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. This will prevent buffer overruns from occurring (see Figure 14-13).

**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.

#### 14.3.4.2 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.

#### 14.3.4.3 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 14-9).

**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.

#### 14.3.4.4 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 14-11).


© 2009-2016 Microchip Technology Inc.

#### 14.3.4.5 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 $I^2C$ master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the $I^2C$ bus have deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 14-12).

**FIGURE 14-12: CLOCK SYNCHRONIZATION TIMING**

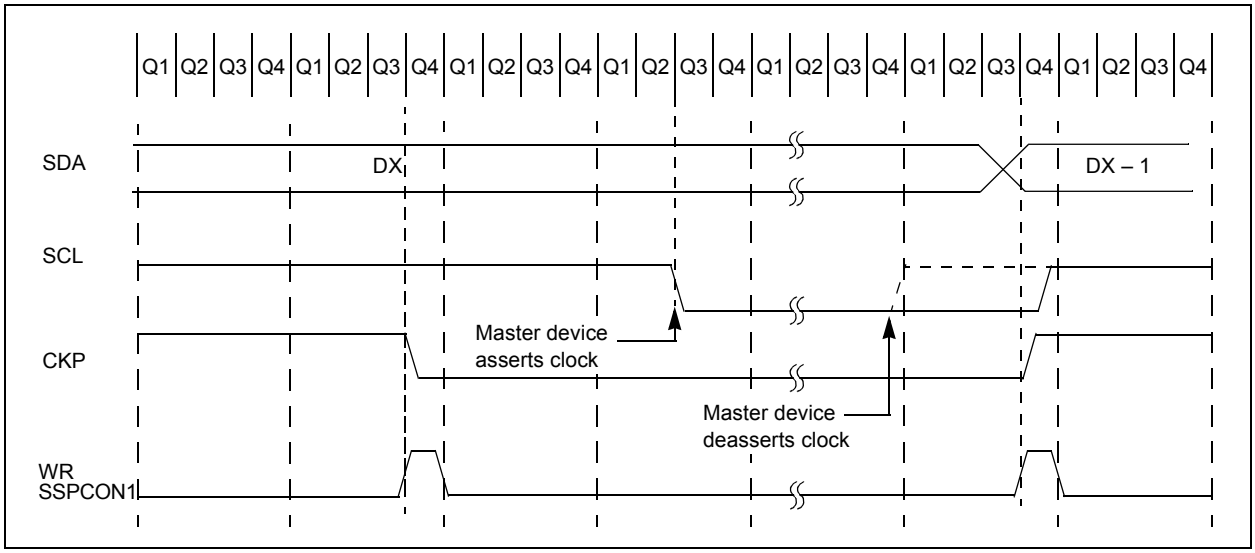
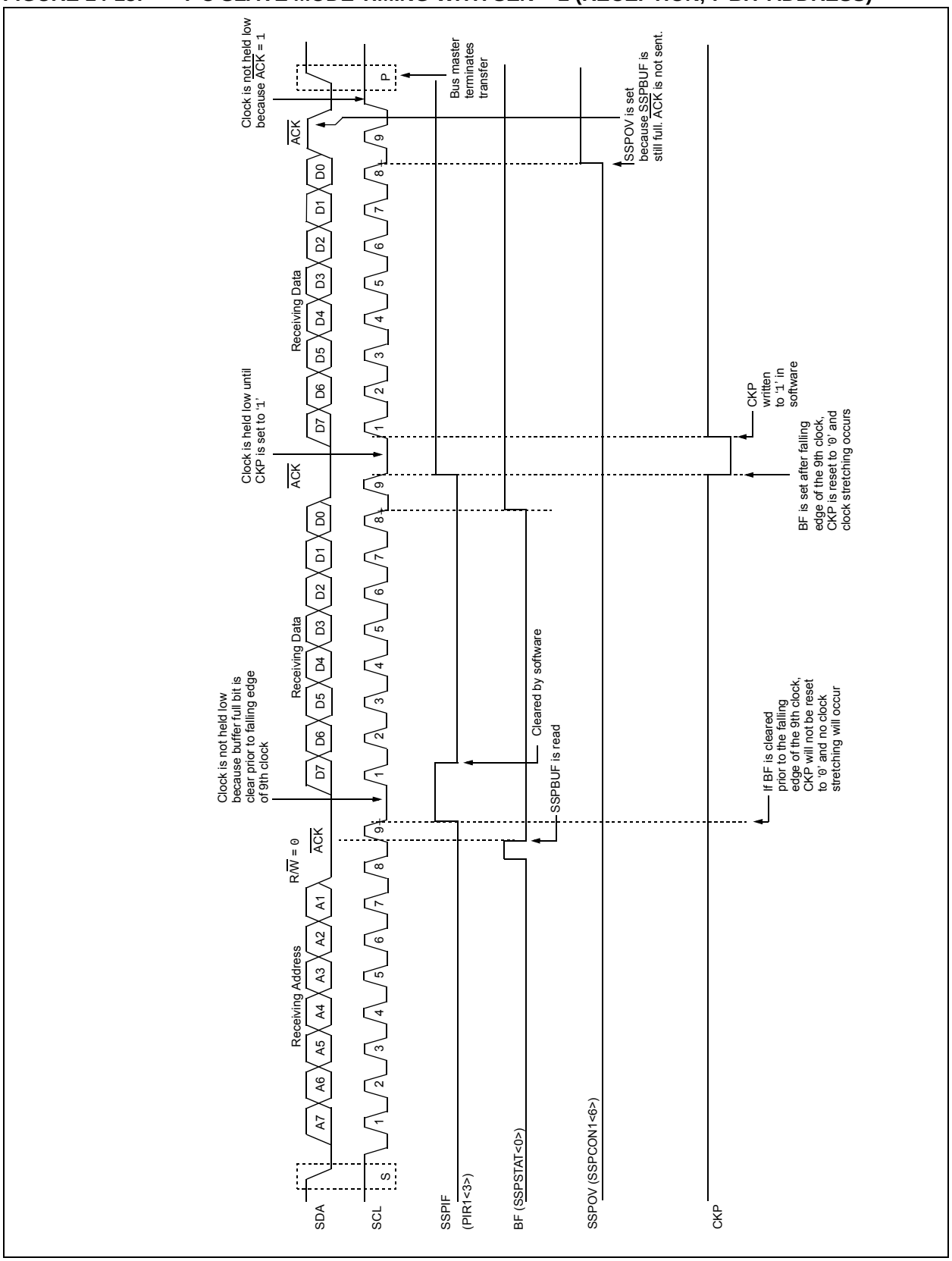


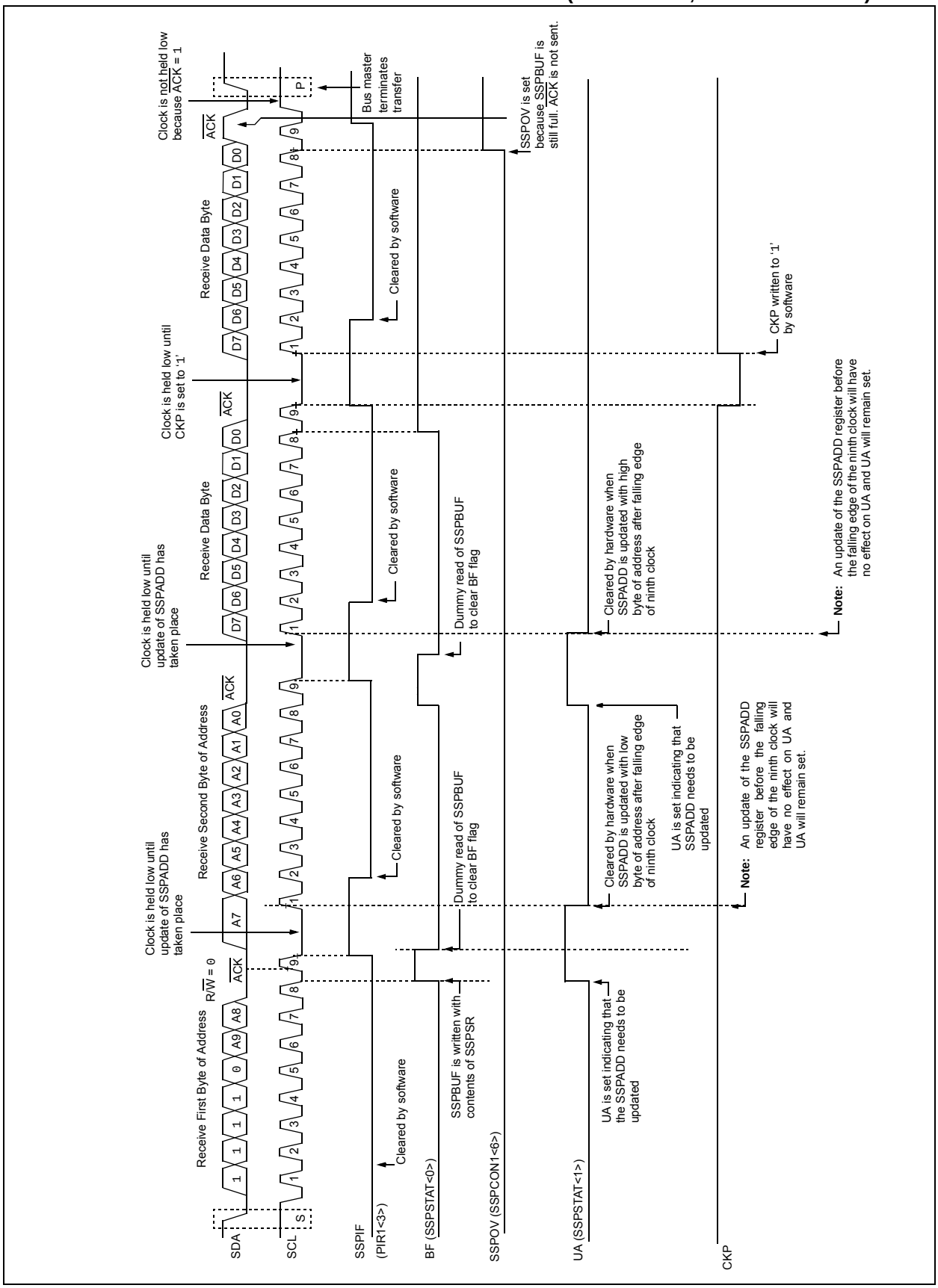
**FIGURE 14-13: I²C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)**





© 2009-2016 Microchip Technology Inc.

**FIGURE 14-14: I²C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESS)**



### 14.3.5 GENERAL CALL ADDRESS SUPPORT

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

The general call address is one of eight addresses reserved for specific purposes by the $I^2C$ protocol. It consists of all '0's with R/$\overline{W}$ = 0.

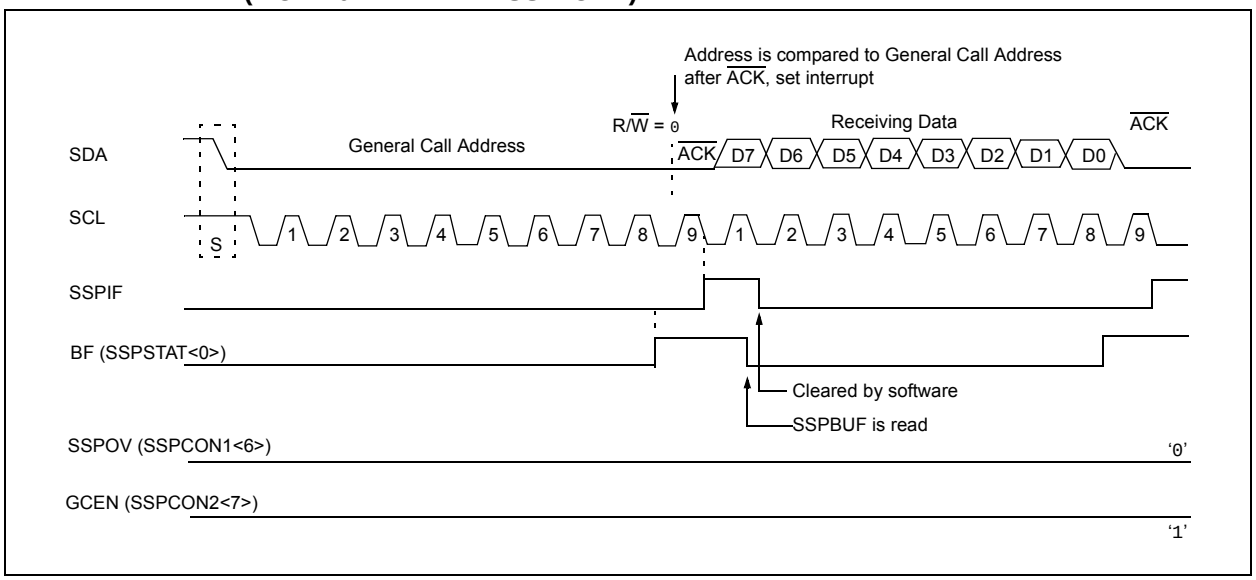
The general call address is recognized when the GCEN bit of the SSPCON2 is set. Following a Start bit detect, eight bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

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

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

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match and the UA bit 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 14-15).

**FIGURE 14-15: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESS MODE)**





© 2009-2016 Microchip Technology Inc.

### 14.3.6 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 $I^2C$ bus may be taken when the P bit is set, or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all $I^2C$ bus operations based on Start and Stop bit conditions.

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

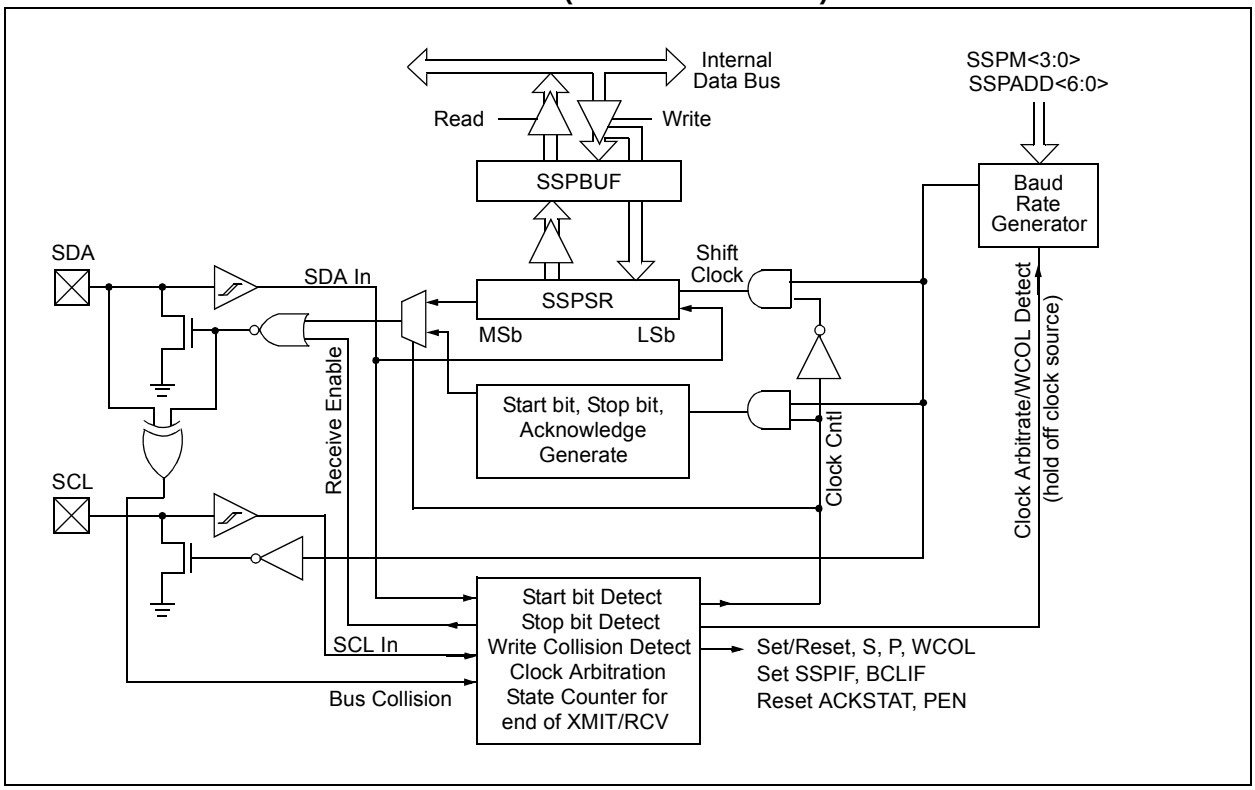
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the $I^2C$ port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate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

**Note:** The MSSP module, when configured in $I^2C$ Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

The following events will cause the 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 14-16: MSSP BLOCK DIAGRAM ($I^2C$ MASTER MODE)**



#### 14.3.6.1 I²C Master Mode Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (seven bits) and the Read/Write (R/$\overline{W}$) bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

A Baud Rate Generator is used to set the clock frequency output on SCL. See **Section 14.3.7 "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 $\overline{ACK}$ 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 $\overline{ACK}$ 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.


© 2009-2016 Microchip Technology Inc.

### 14.3.7 BAUD RATE

In I²C Master mode, the Baud Rate Generator (BRG) reload value is placed in the SSPADD register (Figure 14-17). 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 14-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

**EQUATION 14-1: CLOCK RATES**

$$F_{SCL} = \frac{F_{OSC}}{(SSPADD + 1)(4)}$$

**FIGURE 14-17: BAUD RATE GENERATOR BLOCK DIAGRAM**

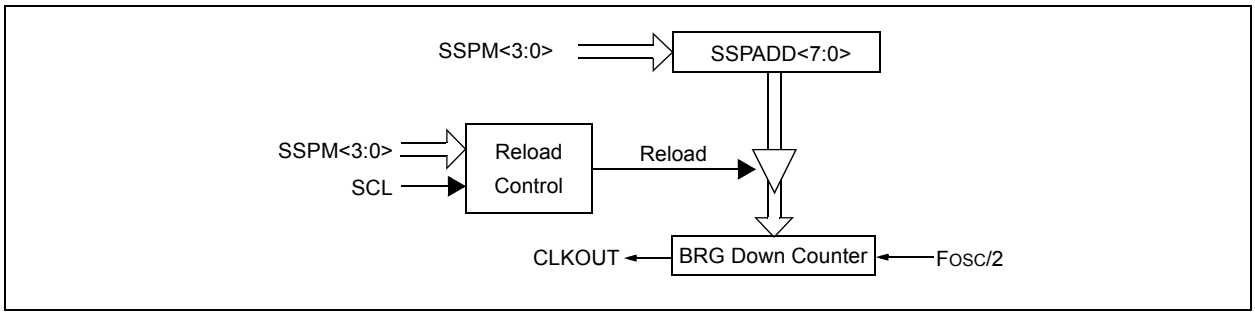



**TABLE 14-3: I²C CLOCK RATE W/BRG**

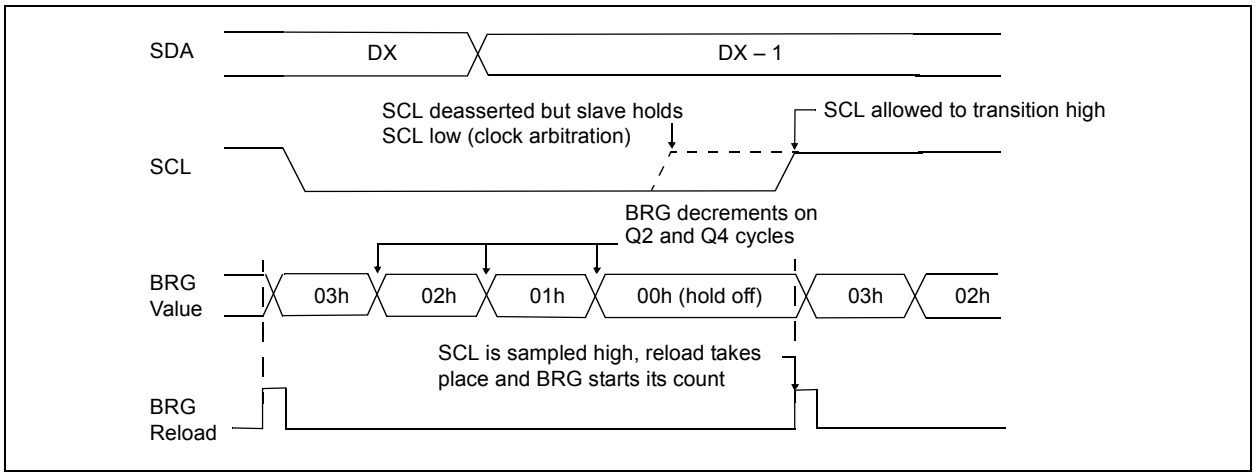
| FOSC | FCY | BRG Value | FSCL (2 Rollovers of BRG) |
|---|---|---|---|
| 48 MHz | 12 MHz | 0Bh | 1 MHz(1) |
| 48 MHz | 12 MHz | 1Dh | 400 kHz |
| 48 MHz | 12 MHz | 77h | 100 kHz |
| 40 MHz | 10 MHz | 18h | 400 kHz(1) |
| 40 MHz | 10 MHz | 1Fh | 312.5 kHz |
| 40 MHz | 10 MHz | 63h | 100 kHz |
| 16 MHz | 4 MHz | 09h | 400 kHz(1) |
| 16 MHz | 4 MHz | 0Ch | 308 kHz |
| 16 MHz | 4 MHz | 27h | 100 kHz |
| 4 MHz | 1 MHz | 02h | 333 kHz(1) |
| 4 MHz | 1 MHz | 09h | 100 kHz |
| 4 MHz | 1 MHz | 00h | 1 MHz(1) |

**Note 1:** The I²C interface does not conform to the 400 kHz I²C 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.

#### 14.3.7.1 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 14-18).

**FIGURE 14-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**





© 2009-2016 Microchip Technology Inc.

### 14.3.8 I²C 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.

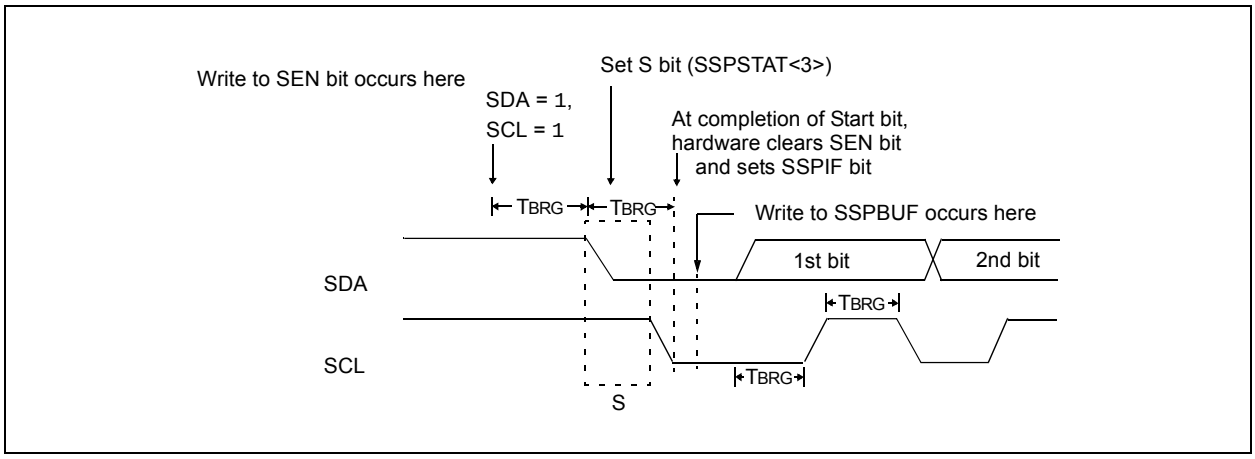
| **Note:** | If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state. |
|---|---|

#### 14.3.8.1 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 queuing of events is not allowed, writing to the lower five bits of SSPCON2 is disabled until the Start condition is complete. |
|---|---|

**FIGURE 14-19: FIRST START BIT TIMING**



### 14.3.9 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit of the SSPCON2 register is programmed high and the I²C logic module is in the Idle state. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded 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'.

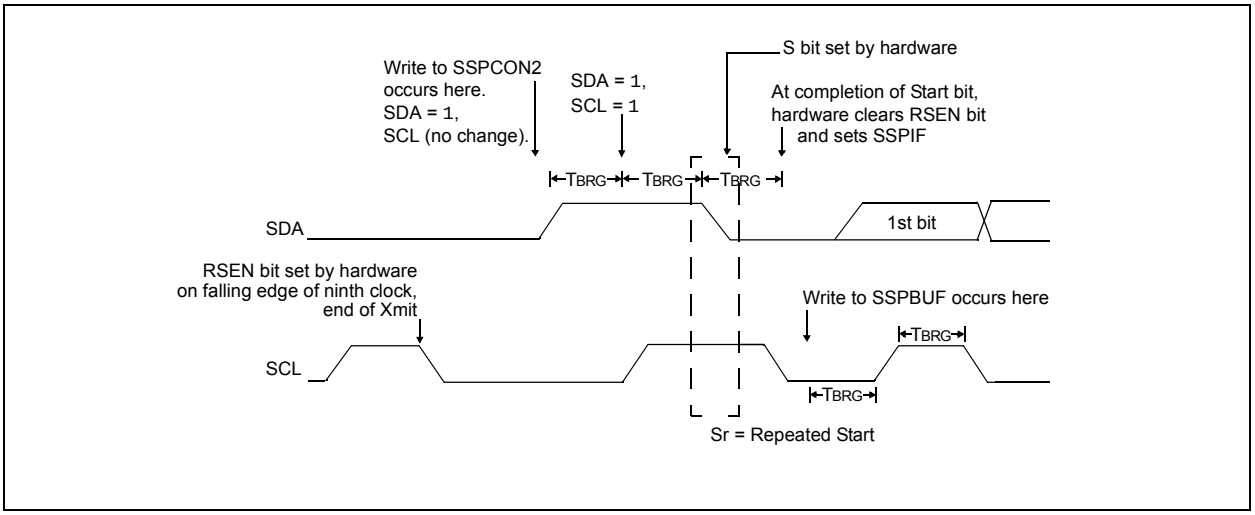
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode or the default first address in 10-bit mode. After the first eight bits are transmitted and an $\overline{ACK}$ is received, the user may then transmit an additional eight bits of address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

#### 14.3.9.1 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 14-20: REPEAT START CONDITION WAVEFORM**





© 2009-2016 Microchip Technology Inc.

### 14.3.10 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPBUF register. This action will set the Buffer Full flag bit, BF, and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted (see data hold time specification parameter 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 14-21).

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.

#### 14.3.10.1 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.

#### 14.3.10.2 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.

#### 14.3.10.3 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.

### 14.3.11 I²C 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.

#### 14.3.11.1 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.

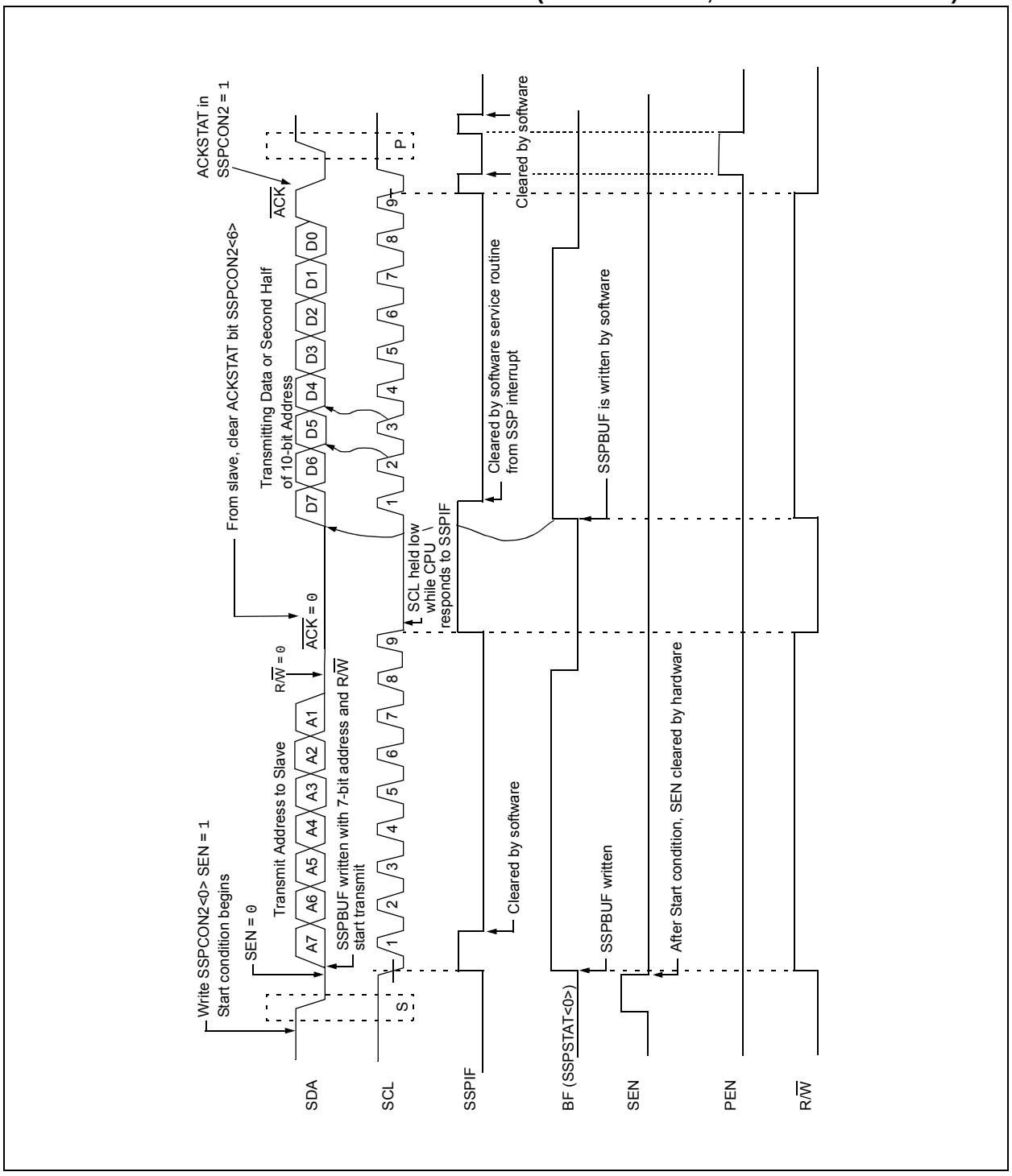
#### 14.3.11.2 SSPOV Status Flag

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.

#### 14.3.11.3 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).

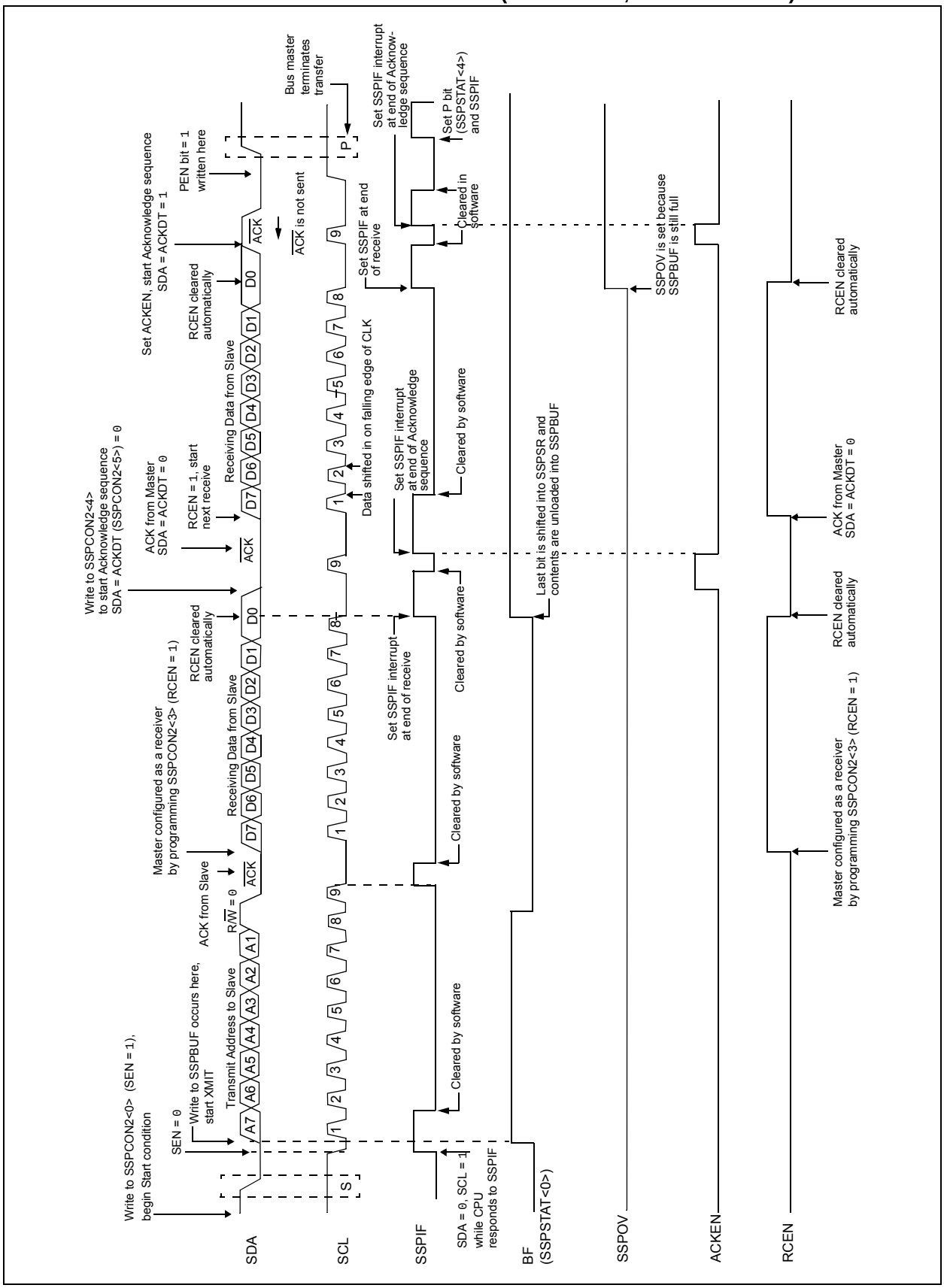
**FIGURE 14-21: I²C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)**





© 2009-2016 Microchip Technology Inc.

**FIGURE 14-22: I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)**



### 14.3.12 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 14-23).

#### 14.3.12.1 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).

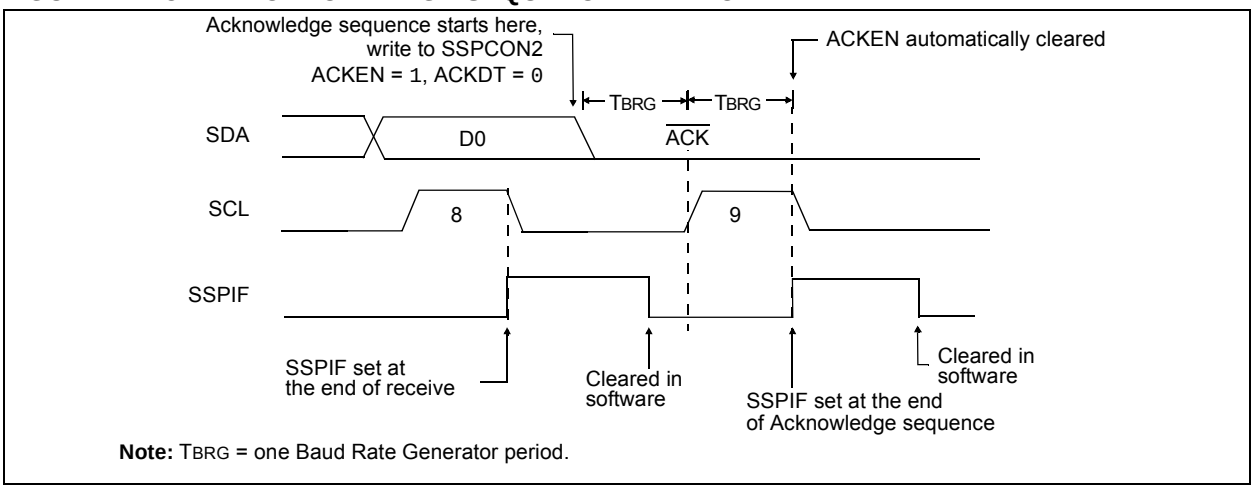
### 14.3.13 STOP CONDITION TIMING

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 14-24).

#### 14.3.13.1 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).

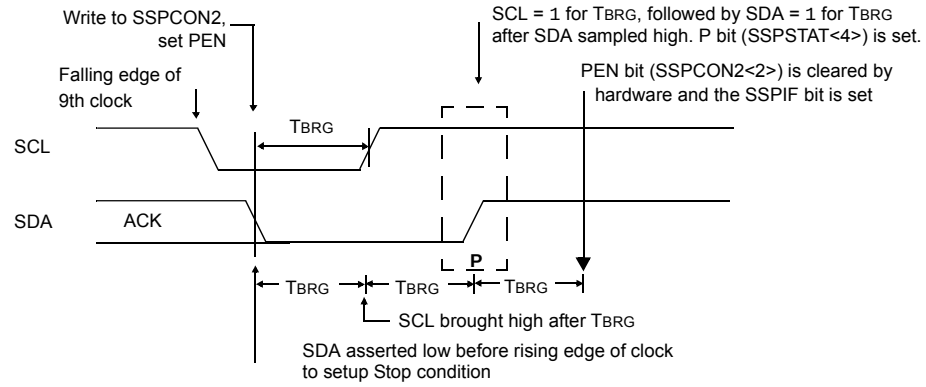
**FIGURE 14-23: ACKNOWLEDGE SEQUENCE WAVEFORM**




**Note:** TBRG = one Baud Rate Generator period.


© 2009-2016 Microchip Technology Inc.

**FIGURE 14-24: STOP CONDITION RECEIVE OR TRANSMIT MODE**




**Note:** TBRG = one Baud Rate Generator period.

### 14.3.14 SLEEP OPERATION

While in Sleep mode, the I$^2$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).

### 14.3.15 EFFECTS OF A RESET

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

### 14.3.16 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$^2$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

### 14.3.17 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$^2$C port to its Idle state (Figure 14-25).

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$^2$C bus is free, the user can resume communication by asserting a Start condition.

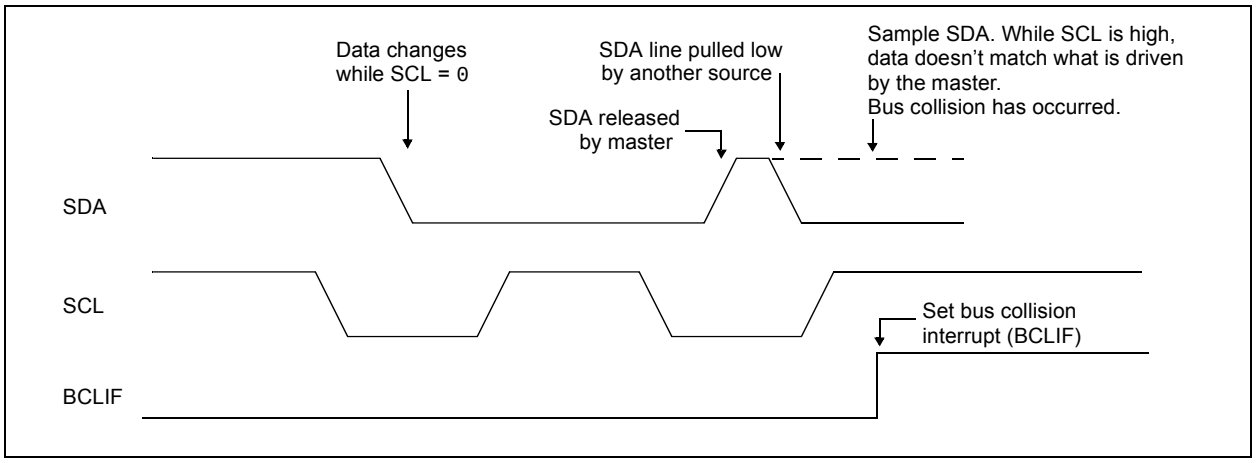
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I$^2$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$^2$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 14-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**





© 2009-2016 Microchip Technology Inc.

#### 14.3.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the Start condition (Figure 14-26).

b) SCL is sampled low before SDA is asserted low (Figure 14-27).

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 14-26).

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 14-28). 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 14-26: BUS COLLISION DURING START CONDITION (SDA ONLY)**

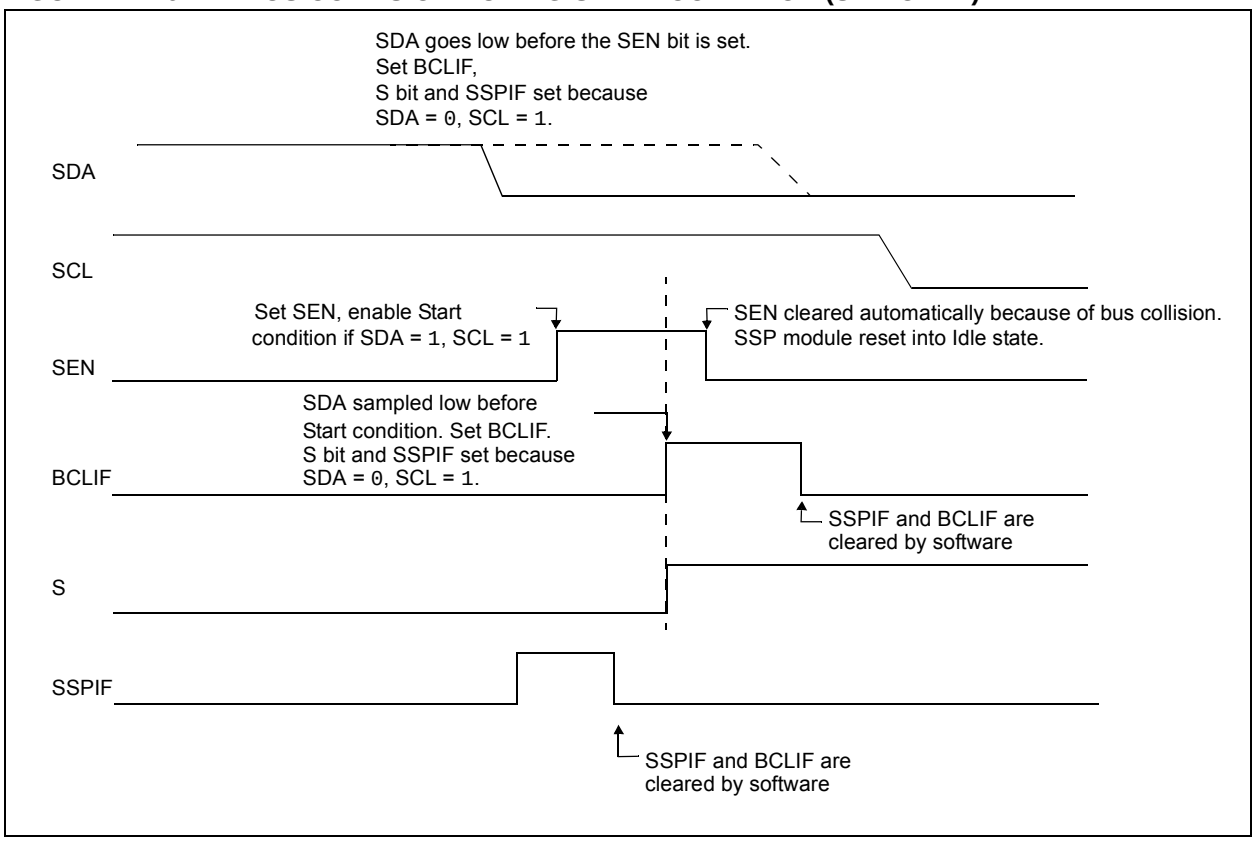


**FIGURE 14-27: BUS COLLISION DURING START CONDITION (SCL = 0)**

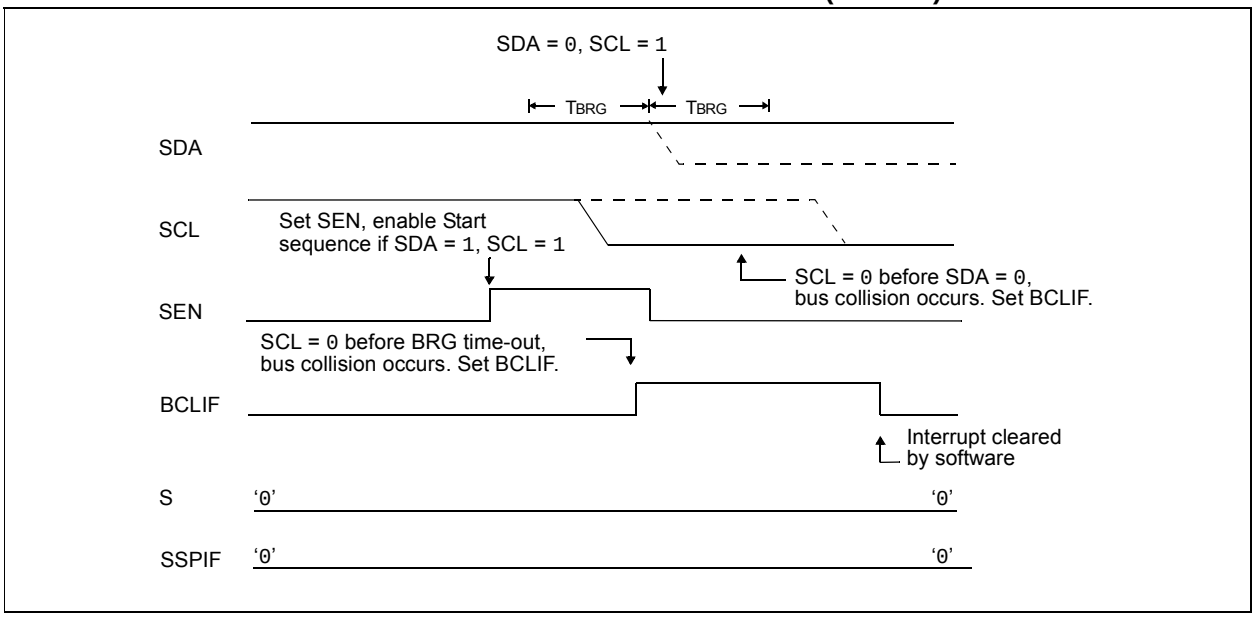
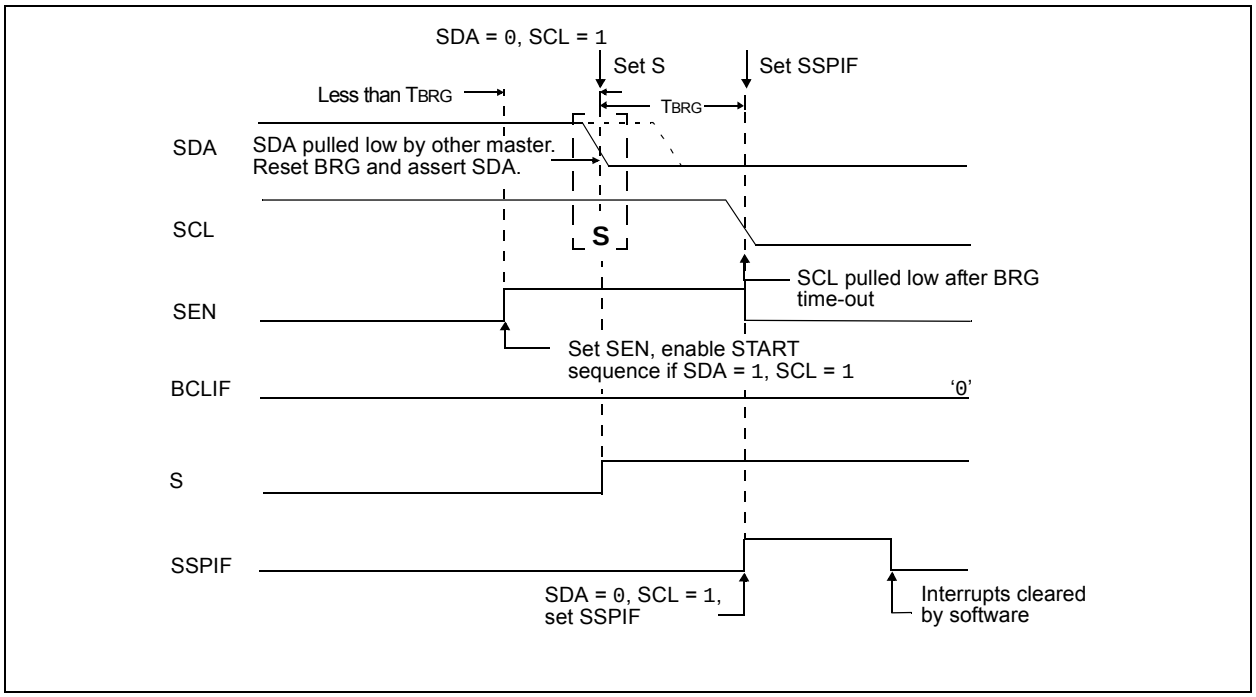



**FIGURE 14-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION**





© 2009-2016 Microchip Technology Inc.

#### 14.3.17.2 Bus Collision During a Repeated Start Condition

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

a) A low level is sampled on SDA when SCL goes from low level to high level.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD 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 14-29). 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 14-30.

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 14-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**

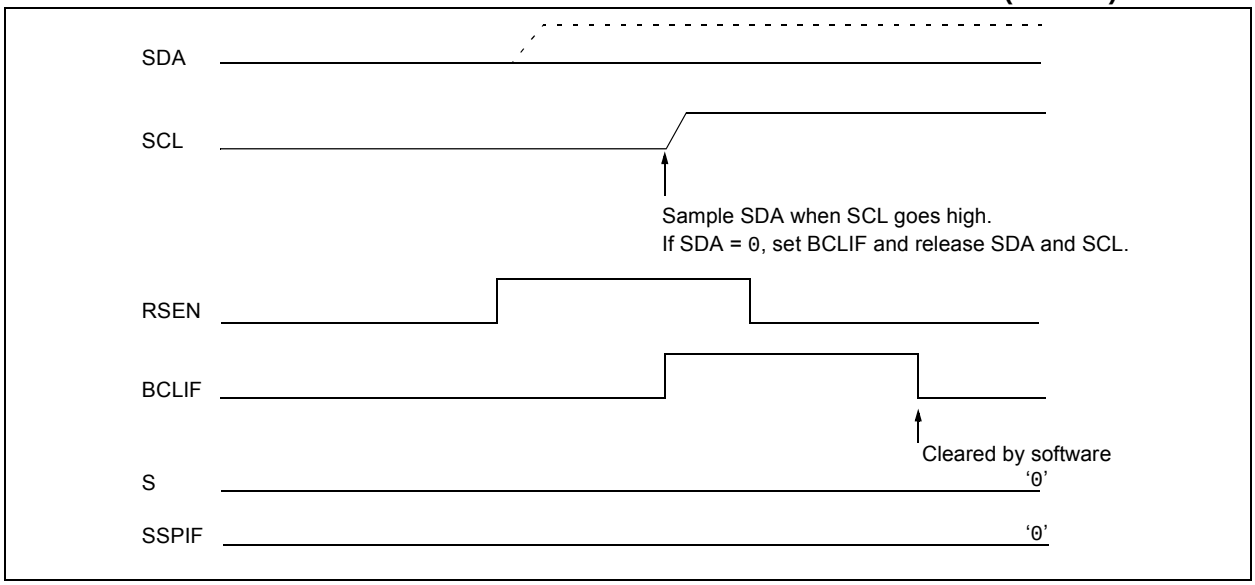
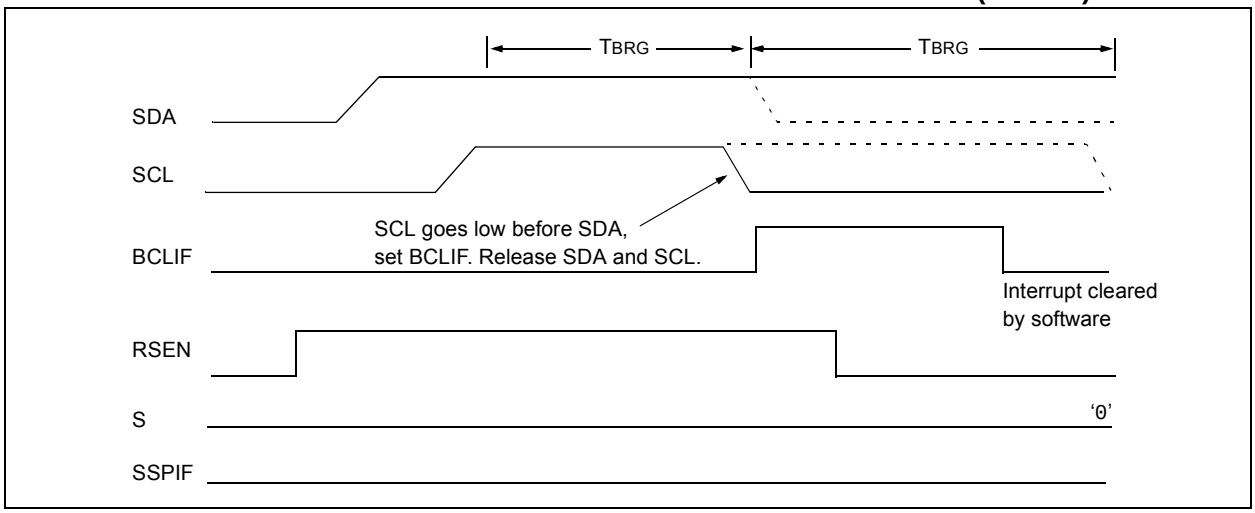



**FIGURE 14-30: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)**



#### 14.3.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD 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 14-31). 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 14-32).

**FIGURE 14-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

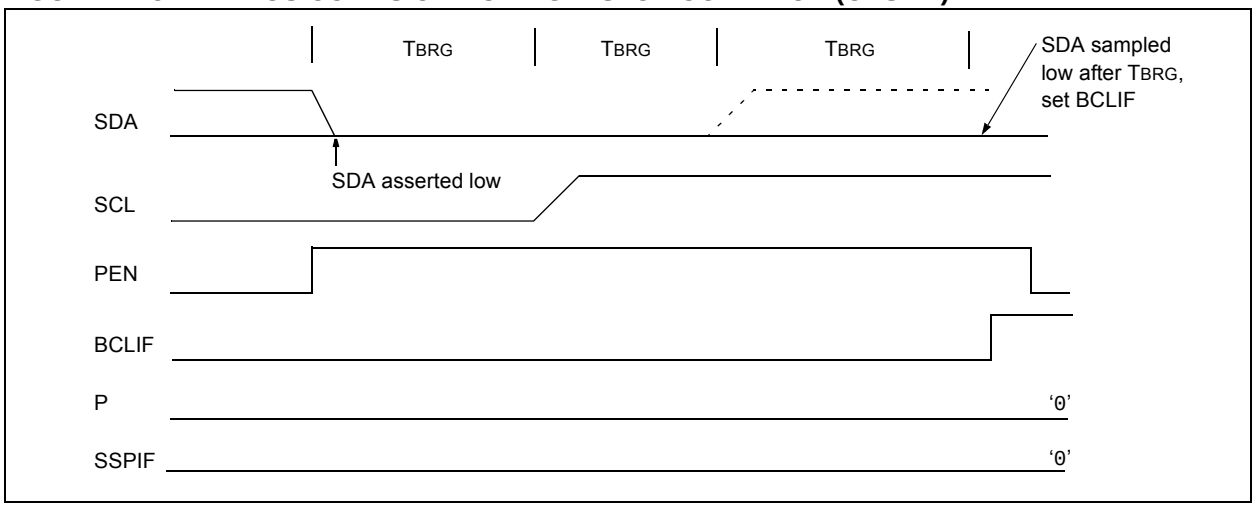
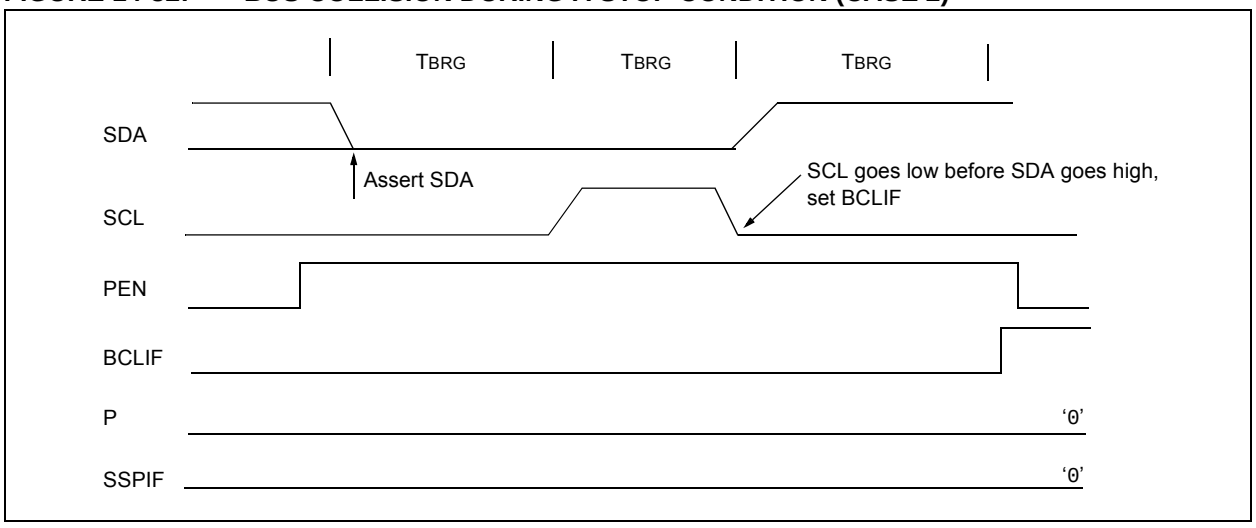



**FIGURE 14-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)**





© 2009-2016 Microchip Technology Inc.

**TABLE 14-4: SUMMARY OF REGISTERS ASSOCIATED WITH $I^2C$**

| 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 | 248 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | — | TMR3IP | — | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | — | TMR3IE | — | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | — | TMR3IF | — | 248 |
| SSPADD | SSP Address Register in $I^2C$ Slave Mode. SSP Baud Rate Reload Register in $I^2C$ Master Mode. | | | | | | | | 246 |
| SSPBUF | SSP Receive Buffer/Transmit Register | | | | | | | | 246 |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 246 |
| SSPCON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 246 |
| SSPMSK | MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0 | 248 |
| SSPSTAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 246 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | — | — | — | — | 248 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by $I^2C$.

## 15.0 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:

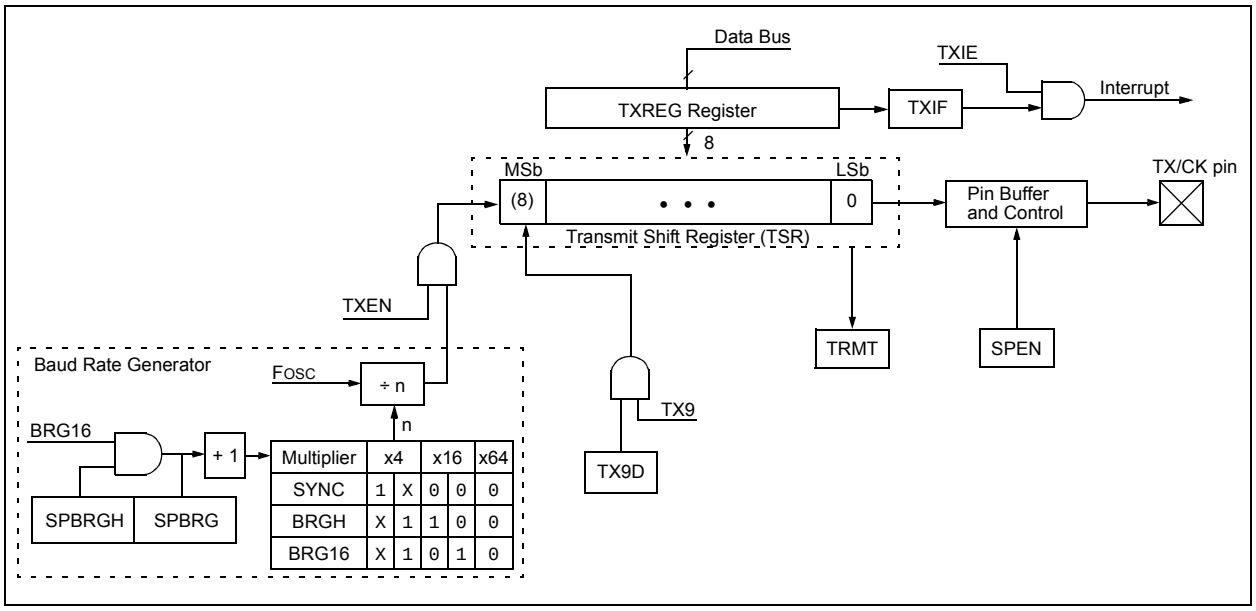
- Full-duplex asynchronous transmit and receive
- Two-character input buffer
- One-character output buffer
- Programmable 8-bit or 9-bit character length
- Address detection in 9-bit mode
- Input buffer overrun error detection
- Received character framing error detection
- Half-duplex synchronous master
- 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 15-1 and Figure 15-2.

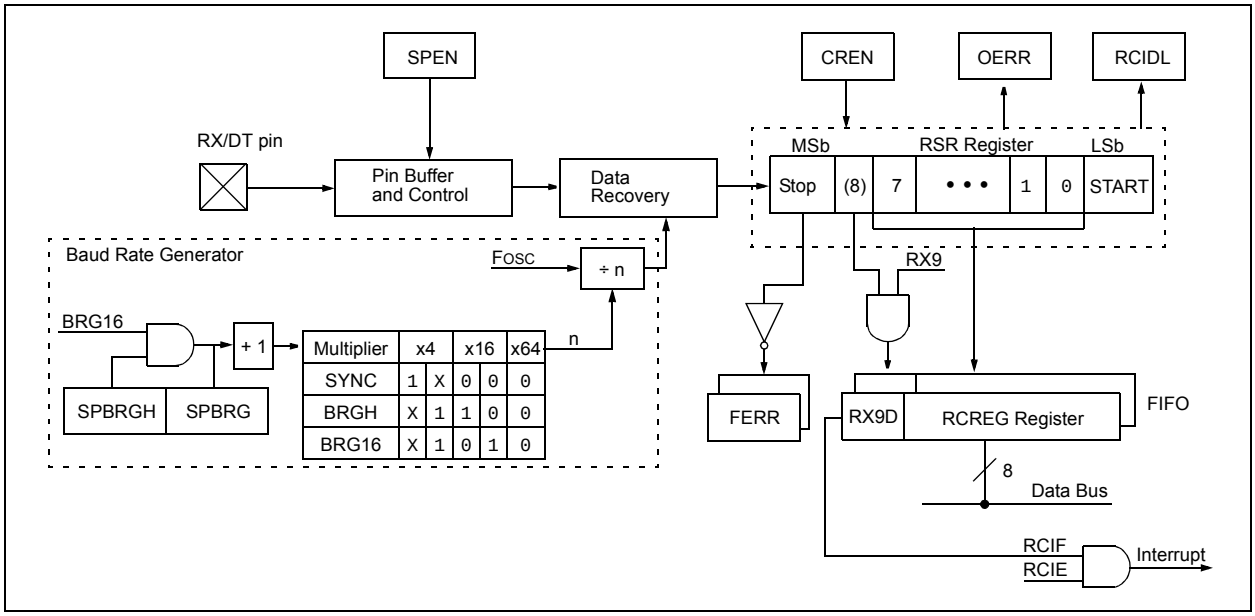
**FIGURE 15-1: EUSART TRANSMIT BLOCK DIAGRAM**





© 2009-2016 Microchip Technology Inc.

**FIGURE 15-2: 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 15-1, Register 15-2 and Register 15-3, 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.

## 15.1 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 15-5 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.

### 15.1.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 15-1. 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.

#### 15.1.1.1 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.

#### 15.1.1.2 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.

#### 15.1.1.3 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.

#### 15.1.1.4 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.


© 2009-2016 Microchip Technology Inc.

#### 15.1.1.5 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. |
|---|---|

#### 15.1.1.6 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 15.1.2.8 "Address Detection"** for more information on the Address mode.

#### 15.1.1.7 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 15.3 "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 INTCON 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 15-3: ASYNCHRONOUS TRANSMISSION**

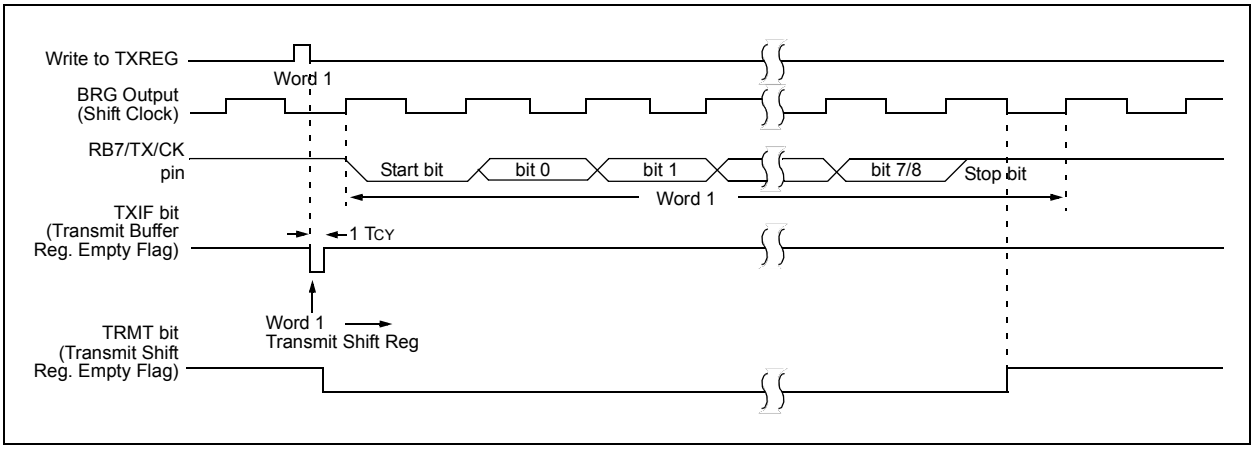
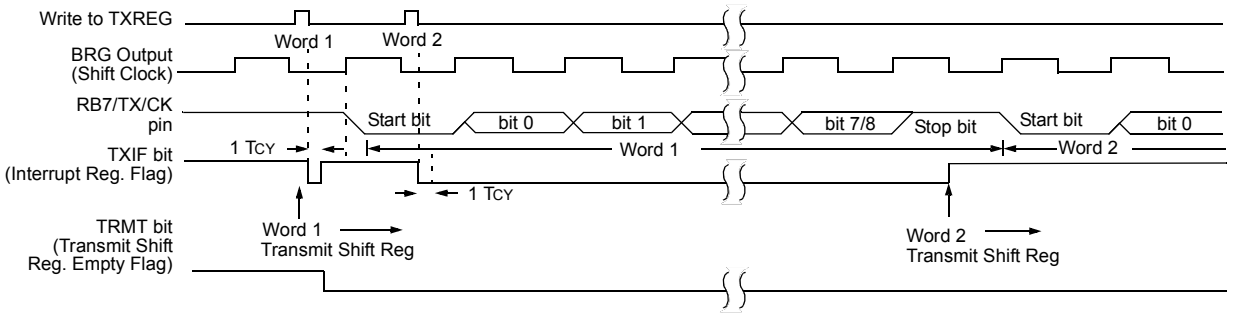


**FIGURE 15-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**




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

**TABLE 15-1: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | DTRXP | CKTXP | BRG16 | — | WUE | ABDEN | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 247 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 247 |
| SPBRGH | EUSART Baud Rate Generator Register, High Byte | | | | | | | | 247 |
| TXREG | EUSART Transmit Register | | | | | | | | 247 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 247 |

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


© 2009-2016 Microchip Technology Inc.

### 15.1.2 EUSART ASYNCHRONOUS RECEIVER

The Asynchronous mode would typically be used in RS-232 systems. The receiver block diagram is shown in Figure 15-2. 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.

#### 15.1.2.1 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. |

#### 15.1.2.2 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 15.1.2.5 "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 15.1.2.6 "Receive Overrun Error"** for more information on overrun errors. |

#### 15.1.2.3 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.

#### 15.1.2.4 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.

#### 15.1.2.5 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. |
|---|---|

#### 15.1.2.6 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.

#### 15.1.2.7 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 nine 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 eight Least Significant bits from the RCREG.

#### 15.1.2.8 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.


© 2009-2016 Microchip Technology Inc.

#### 15.1.2.9 Asynchronous Reception Set-up

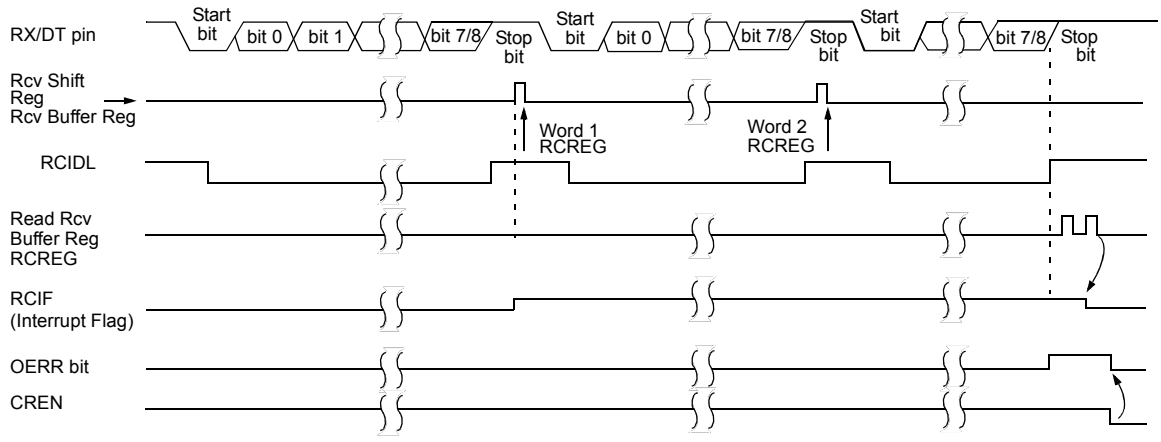
1. Initialize the SPBRGH:SPBRG register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 15.3 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the serial port by setting the SPEN bit and the RX/DT pin TRIS bit. The SYNC bit must be clear for asynchronous operation.
3. 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.

#### 15.1.2.10 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 15.3 "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 15-5: ASYNCHRONOUS RECEPTION**




**Note:** This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.

**TABLE 15-2: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | DTRXP | CKTXP | BRG16 | — | WUE | ABDEN | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| RCREG | EUSART Receive Register | | | | | | | | 247 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 247 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 247 |
| SPBRGH | EUSART Baud Rate Generator Register, High Byte | | | | | | | | 247 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 248 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 247 |

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


© 2009-2016 Microchip Technology Inc.

## 15.2 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (HFINTOSC). However, the HFINTOSC frequency may drift as VDD 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 2.7.1 "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 15.3.1 "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 15-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-1 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CSRC | TX9 | TXEN(1) | SYNC | SENDB | BRGH | TRMT | TX9D |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit enabled
0 = Transmit disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed
Synchronous mode:
Don't care

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** Ninth bit of Transmit Data
Can be address/data bit or a parity bit.

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

**REGISTER 15-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-0 | R-x |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **SPEN:** Serial Port Enable bit

1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-bit Receive Enable bit

1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit

Asynchronous mode:

Don't care

Synchronous mode – Master:

1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.

Synchronous mode – Slave

Don't care

bit 4 **CREN:** Continuous Receive Enable bit

Asynchronous mode:

1 = Enables receiver
0 = Disables receiver

Synchronous mode:

1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit

Asynchronous mode 9-bit (RX9 = 1):

1 = Enables address detection, 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

Asynchronous mode 8-bit (RX9 = 0):

Don't care

bit 2 **FERR:** Framing Error bit

1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit

1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** Ninth bit of Received Data

This can be address/data bit or a parity bit and must be calculated by user firmware.


© 2009-2016 Microchip Technology Inc.

**REGISTER 15-3: 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'

bit 1 **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

## 15.3 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 15-3 contains the formulas for determining the baud rate. Example 15-1 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 15-5. 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 15-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$Desired\ Baud\ Rate = \frac{FOSC}{64([SPBRGH:SPBRG] + 1)}$$

Solving for SPBRGH:SPBRG:

$$X = \left(\frac{FOSC}{64^* (\text{Desired Baud Rate})}\right)^{-1}$$

$$= \left(\frac{16{,}000{,}000}{64^* 9600}\right)^{-1}$$

$$= [25.042] = 25$$

$$Calculated\ Baud\ Rate = \frac{16000000}{64(25 + 1)}$$

$$= 9615$$

$$Error = \frac{Calc.\ Baud\ Rate - Desired\ Baud\ Rate}{Desired\ Baud\ Rate}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

### TABLE 15-3: 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 15-4: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | DTRXP | CKTXP | BRG16 | — | WUE | ABDEN | 247 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 247 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 247 |
| SPBRGH | EUSART Baud Rate Generator Register, High Byte | | | | | | | | 247 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 247 |

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


© 2009-2016 Microchip Technology Inc.

**TABLE 15-5: 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 |

**TABLE 15-5: 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 | — | — | — |


© 2009-2016 Microchip Technology Inc.

**TABLE 15-5: 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 | — | — | — |

### 15.3.1 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 15-6). 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 15-6. 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 15-6. 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 15.3.3 “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.

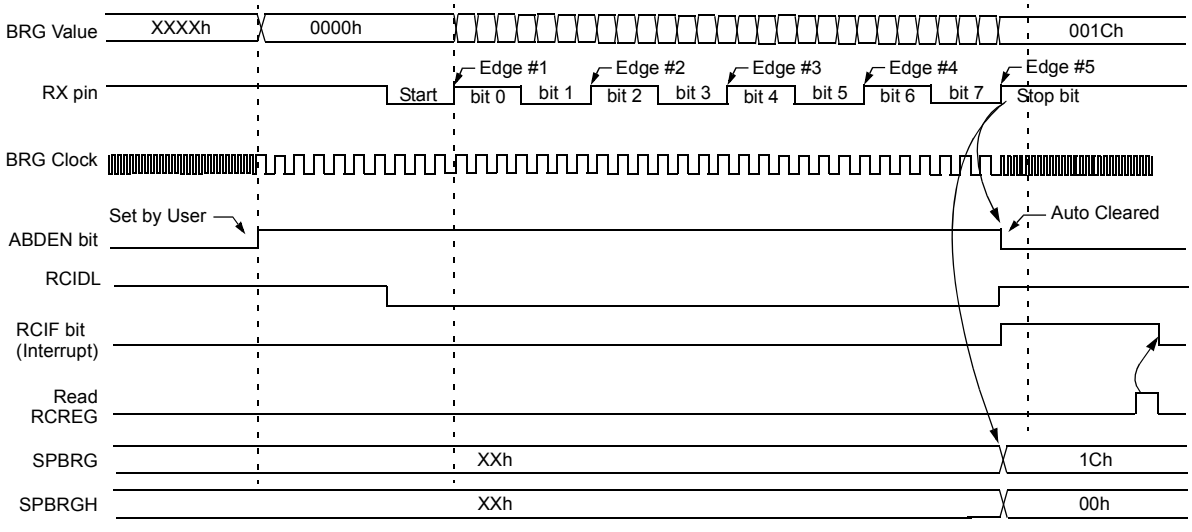
**3:** During the auto-baud process, the auto-baud counter starts counting at 1. Upon completion of the auto-baud sequence, to achieve maximum accuracy, subtract 1 from the SPBRGH:SPBRG register pair.

**TABLE 15-6: 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 15-6: AUTOMATIC BAUD RATE CALIBRATION**




**Note 1:** The ABD sequence requires the **EUSART** module to be configured in Asynchronous mode.


© 2009-2016 Microchip Technology Inc.

### 15.3.2 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.

### 15.3.3 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 15-7), and asynchronously if the device is in Sleep mode (Figure 15-8). 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.

#### 15.3.3.1 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 Start-up Time

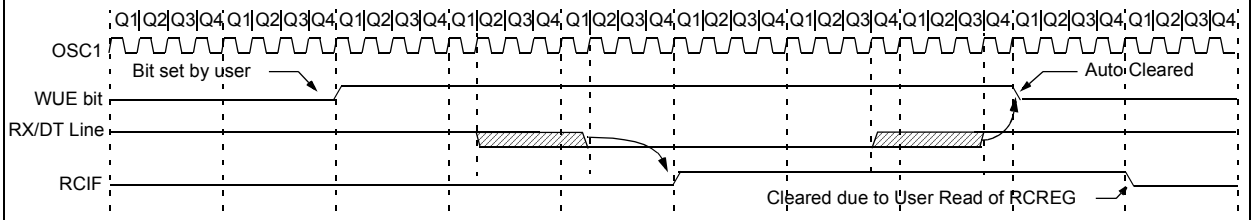
Oscillator start-up time must be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL mode). The Sync Break (or wake-up signal) character must be of sufficient length, and be followed by a sufficient interval, to allow enough time for the selected 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.

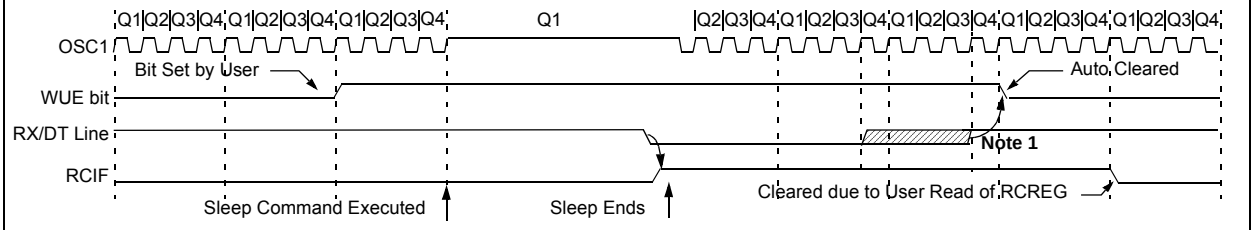
To ensure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process before setting the WUE bit. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**FIGURE 15-7: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION**




**Note 1:** The **EUSART** remains in Idle while the WUE bit is set.

**FIGURE 15-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**




**Note 1:** If the wake-up event requires long oscillator warm-up time, the automatic clearing of the WUE bit can occur while the `stposc` signal is still active. This sequence should not depend on the presence of Q clocks.

**2:** The **EUSART** remains in Idle while the WUE bit is set.


© 2009-2016 Microchip Technology Inc.

### 15.3.4 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 15-9 for the timing of the Break character sequence.

#### 15.3.4.1 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.

### 15.3.5 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.

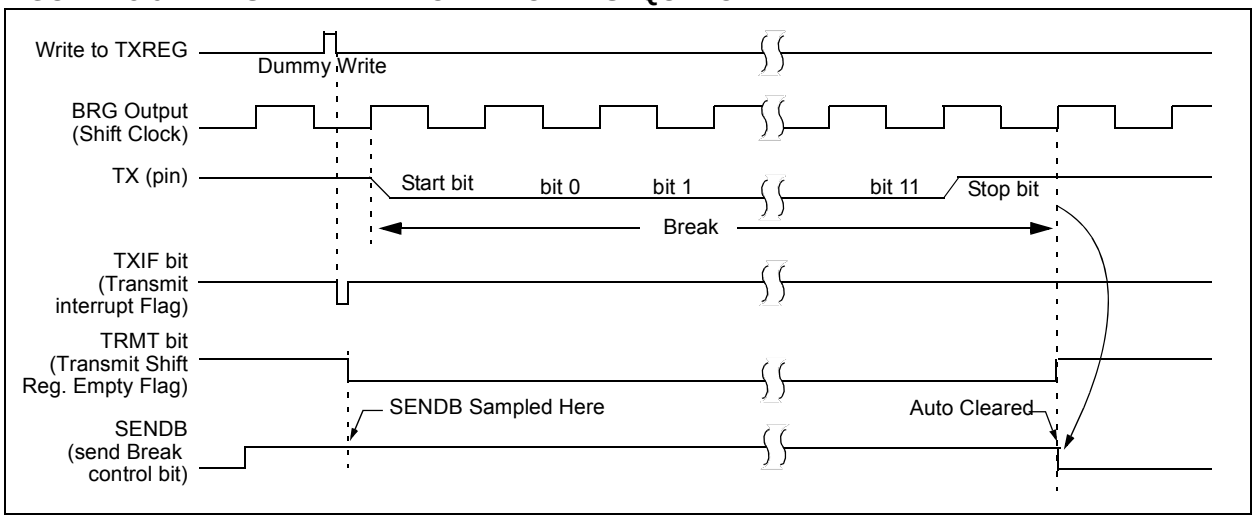
A Break character has been received when;

- RCIF bit is set
- FERR bit is set
- RCREG = 00h

The second method uses the Auto-Wake-up feature described in **Section 15.3.3 "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 15-9: SEND BREAK CHARACTER SEQUENCE**



## 15.4 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.

### 15.4.1 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.

#### 15.4.1.1 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.

#### 15.4.1.2 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.

#### 15.4.1.3 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. |
|---|---|

#### 15.4.1.4 Data Polarity

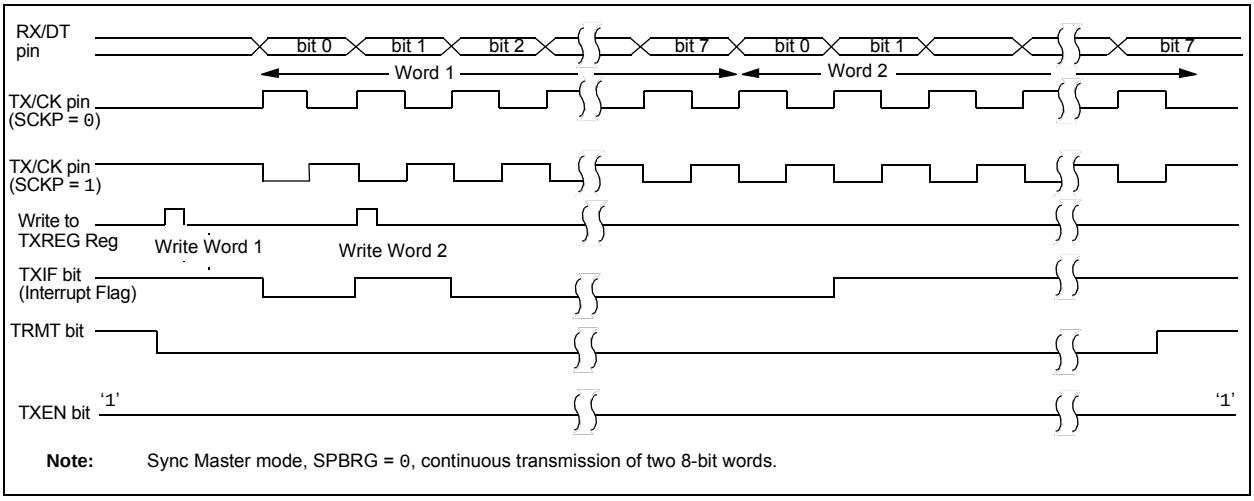
The polarity of the transmit and receive data can be controlled with the DTRXP bit of the BAUDCON register. The default state of this bit is '0' which selects high true transmit and receive data. Setting the DTRXP bit to '1' will invert the data resulting in low true transmit and receive data.


© 2009-2016 Microchip Technology Inc.

#### 15.4.1.5 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 15.3 "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 15-10: SYNCHRONOUS TRANSMISSION**




**Note:** Sync Master mode, SPBRG = 0, continuous transmission of two 8-bit words.

**FIGURE 15-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**

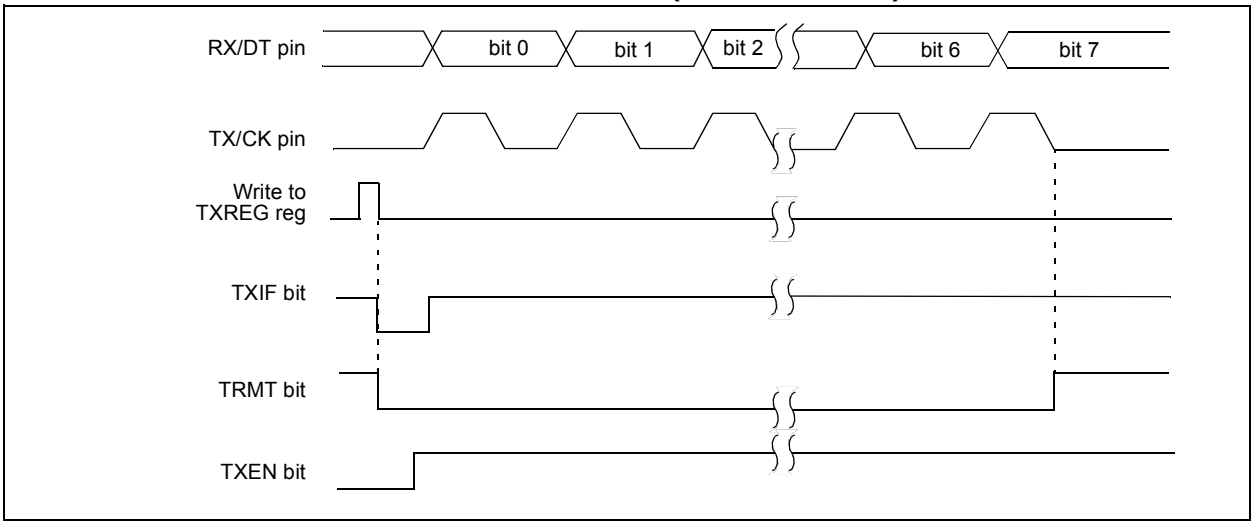


**TABLE 15-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on page |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | DTRXP | CKTXP | BRG16 | — | WUE | ABDEN | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 247 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 247 |
| SPBRGH | EUSART Baud Rate Generator Register, High Byte | | | | | | | | 247 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 248 |
| TXREG | EUSART Transmit Register | | | | | | | | 247 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 247 |

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

#### 15.4.1.6 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 unread characters in the receive FIFO.

#### 15.4.1.7 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.

#### 15.4.1.8 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.


© 2009-2016 Microchip Technology Inc.

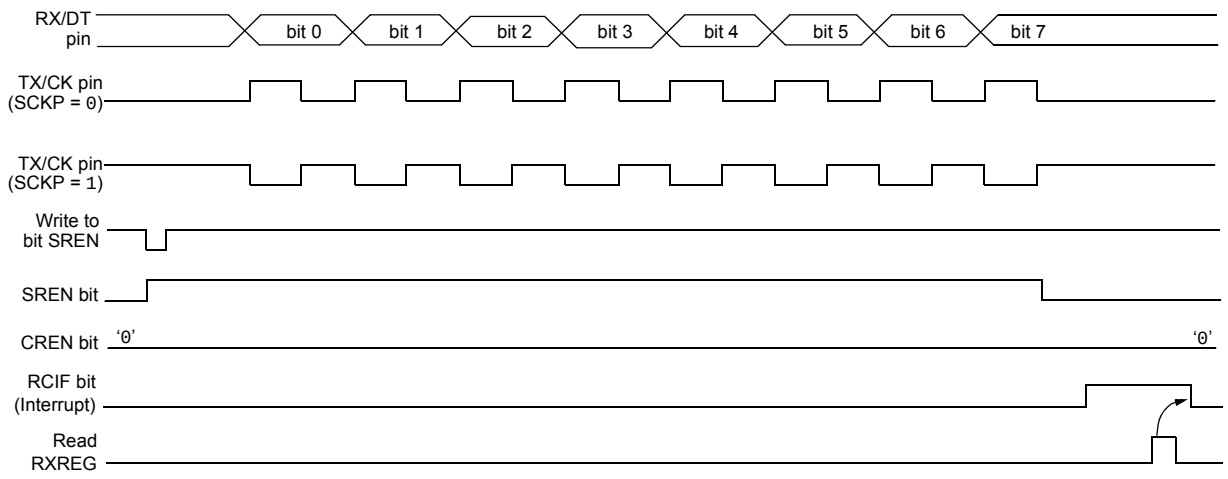
#### 15.4.1.9 Receiving 9-bit Characters

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 eight Least Significant bits from the RCREG.

#### 15.4.1.10 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.

**FIGURE 15-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**




**Note:** Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0.

**TABLE 15-8: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | DTRXP | CKTXP | BRG16 | — | WUE | ABDEN | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| RCREG | EUSART Receive Register | | | | | | | | 247 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 247 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 247 |
| SPBRGH | EUSART Baud Rate Generator Register, High Byte | | | | | | | | 247 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 247 |

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


© 2009-2016 Microchip Technology Inc.

### 15.4.2 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.

#### 15.4.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see **Section 15.4.1.3 "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.

#### 15.4.2.2 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 eight bits to the TXREG register.

**TABLE 15-9: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | DTRXP | CKTXP | BRG16 | — | WUE | ABDEN | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 247 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 247 |
| SPBRGH | EUSART Baud Rate Generator Register, High Byte | | | | | | | | 247 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 248 |
| TXREG | EUSART Transmit Register | | | | | | | | 247 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 247 |

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

#### 15.4.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (**Section 15.4.1.6 "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.

#### 15.4.2.4 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 eight 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 15-10: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | DTRXP | CKTXP | BRG16 | — | WUE | ABDEN | 247 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| RCREG | EUSART Receive Register | | | | | | | | 247 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 247 |
| SPBRG | EUSART Baud Rate Generator Register, Low Byte | | | | | | | | 247 |
| SPBRGH | EUSART Baud Rate Generator Register, High Byte | | | | | | | | 247 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 247 |

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


© 2009-2016 Microchip Technology Inc.

## 16.0 ANALOG-TO-DIGITAL CONVERTER (ADC) MODULE

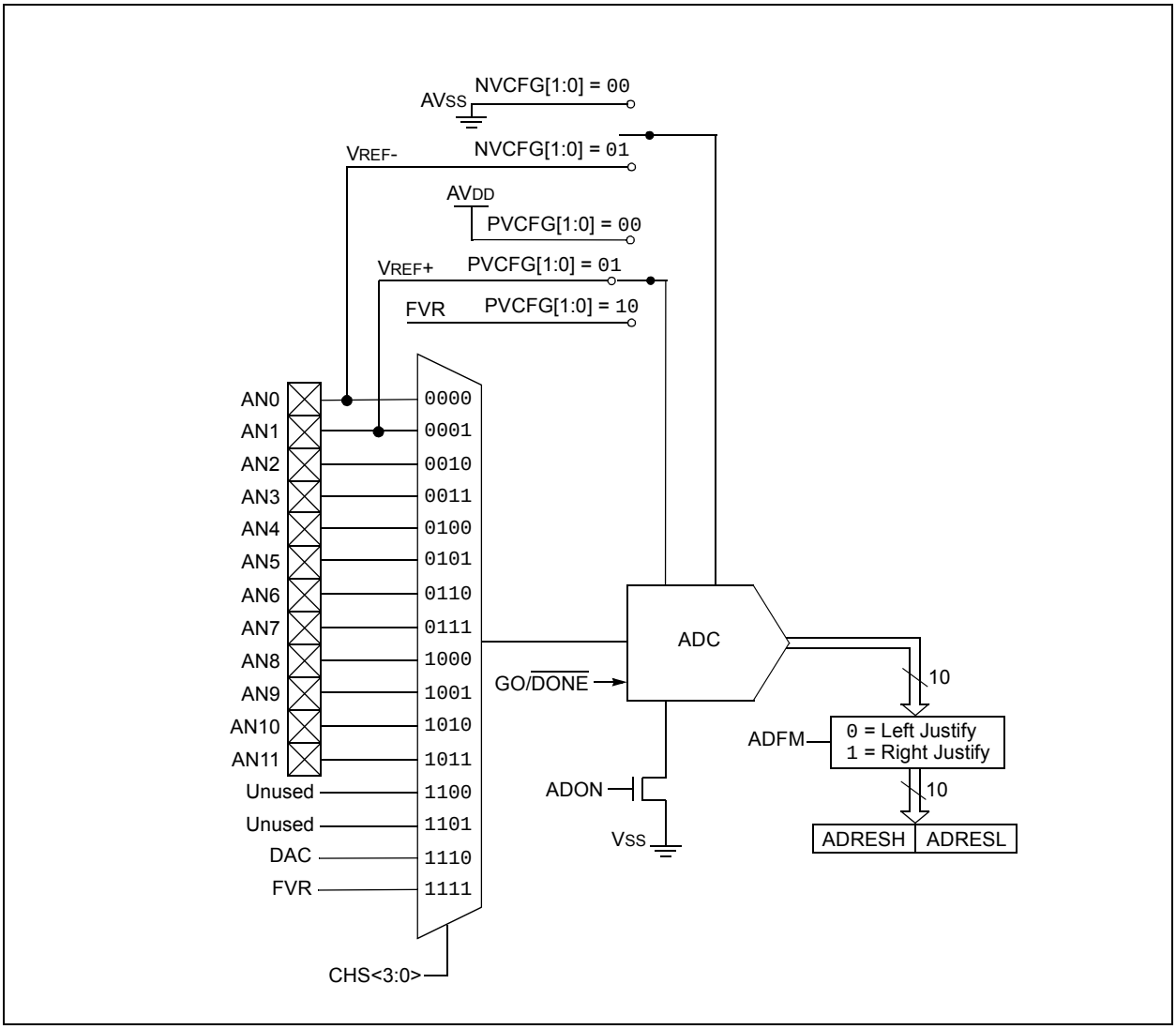
The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. 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 $V_{DD}$, 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 16-1 shows the block diagram of the ADC.

**FIGURE 16-1: ADC BLOCK DIAGRAM**



## 16.1 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

### 16.1.1 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 ($V_{OH}$ or $V_{OL}$) 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. |

### 16.1.2 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 16.2 "ADC Operation"** for more information.

### 16.1.3 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 $V_{DD}$, FVR or an external voltage source. The negative voltage reference can be either $V_{SS}$ or an external voltage source.

### 16.1.4 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 $T_{AD}$. 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.

### 16.1.5 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:

- $F_{OSC}/2$
- $F_{OSC}/4$
- $F_{OSC}/8$
- $F_{OSC}/16$
- $F_{OSC}/32$
- $F_{OSC}/64$
- $F_{RC}$ (dedicated internal oscillator)

The time to complete one bit conversion is defined as $T_{AD}$. One full 10-bit conversion requires 11 $T_{AD}$ periods as shown in Figure 16-3.

For correct conversion, the appropriate $T_{AD}$ specification must be met. See A/D conversion requirements in Table 26-20 for more information. Table 16-1 gives examples of appropriate ADC clock selections.

| | |
|---|---|
| **Note:** | Unless using the $F_{RC}$, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result. |


© 2009-2016 Microchip Technology Inc.

### 16.1.6 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 16.1.6 "Interrupts"** for more information.

**TABLE 16-1: ADC CLOCK PERIOD ($T_{AD}$) vs. DEVICE OPERATING FREQUENCIES**

| ADC Clock Period ($T_{AD}$) | | Device Frequency ($F_{OSC}$) | | | |
|---|---|---|---|---|---|
| **ADC Clock Source** | **ADCS<2:0>** | **48 MHz** | **16 MHz** | **4 MHz** | **1 MHz** |
| $F_{OSC}$/2 | 000 | 41.67 ns(2) | 125 ns(2) | 500 ns(2) | 2.0 µs |
| $F_{OSC}$/4 | 100 | 83.33 ns(2) | 250 ns(2) | 1.0 µs | 4.0 µs |
| $F_{OSC}$/8 | 001 | 167 ns(2) | 500 ns(2) | 2.0 µs | 8.0 µs(3) |
| $F_{OSC}$/16 | 101 | 333 ns(2) | 1.0 µs | 4.0 µs | 16.0 µs(3) |
| $F_{OSC}$/32 | 010 | 667 ns(2) | 2.0 µs | 8.0 µs(3) | 32.0 µs(3) |
| $F_{OSC}$/64 | 110 | 1.33 µs | 4.0 µs | 16.0 µs(3) | 64.0 µs(3) |
| $F_{RC}$ | x11 | 1-4 µs(1,4) | 1-4 µs(1,4) | 1-4 µs(1,4) | 1-4 µs(1,4) |

**Legend:** Shaded cells are outside of recommended range.

**Note 1:** The $F_{RC}$ source has a typical $T_{AD}$ time of 1.7 µs.

**2:** These values violate the minimum required $T_{AD}$ time.

**3:** For faster conversion times, the selection of another clock source is recommended.

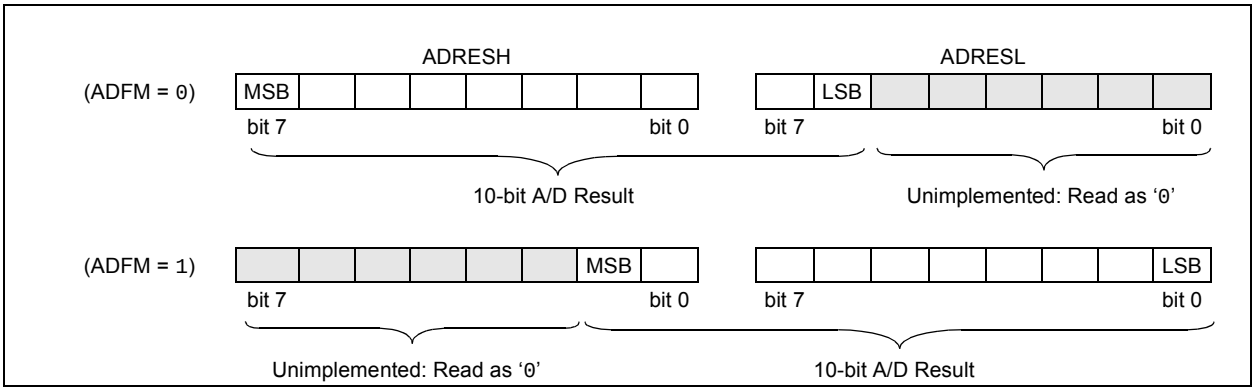
**4:** When the device frequency is greater than 1 MHz, the $F_{RC}$ clock source is only recommended if the conversion will be performed during Sleep.

### 16.1.7 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 16-2 shows the two output formats.

**FIGURE 16-2: 10-BIT A/D CONVERSION RESULT FORMAT**



## 16.2 ADC Operation

### 16.2.1 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 16-3 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 16-4 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 16.2.9 "A/D Conversion Procedure"**. |
|---|---|

**FIGURE 16-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**

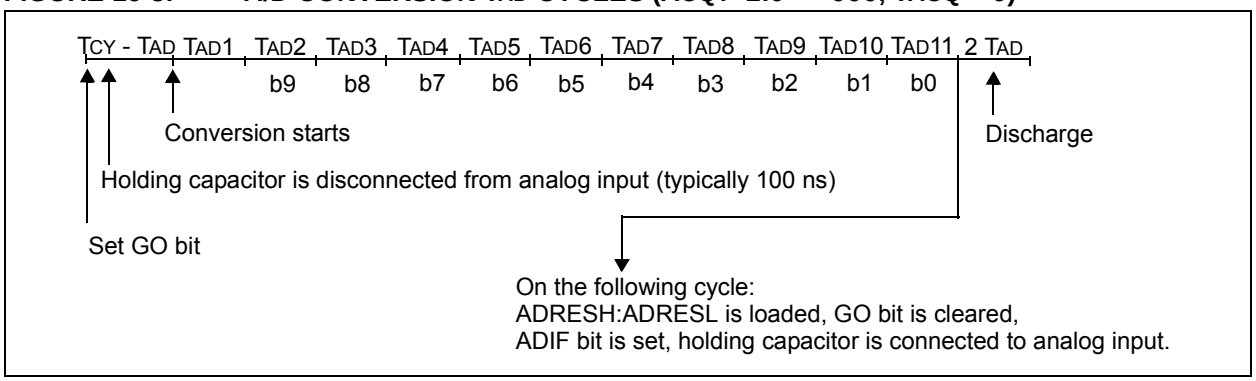
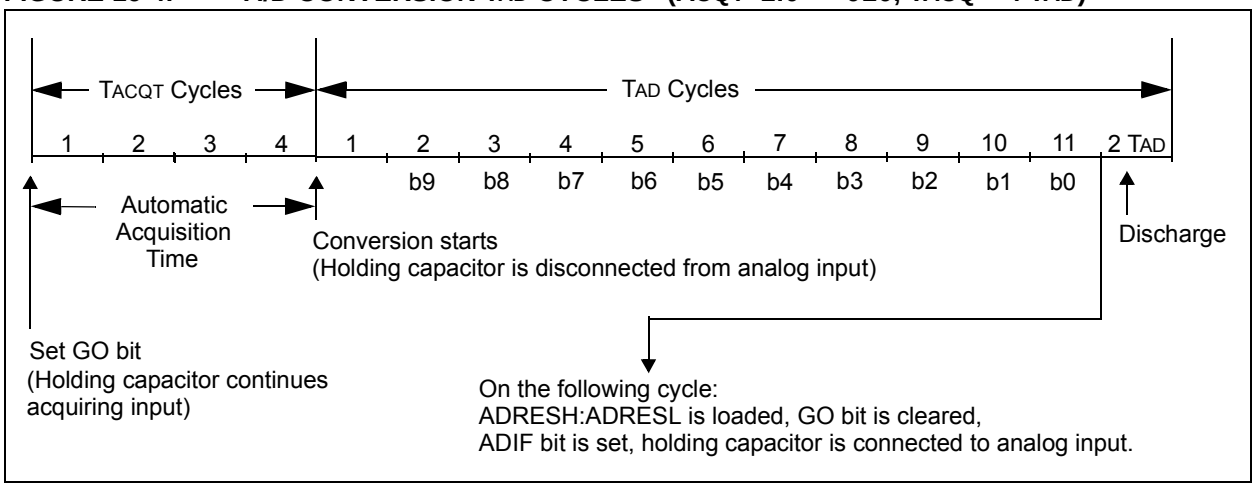



**FIGURE 16-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**





© 2009-2016 Microchip Technology Inc.

### 16.2.2 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

### 16.2.3 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.

### 16.2.4 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. |

### 16.2.5 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.

### 16.2.6 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.

### 16.2.7 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.

### 16.2.8 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 13.3.4 "Special Event Trigger"** for more information.

### 16.2.9 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(1)
4. Wait the required acquisition time(2).
5. Start conversion by setting the GO/$\overline{\text{DONE}}$ bit.
6. Wait for ADC conversion to complete by one of the following:
   - Polling the GO/$\overline{\text{DONE}}$ 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 16.3 "A/D Acquisition Requirements"**.

**EXAMPLE 16-1: 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:
BTFSC     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
```


© 2009-2016 Microchip Technology Inc.

### 16.2.10 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 8-14 and Register 8-15, respectively. |

**REGISTER 16-1: ADCON0: A/D CONTROL REGISTER 0**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON |
| bit 7 | | | | | | | bit 0 |

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

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

bit 5-2 **CHS<3:0>: Analog Channel Select bits**

0000 = AN0
0001 = AN1
0010 = AN2
0011 = AN3
0100 = AN4
0101 = AN5
0110 = AN6
0111 = AN7
1000 = AN8
1001 = AN9
1010 = AN10
1011 = AN11
1100 = Reserved
1101 = Reserved
1110 = DAC(2)
1111 = FVR(2)

bit 1 **GO/$\overline{\text{DONE}}$:** 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.

**2:** See **Section 20.0 "Fixed Voltage Reference (FVR)"** for more information.

**REGISTER 16-2: 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 | | | | | | | 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 **Unimplemented:** Read as '0'

bit 3-2 **PVCFG<1:0>:** Positive Voltage Reference select bit

00 = Positive voltage reference supplied internally by $V_{DD}$.
01 = Positive voltage reference supplied externally through $V_{REF}$+ pin.
10 = Positive voltage reference supplied internally through FVR.
11 = Reserved.

bit 1-0 **NVCFG<1:0>:** Negative Voltage Reference select bit

00 = Negative voltage reference supplied internally by $V_{SS}$.
01 = Negative voltage reference supplied externally through $V_{REF}$- pin.
10 = Reserved.
11 = Reserved.


© 2009-2016 Microchip Technology Inc.

**REGISTER 16-3: ADCON2: A/D CONTROL REGISTER 2**

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 |
| bit 7 | | | | | | | bit 0 |

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

bit 7 **ADFM:** A/D Conversion Result Format Select bit
1 = Right justified
0 = Left justified

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

bit 5-3 **ACQT<2:0>:** A/D Acquisition Time Select bits. 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(1)
001 = 2 TAD
010 = 4 TAD
011 = 6 TAD
100 = 8 TAD
101 = 12 TAD
110 = 16 TAD
111 = 20 TAD

bit 2-0 **ADCS<2:0>:** A/D Conversion Clock Select bits
000 = FOSC/2
001 = FOSC/8
010 = FOSC/32
011 = FRC(1) (clock derived from a dedicated internal oscillator = 600 kHz nominal)
100 = FOSC/4
101 = FOSC/16
110 = FOSC/64
111 = FRC(1) (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.

**REGISTER 16-4: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) 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 |
|---|---|---|---|---|---|---|---|
| ADRES9 | ADRES8 | ADRES7 | ADRES6 | ADRES5 | ADRES4 | ADRES3 | ADRES2 |
| 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 **ADRES<9:2>**: ADC Result Register bits
Upper 8 bits of 10-bit conversion result

**REGISTER 16-5: 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 | 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 **ADRES<1:0>**: ADC Result Register bits
Lower 2 bits of 10-bit conversion result

bit 5-0 **Reserved**: Do not use.

**REGISTER 16-6: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 1**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | ADRES9 | ADRES8 |
| bit 7 | | | | | | | bit 0 |

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

bit 7-2 **Reserved**: Do not use.

bit 1-0 **ADRES<9:8>**: ADC Result Register bits
Upper 2 bits of 10-bit conversion result

**REGISTER 16-7: ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 1**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| ADRES7 | ADRES6 | ADRES5 | ADRES4 | ADRES3 | ADRES2 | ADRES1 | ADRES0 |
| 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 **ADRES<7:0>**: ADC Result Register bits
Lower 8 bits of 10-bit conversion result


© 2009-2016 Microchip Technology Inc.

## 16.3 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 16-5. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), see Figure 16-5. **The maximum recommended impedance for analog sources is 10 kΩ.** 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 16-1 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 16-1: ACQUISITION TIME EXAMPLE**

*Assumptions:* $Temperature = 50°C$ *and external impedance of* $10k\Omega$ $3.0V$ $V_{DD}$

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \\ &= 5\mu s + T_C + [(Temperature - 25°C)(0.05\mu s/°C)] \end{aligned}$$

*The value for* $T_C$ *can be approximated with the following equations:*

$$V_{APPLIED}\left(1 - \frac{1}{2047}\right) = V_{CHOLD} \quad ;[1]\ V_{CHOLD}\ \text{charged to within 1/2 lsb}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD} \quad ;[2]\ V_{CHOLD}\ \text{charge response to}\ V_{APPLIED}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{2047}\right) \quad ;\text{combining [1] and [2]}$$

*Solving for* $T_C$*:*

$$\begin{aligned} T_C &= -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln(1/2047) \\ &= -13.5pF(1k\Omega + 700\Omega + 10k\Omega)\ ln(0.0004885) \\ &= 1.20\mu s \end{aligned}$$

*Therefore:*

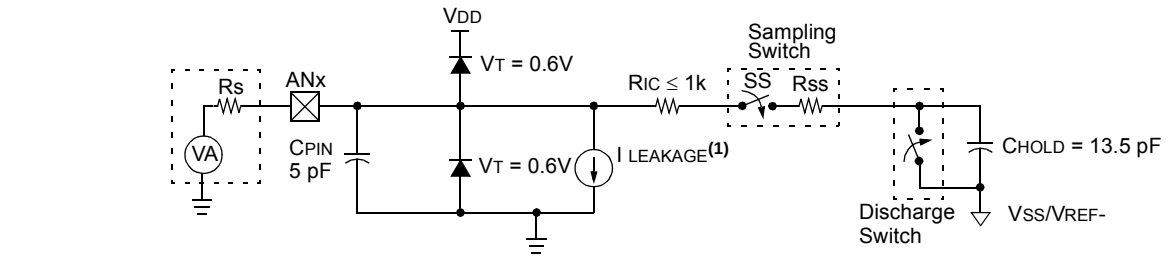
$$\begin{aligned} T_{ACQ} &= 5\mu s + 1.20\mu s + [(50°C - 25°C)(0.05\mu s/°C)] \\ &= 7.45\mu s \end{aligned}$$

**Note 1:** The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

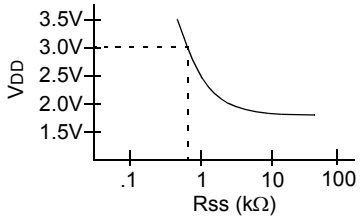
**2:** The charge holding capacitor (CHOLD) is discharged after each conversion.

**3:** The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

**FIGURE 16-5: ANALOG INPUT MODEL**

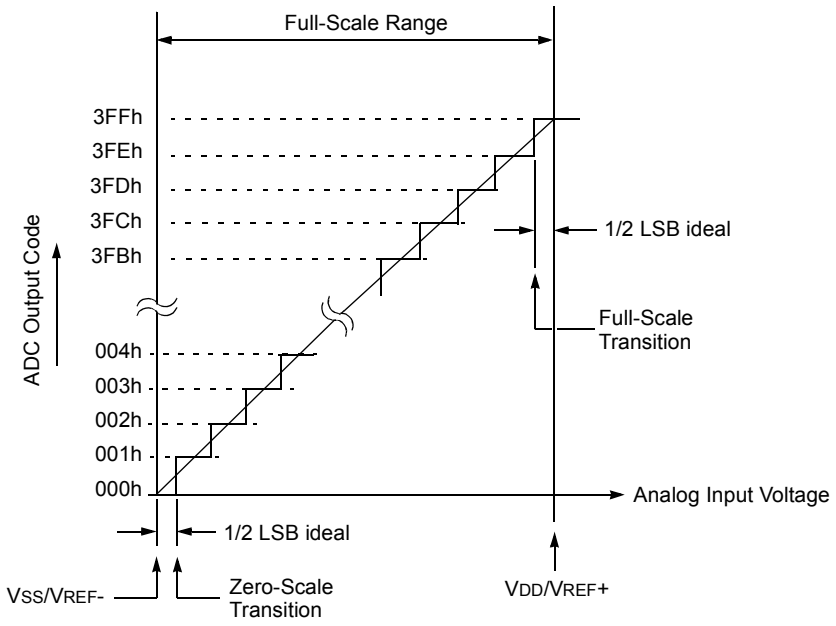



**Legend:** CPIN = Input Capacitance
VT = Threshold Voltage
I LEAKAGE = Leakage current at the pin due to various junctions
RIC = Interconnect Resistance
SS = Sampling Switch
CHOLD = Sample/Hold Capacitance




**Note 1:** See **Section 26.0 "Electrical Specifications"**.

**FIGURE 16-6: ADC TRANSFER FUNCTION**





© 2009-2016 Microchip Technology Inc.

**TABLE 16-2: 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 |
|---|---|---|---|---|---|---|---|---|---|
| ADRESH | A/D Result Register, High Byte | | | | | | | | 247 |
| ADRESL | A/D Result Register, Low Byte | | | | | | | | 247 |
| ADCON0 | — | — | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON | 247 |
| ADCON1 | — | — | — | — | PVCFG1 | PVCFG0 | NVCFG1 | NVCFG0 | 247 |
| ADCON2 | ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 247 |
| ANSEL | ANS7 | ANS6 | ANS5 | ANS4 | ANS3 | ANS2 | ANS1 | ANS0 | 248 |
| ANSELH | — | — | — | — | ANS11 | ANS10 | ANS9 | ANS8 | 248 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR1 | — | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 248 |
| PIE1 | — | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 248 |
| PIR1 | — | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 248 |
| TRISA | — | — | TRISA5 | TRISA4 | —(1) | TRISA2 | TRISA1 | TRISA0 | 248 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | — | — | — | — | 248 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 248 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.

**Note 1:** Unimplemented, read as '1'.

## 17.0 COMPARATOR MODULE

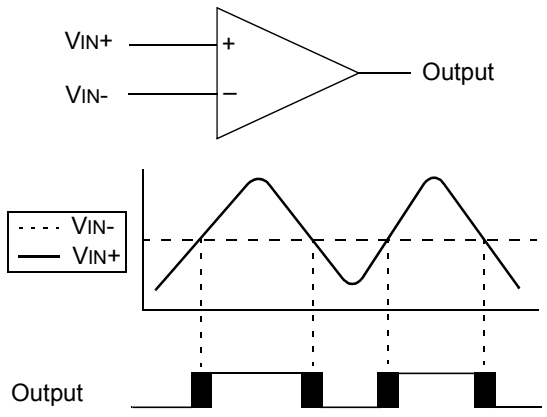
Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. The comparators are very useful mixed signal building blocks because they provide analog functionality independent of the program execution. 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

### 17.1 Comparator Overview

A single comparator is shown in Figure 17-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at VIN+ is less than the analog voltage at VIN-, the output of the comparator is a digital low level. When the analog voltage at VIN+ is greater than the analog voltage at VIN-, the output of the comparator is a digital high level.

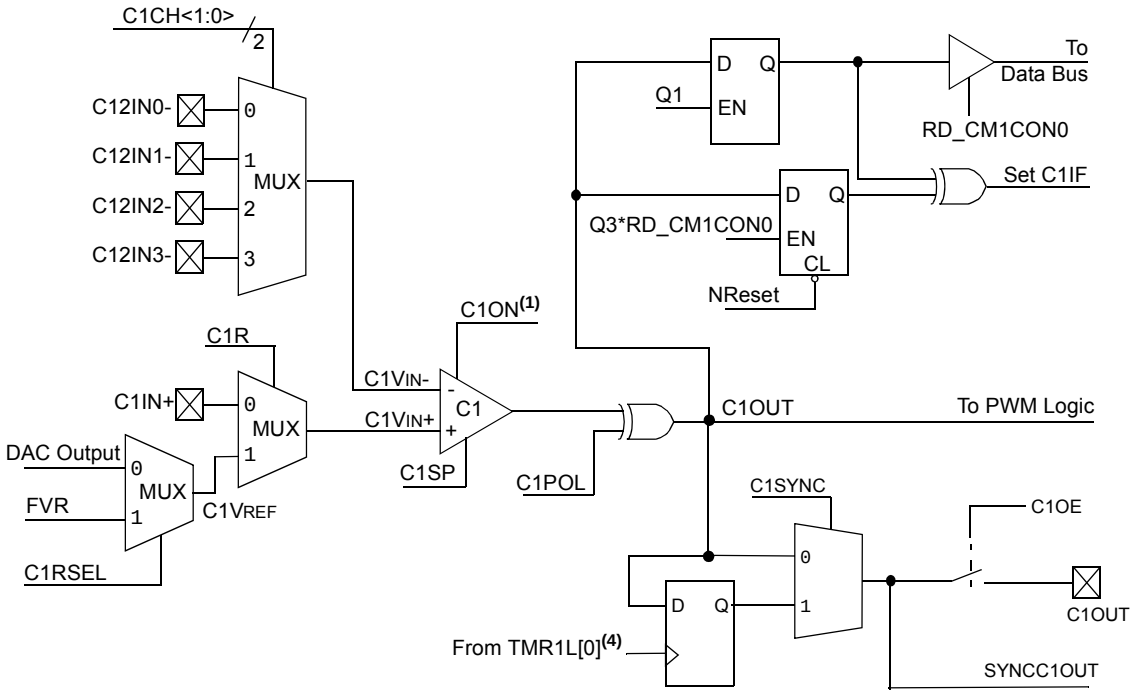
**FIGURE 17-1: SINGLE COMPARATOR**




**Note:** The black areas of the output of the comparator represents the uncertainty due to input offsets and response time.

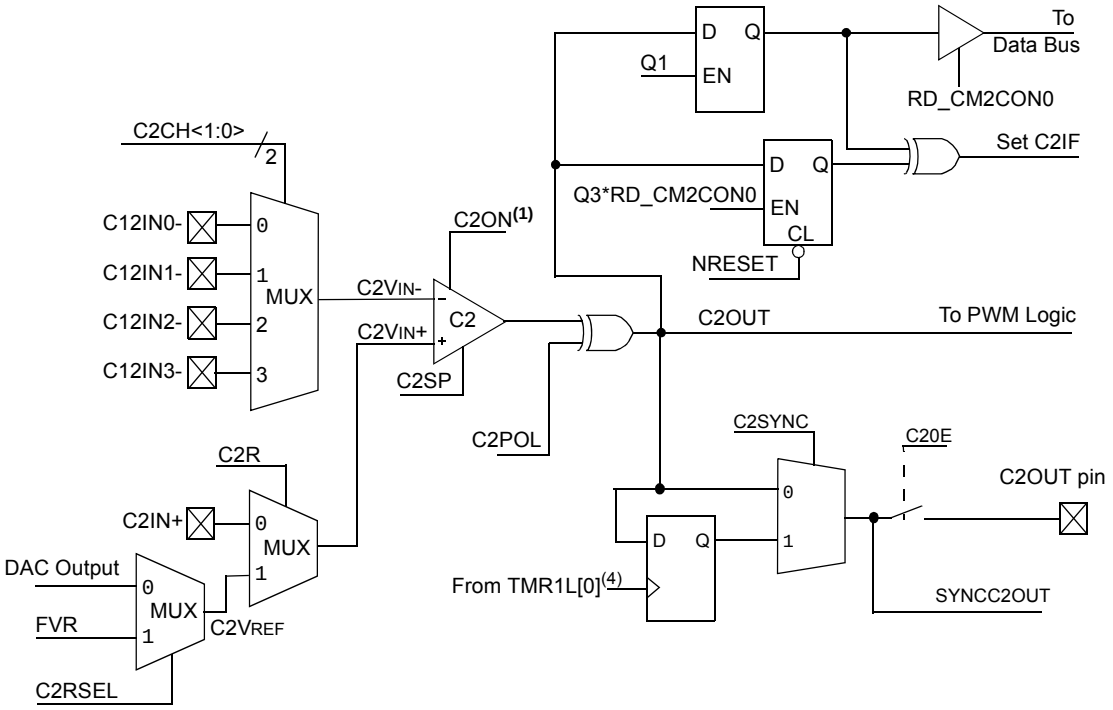

© 2009-2016 Microchip Technology Inc.

**FIGURE 17-2: COMPARATOR C1 SIMPLIFIED BLOCK DIAGRAM**




**Note 1:** When C1ON = 0, the C1 comparator will produce a '0' output to the XOR Gate.
**2:** Q1 and Q3 are phases of the four-phase system clock (FOSC).
**3:** Q1 is held high during Sleep mode.
**4:** Positive going pulse generated on both falling and rising edges of the bit.

**FIGURE 17-3: COMPARATOR C2 SIMPLIFIED BLOCK DIAGRAM**




**Note 1:** When C2ON = 0, the C2 comparator will produce a '0' output to the XOR Gate.
**2:** Q1 and Q3 are phases of the four-phase system clock (FOSC).
**3:** Q1 is held high during Sleep mode.
**4:** Positive going pulse generated on both falling and rising edges of the bit.


© 2009-2016 Microchip Technology Inc.

## 17.2 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 17-1 and 17-2, respectively) contain the control and status bits for the following:

- Enable
- Input selection
- Reference selection
- Output selection
- Output polarity
- Speed selection

### 17.2.1 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.

### 17.2.2 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. |

### 17.2.3 COMPARATOR REFERENCE SELECTION

Setting the CxR bit of the CMxCON0 register directs an internal voltage reference or an analog input pin to the noninverting input of the comparator. See **Section 20.0 "Fixed Voltage Reference (FVR)"** for more information on the Internal Voltage Reference module.

### 17.2.4 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

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

### 17.2.5 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 17-1 shows the output state versus input conditions, including polarity control.

**TABLE 17-1: 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 |

### 17.2.6 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'.

## 17.3 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 26.0 "Electrical Specifications"** for more details.

## 17.4 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 17-2 and Figure 17-3). 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 17-4 and 17-5.

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.

### 17.4.1 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 be occur in the final Write phase.

**FIGURE 17-4: COMPARATOR INTERRUPT TIMING W/O CMxCON0 READ**

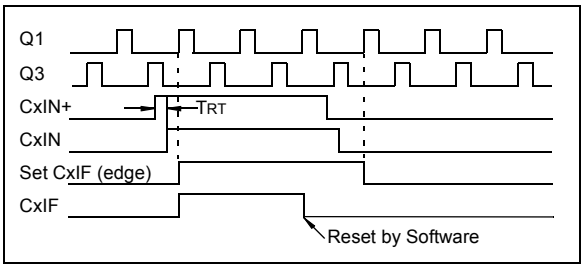
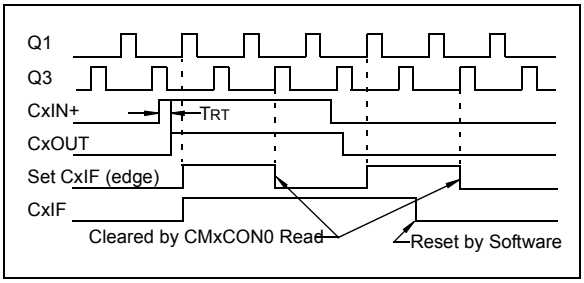



**FIGURE 17-5: COMPARATOR INTERRUPT TIMING WITH CMxCON0 READ**




**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 μs for bias settling then clear the mismatch condition and interrupt flags before enabling comparator interrupts.


© 2009-2016 Microchip Technology Inc.

## 17.5 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 **Section 26.0 "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.

## 17.6 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.

**REGISTER 17-1: 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 | C1CH0 |
| 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 **C1ON:** Comparator C1 Enable bit
1 = Comparator C1 is enabled
0 = Comparator C1 is disabled

bit 6 **C1OUT:** Comparator C1 Output bit
If C1POL = 1 (inverted polarity):
C1OUT = 0 when C1VIN+ > C1VIN-
C1OUT = 1 when C1VIN+ < C1VIN-
If C1POL = 0 (non-inverted polarity):
C1OUT = 1 when C1VIN+ > C1VIN-
C1OUT = 0 when C1VIN+ < C1VIN-

bit 5 **C1OE:** Comparator C1 Output Enable bit
1 = C1OUT is present on the C1OUT pin(1)
0 = C1OUT is internal only

bit 4 **C1POL:** Comparator C1 Output Polarity Select bit
1 = C1OUT logic is inverted
0 = C1OUT logic is not inverted

bit 3 **C1SP:** 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 **C1R:** Comparator C1 Reference Select bit (noninverting input)
1 = C1VIN+ connects to C1VREF output
0 = C1VIN+ connects to C12IN+ pin

bit 1-0 **C1CH<1:0>:** Comparator C1 Channel Select bit
00 = C12IN0- pin of C1 connects to C1VIN-
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.


© 2009-2016 Microchip Technology Inc.

**REGISTER 17-2: 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 **C2ON:** Comparator C2 Enable bit
1 = Comparator C2 is enabled
0 = Comparator C2 is disabled

bit 6 **C2OUT:** Comparator C2 Output bit
If C2POL = 1 (inverted polarity):
C2OUT = 0 when C2VIN+ > C2VIN-
C2OUT = 1 when C2VIN+ < C2VIN-
If C2POL = 0 (non-inverted polarity):
C2OUT = 1 when C2VIN+ > C2VIN-
C2OUT = 0 when C2VIN+ < C2VIN-

bit 5 **C2OE:** Comparator C2 Output Enable bit
1 = C2OUT is present on C2OUT pin(1)
0 = C2OUT is internal only

bit 4 **C2POL:** Comparator C2 Output Polarity Select bit
1 = C2OUT logic is inverted
0 = C2OUT logic is not inverted

bit 3 **C2SP:** 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 **C2R:** Comparator C2 Reference Select bits (noninverting input)
1 = C2VIN+ connects to C2VREF
0 = C2VIN+ connects to C2IN+ pin

bit 1-0 **C2CH<1:0>:** Comparator C2 Channel Select bits
00 = C12IN0- pin of C2 connects to C2VIN-
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.

## 17.7 Analog Input Connection Considerations

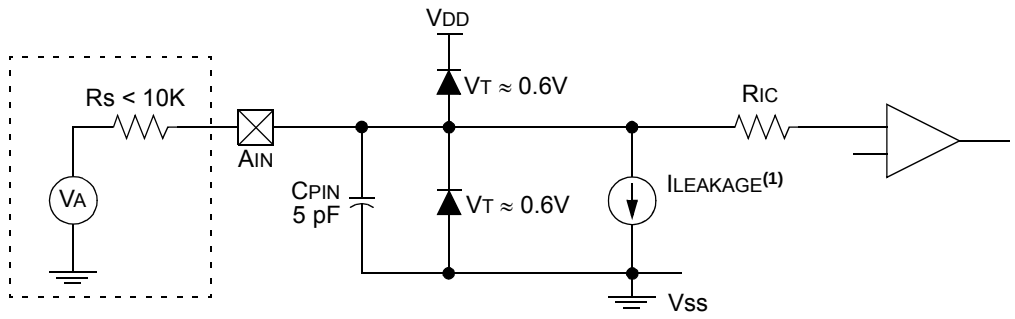
A simplified circuit for an analog input is shown in Figure 17-6. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range 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 kΩ 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 17-6: ANALOG INPUT MODEL**




**Legend:** CPIN = Input Capacitance
ILEAKAGE = Leakage Current at the pin due to various junctions
RIC = Interconnect Resistance
RS = Source Impedance
VA = Analog Voltage
VT = Threshold Voltage

**Note 1:** See **Section 26.0 "Electrical Specifications"**.


© 2009-2016 Microchip Technology Inc.

## 17.8 Additional Comparator Features

There are four additional comparator features:

- Simultaneous read of comparator outputs
- Internal reference selection
- Hysteresis selection
- Output Synchronization

### 17.8.1 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. |

### 17.8.2 INTERNAL REFERENCE SELECTION

There are two internal voltage references available to the noninverting input of each comparator. One of these is the Fixed Voltage Reference (FVR) and the other is the variable Digital-to-Analog Converter ($CV_{REF}$/DAC). The CxRSEL bit of the CM2CON register determines which of these references is routed to the Digital-to-Analog Converter output ($CV_{REF}$/DAC). Further routing to the comparator is accomplished by the CxR bit of the CMxCON0 register. See **20.0 "Fixed Voltage Reference (FVR)"** and Figure 17-2 and Figure 17-3 for more detail.

### 17.8.3 COMPARATOR HYSTERESIS

The Comparator Cx have selectable hysteresis. The hysteresis can be enabled by setting the CxHYS bit of the CM2CON1 register. See **Section 26.0 "Electrical Specifications"** for more details.

### 17.8.4 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 17-2 and Figure 17-3) and the Timer1 Block Diagram (Figure 17-2) for more information.

**REGISTER 17-3: CMCON0: 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 **MC1OUT:** Mirror Copy of C1OUT bit

bit 6 **MC2OUT:** Mirror Copy of C2OUT bit

bit 5 **C1RSEL:** Comparator C1 Reference Select bit
- 1 = FVR routed to C1VREF input
- 0 = CVREF/DAC1OUT routed to C1VREF input

bit 4 **C2RSEL:** Comparator C2 Reference Select bit
- 1 = FVR routed to C2VREF input
- 0 = CVREF/DAC1OUT routed to C2VREF input

bit 3 **C1HYS:** Comparator C1 Hysteresis Enable bit
- 1 = Comparator C1 hysteresis enabled
- 0 = Comparator C1 hysteresis disabled

bit 2 **C2HYS:** Comparator C2 Hysteresis Enable bit
- 1 = Comparator C2 hysteresis enabled
- 0 = Comparator C2 hysteresis disabled

bit 1 **C1SYNC:** C1 Output Synchronous Mode bit
- 1 = C1 output is synchronous to rising edge to TMR1 clock
- 0 = C1 output is asynchronous

bit 0 **C2SYNC:** C2 Output Synchronous Mode bit
- 1 = C2 output is synchronous to rising edge to TMR1 clock
- 0 = C2 output is asynchronous


© 2009-2016 Microchip Technology Inc.

**TABLE 17-2: 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 |
|---|---|---|---|---|---|---|---|---|---|
| ANSEL | ANS7 | ANS6 | ANS5 | ANS4 | ANS3 | ANS2 | ANS1 | ANS0 | 248 |
| CM1CON0 | C1ON | C1OUT | C1OE | C1POL | C1SP | C1R | C1CH1 | C1CH0 | 248 |
| CM2CON0 | C2ON | C2OUT | C2OE | C2POL | C2SP | C2R | C2CH1 | C2CH0 | 248 |
| CM2CON1 | MC1OUT | MC2OUT | C1RSEL | C2RSEL | C1HYS | C2HYS | C1SYNC | C2SYNC | 248 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RABIE | TMR0IF | INT0IF | RABIF | 245 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | — | TMR3IP | — | 248 |
| LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | 248 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | — | TMR3IE | — | 248 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | — | TMR3IF | — | 248 |
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | 248 |
| VREFCON0 | FVR1EN | FVR1ST | FVR1S<1:0> | | — | — | — | — | 247 |
| VREFCON1 | D1EN | D1LPS | DAC1OE | --- | D1PSS<1:0> | | — | D1NSS | 247 |
| TRISA | — | — | TRISA5 | TRISA4 | —(1) | TRISA2 | TRISA1 | TRISA0 | 248 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 248 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

**Note 1:** Unimplemented, read as '1'.

## 18.0 POWER-MANAGED MODES

PIC18(L)F1XK22 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.

## 18.1 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 18-1.

### 18.1.1 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

### 18.1.2 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 2.9 "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 18-1: POWER-MANAGED MODES**

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

**Note 1:** IDLEN reflects its value when the SLEEP instruction is executed.
**2:** Includes HFINTOSC and HFINTOSC postscaler, as well as the LFINTOSC source.


© 2009-2016 Microchip Technology Inc.

### 18.1.3 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.

## 18.2 Run Modes

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

### 18.2.1 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 2.11 "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.

### 18.2.2 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. |

### 18.2.3 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.

## 18.3 Sleep Mode

The Power-Managed Sleep mode in the PIC18(L)F1XK22 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 18-1) 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 18-2), 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 23.0 "Special Features of the CPU"**). In either case, the OSTS bit is set when the primary clock is providing the device clocks. The IDLEN and SCS bits are not affected by the wake-up.

## 18.4 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 18-1: TRANSITION TIMING FOR ENTRY TO SLEEP MODE**

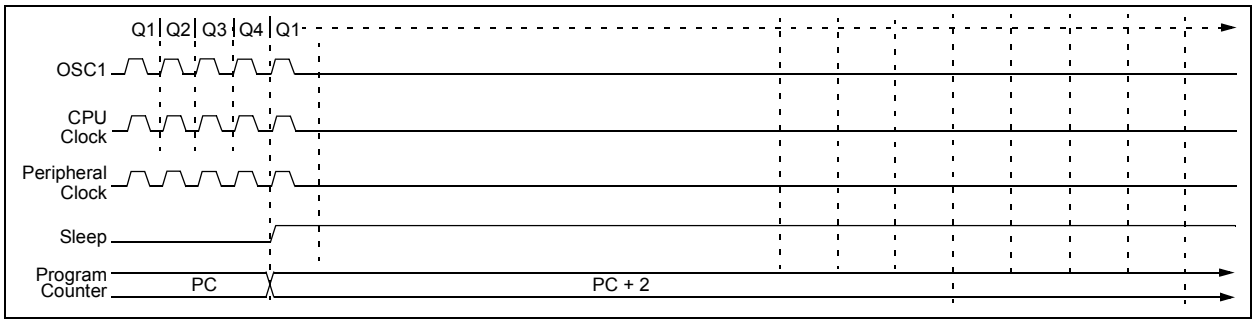
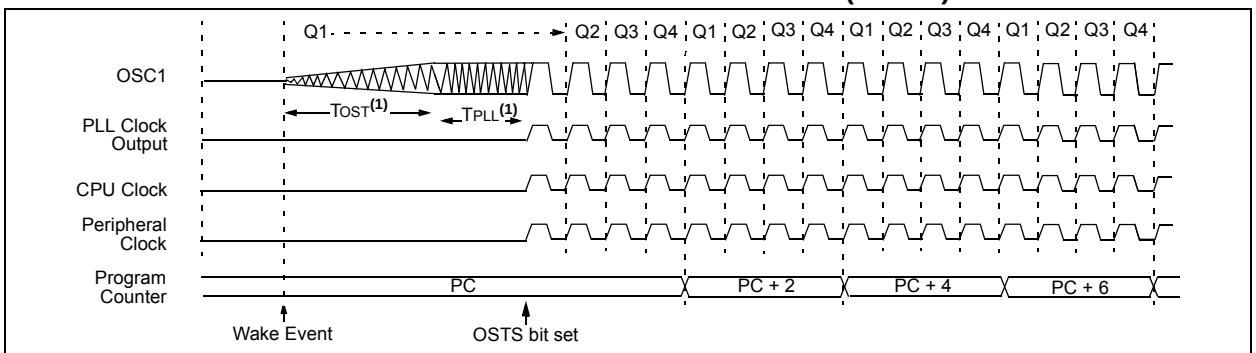



**FIGURE 18-2: TRANSITION TIMING FOR WAKE FROM SLEEP (HSPLL)**




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


© 2009-2016 Microchip Technology Inc.

### 18.4.1 PRI_IDLE MODE

This mode is unique among the three low-power Idle modes, in that it does not disable the primary device clock. For timing sensitive applications, this allows for the fastest resumption of device operation with its more accurate primary clock source, since the clock source does not have to “warm-up” or transition from another oscillator.

PRI_IDLE mode is entered from PRI_RUN mode by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set IDLEN first, then clear the SCS bits and execute SLEEP. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified by the FOSC<3:0> Configuration bits. The OSTS bit remains set (see Figure 18-3).

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 OSTS bit remains set. The IDLEN and SCS bits are not affected by the wake-up (see Figure 18-4).

### 18.4.2 SEC_IDLE MODE

In SEC_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the Timer1 oscillator. This mode is entered from SEC_RUN by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set the IDLEN bit first, then set the SCS<1:0> bits to ‘01’ and execute SLEEP. When the clock source is switched to the Timer1 oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the T1RUN bit is set.

When a wake event occurs, the peripherals continue to be clocked from the Timer1 oscillator. After an interval of TCSD following the wake event, the CPU begins executing code being clocked by the Timer1 oscillator. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run (see Figure 18-4).

| **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 18-3: TRANSITION TIMING FOR ENTRY TO IDLE MODE**

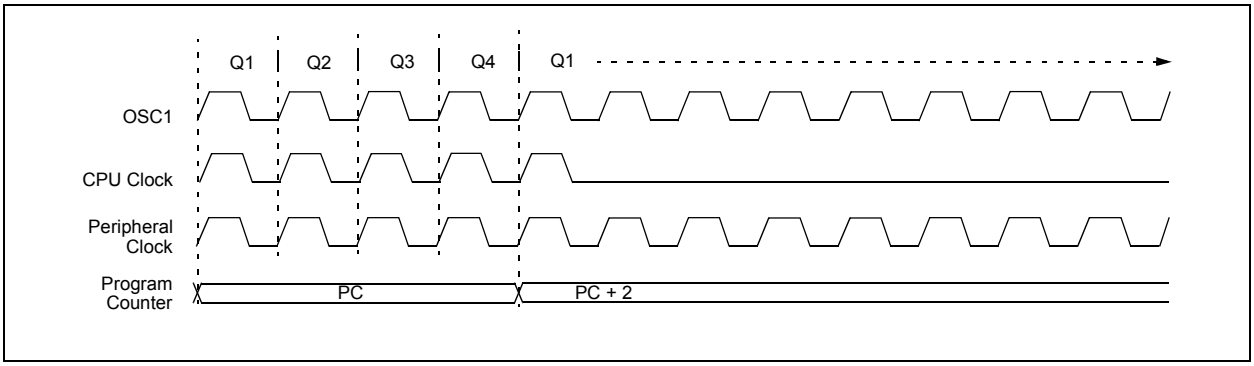
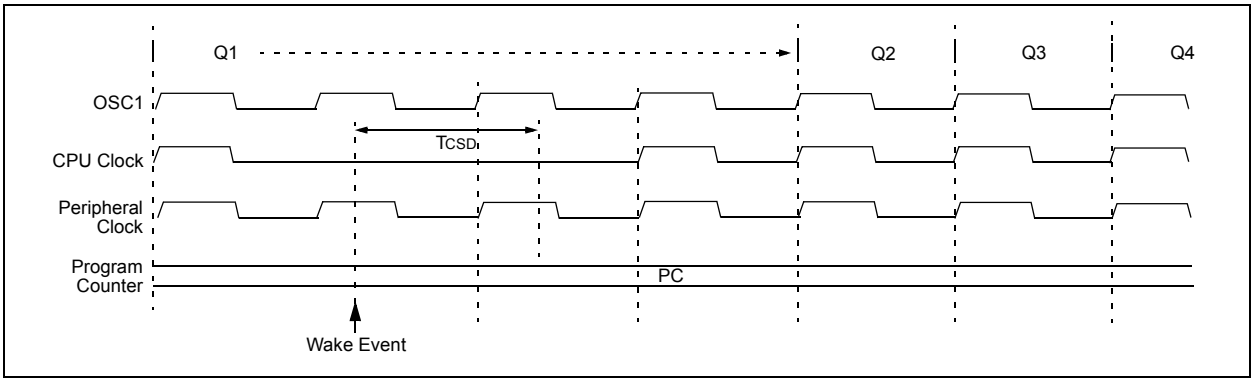



**FIGURE 18-4: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE**



### 18.4.3 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.

## 18.5 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 18.2 “Run Modes”**, **Section 18.3 “Sleep Mode”** and **Section 18.4 “Idle Modes”**).

### 18.5.1 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.

### 18.5.2 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 18.2 “Run Modes”** and **Section 18.3 “Sleep Mode”**). If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see **Section 23.2 “Watchdog Timer (WDT)”**).

The WDT timer and postscaler are cleared by any one of the following:

- Executing a SLEEP instruction
- Executing a CLRWDT 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


© 2009-2016 Microchip Technology Inc.

### 18.5.3 EXIT BY RESET

Exiting Sleep and Idle modes by Reset causes code execution to restart at address 0. See **Section 22.0 "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 18-2.

### 18.5.4 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 INTOSCIO 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 18-2: 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(1) | OSTS |
| | HSPLL | | |
| | EC, RC | | |
| | HFINTOSC(2) | | IOSF |
| T1OSC or LFINTOSC(1) | LP, XT, HS | TOST(3) | OSTS |
| | HSPLL | TOST + $t_{PLL}$(3) | |
| | EC, RC | TCSD(1) | |
| | HFINTOSC(1) | TIOBST(4) | IOSF |
| HFINTOSC(2) | LP, XT, HS | TOST(4) | OSTS |
| | HSPLL | TOST + $t_{PLL}$(3) | |
| | EC, RC | TCSD(1) | |
| | HFINTOSC(1) | None | IOSF |
| None (Sleep mode) | LP, XT, HS | TOST(3) | OSTS |
| | HSPLL | TOST + $t_{PLL}$(3) | |
| | EC, RC | TCSD(1) | |
| | HFINTOSC(1) | TIOBST(4) | 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 18.4 "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. $t_{PLL}$ is the PLL Lock-out Timer (parameter F12).

**4:** Execution continues during the HFINTOSC stabilization period, TIOBST.

## 19.0 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 $\overline{Q}$ output
- Firmware Set and Reset
- SR Latch

### 19.1 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.

### 19.2 Latch Output

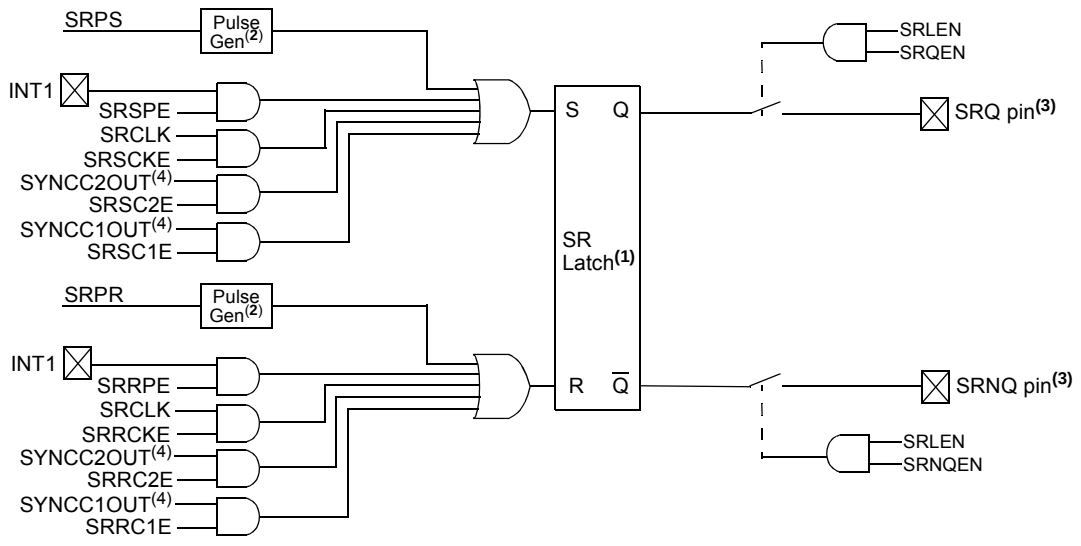
The SRQEN and SRNQEN bits of the SRCON0 register control the latch output selection. Both of the SR latch's outputs may be directly output to an independent I/O pin. Control is determined by the state of bits SRQEN and SRNQEN in registers SRCON0.

The applicable TRIS bit of the corresponding port must be cleared to enable the port pin output driver.

### 19.3 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 19-1: SR LATCH SIMPLIFIED BLOCK DIAGRAM**




**Note 1:** If R = 1 and S = 1 simultaneously, Q = 0, $\overline{Q}$ = 1
**2:** Pulse generator causes a 2 Q-state pulse width.
**3:** Output shown for reference only. See I/O port pin block diagram for more detail.
**4:** Name denotes the source of connection at the comparator output.


© 2009-2016 Microchip Technology Inc.

**TABLE 19-1: SRCLK FREQUENCY TABLE**

| SRCLK | Divider | Fosc = 20 MHz | Fosc = 16 MHz | Fosc = 8 MHz | Fosc = 4 MHz | Fosc = 1 MHz |
|---|---|---|---|---|---|---|
| 111 | 512 | 25.6 μs | 32 μs | 64 μs | 128 μs | 512 μs |
| 110 | 256 | 12.8 μs | 16 μs | 32 μs | 64 μs | 256 μs |
| 101 | 128 | 6.4 μs | 8 μs | 16 μs | 32 μs | 128 μs |
| 100 | 64 | 3.2 μs | 4 μs | 8 μs | 16 μs | 64 μs |
| 011 | 32 | 1.6 μs | 2 μs | 4 μs | 8 μs | 32 μs |
| 010 | 16 | 0.8 μs | 1 μs | 2 μs | 4 μs | 16 μs |
| 001 | 8 | 0.4 μs | 0.5 μs | 1 μs | 2 μs | 8 μs |
| 000 | 4 | 0.2 μs | 0.25 μs | 0.5 μs | 1 μs | 4 μs |

**REGISTER 19-1: 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 | 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 **SRLEN:** SR Latch Enable bit(1)
- 1 = SR latch is enabled
- 0 = SR latch is disabled

bit 6-4 **SRCLK<2:0>(1):** 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
- 1 = Q is present on the SRQ pin
- 0 = Q is internal only

bit 2 **SRNQEN:** SR Latch $\overline{Q}$ Output Enable bit
- 1 = $\overline{Q}$ is present on the SRNQ pin
- 0 = $\overline{Q}$ is internal only

bit 1 **SRPS:** Pulse Set Input of the SR Latch bit
- 1 = Pulse input
- 0 = Always reads back '0'

bit 0 **SRPR:** Pulse Reset Input of the SR Latch bit
- 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.

**REGISTER 19-2: 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 **SRSPE:** 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 **SRSCKE:** 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 **SRSC2E:** 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 **SRSC1E:** 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 **SRRPE:** SR Latch Peripheral Reset Enable bit
1 = INT1 pin resets SR latch
0 = INT1 pin has no effect on SR latch

bit 2 **SRRCKE:** 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 **SRRC2E:** 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 **SRRC1E:** SR Latch C1 Reset Enable bit
1 = C1 Comparator output resets SR latch
0 = C1 Comparator output has no effect on SR latch

**TABLE 19-2: 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 |
|---|---|---|---|---|---|---|---|---|---|
| CM2CON1 | MC1OUT | MC2OUT | C1RSEL | C2RSEL | C1HYS | C2HYS | C1SYNC | C2SYNC | 248 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 245 |
| SRCON0 | SRLEN | SRCLK2 | SRCLK1 | SRCLK0 | SRQEN | SRNQEN | SRPS | SRPR | 248 |
| SRCON1 | SRSPE | SRSCKE | SRSC2E | SRSC1E | SRRPE | SRRCKE | SRRC2E | SRRC1E | 248 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 248 |

**Legend:** Shaded cells are not used with the SR Latch module.


© 2009-2016 Microchip Technology Inc.

## 20.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of VDD, with 1.024V, 2.048V or 4.096V selectable output levels. The output of the FVR can be configured to supply a reference voltage to the following:

- ADC input channel
- ADC positive reference
- Comparator positive input
- Digital-to-Analog Converter (DAC)

The FVR can be enabled by setting the FVR1EN bit of the VREFCON0 register.

### 20.1 Independent Gain Amplifiers

The output of the FVR supplied to the ADC, Comparators and DAC is routed through an independent programmable gain amplifier. The amplifier can be configured to amplify the 1.024V reference voltage by 1x, 2x or 4x, to produce the three possible voltage levels.

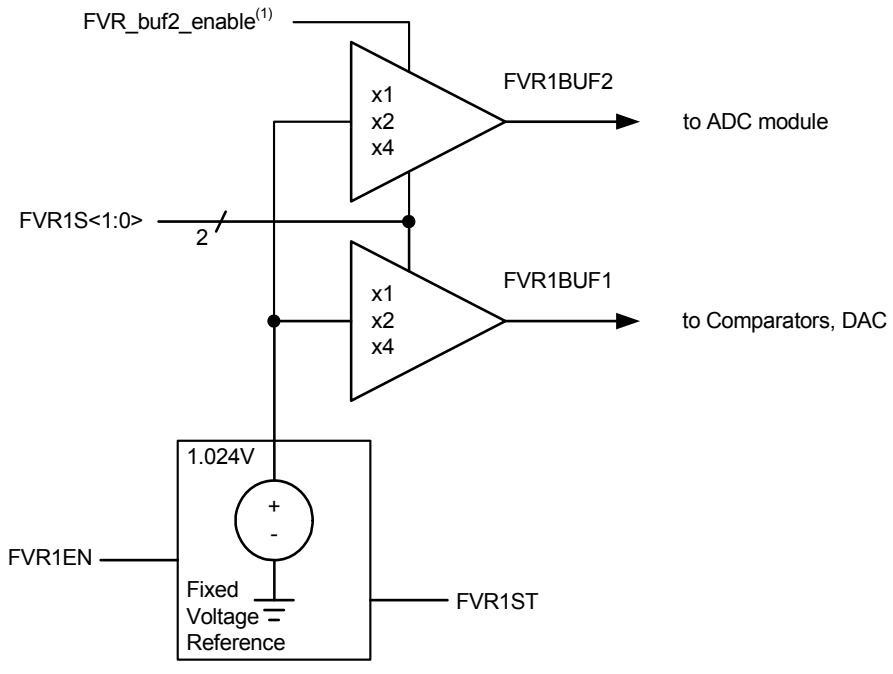
The FVR1S<1:0> bits of the VREFCON0 register are used to enable and configure the gain amplifier settings for the reference supplied to the DAC and Comparator modules. When the ADC module is configured to use the FVR output, (FVR1BUF2) the reference is buffered through an additional unity gain amplifier. This buffer is disabled if the ADC is not configured to use the FVR.

For specific use of the FVR, refer to the specific module sections: **Section 16.0 “Analog-to-Digital Converter (ADC) Module”**, **Section 21.0 “Digital-to-Analog Converter (DAC) Module”** and **Section 17.0 “Comparator Module”**.

### 20.2 FVR Stabilization Period

When the Fixed Voltage Reference module is enabled, it requires time for the reference and amplifier circuits to stabilize. Once the circuits stabilize and are ready for use, the FVR1ST bit of the VREFCON0 register will be set. See Table 26-23 for the minimum delay requirement.

**FIGURE 20-1: VOLTAGE REFERENCE BLOCK DIAGRAM**




Note 1: FVR_buf2_enable = ‘1’ when (ADON = ‘1’)AND [(PVCFG<1:0> = ‘10’) OR ( CHS<4:0> = ‘11111’)]

## 20.3 Register Definitions: FVR Control

**REGISTER 20-1: VREFCON0: FIXED VOLTAGE REFERENCE CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-1 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| FVR1EN | FVR1ST | FVR1S<1:0> | | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **FVR1EN:** Fixed Voltage Reference Enable bit
0 = Fixed Voltage Reference is disabled
1 = Fixed Voltage Reference is enabled

bit 6 **FVR1ST:** Fixed Voltage Reference Ready Flag bit
0 = Fixed Voltage Reference output is not ready or not enabled
1 = Fixed Voltage Reference output is ready for use

bit 5-4 **FVR1S<1:0>:** Fixed Voltage Reference Selection bits
00 = Fixed Voltage Reference Peripheral output is off
01 = Fixed Voltage Reference Peripheral output is 1x (1.024V)
10 = Fixed Voltage Reference Peripheral output is 2x (2.048V)(1)
11 = Fixed Voltage Reference Peripheral output is 4x (4.096V)(1)

bit 3-2 **Reserved:** Read as '0'. Maintain these bits clear.

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

**Note 1:** Fixed Voltage Reference output cannot exceed VDD.

**TABLE 20-1: SUMMARY OF REGISTERS ASSOCIATED WITH FIXED VOLTAGE REFERENCE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page |
|---|---|---|---|---|---|---|---|---|---|
| VREFCON0 | FVR1EN | FVR1ST | FVR1S<1:0> | | — | — | — | — | 232 |

**Legend:** — = unimplemented locations, read as '0'. Shaded bits are not used by the FVR module.


© 2009-2016 Microchip Technology Inc.

## 21.0 DIGITAL-TO-ANALOG CONVERTER (DAC) MODULE

The Digital-to-Analog Converter supplies a variable voltage reference, ratiometric with the input source, with 32 selectable output levels.

The input of the DAC can be connected to:

- External VREF pins
- VDD supply voltage
- FVR (Fixed Voltage Reference)

The output of the DAC can be configured to supply a reference voltage to the following:

- Comparator positive input
- ADC input channel
- DAC1OUT pin

The Digital-to-Analog Converter (DAC) can be enabled by setting the D1EN bit of the VREFCON1 register.

### 21.1 Output Voltage Selection

The DAC has 32 voltage level ranges. The 32 levels are set with the DAC1R<4:0> bits of the VREFCON2 register.

The DAC output voltage is determined by the following equations:

**EQUATION 21-1: DAC OUTPUT VOLTAGE**

$$V_{OUT} = \left( (V_{SRC+} - V_{SRC-}) \times \frac{DACR<4:0>}{2^5} \right) + V_{SRC-}$$

$$V_{SRC+} = V_{DD}, V_{REF+} \text{ or } FVR1$$

$$V_{SRC-} = V_{SS} \text{ or } V_{REF-}$$

### 21.2 Ratiometric Output Level

The DAC output value is derived using a resistor ladder with each end of the ladder tied to a positive and negative voltage reference input source. If the voltage of either input source fluctuates, a similar fluctuation will result in the DAC output value.

The value of the individual resistors within the ladder can be found in **Section 26.0 "Electrical Specifications"**.

### 21.3 Low-Power Voltage State

In order for the DAC module to consume the least amount of power, one of the two voltage reference input sources to the resistor ladder must be disconnected. Either the positive voltage source, (VSRC+), or the negative voltage source, (VSRC-) can be disabled.

The negative voltage source is disabled by setting the D1LPS bit in the VREFCON1 register. Clearing the D1LPS bit in the VREFCON1 register disables the positive voltage source.

### 21.4 Output Clamped to Positive Voltage Source

The DAC output voltage can be set to VSRC+ with the least amount of power consumption by performing the following:

- Clearing the D1EN bit in the VREFCON1 register.
- Setting the D1LPS bit in the VREFCON1 register.
- Configuring the D1PSS bits to the proper positive source.
- Configuring the DAC1Rx bits to '11111' in the VREFCON2 register.

This is also the method used to output the voltage level from the FVR to an output pin. See **Section 21.6 "DAC Voltage Reference Output"** for more information.

### 21.5 Output Clamped to Negative Voltage Source

The DAC output voltage can be set to VSRC- with the least amount of power consumption by performing the following:

- Clearing the D1EN bit in the VREFCON1 register.
- Clearing the DAC1R bit in the VREFCON1 register.
- Configuring the D1PSS bits to the proper negative source.
- Configuring the DAC1Rx bits to '00000' in the VREFCON2 register.

This allows the comparator to detect a zero-crossing while not consuming additional current through the DAC module.

### 21.6 DAC Voltage Reference Output

The DAC can be output to the DAC1OUT (CVREF) pin by setting the DAC1OE bit of the VREFCON1 register to '1'. Selecting the DAC reference voltage for output on the DAC1OUT pin automatically overrides the digital output buffer and digital input threshold detector functions of that pin. Reading the DAC1OUT pin when it has been configured for DAC reference voltage output will always return a '0'.

Due to the limited current drive capability, a buffer must be used on the DAC voltage reference output for external connections to DAC1OUT. Figure 21-2 shows an example buffering technique.

**FIGURE 21-1: DIGITAL-TO-ANALOG CONVERTER BLOCK DIAGRAM**

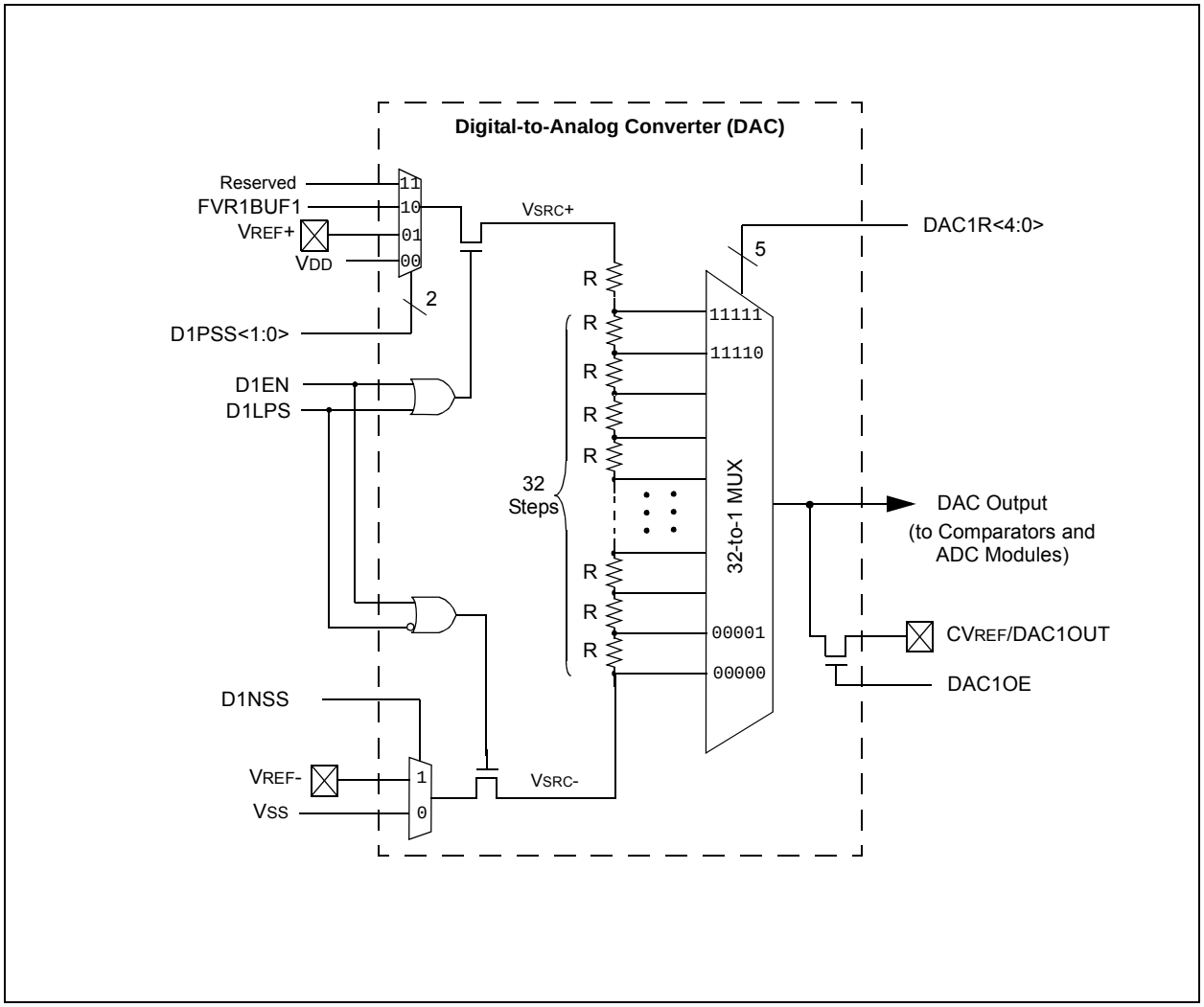
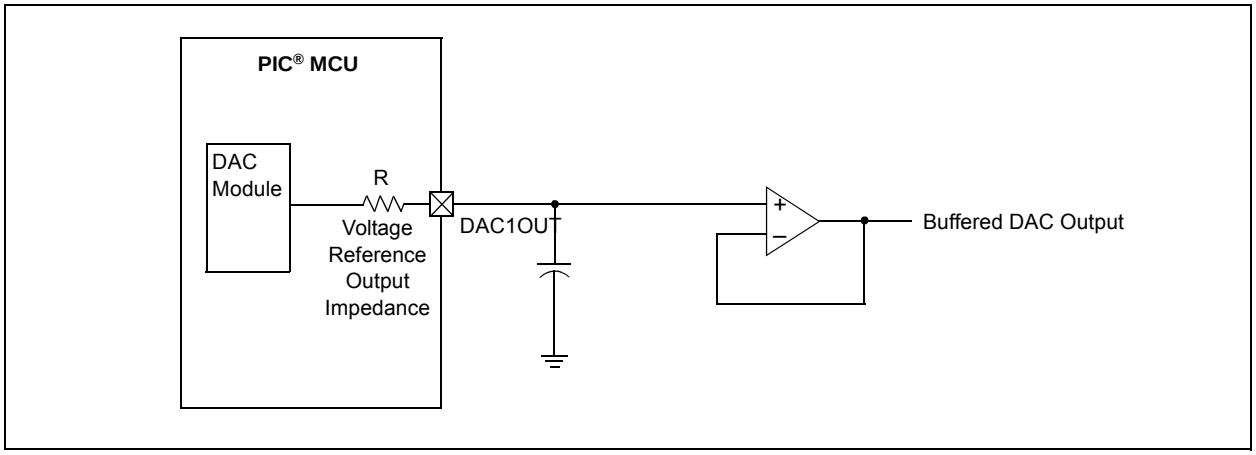



**FIGURE 21-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE**





© 2009-2016 Microchip Technology Inc.

## 21.7 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the VREFCON1 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

## 21.8 Effects of a Reset

A device Reset affects the following:

- DAC is disabled
- DAC output voltage is removed from the DAC1OUT pin
- The DAC1R<4:0> range select bits are cleared

## 21.9 Register Definitions: DAC Control

**REGISTER 21-1: VREFCON1: VOLTAGE REFERENCE CONTROL REGISTER 0**

| 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 | — | D1PSS<1:0> | | — | D1NSS |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **D1EN:** DAC Enable bit
1 = DAC is enabled
0 = DAC is disabled

bit 6 **D1LPS:** DAC Low-Power Voltage Source Select bit
1 = DAC Positive reference source selected
0 = DAC Negative reference source selected

bit 5 **DAC1OE:** DAC Voltage Output Enable bit
1 = DAC voltage level is also an output on the DAC1OUT ($CV_{REF}$) pin
0 = DAC voltage level is disconnected from the DAC1OUT ($CV_{REF}$) pin

bit 4 **Unimplemented:** Read as ‘0’

bit 3-2 **D1PSS<1:0>:** DAC Positive Source Select bits
00 = $V_{DD}$
01 = $V_{REF}+$
10 = FVR1BUF1 output
11 = Reserved, do not use

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **D1NSS:** DAC Negative Source Select bits
1 = $V_{REF}-$
0 = $V_{SS}$

**REGISTER 21-2: VREFCON2: VOLTAGE REFERENCE CONTROL REGISTER 1**

| U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | DAC1R<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4-0 **DAC1R<4:0>:** DAC Voltage Output Select bits

$V_{OUT} = ((V_{SRC}+) - (V_{SRC}-))*(DAC1R<4:0>/(2^5)) + V_{SRC}-$

**TABLE 21-1: REGISTERS ASSOCIATED WITH DAC MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page |
|---|---|---|---|---|---|---|---|---|---|
| VREFCON0 | FVR1EN | FVR1ST | FVR1S<1:0> | | — | — | — | — | 232 |
| VREFCON1 | D1EN | D1LPS | DAC1OE | — | D1PSS<1:0> | | — | D1NSS | 235 |
| VREFCON2 | — | — | — | DAC1R<4:0> | | | | | 236 |

**Legend:** — = Unimplemented locations, read as ‘0’. Shaded bits are not used by the DAC module.


© 2009-2016 Microchip Technology Inc.

## 22.0 RESET

The PIC18(L)F1XK22 devices differentiate between various kinds of Reset:

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

This section discusses Resets generated by $\overline{\text{MCLR}}$, POR and BOR and covers the operation of the various start-up timers. Stack Reset events are covered in **Section 3.1.2.4 "Stack Overflow and Underflow Resets"**. WDT Resets are covered in **Section 23.2 "Watchdog Timer (WDT)"**.

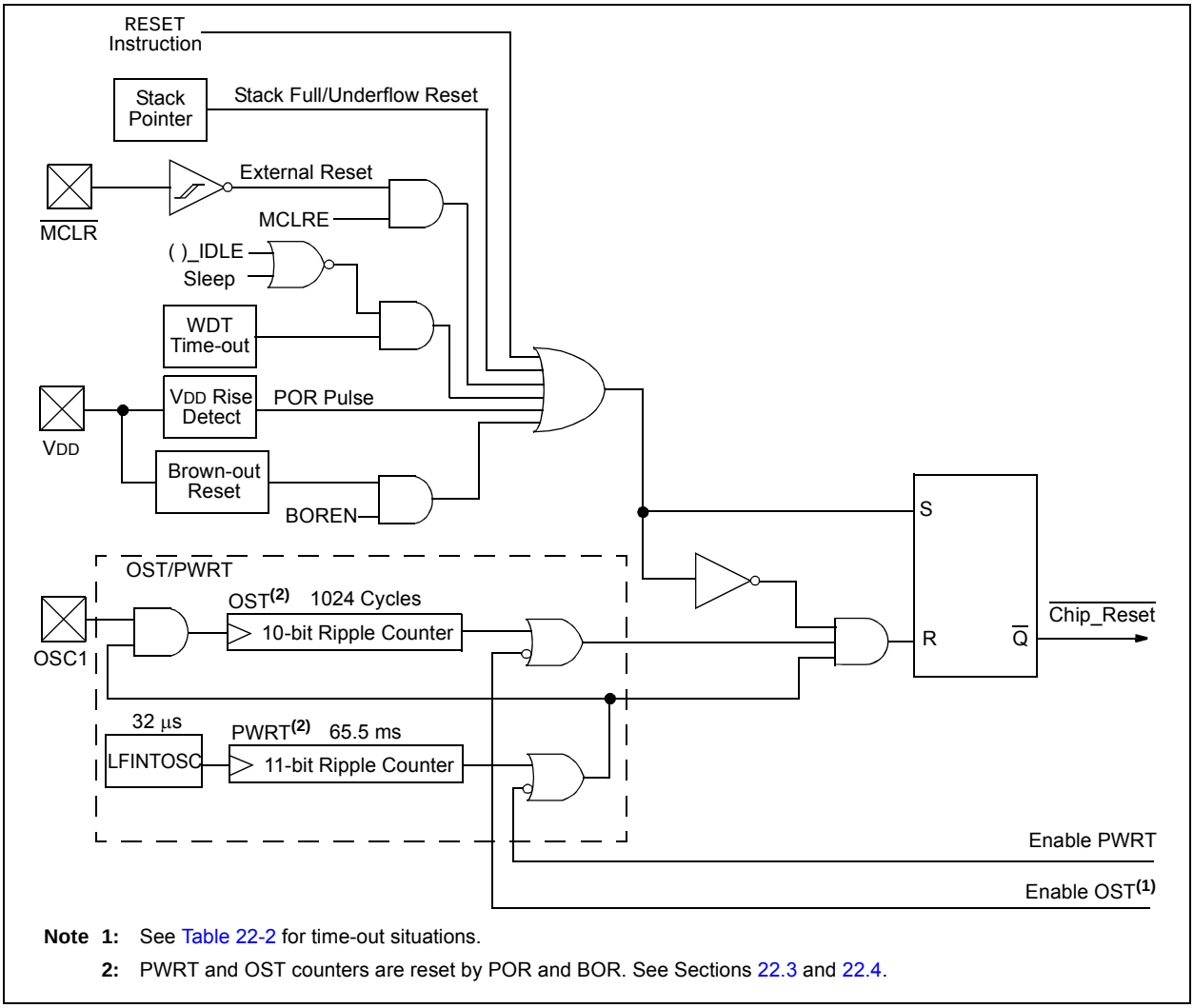
A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 22-1.

### 22.1 RCON Register

Device Reset events are tracked through the RCON register (Register 22-1). 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 22.6 "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 22.4 "Brown-out Reset (BOR)"**.

**FIGURE 22-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**




**Note 1:** See Table 22-2 for time-out situations.
**2:** PWRT and OST counters are reset by POR and BOR. See Sections 22.3 and 22.4.

**REGISTER 22-1: 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(1) | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$(2) | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

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

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

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

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

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
1 = The RESET instruction was not executed (set by firmware 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 **$\overline{TO}$:** Watchdog Time-out Flag bit
1 = Set by power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
1 = Set by power-up or by the CLRWDT instruction
0 = Set by execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit(2)
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit(3)
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 $\overline{POR}$ is determined by the type of device Reset. See the notes following this register and **Section 22.6 "Reset State of Registers"** for additional information.
**3:** See Table 22-3.


© 2009-2016 Microchip Technology Inc.

## 22.2 Master Clear ($\overline{MCLR}$)

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

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

In PIC18(L)F1XK22 devices, the $\overline{MCLR}$ input can be disabled with the MCLRE Configuration bit. When $\overline{MCLR}$ is disabled, the pin becomes a digital input. See **Section 8.1 "PORTA, TRISA and LATA Registers"** for more information.

## 22.3 Power-on Reset (POR)

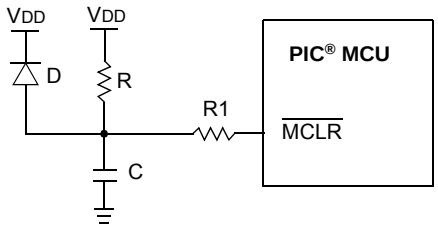
A Power-on Reset pulse is generated on-chip whenever VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.

To take advantage of the POR circuitry, tie the $\overline{MCLR}$ pin through a resistor (1 kΩ to 10 kΩ) to VDD. This will eliminate external RC components usually needed to create a Power-on Reset delay.

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

POR events are captured by the $\overline{POR}$ 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. $\overline{POR}$ 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 22-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)**




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

**2:** R < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

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

## 22.4 Brown-out Reset (BOR)

PIC18(L)F1XK22 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 22-1.

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 $V_{DD}$ below $V_{BOR}$ for greater than $T_{BOR}$ will reset the device. A Reset may or may not occur if $V_{DD}$ falls below $V_{BOR}$ for less than $T_{BOR}$. The chip will remain in Brown-out Reset until $V_{DD}$ rises above $V_{BOR}$.

If the Power-up Timer is enabled, it will be invoked after $V_{DD}$ rises above $V_{BOR}$; it then will keep the chip in Reset for an additional time delay, $T_{PWRT}$. If $V_{DD}$ drops below $V_{BOR}$ 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 $V_{DD}$ rises above $V_{BOR}$, 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.

### 22.4.1 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. |
|---|---|

### 22.4.2 DETECTING BOR

When BOR is enabled, the $\overline{BOR}$ 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 $\overline{BOR}$ alone. A more reliable method is to simultaneously check the state of both $\overline{POR}$ and $\overline{BOR}$. This assumes that the $\overline{POR}$ and $\overline{BOR}$ bits are reset to '1' by software immediately after any POR event. If $\overline{BOR}$ is '0' while $\overline{POR}$ is '1', it can be reliably assumed that a BOR event has occurred.

### 22.4.3 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 22-1: 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. |


© 2009-2016 Microchip Technology Inc.

## 22.5 Device Reset Timers

PIC18(L)F1XK22 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

### 22.5.1 POWER-UP TIMER (PWRT)

The Power-up Timer (PWRT) of PIC18(L)F1XK22 devices is an 11-bit counter which uses the LFINTOSC source as the clock input. This yields an approximate time interval of 2048 x 32 μs = 65.6 ms. While the PWRT is counting, the device is held in Reset.

The power-up time delay depends on the LFINTOSC clock and will vary from chip-to-chip due to temperature and process variation. See **Section 26.0 "Electrical Specifications"** for details.

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

### 22.5.2 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.

### 22.5.3 PLL LOCK TIME-OUT

With the PLL enabled in its PLL mode, the time-out sequence following a Power-on Reset is slightly different from other oscillator modes. A separate timer is used to provide a fixed time-out that is sufficient for the PLL to lock to the main oscillator frequency. This PLL lock time-out ($T_{PLL}$) is typically 2 ms and follows the oscillator start-up time-out.

### 22.5.4 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 22-3, Figure 22-4, Figure 22-5, Figure 22-6 and Figure 22-7 all depict time-out sequences on power-up, with the Power-up Timer enabled and the device operating in HS Oscillator mode. Figures 22-3 through 22-6 also apply to devices operating in XT or LP modes. For devices in RC mode and with the PWRT disabled, on the other hand, there will be no time-out at all.

Since the time-outs occur from the POR pulse, if $\overline{\text{MCLR}}$ is kept low long enough, all time-outs will expire, after which, bringing $\overline{\text{MCLR}}$ high will allow program execution to begin immediately (Figure 22-5). This is useful for testing purposes or to synchronize more than one PIC18(L)F1XK22 device operating in parallel.

**TABLE 22-2: TIME-OUT IN VARIOUS SITUATIONS**

| Oscillator Configuration | Power-up(2) and Brown-out | | Exit from Power-Managed Mode |
|---|---|---|---|
| | $\overline{\text{PWRTEN}}$ = 0 | $\overline{\text{PWRTEN}}$ = 1 | |
| HSPLL | 66 ms(1) + 1024 TOSC + 2 ms(2) | 1024 TOSC + 2 ms(2) | 1024 TOSC + 2 ms(2) |
| HS, XT, LP | 66 ms(1) + 1024 TOSC | 1024 TOSC | 1024 TOSC |
| EC, ECIO | 66 ms(1) | — | — |
| RC, RCIO | 66 ms(1) | — | — |
| INTIO1, INTIO2 | 66 ms(1) | — | — |

**Note 1:** 66 ms (65.5 ms) is the nominal Power-up Timer (PWRT) delay.
**2:** 2 ms is the nominal time required for the PLL to lock.

**FIGURE 22-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO $V_{DD}$, $V_{DD}$ RISE < $T_{PWRT}$)**

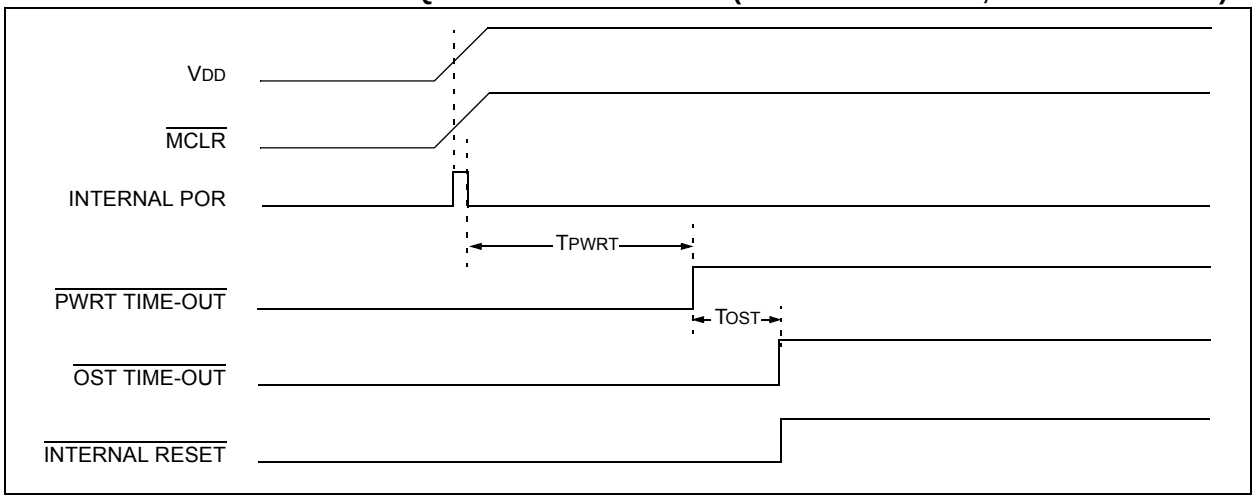



**FIGURE 22-4: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO $V_{DD}$): CASE 1**

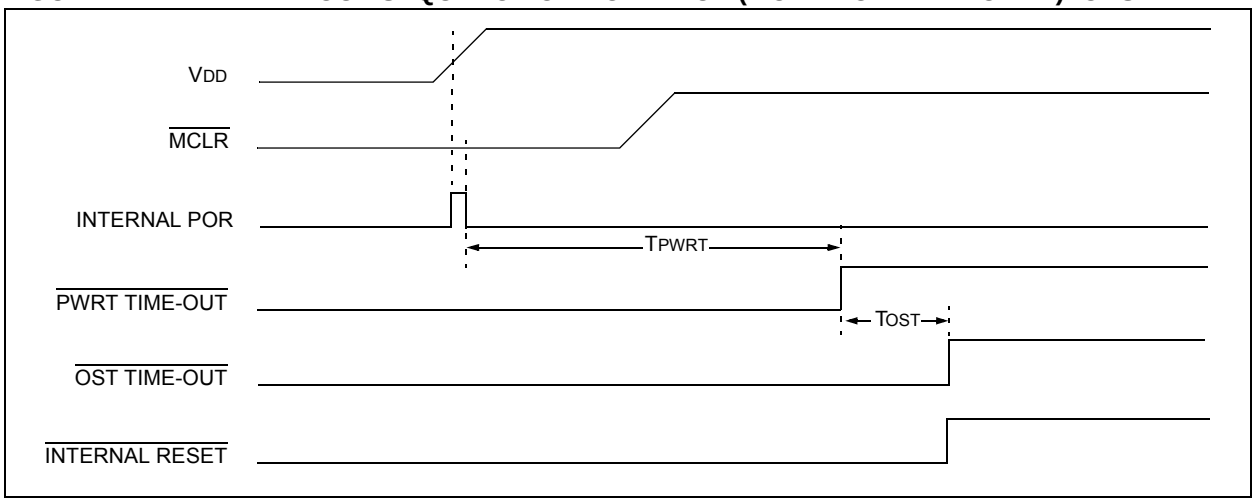
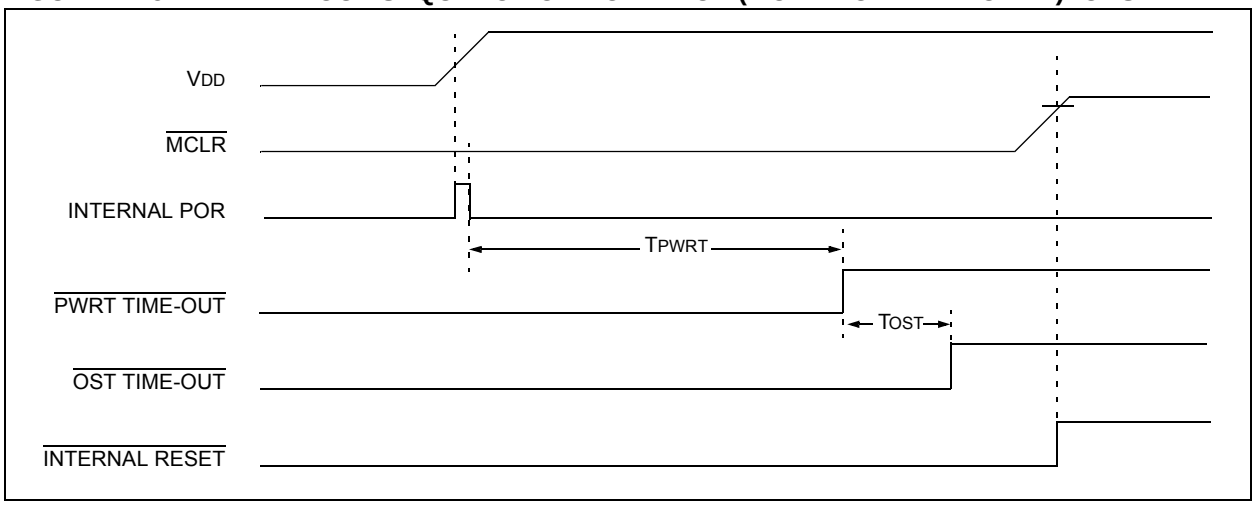



**FIGURE 22-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO $V_{DD}$): CASE 2**





© 2009-2016 Microchip Technology Inc.

**FIGURE 22-6: SLOW RISE TIME ($\overline{MCLR}$ TIED TO $V_{DD}$, $V_{DD}$ RISE > $T_{PWRT}$)**

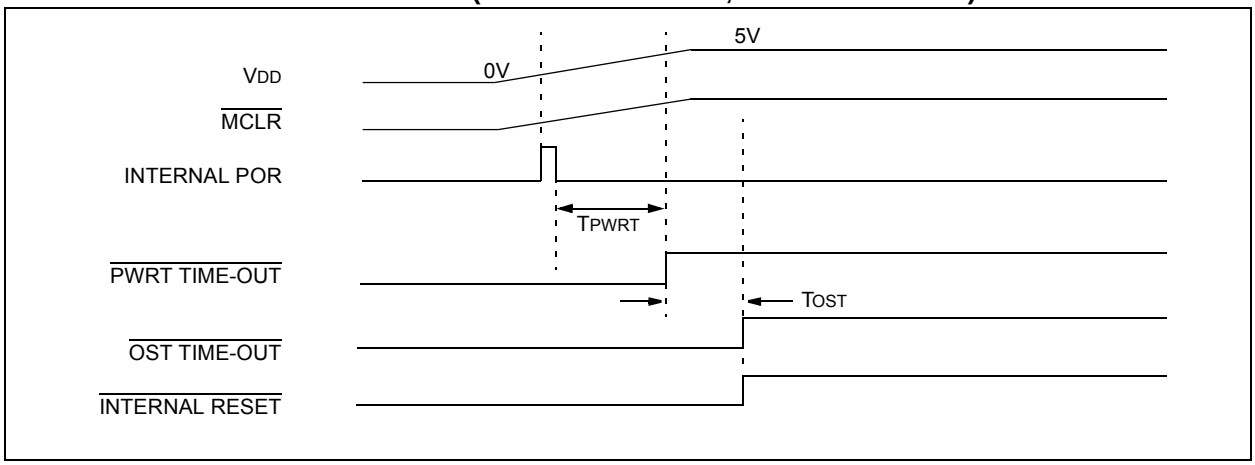
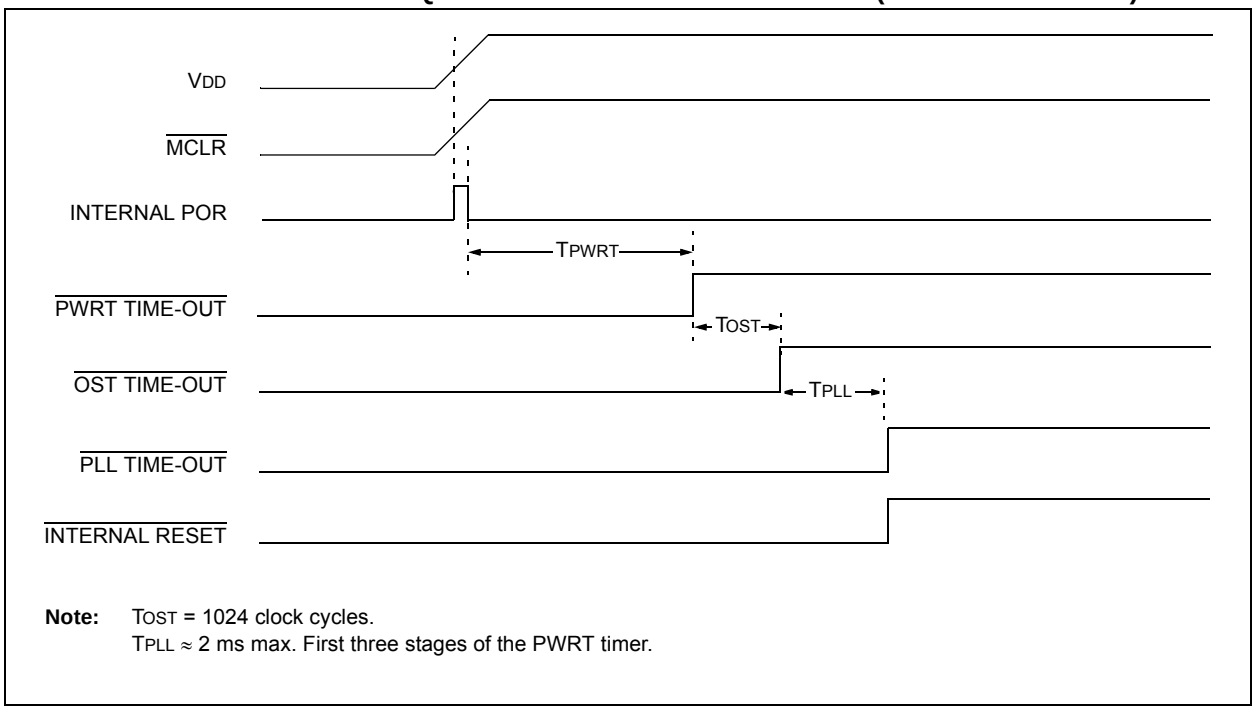



**FIGURE 22-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED ($\overline{MCLR}$ TIED TO $V_{DD}$)**




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

## 22.6 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 22-3. These bits are used by software to determine the nature of the Reset.

Table 22-4 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 22-3: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER**

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

**Legend:** u = unchanged

**Note 1:** When the wake-up is due to an interrupt and the GIEH or GIEL bits are set, the PC is loaded with the interrupt vector (008h or 0018h).

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


© 2009-2016 Microchip Technology Inc.

**TABLE 22-4: 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(3) |
| TOSH | FFEh | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| TOSL | FFDh | 0000 0000 | 0000 0000 | uuuu uuuu(3) |
| STKPTR | FFCh | 00-0 0000 | uu-0 0000 | uu-u uuuu(3) |
| PCLATU | FFBh | ---0 0000 | ---0 0000 | ---u uuuu |
| PCLATH | FFAh | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCL | FF9h | 0000 0000 | 0000 0000 | PC + 2(2) |
| 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(1) |
| INTCON2 | FF1h | 1111 -1-1 | 1111 -1-1 | uuuu -u-u(1) |
| INTCON3 | FF0h | 11-0 0-00 | 11-0 0-00 | uu-u u-uu(1) |
| 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 22-3 for Reset value for specific condition.

**TABLE 22-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Address | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ 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(4) | 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 22-3 for Reset value for specific condition.


© 2009-2016 Microchip Technology Inc.

**TABLE 22-4: 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 |
| VREFCON2 | FBCh | ---0 0000 | ---0 0000 | ---u uuuu |
| VREFCON1 | FBBh | 000- 00-0 | 000- 00-0 | uuu- uu-u |
| VREFCON0 | FBAh | 0001 ---- | 0001 ---- | uuuu ---- |
| 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 | 0-00 0000 | u-uu uuuu | u-uu 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 |
| 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 22-3 for Reset value for specific condition.

**TABLE 22-4: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Address | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|
| IPR2 | FA2h | 1111 1-1- | 1111 1-1- | uuuu u-u- |
| PIR2 | FA1h | 0000 0-0- | 0000 0-0- | uuuu u-u-(1) |
| PIE2 | FA0h | 0000 0-0- | 0000 0-0- | uuuu u-u- |
| 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 1111 | --11 1111 | --uu uuuu |
| LATC | F8Bh | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATB | F8Ah | xxxx ---- | uuuu ---- | uuuu ---- |
| LATA | F89h | --xx xxxx | --uu uuuu | --uu uuuu |
| PORTC | F82h | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTB | F81h | xxxx ---- | uuuu ---- | uuuu ---- |
| PORTA | F80h | --xx xxxx | --xx xxxx | --uu uuuu |
| ANSELH | F7Fh | ---- 1111 | ---- 1111 | ---- uuuu |
| ANSEL | F7Eh | 1111 1111 | 1111 1111 | uuuu uuuu |
| IOCB | F7Ah | 0000 ---- | 0000 ---- | uuuu ---- |
| IOCA | F79h | --00 0000 | --00 0000 | --uu uuuu |
| WPUB | F78h | 1111 ---- | 1111 ---- | uuuu ---- |
| WPUA | F77h | --11 1111 | --11 1111 | --uu uuuu |
| 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 |

**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 22-3 for Reset value for specific condition.


© 2009-2016 Microchip Technology Inc.

## 23.0 SPECIAL FEATURES OF THE CPU

PIC18(L)F1XK22 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, PIC18(L)F1XK22 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.

© 2009-2016 Microchip Technology Inc.

## 23.1 Configuration Bits

The Configuration bits can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh), which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The WR bit in the EECON1 register starts a self-timed write to the Configuration register. In normal operation mode, a TBLWT instruction with the TBLPTR pointing to the Configuration register sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a TBLWT instruction can write a '1' or a '0' into the cell. For additional details on Flash programming, refer to **Section 4.5 "Writing to Flash Program Memory"**.

**TABLE 23-1: CONFIGURATION BITS AND DEVICE IDs**

| File Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Default/ Unprogrammed Value |
|---|---|---|---|---|---|---|---|---|---|---|
| 300001h | CONFIG1H | IESO | FCMEN | PCLKEN | PLL_EN | FOSC3 | FOSC2 | FOSC1 | FOSC0 | 0010 0111 |
| 300002h | CONFIG2L | — | — | — | BORV1 | BORV0 | BOREN1 | BOREN0 | $\overline{\text{PWRTEN}}$ | ---1 1111 |
| 300003h | CONFIG2H | — | — | — | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN | ---1 1111 |
| 300005h | CONFIG3H | MCLRE | — | — | — | HFOFST | — | — | — | 1--- 1--- |
| 300006h | CONFIG4L | $\overline{\text{BKBUG}}$ | ENHCPU | — | — | 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 | WRTC | — | — | — | — | — | 111- ---- |
| 30000Ch | CONFIG7L | — | — | — | — | — | — | EBTR1 | EBTR0 | ---- --11 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — | -1-- ---- |
| 3FFFFEh | DEVID1(1) | DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 | qqqq qqqq(1) |
| 3FFFFFh | DEVID2(1) | 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 23-12 for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.


© 2009-2016 Microchip Technology Inc.

**REGISTER 23-1: 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 | PLL_EN | 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 **IESO:** Internal/External Oscillator Switchover bit
1 = Oscillator Switchover mode enabled
0 = Oscillator Switchover mode disabled

bit 6 **FCMEN:** Fail-Safe Clock Monitor Enable bit
1 = Fail-Safe Clock Monitor enabled
0 = Fail-Safe Clock Monitor disabled

bit 5 **PCLKEN:** Primary Clock Enable bit
1 = Primary Clock enabled
0 = Primary Clock is under software control

bit 4 **PLL_EN:** 4 X PLL Enable bit
1 = Oscillator multiplied by 4
0 = PLL is under software control

bit 3-0 **FOSC<3:0>:** 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

**REGISTER 23-2: 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(1) | BORV0(1) | BOREN1(2) | BOREN0(2) | $\overline{\text{PWRTEN}}$(2) |
| 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 4-3 **BORV<1:0>:** Brown-out Reset Voltage bits(1)
- 11 = $V_{BOR}$ set to 1.9V nominal
- 10 = $V_{BOR}$ set to 2.2V nominal
- 01 = $V_{BOR}$ set to 2.5V nominal
- 00 = $V_{BOR}$ set to 2.85V nominal

bit 2-1 **BOREN<1:0>:** Brown-out Reset Enable bits(2)
- 11 = Brown-out Reset enabled in hardware only (SBOREN is disabled)
- 10 = Brown-out Reset enabled in hardware only and disabled in Sleep mode (SBOREN is disabled)
- 01 = Brown-out Reset enabled and controlled by software (SBOREN is enabled)
- 00 = Brown-out Reset disabled in hardware and software

bit 0 **$\overline{\text{PWRTEN}}$:** Power-up Timer Enable bit(2)
- 1 = PWRT disabled
- 0 = PWRT enabled

**Note 1:** See Table 26-1 for specifications.
**2:** The Power-up Timer is decoupled from Brown-out Reset, allowing these features to be independently controlled.


© 2009-2016 Microchip Technology Inc.

**REGISTER 23-3: 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

**REGISTER 23-4: 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:** $\overline{\text{MCLR}}$ Pin Enable bit
1 = $\overline{\text{MCLR}}$ pin enabled; RA3 input pin disabled
0 = RA3 input pin enabled; $\overline{\text{MCLR}}$ 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 23-5: CONFIG4L: CONFIGURATION REGISTER 4 LOW**

| R/W-1(1) | R/W-0 | U-0 | U-0 | R/P-0 | R/P-1 | U-0 | R/P-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{BKBUG}}$ | ENHCPU | — | — | 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 **$\overline{\text{BKBUG}}$:** Background Debugger Enable bit(1)
1 = Background Debugger disabled
0 = Background Debugger functions enabled

bit 6 **ENHCPU:** Enhanced CPU Enable bit
1 = Enhanced CPU enabled
0 = Enhanced CPU disabled

bit 5-4 **Unimplemented:** Read as '0'

bit 3 **BBSIZ:** Boot BLock Size Select bit
1 = 2 kW boot block size for PIC18(L)F14K22 (1 kW boot block size for PIC18(L)F13K22)
0 = 1 kW boot block size for PIC18(L)F14K22 (512 W boot block size for PIC18(L)F13K22)

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

**Note 1:** $\overline{\text{BKBUG}}$ is only used for ICD device. Otherwise, this bit is unimplemented and reads as '1'.


© 2009-2016 Microchip Technology Inc.

**REGISTER 23-6: 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 23-7: 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’

**REGISTER 23-8: 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 23-9: 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 | WRTC(1) | — | — | — | — | — |
| 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 **WRTC:** Configuration Register Write Protection bit(1)
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.


© 2009-2016 Microchip Technology Inc.

**REGISTER 23-10: 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 23-11: 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'

**REGISTER 23-12: DEVID1: DEVICE ID REGISTER 1 FOR PIC18(L)F1XK22**

| 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 = PIC18(L)F13K22
011 = PIC18(L)F14K22

bit 4-0 **REV<4:0>:** Revision ID bits
These bits are used to indicate the device revision.

**REGISTER 23-13: DEVID2: DEVICE ID REGISTER 2 FOR PIC18(L)F1XK22**

| 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 = PIC18F13K22/PIC18F14K22 devices(1)

**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.


© 2009-2016 Microchip Technology Inc.

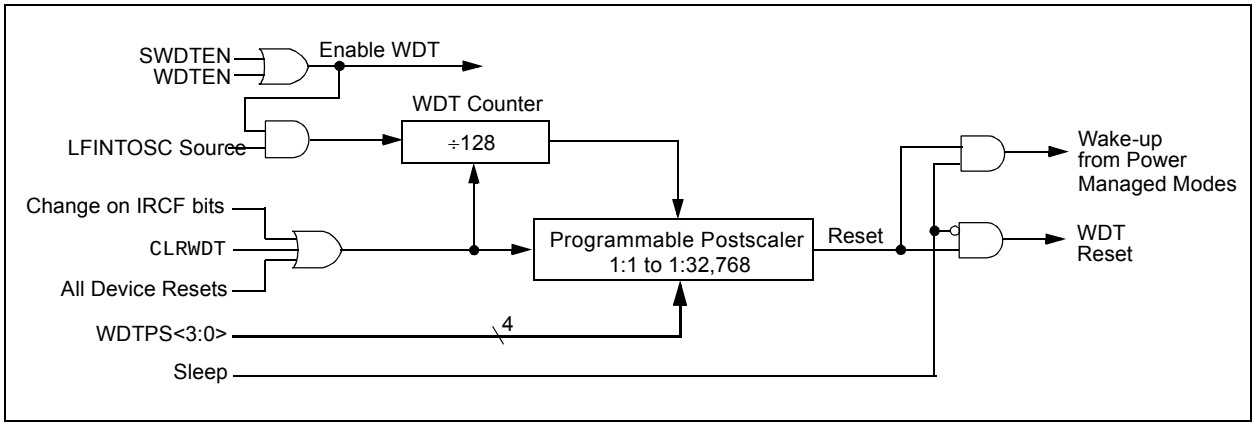
## 23.2 Watchdog Timer (WDT)

For PIC18(L)F1XK22 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-millisecond 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. |

**FIGURE 23-1: WDT BLOCK DIAGRAM**



### 23.2.1 CONTROL REGISTER

Register 23-14 shows the WDTCON register. This is a readable and writable register which contains a control bit that allows software to override the WDT enable Configuration bit, but only if the Configuration bit has disabled the WDT.

**REGISTER 23-14: WDTCON: WATCHDOG TIMER CONTROL REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | SWDTEN(1) |
| bit 7 | | | | | | | bit 0 |

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

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

bit 0 **SWDTEN:** Software Enable or Disable the Watchdog Timer bit(1)
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 23-2: SUMMARY OF WATCHDOG TIMER REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on page |
|---|---|---|---|---|---|---|---|---|---|
| CONFIG2H | — | — | — | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN | 253 |
| RCON | IPEN | SBOREN | — | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 246 |
| WDTCON | — | — | — | — | — | — | — | SWDTEN | 246 |

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

## 23.3 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 23-2 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 23-3.


© 2009-2016 Microchip Technology Inc.

**FIGURE 23-2: CODE-PROTECTED PROGRAM MEMORY FOR PIC18(L)F1XK22**

| Address (from/to) | Device | | | |
|---|---|---|---|---|
| | 14K22 | | 13K22 | |
| | BBSIZ = 1 | BBSIZ = 0 | BBSIZ = 1 | BBSIZ = 0 |
| 0000h<br>03FFh | Boot Block, 4 KB<br>CPB, WRTB, EBTRB | Boot Block, 2 KB<br>CPB, WRTB, EBTRB | Boot Block, 2 KB<br>CPB, WRTB, EBTRB | Boot Block, 1 KB CPB,<br>WRTB, EBTRB |
| 0400h<br>07FFh | | | | Block 0<br>1.512 KB<br>CP0, WRT0, EBTR0 |
| 0800h<br>0BFFh | | Block 0<br>6 KB<br>CP0, WRT0, EBTR0 | Block 0<br>2 KB<br>CP0, WRT0, EBTR0 | |
| 0C00h<br>0FFFh | | | | |
| 1000h<br>1FFFh | Block 0<br>4 KB<br>CP0, WRT0, EBTR0 | | Block 1<br>4 KB<br>CP1, WRT1, EBTR1 | Block 1<br>4 KB<br>CP1, WRT1, EBTR1 |
| 2000h<br>3FFFh | Block 1<br>8 KB<br>CP1, WRT1, EBTR1 | Block 1<br>8 KB<br>CP1, WRT1, EBTR1 | Reads all '0's | Reads all '0's |
| 4000h<br>4FFEh | Reads all '0's | Reads all '0's | | |
| 5000h<br>5FFEh | | | | |
| 6000h<br>6FFEh | | | | |
| 7000h<br>7FFEh | | | | |
| 8000h<br>8FFEh | | | | |
| 9000h<br>9FFEh | | | | |
| A000h<br>AFFEh | | | | |
| B000h<br>BFFEh | | | | |
| C000h<br>CFFEh | | | | |
| D000h<br>DFFEh | | | | |
| E000h<br>EFFEh | | | | |
| F000h<br>FFFEh | | | | |
| H000h<br>HFFEh | | | | |

**Note:** Refer to the test section for requirements on test memory mapping.

**TABLE 23-3: SUMMARY OF CODE PROTECTION REGISTERS**

| File Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 300008h | CONFIG5L | — | — | — | — | — | — | CP1 | CP0 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — |
| 30000Ah | CONFIG6L | — | — | — | — | — | — | WRT1 | WRT0 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC(1) | — | — | — | — | — |
| 30000Ch | CONFIG7L | — | — | — | — | — | — | EBTR1 | EBTR0 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — |

**Legend:** Shaded cells are unimplemented.

**Note 1:** Unimplemented in PIC18F13K22 and PIC18F14K22 devices; maintain this bit set.

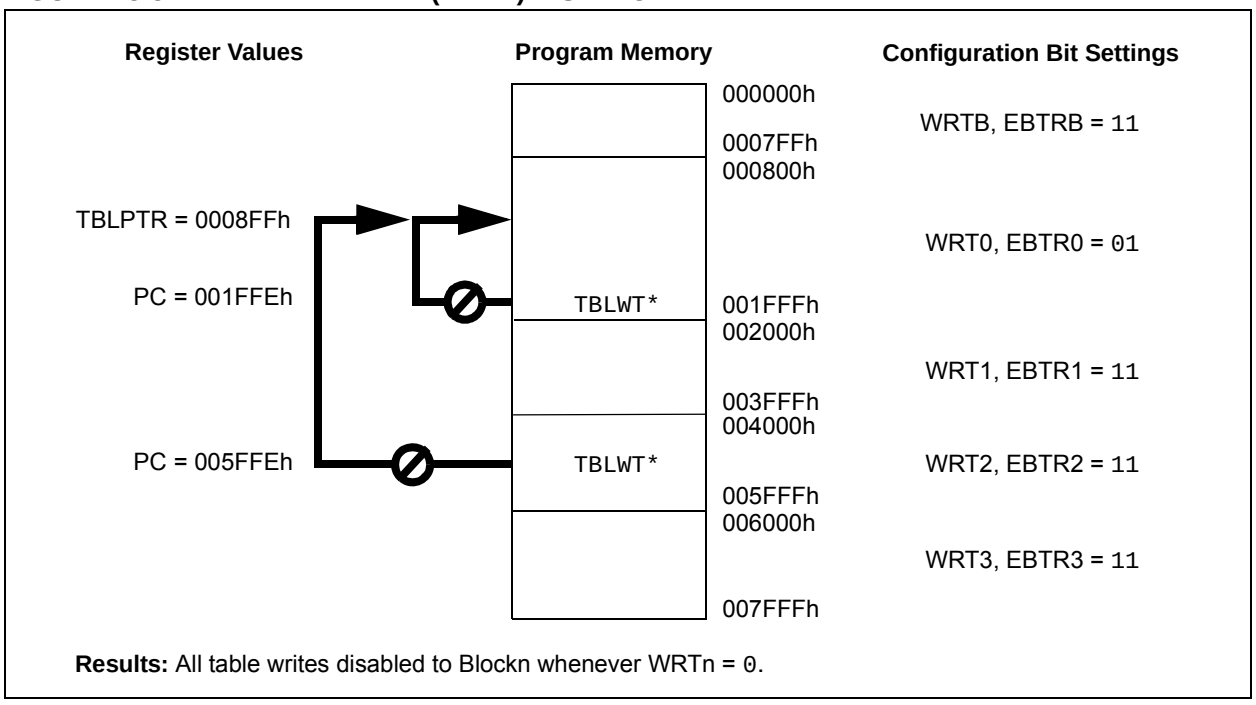
### 23.3.1 PROGRAM MEMORY CODE PROTECTION

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 23-3 through 23-5 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. |

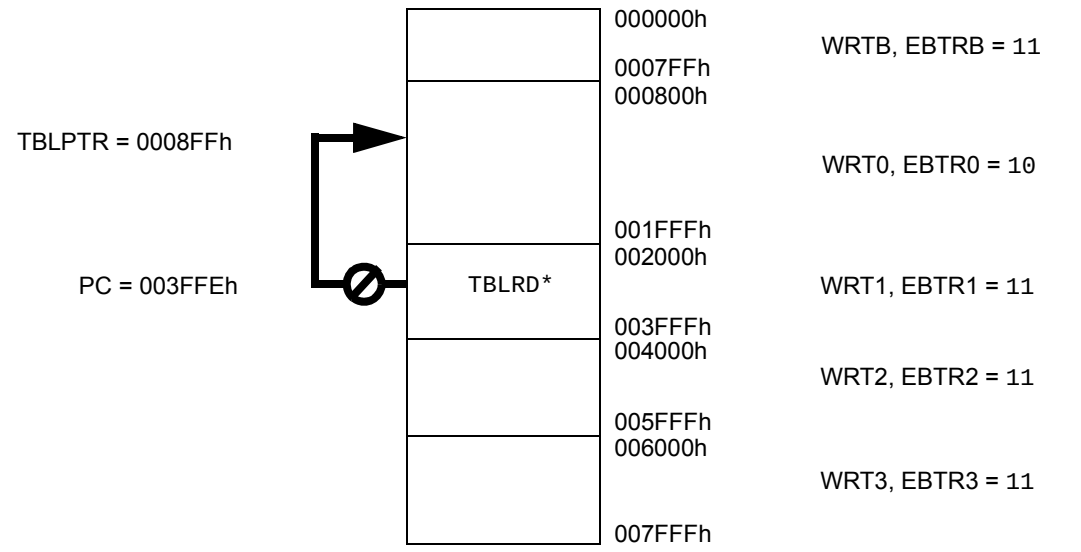
**FIGURE 23-3: TABLE WRITE (WRTn) DISALLOWED**




**Results:** All table writes disabled to Blockn whenever WRTn = 0.

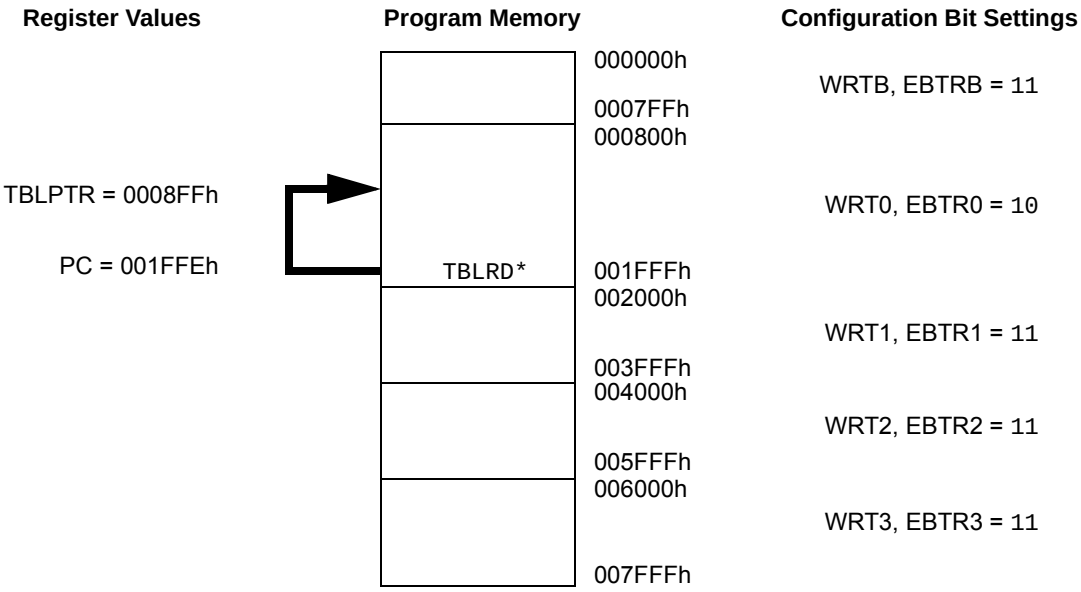

© 2009-2016 Microchip Technology Inc.

**FIGURE 23-4: EXTERNAL BLOCK TABLE READ (EBTRn) DISALLOWED**




**Results:** All table reads from external blocks to Blockn are disabled whenever EBTRn = 0.
TABLAT register returns a value of '0'.

**FIGURE 23-5: EXTERNAL BLOCK TABLE READ (EBTRn) ALLOWED**




**Results:** Table reads permitted within Blockn, even when EBTRBn = 0.
TABLAT register returns the value of the data at the location TBLPTR.

### 23.3.2 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.

### 23.3.3 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.

## 23.4 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.

## 23.5 In-Circuit Serial Programming

PIC18(L)F1XK22 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.

## 23.6 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 23-4 shows which resources are required by the background debugger.

**TABLE 23-4: DEBUGGER RESOURCES**

| | |
|---|---|
| I/O pins: | RA0, RA1 |
| 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:

- $\overline{MCLR}$/VPP/RA3
- VDD
- VSS
- RA0
- RA1

This will interface to the In-Circuit Debugger module available from Microchip or one of the third party development tool companies.

## 23.7 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 $\overline{MCLR}$/VPP/RA3 pin, but the RC3/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 $\overline{MCLR}$/VPP/RA3 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 $\overline{MCLR}$ 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 RC3 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. RC3/PGM then becomes available as the digital I/O pin, RC3. The LVP bit may be set or cleared only when using standard high-voltage programming (VIHH applied to the $\overline{MCLR}$/VPP/RA3 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.


© 2009-2016 Microchip Technology Inc.

## 24.0 INSTRUCTION SET SUMMARY

PIC18(L)F1XK22 devices incorporate the standard set of 75 PIC18 core instructions, as well as an extended set of eight new instructions, for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

### 24.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are four instructions that require two program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18 instruction set summary in Table 24-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 24-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The destination of the result (specified by ‘d’)
3. The accessed memory (specified by ‘a’)

The file register designator ‘f’ specifies which file register is to be used by the instruction. The destination designator ‘d’ specifies where the result of the operation is to be placed. If ‘d’ is zero, the result is placed in the WREG register. If ‘d’ is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The bit in the file register (specified by ‘b’)
3. The accessed memory (specified by ‘a’)

The bit field designator ‘b’ selects the number of the bit affected by the operation, while the file register designator ‘f’ represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by ‘k’)
- The desired FSR register to load the literal value into (specified by ‘f’)
- No operand required (specified by ‘—’)

The **control** instructions may use some of the following operands:

- A program memory address (specified by ‘n’)
- The mode of the CALL or RETURN instructions (specified by ‘s’)
- The mode of the table read and table write instructions (specified by ‘m’)
- No operand required (specified by ‘—’)

All instructions are a single word, except for four double-word instructions. These instructions were made double-word to contain the required information in 32 bits. In the second word, the 4 MSbs are ‘1’s. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1μs. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2μs. Two-word branch instructions (if true) would take 3μs.

Figure 24-1 shows the general formats that the instructions can have. All examples use the convention ‘nnh’ to represent a hexadecimal number.

The Instruction Set Summary, shown in Table 24-2, lists the standard instructions recognized by the Microchip Assembler (MPASM™).

**Section 24.1.1 “Standard Instruction Set”** provides a description of each instruction.

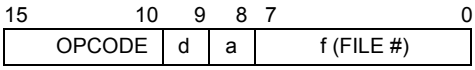
**TABLE 24-1: OPCODE FIELD DESCRIPTIONS**

| Field | Description |
|---|---|
| a | RAM access bit<br>a = 0: RAM location in Access RAM (BSR register is ignored)<br>a = 1: RAM bank is specified by BSR register |
| bbb | Bit address within an 8-bit file register (0 to 7). |
| BSR | Bank Select Register. Used to select the current RAM bank. |
| C, DC, Z, OV, N | ALU Status bits: **C**arry, **D**igit **C**arry, **Z**ero, **Ov**erflow, **N**egative. |
| d | Destination select bit<br>d = 0: store result in WREG<br>d = 1: store result in file register f |
| dest | Destination: either the WREG register or the specified register file location. |
| f | 8-bit Register file address (00h to FFh) or 2-bit FSR designator (0h to 3h). |
| $f_s$ | 12-bit Register file address (000h to FFFh). This is the source address. |
| $f_d$ | 12-bit Register file address (000h to FFFh). This is the destination address. |
| GIE | Global Interrupt Enable bit. |
| k | Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value). |
| label | Label name. |
| mm<br><br>*<br>*+<br>*-<br>+* | The mode of the TBLPTR register for the table read and table write instructions.<br>Only used with table read and table write instructions:<br>No change to register (such as TBLPTR with table reads and writes)<br>Post-Increment register (such as TBLPTR with table reads and writes)<br>Post-Decrement register (such as TBLPTR with table reads and writes)<br>Pre-Increment register (such as TBLPTR with table reads and writes) |
| n | The relative address (2's complement number) for relative branch instructions or the direct address for CALL/BRANCH and RETURN instructions. |
| PC | Program Counter. |
| PCL | Program Counter Low Byte. |
| PCH | Program Counter High Byte. |
| PCLATH | Program Counter High Byte Latch. |
| PCLATU | Program Counter Upper Byte Latch. |
| $\overline{PD}$ | Power-down bit. |
| PRODH | Product of Multiply High Byte. |
| PRODL | Product of Multiply Low Byte. |
| s | Fast Call/Return mode select bit<br>s = 0: do not update into/from shadow registers<br>s = 1: certain registers loaded into/from shadow registers (Fast mode) |
| TBLPTR | 21-bit Table Pointer (points to a Program Memory location). |
| TABLAT | 8-bit Table Latch. |
| $\overline{TO}$ | Time-out bit. |
| TOS | Top-of-Stack. |
| u | Unused or unchanged. |
| WDT | Watchdog Timer. |
| WREG | Working register (accumulator). |
| x | Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools. |
| $z_s$ | 7-bit offset value for indirect addressing of register files (source). |
| $z_d$ | 7-bit offset value for indirect addressing of register files (destination). |
| { } | Optional argument. |
| [text] | Indicates an indexed address. |
| (text) | The contents of text. |
| [expr]<n> | Specifies bit n of the register indicated by the pointer expr. |
| → | Assigned to. |
| < > | Register bit field. |
| ∈ | In the set of. |
| italics | User defined term (font is Courier). |


© 2009-2016 Microchip Technology Inc.

**FIGURE 24-1: GENERAL FORMAT FOR INSTRUCTIONS**

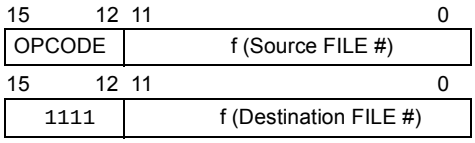
**Byte-oriented** file register operations — **Example Instruction**




`ADDWF MYREG, W, B`

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

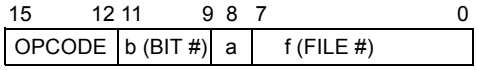
**Byte to Byte** move operations (2-word)




`MOVFF MYREG1, MYREG2`

f = 12-bit file register address

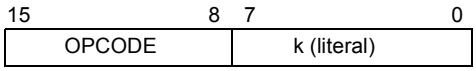
**Bit-oriented** file register operations




`BSF MYREG, bit, B`

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

**Literal** operations

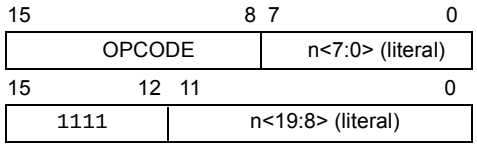



`MOVLW 7Fh`

k = 8-bit immediate value

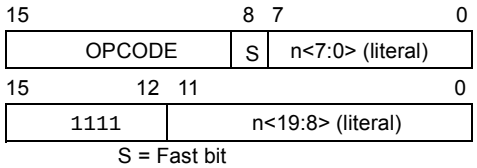
**Control** operations

**CALL, GOTO and Branch** operations



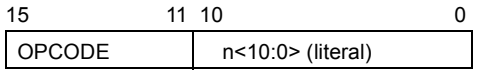

`GOTO Label`

n = 20-bit immediate value

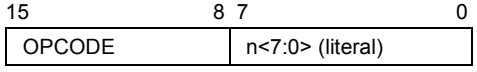



`CALL MYFUNC`

S = Fast bit




`BRA MYFUNC`




`BC MYFUNC`

**TABLE 24-2: PIC18FXXXX INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **BYTE-ORIENTED OPERATIONS** | | | | | | | | | |
| ADDWF | f, d, a | Add WREG and f | 1 | 0010 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ADDWFC | f, d, a | Add WREG and CARRY bit to f | 1 | 0010 | 00da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ANDWF | f, d, a | AND WREG with f | 1 | 0001 | 01da | ffff | ffff | Z, N | 1, 2 |
| CLRF | f, a | Clear f | 1 | 0110 | 101a | ffff | ffff | Z | 2 |
| COMF | f, d, a | Complement f | 1 | 0001 | 11da | ffff | ffff | Z, N | 1, 2 |
| CPFSEQ | f, a | Compare f with WREG, skip = | 1 (2 or 3) | 0110 | 001a | ffff | ffff | None | 4 |
| CPFSGT | f, a | Compare f with WREG, skip > | 1 (2 or 3) | 0110 | 010a | ffff | ffff | None | 4 |
| CPFSLT | f, a | Compare f with WREG, skip < | 1 (2 or 3) | 0110 | 000a | ffff | ffff | None | 1, 2 |
| DECF | f, d, a | Decrement f | 1 | 0000 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| DECFSZ | f, d, a | Decrement f, Skip if 0 | 1 (2 or 3) | 0010 | 11da | ffff | ffff | None | 1, 2, 3, 4 |
| DCFSNZ | f, d, a | Decrement f, Skip if Not 0 | 1 (2 or 3) | 0100 | 11da | ffff | ffff | None | 1, 2 |
| INCF | f, d, a | Increment f | 1 | 0010 | 10da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| INCFSZ | f, d, a | Increment f, Skip if 0 | 1 (2 or 3) | 0011 | 11da | ffff | ffff | None | 4 |
| INFSNZ | f, d, a | Increment f, Skip if Not 0 | 1 (2 or 3) | 0100 | 10da | ffff | ffff | None | 1, 2 |
| IORWF | f, d, a | Inclusive OR WREG with f | 1 | 0001 | 00da | ffff | ffff | Z, N | 1, 2 |
| MOVF | f, d, a | Move f | 1 | 0101 | 00da | ffff | ffff | Z, N | 1 |
| MOVFF | $f_s$, $f_d$ | Move $f_s$ (source) to 1st word<br>$f_d$ (destination) 2nd word | 2 | 1100<br>1111 | ffff<br>ffff | ffff<br>ffff | ffff<br>ffff | None | |
| MOVWF | f, a | Move WREG to f | 1 | 0110 | 111a | ffff | ffff | None | |
| MULWF | f, a | Multiply WREG with f | 1 | 0000 | 001a | ffff | ffff | None | 1, 2 |
| NEGF | f, a | Negate f | 1 | 0110 | 110a | ffff | ffff | C, DC, Z, OV, N | |
| RLCF | f, d, a | Rotate Left f through Carry | 1 | 0011 | 01da | ffff | ffff | C, Z, N | 1, 2 |
| RLNCF | f, d, a | Rotate Left f (No Carry) | 1 | 0100 | 01da | ffff | ffff | Z, N | |
| RRCF | f, d, a | Rotate Right f through Carry | 1 | 0011 | 00da | ffff | ffff | C, Z, N | |
| RRNCF | f, d, a | Rotate Right f (No Carry) | 1 | 0100 | 00da | ffff | ffff | Z, N | |
| SETF | f, a | Set f | 1 | 0110 | 100a | ffff | ffff | None | 1, 2 |
| SUBFWB | f, d, a | Subtract f from WREG with borrow | 1 | 0101 | 01da | ffff | ffff | C, DC, Z, OV, N | |
| SUBWF | f, d, a | Subtract WREG from f | 1 | 0101 | 11da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| SUBWFB | f, d, a | Subtract WREG from f with borrow | 1 | 0101 | 10da | ffff | ffff | C, DC, Z, OV, N | |
| SWAPF | f, d, a | Swap nibbles in f | 1 | 0011 | 10da | ffff | ffff | None | 4 |
| TSTFSZ | f, a | Test f, skip if 0 | 1 (2 or 3) | 0110 | 011a | ffff | ffff | None | 1, 2 |
| XORWF | f, d, a | Exclusive OR WREG with f | 1 | 0001 | 10da | ffff | ffff | Z, N | |

**Note 1:** When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

**2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

**3:** If 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.


© 2009-2016 Microchip Technology Inc.

**TABLE 24-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **BIT-ORIENTED OPERATIONS** | | | | | | | | | |
| BCF | f, b, a | Bit Clear f | 1 | 1001 | bbba | ffff | ffff | None | 1, 2 |
| BSF | f, b, a | Bit Set f | 1 | 1000 | bbba | ffff | ffff | None | 1, 2 |
| BTFSC | f, b, a | Bit Test f, Skip if Clear | 1 (2 or 3) | 1011 | bbba | ffff | ffff | None | 3, 4 |
| BTFSS | f, b, a | Bit Test f, Skip if Set | 1 (2 or 3) | 1010 | bbba | ffff | ffff | None | 3, 4 |
| BTG | f, b, a | Bit Toggle f | 1 | 0111 | bbba | ffff | ffff | None | 1, 2 |
| **CONTROL OPERATIONS** | | | | | | | | | |
| BC | n | Branch if Carry | 1 (2) | 1110 | 0010 | nnnn | nnnn | None | |
| BN | n | Branch if Negative | 1 (2) | 1110 | 0110 | nnnn | nnnn | None | |
| BNC | n | Branch if Not Carry | 1 (2) | 1110 | 0011 | nnnn | nnnn | None | |
| BNN | n | Branch if Not Negative | 1 (2) | 1110 | 0111 | nnnn | nnnn | None | |
| BNOV | n | Branch if Not Overflow | 1 (2) | 1110 | 0101 | nnnn | nnnn | None | |
| BNZ | n | Branch if Not Zero | 1 (2) | 1110 | 0001 | nnnn | nnnn | None | |
| BOV | n | Branch if Overflow | 1 (2) | 1110 | 0100 | nnnn | nnnn | None | |
| BRA | n | Branch Unconditionally | 2 | 1101 | 0nnn | nnnn | nnnn | None | |
| BZ | n | Branch if Zero | 1 (2) | 1110 | 0000 | nnnn | nnnn | None | |
| CALL | k, s | Call subroutine 1st word | 2 | 1110 | 110s | kkkk | kkkk | None | |
| | | 2nd word | | 1111 | kkkk | kkkk | kkkk | | |
| CLRWDT | — | Clear Watchdog Timer | 1 | 0000 | 0000 | 0000 | 0100 | $\overline{TO}$, $\overline{PD}$ | |
| DAW | — | Decimal Adjust WREG | 1 | 0000 | 0000 | 0000 | 0111 | C | |
| GOTO | k | Go to address 1st word | 2 | 1110 | 1111 | kkkk | kkkk | None | |
| | | 2nd word | | 1111 | kkkk | kkkk | kkkk | | |
| NOP | — | No Operation | 1 | 0000 | 0000 | 0000 | 0000 | None | |
| NOP | — | No Operation | 1 | 1111 | xxxx | xxxx | xxxx | None | 4 |
| POP | — | Pop top of return stack (TOS) | 1 | 0000 | 0000 | 0000 | 0110 | None | |
| PUSH | — | Push top of return stack (TOS) | 1 | 0000 | 0000 | 0000 | 0101 | None | |
| RCALL | n | Relative Call | 2 | 1101 | 1nnn | nnnn | nnnn | None | |
| RESET | | Software device Reset | 1 | 0000 | 0000 | 1111 | 1111 | All | |
| RETFIE | s | Return from interrupt enable | 2 | 0000 | 0000 | 0001 | 000s | GIE/GIEH, PEIE/GIEL | |
| RETLW | k | Return with literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| RETURN | s | Return from Subroutine | 2 | 0000 | 0000 | 0001 | 001s | None | |
| SLEEP | — | Go into Standby mode | 1 | 0000 | 0000 | 0000 | 0011 | $\overline{TO}$, $\overline{PD}$ | |

**Note 1:** When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

**2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

**3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

**4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

**TABLE 24-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **LITERAL OPERATIONS** | | | | | | | | | |
| ADDLW | k | Add literal and WREG | 1 | 0000 | 1111 | kkkk | kkkk | C, DC, Z, OV, N | |
| ANDLW | k | AND literal with WREG | 1 | 0000 | 1011 | kkkk | kkkk | Z, N | |
| IORLW | k | Inclusive OR literal with WREG | 1 | 0000 | 1001 | kkkk | kkkk | Z, N | |
| LFSR | f, k | Move literal (12-bit) 2nd word | 2 | 1110 | 1110 | 00ff | kkkk | None | |
| | | to FSR(f) 1st word | | 1111 | 0000 | kkkk | kkkk | | |
| MOVLB | k | Move literal to BSR<3:0> | 1 | 0000 | 0001 | 0000 | kkkk | None | |
| MOVLW | k | Move literal to WREG | 1 | 0000 | 1110 | kkkk | kkkk | None | |
| MULLW | k | Multiply literal with WREG | 1 | 0000 | 1101 | kkkk | kkkk | None | |
| RETLW | k | Return with literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| SUBLW | k | Subtract WREG from literal | 1 | 0000 | 1000 | kkkk | kkkk | C, DC, Z, OV, N | |
| XORLW | k | Exclusive OR literal with WREG | 1 | 0000 | 1010 | kkkk | kkkk | Z, N | |
| **DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS** | | | | | | | | | |
| TBLRD* | | Table Read | 2 | 0000 | 0000 | 0000 | 1000 | None | |
| TBLRD*+ | | Table Read with post-increment | | 0000 | 0000 | 0000 | 1001 | None | |
| TBLRD*- | | Table Read with post-decrement | | 0000 | 0000 | 0000 | 1010 | None | |
| TBLRD+* | | Table Read with pre-increment | | 0000 | 0000 | 0000 | 1011 | None | |
| TBLWT* | | Table Write | 2 | 0000 | 0000 | 0000 | 1100 | None | |
| 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.


© 2009-2016 Microchip Technology Inc.

### 24.1.1 STANDARD INSTRUCTION SET

**ADDLW** **ADD literal to W**

Syntax: ADDLW k

Operands: $0 \le k \le 255$

Operation: $(W) + k \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1111 | kkkk | kkkk |

Description: The contents of W are added to the 8-bit literal 'k' and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: ADDLW 15h

Before Instruction
W = 10h
After Instruction
W = 25h

**ADDWF** **ADD W to f**

Syntax: ADDWF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(W) + (f) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 01da | ffff | ffff |

Description: Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: ADDWF REG, 0, 0

Before Instruction
W = 17h
REG = 0C2h
After Instruction
W = 0D9h
REG = 0C2h

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

### ADDWFC — ADD W and CARRY bit to f

Syntax: ADDWFC f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) + (f) + (C) → dest

Status Affected: N,OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 00da | ffff | ffff |

Description: Add W, the CARRY flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: ADDWFC REG, 0, 1

Before Instruction
CARRY bit = 1
REG = 02h
W = 4Dh
After Instruction
CARRY bit = 0
REG = 02h
W = 50h

### ANDLW — AND literal with W

Syntax: ANDLW k

Operands: $0 \le k \le 255$

Operation: (W) .AND. k → W

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1011 | kkkk | kkkk |

Description: The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: ANDLW 05Fh

Before Instruction
W = A3h
After Instruction
W = 03h


© 2009-2016 Microchip Technology Inc.

### ANDWF — AND W with f

Syntax: ANDWF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .AND. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 01da | ffff | ffff |

Description: The contents of W are AND'ed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ANDWF REG, 0, 0`

Before Instruction
W = 17h
REG = C2h
After Instruction
W = 02h
REG = C2h

### BC — Branch if Carry

Syntax: BC n

Operands: $-128 \le n \le 127$

Operation: if CARRY bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0010 | nnnn | nnnn |

Description: If the CARRY bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:
```
HERE    BC   5
```
Before Instruction
PC = address (HERE)
After Instruction
If CARRY = 1;
PC = address (HERE + 12)
If CARRY = 0;
PC = address (HERE + 2)

## BCF — Bit Clear f

Syntax: BCF f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $0 \rightarrow$ f<b>

Status Affected: None

Encoding:

| 1001 | bbba | ffff | ffff |
|---|---|---|---|

Description: Bit 'b' in register 'f' is cleared.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (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 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

## BN — Branch if Negative

Syntax: BN n

Operands: $-128 \le n \le 127$

Operation: if NEGATIVE bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0110 | nnnn | nnnn |
|---|---|---|---|

Description: If the NEGATIVE bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:
```
HERE    BN   Jump
```

Before Instruction
PC = address (HERE)
After Instruction
If NEGATIVE = 1;
PC = address (Jump)
If NEGATIVE = 0;
PC = address (HERE + 2)


© 2009-2016 Microchip Technology Inc.

**BNC** **Branch if Not Carry**

Syntax: BNC n

Operands: -128 ≤ n ≤ 127

Operation: if CARRY bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0011 | nnnn | nnnn |
|---|---|---|---|

Description: If the CARRY bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNC Jump

```
Before Instruction
    PC          =   address (HERE)
After Instruction
    If CARRY    =   0;
        PC      =   address (Jump)
    If CARRY    =   1;
        PC      =   address (HERE + 2)
```

**BNN** **Branch if Not Negative**

Syntax: BNN n

Operands: -128 ≤ n ≤ 127

Operation: if NEGATIVE bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0111 | nnnn | nnnn |
|---|---|---|---|

Description: If the NEGATIVE bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNN Jump

```
Before Instruction
    PC          =   address (HERE)
After Instruction
    If NEGATIVE =   0;
        PC      =   address (Jump)
    If NEGATIVE =   1;
        PC      =   address (HERE + 2)
```

**BNOV** **Branch if Not Overflow**

Syntax: BNOV n

Operands: -128 ≤ n ≤ 127

Operation: if OVERFLOW bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0101 | nnnn | nnnn |
|---|---|---|---|

Description: If the OVERFLOW bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNOV Jump

Before Instruction
PC = address (HERE)
After Instruction
If OVERFLOW = 0;
PC = address (Jump)
If OVERFLOW = 1;
PC = address (HERE + 2)

**BNZ** **Branch if Not Zero**

Syntax: BNZ n

Operands: -128 ≤ n ≤ 127

Operation: if ZERO bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0001 | nnnn | nnnn |
|---|---|---|---|

Description: If the ZERO bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNZ Jump

Before Instruction
PC = address (HERE)
After Instruction
If ZERO = 0;
PC = address (Jump)
If ZERO = 1;
PC = address (HERE + 2)


© 2009-2016 Microchip Technology Inc.

## 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 2-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BRA   Jump
```
Before Instruction
PC = address (HERE)
After Instruction
PC = address (Jump)

## BSF Bit Set f

Syntax: BSF f, b {,a}

Operands: 0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: 1 → f<b>

Status Affected: None

Encoding:

| 1000 | bbba | ffff | ffff |
|---|---|---|---|

Description: Bit 'b' in register 'f' is set.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (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 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example:
```
BSF     FLAG_REG, 7, 1
```
Before Instruction
FLAG_REG = 0Ah
After Instruction
FLAG_REG = 8Ah

**BTFSC** **Bit Test File, Skip if Clear**

Syntax: BTFSC f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

| 1011 | bbba | ffff | ffff |
|---|---|---|---|

Description: If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-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 $f \le 95$ (5Fh).
See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BTFSC   FLAG, 1, 0
FALSE   :
TRUE    :
```
Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

**BTFSS** **Bit Test File, Skip if Set**

Syntax: BTFSS f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

| 1010 | bbba | ffff | ffff |
|---|---|---|---|

Description: If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 2-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 $f \le 95$ (5Fh).
See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    BTFSS   FLAG, 1, 0
FALSE   :
TRUE    :
```
Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)


© 2009-2016 Microchip Technology Inc.

## BTG Bit Toggle f

Syntax: BTG f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: $(\overline{f<b>}) \rightarrow f<b>$

Status Affected: None

Encoding:

| 0111 | bbba | ffff | ffff |
|---|---|---|---|

Description: Bit 'b' in data memory location 'f' is inverted.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `BTG PORTC, 4, 0`

Before Instruction:
PORTC = 0111 0101 [75h]
After Instruction:
PORTC = 0110 0101 [65h]

## BOV Branch if Overflow

Syntax: BOV n

Operands: $-128 \le n \le 127$

Operation: if OVERFLOW bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| 1110 | 0100 | nnnn | nnnn |
|---|---|---|---|

Description: If the OVERFLOW bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:
```
HERE    BOV   Jump
```
Before Instruction
PC = address (HERE)
After Instruction
If OVERFLOW = 1;
PC = address (Jump)
If OVERFLOW = 0;
PC = address (HERE + 2)

**BZ** **Branch if Zero**

Syntax: BZ n

Operands: -128 ≤ n ≤ 127

Operation: if ZERO bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0000 | nnnn | nnnn |
|---|---|---|---|

Description: If the ZERO bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:
```
HERE    BZ   Jump
Before Instruction
    PC          =   address (HERE)
After Instruction
    If ZERO     =   1;
        PC      =   address (Jump)
    If ZERO     =   0;
        PC      =   address (HERE + 2)
```

**CALL** **Subroutine Call**

Syntax: CALL k {,s}

Operands: 0 ≤ k ≤ 1048575
s ∈ [0,1]

Operation: (PC) + 4 → TOS,
k → PC<20:1>,
if s = 1
(W) → WS,
(Status) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (k<7:0>) | 1110 | 110s | $k_7$kkk | kkkk$_0$ |
| 2nd word(k<19:8>) | 1111 | $k_{19}$kkk | kkkk | kkkk$_8$ |

Description: Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a 2-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0>, | PUSH PC to stack | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    CALL   THERE, 1
Before Instruction
    PC      =   address (HERE)
After Instruction
    PC      =   address (THERE)
    TOS     =   address (HERE + 4)
    WS      =   W
    BSRS    =   BSR
    STATUSS =   Status
```


© 2009-2016 Microchip Technology Inc.

**CLRF** **Clear f**

Syntax: CLRF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: 000h → f
1 → Z

Status Affected: Z

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 101a | ffff | ffff |

Description: Clears the contents of the specified register.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: CLRF FLAG_REG, 1

Before Instruction
FLAG_REG = 5Ah
After Instruction
FLAG_REG = 00h

**CLRWDT** **Clear Watchdog Timer**

Syntax: CLRWDT

Operands: None

Operation: 000h → WDT,
000h → WDT postscaler,
1 → $\overline{TO}$,
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0100 |

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the post-scaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | No operation |

Example: CLRWDT

Before Instruction
WDT Counter = ?
After Instruction
WDT Counter = 00h
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

## COMF — Complement f

Syntax: COMF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(\bar{f}) \rightarrow$ dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 11da | ffff | ffff |

Description: The contents of register 'f' are complemented. If 'd' is '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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `COMF REG, 0, 0`

Before Instruction
REG = 13h
After Instruction
REG = 13h
W = ECh

## CPFSEQ — Compare f with W, skip if f = W

Syntax: CPFSEQ f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 001a | ffff | ffff |

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)


© 2009-2016 Microchip Technology Inc.

**CPFSGT** **Compare f with W, skip if f > W**

Syntax: CPFSGT f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) > (W)
(unsigned comparison)

Status Affected: None

Encoding:

| 0110 | 010a | ffff | ffff |
|---|---|---|---|

Description: Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction.
If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE      CPFSGT REG, 0
NGREATER  :
GREATER   :
```
Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG > W;
PC = Address (GREATER)
If REG ≤ W;
PC = Address (NGREATER)

**CPFSLT** **Compare f with W, skip if f < W**

Syntax: CPFSLT f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) < (W)
(unsigned comparison)

Status Affected: None

Encoding:

| 0110 | 000a | ffff | ffff |
|---|---|---|---|

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-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

Cycles: 1(2)
**Note:** Three cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```
Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG < W;
PC = Address (LESS)
If REG ≥ W;
PC = Address (NLESS)

## DAW Decimal Adjust W Register

Syntax: DAW

Operands: None

Operation:
If [W<3:0> > 9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;

If [W<7:4> + DC > 9] or [C = 1] then
(W<7:4>) + 6 + DC → W<7:4>;
else
(W<7:4>) + DC → W<7:4>

Status Affected: C

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0111 |

Description: DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register W | Process Data | Write W |

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 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) – 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 01da | ffff | ffff |

Description: Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: DECF CNT, 1, 0

Before Instruction
CNT = 01h
Z = 0

After Instruction
CNT = 00h
Z = 1


© 2009-2016 Microchip Technology Inc.

**DECFSZ** **Decrement f, skip if 0**

Syntax: DECFSZ f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

| 0010 | 11da | ffff | ffff |
|---|---|---|---|

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    DECFSZ  CNT, 1, 1
        GOTO    LOOP
CONTINUE
```
Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT - 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE + 2)

**DCFSNZ** **Decrement f, skip if not 0**

Syntax: DCFSNZ f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

| 0100 | 11da | ffff | ffff |
|---|---|---|---|

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (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 2-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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    DCFSNZ  TEMP, 1, 0
ZERO    :
NZERO   :
```
Before Instruction
TEMP = ?
After Instruction
TEMP = TEMP – 1,
If TEMP = 0;
PC = Address (ZERO)
If TEMP ≠ 0;
PC = Address (NZERO)

## GOTO — Unconditional Branch

Syntax: GOTO k

Operands: $0 \le k \le 1048575$

Operation: $k \rightarrow PC<20:1>$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (k<7:0>) | 1110 | $k_7kkk$ | $kkkk_0$ |
| 2nd word(k<19:8>) | 1111 | $k_{19}kkk$ | $kkkk_8$ |

Description: GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a 2-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0>, | No operation | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation | No operation | No operation |

Example: GOTO THERE

After Instruction
PC = Address (THERE)

## INCF — Increment f

Syntax: INCF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) + 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 10da | ffff | ffff |

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: INCF CNT, 1, 0

Before Instruction
CNT = FFh
Z = 0
C = ?
DC = ?
After Instruction
CNT = 00h
Z = 1
C = 1
DC = 1


© 2009-2016 Microchip Technology Inc.

## INCFSZ — Increment f, skip if 0

Syntax: INCFSZ f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) + 1 → dest,
skip if result = 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 11da | ffff | ffff |

Description: 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 2-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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

## INFSNZ — Increment f, skip if not 0

Syntax: INFSNZ f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) + 1 → dest,
skip if result ≠ 0

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 10da | ffff | ffff |

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    INFSNZ  REG, 1, 0
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

### IORLW Inclusive OR literal with W

Syntax: IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → W

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1001 | kkkk | kkkk |

Description: The contents of W are ORed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: IORLW 35h

Before Instruction
W = 9Ah
After Instruction
W = BFh

### IORWF Inclusive OR W with f

Syntax: IORWF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .OR. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 00da | ffff | ffff |

Description: Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: IORWF RESULT, 0, 1

Before Instruction
RESULT = 13h
W = 91h
After Instruction
RESULT = 13h
W = 93h


© 2009-2016 Microchip Technology Inc.

## LFSR — Load FSR

Syntax: LFSR f, k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 1111 | 1110 0000 | 00ff $k_7$kkk | $k_{11}$kkk kkkk |

Description: The 12-bit literal 'k' is loaded into the File Select Register pointed to by 'f'.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' MSB | Process Data | Write literal 'k' MSB to FSRfH |
| Decode | Read literal 'k' LSB | Process Data | Write literal 'k' to FSRfL |

Example: `LFSR 2, 3ABh`

After Instruction
FSR2H = 03h
FSR2L = ABh

## MOVF — Move f

Syntax: MOVF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $f \rightarrow dest$

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 00da | ffff | ffff |

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write W |

Example: `MOVF REG, 0, 0`

Before Instruction
REG = 22h
W = FFh
After Instruction
REG = 22h
W = 22h

**MOVFF** **Move f to f**

Syntax: MOVFF $f_s$,$f_d$

Operands: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (source) | 1100 | ffff | ffff$_s$ |
| 2nd word (destin.) | 1111 | ffff | ffff$_d$ |

Description: The contents of source register '$f_s$' are moved to destination register '$f_d$'. Location of source '$f_s$' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination '$f_d$' can also be anywhere from 000h to FFFh.
Either source or destination can be W (a useful special situation).
MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).
The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' (src) | Process Data | No operation |
| Decode | No operation<br>No dummy read | No operation | Write register 'f' (dest) |

Example: MOVFF REG1, REG2

Before Instruction
REG1 = 33h
REG2 = 11h
After Instruction
REG1 = 33h
REG2 = 33h

**MOVLB** **Move literal to low nibble in BSR**

Syntax: MOVLB k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow$ BSR

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0001 | 0000 | kkkk |

Description: The 8-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '0', regardless of the value of $k_7$:$k_4$.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write literal 'k' to BSR |

Example: MOVLB 5

Before Instruction
BSR Register = 02h
After Instruction
BSR Register = 05h


© 2009-2016 Microchip Technology Inc.

**MOVLW** **Move literal to W**

Syntax: MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

Status Affected: None

Encoding:

| 0000 | 1110 | kkkk | kkkk |
|---|---|---|---|

Description: The 8-bit literal ‘k’ is loaded into W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘k’ | Process Data | Write to W |

Example: MOVLW 5Ah

After Instruction

W = 5Ah

**MOVWF** **Move W to f**

Syntax: MOVWF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(W) \rightarrow f$

Status Affected: None

Encoding:

| 0110 | 111a | ffff | ffff |
|---|---|---|---|

Description: Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank.
If ‘a’ is ‘0’, the Access Bank is selected.
If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default).
If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write register ‘f’ |

Example: MOVWF REG, 0

Before Instruction

W = 4Fh
REG = FFh

After Instruction

W = 4Fh
REG = 4Fh

**MULLW** **Multiply literal with W**

Syntax: MULLW k

Operands: $0 \le k \le 255$

Operation: (W) x k → PRODH:PRODL

Status Affected: None

Encoding:

| 0000 | 1101 | kkkk | kkkk |
|---|---|---|---|

Description: An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.
None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write registers PRODH: PRODL |

Example: `MULLW 0C4h`

Before Instruction
W = E2h
PRODH = ?
PRODL = ?
After Instruction
W = E2h
PRODH = ADh
PRODL = 08h

**MULWF** **Multiply W with f**

Syntax: MULWF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (W) x (f) → PRODH:PRODL

Status Affected: None

Encoding:

| 0000 | 001a | ffff | ffff |
|---|---|---|---|

Description: An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged.
None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write registers PRODH: PRODL |

Example: `MULWF REG, 1`

Before Instruction
W = C4h
REG = B5h
PRODH = ?
PRODL = ?
After Instruction
W = C4h
REG = B5h
PRODH = 8Ah
PRODL = 94h


© 2009-2016 Microchip Technology Inc.

**NEGF** **Negate f**

Syntax: NEGF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(\overline{f}) + 1 \rightarrow f$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 110a | ffff | ffff |

Description: Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (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 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: NEGF REG, 1

Before Instruction
REG = 0011 1010 [3Ah]
After Instruction
REG = 1100 0110 [C6h]

**NOP** **No Operation**

Syntax: NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 1111 | 0000 xxxx | 0000 xxxx | 0000 xxxx |

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |

Example:

None.

**POP** **Pop Top of Return Stack**

Syntax: POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0110 |

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | POP TOS value | No operation |

Example:

```
POP
GOTO    NEW
```

Before Instruction
TOS = 0031A2h
Stack (1 level down) = 014332h
After Instruction
TOS = 014332h
PC = NEW

**PUSH** **Push Top of Return Stack**

Syntax: PUSH

Operands: None

Operation: (PC + 2) → TOS

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0101 |

Description: The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | PUSH PC + 2 onto return stack | No operation | No operation |

Example:

```
PUSH
```

Before Instruction
TOS = 345Ah
PC = 0124h
After Instruction
PC = 0126h
TOS = 0126h
Stack (1 level down) = 345Ah


© 2009-2016 Microchip Technology Inc.

**RCALL** **Relative Call**

Syntax: RCALL n

Operands: $-1024 \leq n \leq 1023$

Operation: (PC) + 2 → TOS,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1101 | 1nnn | nnnn | nnnn |

Description: Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a 2-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n'<br>PUSH PC to stack | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: HERE RCALL Jump

Before Instruction
PC = Address (HERE)
After Instruction
PC = Address (Jump)
TOS = Address (HERE + 2)

**RESET** **Reset**

Syntax: RESET

Operands: None

Operation: Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.

Status Affected: All

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 1111 | 1111 |

Description: This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset by software.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Start Reset | No operation | No operation |

Example: RESET

After Instruction
Registers = Reset Value
Flags* = Reset Value

**RETFIE** **Return from Interrupt**

Syntax: RETFIE {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 000s |

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | POP PC from stack<br>Set GIEH or GIEL |
| No operation | No operation | No operation | No operation |

Example:

```
RETFIE  1
```

After Interrupt
PC = TOS
W = WS
BSR = BSRS
Status = STATUSS
GIE/GIEH, PEIE/GIEL = 1

**RETLW** **Return literal to W**

Syntax: RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W,
(TOS) → PC,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1100 | kkkk | kkkk |

Description: W is loaded with the 8-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | POP PC from stack, Write to W |
| No operation | No operation | No operation | No operation |

Example:

```
    CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
    :
TABLE
    ADDWF PCL  ; W = offset
    RETLW k0   ; Begin table
    RETLW k1   ;
    :
    :
    RETLW kn   ; End of table
```

Before Instruction
W = 07h
After Instruction
W = value of kn


© 2009-2016 Microchip Technology Inc.

## RETURN — Return from Subroutine

Syntax: RETURN {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
if s = 1
(WS) → W,
(STATUSS) → Status,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 001s |

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's'= 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | POP PC from stack |
| No operation | No operation | No operation | No operation |

Example: RETURN

After Instruction:
PC = TOS

## RLCF — Rotate Left f through Carry

Syntax: RLCF f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

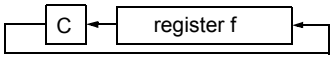
Operation: (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C, N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 01da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the left through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example:
```
RLCF    REG, 0, 0
```
Before Instruction
```
REG  =  1110 0110
C    =  0
```
After Instruction
```
REG  =  1110 0110
W    =  1100 1100
C    =  1
```

**RLNCF** **Rotate Left f (No Carry)**

Syntax: RLNCF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

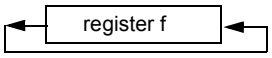
Operation: (f<n>) → dest<n + 1>,
(f<7>) → dest<0>

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 01da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RLNCF REG, 1, 0

Before Instruction
REG = 1010 1011
After Instruction
REG = 0101 0111

**RRCF** **Rotate Right f through Carry**

Syntax: RRCF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

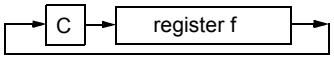
Operation: (f<n>) → dest<n – 1>,
(f<0>) → C,
(C) → dest<7>

Status Affected: C, N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 00da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the right through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RRCF REG, 0, 0

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 0111 0011
C = 0


© 2009-2016 Microchip Technology Inc.

**RRNCF** **Rotate Right f (No Carry)**

Syntax: RRNCF f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

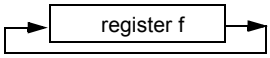
Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 00da | ffff | ffff |

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.




Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: RRNCF REG, 1, 0

Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

**SETF** **Set f**

Syntax: SETF f {,a}

Operands: 0 ≤ f ≤ 255
a ∈ [0,1]

Operation: FFh → f

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 100a | ffff | ffff |

Description: The contents of the specified register are set to FFh.
If 'a' is '0', the Access Bank 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 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: SETF REG, 1

Before Instruction
REG = 5Ah
After Instruction
REG = FFh

## SLEEP — Enter Sleep mode

Syntax: SLEEP

Operands: None

Operation: 00h → WDT,
0 → $\overline{\text{WDT}}$ postscaler,
1 → $\overline{\text{TO}}$,
0 → $\overline{\text{PD}}$

Status Affected: $\overline{\text{TO}}$, $\overline{\text{PD}}$

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0011 |

Description: The Power-down Status bit ($\overline{\text{PD}}$) is cleared. The Time-out Status bit ($\overline{\text{TO}}$) is set. Watchdog Timer and its postscaler are cleared.
The processor is put into Sleep mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | Go to Sleep |

Example: SLEEP

Before Instruction
$\overline{\text{TO}}$ = ?
$\overline{\text{PD}}$ = ?
After Instruction
$\overline{\text{TO}}$ = 1 †
$\overline{\text{PD}}$ = 0

† If WDT causes wake-up, this bit is cleared.

## SUBFWB — Subtract f from W with borrow

Syntax: SUBFWB f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (W) – (f) – ($\overline{\text{C}}$) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 01da | ffff | ffff |

Description: Subtract register 'f' and CARRY flag (borrow) from W (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored in register 'f' (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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1:
```
SUBFWB   REG, 1, 0
```
Before Instruction
REG = 3
W = 2
C = 1
After Instruction
REG = FF
W = 2
C = 0
Z = 0
N = 1 ; result is negative

Example 2:
```
SUBFWB   REG, 0, 0
```
Before Instruction
REG = 2
W = 5
C = 1
After Instruction
REG = 2
W = 3
C = 1
Z = 0
N = 0 ; result is positive

Example 3:
```
SUBFWB   REG, 1, 0
```
Before Instruction
REG = 1
W = 2
C = 0
After Instruction
REG = 0
W = 2
C = 1
Z = 1 ; result is zero
N = 0


© 2009-2016 Microchip Technology Inc.

**SUBLW** **Subtract W from literal**

Syntax: SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

| 0000 | 1000 | kkkk | kkkk |
|---|---|---|---|

Description: W is subtracted from the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example 1: SUBLW 02h

Before Instruction
W = 01h
C = ?
After Instruction
W = 01h
C = 1 ; result is positive
Z = 0
N = 0

Example 2: SUBLW 02h

Before Instruction
W = 02h
C = ?
After Instruction
W = 00h
C = 1 ; result is zero
Z = 1
N = 0

Example 3: SUBLW 02h

Before Instruction
W = 03h
C = ?
After Instruction
W = FFh ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

**SUBWF** **Subtract W from f**

Syntax: SUBWF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(f) - (W) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

| 0101 | 11da | ffff | ffff |
|---|---|---|---|

Description: Subtract W from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: SUBWF REG, 1, 0

Before Instruction
REG = 3
W = 2
C = ?
After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2: SUBWF REG, 0, 0

Before Instruction
REG = 2
W = 2
C = ?
After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: SUBWF REG, 1, 0

Before Instruction
REG = 1
W = 2
C = ?
After Instruction
REG = FFh ;(2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

© 2009-2016 Microchip Technology Inc.

**SUBWFB** **Subtract W from f with Borrow**

Syntax: SUBWFB f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – (W) – $(\overline{C})$ → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| 0101 | 10da | ffff | ffff |
|---|---|---|---|

Description: Subtract W and the CARRY flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1:
```
SUBWFB  REG, 1, 0
Before Instruction
    REG  =  19h   (0001 1001)
    W    =  0Dh   (0000 1101)
    C    =  1
After Instruction
    REG  =  0Ch   (0000 1011)
    W    =  0Dh   (0000 1101)
    C    =  1
    Z    =  0
    N    =  0     ; result is positive
```

Example 2:
```
SUBWFB  REG, 0, 0
Before Instruction
    REG  =  1Bh   (0001 1011)
    W    =  1Ah   (0001 1010)
    C    =  0
After Instruction
    REG  =  1Bh   (0001 1011)
    W    =  00h
    C    =  1
    Z    =  1     ; result is zero
    N    =  0
```

Example 3:
```
SUBWFB  REG, 1, 0
Before Instruction
    REG  =  03h   (0000 0011)
    W    =  0Eh   (0000 1101)
    C    =  1
After Instruction
    REG  =  F5h   (1111 0100)
                  ; [2's comp]
    W    =  0Eh   (0000 1101)
    C    =  0
    Z    =  0
    N    =  1     ; result is negative
```

**SWAPF** **Swap f**

Syntax: SWAPF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<3:0>) → dest<7:4>,
(f<7:4>) → dest<3:0>

Status Affected: None

Encoding:

| 0011 | 10da | ffff | ffff |
|---|---|---|---|

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (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 **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example:
```
SWAPF   REG, 1, 0
Before Instruction
    REG  =  53h
After Instruction
    REG  =  35h
```


© 2009-2016 Microchip Technology Inc.

**TBLRD Table Read**

Syntax: TBLRD ( *; *+; *-; +*)

Operands: None

Operation: if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT;
TBLPTR – No Change;
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) + 1 → TBLPTR;
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) – 1 → TBLPTR;
if TBLRD +*,
(TBLPTR) + 1 → TBLPTR;
(Prog Mem (TBLPTR)) → TABLAT;

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 10nn<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word
TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation (Read Program Memory) | No operation | No operation (Write TABLAT) |

**TBLRD Table Read (Continued)**

Example1:
```
TBLRD   *+ ;
```
Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
MEMORY (00A356h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 00A357h

Example2:
```
TBLRD   +* ;
```
Before Instruction
TABLAT = AAh
TBLPTR = 01A357h
MEMORY (01A357h) = 12h
MEMORY (01A358h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 01A358h

**TBLWT** **Table Write**

Syntax: TBLWT ( *; *+; *-; +*)

Operands: None

Operation:
if TBLWT*,
(TABLAT) → Holding Register;
TBLPTR – No Change;
if TBLWT*+,
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR;
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register;

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 11nn<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |

Description: This instruction uses the 3 LSBs of TBLPTR to determine which of the eight holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 4.0 "Flash Program Memory"** for additional details on programming Flash memory.)
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-MByte address range.
The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word
TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation (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


© 2009-2016 Microchip Technology Inc.

**TSTFSZ** **Test f, skip if 0**

Syntax: TSTFSZ f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: skip if f = 0

Status Affected: None

Encoding:

| 0110 | 011a | ffff | ffff |
|---|---|---|---|

Description: If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a NOP is executed, making this a 2-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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    TSTFSZ  CNT, 1
NZERO   :
ZERO    :
```
Before Instruction
PC = Address (HERE)
After Instruction
If CNT = 00h,
PC = Address (ZERO)
If CNT ≠ 00h,
PC = Address (NZERO)

**XORLW** **Exclusive OR literal with W**

Syntax: XORLW k

Operands: $0 \le k \le 255$

Operation: (W) .XOR. k → W

Status Affected: N, Z

Encoding:

| 0000 | 1010 | kkkk | kkkk |
|---|---|---|---|

Description: The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example:
```
XORLW   0AFh
```
Before Instruction
W = B5h
After Instruction
W = 1Ah

**XORWF** **Exclusive OR W with f**

Syntax: XORWF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) .XOR. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 10da | ffff | ffff |

Description: Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (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 $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `XORWF REG, 1, 0`

Before Instruction
REG = AFh
W = B5h
After Instruction
REG = 1Ah
W = B5h


© 2009-2016 Microchip Technology Inc.

## 24.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F1XK22 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 24-3. Detailed descriptions are provided in **Section 24.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 24-1 (page 266) apply to both the standard and extended PIC18 instruction sets.

| | |
|---|---|
| **Note:** | The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler. |

### 24.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets (" [ ]"). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 24.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**.

| | |
|---|---|
| **Note:** | In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{ }"). |

**TABLE 24-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected |
|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | |
| ADDFSR | f, k | Add literal to FSR | 1 | 1110 | 1000 | ffkk | kkkk | None |
| ADDULNK | k | Add literal to FSR2 and return | 2 | 1110 | 1000 | 11kk | kkkk | None |
| CALLW | | Call subroutine using WREG | 2 | 0000 | 0000 | 0001 | 0100 | None |
| MOVSF | $z_s$, $f_d$ | Move $z_s$ (source) to 1st word | 2 | 1110 | 1011 | 0zzz | zzzz | None |
| | | $f_d$ (destination) 2nd word | | 1111 | ffff | ffff | ffff | |
| MOVSS | $z_s$, $z_d$ | Move $z_s$ (source) to 1st word | 2 | 1110 | 1011 | 1zzz | zzzz | None |
| | | $z_d$ (destination) 2nd word | | 1111 | xxxx | xzzz | zzzz | |
| PUSHL | k | Store literal at FSR2, decrement FSR2 | 1 | 1110 | 1010 | kkkk | kkkk | None |
| SUBFSR | f, k | Subtract literal from FSR | 1 | 1110 | 1001 | ffkk | kkkk | None |
| SUBULNK | k | Subtract literal from FSR2 and return | 2 | 1110 | 1001 | 11kk | kkkk | None |

### 24.2.2 EXTENDED INSTRUCTION SET

**ADDFSR** **Add Literal to FSR**

Syntax: ADDFSR f, k

Operands: $0 \le k \le 63$
$f \in [0, 1, 2]$

Operation: FSR(f) + k → FSR(f)

Status Affected: None

Encoding:

| 1110 | 1000 | ffkk | kkkk |
|---|---|---|---|

Description: The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to FSR |

Example: `ADDFSR 2, 23h`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

**ADDULNK** **Add Literal to FSR2 and Return**

Syntax: ADDULNK k

Operands: $0 \le k \le 63$

Operation: FSR2 + k → FSR2,
(TOS) → PC

Status Affected: None

Encoding:

| 1110 | 1000 | 11kk | kkkk |
|---|---|---|---|

Description: The 6-bit literal 'k' is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS.
The instruction takes two cycles to execute; a NOP is performed during the second cycle.
This may be thought of as a special case of the ADDFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to FSR |
| No Operation | No Operation | No Operation | No Operation |

Example: `ADDULNK 23h`

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 0422h
PC = (TOS)

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).


© 2009-2016 Microchip Technology Inc.

**CALLW** **Subroutine Call Using WREG**

Syntax: CALLW

Operands: None

Operation: (PC + 2) → TOS,
(W) → PCL,
(PCLATH) → PCH,
(PCLATU) → PCU

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 0100 |

Description: First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched into PCH and PCU, respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched. Unlike CALL, there is no option to update W, Status or BSR.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read WREG | PUSH PC to stack | No operation |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    CALLW
```
Before Instruction
```
PC      =   address (HERE)
PCLATH  =   10h
PCLATU  =   00h
W       =   06h
```
After Instruction
```
PC      =   001006h
TOS     =   address (HERE + 2)
PCLATH  =   10h
PCLATU  =   00h
W       =   06h
```

**MOVSF** **Move Indexed to f**

Syntax: MOVSF [$z_s$], $f_d$

Operands: $0 \le z_s \le 127$
$0 \le f_d \le 4095$

Operation: ((FSR2) + $z_s$) → $f_d$

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (source) | 1110 | 1011 | 0zzz | $zzzz_s$ |
| 2nd word (destin.) | 1111 | ffff | ffff | $ffff_d$ |

Description: The contents of the source register are moved to destination register '$f_d$'. The actual address of the source register is determined by adding the 7-bit literal offset '$z_s$' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal '$f_d$' in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh).
The MOVSF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
If the resultant source address points to an indirect addressing register, the value returned will be 00h.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | No operation<br>No dummy read | No operation | Write register 'f' (dest) |

Example:
```
MOVSF   [05h], REG2
```
Before Instruction
```
FSR2            =   80h
Contents
of 85h          =   33h
REG2            =   11h
```
After Instruction
```
FSR2            =   80h
Contents
of 85h          =   33h
REG2            =   33h
```

**MOVSS** **Move Indexed to Indexed**

Syntax: MOVSS [$z_s$], [$z_d$]

Operands: $0 \le z_s \le 127$
$0 \le z_d \le 127$

Operation: ((FSR2) + $z_s$) → ((FSR2) + $z_d$)

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (source) | 1110 | 1011 | 1zzz | $zzzz_s$ |
| 2nd word (dest.) | 1111 | xxxx | xzzz | $zzzz_d$ |

Description: The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets '$z_s$' or '$z_d$', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).
The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | Determine dest addr | Determine dest addr | Write to dest reg |

Example: `MOVSS [05h], [06h]`

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 11h
After Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 33h

**PUSHL** **Store Literal at FSR2, Decrement FSR2**

Syntax: PUSHL k

Operands: $0 \le k \le 255$

Operation: k → (FSR2),
FSR2 – 1 → FSR2

Status Affected: None

Encoding:

| 1110 | 1010 | kkkk | kkkk |
|---|---|---|---|

Description: The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation. This instruction allows users to push values onto a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process data | Write to destination |

Example: `PUSHL 08h`

Before Instruction
FSR2H:FSR2L = 01ECh
Memory (01ECh) = 00h

After Instruction
FSR2H:FSR2L = 01EBh
Memory (01ECh) = 08h


© 2009-2016 Microchip Technology Inc.

**SUBFSR** **Subtract Literal from FSR**

Syntax: SUBFSR f, k

Operands: $0 \le k \le 63$

$f \in [0, 1, 2]$

Operation: FSR(f) – k → FSRf

Status Affected: None

Encoding:

| 1110 | 1001 | ffkk | kkkk |
|---|---|---|---|

Description: The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: SUBFSR 2, 23h

Before Instruction
FSR2 = 03FFh

After Instruction
FSR2 = 03DCh

**SUBULNK** **Subtract Literal from FSR2 and Return**

Syntax: SUBULNK k

Operands: $0 \le k \le 63$

Operation: FSR2 – k → FSR2
(TOS) → PC

Status Affected: None

Encoding:

| 1110 | 1001 | 11kk | kkkk |
|---|---|---|---|

Description: The 6-bit literal 'k' is subtracted from the contents of the FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a NOP is performed during the second cycle.
This may be thought of as a special case of the SUBFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |
| No Operation | No Operation | No Operation | No Operation |

Example: SUBULNK 23h

Before Instruction
FSR2 = 03FFh
PC = 0100h

After Instruction
FSR2 = 03DCh
PC = (TOS)

### 24.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

| **Note:** | Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely. |
|---|---|

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing mode (**Section 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 24.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**).

Although the Indexed Literal Offset Addressing mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset Addressing mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

#### 24.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument, 'f', in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value, 'k'. As already noted, this occurs only when 'f' is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets ("[ ]"). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within brackets, will generate an error in the MPASM™ assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be '0'. This is in contrast to standard operation (extended instruction set disabled) when 'a' is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM assembler.

The destination argument, 'd', functions as before.

In the latest versions of the MPASM assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, /y, or the PE directive in the source listing.

### 24.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18(L)F1XK22, it is very important to consider the type of code. A large, re-entrant application that is written in 'C' and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.


© 2009-2016 Microchip Technology Inc.

**ADDWF** **ADD W to Indexed (Indexed Literal Offset mode)**

Syntax: ADDWF [k] {,d}

Operands: $0 \le k \le 95$
$d \in [0,1]$

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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process Data | Write to destination |

Example: ADDWF [OFST], 0

Before Instruction
W = 17h
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 20h
After Instruction
W = 37h
Contents of 0A2Ch = 20h

**BSF** **Bit Set Indexed (Indexed Literal Offset mode)**

Syntax: BSF [k], b

Operands: $0 \le f \le 95$
$0 \le b \le 7$

Operation: 1 → ((FSR2) + k)<b>

Status Affected: None

Encoding:

| 1000 | bbb0 | kkkk | kkkk |
|---|---|---|---|

Description: Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: BSF [FLAG_OFST], 7

Before Instruction
FLAG_OFST = 0Ah
FSR2 = 0A00h
Contents of 0A0Ah = 55h
After Instruction
Contents of 0A0Ah = D5h

**SETF** **Set Indexed (Indexed Literal Offset mode)**

Syntax: SETF [k]

Operands: $0 \le k \le 95$

Operation: FFh → ((FSR2) + k)

Status Affected: None

Encoding:

| 0110 | 1000 | kkkk | kkkk |
|---|---|---|---|

Description: The contents of the register indicated by FSR2, offset by 'k', are set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process Data | Write register |

Example: SETF [OFST]

Before Instruction
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 00h
After Instruction
Contents of 0A2Ch = FFh

### 24.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18(L)F1XK22 family of devices. This includes the MPLAB® C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.


© 2009-2016 Microchip Technology Inc.

## 25.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICkit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits and Starter Kits
- Third-party development tools

### 25.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

## 25.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. 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. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

## 25.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

## 25.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

## 25.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility


© 2009-2016 Microchip Technology Inc.

## 25.6 MPLAB X SIM Software Simulator

The MPLAB X 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 X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 25.7 MPLAB REAL ICE In-Circuit Emulator System

The 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 all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 25.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 25.9 PICkit 3 In-Circuit Debugger/ Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

## 25.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at $V_{DDMIN}$ and $V_{DDMAX}$ for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.

## 25.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

## 25.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®


© 2009-2016 Microchip Technology Inc.

# 26.0 ELECTRICAL SPECIFICATIONS

## 26.1 Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on pins with respect to VSS
- on VDD pin
  - PIC18F1XK22 ..... -0.3V to +6.5V
  - PIC18LF1XK22 ..... -0.3V to +4.0V
- on $\overline{\text{MCLR}}$ pin ..... -0.3V to +9.0V
- on all other pins ..... -0.3V to (VDD + 0.3V)

Maximum current(1)
- on VSS pin
  - -40°C ≤ TA ≤ +85°C, Industrial ..... 250 mA
  - -40°C ≤ TA ≤ +125°C, Extended ..... 85 mA
- on VDD pin
  - -40°C ≤ TA ≤ +85°C,Industrial ..... 250 mA
  - -40°C ≤ TA ≤ +125°C, Extended ..... 85 mA
- sunk by all ports ..... 250 mA
- sourced by all ports ..... 250 mA

Maximum output current
- sunk by any I/O pin ..... ±50 mA
- sourced by any I/O pin ..... ±50 mA

Clamp current, IK (VPIN < 0 or VPIN > VDD) ..... ±20 mA
Total power dissipation(2) ..... 800 mW

**Note 1:** Maximum current rating requires even load distribution across I/O pins. Maximum current rating may be limited by the device package power dissipation characterizations, see Table 26-8 to calculate device specifications.

**2:** Power dissipation is calculated as follows:
$P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma\{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma(V_{OL} \times I_{OL})$.

† 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.

## 26.2 Standard Operating Conditions

The standard operating conditions for any device are defined as:

Operating Voltage: $V_{DDMIN} \leq V_{DD} \leq V_{DDMAX}$
Operating Temperature: $T_{A_MIN} \leq T_A \leq T_{A_MAX}$

**$V_{DD}$ — Operating Supply Voltage(1)**

PIC18LF1XK22

- $V_{DDMIN}$ (Fosc ≤ 16 MHz) ..... +1.8V
- $V_{DDMIN}$ (Fosc ≤ 20 MHz) ..... +2.0V
- $V_{DDMIN}$ (Fosc ≤ 64 MHz) ..... +3.0V
- $V_{DDMAX}$ ..... +3.6V

PIC18F1XK22

- $V_{DDMIN}$ (Fosc ≤ 20 MHz) ..... +2.3V
- $V_{DDMIN}$ (Fosc ≤ 64 MHz) ..... +3.0V
- $V_{DDMAX}$ ..... +5.5V

**$T_A$ — Operating Ambient Temperature Range**

Industrial Temperature

- $T_{A_MIN}$ ..... -40°C
- $T_{A_MAX}$ ..... +85°C

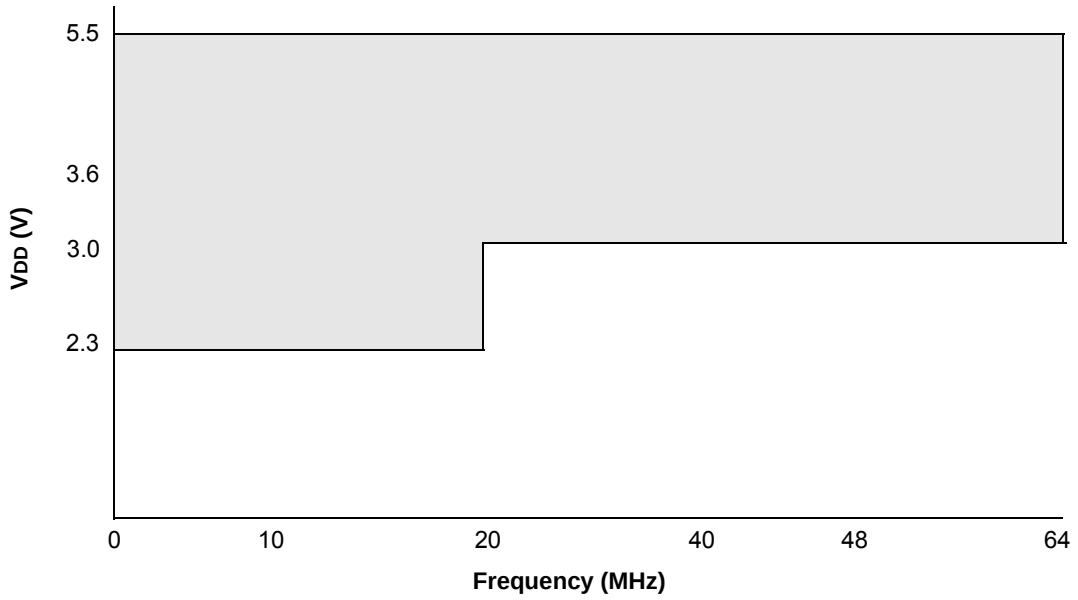
Extended Temperature

- $T_{A_MIN}$ ..... -40°C
- $T_{A_MAX}$ ..... +125°C

**Note 1:** See Parameter D001, DC Characteristics: Supply Voltage.

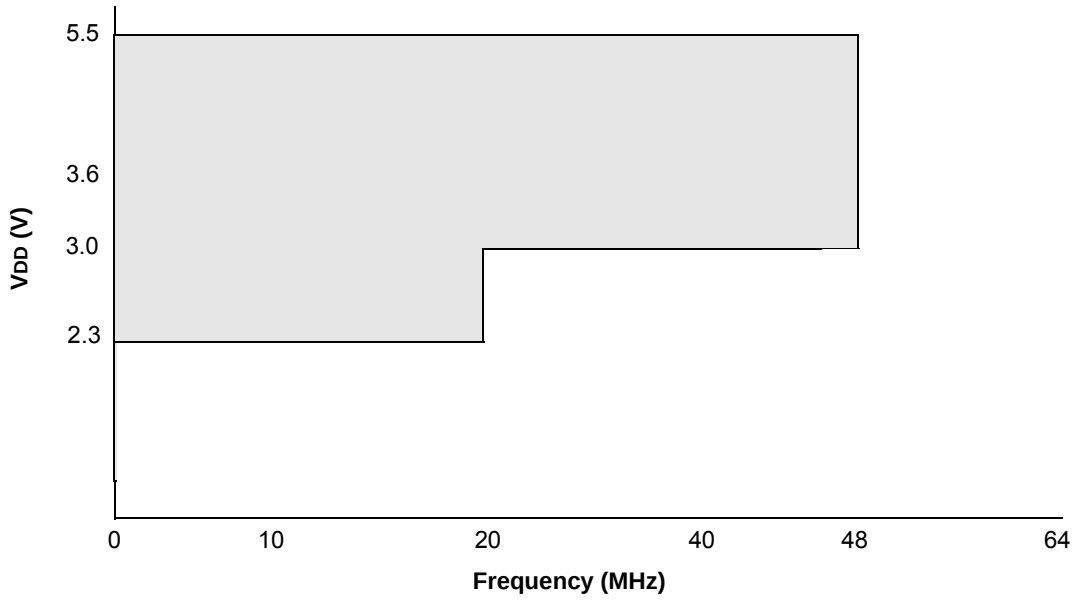

© 2009-2016 Microchip Technology Inc.

**FIGURE 26-1: PIC18F1XK22 VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +85°C**



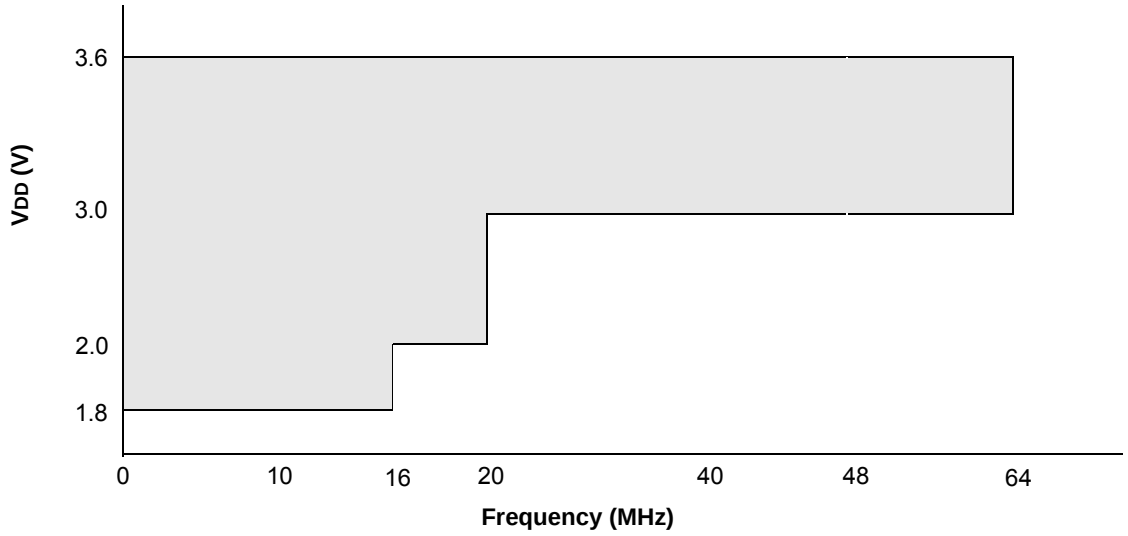

**Note 1:** The shaded region indicates the permissible combinations of voltage and frequency.
**2:** Refer to Table 26-13 for each Oscillator mode's supported frequencies.

**FIGURE 26-2: PIC18F1XK22 VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C**



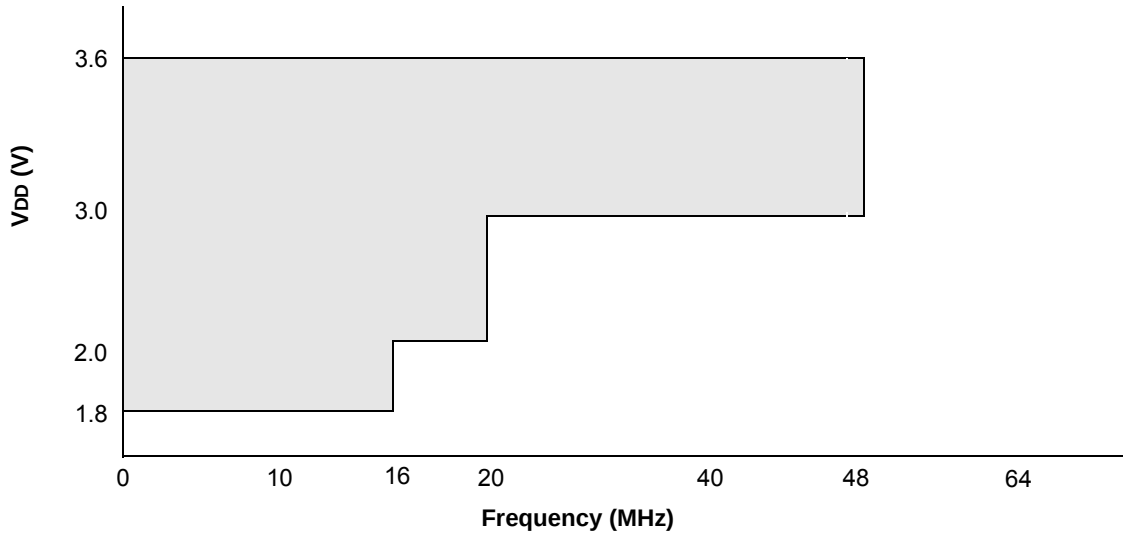

**Note 1:** The shaded region indicates the permissible combinations of voltage and frequency.
**2:** Refer to Table 26-13 for each Oscillator mode's supported frequencies.

**FIGURE 26-3: PIC18LF1XK22 VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +85°C**




**Note 1:** The shaded region indicates the permissible combinations of voltage and frequency.
**2:** Refer to Table 26-13 for each Oscillator mode's supported frequencies.

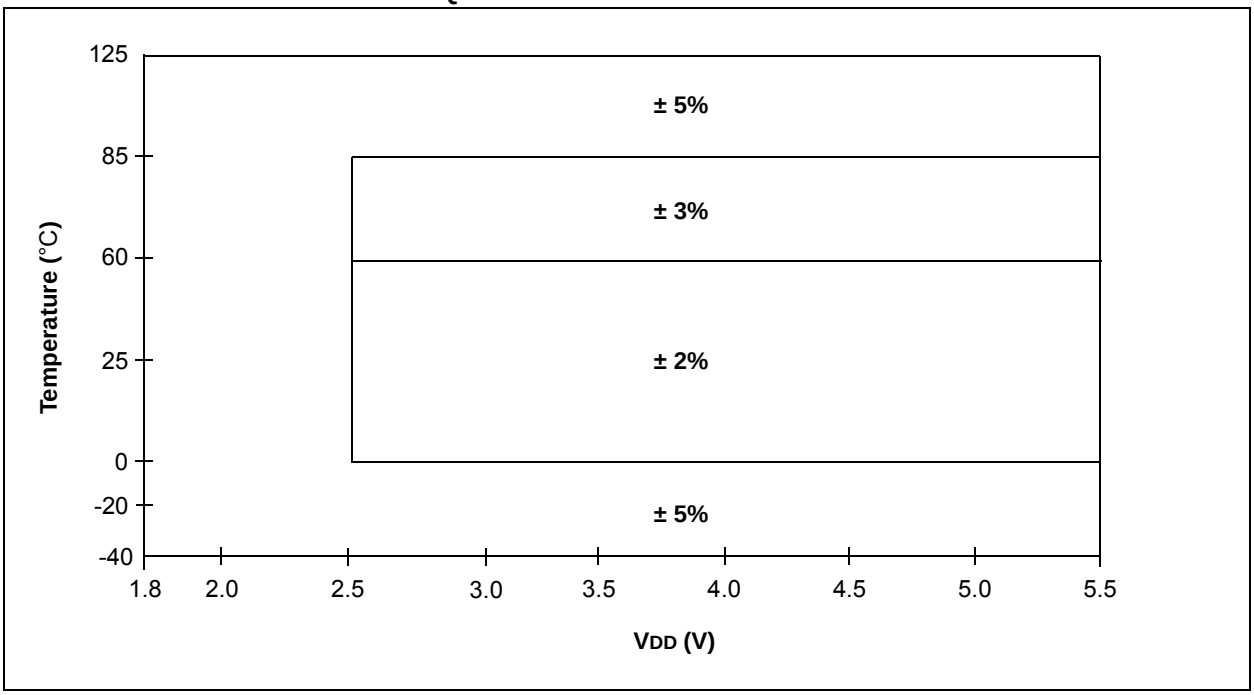
**FIGURE 26-4: PIC18LF1XK22 VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C**




**Note 1:** The shaded region indicates the permissible combinations of voltage and frequency.
**2:** Refer to Table 26-13 for each Oscillator mode's supported frequencies.


© 2009-2016 Microchip Technology Inc.

**FIGURE 26-5: HFINTOSC FREQUENCY ACCURACY OVER DEVICE VDD AND TEMPERATURE**



## 26.3 DC Characteristics

### TABLE 26-1: SUPPLY VOLTAGE

| PIC18LF1XK22 | | | Standard Operating Conditions (unless otherwise stated) | | | | |
|---|---|---|---|---|---|---|---|
| PIC18F1XK22 | | | Standard Operating Conditions (unless otherwise stated) | | | | |
| Param. No. | Sym. | Characteristic | Min. | Typ.† | Max. | Units | Conditions |
| D001 | VDD | **Supply Voltage** | | | | | |
| | | PIC18LF1XK22 | 1.8 | — | 3.6 | V | FOSC ≤ 16 MHz |
| | | | 2.0 | — | 3.6 | V | FOSC ≤ 20 MHz |
| | | | 3.0 | — | 3.6 | V | FOSC ≤ 64 MHz ≤ 85°C |
| | | | 3.0 | — | 3.6 | V | FOSC ≤ 48 MHz ≤ 125°C |
| D001 | | PIC18F1XK22 | 2.3 | — | 5.5 | V | FOSC ≤ 20 MHz |
| | | | 3.0 | — | 5.5 | V | FOSC ≤ 64 MHz ≤ 85°C |
| | | | 3.0 | — | 5.5 | V | FOSC ≤ 48 MHz ≤ 125°C |
| D002* | VDR | **RAM Data Retention Voltage(1)** | | | | | |
| | | PIC18LF1XK22 | 1.5 | — | — | V | Device in Sleep mode |
| D002* | | PIC18F1XK22 | 1.7 | — | — | V | Device in Sleep mode |
| | VPOR* | **Power-on Reset Release Voltage** | — | 1.6 | — | V | |
| | VPORR* | **Power-on Reset Rearm Voltage** | — | 0.8 | — | V | |
| D004* | SVDD | **VDD Rise Rate** to ensure internal Power-on Reset signal | 0.05 | — | — | V/ms | |

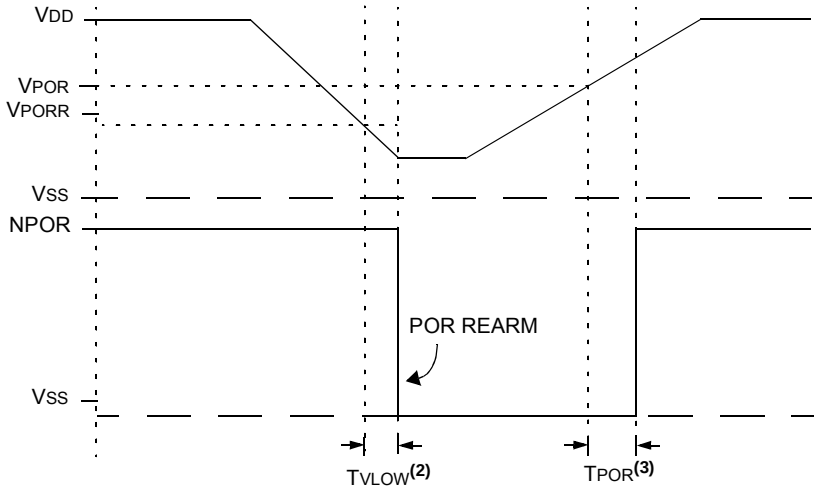
* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 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.


© 2009-2016 Microchip Technology Inc.

**FIGURE 26-6: POR AND POR REARM WITH SLOW RISING VDD**




**Note 1:** When NPOR is low, the device is held in Reset.
**2:** TPOR 1 μs typical.
**3:** TVLOW 2.7 μs typical.

**TABLE 26-2: RC RUN SUPPLY CURRENT**

| PIC18LF1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|---|---|
| PIC18F1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D008 | Supply Current ($I_{DD}$)(1, 2, 4, 5) | 6 | 9 | µA | -40°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 31 kHz(4) (**RC_RUN** mode, LFINTOSC source) |
| | | 7 | 10 | µA | +25°C | | |
| | | 8 | 14 | µA | +85°C | | |
| | | 11 | 17 | µA | +125°C | | |
| D008A | | 11 | 15 | µA | -40°C | $V_{DD}$ = 3.0V | |
| | | 12 | 16 | µA | +25°C | | |
| | | 13 | 25 | µA | +85°C | | |
| | | 17 | 28 | µA | +125°C | | |
| D008 | | 22 | 45 | µA | -40°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 31 kHz(4) (**RC_RUN** mode, LFINTOSC source) |
| | | 23 | 48 | µA | +25°C | | |
| | | 25 | 50 | µA | +85°C | | |
| | | 28 | 55 | µA | +125°C | | |
| D008A | | 25 | 50 | µA | -40°C | $V_{DD}$ = 3.0V | |
| | | 27 | 55 | µA | +25°C | | |
| | | 30 | 60 | µA | +85°C | | |
| | | 32 | 75 | µA | +125°C | | |
| D008B | | 30 | 55 | µA | -40°C | $V_{DD}$ = 5.0V | |
| | | 33 | 60 | µA | +25°C | | |
| | | 37 | 65 | µA | +85°C | | |
| | | 40 | 80 | µA | +125°C | | |
| D009 | | 0.4 | 0.5 | mA | -40°C to +125°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 1 MHz (**RC_RUN** mode, HFINTOSC source) |
| D009A | | 0.6 | 0.8 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D009 | | 0.45 | 0.55 | mA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 1 MHz (**RC_RUN** mode, HFINTOSC source) |
| D009A | | 0.60 | 0.82 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D009B | | 0.80 | 1.0 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |
| D010 | | 1.9 | 2.5 | mA | -40°C to +125°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 16 MHz (**RC_RUN** mode, HF-INTOSC source) |
| D010A | | 3.5 | 4.4 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D010 | | 2.4 | 3.5 | mA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 16 MHz (**RC_RUN** mode, HF-INTOSC source) |
| D010A | | 3.5 | 4.6 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D010B | | 3.7 | 4.7 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |

* These parameters are characterized but not tested.

**Note 1:** The test conditions for all $I_{DD}$ measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to $V_{DD}$; $\overline{MCLR}$ = $V_{DD}$; 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 $R_{EXT}$ is not included. The current through the resistor can be extended by the formula $I_R = V_{DD}/2R_{EXT}$ (mA) with $R_{EXT}$ in kΩ.

**4:** FVR and BOR are disabled.

**5:** When a single temperature range is provided for a parameter, the specification applies to both industrial and extended temperature devices.


© 2009-2016 Microchip Technology Inc.

**TABLE 26-3: RC IDLE SUPPLY CURRENT**

| PIC18LF1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|---|---|
| PIC18F1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D011 | Supply Current ($I_{DD}$)(1, 2, 4, 5) | 2 | 5 | µA | -40°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 31 kHz(4) (**RC_IDLE** mode, LFINTOSC source) |
| | | 2 | 6 | µA | +25°C | | |
| | | 3 | 9 | µA | +85°C | | |
| | | 8 | 11 | µA | +125°C | | |
| D011A | | 4 | 8 | µA | -40°C | $V_{DD}$ = 3.0V | |
| | | 4 | 10 | µA | +25°C | | |
| | | 5 | 13 | µA | +85°C | | |
| | | 8 | 15 | µA | +125°C | | |
| D011 | | 20 | 28 | µA | -40°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 31 kHz(4) (**RC_IDLE** mode, LFINTOSC source) |
| | | 21 | 35 | µA | +25°C | | |
| | | 23 | 41 | µA | +85°C | | |
| | | 24 | 50 | µA | +125°C | | |
| D011A | | 23 | 35 | µA | -40°C | $V_{DD}$ = 3.0V | |
| | | 25 | 40 | µA | +25°C | | |
| | | 28 | 46 | µA | +85°C | | |
| | | 30 | 65 | µA | +125°C | | |
| D011B | | 28 | 43 | µA | -40°C | $V_{DD}$ = 5.0V | |
| | | 30 | 48 | µA | +25°C | | |
| | | 32 | 51 | µA | +85°C | | |
| | | 33 | 71 | µA | +125°C | | |
| D012 | | 0.30 | 0.45 | mA | -40°C to +125°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 1 MHz (**RC_IDLE** mode, HF-INTOSC source) |
| D012A | | 0.45 | 0.60 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D012 | | 0.32 | 0.45 | mA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 1 MHz (**RC_IDLE** mode, HF-INTOSC source) |
| D012A | | 0.47 | 0.62 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D012B | | 0.63 | 0.78 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |
| D013 | | 0.89 | 1.20 | mA | -40°C to +125°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 16 MHz (**RC_IDLE** mode, HF-INTOSC source) |
| D013A | | 1.45 | 2.00 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D013 | | 1.10 | 1.50 | mA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 16 MHz (**RC_IDLE** mode, HF-INTOSC source) |
| D013A | | 1.45 | 2.00 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D013B | | 1.53 | 2.20 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |

* These parameters are characterized but not tested.

**Note 1:** The test conditions for all $I_{DD}$ measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to $V_{DD}$; $\overline{MCLR}$ = $V_{DD}$; 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 $R_{EXT}$ is not included. The current through the resistor can be extended by the formula $I_R = V_{DD}/2R_{EXT}$ (mA) with $R_{EXT}$ in kΩ.

**4:** FVR and BOR are disabled.

**5:** When a single temperature range is provided for a parameter, the specification applies to both industrial and extended temperature devices.

**TABLE 26-4: PRIMARY RUN SUPPLY CURRENT**

| PIC18LF1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|---|---|
| PIC18F1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D014 | Supply Current ($I_{DD}$)(1, 2, 4, 5) | 0.20 | 0.32 | mA | -40°C to +125°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 1 MHz (**PRI_RUN**, EC Med Osc) |
| D014A | | 0.27 | 0.39 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D014 | | .20 | .32 | mA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 1 MHz (**PRI_RUN**, EC Med Osc) |
| D014A | | .27 | .39 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D014B | | .30 | .42 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |
| D015 | | 1.7 | 2.6 | mA | -40°C to +125°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 16 MHz (**PRI_RUN**, EC High Osc) |
| D015A | | 3.0 | 4.2 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D015 | | 2.4 | 3.2 | mA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 16 MHz (**PRI_RUN**, EC High Osc) |
| D015A | | 3.0 | 4.2 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D015B | | 3.3 | 4.4 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |
| D016 | | 11.5 | 14.0 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | $F_{OSC}$ = 64 MHz (**PRI_RUN**, EC High Osc) |
| D016 | | 11.9 | 14.4 | mA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 64 MHz (**PRI_RUN**, EC High Osc) |
| D016A | | 12.1 | 14.6 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |
| D017 | | 2.1 | 2.9 | mA | -40°C to +125°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 4 MHz 16 MHz Internal (**PRI_RUN HS+PLL**) |
| D017A | | 3.1 | 4.0 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D017 | | 2.1 | 2.9 | mA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 4 MHz 16 MHz Internal (**PRI_RUN HS+PLL**) |
| D017A | | 3.1 | 4.0 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D017B | | 3.3 | 4.5 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |
| D018 | | 10 | 15 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | $F_{OSC}$ = 16 MHz 64 MHz Internal (**PRI_RUN HS+PLL**) |
| D018 | | 12.4 | 15.4 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | $F_{OSC}$ = 16 MHz 64 MHz Internal (**PRI_RUN HS+PLL**) |
| D018A | | 12.6 | 15.6 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |

* These parameters are characterized but not tested.

**Note 1:** The test conditions for all $I_{DD}$ measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to $V_{DD}$; $\overline{MCLR}$ = $V_{DD}$; 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 $R_{EXT}$ is not included. The current through the resistor can be extended by the formula $I_R = V_{DD}/2R_{EXT}$ (mA) with $R_{EXT}$ in kΩ.

**4:** FVR and BOR are disabled.

**5:** When a single temperature range is provided for a parameter, the specification applies to both industrial and extended temperature devices.


© 2009-2016 Microchip Technology Inc.

**TABLE 26-5: PRIMARY IDLE SUPPLY CURRENT**

| PIC18LF1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|---|---|
| PIC18F1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D019 | Supply Current ($I_{DD}$)(1, 2, 4, 5) | 70 | 105 | µA | -40°C to +125°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 1 MHz (**PRI_IDLE** mode, EC Med Osc) |
| D019A | | 140 | 180 | µA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D019 | | 80 | 120 | µA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 1 MHz (**PRI_IDLE** mode, EC Med Osc) |
| D019A | | 140 | 180 | µA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D019B | | 151 | 230 | µA | -40°C to +125°C | $V_{DD}$ = 5.0V | |
| D020 | | 1.0 | 1.8 | mA | -40°C to +125°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 16 MHz (**PRI_IDLE**mode, EC High Osc) |
| D020A | | 1.2 | 2.0 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D020 | | 1.0 | 1.8 | mA | -40°C to +125°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 16 MHz (**PRI_IDLE**mode, EC High Osc) |
| D020A | | 1.2 | 2.0 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | |
| D020B | | 1.4 | 2.1 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |
| D021 | | 5.0 | 7.0 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | $F_{OSC}$ = 64 MHz (**PRI_IDLE**mode, EC High Osc) |
| D021 | | 5.2 | 6.2 | mA | -40°C to +125°C | $V_{DD}$ = 3.0V | $F_{OSC}$ = 64 MHz (**PRI_IDLE**mode, EC High Osc) |
| D021A | | 5.3 | 6.3 | mA | -40°C to +125°C | $V_{DD}$ = 5.0V | |

* These parameters are characterized but not tested.

**Note 1:** The test conditions for all $I_{DD}$ measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to $V_{DD}$; $\overline{MCLR}$ = $V_{DD}$; 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 $R_{EXT}$ is not included. The current through the resistor can be extended by the formula $I_R = V_{DD}/2R_{EXT}$ (mA) with $R_{EXT}$ in kΩ.

**4:** FVR and BOR are disabled.

**5:** When a single temperature range is provided for a parameter, the specification applies to both industrial and extended temperature devices.

**TABLE 26-6: SECONDARY RUN SUPPLY CURRENT**

| PIC18LF1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|---|---|
| PIC18F1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D022 | Supply Current ($I_{DD}$)(1, 2, 4) | 6 | 9 | µA | -40°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 32 kHz(3) (**SEC_RUN** mode, Timer1 as clock) |
| | | 6 | 10 | µA | +25°C | | |
| | | 7 | 14 | µA | +85°C | | |
| | | 11 | 17 | µA | +125°C | | |
| D022A | | 11 | 15 | µA | -40°C | $V_{DD}$ = 3.0V | |
| | | 11 | 16 | µA | +25°C | | |
| | | 12 | 25 | µA | +85°C | | |
| | | 26 | 28 | µA | +125°C | | |
| D022 | | 22 | 65 | µA | -40°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 32 kHz(3) (**SEC_RUN** mode, Timer1 as clock) |
| | | 23 | 67 | µA | +25°C | | |
| | | 25 | 69 | µA | +85°C | | |
| | | 28 | 75 | µA | +125°C | | |
| D022A | | 25 | 70 | µA | -40°C | $V_{DD}$ = 3.0V | |
| | | 27 | 72 | µA | +25°C | | |
| | | 30 | 74 | µA | +85°C | | |
| | | 32 | 77 | µA | +125°C | | |
| D022B | | 30 | 75 | µA | -40°C | $V_{DD}$ = 5.0V | |
| | | 32 | 77 | µA | +25°C | | |
| | | 34 | 79 | µA | +85°C | | |
| | | 35 | 83 | µA | +125°C | | |

* These parameters are characterized but not tested.

**Note 1:** The test conditions for all $I_{DD}$ measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to $V_{DD}$; $\overline{MCLR}$ = $V_{DD}$; 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 $R_{EXT}$ is not included. The current through the resistor can be extended by the formula $I_R = V_{DD}/2R_{EXT}$ (mA) with $R_{EXT}$ in kΩ.

**4:** FVR and BOR are disabled.


© 2009-2016 Microchip Technology Inc.

**TABLE 26-7: SECONDARY IDLE SUPPLY CURRENT**

| PIC18LF1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|---|---|
| PIC18F1XK22 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D023 | Supply Current ($I_{DD}$)(1, 2, 4) | 2 | 5 | µA | -40°C | $V_{DD}$ = 1.8V | $F_{OSC}$ = 32 kHz(3) (**SEC_IDLE** mode, Timer1 as clock) |
| | | 2 | 5 | µA | +25°C | | |
| | | 3 | 9 | µA | +85°C | | |
| | | 8 | 11 | µA | +125°C | | |
| D023A | | 4 | 8 | µA | -40°C | $V_{DD}$ = 3.0V | |
| | | 5 | 10 | µA | +25°C | | |
| | | 9 | 20 | µA | +85°C | | |
| | | 20 | 23 | µA | +125°C | | |
| D023 | | 20 | 40 | µA | -40°C | $V_{DD}$ = 2.3V | $F_{OSC}$ = 32 kHz(3) (**SEC_IDLE** mode, Timer1 as clock) |
| | | 21 | 41 | µA | +25°C | | |
| | | 23 | 44 | µA | +85°C | | |
| | | 24 | 47 | µA | +125°C | | |
| D023A | | 23 | 45 | µA | -40°C | $V_{DD}$ = 3.0V | |
| | | 25 | 47 | µA | +25°C | | |
| | | 28 | 49 | µA | +85°C | | |
| | | 30 | 52 | µA | +125°C | | |
| D023B | | 28 | 50 | µA | -40°C | $V_{DD}$ = 5.0V | |
| | | 30 | 54 | µA | +25°C | | |
| | | 32 | 59 | µA | +85°C | | |
| | | 33 | 62 | µA | +125°C | | |

* These parameters are characterized but not tested.

**Note 1:** The test conditions for all $I_{DD}$ measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to $V_{DD}$; $\overline{MCLR}$ = $V_{DD}$; 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 $R_{EXT}$ is not included. The current through the resistor can be extended by the formula $I_R = V_{DD}/2R_{EXT}$ (mA) with $R_{EXT}$ in kΩ.

**4:** FVR and BOR are disabled.

**TABLE 26-8: POWER-DOWN CURRENT**

| PIC18LF1XK22 | | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|---|---|---|
| PIC18F1XK22 | | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Min. | Typ.† | Max. +85°C | Max. +125°C | Units | Conditions | |
| | | | | | | | VDD | Note |
| | **Power-down Base Current (IPD)(2)** | | | | | | | |
| D027 | | — | 0.034 | 1.0 | 9.0 | µA | 1.8 | WDT, BOR, FVR, T1OSC disabled, all Peripherals Inactive |
| | | — | 0.071 | 2.0 | 10 | µA | 3.0 | |
| D027 | | — | 17 | 40 | 55 | µA | 2.3 | WDT, BOR, FVR and T1OSC disabled, all Peripherals Inactive |
| | | — | 18 | 43 | 65 | µA | 3.0 | |
| | | — | 20 | 45 | 80 | µA | 5.0 | |
| | **Power-down Module Current** | | | | | | | |
| D028 | | — | .46 | 1.3 | 10 | µA | 1.8 | LPWDT Current(1) |
| | | — | .74 | 3.0 | 11 | µA | 3.0 | |
| D028 | | — | 18 | 44 | 60 | µA | 2.3 | LPWDT Current(1) |
| | | — | 21 | 46 | 70 | µA | 3.0 | |
| | | — | 22 | 48 | 85 | µA | 5.0 | |
| D029 | | — | 12 | 20 | 28 | µA | 1.8 | FVR Current (3) |
| | | — | 14 | 22 | 30 | µA | 3.0 | |
| D029 | | — | 40 | 65 | 80 | µA | 2.3 | FVR Current(3) |
| | | — | 50 | 70 | 85 | µA | 3.0 | |
| | | — | 70 | 120 | 135 | µA | 5.0 | |
| D030 | | — | 12 | 17 | 23 | µA | 3.0 | BOR Current(1, 3) |
| D030 | | — | 30 | 55 | 80 | µA | 3.0 | BOR Current(1, 3) |
| | | — | 64 | 100 | 120 | µA | 5.0 | |
| D031 | | — | .65 | 1.5 | 11 | µA | 1.8 | T1OSC Current(1) |
| | | — | 0.90 | 4.0 | 12 | µA | 3.0 | |
| D031 | | — | 19 | 45 | 60 | µA | 2.3 | T1OSC Current(1) |
| | | — | 20 | 50 | 70 | µA | 3.0 | |
| | | — | 22 | 55 | 80 | µA | 5.0 | |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 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 Δ 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.


© 2009-2016 Microchip Technology Inc.

**TABLE 26-8: POWER-DOWN CURRENT (CONTINUED)**

| PIC18LF1XK22 | | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|---|---|---|
| PIC18F1XK22 | | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Min. | Typ.† | Max. +85°C | Max. +125°C | Units | Conditions | |
| | | | | | | | VDD | Note |
| | **Power-down Module Current** | | | | | | | |
| D032 | | — | .7 | 1.0 | 9.0 | µA | 1.8 | A/D Current(1, 4), no conversion in progress |
| | | — | .8 | 3.0 | 10 | µA | 3.0 | |
| D032 | | — | 19 | 42 | 60 | µA | 2.3 | A/D Current(1, 4), no conversion in progress |
| | | — | 20 | 44 | 65 | µA | 3.0 | |
| | | — | 22 | 46 | 80 | µA | 5.0 | |
| D033 | | — | 8 | 30 | 32 | µA | 1.8 | Comparator Current, low power C1 and C2 enabled |
| | | — | 11 | 32 | 35 | µA | 3.0 | |
| D033 | | — | 23 | 55 | 65 | µA | 2.3 | Comparator Current, low power C1 and C2 enabled |
| | | — | 31 | 65 | 75 | µA | 3.0 | |
| | | — | 33 | 75 | 95 | µA | 5.0 | |
| D033A | | — | 44 | 110 | 160 | µA | 1.8 | Comparator Current, high power C1 and C2 enabled |
| | | — | 65 | 130 | 165 | µA | 3.0 | |
| D033A | | — | 77 | 137 | 155 | µA | 2.3 | Comparator Current, high power C1 and C2 enabled |
| | | — | 84 | 140 | 165 | µA | 3.0 | |
| | | — | 90 | 150 | 180 | µA | 5.0 | |
| D034 | | — | 13 | 18 | 33 | µA | 1.8 | FVR Current |
| | | — | 22 | 30 | 40 | µA | 3.0 | |
| D034 | | — | 33 | 55 | 85 | µA | 2.3 | FVR Current |
| | | — | 35 | 80 | 95 | µA | 3.0 | |
| | | — | 48 | 90 | 120 | µA | 5.0 | |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 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 Δ 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.

**TABLE 26-9: I/O PORTS**

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) | | | | |
|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | Min. | Typ.† | Max. | Units | Conditions |
| | VIL | **Input Low Voltage** | | | | | |
| | | I/O ports: | | | | | |
| D036 | | with TTL buffer | VSS | — | 0.8 | V | 4.5V ≤ VDD ≤ 5.5V |
| D036A | | | VSS | — | 0.15 VDD | V | 1.8V ≤ VDD ≤ 4.5V |
| D036B | | | VSS | — | 0.2 VDD | V | 2.0V ≤ VDD ≤ 5.5V |
| D037 | | with Schmitt Trigger buffer | VSS | — | 0.2 VDD | V | 1.8V ≤ VDD ≤ 5.5V |
| D037A | | with $I^2C$ levels | VSS | — | 0.3 VDD | V | |
| D037B | | with SMBus levels | VSS | — | 0.8 VDD | V | 2.7V ≤ VDD ≤ 5.5V |
| D038 | | $\overline{MCLR}$ | VSS | — | 0.2 VDD | V | |
| D039 | | OSC1 | VSS | — | 0.3 VDD | V | HS, HSPLL modes |
| D039A | | OSC1 | VSS | — | 0.2 VDD | V | EC, RC modes(1) |
| D039B | | OSC1 | VSS | — | 0.3 VDD | V | XT, LP modes |
| D039C | | T1CKI | VSS | — | 0.3 VDD | V | |
| | VIH | **Input High Voltage** | | | | | |
| | | I/O ports: | | | | | |
| D040 | | with TTL buffer | 2.0 | — | VDD | V | 4.5V ≤ VDD ≤ 5.5V |
| D040A | | | 0.25 VDD + 0.8 | — | VDD | V | 1.8V ≤ VDD ≤ 4.5V |
| D041 | | with Schmitt Trigger buffer | 0.8 VDD | — | VDD | V | 1.8V ≤ VDD ≤ 5.5V |
| D041A | | with $I^2C$ levels | 0.7 VDD | — | VDD | V | |
| D037A | | with SMBus levels | 2.1 | — | VDD | V | 2.7V ≤ VDD ≤ 5.5V |
| D042 | | $\overline{MCLR}$ | 0.8 VDD | — | VDD | V | |
| D042A | | $\overline{MCLR}$ | 0.9 VDD | — | 0.3 VDD | V | 1.8V ≤ VDD ≤ 2.4V |
| D043 | | OSC1 | 0.7 VDD | — | VDD | V | HS, HSPLL modes |
| D043A | | OSC1 | 0.8 VDD | — | VDD | V | EC mode |
| D043B | | OSC1 | 0.9 VDD | — | VDD | V | RC mode(1) |
| D043C | | OSC1 | 1.6 | — | VDD | V | XT, LP modes |
| D043E | | T1CKI | 1.6 | — | VDD | V | |
| | IIL | **Input Leakage Current(2)** | | | | | |
| D060 | | I/O ports | — | ± 5 | ± 100 | nA | VSS ≤ VPIN ≤ VDD, Pin at high-impedance, -40°C to 85°C |
| | | | — | ± 5 | ± 1000 | nA | VSS ≤ VPIN ≤ VDD, 85°C to 125°C |
| D061 | | $\overline{MCLR}$(3) | — | ± 50 | ± 200 | nA | VSS ≤ VPIN ≤ VDD |
| | IPUR | **PORTB Weak Pull-up Current** | | | | | |
| D070* | | | 50 | 250 | 400 | µA | VDD = 5.0V, VPIN = VSS |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 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 $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**4:** Including OSC2 in CLKOUT mode.


© 2009-2016 Microchip Technology Inc.

**TABLE 26-9: I/O PORTS (CONTINUED)**

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| D080 | VOL | **Output Low Voltage(4)** | | | | | |
| | | I/O ports | — | — | VSS+0.6<br>VSS+0.6<br>VSS+0.6 | V | IOL = 8 mA, VDD = 5V<br>IOL = 6 mA, VDD = 3.3V<br>IOL = 3 mA, VDD = VDDMIN |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 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 $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**4:** Including OSC2 in CLKOUT mode.

**TABLE 26-9: I/O PORTS (CONTINUED)**

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) | | | | |
|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | Min. | Typ.† | Max. | Units | Conditions |
| | VOH | **Output High Voltage(4)** | | | | | |
| D090 | | I/O ports | VDD-0.7<br>VDD-0.7<br>VDD-0.7 | — | — | V | IOH = 3.5 mA, VDD = 5V<br>IOH = 3 mA, VDD = 3.3V<br>IOH = 2 mA, VDD = VDDMIN |
| | | **Capacitive Loading Specs on Output Pins** | | | | | |
| D101* | COSC2 | 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 | |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 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 $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**4:** Including OSC2 in CLKOUT mode.


© 2009-2016 Microchip Technology Inc.

**TABLE 26-10: MEMORY PROGRAMMING REQUIREMENTS**

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) | | | | |
|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | Min. | Typ.† | Max. | Units | Conditions |
| | | **Internal Program Memory Programming Specifications(1)** | | | | | |
| D110 | VPP | Voltage on $\overline{MCLR}$/VPP/RA3 pin | 8 | — | 9 | V | **(Note 3, Note 4)** |
| D113 | IDDP | Supply Current during Programming | — | — | 10 | mA | |
| | | **Data EEPROM Memory(2)** | | | | | |
| D120 | ED | Byte Endurance | 100K | — | — | E/W | -40°C to +85°C |
| D121 | VDRW | VDD for Read/Write | VDDMIN | — | VDDMAX | V | Using EECON to read/write |
| D122 | TDEW | Erase/Write Cycle Time | — | 3 | 4 | ms | |
| D123 | TRETD | Characteristic Retention | — | 40 | — | Year | Provided no other specifications are violated |
| D124 | TREF | Number of Total Erase/Write Cycles before Refresh(2) | 1M | 10M | — | E/W | -40°C to +85°C |
| D130 | | **Program Flash Memory** | | | | | |
| | EP | Cell Endurance | 10k | — | — | E/W | Temperature during programming: 10°C ≤ TA ≤ 40°C |
| D131 | VPR | VDD for Read | VDDMIN | — | VDDMAX | V | |
| D131A | | Voltage on $\overline{MCLR}$/VPP during Erase/Program | 8.0 | — | 9.0 | V | Temperature during programming: 10°C ≤ TA ≤ 40°C |
| D131B | VBE | VDD for Bulk Erase | 2.7 | — | VDDMAX | V | Temperature during programming: 10°C ≤ TA ≤ 40°C |
| D132 | VPEW | VDD for Write or Row Erase | 2.2<br>VDDMIN | —<br>— | VDDMAX<br>VDDMAX | V | PIC18LF1XK22<br>PIC18F1XK22 |
| D132A | IPPPGM | Current on $\overline{MCLR}$/VPP during Erase/Write | — | 1.0 | — | mA | Temperature during programming: 10°C ≤ TA ≤ 40°C |
| D132B | IDDPGM | Current on VDD during Erase/Write | — | 5.0 | — | mA | Temperature during programming: 10°C ≤ TA ≤ 40°C |
| D133 | TPEW | Erase/Write cycle time | — | 2.0 | 2.8 | ms | Temperature during programming: 10°C ≤ TA ≤ 40°C |
| D134 | TRETD | Characteristic Retention | — | 40 | — | Year | Provided no other specifications are violated |

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** These specifications are for programming the on-chip program memory through the use of table write instructions.

**2:** Refer to **Section 5.8 "Using the Data EEPROM"** for a more detailed discussion on data EEPROM endurance.

**3:** Required only if single-supply programming is disabled.

**4:** The MPLAB ICD 2 does not support variable VPP output. Circuitry to limit the ICD 2 VPP voltage must be placed between the ICD 2 and target system when programming or debugging with the ICD 2.

**TABLE 26-11: THERMAL CHARACTERISTICS**

| Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Typ.** | **Units** | **Conditions** |
| TH01 | $\theta_{JA}$ | Thermal Resistance Junction to Ambient | 62.2 | °C/W | 20-pin PDIP package |
| | | | 75.0 | °C/W | 20-pin SOIC package |
| | | | 89.3 | °C/W | 20-pin SSOP package |
| | | | 43.0 | °C/W | 20-pin QFN 4x4mm package |
| TH02 | $\theta_{JC}$ | Thermal Resistance Junction to Case | 27.5 | °C/W | 20-pin PDIP package |
| | | | 23.1 | °C/W | 20-pin SOIC package |
| | | | 31.1 | °C/W | 20-pin SSOP package |
| | | | 5.3 | °C/W | 20-pin QFN 4x4mm package |
| TH03 | $T_{JMAX}$ | Maximum Junction Temperature | 150 | °C | |
| TH04 | PD | Power Dissipation | — | W | PD = $P_{INTERNAL}$ + $P_{I/O}$ |
| TH05 | $P_{INTERNAL}$ | Internal Power Dissipation | — | W | $P_{INTERNAL}$ = $I_{DD}$ x $V_{DD}$(1) |
| TH06 | $P_{I/O}$ | I/O Power Dissipation | — | W | $P_{I/O}$ = Σ ($I_{OL}$ * $V_{OL}$) + Σ ($I_{OH}$ * ($V_{DD}$ - $V_{OH}$)) |
| TH07 | $P_{DER}$ | Derated Power | — | W | $P_{DER}$ = $P_{DMAX}$ ($T_J$ - $T_A$)/$\theta_{JA}$(2) |

**Note 1:** $I_{DD}$ is current to run the chip alone without driving any load on the output pins.
**2:** $T_A$ = Ambient Temperature.
**3:** $T_J$ = Junction Temperature.


© 2009-2016 Microchip Technology Inc.

## 26.4 Timing Parameter Symbology

The timing parameter symbols have been created with one of the following formats:

1. TppS2ppS
2. TppS

| **T** | | | |
|---|---|---|---|
| F | Frequency | T | Time |

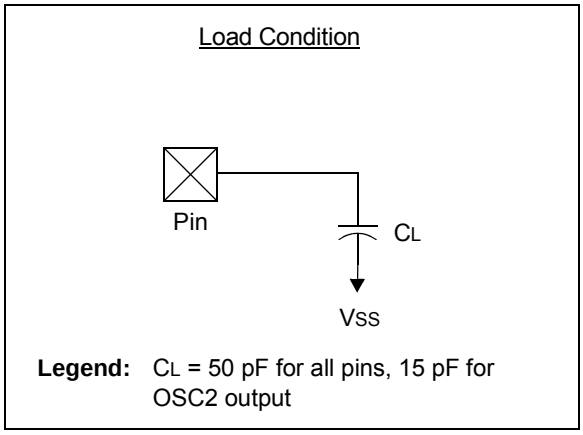
Lowercase letters (pp) and their meanings:

| **pp** | | | |
|---|---|---|---|
| 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** | | | |
|---|---|---|---|
| F | Fall | P | Period |
| H | High | R | Rise |
| I | Invalid (High-impedance) | V | Valid |
| L | Low | Z | High-impedance |

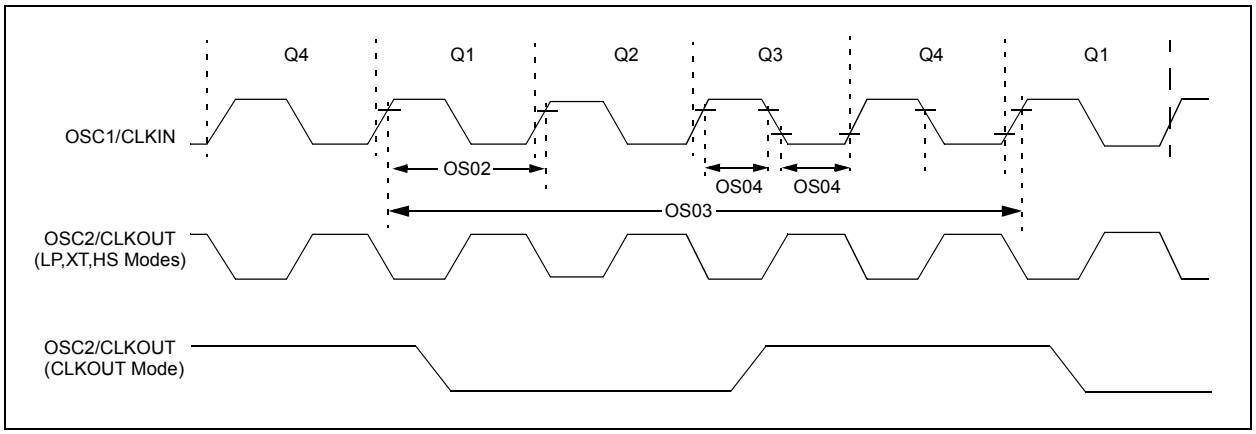
**FIGURE 26-7: LOAD CONDITIONS**




**Legend:** CL = 50 pF for all pins, 15 pF for OSC2 output

## 26.5 AC Characteristics: PIC18(L)F1XK22-I/E

**FIGURE 26-8: CLOCK TIMING**





© 2009-2016 Microchip Technology Inc.

**TABLE 26-12: EXTERNAL CLOCK TIMING REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| 1A | FOSC | External CLKIN Frequency(1) | DC | 48 | MHz | EC, ECIO Oscillator mode, (Extended Range Devices) |
| | | | DC | 64 | MHz | EC, ECIO Oscillator mode, (Industrial Range Devices) |
| | | Oscillator Frequency(1) | DC | 4 | MHz | RC Oscillator mode |
| | | | 0.1 | 4 | MHz | XT Oscillator mode |
| | | | 4 | 25 | MHz | HS Oscillator mode |
| | | | 4 | 16 | MHz | HS + PLL Oscillator mode, (Industrial Range Devices) |
| | | | 4 | 12 | MHz | HS + PLL Oscillator mode, (Extended Range Devices) |
| | | | 5 | 33 | kHz | LP Oscillator mode |
| 1 | TOSC | External CLKIN Period(1) | 20.8 | — | ns | EC, ECIO, Oscillator mode (Extended Range Devices) |
| | | | 15.6 | — | ns | EC, ECIO, Oscillator mode, (Industrial Range Devices) |
| | | Oscillator Period(1) | 250 | — | ns | RC Oscillator mode |
| | | | 250 | 10,000 | ns | XT Oscillator mode |
| | | | 40 | 250 | ns | HS Oscillator mode |
| | | | 62.5 | 250 | ns | HS + PLL Oscillator mode, (Industrial range devices) |
| | | | 83.3 | 250 | ns | HS + PLL Oscillator mode, (Extended Range Devices) |
| | | | 30 | 200 | μs | LP Oscillator mode |
| 2 | TCY | Instruction Cycle Time(1) | 62.5 | — | ns | TCY = 4/FOSC |
| 3 | TOSL, TOSH | External Clock in (OSC1) High or Low Time | 30 | — | ns | XT Oscillator mode |
| | | | 2.5 | — | μs | LP Oscillator mode |
| | | | 10 | — | ns | HS Oscillator mode |
| 4 | TOSR, TOSF | External Clock in (OSC1) Rise or Fall Time | — | 20 | ns | XT Oscillator mode |
| | | | — | 50 | ns | LP Oscillator mode |
| | | | — | 7.5 | ns | HS Oscillator mode |

**Note 1:** Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations except PLL. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKIN pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

**TABLE 26-13: OSCILLATOR PARAMETERS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Freq. Tolerance** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| OS08 | HFOSC | Internal Calibrated HFINTOSC Frequency(2) | ±2% | — | 16.0 | — | MHz | 0°C ≤ TA ≤ 60°C |
| | | | ±3% | — | 16.0 | — | MHz | 60°C ≤ TA ≤ +85°C |
| | | | ±5% | — | 16.0 | — | MHz | |
| OS09 | LFOSC | Internal LFINTOSC Frequency | 0 | — | 31.25 | — | kHz | |
| OS10* | TIOSC ST | HFINTOSC Wake-up from Sleep Start-up Time | — | — | 5 | 8 | µs | VDD = 2.0V, -40°C to +85°C |
| | | | — | — | 5 | 8 | µs | VDD = 3.0V, -40°C to +85°C |
| | | | — | — | 5 | 8 | µs | VDD = 5.0V, -40°C to +85°C |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 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 µF and 0.01 µF values in parallel are recommended.

**3:** By design.

**TABLE 26-14: PLL CLOCK TIMING SPECIFICATIONS (VDD = 1.8V TO 5.5V)**

| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
|---|---|---|---|---|---|---|---|
| F10 | FOSC | Oscillator Frequency Range | 4 | — | 5 | MHz | VDD = 1.8-3.0V |
| | | | 4 | — | 16 | MHz | VDD = 3.0-5.0V, -40°C to +85°C |
| | | | 4 | — | 12 | MHz | VDD = 3.0-5.0V, 125°C |
| F11 | FSYS | On-Chip VCO System Frequency | 16 | — | 20 | MHz | VDD = 1.8-3.0V |
| | | | 16 | — | 64 | MHz | VDD = 3.0-5.0V, -40°C to +85°C |
| | | | 16 | — | 48 | MHz | VDD = 3.0-5.0V, 125°C |
| F12 | $t_{rc}$ | PLL Start-up Time (Lock Time) | — | — | 2 | ms | |
| F13* | ΔCLK | CLKOUT Stability (Jitter) | -0.25 | — | +0.25 | % | |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.


© 2009-2016 Microchip Technology Inc.

**FIGURE 26-9: CLKOUT AND I/O TIMING**

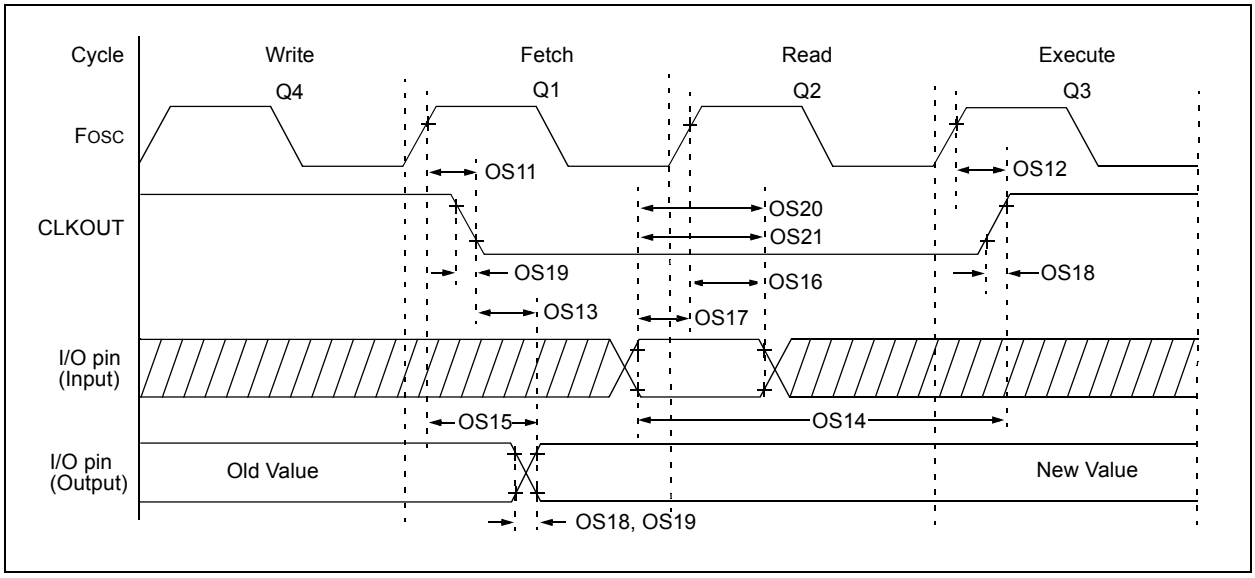



**TABLE 26-15: CLKOUT AND I/O TIMING PARAMETERS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| OS11 | TOSH2CKL | Fosc↑ to CLKOUT↓ (1) | — | — | 70 | ns | VDD = 3.3-5.0V |
| OS12 | TOSH2CKH | Fosc↑ to CLKOUT↑ (1) | — | — | 72 | ns | VDD = 3.3-5.0V |
| OS13 | TCKL2IOV | CLKOUT↓ to Port out valid(1) | — | — | 20 | ns | |
| OS14 | TIOV2CKH | Port input valid before CLKOUT↑(1) | TOSC + 200 ns | — | — | ns | |
| OS15 | TOSH2IOV | Fosc↑ (Q1 cycle) to Port out valid | — | 50 | 70* | ns | VDD = 3.3-5.0V |
| OS16 | TOSH2IOI | Fosc↑ (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↑ (Q2 cycle) (I/O in setup time) | 20 | — | — | ns | |
| OS18 | TIOR | Port output rise time(2) | —<br>— | 40<br>15 | 72<br>32 | ns | VDD = 1.8V<br>VDD = 3.3-5.0V |
| OS19 | TIOF | Port output fall time(2) | —<br>— | 28<br>15 | 55<br>30 | ns | VDD = 1.8V<br>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 | 25 | — | — | ns | |

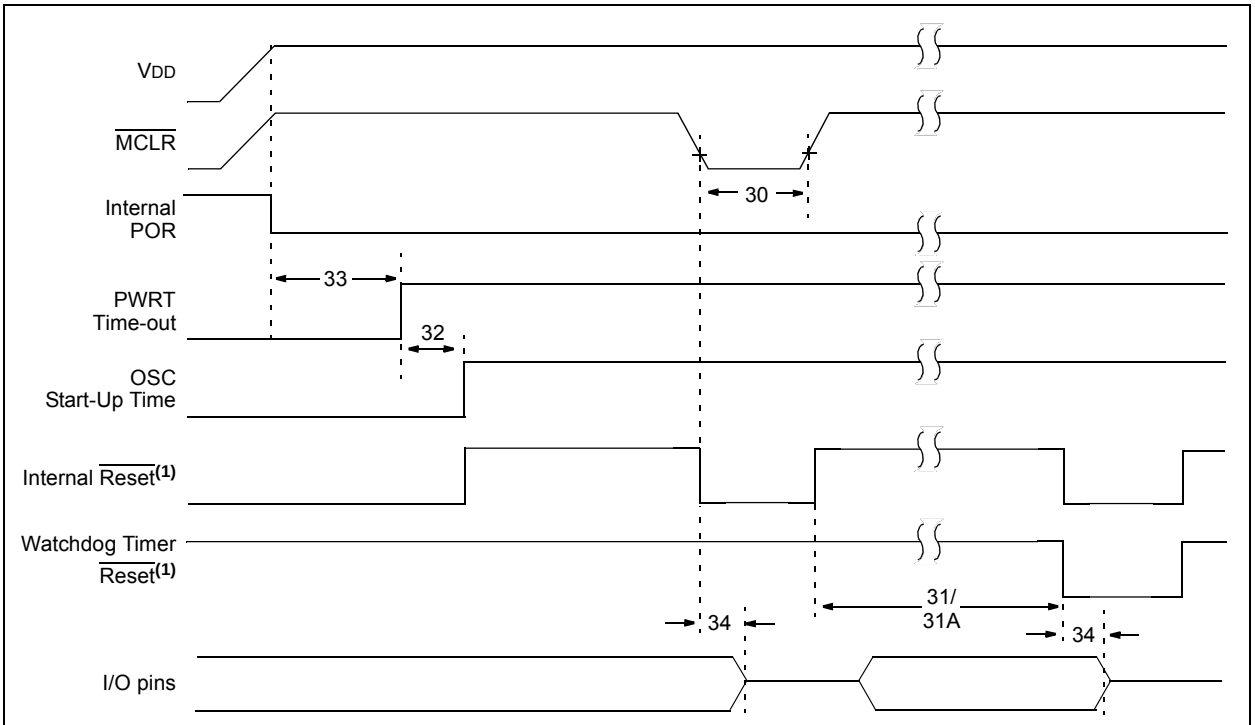
* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated.

**Note 1:** Measurements are taken in RC mode where CLKOUT output is 4 x TOSC.

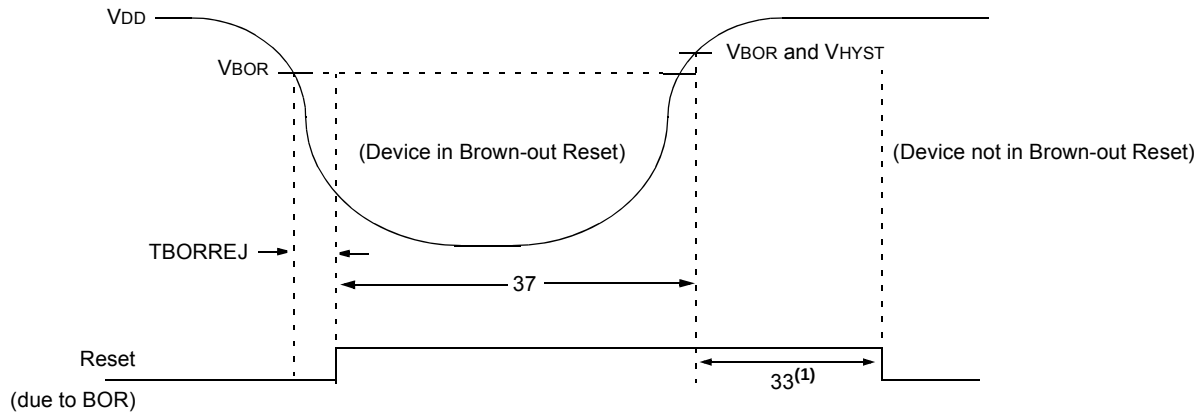
**2:** Includes OSC2 in CLKOUT mode.

**FIGURE 26-10: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**




**Note 1:** Asserted low.

**FIGURE 26-11: BROWN-OUT RESET TIMING AND CHARACTERISTICS**




**Note 1:** 64 ms delay only if PWRTE bit in the Configuration Word register is programmed to '0'. 2 ms delay if PWRTE = 0.


© 2009-2016 Microchip Technology Inc.

**TABLE 26-16: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, AND BROWN-OUT RESET PARAMETERS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| 30 | TMCL | $\overline{MCLR}$ Pulse Width (low) | 2<br>5 | —<br>— | —<br>— | µs<br>µs | VDD = 3.3-5V, -40°C to +85°C<br>VDD = 3.3-5V |
| 31 | TWDT | Standard Watchdog Timer Time-out Period (1:16 Prescaler) | 10<br>10 | 17<br>17 | 27<br>30 | ms<br>ms | VDD = 3.3V-5V, -40°C to +85°C<br>VDD = 3.3V-5V |
| 31A | TWDTLP | Low-Power Watchdog Timer Time-out Period (No Prescaler) | 10<br>10 | 18<br>18 | 27<br>33 | ms<br>ms | VDD = 3.3V-5V, -40°C to +85°C<br>VDD = 3.3V-5V |
| 32 | TOST | Oscillator Start-up Timer Period(1,2) | — | 1024 | — | TOSC | **(Note 3)** |
| 33* | TPWRT | Power-up Timer Period, $\overline{PWRTE}$ = 0 | 40 | 65 | 140 | ms | |
| 34* | TIOZ | I/O high-impedance from $\overline{MCLR}$ Low or Watchdog Timer Reset | — | — | 2.73 | µs | |
| 35 | VBOR | Brown-out Reset Voltage | 1.75<br>2.05<br>2.35<br>2.65 | 1.9<br>2.2<br>2.5<br>2.85 | 2.05<br>2.35<br>2.65<br>3.05 | V<br>V<br>V<br>V | BORV = 1.9V(5)<br>BORV = 2.2V(5)<br>BORV = 2.7V<br>BORV = 2.85V |
| 36* | VHYST | Brown-out Reset Hysteresis | 0 | 25 | 50 | mV | -40°C to +85°C |
| 37* | TBORDC | Brown-out Reset DC Response Time | 0 | 3 | 35 | µs | VDD ≤ VBOR |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3V, 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 µF and 0.01 µF values in parallel are recommended.

**5:** PIC18LF1XK22 devices only.

**FIGURE 26-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**

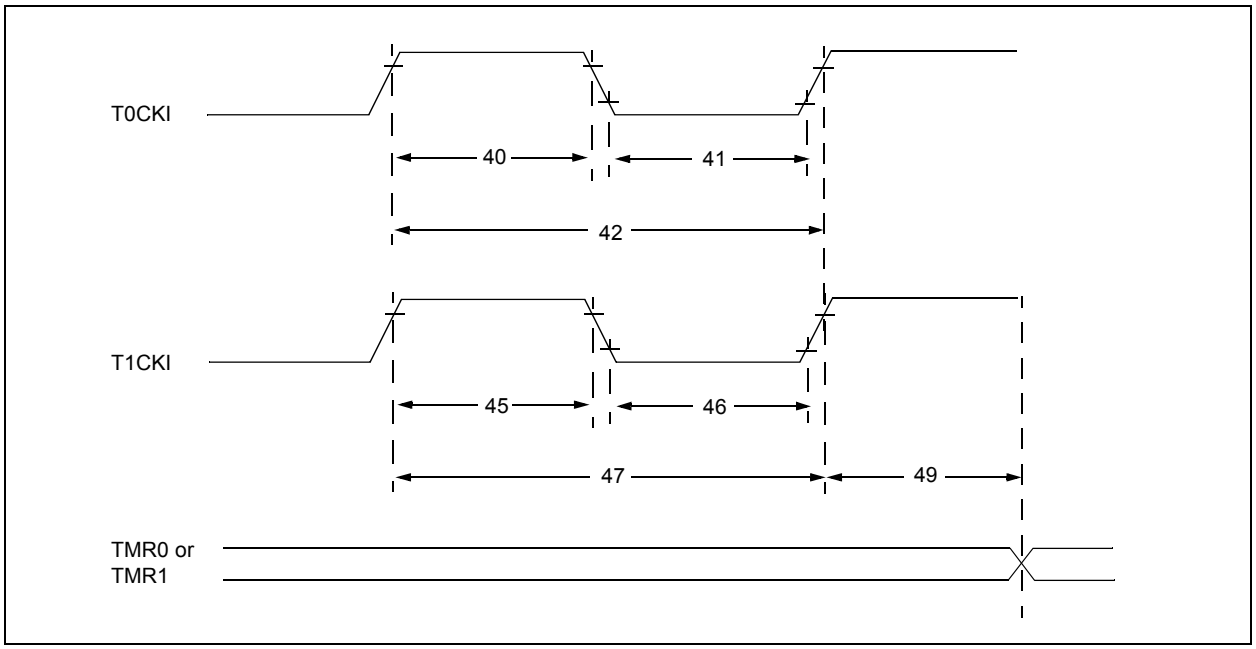



**TABLE 26-17: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

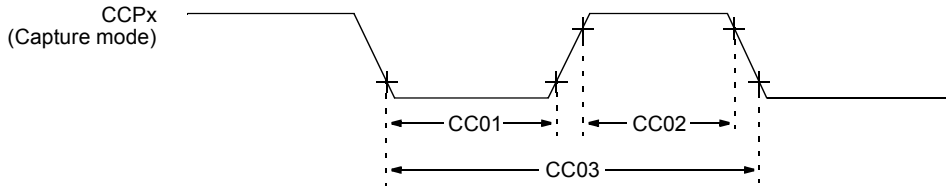
| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| 40* | TT0H | T0CKI High-Pulse Width | No Prescaler | 0.5 TCY + 20 | — | — | 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 TCY + 40 / 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 TCY + 40 / N | — | — | ns | N = prescale value (1, 2, 4, 8) |
| | | | Asynchronous | 60 | — | — | ns | |
| 48 | FT1 | 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 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.


© 2009-2016 Microchip Technology Inc.

**FIGURE 26-13: CAPTURE/COMPARE/PWM TIMINGS (CCP)**




**Note:** Refer to Figure 26-7 for load conditions.

**TABLE 26-18: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **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 | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC03* | TccP | CCPx Input Period | | $\frac{3T_{CY} + 40}{N}$ | — | — | ns | N = prescale value (1, 4 or 16) |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**TABLE 26-19: PIC18(L)F1XK22 A/D CONVERTER (ADC) CHARACTERISTICS**

| Standard Operating Conditions (unless otherwise stated)<br>Operating temperature: Tested at 25°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| AD01 | NR | Resolution | — | — | 10 | bit | |
| AD02 | EIL | Integral Error | — | — | ±2 | LSb | VREF = 3.0V |
| AD03 | EDL | Differential Error | — | — | ±1.5 | LSb | No missing codes<br>VREF = 3.0V |
| AD04 | EOFF | Offset Error | — | — | ±3 | LSb | VREF = 3.0V |
| AD05 | EGN | Gain Error | — | — | ±3 | LSb | VREF = 3.0V |
| AD06 | VREF | Change in Reference Voltage = VREF+ - VREF- (2), (3) | 1.8 | — | VDD | V | 1.8 ≤ VREF+ ≤ VDD + 0.3V<br>VSS - 0.3V ≤ VREF- ≤ VREF+ - 1.8V |
| AD07 | VAIN | Full-Scale Range | VSS | — | VREF | V | |
| AD08 | ZAIN | Recommended Impedance of Analog Voltage Source | — | — | 10 | kΩ | Can go higher if external 0.01 µF capacitor is present on input pin. |

* These parameters are characterized but not tested.

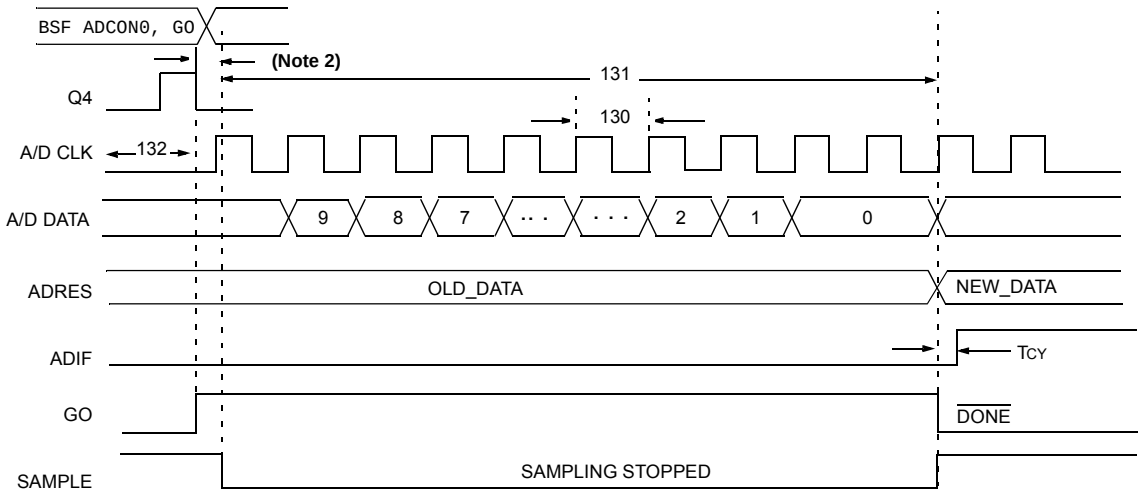
† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Total Absolute Error includes integral, differential, offset and gain errors.

**2:** ADC VREF is from external VREF, VDD pin or FVR, whichever is selected as reference input.

**3:** FVR voltage selected must be 2.048V or 4.096V.

**FIGURE 26-14: A/D CONVERSION TIMING**




**Note 1:** If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

**2:** This is a minimal RC delay (typically 100 ns), which also disconnects the holding capacitor from the analog input.

**TABLE 26-20: A/D CONVERSION REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| 130* | TAD | A/D Clock Period | 0.7 | 25.0(1) | µs | TOSC based, VREF ≥ 3.0V, -40°C to +85°C |
| | | | 0.7 | 4.0(1) | µs | TOSC based, VREF ≥ 3.0V, -40°C to +125°C |
| | | | 1.0 | 4.0 | µs | A/D RC mode |
| 131 | TCNV | Conversion Time (not including acquisition time)(2) | 12 | 12 | TAD | |
| 132* | TACQ | Acquisition Time(3) | 1.4 | 5.0 | µs | VDD ≥ 3.0V, RS = 50Ω |
| 135 | TSWC | Switching Time from Convert - Sample | — | —(4) | — | |
| 136 | TDIS | Discharge Time | 2 | 2 | µs | |

* These parameters are characterized but not tested.

**Note 1:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

**2:** ADRES register may be read on the following TCY cycle.

**3:** The time for the holding capacitor to acquire the 'new' input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD). The source impedance (RS) on the input channels is 50.

**4:** On the following cycle of the device clock.


© 2009-2016 Microchip Technology Inc.

**TABLE 26-21: COMPARATOR SPECIFICATIONS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristics** | **Min.** | **Typ.** | **Max.** | **Units** | **Comments** |
| CM01 | VIOFF | Input Offset Voltage | — | 10 | 50 | mV | VREF = VDD/2, High-Power mode |
| | | | — | 12 | 80 | mV | VREF = VDD/2, Low-Power mode |
| CM02 | VICM | Input Common-mode Voltage | VSS | — | VDD | V | |
| CM04 | TRESP | Response Time | — | 200 | 400 | ns | High-Power mode |
| | | | — | 300 | 600 | ns | Low-Power mode |
| CM05 | TMC2OV | Comparator Mode Change to Output Valid* | — | — | 10 | µs | |

* 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 26-22: DIGITAL-TO-ANALOG CONVERTER (DAC) SPECIFICATIONS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristics** | **Min.** | **Typ.** | **Max.** | **Units** | **Comments** |
| DAC01* | CLSB | Step Size | — | VDD/32 | — | V | |
| DAC02* | CACC | Absolute Accuracy | — | — | ± 1/2 | LSb | |
| DAC03* | CR | Unit Resistor Value (R) | — | 5k | — | Ω | |
| DAC04* | CST | Settling Time(1) | — | — | 10 | µs | |

* These parameters are characterized but not tested.

**Note 1:** Settling time measured while DACR<4:0> transitions from '0000' to '1111'.

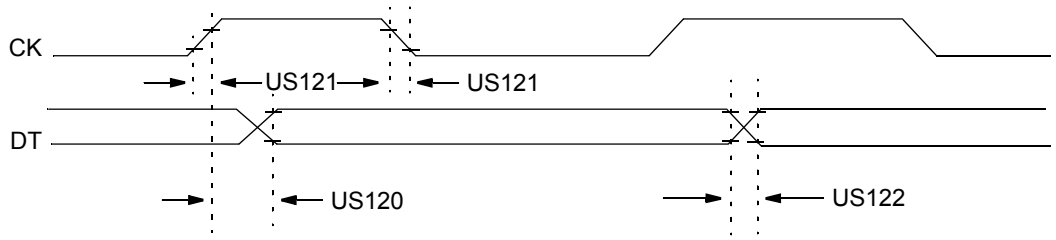
**TABLE 26-23: FIXED VOLTAGE REFERENCE (FVR) SPECIFICATIONS**

| VR Voltage Reference Specifications | | | Standard Operating Conditions (unless otherwise stated) | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristics** | **Min.** | **Typ.** | **Max.** | **Units** | **Comments** |
| D003 | VADFVR | **Fixed Voltage Reference Voltage for ADC, Initial Accuracy** | -8 | — | 6 | % | 1.024V, VDD ≥ 2.5V(1)<br>2.048V, VDD ≥ 2.5V<br>4.096V, VDD ≥ 4.75V |
| D003A | VCDAFVR | Fixed Voltage Reference Voltage for Comparator and DAC, Initial Accuracy | -11 | — | 7 | % | 1.024V, VDD ≥ 2.5V<br>2.048V, VDD ≥ 2.5V<br>4.096V, VDD ≥ 4.75V |
| D004* | SVDD | **VDD Rise Rate** to ensure internal Power-on Reset signal | 0.05 | — | — | V/ms | See **Section 22.3 "Power-on Reset (POR)"** for details. |

* These parameters are characterized but not tested.

**Note 1:** For proper operation, the minimum value of the ADC positive voltage reference must be 1.8V or greater. When selecting the FVR or the VREF+ pin as the source of the ADC positive voltage reference, be aware that the voltage must be 1.8V or greater.

**FIGURE 26-15: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

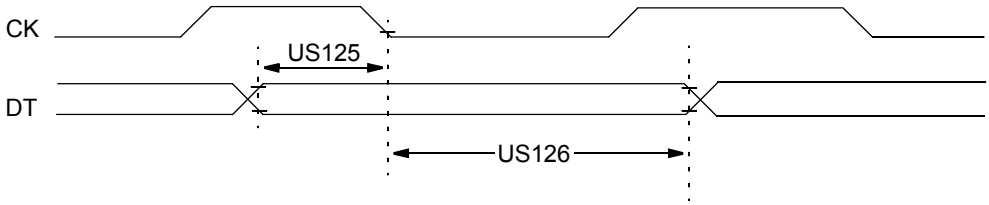



**Note:** Refer to Figure 26-7 for load conditions.

**TABLE 26-24: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Symbol** | **Characteristic** | | **Min.** | **Max.** | **Units** | **Conditions** |
| US120 | TCKH2DTV | SYNC XMIT (Master and Slave) Clock high to data-out valid | 3.0-5.5V | — | 80 | ns | |
| | | | 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 26-16: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**



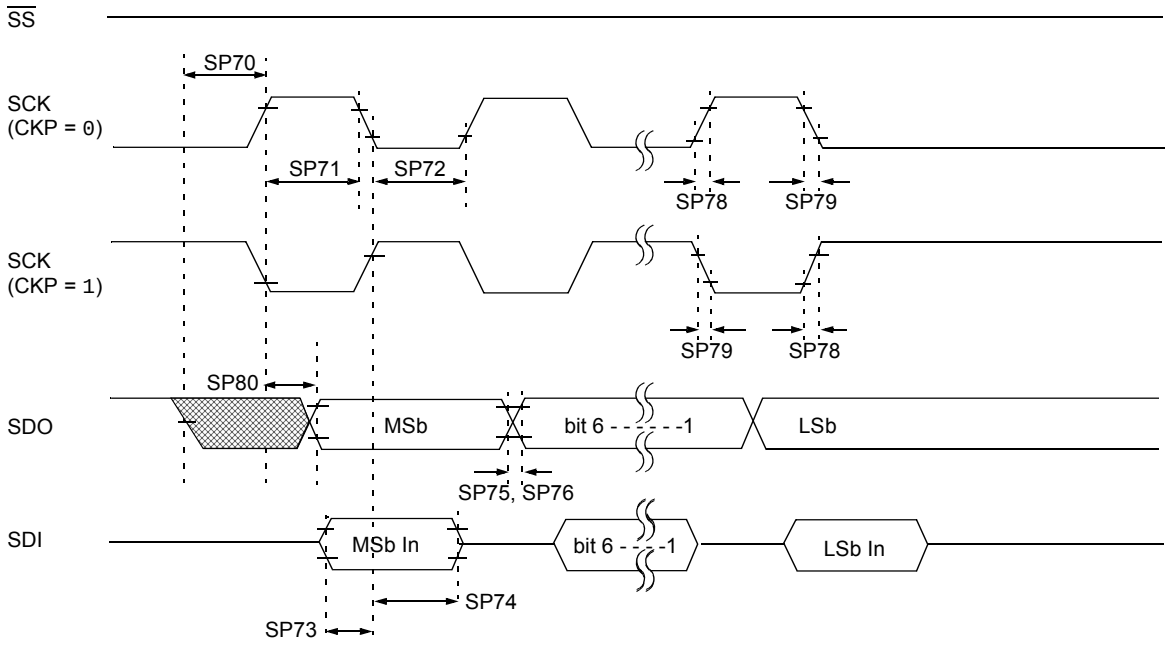

**Note:** Refer to Figure 26-7 for load conditions.

**TABLE 26-25: USART SYNCHRONOUS RECEIVE REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | |
|---|---|---|---|---|---|---|
| **Param. No.** | **Symbol** | **Characteristic** | **Min.** | **Max.** | **Units** | **Conditions** |
| US125 | TDTV2CKL | SYNC RCV (Master and Slave) Data-hold before CK ↓ (DT hold time) | 10 | — | ns | |
| US126 | TCKL2DTL | Data-hold after CK ↓ (DT hold time) | 15 | — | ns | |

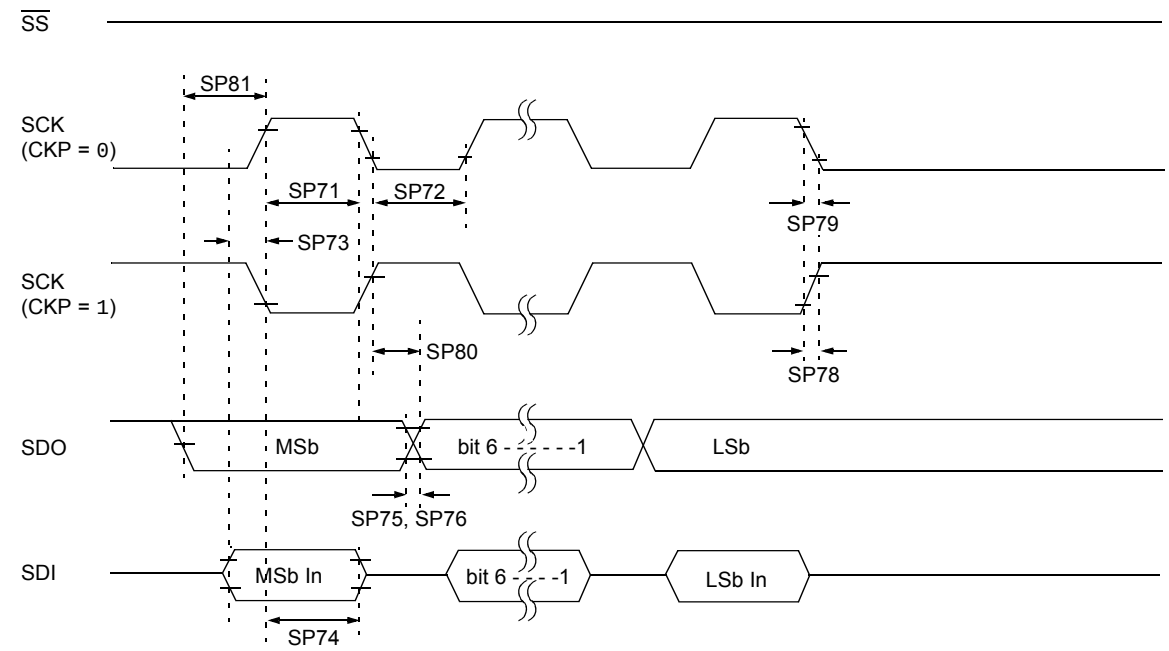

© 2009-2016 Microchip Technology Inc.

**FIGURE 26-17: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)**



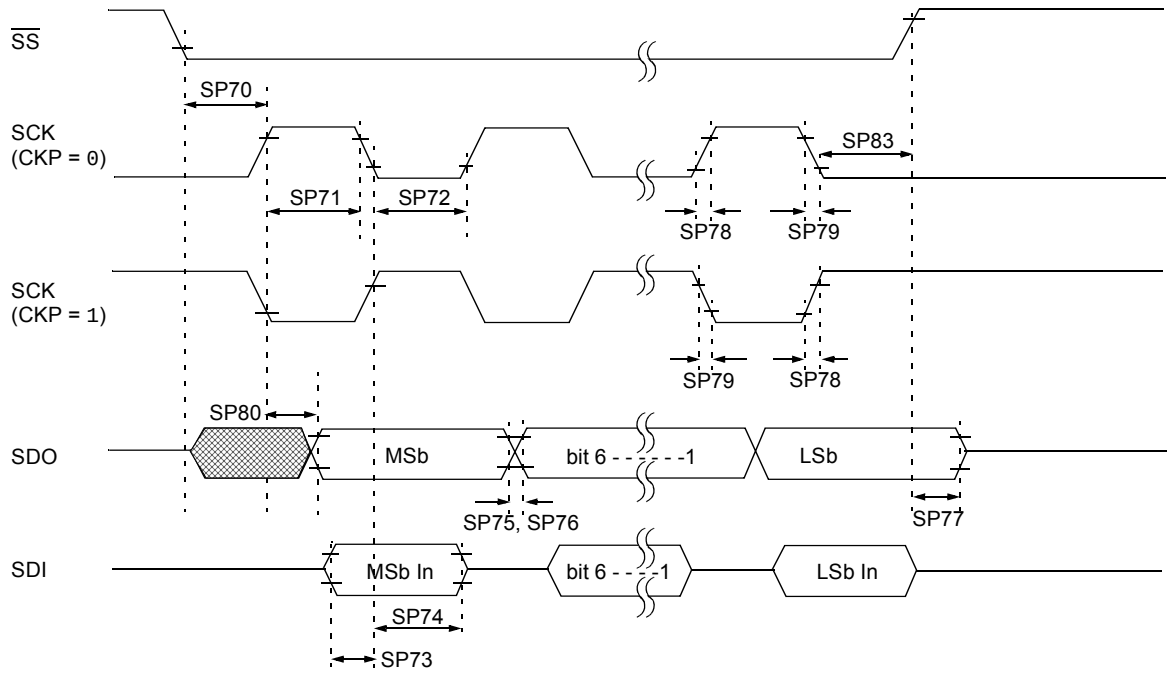

**Note:** Refer to Figure 26-7 for load conditions.

**FIGURE 26-18: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)**



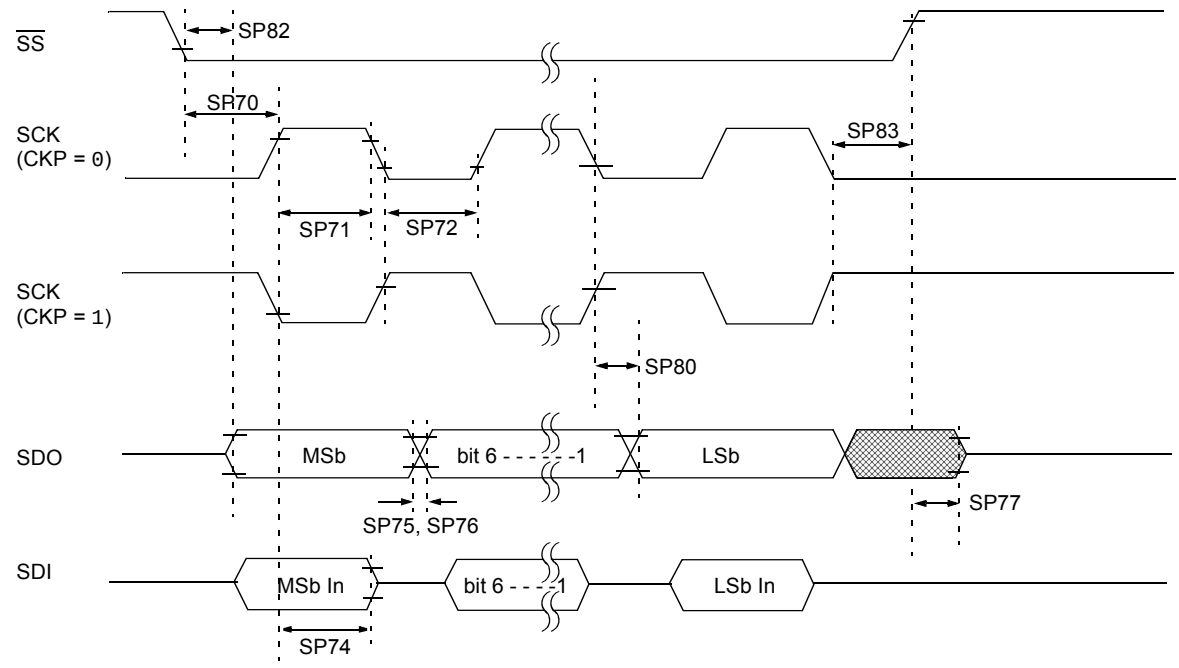

**Note:** Refer to Figure 26-7 for load conditions.

**FIGURE 26-19: SPI SLAVE MODE TIMING (CKE = 0)**




**Note:** Refer to Figure 26-7 for load conditions.

**FIGURE 26-20: SPI SLAVE MODE TIMING (CKE = 1)**




**Note:** Refer to Figure 26-7 for load conditions.


© 2009-2016 Microchip Technology Inc.

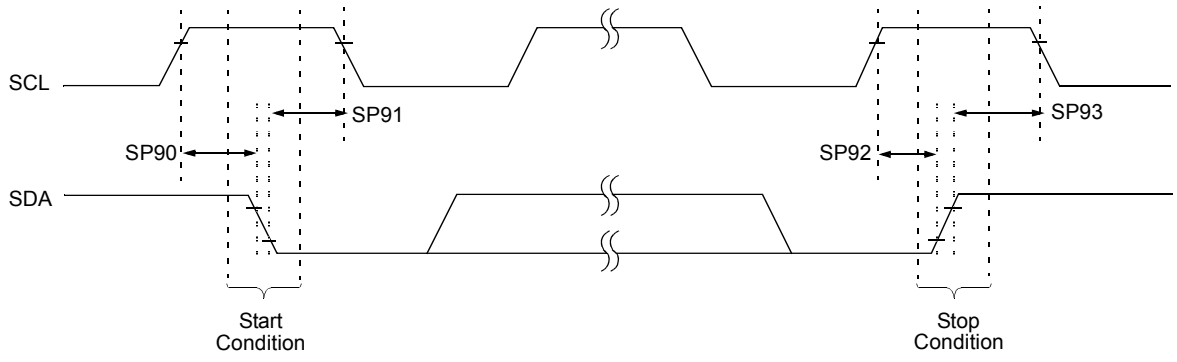
**TABLE 26-26: SPI MODE REQUIREMENTS**

| 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 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**FIGURE 26-21: $I^2C$ BUS START/STOP BITS TIMING**



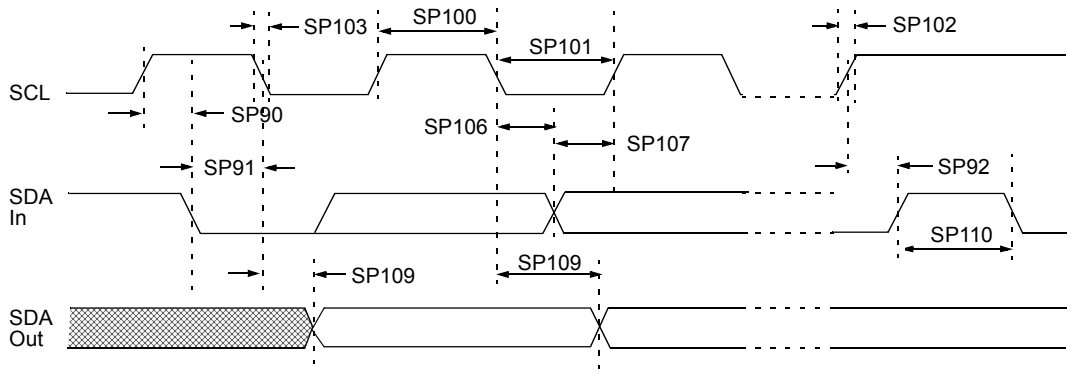

**Note**: Refer to Figure 26-7 for load conditions.

**TABLE 26-27: I²C BUS START/STOP BITS REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min. | Typ. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| SP90* | TSU:STA | Start condition Setup time | 100 kHz mode | 4700 | — | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 600 | — | — | | |
| SP91* | THD:STA | Start condition Hold time | 100 kHz mode | 4000 | — | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 600 | — | — | | |
| SP92* | TSU:STO | Stop condition Setup time | 100 kHz mode | 4700 | — | — | ns | |
| | | | 400 kHz mode | 600 | — | — | | |
| SP93 | THD:STO | Stop condition Hold time | 100 kHz mode | 4000 | — | — | ns | |
| | | | 400 kHz mode | 600 | — | — | | |

* These parameters are characterized but not tested.

**FIGURE 26-22: I²C BUS DATA TIMING**




**Note:** Refer to Figure 26-7 for load conditions.


© 2009-2016 Microchip Technology Inc.

**TABLE 26-28: I²C 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) I²C bus device can be used in a Standard mode (100 kHz) I²C bus system, but the requirement TSU:DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I²C bus specification), before the SCL line is released.

## 27.0 DC AND AC CHARACTERISTICS GRAPHS AND CHARTS

The graphs and tables provided in this section are for **design guidance** and are **not tested**.

In some graphs or tables, the data presented are **outside specified operating range** (i.e., outside specified $V_{DD}$ range). This is for **information only** and devices are ensured to operate properly only within the specified range.

| | |
|---|---|
| **Note:** | The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range. |

**"Typical" represents the mean of the distribution at 25°C. "MAXIMUM", "Max.", "MINIMUM" or "Min." represents (mean + 3σ) or (mean - 3σ) respectively, where σ is a standard deviation, over each temperature range.**


© 2009-2016 Microchip Technology Inc.

**FIGURE 27-1: PIC18LF1XK22 TYPICAL BASE IPD**

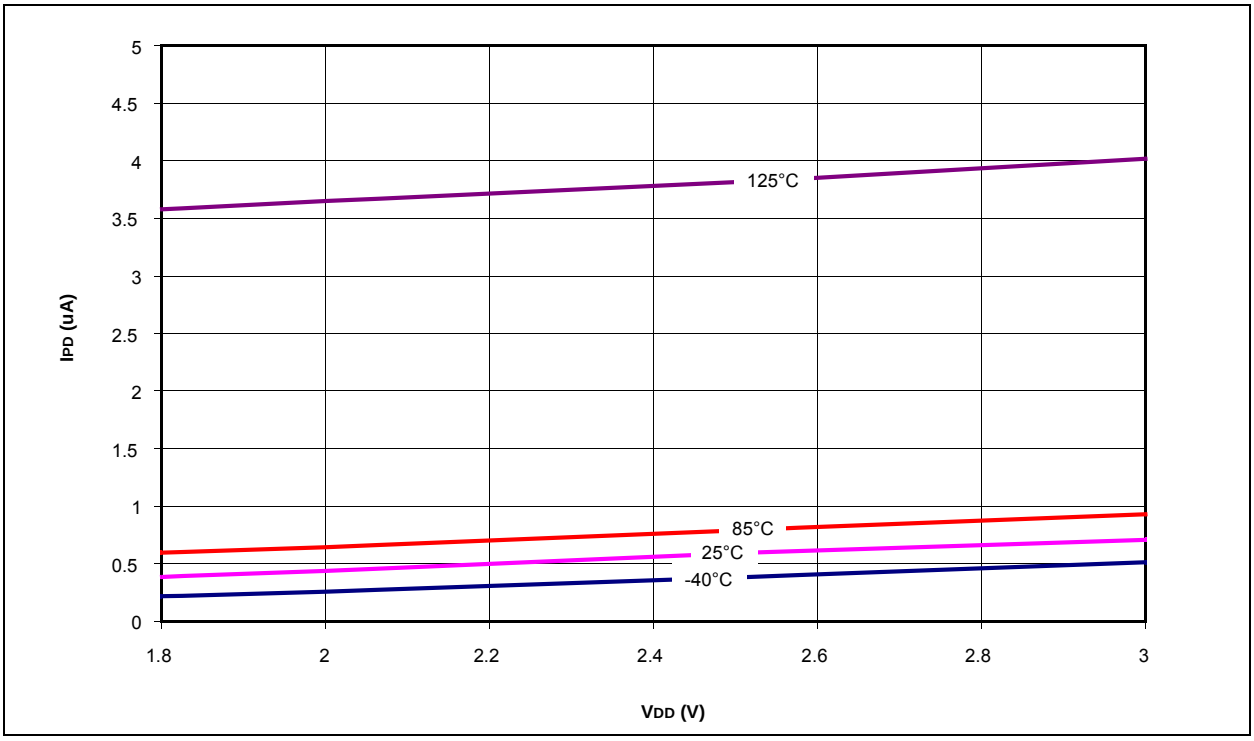



**FIGURE 27-2: PIC18LF1XK22 TYPICAL IPD FOR WATCHDOG TIMER**

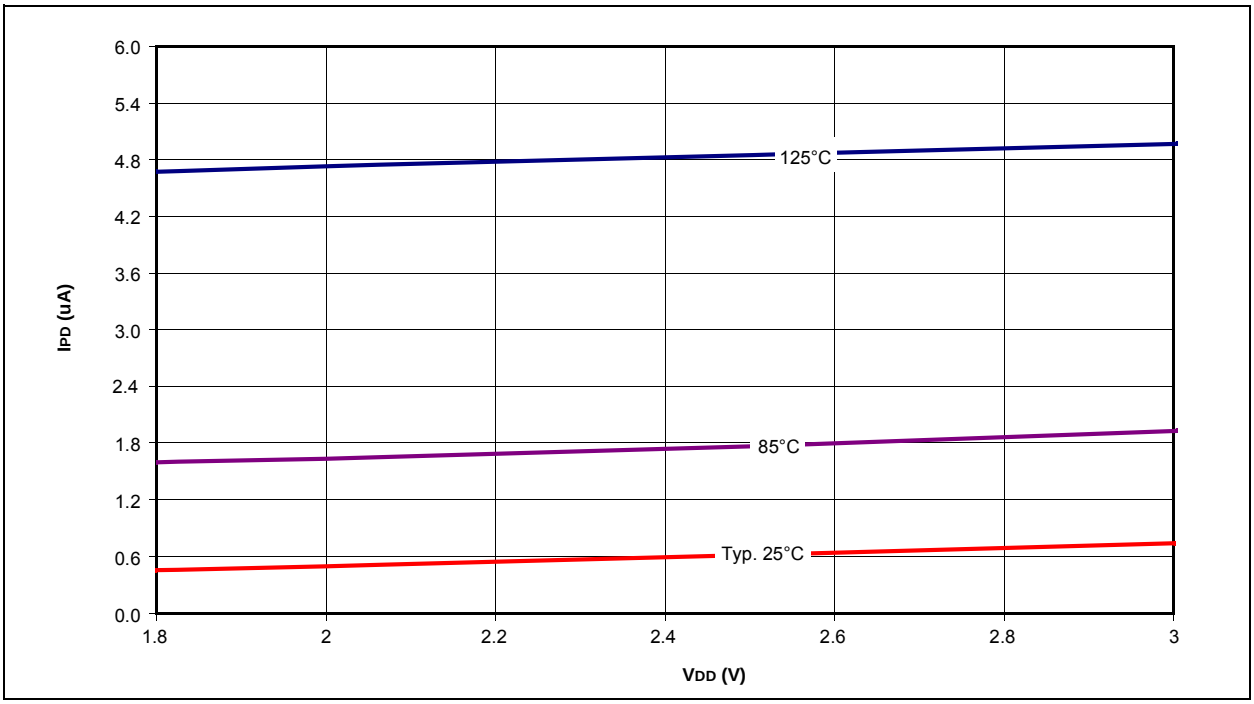


**FIGURE 27-3: PIC18LF1XK22 TYPICAL IPD FOR BROWN-OUT RESET**

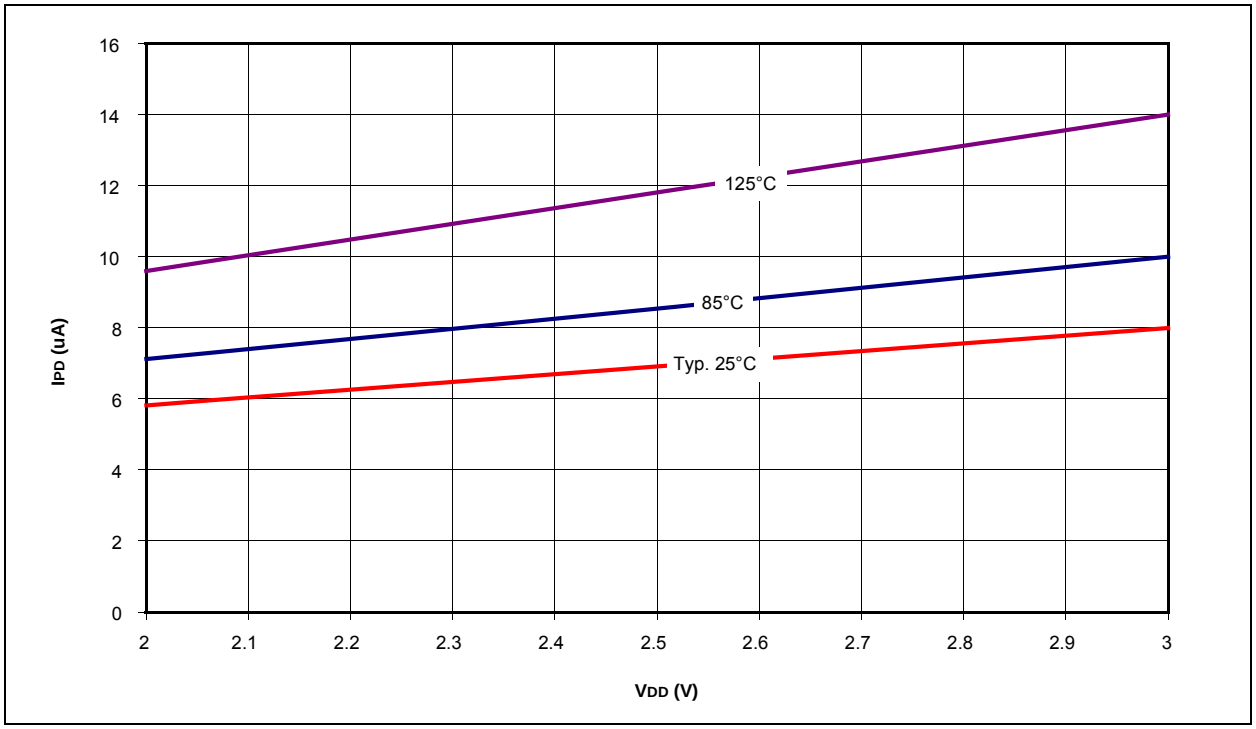
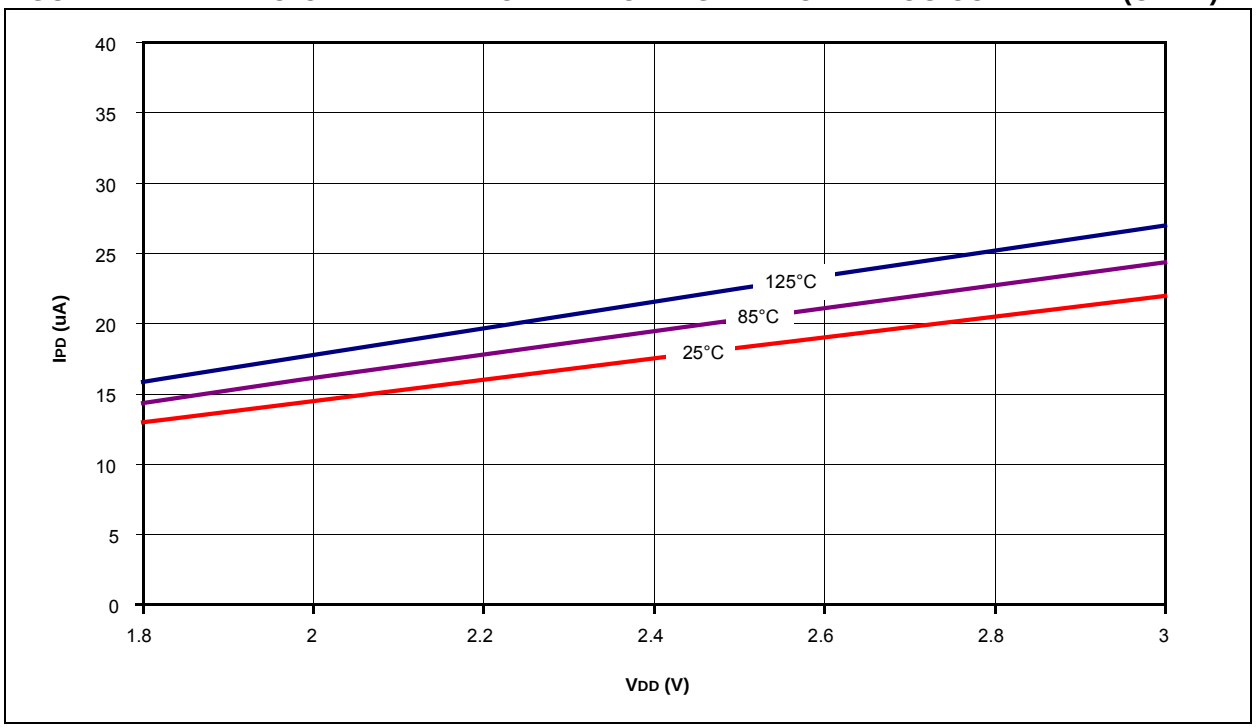



**FIGURE 27-4: PIC18LF1XK22 TYPICAL IPD FOR DIGITAL-TO-ANALOG CONVERTER (CVREF)**





© 2009-2016 Microchip Technology Inc.

**FIGURE 27-5: PIC18LF1XK22 ICOMP – TYPICAL IPD FOR COMPARATOR IN LOW-POWER MODE**

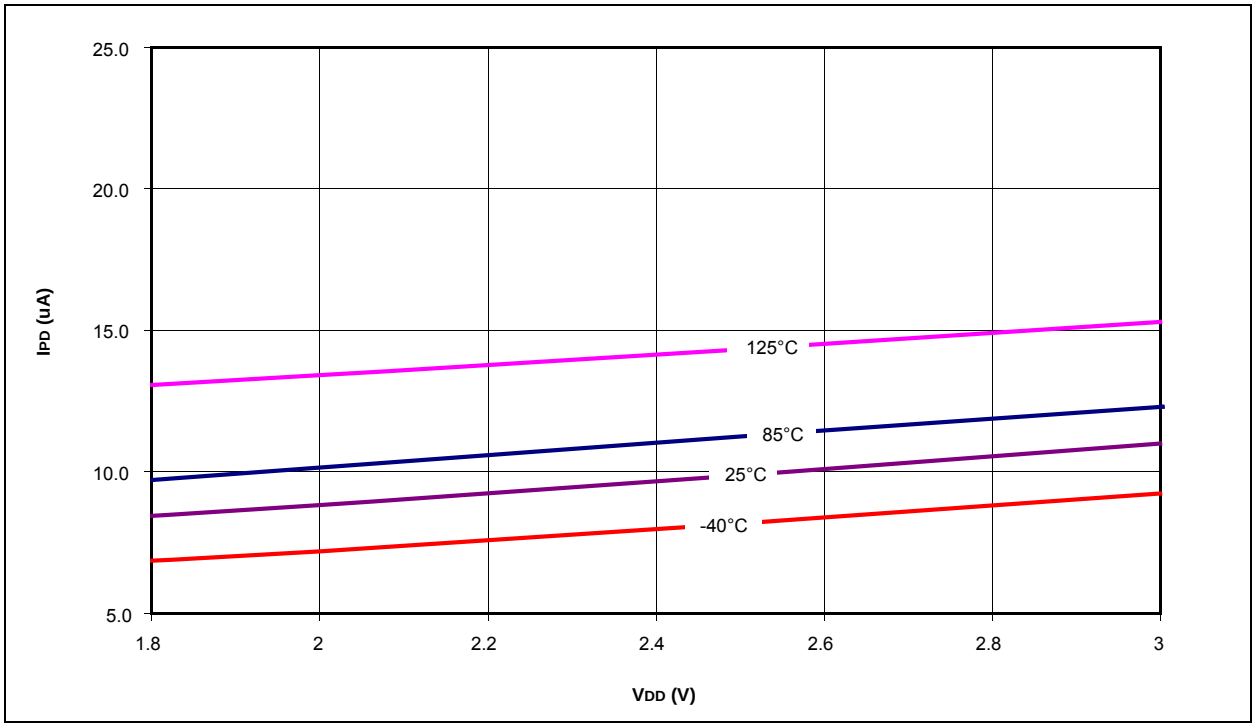



**FIGURE 27-6: PIC18LF1XK22 ICOMP – TYPICAL IPD FOR COMPARATOR IN HIGH-POWER MODE**

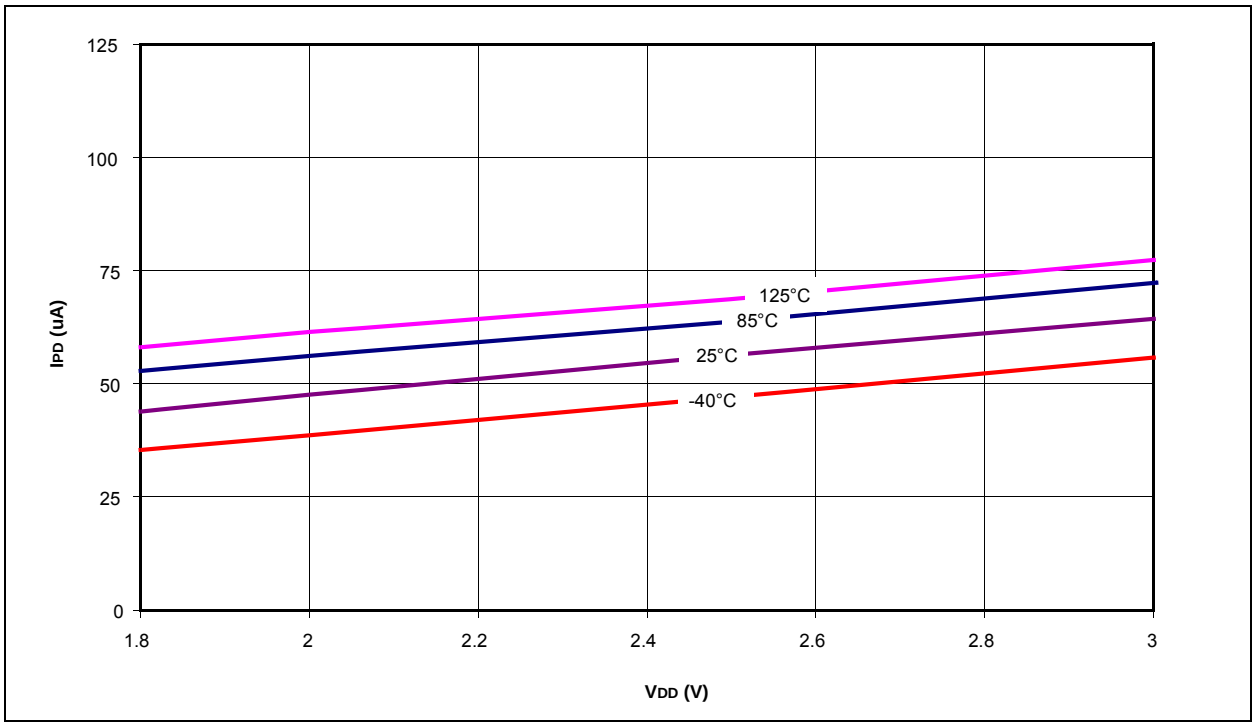


**FIGURE 27-7: PIC18LF1XK22 TYPICAL RC_RUN 31 kHz IDD**

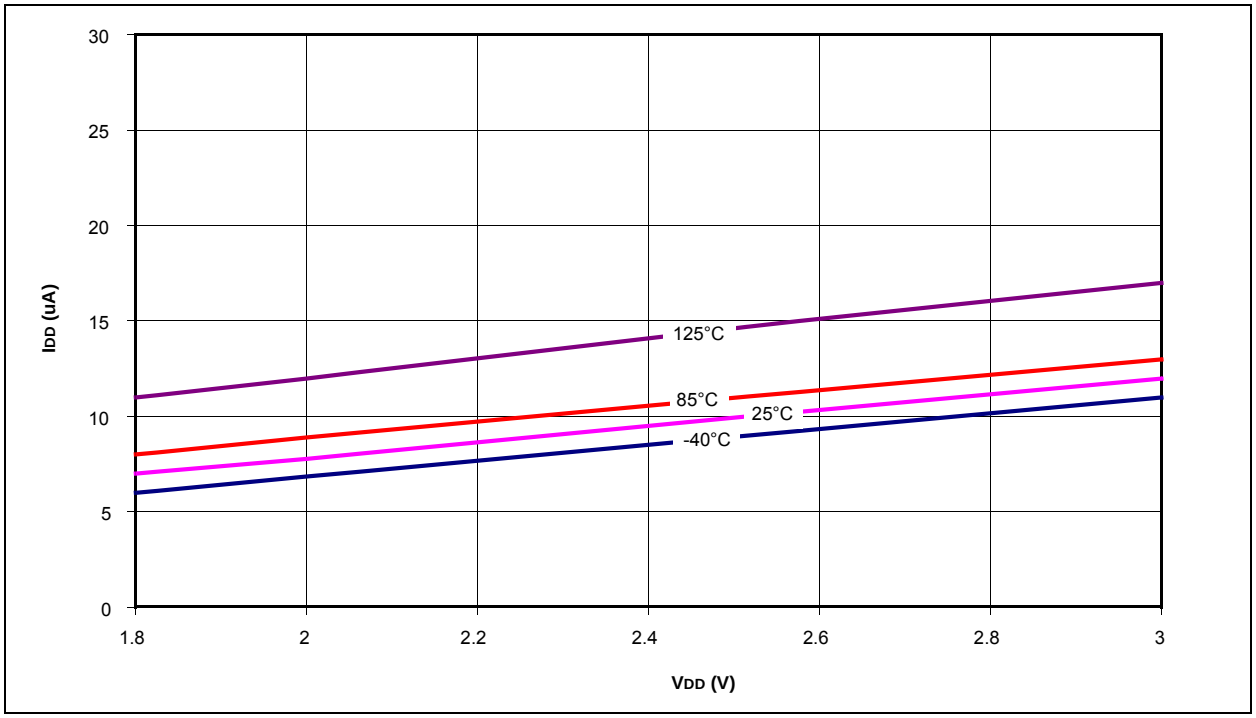
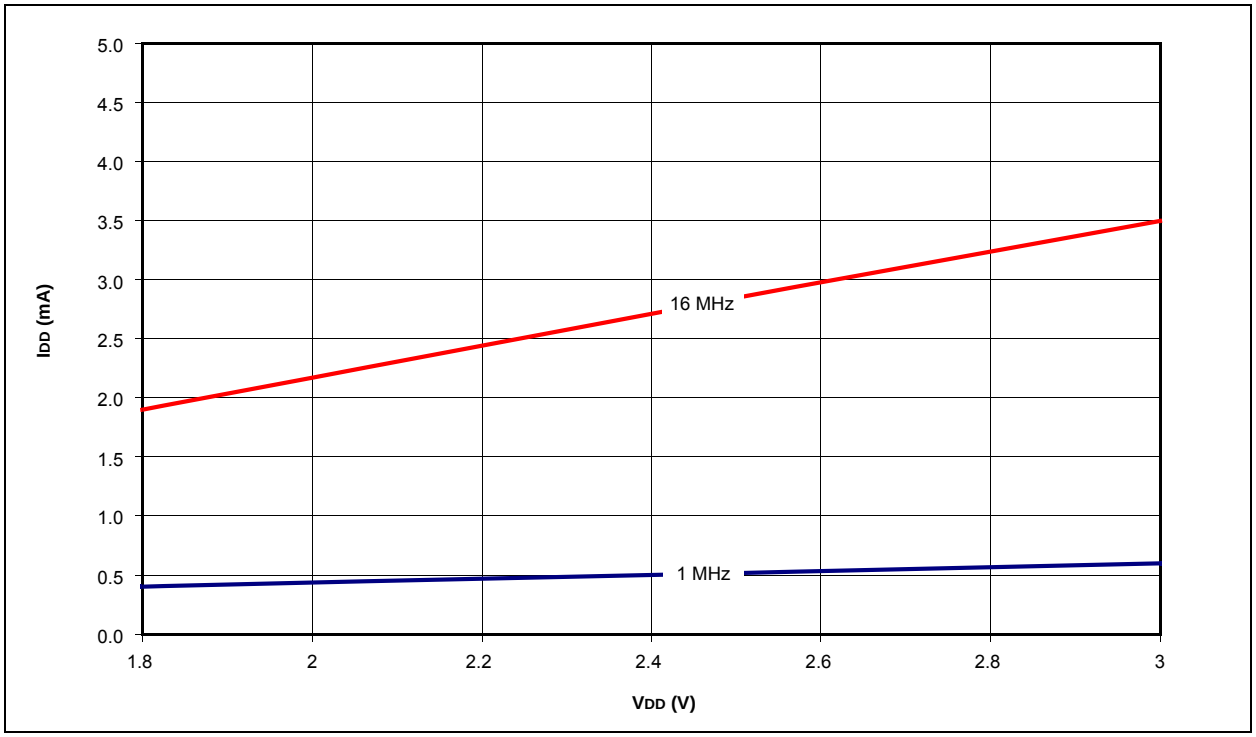



**FIGURE 27-8: PIC18LF1XK22 TYPICAL RC_RUN IDD**





© 2009-2016 Microchip Technology Inc.

**FIGURE 27-9: PIC18LF1XK22 TYPICAL PRI_RUN IDD (EC)**

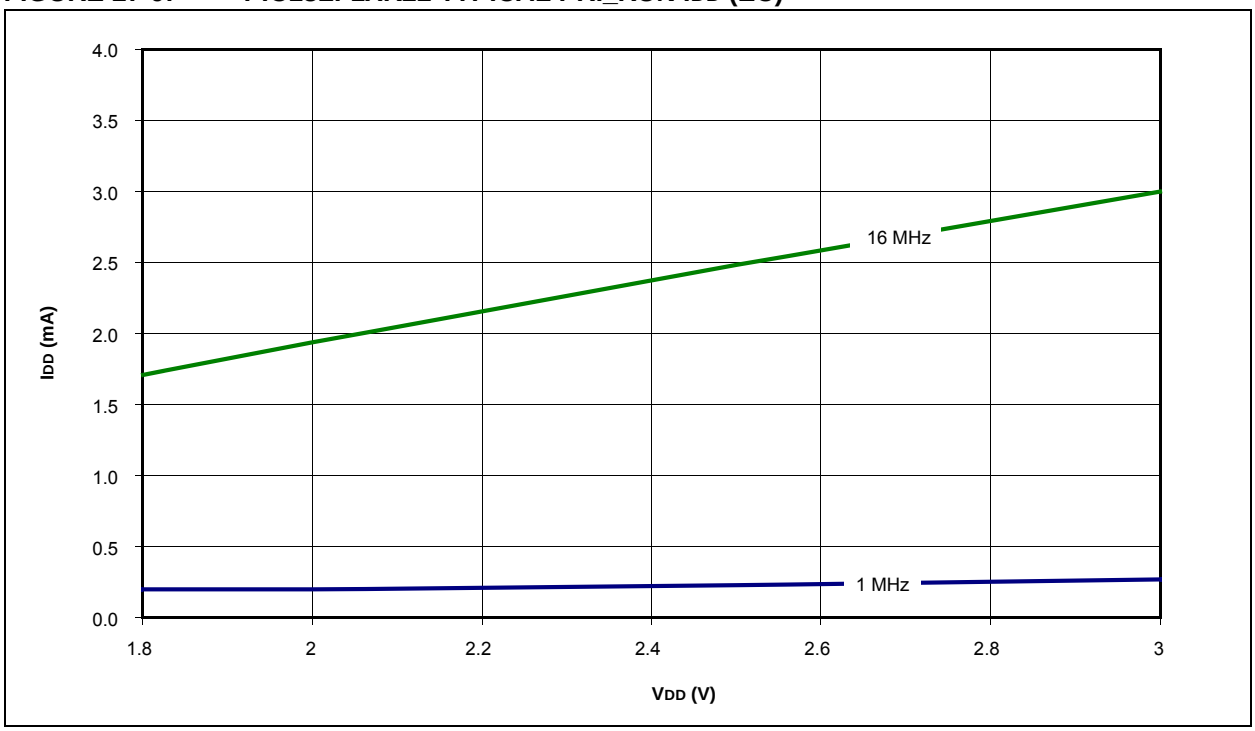



**FIGURE 27-10: PIC18LF1XK22 TYPICAL PRI_RUN IDD (HS + PLL)**

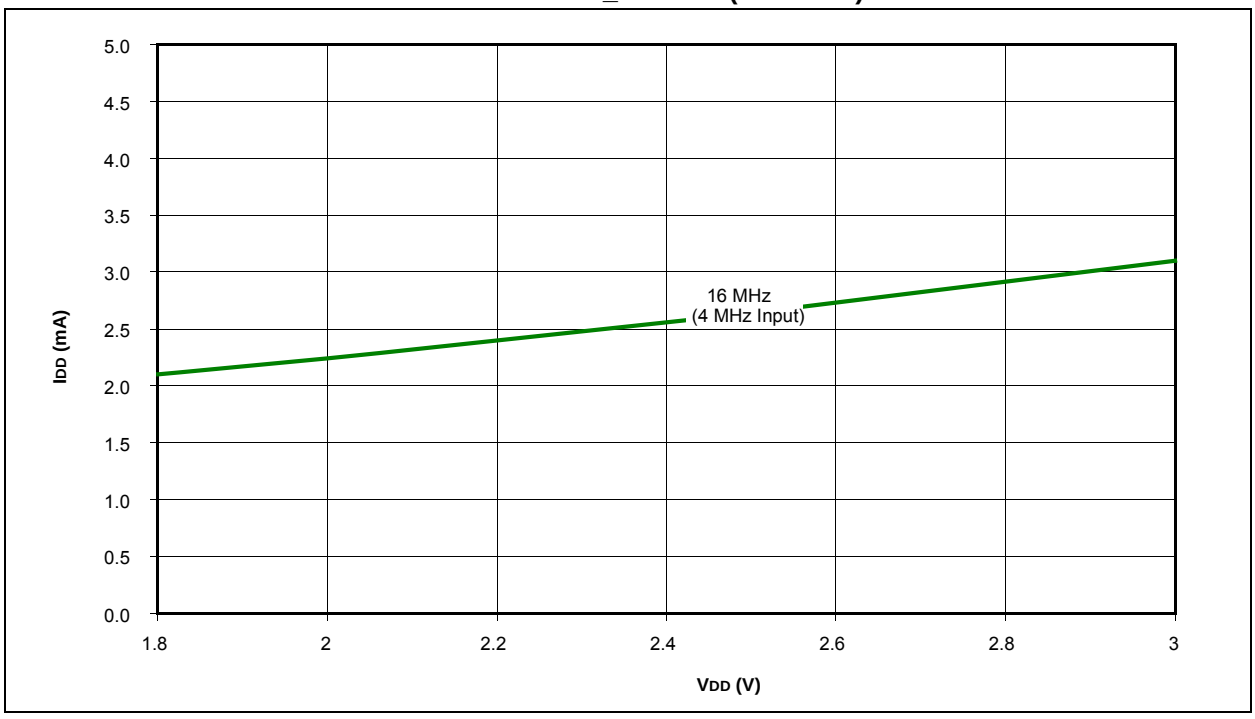


**FIGURE 27-11: MEMLOW TYPICAL BASE IPD**

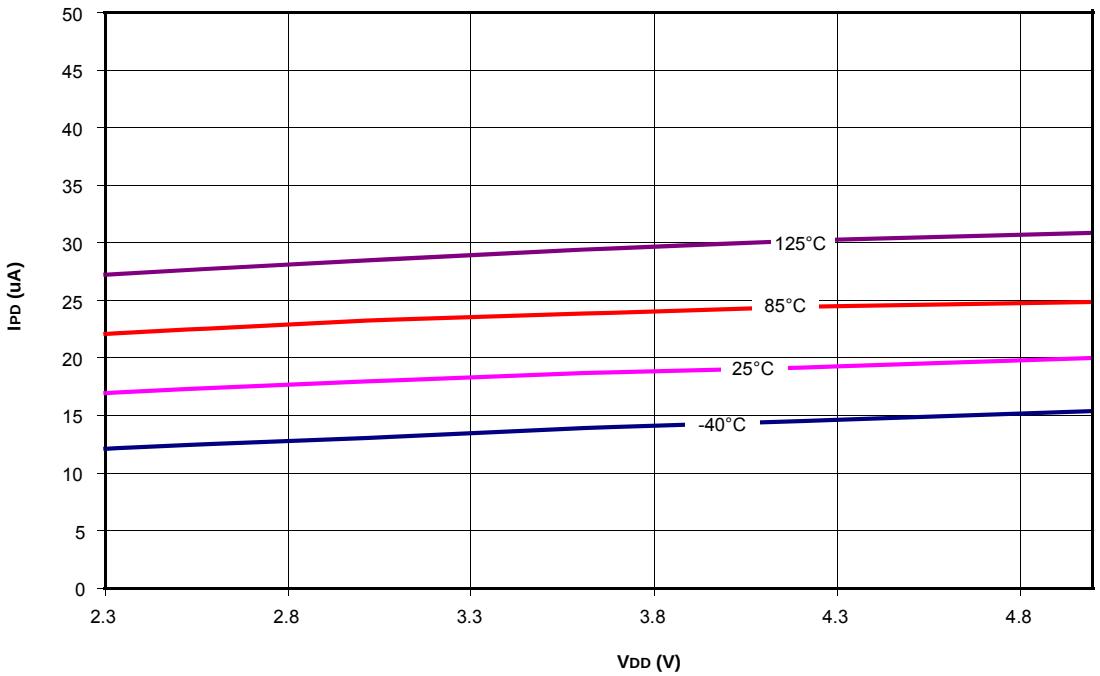
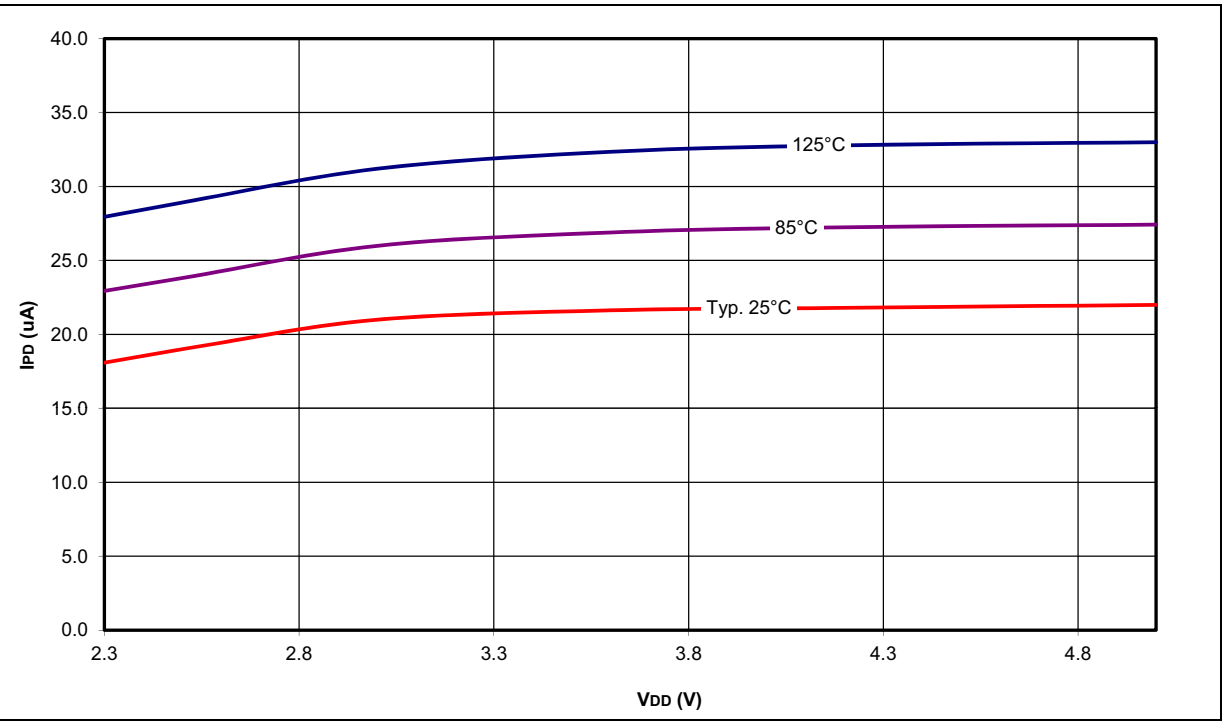



**FIGURE 27-12: MEMLOW TYPICAL IPD FOR WATCHDOG TIMER**





© 2009-2016 Microchip Technology Inc.

**FIGURE 27-13: MEMLOW TYPICAL IPD FOR BROWN-OUT RESET**

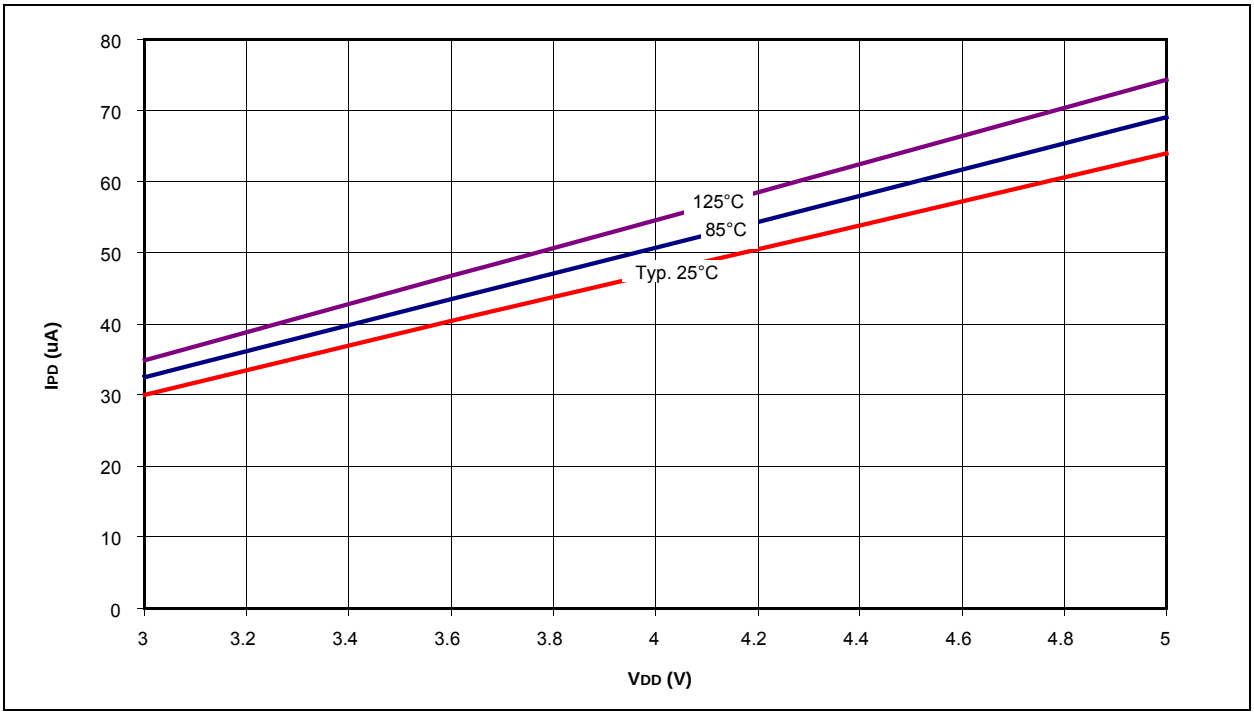



**FIGURE 27-14: MEMLOW TYPICAL IPD FOR DIGITAL-TO-ANALOG CONVERTER (CVREF)**

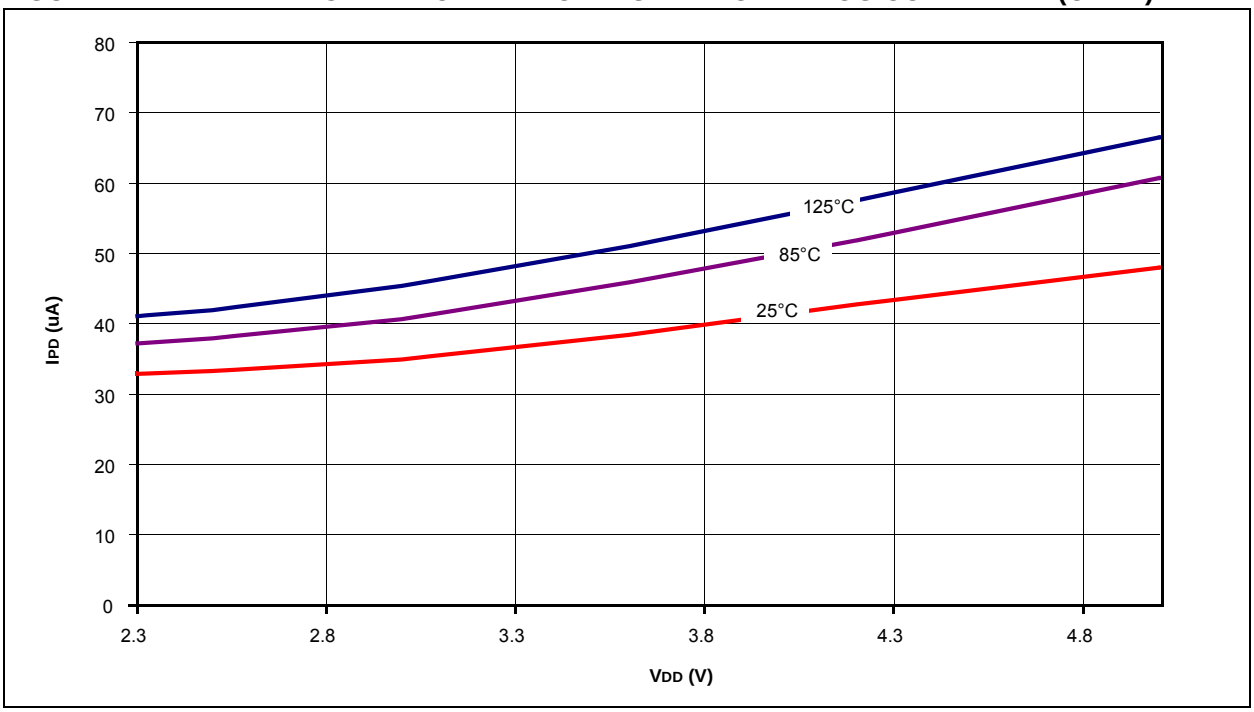


**FIGURE 27-15: MEMLOW ICOMP – TYPICAL IPD FOR COMPARATOR IN LOW-POWER MODE**

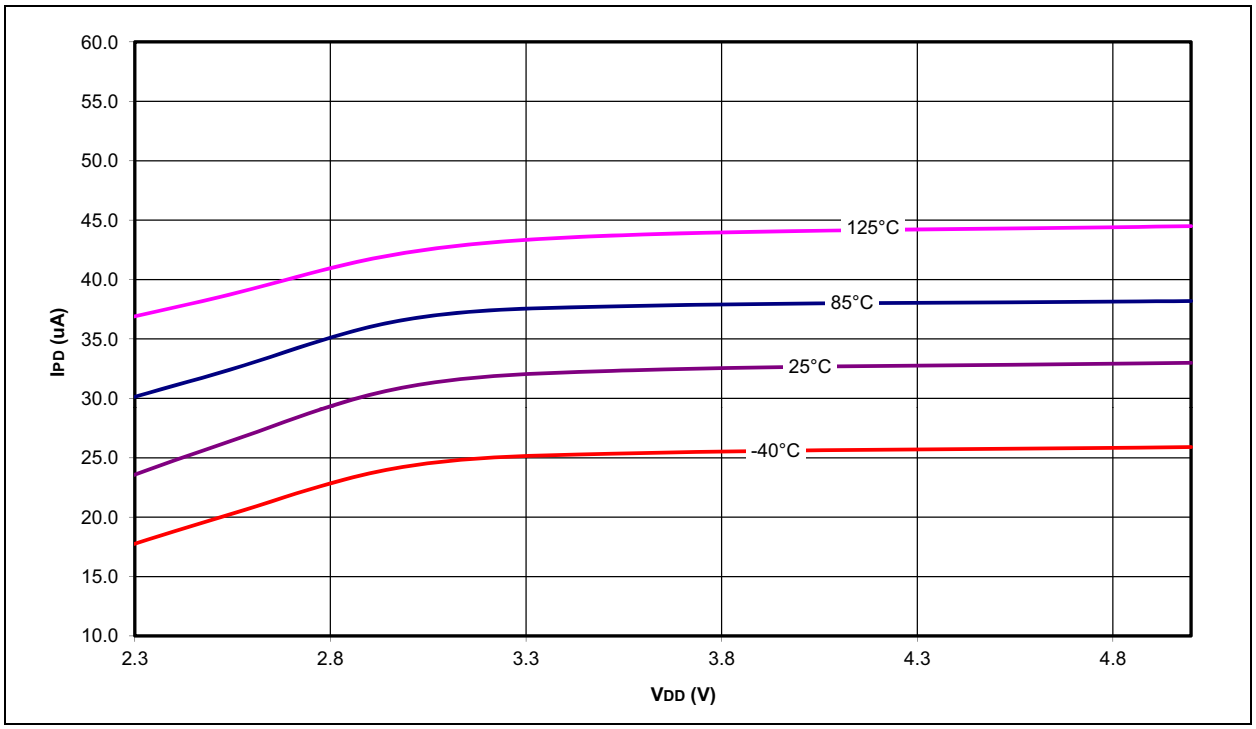
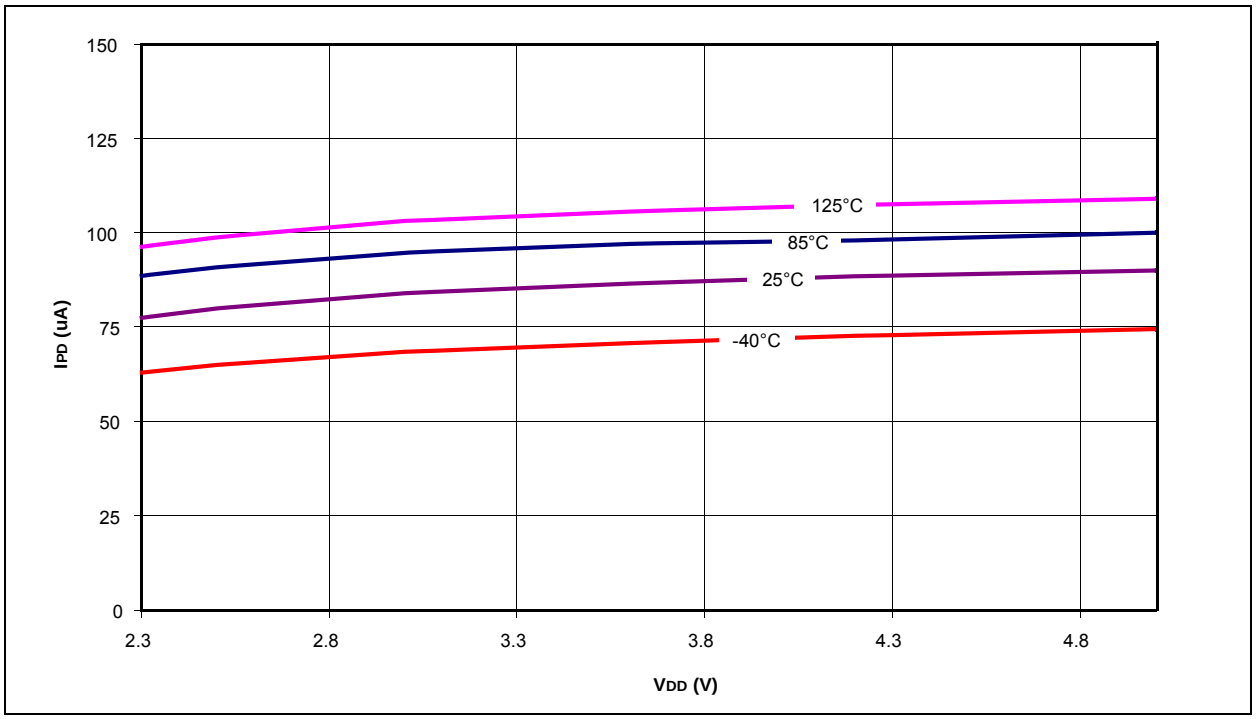



**FIGURE 27-16: MEMLOW ICOMP – TYPICAL IPD FOR COMPARATOR IN HIGH-POWER MODE**





© 2009-2016 Microchip Technology Inc.

**FIGURE 27-17: MEMLOW TYPICAL RC_RUN 31 kHz IDD**

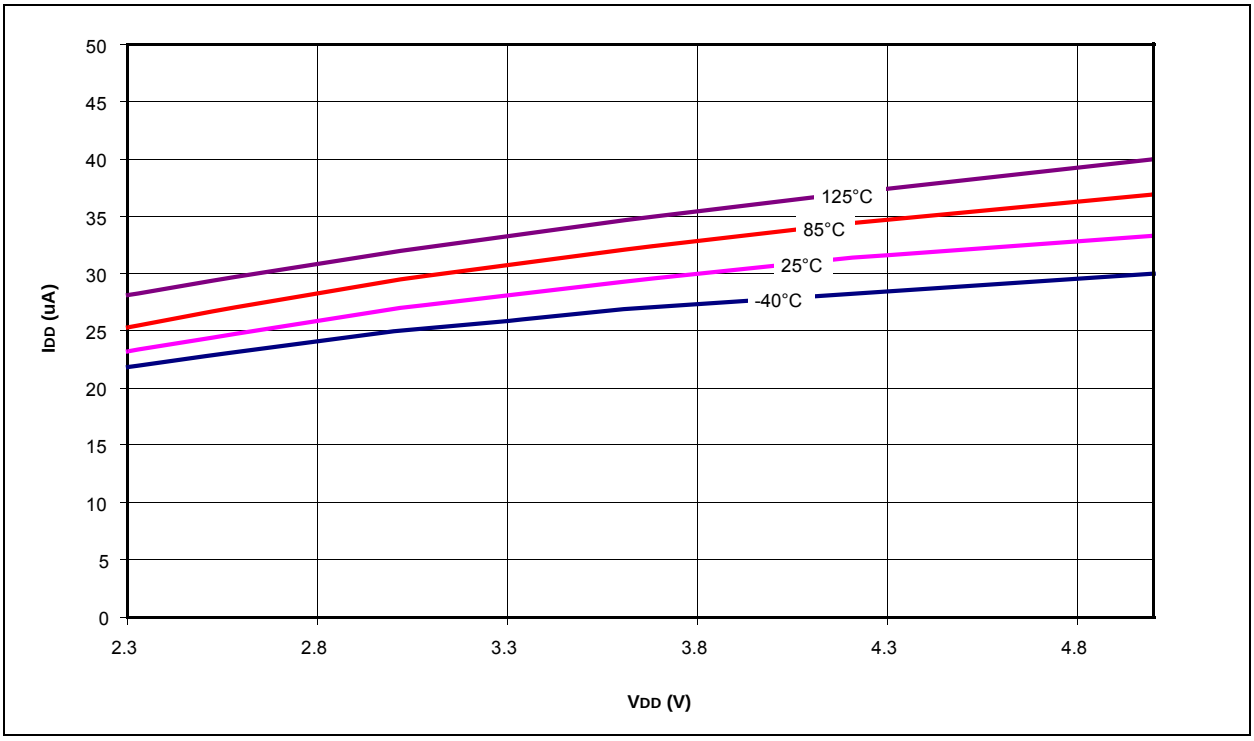



**FIGURE 27-18: MEMLOW TYPICAL RC_RUN IDD**

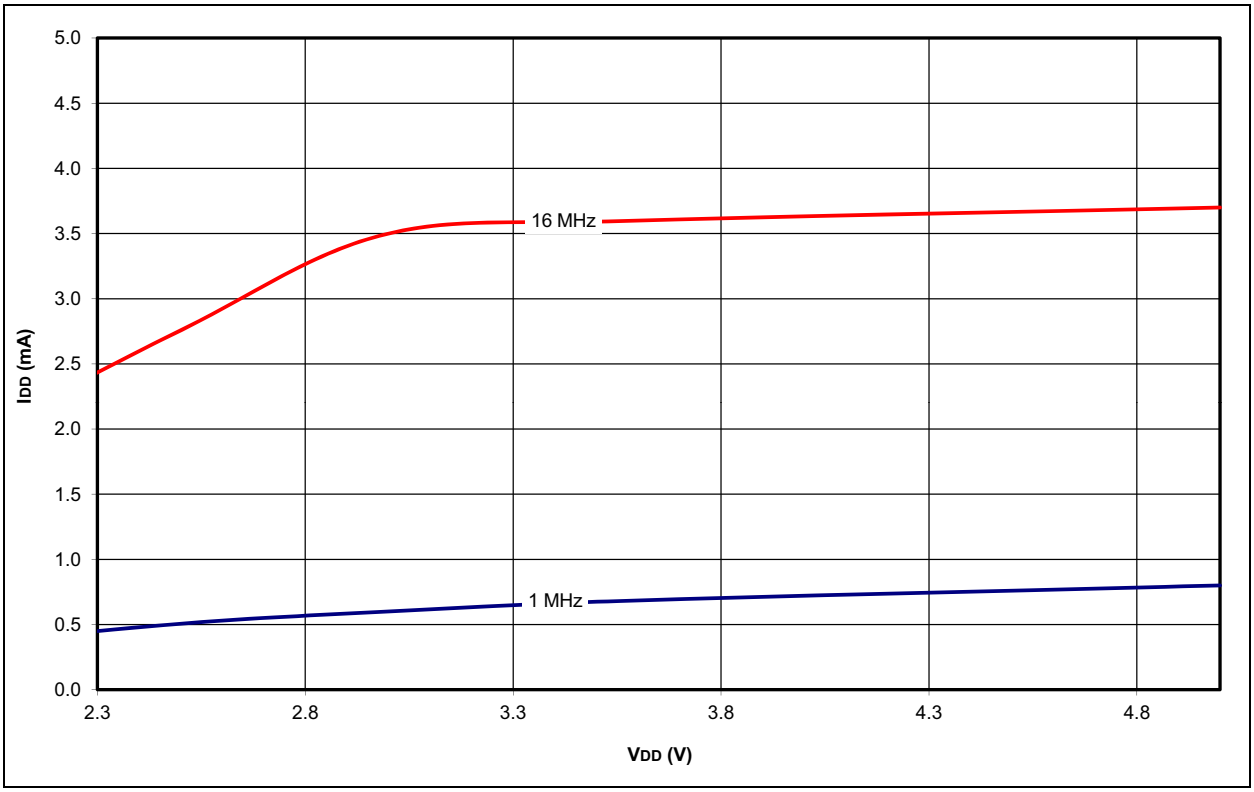


**FIGURE 27-19: MEMLOW TYPICAL PRI_RUN IDD (EC)**

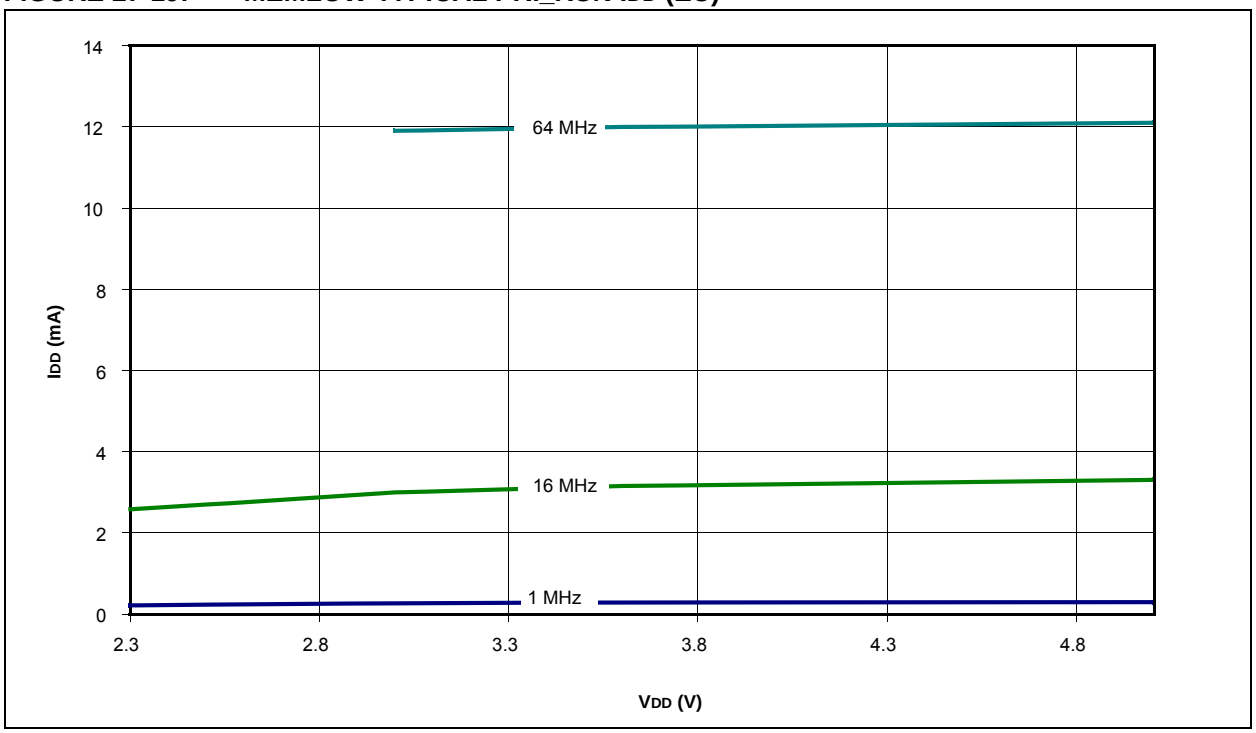
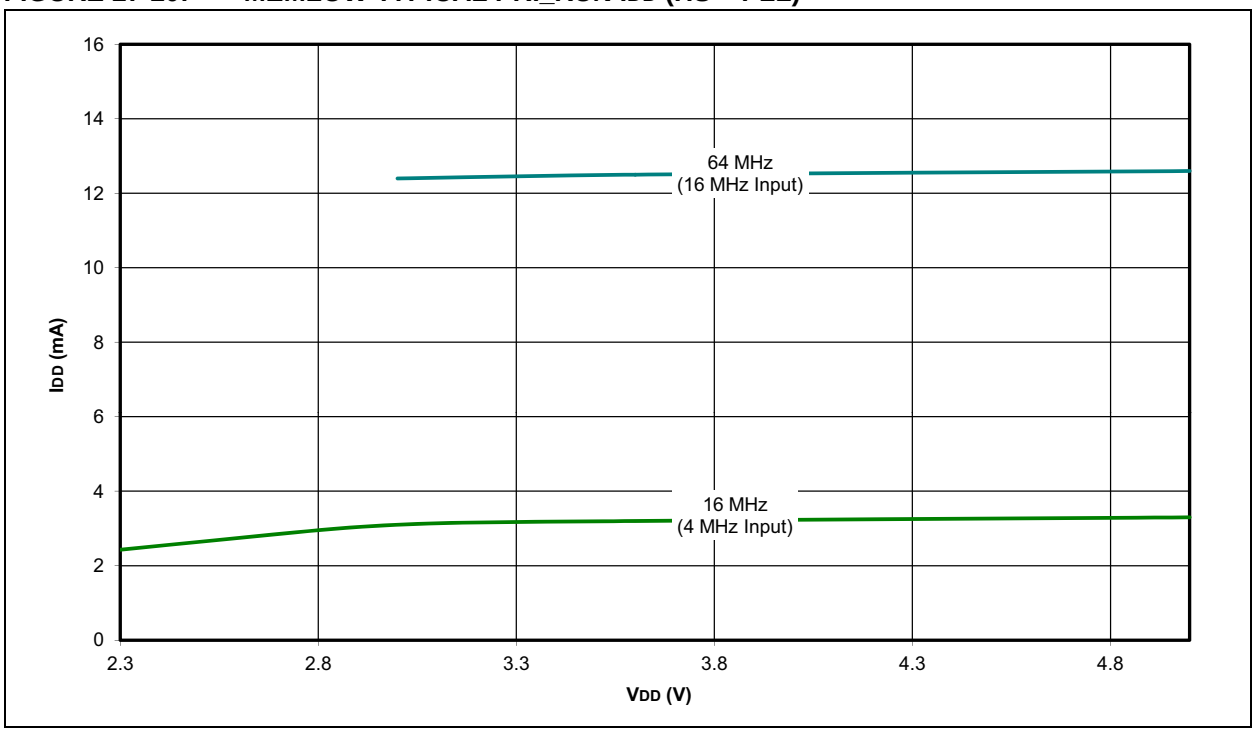



**FIGURE 27-20: MEMLOW TYPICAL PRI_RUN IDD (HS + PLL)**





© 2009-2016 Microchip Technology Inc.

**FIGURE 27-21: PIC18(L)F1XK22 TTL BUFFER TYPICAL $V_{IH}$**

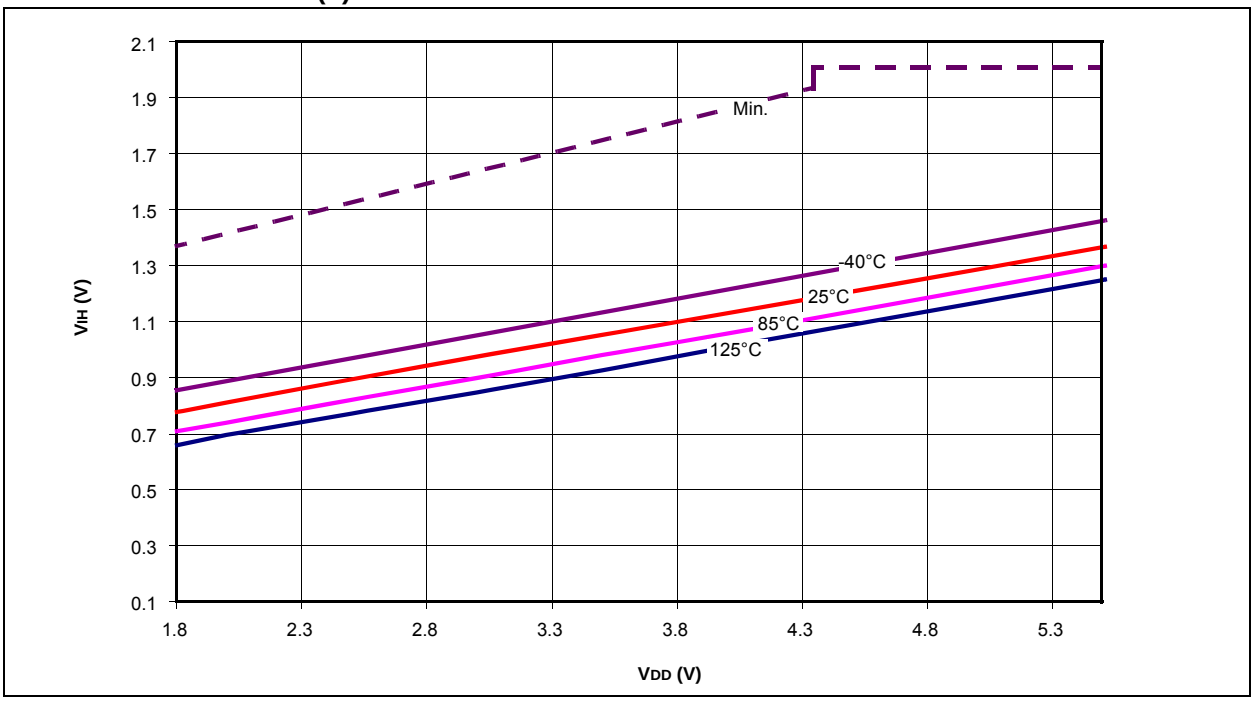



**FIGURE 27-22: PIC18(L)F1XK22 SCHMITT TRIGGER BUFFER TYPICAL $V_{IH}$**

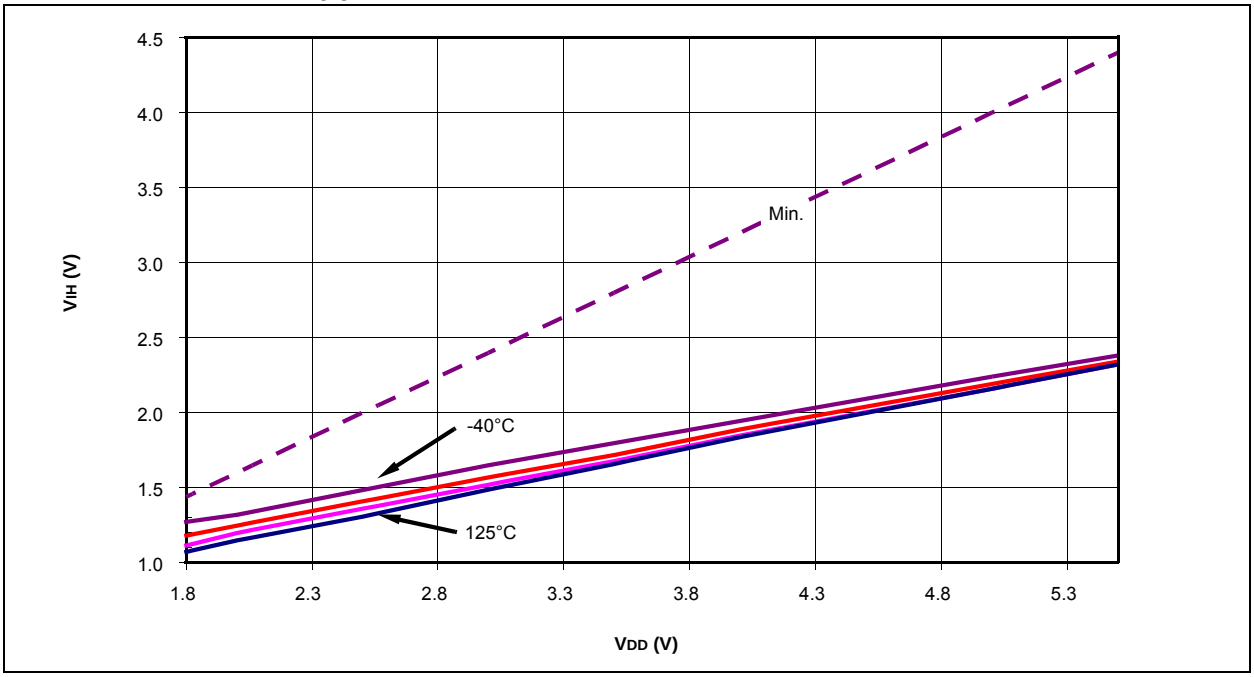


**FIGURE 27-23: PIC18(L)F1XK22 TTL BUFFER TYPICAL VIL**

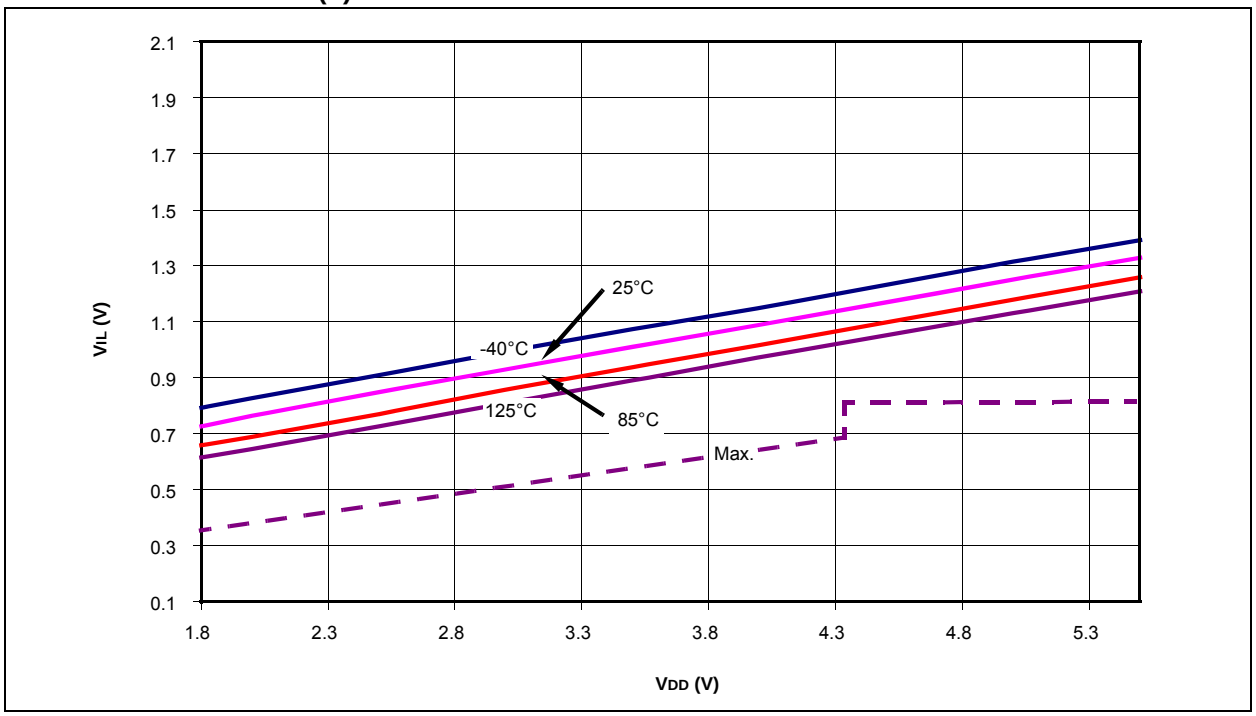
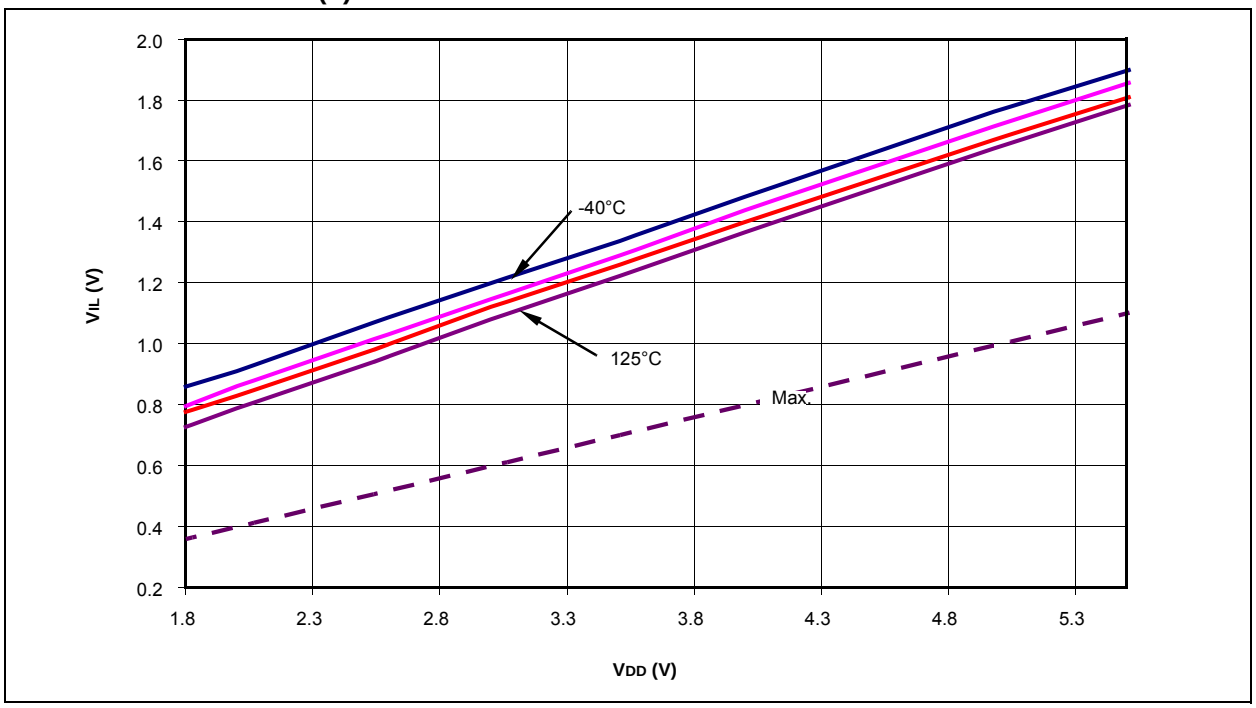



**FIGURE 27-24: PIC18(L)F1XK22 SCHMITT BUFFER TYPICAL VIL**





© 2009-2016 Microchip Technology Inc.

**FIGURE 27-25: MEMLOW TYPICAL LF-INTOSC FREQUENCY (MAX./MIN. = 31.25 kHz ± 15%)**

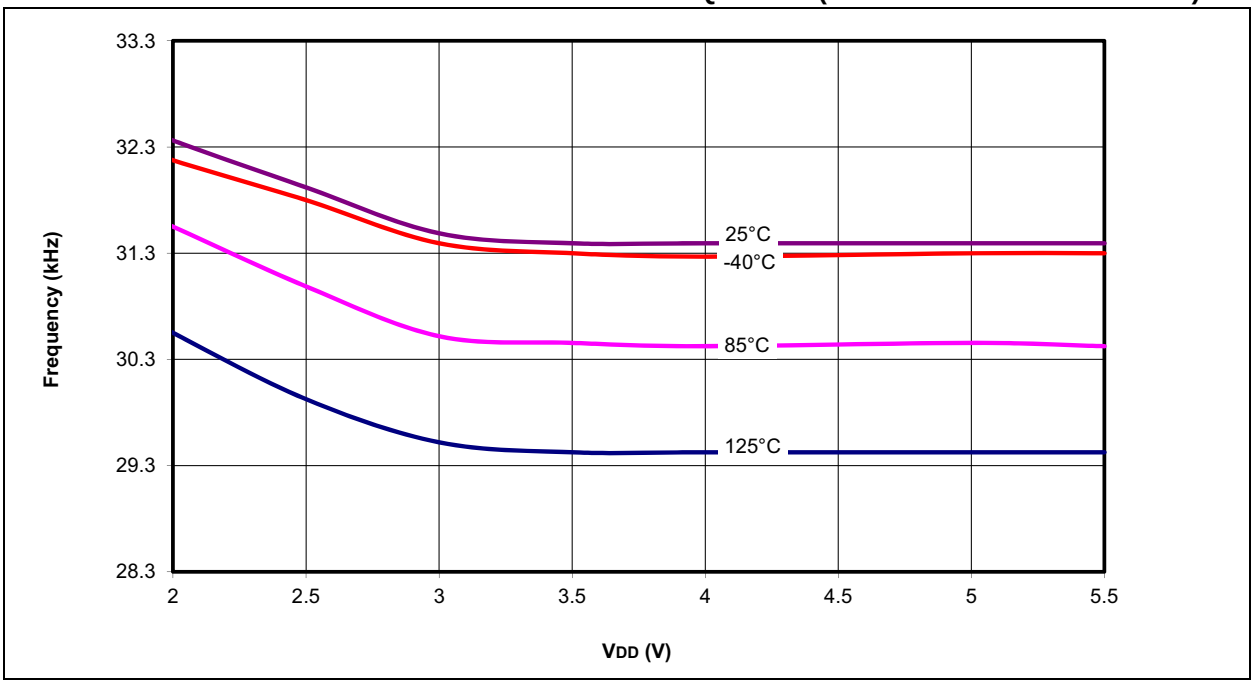



**FIGURE 27-26: MEMLOW TYPICAL LF-INTOSC FREQUENCY (MAX./MIN. = 31.25 kHz ± 15%)**

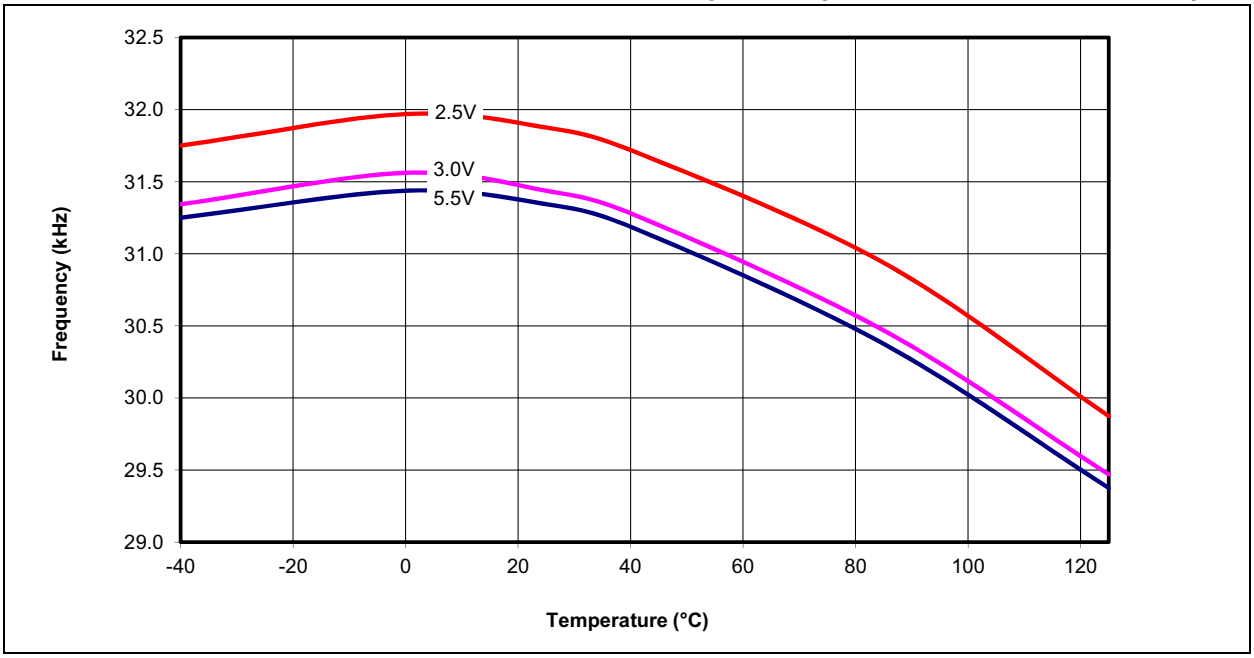


**FIGURE 27-27: PIC18LF1XK22 TYPICAL LF-INTOSC FREQUENCY (MAX./MIN. = 31.25 kHz ± 15%)**

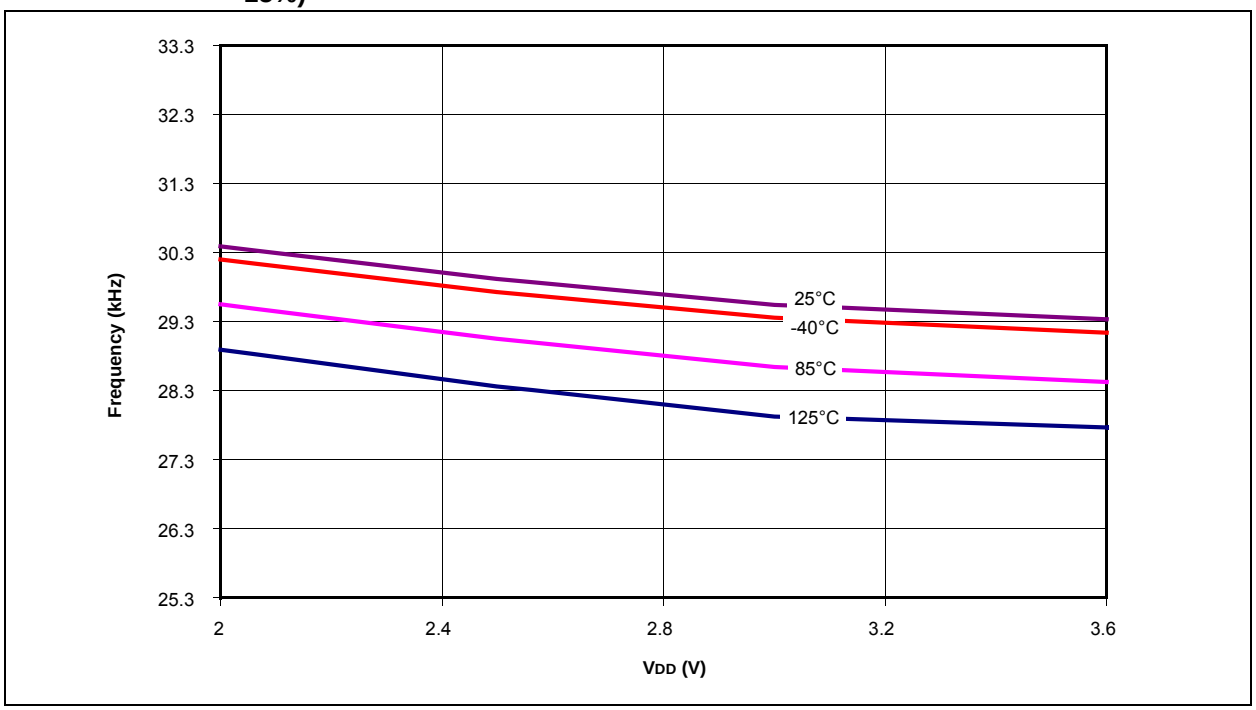
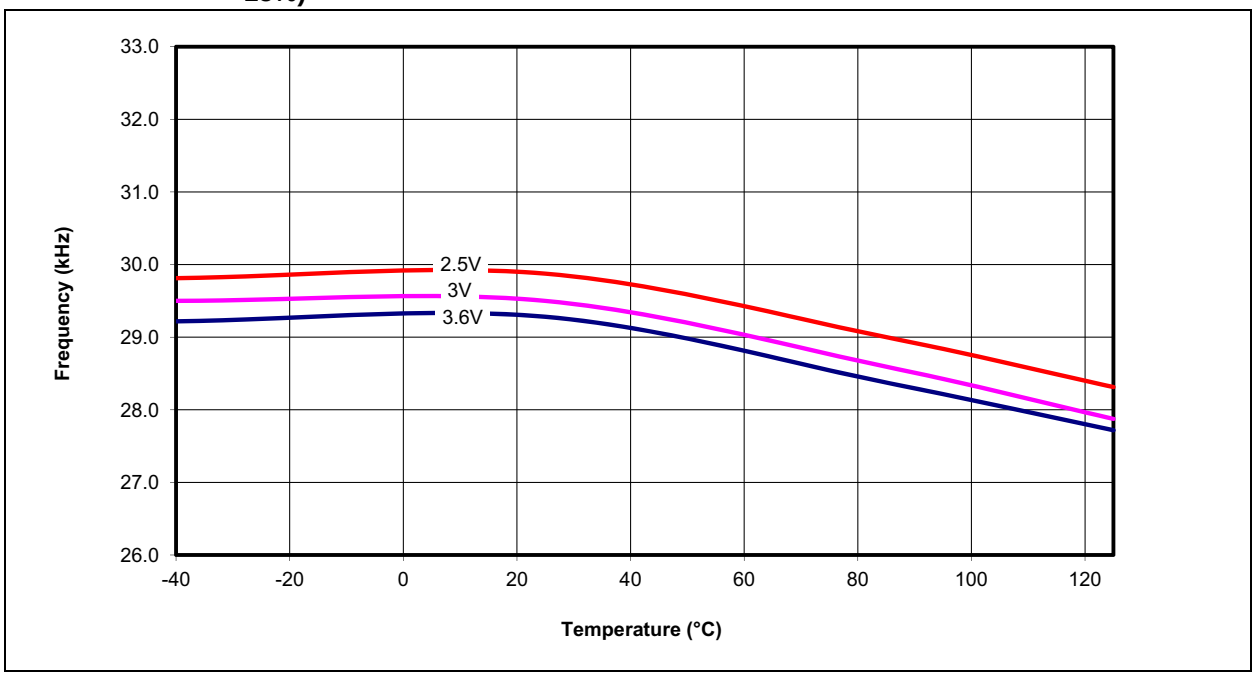



**FIGURE 27-28: PIC18LF1XK22 TYPICAL LF-INTOSC FREQUENCY (MAX./MIN. = 31.25 kHz ± 15%)**





© 2009-2016 Microchip Technology Inc.

**FIGURE 27-29: MEMLOW TYPICAL VOH VS. IOH**

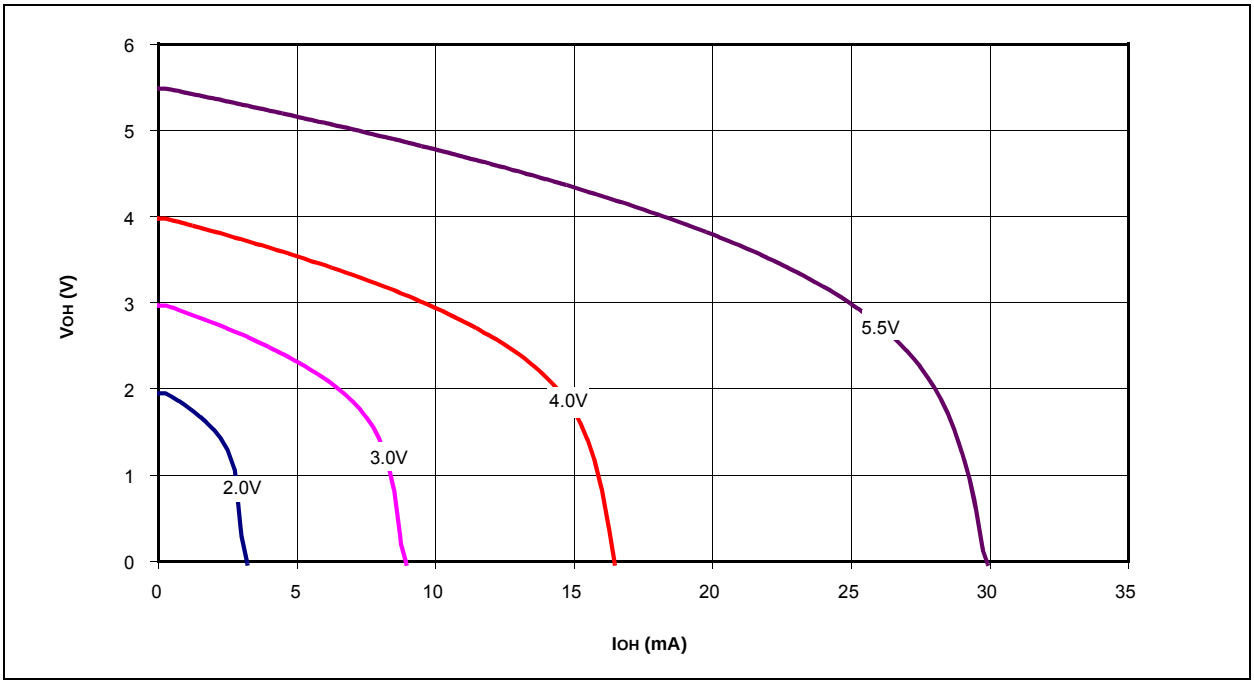
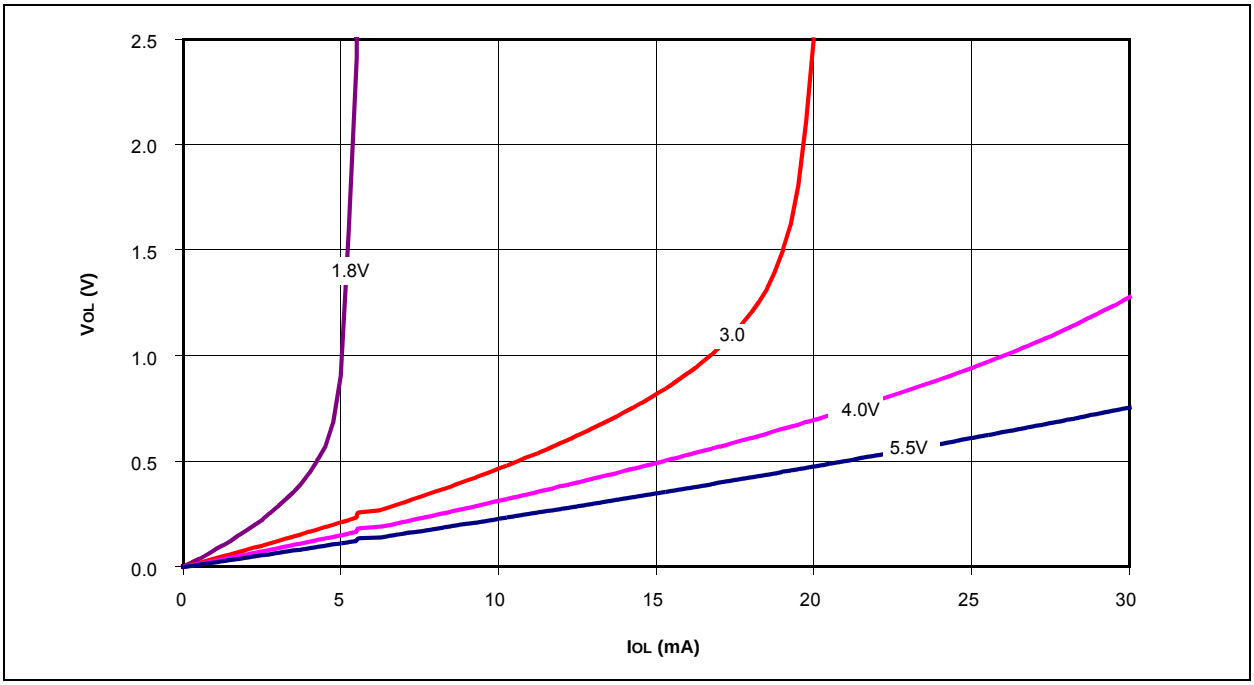


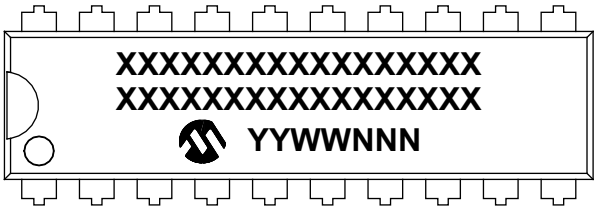

**FIGURE 27-30: MEMLOW TYPICAL VOL VS. IOL**



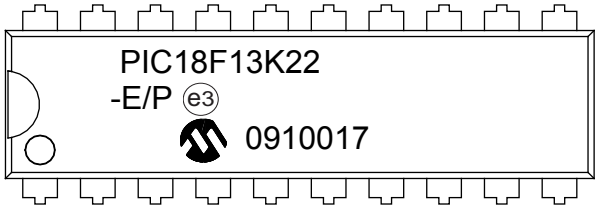
## 28.0 PACKAGING INFORMATION

### 28.1 Package Marking Information

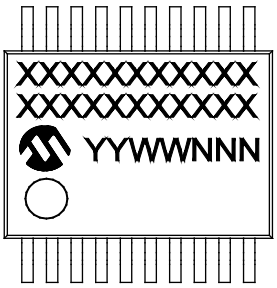
20-Lead PDIP (300 mil)



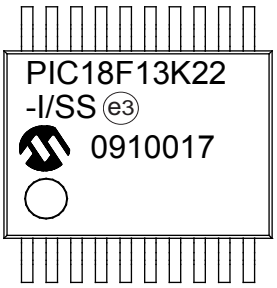

Example




20-Lead SSOP (5.30 mm)




Example



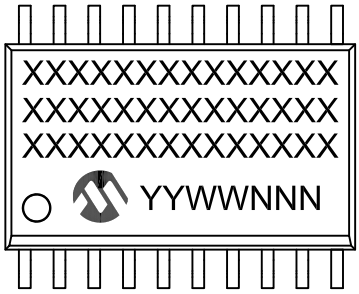

| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week ‘01’) |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC® designator for Matte Tin (Sn) |
| | * | This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package. |

**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

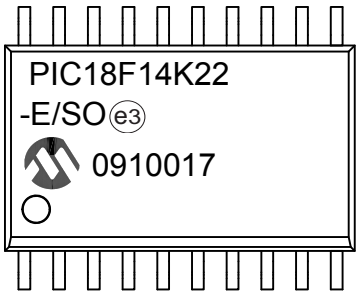

© 2009-2016 Microchip Technology Inc.

**Package Marking Information (Continued)**

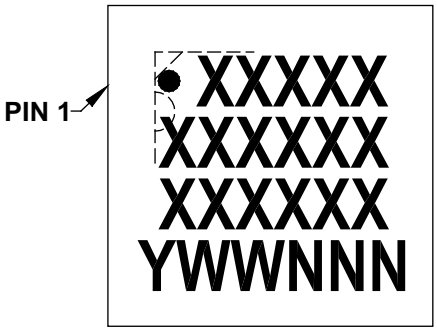
20-Lead SOIC (7.50 mm)



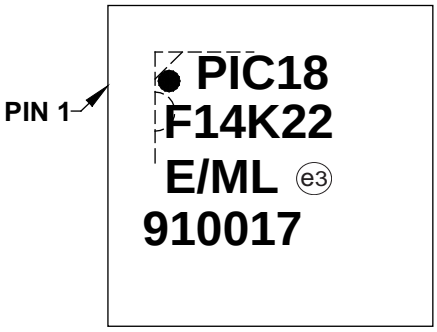

Example




20-Lead QFN (4x4x0.9 mm)




Example




| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week ‘01’) |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC® designator for Matte Tin (Sn) |
| | * | This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package. |

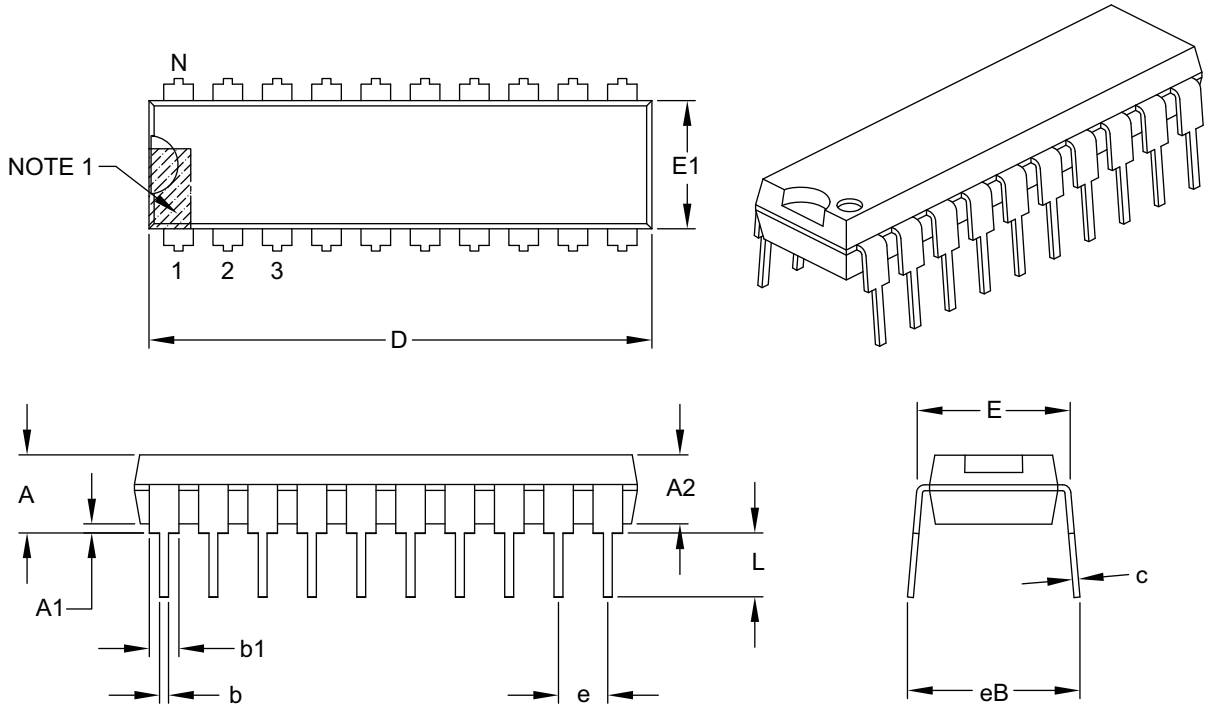
**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

## 28.2 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




| Units | | INCHES | | |
|---|---|---|---|---|
| Dimension Limits | | 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.

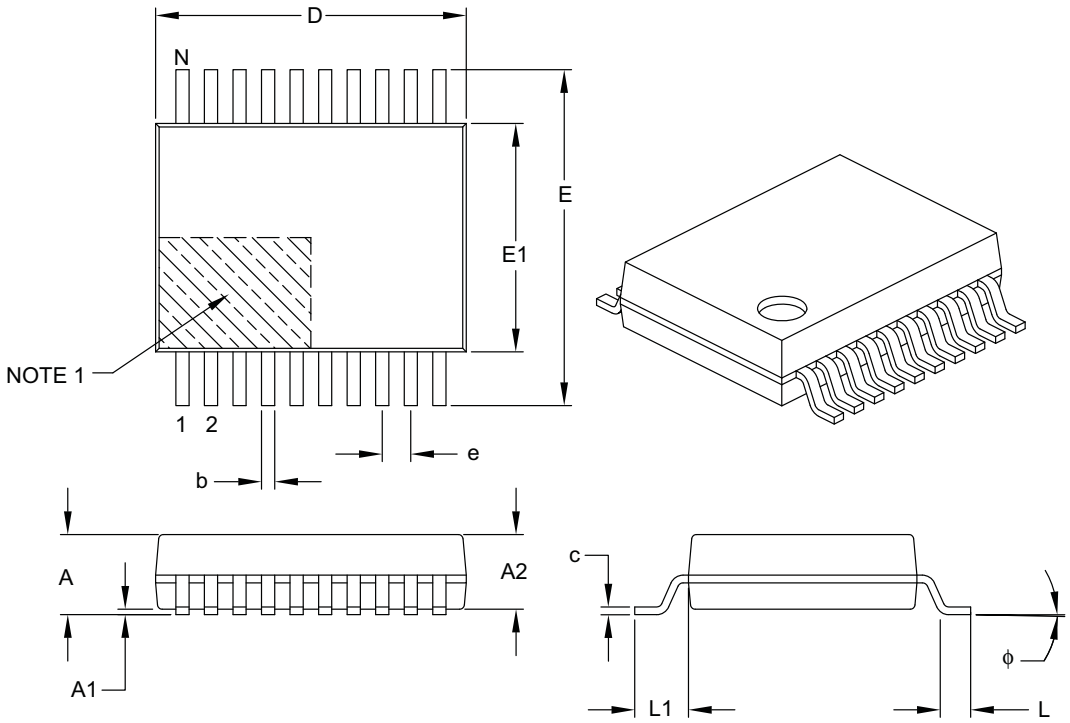
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-019B


© 2009-2016 Microchip Technology Inc.

### 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 | | 20 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | – | – | 2.00 |
| Molded Package Thickness | A2 | 1.65 | 1.75 | 1.85 |
| Standoff | A1 | 0.05 | – | – |
| Overall Width | E | 7.40 | 7.80 | 8.20 |
| Molded Package Width | E1 | 5.00 | 5.30 | 5.60 |
| Overall Length | D | 6.90 | 7.20 | 7.50 |
| Foot Length | L | 0.55 | 0.75 | 0.95 |
| Footprint | L1 | | 1.25 REF | |
| Lead Thickness | c | 0.09 | – | 0.25 |
| Foot Angle | ϕ | 0° | 4° | 8° |
| Lead Width | b | 0.22 | – | 0.38 |

**Notes:**

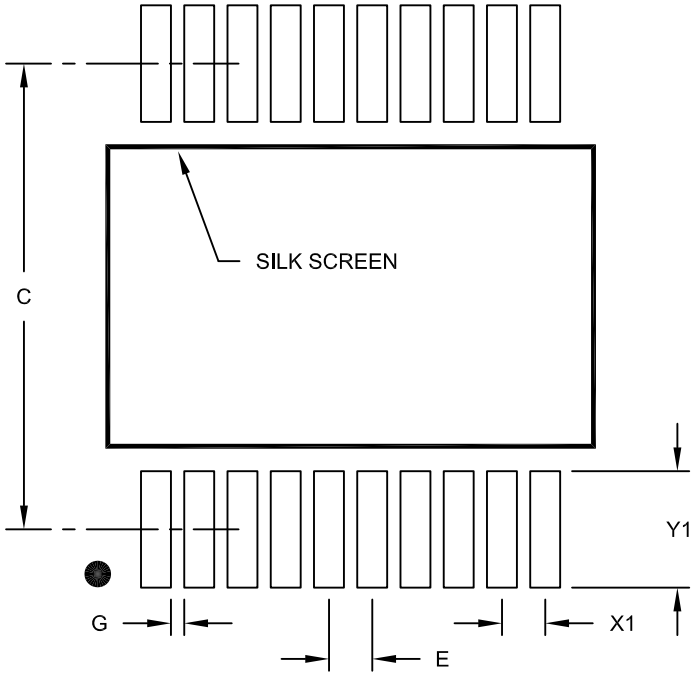
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-072B

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




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.65 BSC | |
| Contact Pad Spacing | C | | 7.20 | |
| Contact Pad Width (X20) | X1 | | | 0.45 |
| Contact Pad Length (X20) | Y1 | | | 1.75 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

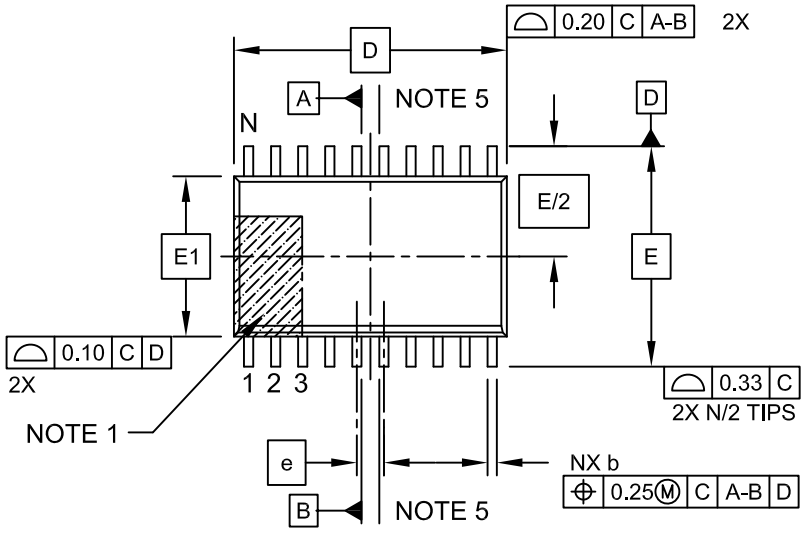
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2072A

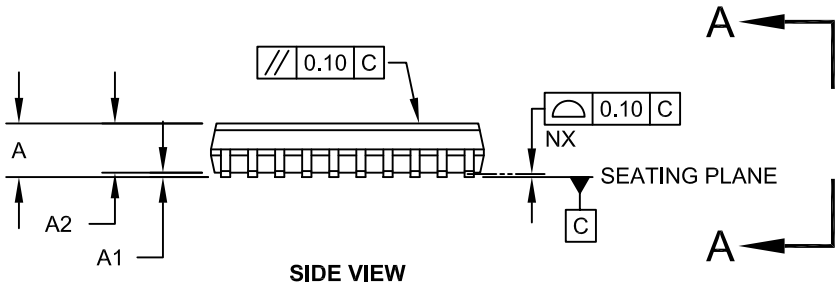

© 2009-2016 Microchip Technology Inc.

### 20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

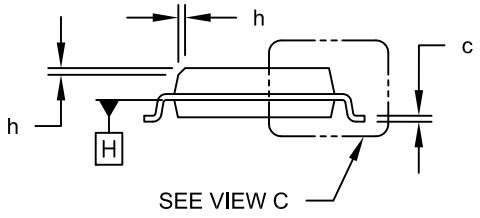
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




**TOP VIEW**




**SIDE VIEW**

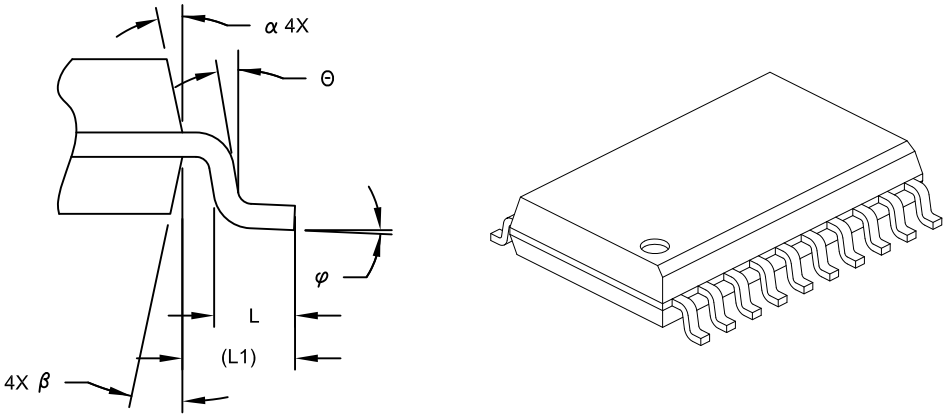



**VIEW A-A**

Microchip Technology Drawing C04-094C Sheet 1 of 2

## 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 | |
| Lead Angle | Θ | 0° | - | - |
| Foot Angle | φ | 0° | - | 8° |
| Lead Thickness | c | 0.20 | - | 0.33 |
| Lead Width | b | 0.31 | - | 0.51 |
| Mold Draft Angle Top | α | 5° | - | 15° |
| Mold Draft Angle Bottom | β | 5° | - | 15° |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

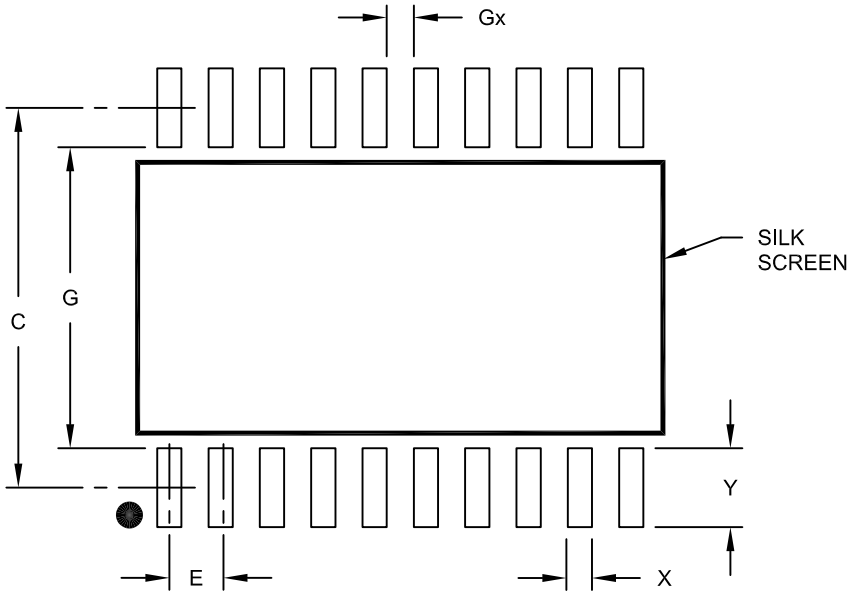
   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing No. C04-094C Sheet 2 of 2


© 2009-2016 Microchip Technology Inc.

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




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 1.27 BSC | |
| Contact Pad Spacing | C | | 9.40 | |
| Contact Pad Width (X20) | X | | | 0.60 |
| Contact Pad Length (X20) | Y | | | 1.95 |
| Distance Between Pads | Gx | 0.67 | | |
| Distance Between Pads | G | 7.45 | | |

Notes:

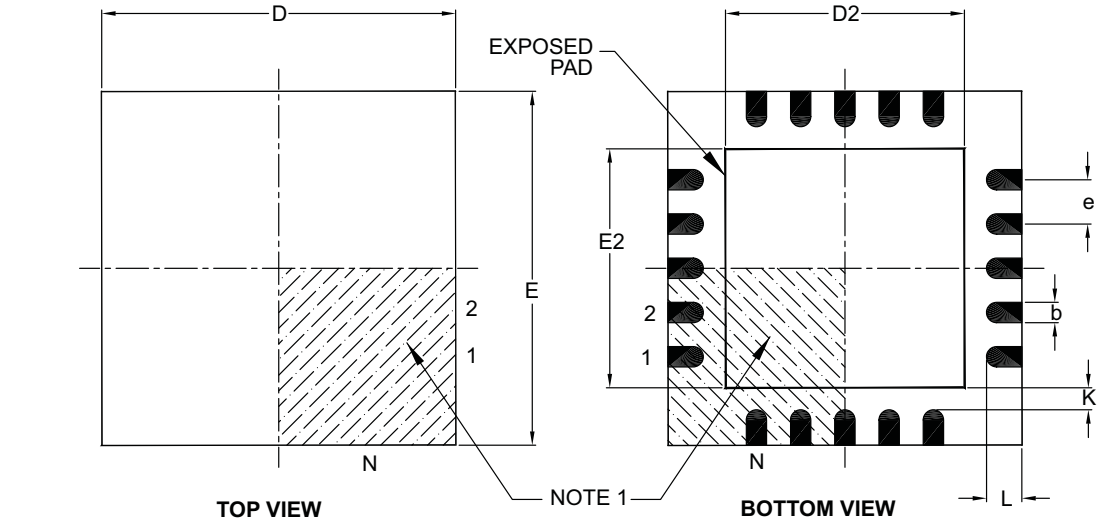
1. Dimensioning and tolerancing per ASME Y14.5M

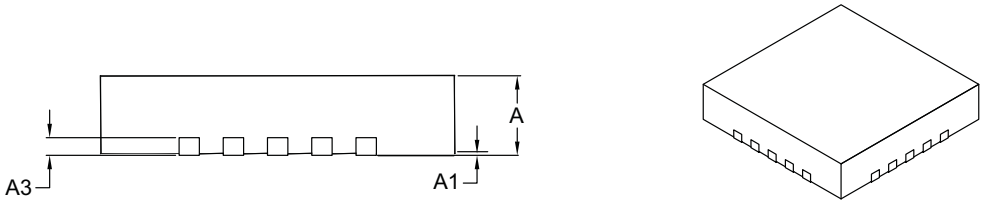
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2094A

### 20-Lead Plastic Quad Flat, No Lead Package (ML) – 4x4x0.9 mm Body [QFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 20 | |
| Pitch | e | | 0.50 BSC | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Contact Thickness | A3 | | 0.20 REF | |
| Overall Width | E | | 4.00 BSC | |
| Exposed Pad Width | E2 | 2.60 | 2.70 | 2.80 |
| Overall Length | D | | 4.00 BSC | |
| Exposed Pad Length | D2 | 2.60 | 2.70 | 2.80 |
| Contact Width | b | 0.18 | 0.25 | 0.30 |
| Contact Length | L | 0.30 | 0.40 | 0.50 |
| Contact-to-Exposed Pad | K | 0.20 | – | – |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

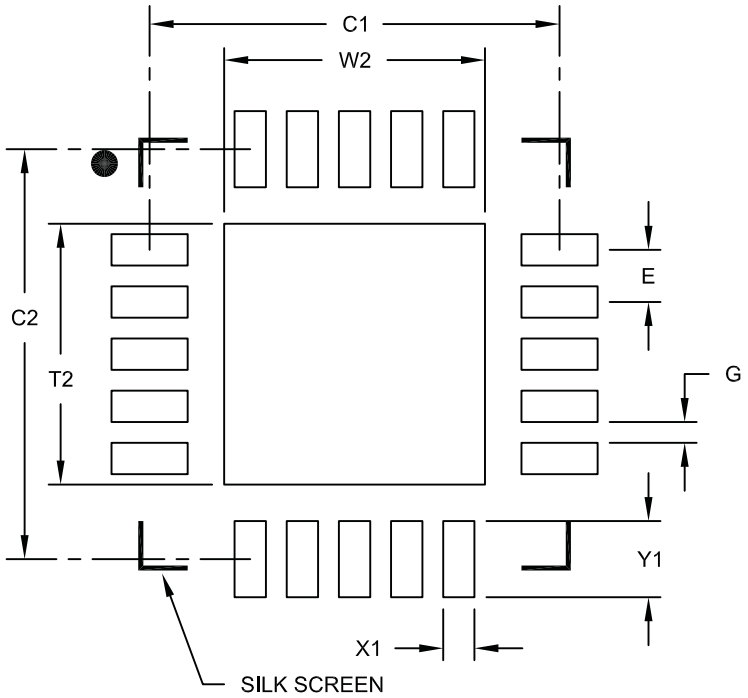
BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-126B


© 2009-2016 Microchip Technology Inc.

20-Lead Plastic Quad Flat, No Lead Package (ML) - 4x4 mm Body [QFN]
With 0.40 mm Contact Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.50 BSC | |
| Optional Center Pad Width | W2 | | | 2.50 |
| Optional Center Pad Length | T2 | | | 2.50 |
| Contact Pad Spacing | C1 | | 3.93 | |
| Contact Pad Spacing | C2 | | 3.93 | |
| Contact Pad Width | X1 | | | 0.30 |
| Contact Pad Length | Y1 | | | 0.73 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2126A

## APPENDIX A: REVISION HISTORY

### Revision A (February 2009)

Original data sheet for PIC18(L)F1XK22 devices.

### Revision B (04/2009)

Revised data sheet title; Revised Peripheral Features section; Revised Table 3-1, Table 3-2; Revised Example 15-1; Revised Table 21-4.

### Revision C (10/2009)

Updated Table 1-1; Updated the “Electrical Specifications” section (Figures 25-1 to 25-4; sub-sections 25.1, 25.2, 25.3, 25.4, 25.5, 25.6, 25.7, 25.8, Added Param No. OS09 to Table 25-2; Added Param No. D003A and Note 1 to Table 25-12); Added graphs to the “DC and AC Characteristics Graphs and Charts” section; Other minor corrections.

### Revision D (05/2010)

Revised Section 1.3 (deleted #2); Revised Figure 1-1; Added Table 2-4; Removed register EEADRH from Tables 3-1 and 3-2; Revised Section 5 (Data EEPROM Memory); Updated Example 5-2 and Table 5-1; Revised Section 13.4.4 (Enhanced PWM Auto-Shut-down Mode); Added Note 4 below Register 13-2; Revised Figure 16-1; Revised Equation 20-1; Removed sub-section 20.1.3 (Output Clamped to VSS); Updated Figure 20-1; Revised Tables 21-4 and Table 22-1; Updated Register 22-5, Figure 25-5, Table 25-2, Table 25-8, Table 25-10 and Table 25-12; Updated the Electrical Specification section; Other minor corrections.

### Revision E (10/2011)

Updated data sheet to new format; Updated the Pin Diagrams; Updated the Electrical Specifications section; Updated the Packaging Information section; Updated Table B-1; Updated the Product Identification System section; Other minor corrections.

### Revision F (04/2016)

Updated Analog Features section on page 1; Updated Tables 1-2, 3-2, 8-5, 8-6, 16-2 and 22-4; Added Note 3 to Tables 3-2, 8-1 and 8-2; Added Note 1 to Tables 9-1, 10-2, 12-1 and 17-2, and Register 8-4; Updated Figures 3-7, 9-1 and 9-2; Updated Registers 13-2, 16-2, 19-1; Updated Section 1.1.2, 7.9 and 8.1; Replaced chapter 20.0 (Voltage References) with chapter 20.0 (Fixed Voltage Reference) and 21.0 (Digital-to-Analog Converter (DAC) Module); Updated Chapter 26.0 (Electrical Specifications); Other minor corrections.


© 2009-2016 Microchip Technology Inc.

## APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table .

**TABLE B-1: DEVICE DIFFERENCES**

| Features | PIC18F13K22 | PIC18F14K22 | PIC18LF13K22 | PIC18LF14K22 |
|---|---|---|---|---|
| Program Memory (Bytes) | 8192 | 16384 | 8192 | 16384 |
| Program Memory (Instructions) | 4096 | 8192 | 4096 | 8192 |
| Data Memory SRAM (bytes) | 256 | 512 | 256 | 512 |
| Data Memory EEPROM (bytes) | 256 | 256 | 256 | 256 |
| $V_{DD}$ Min(V) | 2.3 | 2.3 | 1.8 | 1.8 |
| $V_{DD}$ Max(V) | 5.5 | 5.5 | 3.6 | 3.6 |
| Packages | 20-pin PDIP<br>20-pin SOIC<br>20-pin SSOP<br>20-Pin QFN | 20-pin PDIP<br>20-pin SOIC<br>20-pin SSOP<br>20-Pin QFN | 20-pin PDIP<br>20-pin SOIC<br>20-pin SSOP<br>20-Pin QFN | 20-pin PDIP<br>20-pin SOIC<br>20-pin SSOP<br>20-Pin QFN |

© 2009-2016 Microchip Technology Inc.

## THE MICROCHIP WEBSITE

Microchip provides online support via our WWW site at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website 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 website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

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 website at: http://microchip.com/support**


© 2009-2016 Microchip Technology Inc.

## PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| PART NO. | [X](2) | - | X | /XX | XXX |
|---|---|---|---|---|---|
| Device | Tape and Reel Option | | Temperature Range | Package | Pattern |

| | |
|---|---|
| **Device:** | PIC18F13K22, PIC18LF13K22<br>PIC18F14K22, PIC18LF14K22 |
| **Tape and Reel Option:** | Blank = standard packaging (tube or tray)<br>T = Tape and Reel(1), (2) |
| **Temperature Range:** | E = -40°C to +125°C (Extended)<br>I = -40°C to +85°C (Industrial) |
| **Package:** | ML = QFN<br>P = PDIP<br>SO = SOIC<br>SS = SSOP |
| **Pattern:** | QTP, SQTP, Code or Special Requirements (blank otherwise) |

**Examples:**

a) PIC18F14K22-E/P 301 = Extended temp., PDIP package, QTP pattern #301.

b) PIC18LF14K22-E/SO = Extended temp., SOIC package.

c) PIC18LF14K22-E/ML = Extended temp., QFN package.

d) PIC18F13K22T-I/SS = Industrial temp., SSOP package, Tape and Reel.

**Note 1:** Tape and Reel option is available for ML, MV, PT, SO and SS packages with industrial Temperature Range only.

**2:** Tape and Reel identifier only appears in catalog part number description. This identifier is used for ordering purposes and is not printed on the device package.

**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 unless otherwise stated.

**Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, flexPWR, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, PICSTART, PIC$^{32}$ logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

The Embedded Control Solutions Company and mTouch are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, ECAN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, KleerNet, KleerNet logo, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, RightTouch logo, REAL ICE, SQI, Serial Quad I/O, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, 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.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

© 2009-2016, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-0464-4

**QUALITY MANAGEMENT SYSTEM**
**CERTIFIED BY DNV**
**═ ISO/TS 16949 ═**

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*


© 2009-2016 Microchip Technology Inc.

# 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://www.microchip.com/support
Web Address:
www.microchip.com

**Atlanta**
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

**Austin, TX**
Tel: 512-257-3370

**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**
Novi, MI
Tel: 248-848-4000

**Houston, TX**
Tel: 281-894-5983

**Indianapolis**
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

**New York, NY**
Tel: 631-435-6000

**San Jose, CA**
Tel: 408-735-9110

**Canada - Toronto**
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: 852-2943-5100
Fax: 852-2401-3431

**Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**China - Beijing**
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

**China - Chengdu**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**China - Chongqing**
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

**China - Dongguan**
Tel: 86-769-8702-9880

**China - Hangzhou**
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

**China - Hong Kong SAR**
Tel: 852-2943-5100
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: 86-755-8864-2200
Fax: 86-755-8203-1760

**China - Wuhan**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**China - Xian**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

## ASIA/PACIFIC

**China - Xiamen**
Tel: 86-592-2388138
Fax: 86-592-2388130

**China - Zhuhai**
Tel: 86-756-3210040
Fax: 86-756-3210049

**India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

**India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**India - Pune**
Tel: 91-20-3019-1500

**Japan - Osaka**
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

**Japan - Tokyo**
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

**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: 886-3-5778-366
Fax: 886-3-5770-955

**Taiwan - Kaohsiung**
Tel: 886-7-213-7828

**Taiwan - Taipei**
Tel: 886-2-2508-8600
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 - Dusseldorf**
Tel: 49-2129-3766400

**Germany - Karlsruhe**
Tel: 49-721-625370

**Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Italy - Venice**
Tel: 39-049-7625286

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Poland - Warsaw**
Tel: 48-22-3325737

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**Sweden - Stockholm**
Tel: 46-8-5090-4654

**UK - Wokingham**
Tel: 44-118-921-5800
Fax: 44-118-921-5820

07/14/15

© 2009-2016 Microchip Technology Inc.

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Microchip:

PIC18F13K22-I/P PIC18F13K22-I/ML PIC18F13K22-E/SO PIC18F14K22-I/P PIC18F13K22-E/ML PIC18LF13K22-I/ML PIC18F14K22T-I/SO PIC18LF14K22-I/SS PIC18LF14K22-I/SO PIC18F13K22T-I/SS PIC18F14K22-E/SS PIC18F14K22T-I/ML PIC18F13K22-E/SS PIC18F13K22T-I/SO PIC18F13K22-I/SS PIC18F14K22-E/ML PIC18F14K22-I/ML PIC18LF13K22-I/SS PIC18LF13K22-I/SO PIC18LF14K22-E/P PIC18F14K22T-I/SS PIC18F13K22-I/SO PIC18LF14K22-I/ML PIC18LF14K22-I/P PIC18LF14K22-E/SS PIC18LF14K22-E/SO PIC18LF13K22-I/P PIC18LF14K22T-I/SO PIC18LF13K22-E/SS PIC18LF14K22T-I/SS PIC18LF14K22-E/ML PIC18LF13K22-E/P PIC18F13K22T-I/ML PIC18LF14K22T-I/ML PIC18LF13K22-E/ML PIC18LF13K22T-I/ML PIC18LF13K22-E/SO PIC18LF13K22T-I/SS PIC18LF13K22T-I/SO PIC18F13K22-E/P PIC18F14K22-E/P PIC18F14K22-E/SO PIC18F14K22-I/SO PIC18F14K22-I/SS

Стандарт Электрон Связь

Мы молодая и активно развивающаяся компания в области поставок электронных компонентов. Мы поставляем электронные компоненты отечественного и импортного производства напрямую от производителей и с крупнейших складов мира.

Благодаря сотрудничеству с мировыми поставщиками мы осуществляем комплексные и плановые поставки широчайшего спектра электронных компонентов.

Собственная эффективная логистика и склад в обеспечивает надежную поставку продукции в точно указанные сроки по всей России.

Мы осуществляем техническую поддержку нашим клиентам и предпродажную проверку качества продукции. На все поставляемые продукты мы предоставляем гарантию .

Осуществляем поставки продукции под контролем ВП МО РФ на предприятия военно-промышленного комплекса России , а также работаем в рамках 275 ФЗ с открытием отдельных счетов в уполномоченном банке. Система менеджмента качества компании соответствует требованиям ГОСТ ISO 9001.

Минимальные сроки поставки, гибкие цены, неограниченный ассортимент и индивидуальный подход к клиентам являются основой для выстраивания долгосрочного и эффективного сотрудничества с предприятиями радиоэлектронной промышленности, предприятиями ВПК и научно-исследовательскими институтами России.

С нами вы становитесь еще успешнее!

**Наши контакты:**

**Телефон:** +7 812 627 14 35

**Электронная почта:** sales@st-electron.ru

**Адрес:** 198099, Санкт-Петербург,
Промышленная ул, дом № 19, литера Н,
помещение 100-Н Офис 331