



MICROCHIP

PIC18(L)F2X/45K50

28/40/44-Pin, Low-Power, High-Performance Microcontrollers with XLP Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Crystal-less Full Speed (12 Mb/s) and Low-Speed Operation (1.5 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 Bidirectional)
- 1 Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver

Flexible Oscillator Structure:

- 3x and 4xPLL Clock Multipliers
- Two External Clock modes, Up to 48 MHz (12 MIPS)
- Internal 31 kHz Oscillator
- Internal Oscillator, 31 kHz to 16 MHz
 - Factory calibrated to $\pm 1\%$
 - Self-tune to $\pm 0.20\%$ max. from USB or secondary oscillator
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
 - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- Up to 33 I/O pins plus 3 Input-Only Pins
 - High-current Sink/Source 25 mA/25 mA
 - Three programmable external interrupts
 - 11 programmable interrupts-on-change
 - Nine programmable weak pull-ups
 - Programmable slew rate
- SR Latch
- Enhanced Capture/Compare/PWM (ECCP) module:
 - One, two or four PWM outputs
 - Selectable polarity
 - Programmable dead time
 - Auto-shutdown and auto-restart
 - Pulse steering control
- Capture/Compare/PWM (CCP) module
- Master Synchronous Serial Port (MSSP) module Supporting 3-Wire SPI (all four modes) and I²C™ Master and Slave modes
- Two Analog Comparators with Input Multiplexing
- 10-Bit Analog-to-Digital (A/D) Converter module:
 - Up to 25 input channels
 - Auto-acquisition capability
 - Conversion available during Sleep

- Digital-to-Analog Converter (DAC) module:
 - Fixed Voltage Reference (FVR) with 1.024V, 2.048V and 4.096V output levels
 - 5-bit rail-to-rail resistive DAC with positive and negative reference selection
- High/Low-Voltage Detect module
- Charge Time Measurement Unit (CTMU):
 - Supports capacitive touch sensing for touch screens and capacitive switches
- Enhanced USART module:
 - Supports RS-485, RS-232 and LIN/J2602
 - Auto-wake-up on Start bit
 - Auto-Baud Detect

Extreme Low-Power Management with XLP:

- Sleep mode: 20 nA, typical
- Watchdog Timer: 300 nA, typical
- Timer1 Oscillator: 800 nA @ 32 kHz
- Peripheral Module Disable

Special Microcontroller Features:

- Low-Power, High-Speed CMOS Flash Technology
- C Compiler Optimized Architecture for Re-Entrant Code
- Power Management Features:
 - Run: CPU on, peripherals on, SRAM on
 - Idle: CPU off, peripherals on, SRAM on
 - Sleep: CPU off, peripherals off, SRAM on
- Priority Levels for Interrupts
- Self-Programmable under Software Control
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
 - Programmable period from 4 ms to 131s
- Single-Supply In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) with Three Breakpoints via Two Pins
- Optional dedicated ICD/ICSP Port (44-pin TQFP Package Only)
- Wide Operating Voltage Range:
 - F devices: 2.3V to 5.5V
 - LF devices: 1.8V to 3.6V
- Flash Program Memory of 10,000 Erase/Write Cycles Minimum and 20-year Data Retention

PIC18(L)F2X/45K50

PIC18(L)F2X/45K50 Family Types

| Device | Program Memory | | Data Memory | | Pins | I/O | 10-Bit A/D Channels | Comparators | CCP/ ECCP | BOR/LVD | CTMU | MSSP | USART | Timers 8-bit/16-bit | USB 2.0 |
|----------------|------------------|-----------------------------|-----------------|---------------------------|-------|-----|---------------------|-------------|--------------|---------|------|------|-------|------------------------|---------|
| | Flash (bytes) | Single-Word Instructions | SRAM (bytes) | Data EEPROM (bytes) | | | | | | | | | | | |
| PIC18(L)F45K50 | 32K | 16384 | 2048 | 256 | 40/44 | 36 | 25-ch | 2 | 1/1 | Yes | Yes | 1 | 1 | 2/2 | Yes |
| PIC18(L)F25K50 | 32K | 16384 | 2048 | 256 | 28 | 25 | 14-ch | 2 | 1/1 | Yes | Yes | 1 | 1 | 2/2 | Yes |
| PIC18(L)F24K50 | 16K | 8192 | 2048 | 256 | 28 | 25 | 14-ch | 2 | 1/1 | Yes | Yes | 1 | 1 | 2/2 | Yes |

Pin Diagrams

FIGURE 1: 28-PIN SPDIP (300 MIL), SOIC, SSOP

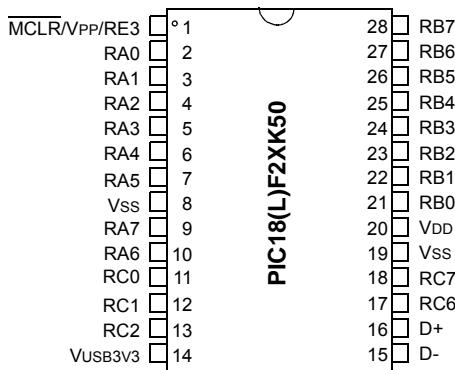
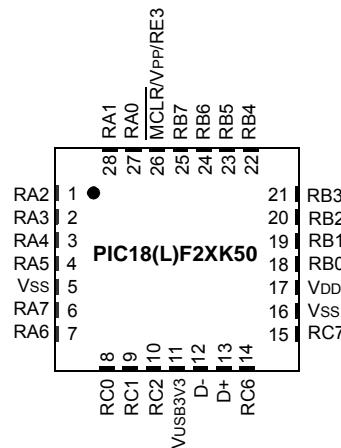


FIGURE 2: 28-PIN QFN



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

PIC18(L)F2X/45K50

FIGURE 3: 40-PIN PDIP (600 MIL)

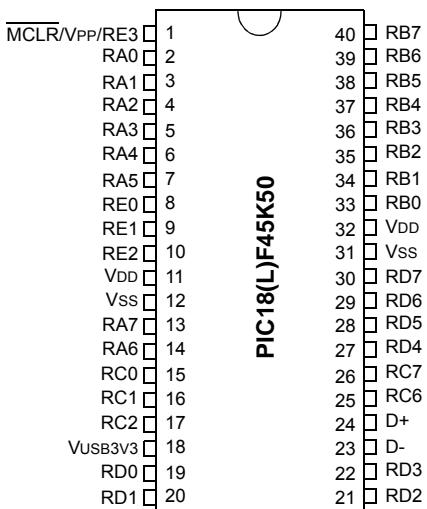
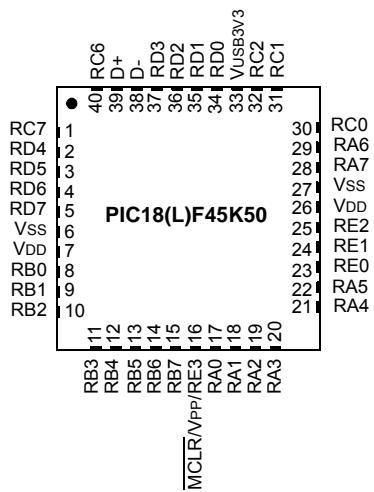


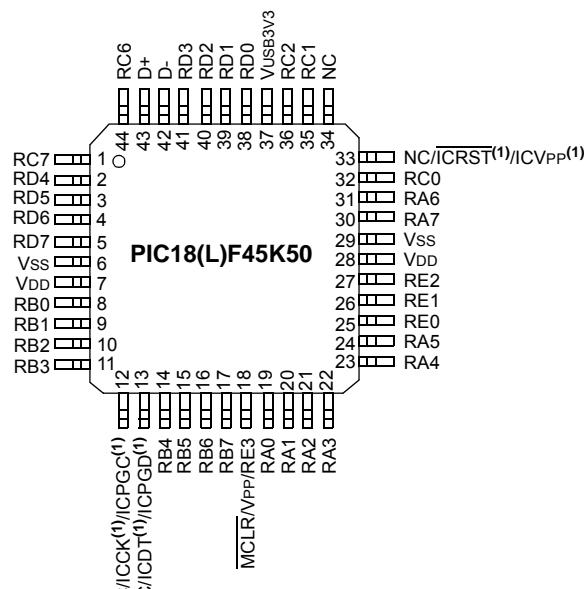
FIGURE 4: 40-PIN UQFN



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

PIC18(L)F2X/45K50

FIGURE 5: 44-PIN TQFP



Note 1: Special ICPORT programming/debug port features available when ICPRT = 1

TABLE 1: PIC18(L)F2X/45K50 PIN SUMMARY

| I/O | 28-Pin SPDIP/SOIC/SSOP | 28-Pin QFN | 40-Pin PDIP | 40-Pin UQFN | 44-Pin TQFP | Analog | Comparator | CTMU | SR Latch | Reference | USB | (E)CCP | EUSART | MSSP | Timers | Interrupts | Pull-up | Basic | ICD |
|-----|------------------------|------------|-------------|-------------|-------------|--------|------------|-------|----------|-------------|-----|---------------------|--------|---------|--------------------------|------------|---------|-----------|-----|
| RA0 | 2 | 27 | 2 | 17 | 19 | AN0 | C12IN0- | — | — | — | — | — | — | — | — | — | — | — | — |
| RA1 | 3 | 28 | 3 | 18 | 20 | AN1 | C12IN1- | CTCMP | — | — | — | — | — | — | — | — | — | — | — |
| RA2 | 4 | 1 | 4 | 19 | 21 | AN2 | C2IN+ | — | — | VREF-DACOUT | — | — | — | — | — | — | — | — | — |
| RA3 | 5 | 2 | 5 | 20 | 22 | AN3 | C1IN+ | — | — | VREF+ | — | — | — | — | — | — | — | — | — |
| RA4 | 6 | 3 | 6 | 21 | 23 | — | C1OUT | — | SRQ | — | — | — | — | — | T0CKI | — | — | — | — |
| RA5 | 7 | 4 | 7 | 22 | 24 | AN4 | C2OUT | — | SRNQ | HLVDIN | — | — | — | — | SS | — | — | — | — |
| RA6 | 10 | 7 | 14 | 29 | 31 | — | — | — | — | — | — | — | — | — | — | — | — | OSC2 CLKO | — |
| RA7 | 9 | 6 | 13 | 28 | 30 | — | — | — | — | — | — | — | — | — | — | — | — | OSC1 CLKI | — |
| RB0 | 21 | 18 | 33 | 8 | 8 | AN12 | — | — | SRI | — | — | FLT0 | — | SDI SDA | — | INT0 | Y | — | — |
| RB1 | 22 | 19 | 34 | 9 | 9 | AN10 | C12IN3- | — | — | — | — | P1C ⁽⁵⁾ | — | SCK SCL | — | INT1 | Y | — | — |
| RB2 | 23 | 20 | 35 | 10 | 10 | AN8 | — | CTED1 | — | — | — | P1B ⁽⁵⁾ | — | — | — | INT2 | Y | — | — |
| RB3 | 24 | 21 | 36 | 11 | 11 | AN9 | C12IN2- | CTED2 | — | — | — | CCP2 ⁽¹⁾ | — | SDO | — | — | — | — | — |
| RB4 | 25 | 22 | 37 | 12 | 14 | AN11 | — | — | — | — | — | P1D ⁽⁵⁾ | — | — | — | IOCB4 | Y | — | — |
| RB5 | 26 | 23 | 38 | 13 | 15 | AN13 | — | — | — | — | — | — | — | — | T1G T3CKI ⁽²⁾ | IOCB5 | — | — | — |
| RB6 | 27 | 24 | 39 | 14 | 16 | — | — | — | — | — | — | — | — | — | — | IOCB6 | Y | PGC | — |

Note 1: Alternate CCP2 pin location based on Configuration bit.

2: Alternate T3CKI pin location based on Configuration bits.

3: Pins are enabled when ICPRT = 1, otherwise, they are disabled.

4: Location on 40/44-Pin parts (PIC18(L)F45K50). Function not on this pin on 28-Pin parts (PIC18(L)F2XK50).

5: Location on 28-Pin parts (PIC18(L)F2XK50). Function not on this pin on 40/44-Pin parts (PIC18(L)F45K50).

6: Alternate SDO pin location based on Configuration bits.

7: RE3 can be used for digital input only (no output functionality).

TABLE 1: PIC18(L)F2X/45K50 PIN SUMMARY (CONTINUED)

| IO | 28-Pin SPDIP/SOIC/SSOP | 28-Pin QFN | 40-Pin PDIP | 40-Pin UQFN | 44-Pin TQFP | Analog | Comparator | CTMU | SR Latch | Reference | USB | (E)CCP | EUSART | MSSP | Timers | Interrupts | Pull-up | Basic | LCD |
|-----|------------------------|------------|-------------|-------------|-------------|--------|------------|-------|----------|-----------|---------|--------------------|--------------------|--------------------------------|--------|------------|---------|---------|-----|
| RB7 | 28 | 25 | 40 | 15 | 17 | — | — | — | — | — | — | — | — | — | — | IOCB7 | Y | PGD | — |
| RC0 | 11 | 8 | 15 | 30 | 32 | — | — | — | — | — | — | — | — | SOSCO T1CKI T3CKI T3G | IOCC0 | — | — | — | |
| RC1 | 12 | 9 | 16 | 31 | 35 | — | — | — | — | — | — | CCP2 | — | — | SOSCI | IOCC1 | — | — | — |
| RC2 | 13 | 10 | 17 | 32 | 36 | AN14 | — | CTPLS | — | — | — | CCP1 P1A | — | — | — | IOCC2 | — | — | — |
| — | 14 | 11 | 18 | 33 | 37 | — | — | — | — | — | VUSB3V3 | — | — | — | — | — | — | VDDCORE | — |
| — | 15 | 12 | 23 | 38 | 42 | — | — | — | — | — | D- | — | — | — | — | IOCC4 | — | — | — |
| — | 16 | 13 | 24 | 39 | 43 | — | — | — | — | — | D+ | — | — | — | — | IOCC5 | — | — | — |
| RC6 | 17 | 14 | 25 | 40 | 44 | AN18 | — | — | — | — | — | TX CK | — | — | — | IOCC6 | — | — | — |
| RC7 | 18 | 15 | 26 | 1 | 1 | AN19 | — | — | — | — | — | RX DT | SDO ⁽⁶⁾ | — | — | IOCC7 | — | — | — |
| RD0 | — | — | 19 | 34 | 38 | AN20 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD1 | — | — | 20 | 35 | 39 | AN21 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD2 | — | — | 21 | 36 | 40 | AN22 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD3 | — | — | 22 | 37 | 41 | AN23 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD4 | — | — | 27 | 2 | 2 | AN24 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD5 | — | — | 28 | 3 | 3 | AN25 | — | — | — | — | — | P1B ⁽⁴⁾ | — | — | — | — | — | — | — |
| RD6 | — | — | 29 | 4 | 4 | AN26 | — | — | — | — | — | P1C ⁽⁴⁾ | — | — | — | — | — | — | — |
| RD7 | — | — | 30 | 5 | 5 | AN27 | — | — | — | — | — | P1D ⁽⁴⁾ | — | — | — | — | — | — | — |

Note 1: Alternate CCP2 pin location based on Configuration bit.

2: Alternate T3CKI pin location based on Configuration bits.

3: Pins are enabled when ICPRT = 1, otherwise, they are disabled.

4: Location on 40/44-Pin parts (PIC18(L)F45K50). Function not on this pin on 28-Pin parts (PIC18(L)F2XK50).

5: Location on 28-Pin parts (PIC18(L)F2XK50). Function not on this pin on 40/44-Pin parts (PIC18(L)F45K50).

6: Alternate SDO pin location based on Configuration bits.

7: RE3 can be used for digital input only (no output functionality).

TABLE 1: PIC18(L)F2X/45K50 PIN SUMMARY (CONTINUED)

| O/I | 28-Pin SPDIP/SOIC/SSOP | 28-Pin QFN | 40-Pin PDIP | 40-Pin UQFN | 44-Pin TQFP | Analog | Comparator | CTMU | SR Latch | Reference | USB | (E)CCP | EUSART | MSSP | Timers | Interrupts | Pull-up | Basic | ICD |
|--------------------|------------------------|------------|-------------|-------------|-------------------|--------|------------|------|----------|-----------|-----|--------|--------|------|--------|------------|----------------------|----------------------|-----|
| RE0 | — | — | 8 | 23 | 25 | AN5 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RE1 | — | — | 9 | 24 | 26 | AN6 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RE2 | — | — | 10 | 25 | 27 | AN7 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RE3 ⁽⁷⁾ | 1 | 26 | 1 | 16 | 18 | — | — | — | — | — | — | — | — | — | — | — | Y | <u>MCLR</u> VPP | — |
| — | 20 | 17 | 11, 32 | 7, 26 | 7, 28 | — | — | — | — | — | — | — | — | — | — | — | VDD | — | — |
| — | 8, 19 | 5, 16 | 12, 31 | 6, 27 | 6, 29 | — | — | — | — | — | — | — | — | — | — | — | VSS | — | — |
| — | — | — | — | — | 12 ⁽³⁾ | — | — | — | — | — | — | — | — | — | — | — | ICPGC ⁽³⁾ | ICCK ⁽³⁾ | — |
| — | — | — | — | — | 13 ⁽³⁾ | — | — | — | — | — | — | — | — | — | — | — | ICPGD ⁽³⁾ | ICDT ⁽³⁾ | — |
| — | — | — | — | — | 33 ⁽³⁾ | — | — | — | — | — | — | — | — | — | — | — | ICVPP ⁽³⁾ | ICRST ⁽³⁾ | — |

Note 1: Alternate CCP2 pin location based on Configuration bit.

2: Alternate T3CKI pin location based on Configuration bits.

3: Pins are enabled when ICPRT = 1, otherwise, they are disabled.

4: Location on 40/44-Pin parts (PIC18(L)F45K50). Function not on this pin on 28-Pin parts (PIC18(L)F2XK50).

5: Location on 28-Pin parts (PIC18(L)F2XK50). Function not on this pin on 40/44-Pin parts (PIC18(L)F45K50).

6: Alternate SDO pin location based on Configuration bits.

7: RE3 can be used for digital input only (no output functionality).

PIC18(L)F2X/45K50

Table of Contents

| | | |
|------|---|-----|
| 1.0 | Device Overview | 10 |
| 2.0 | Guidelines for Getting Started with PIC18(L)F2X/45K50 Microcontrollers | 23 |
| 3.0 | Oscillator Module (With Fail-Safe Clock Monitor)..... | 28 |
| 4.0 | Power-Managed Modes | 52 |
| 5.0 | Reset | 63 |
| 6.0 | Memory Organization | 73 |
| 7.0 | Flash Program Memory | 95 |
| 8.0 | Data EEPROM Memory | 104 |
| 9.0 | 8 x 8 Hardware Multiplier..... | 109 |
| 10.0 | Interrupts | 111 |
| 11.0 | I/O Ports | 128 |
| 12.0 | Timer0 Module..... | 153 |
| 13.0 | Timer1/3 Module with Gate Control..... | 156 |
| 14.0 | Timer2 Module..... | 168 |
| 15.0 | Capture/Compare/PWM Modules..... | 172 |
| 16.0 | Master Synchronous Serial Port (MSSP) Module | 203 |
| 17.0 | Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) | 258 |
| 18.0 | Analog-to-Digital Converter (ADC) Module | 287 |
| 19.0 | Comparator Module..... | 301 |
| 20.0 | Charge Time Measurement Unit (CTMU)..... | 310 |
| 21.0 | SR Latch..... | 325 |
| 22.0 | Fixed Voltage Reference (FVR)..... | 330 |
| 23.0 | Digital-to-Analog Converter (DAC) Module | 332 |
| 24.0 | Universal Serial Bus (USB) | 336 |
| 25.0 | High/Low-Voltage Detect (HLVD) | 364 |
| 26.0 | Special Features of the CPU | 370 |
| 27.0 | Instruction Set Summary | 390 |
| 28.0 | Development Support..... | 440 |
| 29.0 | Electrical Specifications | 444 |
| 30.0 | DC and AC Characteristics Graphs and Charts | 485 |
| 31.0 | Packaging Information..... | 486 |
| | Appendix A: Revision History..... | 503 |
| | Appendix B: Device Differences..... | 504 |
| | The Microchip Web Site | 505 |
| | Customer Change Notification Service | 505 |
| | Customer Support..... | 505 |
| | Product Identification System..... | 506 |

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 Web site at:

<http://www.microchip.com>

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS3000000A is version A of document DS3000000).

Errata

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

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

- Microchip's Worldwide Web site; <http://www.microchip.com>
- Your local Microchip sales office (see last page)

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

Customer Notification System

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

PIC18(L)F2X/45K50

1.0 DEVICE OVERVIEW

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

- PIC18(L)F45K50
- PIC18(L)F25K50
- PIC18(L)F24K50

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Flash program memory. On top of these features, the PIC18(L)F2X/45K50 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)F2X/45K50 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **Peripheral Module Disable bits:** User code can power down individual peripheral modules during Run and Idle modes for further lowering dynamic power reduction.
- **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 29.0 “Electrical Specifications”](#) for values.

1.1.2 UNIVERSAL SERIAL BUS (USB)

Devices in the PIC18(L)F2X/45K50 family incorporate a fully-featured USB communications module with a built-in transceiver that is compliant with the USB Specification Revision 2.0. The module supports both low-speed and full-speed communication for all supported data transfer types. The device incorporates its own on-chip transceiver and 3.3V regulator for USB.

1.1.3 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18(L)F2X/45K50 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
- Six External Clock modes, offering the option of using two pins (oscillator input and a divide-by-four clock output) or one pin (oscillator input, with the second pin reassigned as general I/O)
- Two External RC Oscillator modes with the same pin options as the External Clock modes
- An internal oscillator block which contains a 16 MHz HFINTOSC oscillator and a 31 kHz INTRC 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.
- 3x and 4x Phase Lock Loop (PLL) frequency multipliers, available to both external and internal oscillator modes, which allows clock speeds of up to 48 MHz. Used with the internal oscillator, the PLL gives users a complete selection of clock speeds, from 31 kHz to 48 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:

- **Active Clock Tuning:** This option allows the internal oscillator to automatically tune itself to match USB host or external 32.768 kHz secondary oscillator clock sources. Full-speed USB operation can now meet specification requirements without an external crystal, enabling lower-cost designs.
- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the INTRC. 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.

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. By using a bootloader routine located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18(L)F2X/45K50 family introduces an optional extension to the PIC18 instruction set, which adds eight new instructions and an Indexed Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **Enhanced CCP module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include:
 - Auto-shutdown, for disabling PWM outputs on interrupt or other select conditions
 - 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 EUSART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include automatic baud rate detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-Bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduce code overhead.
- **Dedicated ICD/ICSP™ Port:** These devices introduce the use of debugger and programming pins that are not multiplexed with other microcontroller features. Offered as an option in the TQFP packaged devices, this feature allows users to develop I/O intensive applications while retaining the ability to program and debug in the circuit.

- **Charge Time Measurement Unit (CTMU):** The CTMU is a flexible analog module that provides accurate differential time measurement between pulse sources, as well as asynchronous pulse generation. Together with other on-chip analog modules, the CTMU can precisely measure time, measure capacitance or relative changes in capacitance or generate output pulses that are independent of the system clock.
- **SR Latch Output:** A single SR latch with multiple Set and Reset inputs as well as separate latch outputs.

1.3 Details on Individual Family Members

Devices in the PIC18(L)F2X/45K50 family are available in 28-pin and 40/44-pin packages. The block diagram for the device family is shown in [Figure 1-1](#).

The devices have the following differences:

1. Flash program memory
2. A/D channels
3. I/O ports
4. Input Voltage Range/Power Consumption

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 the pin summary table ([Table 1](#)) and I/O description tables ([Table 1-2](#) and [Table 1-3](#)).

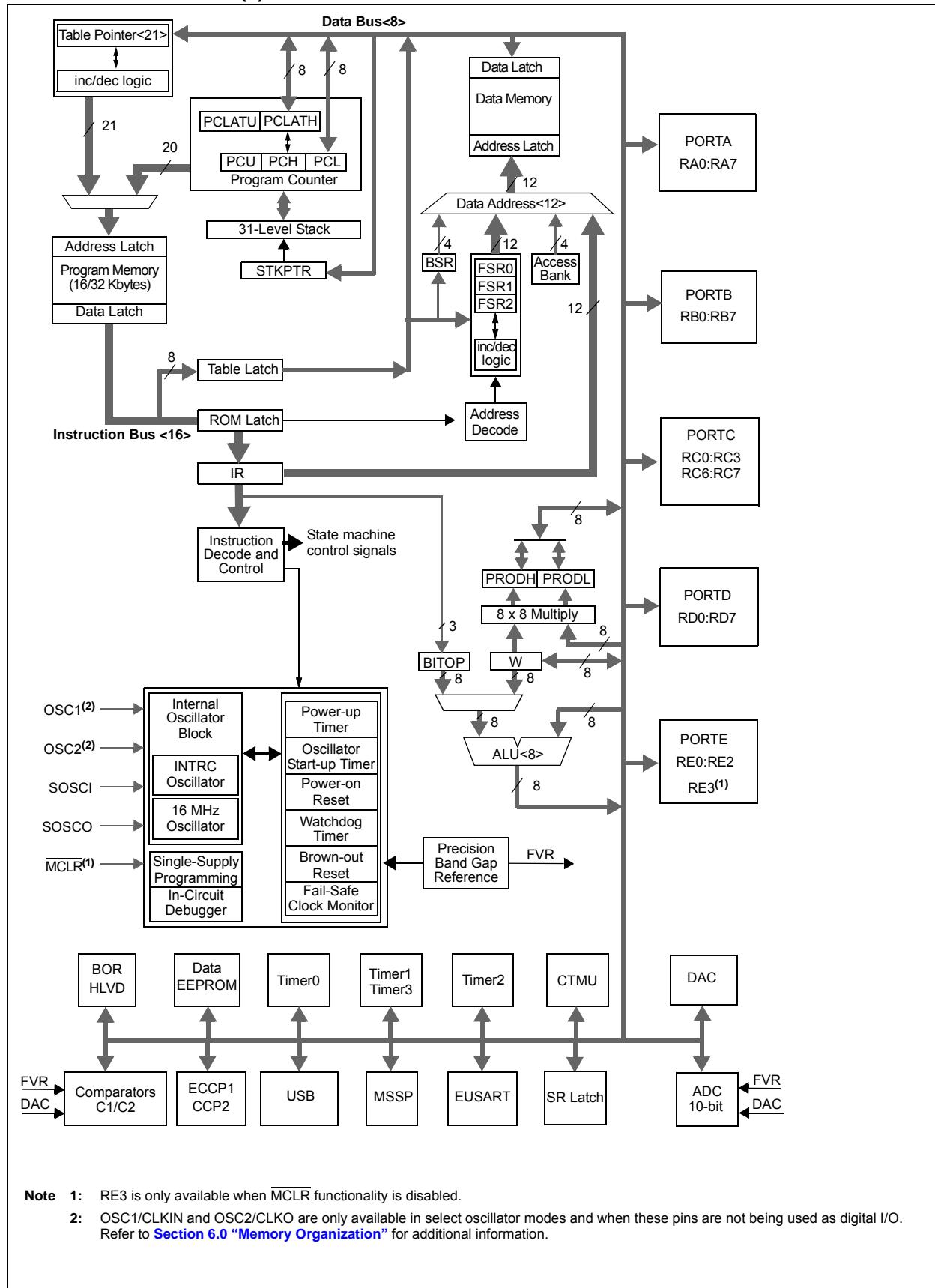
PIC18(L)F2X/45K50

TABLE 1-1: DEVICE FEATURES

| Features | PIC18(L)F24K50 | PIC18(L)F25K50 | PIC18(L)F45K50 |
|---|--|--|---|
| Program Memory (Bytes) | 16384 | 32768 | 32768 |
| Program Memory (Instructions) | 8192 | 16384 | 16384 |
| Data Memory (Bytes) | 2048 | 2048 | 2048 |
| Data EEPROM Memory (Bytes) | 256 | 256 | 256 |
| I/O Ports | A, B, C, E ⁽¹⁾ | A, B, C, E ⁽¹⁾ | A, B, C, D, E |
| Capture/Compare/PWM Modules (CCP) | 1 | 1 | 1 |
| Enhanced CCP Modules (ECCP) | 1 | 1 | 1 |
| 10-bit Analog-to-Digital Module (ADC) | 3 internal 14 input | 3 internal 14 input | 3 internal 25 input |
| Packages | 28-pin SPDIP 28-pin SOIC 28-pin SSOP 28-pin QFN | 28-pin SPDIP 28-pin SOIC 28-pin SSOP 28-pin QFN | 40-pin PDIP 40-pin UQFN 44-pin TQFP |
| Interrupt Sources | 25 | | |
| Timers (16-bit) | 2 | | |
| Serial Communications | MSSP, EUSART | | |
| SR Latch | Yes | | |
| Charge Time Measurement Unit Module (CTMU) | Yes | | |
| Programmable High/Low-Voltage Detect (HLVD) | Yes | | |
| Programmable Brown-out Reset (BOR) | Yes | | |
| Resets (and Delays) | POR, BOR, LPBOR RESET Instruction, Stack Overflow, Stack Underflow (PWRT, OST), MCLR, WDT | | |
| Instruction Set | 75 Instructions; 83 with Extended Instruction Set enabled | | |
| Operating Frequency | DC – 48 MHz | | |

Note 1: PORTE contains the single RE3 read-only bit.

FIGURE 1-1: PIC18(L)F2X/45K50 FAMILY BLOCK DIAGRAM



PIC18(L)F2X/45K50

TABLE 1-2: PIC18(L)F2XK50 PINOUT I/O DESCRIPTIONS

| Pin Number | | Pin Name | Pin Type | Buffer Type | Description |
|-------------------------|-----|------------------------------|----------|-------------|--|
| SPDIP, SOIC, SSOP | QFN | | | | |
| 2 | 27 | RA0/C12IN0-/AN0 | | | |
| | | RA0 | I/O | TTL/DIG | Digital I/O. |
| | | C12IN0- | I | Analog | Comparators C1 and C2 inverting input. |
| 3 | 28 | AN0 | I | Analog | Analog input 0. |
| | | RA1/C12IN1-/AN1 | | | |
| | | RA1 | I/O | TTL/DIG | Digital I/O. |
| 4 | 1 | C12IN1- | I | Analog | Comparators C1 and C2 inverting input. |
| | | AN1 | I | Analog | Analog input 1. |
| | | RA2/C2IN+/AN2/DACOUT/VREF- | | | |
| | | RA2 | I/O | TTL/DIG | Digital I/O. |
| | | C2IN+ | I | Analog | Comparator C2 non-inverting input. |
| 5 | 2 | AN2 | I | Analog | Analog input 2. |
| | | DACOUT | O | Analog | DAC Reference output. |
| | | VREF- | I | Analog | A/D reference voltage (low) input. |
| | | RA3/C1IN+/AN3/VREF+ | | | |
| 6 | 3 | RA3 | I/O | TTL/DIG | Digital I/O. |
| | | C1IN+ | I | Analog | Comparator C1 non-inverting input. |
| | | AN3 | I | Analog | Analog input 3. |
| | | VREF+ | I | Analog | A/D reference voltage (high) input. |
| 7 | 4 | RA4/C1OUT/SRQ/T0CKI | | | |
| | | RA4 | I/O | ST/DIG | Digital I/O. |
| | | C1OUT | O | DIG | Comparator C1 output. |
| | | SRQ | O | DIG | SR latch Q output. |
| | | T0CKI | I | ST | Timer0 external clock input. |
| 10 | 7 | RA5/C2OUT/SRNQ/SS/HLDVIN/AN4 | | | |
| | | RA5 | I/O | TTL/DIG | Digital I/O. |
| | | C2OUT | O | DIG | Comparator C2 output. |
| | | SRNQ | O | DIG | SR latch \bar{Q} output. |
| | | SS | I | TTL | SPI slave select input (MSSP). |
| | | HLDVIN | I | Analog | High/Low-Voltage Detect input. |
| | | AN4 | I | Analog | Analog input 4. |
| | | RA6/CLKO/OSC2 | | | |
| | | RA6 | I/O | TTL/DIG | Digital I/O. |
| | | CLKO | O | DIG | Outputs 1/4 the frequency of OSC1 and denotes the instruction cycle rate. |
| | | OSC2 | O | — | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator modes. |

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

Note 1: Default pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are set.
2: Alternate pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are clear.

PIC18(L)F2X/45K50

TABLE 1-2: PIC18(L)F2XK50 PINOUT I/O DESCRIPTIONS (CONTINUED)

| Pin Number | | Pin Name | Pin Type | Buffer Type | Description |
|-------------------------|-----|-----------------------------------|----------|-------------------|---|
| SPDIP, SOIC, SSOP | QFN | | | | |
| 9 | 6 | RA7/CLKI/OSC1 | | | |
| | | RA7 | I/O | TTL/DIG | Digital I/O. |
| | | CLKI | I | CMOS | External clock source input. Always associated with pin function OSC1. |
| | | OSC1 | I | ST | Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise. |
| 21 | 18 | RB0/INT0/FLT0/SRI/SDI/SDA/AN12 | | | |
| | | RB0 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | INT0 | I | ST | External interrupt 0. |
| | | FLT0 | I | ST | PWM Fault input for ECCP auto-shutdown. |
| | | SRI | I | ST | SR latch input. |
| | | SDI | I | ST | SPI data in (MSSP). |
| | | SDA | I/O | I ² C™ | I ² C data I/O (MSSP). |
| | | AN12 | I | Analog | Analog input 12. |
| 22 | 19 | RB1/INT1/P1C/SCK/SCL/C12IN3-/AN10 | | | |
| | | RB1 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | INT1 | I | ST | External interrupt 1. |
| | | P1C | O | DIG | Enhanced CCP1 PWM output. |
| | | SCK | I/O | ST/DIG | Synchronous serial clock input/output for SPI mode (MSSP). |
| | | SCL | I/O | I ² C | Synchronous serial clock input/output for I ² C mode (MSSP). |
| | | C12IN3- | I | Analog | Comparators C1 and C2 inverting input. |
| | | AN10 | I | Analog | Analog input 10. |
| 23 | 20 | RB2/INT2/CTED1/P1B/AN8 | | | |
| | | RB2 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | INT2 | I | ST | External interrupt 2. |
| | | CTED1 | I | ST | CTMU Edge 1 input. |
| | | P1B | O | DIG | Enhanced CCP1 PWM output. |
| | | AN8 | I | Analog | Analog input 8. |
| 24 | 21 | RB3/CTED2/CCP2/SDO/C12IN2-/AN9 | | | |
| | | RB3 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | CTED2 | I | ST | CTMU Edge 2 input. |
| | | CCP2 ⁽²⁾ | I/O | ST/DIG | Alternate Capture 2 input/Compare 2 output/PWM 2 output. |
| | | SDO ⁽¹⁾ | O | DIG | SPI data out (MSSP). |
| | | C12IN2- | I | Analog | Comparators C1 and C2 inverting input. |
| | | AN9 | I | Analog | Analog input 9. |

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

Note 1: Default pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are set.

2: Alternate pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are clear.

PIC18(L)F2X/45K50

TABLE 1-2: PIC18(L)F2XK50 PINOUT I/O DESCRIPTIONS (CONTINUED)

| Pin Number | | Pin Name | Pin Type | Buffer Type | Description |
|-------------------------|-----|---------------------------------|----------|-------------|---|
| SPDIP, SOIC, SSOP | QFN | | | | |
| 25 | 22 | RB4/IOCB4/P1D/AN11 | | | |
| | | RB4 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | IOCB4 | I | TTL | Interrupt-on-change pin. |
| | | P1D | O | DIG | Enhanced CCP1 PWM output. |
| 26 | 23 | AN11 | I | Analog | Analog input 11. |
| | | RB5/IOCB5/T3CKI/T1G/AN13 | | | |
| | | RB5 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | IOCB5 | I | TTL | Interrupt-on-change pin. |
| | | T3CKI ⁽²⁾ | I | ST | Alternate Timer3 clock input. |
| 27 | 24 | T1G | I | ST | Timer1 external clock gate input. |
| | | AN13 | I | Analog | Analog input 13. |
| | | RB6/IOCB6/PGC | | | |
| 28 | 25 | RB6 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | IOCB6 | I | TTL | Interrupt-on-change pin. |
| | | PGC | I/O | ST | In-Circuit Debugger and ICSP™ programming clock pin. |
| 11 | 8 | RB7/IOCB7/PGD | | | |
| | | RB7 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | IOCB7 | I | TTL | Interrupt-on-change pin. |
| | | PGD | I/O | ST/DIG | In-Circuit Debugger and ICSP™ programming data pin. |
| | | RC0/IOCC0/T3CKI/T3G/T1CKI/SOSCO | | | |
| | | RC0 | I/O | ST/DIG | Digital I/O. |
| 12 | 9 | IOCC0 | I | TTL | Interrupt-on-change pin. |
| | | T3CKI ⁽¹⁾ | I | ST | Timer3 clock input. |
| | | T3G | I | ST | Timer3 external clock gate input. |
| | | T1CKI | I | ST | Timer1 clock input. |
| | | SOSCO | O | — | Secondary oscillator output. |
| 12 | 9 | RC1/IOCC1/CCP2/SOSCI | | | |
| | | RC1 | I/O | ST/DIG | Digital I/O. |
| | | IOCC1 | I | TTL | Interrupt-on-change pin. |
| | | CCP2 ⁽¹⁾ | I/O | ST/DIG | Capture 2 input/Compare 2 output/PWM 2 output. |
| | | SOSCI | I | Analog | Secondary oscillator input. |

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

Note 1: Default pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are set.

2: Alternate pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are clear.

PIC18(L)F2X/45K50

TABLE 1-2: PIC18(L)F2XK50 PINOUT I/O DESCRIPTIONS (CONTINUED)

| Pin Number | | Pin Name | Pin Type | Buffer Type | Description |
|-------------------------|-------|-------------------------------|----------|-------------|--|
| SPDIP, SOIC, SSOP | QFN | | | | |
| 13 | 10 | RC2/CTPLS/P1A/CCP1/IOCC2/AN14 | | | |
| | | RC2 | I/O | ST/DIG | Digital I/O. |
| | | CTPLS | O | DIG | CTMU pulse generator output. |
| | | P1A | O | DIG | Enhanced CCP1 PWM output. |
| | | CCP1 | I/O | ST/DIG | Capture 1 input/Compare 1 output/PWM 1 output. |
| | | IOCC2 | I | TTL | Interrupt-on-change pin. |
| | | AN14 | I | Analog | Analog input 14. |
| 14 | 11 | VUSB3V3 | | | |
| | | VUSB3V3 | P | — | Internal 3.3V voltage regulator output, positive supply for USB transceiver. |
| 15 | 12 | D-/IOCC4 | | | |
| | | D- IOCC4 | I/O I | — ST | USB differential minus line input/output. Interrupt-on-change pin. |
| 16 | 13 | D+/IOCC5 | | | |
| | | D+ IOCC5 | I/O I | — ST | USB differential plus line input/output. Interrupt-on-change pin. |
| 17 | 14 | RC6/IOCC6/TX/CK/AN18 | | | |
| | | RC6 | I/O | ST/DIG | Digital I/O. |
| | | IOCC6 | I | TTL | Interrupt-on-change pin. |
| | | TX | O | DIG | EUSART asynchronous transmit. |
| | | CK | I/O | ST | EUSART synchronous clock (see related RX/DT). |
| | | AN18 | I | Analog | Analog input 18. |
| 18 | 15 | RC7/SDO/IOCC7/RX/DT/AN19 | | | |
| | | RC7 | I/O | ST/DIG | Digital I/O. |
| | | SDO ⁽²⁾ | O | DIG | Alternate SPI data out pin assignment (MSSP). |
| | | IOCC7 | I | TTL | Interrupt-on-change pin. |
| | | RX | I | ST | EUSART asynchronous receive. |
| | | DT | I/O | ST/DIG | EUSART synchronous data (see related TX/CK). |
| | | AN19 | I | Analog | Analog input 19. |
| 1 | 26 | RE3/VPP/MCLR | | | |
| | | RE3 | I | ST | Digital input. |
| | | VPP | P | — | Programming voltage input. |
| | | MCLR | I | ST | Active-Low Master Clear (device Reset) input. |
| 20 | 17 | VDD | P | — | Positive supply for logic and I/O pins. |
| 8, 19 | 5, 16 | Vss | P | — | Ground reference for logic and I/O pins. |

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

Note 1: Default pin assignment for SDO, T3CK1 and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are set.

2: Alternate pin assignment for SDO, T3CK1 and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are clear.

PIC18(L)F2X/45K50

TABLE 1-3: PIC18(L)F45K50 PINOUT I/O DESCRIPTIONS

| Pin Number | | | Pin Name | Pin Type | Buffer Type | Description |
|------------|------|------|-----------------------------|----------|-------------|---|
| PDIP | TQFP | UQFN | | | | |
| 2 | 19 | 17 | RA0/C12IN0-/AN0 | | | Digital I/O. Comparators C1 and C2 inverting input. Analog input 0. |
| | | | RA0 | I/O | TTL/DIG | |
| | | | C12IN0- | I | Analog | |
| 3 | 20 | 18 | RA1/C12IN1-/AN1 | | | Digital I/O. Comparators C1 and C2 inverting input. Analog input 1. |
| | | | RA1 | I/O | TTL/DIG | |
| | | | C12IN1- | I | Analog | |
| 4 | 21 | 19 | RA2/C2IN+/AN2/DACOUT/VREF- | | | Digital I/O. Comparator C2 non-inverting input. Analog input 2. DAC Reference output. A/D reference voltage (low) input. |
| | | | RA2 | I/O | TTL/DIG | |
| | | | C2IN+ | I | Analog | |
| | | | AN2 | I | Analog | |
| | | | DACOUT | O | Analog | |
| 5 | 22 | 20 | RA3/C1IN+/AN3/VREF+ | | | Digital I/O. Comparator C1 non-inverting input. Analog input 3. A/D reference voltage (high) input. |
| | | | RA3 | I/O | TTL/DIG | |
| | | | C1IN+ | I | Analog | |
| | | | AN3 | I | Analog | |
| 6 | 23 | 21 | RA4/C1OUT/SRQ/T0CKI | | | Digital I/O. Comparator C1 output. SR latch Q output. Timer0 external clock input. |
| | | | RA4 | I/O | ST/DIG | |
| | | | C1OUT | O | DIG | |
| | | | SRQ | O | TTL | |
| 7 | 24 | 22 | RA5/C2OUT/SRNQ/SS/HLDIN/AN4 | | | Digital I/O. Comparator C2 output. SR latch \bar{Q} output. SPI slave select input (MSSP). High/Low-Voltage Detect input. Analog input 4. |
| | | | RA5 | I/O | TTL/DIG | |
| | | | C2OUT | O | DIG | |
| | | | SRNQ | O | DIG | |
| | | | SS | I | TTL | |
| | | | HLDIN | I | Analog | |
| 14 | 31 | 29 | RA6/CLKO/OSC2 | | | Digital I/O. Outputs 1/4 the frequency of OSC1 and denotes the instruction cycle rate. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| | | | RA6 | I/O | TTL/DIG | |
| | | | CLKO | O | DIG | |
| | | | OSC2 | O | — | |
| 13 | 30 | 28 | RA7/CLKI/OSC1 | | | Digital I/O. External clock source input. Always associated with pin function OSC1. Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise. |
| | | | RA7 | I/O | TTL/DIG | |
| | | | CLKI | I | CMOS | |
| | | | OSC1 | I | ST | |

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

- Note**
- 1: Default pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are set.
 - 2: Alternate pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are clear.
 - 3: Pin is "No Connect", except on PIC18(L)F45K50 TQFP devices with ICPRT Configuration bit set.

PIC18(L)F2X/45K50

TABLE 1-3: PIC18(L)F45K50 PINOUT I/O DESCRIPTIONS (CONTINUED)

| Pin Number | | | Pin Name | Pin Type | Buffer Type | Description |
|------------|------|------|-----------------------------------|----------|-------------------|---|
| PDIP | TQFP | UQFN | | | | |
| 33 | 8 | 8 | RB0/INT0/FLT0/SDI/SDA/SRI/AN12 | | | |
| | | | RB0 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | | INT0 | I | ST | External interrupt 0. |
| | | | FLT0 | I | ST | PWM Fault input for ECCP auto-shutdown. |
| | | | SDI | I | ST | SPI Data in (MSSP). |
| | | | SDA | I/O | I ² C™ | I ² C Data I/O (MSSP). |
| | | | SRI | I | ST | SR latch input. |
| 34 | 9 | 9 | RB1/INT1/P1C/SCK/SCL/C12IN3-/AN10 | | | |
| | | | RB1 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | | INT1 | I | ST | External interrupt 1. |
| | | | P1C | O | DIG | Enhanced CCP1 PWM output. |
| | | | SCK | I/O | ST/DIG | Synchronous serial clock input/output for SPI mode (MSSP). |
| | | | SCL | I/O | I ² C | Synchronous serial clock input/output for I ² C mode (MSSP). |
| | | | C12IN3- | I | Analog | Comparators C1 and C2 inverting input. |
| 35 | 10 | 10 | RB2/P1B/INT2/CTED1/AN8 | | | |
| | | | RB2 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | | P1B | O | DIG | Enhanced CCP1 PWM output. |
| | | | INT2 | I | ST | External interrupt 2. |
| | | | CTED1 | I | ST | CTMU Edge 1 input. |
| 36 | 11 | 11 | RB3/CTED2/SDO/CCP2/C12IN2-/AN9 | | | |
| | | | RB3 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | | CTED2 | I | ST | CTMU Edge 2 input. |
| | | | SDO ⁽¹⁾ | O | DIG | SPI Data out (MSSP). |
| | | | CCP2 ⁽²⁾ | I/O | ST | Alternate Capture 2 input/Compare 2 output/PWM 2 output. |
| | | | C12IN2- | I | Analog | Comparators C1 and C2 inverting input. |
| 37 | 14 | 12 | RB4/IOCB4/P1D/AN11 | | | |
| | | | RB4 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | | IOCB4 | I | TTL | Interrupt-on-change pin. |
| | | | P1D | O | DIG | Enhanced CCP1 PWM output. |
| 38 | 15 | 13 | RB5/IOCB5/T3CKI/T1G/AN13 | | | |
| | | | RB5 | I/O | TTL/DIG | Digital Output or Input with internal pull-up option. |
| | | | IOCB5 | I | TTL | Interrupt-on-change pin. |
| | | | T3CKI ⁽²⁾ | I | ST | Alternate Timer3 clock input. |
| | | | T1G | I | ST | Timer1 external clock gate input. |
| | | | AN13 | I | Analog | Analog input 13. |

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

- Note**
- 1: Default pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are set.
 - 2: Alternate pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are clear.
 - 3: Pin is “No Connect”, except on PIC18(L)F45K50 TQFP devices with ICPRT Configuration bit set.

PIC18(L)F2X/45K50

TABLE 1-3: PIC18(L)F45K50 PINOUT I/O DESCRIPTIONS (CONTINUED)

| Pin Number | | | Pin Name | Pin Type | Buffer Type | Description |
|------------|------|------|---------------------------------|----------|-------------|---|
| PDIP | TQFP | UQFN | | | | |
| 39 | 16 | 14 | RB6/IOCB6/PGC | | | Digital Output or Input with internal pull-up option. Interrupt-on-change pin. In-Circuit Debugger and ICSP™ programming clock pin. |
| | | | RB6 | I/O | TTL/DIG | |
| | | | IOCB6 | I | TTL | |
| 40 | 17 | 15 | RB7/IOCB7/PGD | | | Digital Output or Input with internal pull-up option. Interrupt-on-change pin. In-Circuit Debugger and ICSP™ programming data pin. |
| | | | RB7 | I/O | TTL/DIG | |
| | | | IOCB7 | I | TTL | |
| 15 | 32 | 30 | RC0/IOCC0/T3CKI/T3G/T1CKI/SOSCO | | | Digital I/O. Interrupt-on-change pin. Timer3 clock input. Timer3 external clock gate input. Timer1 clock input. Secondary oscillator output. |
| | | | RC0 | I/O | ST/DIG | |
| | | | IOCC0 | I | TTL | |
| | | | T3CKI ⁽¹⁾ | I | ST | |
| | | | T3G | I | ST | |
| | | | T1CKI | I | ST | |
| 16 | 35 | 31 | RC1/IOCC1/CCP2/SOSCI | | | Digital I/O. Interrupt-on-change pin. Capture 2 input/Compare 2 output/PWM 2 output. Secondary oscillator input. |
| | | | RC1 | I/O | ST/DIG | |
| | | | IOCC1 | I | TTL | |
| | | | CCP2 ⁽¹⁾ | I/O | ST/DIG | |
| 17 | 36 | 32 | RC2/CTPLS/P1A/CCP1/IOCC2/AN14 | | | Digital I/O. CTMU pulse generator output. Enhanced CCP1 PWM output. Capture 1 input/Compare 1 output/PWM 1 output. Interrupt-on-change pin. Analog input 14. |
| | | | RC2 | I/O | ST/DIG | |
| | | | CTPLS | O | DIG | |
| | | | P1A | O | DIG | |
| | | | CCP1 | I/O | ST/DIG | |
| | | | IOCC2 | I | TTL | |
| 18 | 37 | 33 | VUSB3V3 | | | Internal 3.3V voltage regulator output, positive supply for USB transceiver. |
| | | | VUSB3V3 | P | — | |
| 23 | 42 | 38 | D-/IOCC4 | | | USB differential minus line input/output. Interrupt-on-change pin. |
| | | | D- | I/O | — | |
| 24 | 43 | 39 | D+/IOCC5 | | | USB differential plus line input/output. Interrupt-on-change pin. |
| | | | D+ | I/O | — | |
| | | | IOCC5 | I | ST | |

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

- Note**
- 1: Default pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are set.
 - 2: Alternate pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are clear.
 - 3: Pin is “No Connect”, except on PIC18(L)F45K50 TQFP devices with ICPRT Configuration bit set.

PIC18(L)F2X/45K50

TABLE 1-3: PIC18(L)F45K50 PINOUT I/O DESCRIPTIONS (CONTINUED)

| Pin Number | | | Pin Name | Pin Type | Buffer Type | Description |
|------------|------|------|--------------------------|----------|-------------|---|
| PDIP | TQFP | UQFN | | | | |
| 25 | 44 | 40 | RC6/IOCC6/TX/CK/AN18 | | | |
| | | | RC6 | I/O | ST/DIG | Digital I/O. |
| | | | IOCC6 | I | TTL | Interrupt-on-change pin. |
| | | | TX | O | — | EUSART asynchronous transmit. |
| | | | CK | I/O | ST | EUSART synchronous clock (see related RX/DT). |
| | | | AN18 | I | Analog | Analog input 18. |
| 26 | 1 | 1 | RC7/RX/DT/SDO/IOCC7/AN19 | | | |
| | | | RC7 | I/O | ST/DIG | Digital I/O. |
| | | | RX | I | ST | EUSART asynchronous receive. |
| | | | DT | I/O | ST | EUSART synchronous data (see related TX/CK). |
| | | | SDO ⁽²⁾ | O | DIG | Alternate SPI data out (MSSP). |
| | | | IOCC7 | I | TTL | Interrupt-on-change pin. |
| | | | AN19 | I | Analog | Analog input 19. |
| 19 | 38 | 34 | RD0/AN20 | | | |
| | | | RD0 | I/O | ST/DIG | Digital I/O. |
| | | | AN20 | I | Analog | Analog input 20. |
| 20 | 39 | 35 | RD1/AN21 | | | |
| | | | RD1 | I/O | ST/DIG | Digital I/O. |
| | | | AN21 | I | Analog | Analog input 21. |
| 21 | 40 | 36 | RD2/AN22 | | | |
| | | | RD2 | I/O | ST/DIG | Digital I/O |
| | | | AN22 | I | Analog | Analog input 22. |
| 22 | 41 | 37 | RD3/AN23 | | | |
| | | | RD3 | I/O | ST/DIG | Digital I/O. |
| | | | AN23 | I | Analog | Analog input 23. |
| 27 | 2 | 2 | RD4/AN24 | | | |
| | | | RD4 | I/O | ST/DIG | Digital I/O. |
| | | | AN24 | I | Analog | Analog input 24. |
| 28 | 3 | 3 | RD5/P1B/AN25 | | | |
| | | | RD5 | I/O | ST/DIG | Digital I/O. |
| | | | P1B | O | DIG | Enhanced CCP1 PWM output. |
| | | | AN25 | I | Analog | Analog input 25. |
| 29 | 4 | 4 | RD6/P1C/AN26 | | | |
| | | | RD6 | I/O | ST/DIG | Digital I/O. |
| | | | P1C | O | DIG | Enhanced CCP1 PWM output. |
| | | | AN26 | I | Analog | Analog input 26. |
| 30 | 5 | 5 | RD7/P1D/AN27 | | | |
| | | | RD7 | I/O | ST/DIG | Digital I/O. |
| | | | P1D | O | DIG | Enhanced CCP1 PWM output. |
| | | | AN27 | I | Analog | Analog input 27. |

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

- Note 1:** Default pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are set.
- Note 2:** Alternate pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are clear.
- Note 3:** Pin is “No Connect”, except on PIC18(L)F45K50 TQFP devices with ICPRT Configuration bit set.

PIC18(L)F2X/45K50

TABLE 1-3: PIC18(L)F45K50 PINOUT I/O DESCRIPTIONS (CONTINUED)

| Pin Number | | | Pin Name | Pin Type | Buffer Type | Description |
|------------|-------|-------|------------------------------|-------------|------------------|---|
| PDIP | TQFP | UQFN | | | | |
| 8 | 25 | 23 | RE0/AN5 | | | |
| | | | RE0 AN5 | I/O I | ST/DIG Analog | Digital I/O. Analog input 5. |
| 9 | 26 | 24 | RE1/AN6 | | | |
| | | | RE1 AN6 | I/O I | ST/DIG Analog | Digital I/O. Analog input 6. |
| 10 | 27 | 25 | RE2/AN7 | | | |
| | | | RE2 AN7 | I/O I | ST Analog | Digital I/O. Analog input 7. |
| 1 | 18 | 16 | RE3/VPP/MCLR | | | |
| | | | RE3 VPP MCLR | I P I | ST ST | Digital input. Programming voltage input. Active-low Master Clear (device Reset) input. |
| — | 12 | — | ICCK/ICPGC | | | |
| | | | ICCK ICPGC ⁽³⁾ | I/O | ST | Dedicated In-Circuit Debugger clock. |
| — | 13 | — | ICDT/ICPGD | I/O | ST | Dedicated ICSP™ programming clock. |
| | | | | I/O | ST | Dedicated In-Circuit Debugger data. |
| — | 33 | — | ICRST/ICVPP | | | |
| | | | | I I | ST P | Dedicated Master Clear Reset input. Dedicated programming voltage input. |
| 11,32 | 7, 28 | 7, 26 | VDD | P | — | Positive supply for logic and I/O pins. |
| 12,31 | 6, 29 | 6, 27 | Vss | P | — | Ground reference for logic and I/O pins. |
| | 34 | | NC | | | |

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

- Note**
- 1: Default pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are set.
 - 2: Alternate pin assignment for SDO, T3CKI and CCP2 when Configuration bits SDOMX, T3CMX and CCP2MX are clear.
 - 3: Pin is “No Connect”, except on PIC18(L)F45K50 TQFP devices with ICPRT Configuration bit set.

2.0 GUIDELINES FOR GETTING STARTED WITH PIC18(L)F2X/45K50 MICROCONTROLLERS

2.1 Basic Connection Requirements

Getting started with the PIC18(L)F2X/45K50 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All VDD and Vss pins
(see [Section 2.2 “Power Supply Pins”](#))
- MCLR pin
(see [Section 2.3 “Master Clear \(MCLR\) Pin”](#))
- VUSB3V3 pins
(see [Section 2.4 “Voltage Regulator Pins \(V_{USB3V3}\)”](#))

These pins must also be connected if they are being used in the end application:

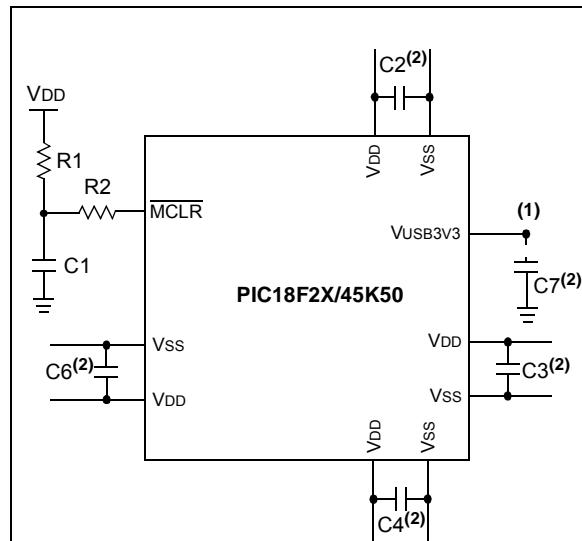
- PGC/PGD pins used for In-Circuit Serial Programming (ICSP) and debugging purposes (see [Section 2.5 “ICSP Pins”](#))
- OSC1 and OSC2 pins when an external oscillator source is used
(see [Section 2.6 “External Oscillator Pins”](#))

Additionally, the following pins may be required:

- VREF+/VREF- pins are used when external voltage reference for analog modules is implemented

The minimum mandatory connections are shown in [Figure 2-1](#).

FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS



Key (all values are recommendations):

C1 through C6: 0.1 µF, 20V ceramic

R1: 10 kΩ

R2: 100Ω to 470Ω

Note 1: See [Section 2.4 “Voltage Regulator Pins \(V_{USB3V3}\)”](#) for explanation of V_{USB3V3} pin connections.

2: The example shown is for a PIC18F device with five VDD/VSS pairs. Other devices may have more or less pairs; adjust the number of decoupling capacitors appropriately.

PIC18(L)F2X/45K50

2.2 Power Supply Pins

2.2.1 DECOUPLING CAPACITORS

The use of decoupling capacitors on every pair of power supply pins, such as VDD and Vss is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** A $0.1\ \mu\text{F}$ (100 nF), 10-20V capacitor is recommended. The capacitor should be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the printed circuit board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of $0.01\ \mu\text{F}$ to $0.001\ \mu\text{F}$. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., $0.1\ \mu\text{F}$ in parallel with $0.001\ \mu\text{F}$).
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

2.2.2 TANK CAPACITORS

On boards with power traces running longer than six inches in length, it is suggested to use a tank capacitor for integrated circuits, including microcontrollers, to supply a local power source. The value of the tank capacitor should be determined based on the trace resistance that connects the power supply source to the device, and the maximum current drawn by the device in the application. In other words, select the tank capacitor so that it meets the acceptable voltage sag at the device. Typical values range from $4.7\ \mu\text{F}$ to $47\ \mu\text{F}$.

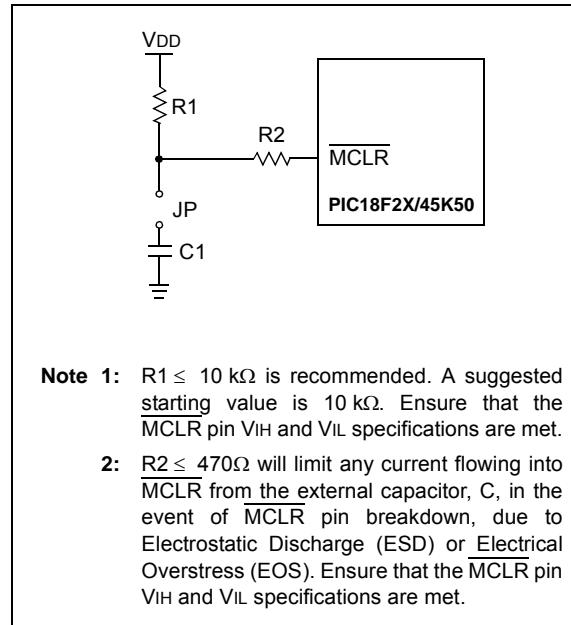
2.3 Master Clear (MCLR) Pin

The MCLR pin provides two specific device functions: Device Reset, and Device Programming and Debugging. If programming and debugging are not required in the end application, a direct connection to VDD may be all that is required. The addition of other components, to help increase the application's resistance to spurious Resets from voltage sags, may be beneficial. A typical configuration is shown in Figure 2-1. Other circuit designs may be implemented, depending on the application's requirements.

During programming and debugging, the resistance and capacitance that can be added to the pin must be considered. Device programmers and debuggers drive the MCLR pin. Consequently, specific voltage levels (VIH and VIL) and fast signal transitions must not be adversely affected. Therefore, specific values of $R1$ and $C1$ will need to be adjusted based on the application and PCB requirements. For example, it is recommended that the capacitor, $C1$, be isolated from the MCLR pin during programming and debugging operations by using a jumper (Figure 2-2). The jumper is replaced for normal run-time operations.

Any components associated with the MCLR pin should be placed within 0.25 inch (6 mm) of the pin.

FIGURE 2-2: EXAMPLE OF MCLR PIN CONNECTIONS



2.4 Voltage Regulator Pins (VUSB3V3)

The on-chip voltage regulator must always be connected directly to either a supply voltage or to an external capacitor.

When the regulator is enabled (F devices), a low-ESR capacitor is required on the VUSB3V3 pin to stabilize the voltage regulator output voltage. The VUSB3V3 pin must not be connected to VDD and is recommended to use a ceramic capacitor connected to ground. Refer to [Section 29.0 “Electrical Specifications”](#) for additional information.

It is recommended that the trace length not exceed 0.25 inch (6 mm). Refer to [Section 29.0 “Electrical Specifications”](#) for additional information.

When the regulator is disabled (LF devices), the VUSB3V3 pin should be externally tied to a voltage source maintained at the VDD level. Refer to [Section 29.0 “Electrical Specifications”](#) for information on VDD and VUSB3V3.

- LF devices (with the name PIC18LF2X/45K50) permanently disable the voltage regulator.

The VDD level of these devices must comply with the “voltage regulator disabled” specification for Parameter D001, in [Section 29.0 “Electrical Specifications”](#).

- F devices permanently enable the voltage regulator.

These devices require an external capacitor on the VUSB3V3 pin. Refer to [Section 29.0 “Electrical Specifications”](#) for additional information.

2.4.1 CONSIDERATIONS FOR CERAMIC CAPACITORS

In recent years, large value, low-voltage, surface-mount ceramic capacitors have become very cost effective in sizes up to a few tens of microfarad. The low-ESR, small physical size and other properties make ceramic capacitors very attractive in many types of applications.

Ceramic capacitors are suitable for use with the internal voltage regulator of this microcontroller. However, some care is needed in selecting the capacitor to ensure that it maintains sufficient capacitance over the intended operating range of the application.

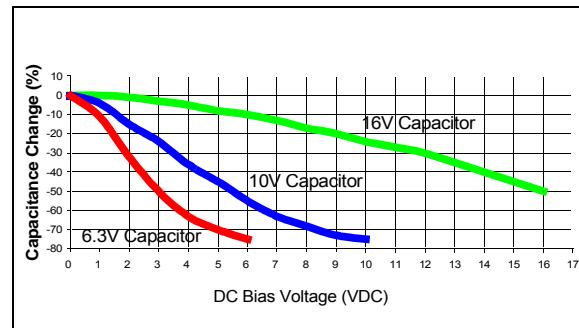
Typical low-cost, ceramic capacitors are available in X5R, X7R and Y5V dielectric ratings (other types are also available, but are less common). The initial tolerance specifications for these types of capacitors are often specified as $\pm 10\%$ to $\pm 20\%$ (X5R and X7R), or $-20\% +80\%$ (Y5V). However, the effective capacitance that these capacitors provide in an application circuit will also vary based on additional factors, such as the applied DC bias voltage and the temperature. The total in-circuit tolerance is, therefore, much wider than the initial tolerance specification.

The X5R and X7R capacitors typically exhibit satisfactory temperature stability (ex: $\pm 15\%$ over a wide temperature range, but consult the manufacturer's data sheets for exact specifications). However, Y5V capacitors typically have extreme temperature tolerance specifications of $+22\% -82\%$. Due to the extreme temperature tolerance, a $10\text{ }\mu\text{F}$ nominal rated Y5V type capacitor may not deliver enough total capacitance to meet minimum internal voltage regulator stability and transient response requirements. Therefore, Y5V capacitors are not recommended for use with the internal regulator if the application must operate over a wide temperature range.

In addition to temperature tolerance, the effective capacitance of large value ceramic capacitors can vary substantially, based on the amount of DC voltage applied to the capacitor. This effect can be very significant, but is often overlooked or is not always documented.

A typical DC bias voltage vs. capacitance graph for X7R type and Y5V type capacitors is shown in [Figure 2-3](#).

FIGURE 2-3: DC BIAS VOLTAGE vs. CAPACITANCE CHARACTERISTICS



When selecting a ceramic capacitor to be used with the internal voltage regulator, it is suggested to select a high-voltage rating, so that the operating voltage is a small percentage of the maximum rated capacitor voltage. For example, choose a ceramic capacitor rated at 16V for the 3.3V VUSB3V3 voltage.

2.5 ICSP Pins

The PGC and PGD pins are used for In-Circuit Serial Programming (ICSP) and debugging purposes. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor is recommended, with the value in the range of a few tens of ohms, not to exceed 100Ω .

Pull-up resistors, series diodes and capacitors on the PGC and PGD pins are not recommended as they will interfere with the programmer/debugger communications to the device. If such discrete components are an application requirement, they should be removed from the circuit during programming and debugging. Alternatively, refer to the AC/DC characteristics and timing requirements information in the respective device Flash programming specification for information on capacitive loading limits, and pin input voltage high (V_{IH}) and input low (V_{IL}) requirements.

For device emulation, ensure that the “Communication Channel Select” (i.e., PGCx/PGDx pins), programmed into the device, matches the physical connections for the ICSP to the Microchip debugger/emulator tool.

For more information on available Microchip development tools connection requirements, refer to [Section 28.0 “Development Support”](#).

2.6 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to [Section 3.0 “Oscillator Module \(With Fail-Safe Clock Monitor\)”](#) for details).

The oscillator circuit should be placed on the same side of the board as the device. Place the oscillator circuit close to the respective oscillator pins with no more than 0.5 inch (12 mm) between the circuit components and the pins. The load capacitors should be placed next to the oscillator itself, on the same side of the board.

Use a grounded copper pour around the oscillator circuit to isolate it from surrounding circuits. The grounded copper pour should be routed directly to the MCU ground. Do not run any signal traces or power traces inside the ground pour. Also, if using a two-sided board, avoid any traces on the other side of the board where the crystal is placed.

Layout suggestions are shown in Figure 2-4. In-line packages may be handled with a single-sided layout that completely encompasses the oscillator pins. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

In planning the application’s routing and I/O assignments, ensure that adjacent port pins, and other signals in close proximity to the oscillator, are benign (i.e., free of high frequencies, short rise and fall times, and other similar noise).

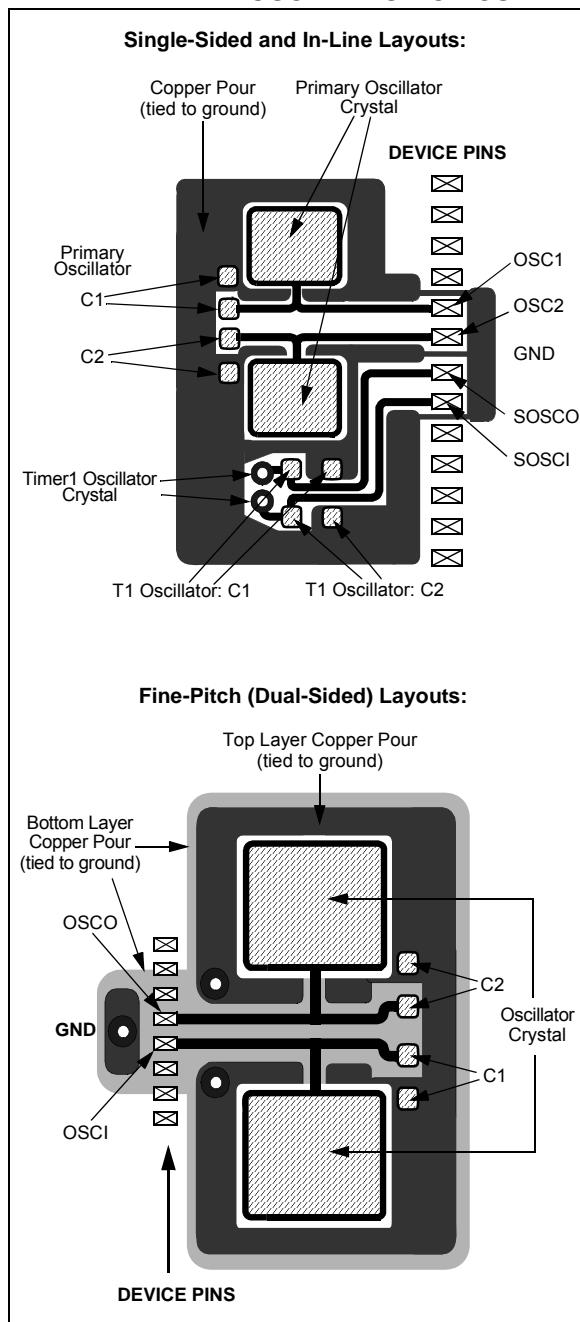
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate web site (www.microchip.com):

- AN826, “*Crystal Oscillator Basics and Crystal Selection for rfPIC™ and PICmicro® Devices*”
- AN849, “*Basic PICmicro® Oscillator Design*”
- AN943, “*Practical PICmicro® Oscillator Analysis and Design*”
- AN949, “*Making Your Oscillator Work*”

2.7 Unused I/Os

Unused I/O pins should be configured as outputs and driven to a logic low state. Alternatively, connect a $1\text{ k}\Omega$ to $10\text{ k}\Omega$ resistor to Vss on unused pins and drive the output to logic low.

FIGURE 2-4: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT



PIC18(L)F2X/45K50

3.0 OSCILLATOR MODULE (WITH FAIL-SAFE CLOCK MONITOR)

3.1 Overview

The oscillator module has a wide variety of clock sources and selection features that allow it to be used in a wide range of applications while maximizing performance and minimizing power consumption. [Figure 3-1](#) illustrates a block diagram of the oscillator module.

Clock sources can be configured from external oscillators, quartz crystal resonators, ceramic resonators and Resistor-Capacitor (RC) circuits. In addition, the system clock source can be configured from one of three internal oscillators, with a choice of speeds selectable via software. Additional clock features include:

- Selectable system clock source between external or internal sources via software.
- Two-Speed Start-up mode, which minimizes latency between external oscillator start-up and code execution.
- Fail-Safe Clock Monitor (FSCM) designed to detect a failure of the external clock source (LP, XT, HS, EC or RC modes) and switch automatically to the internal oscillator.
- Oscillator Start-up Timer (OST) ensures stability of crystal oscillator sources.

The primary clock module can be configured to provide one of six clock sources as the primary clock.

1. RC External Resistor/Capacitor
2. LP Low-Power Crystal
3. XT Crystal/Resonator
4. INTOSC Internal Oscillator
5. HS High-Speed Crystal/Resonator
6. EC External Clock

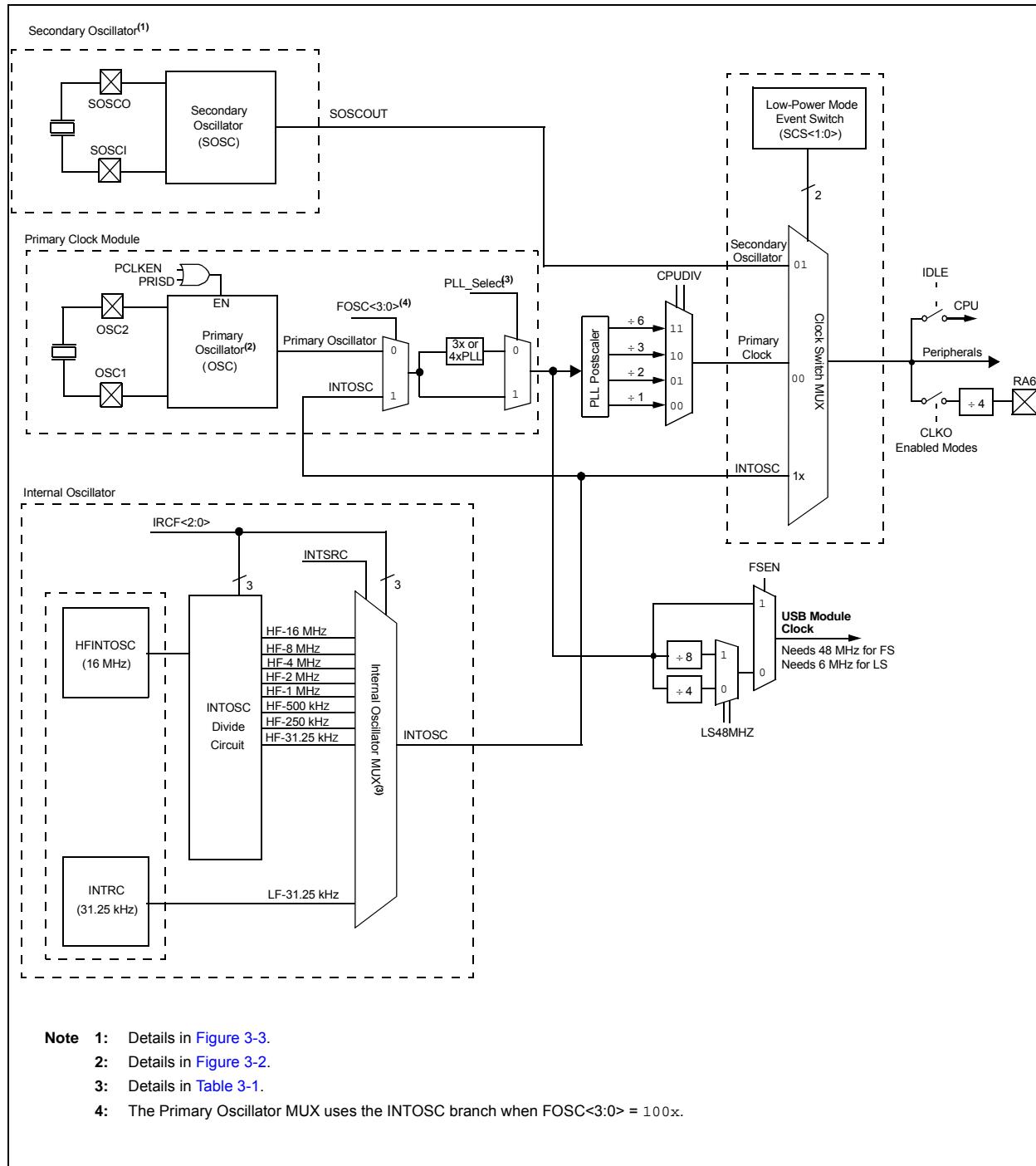
The HS and EC oscillator circuits can be optimized for power consumption and oscillator speed using settings in FOSC<3:0>. Additional FOSC<3:0> selections enable RA6 to be used as I/O or CLKO (Fosc/4) for RC, EC and INTOSC Oscillator modes.

Primary clock modes are selectable by the FOSC<3:0> bits of the CONFIG1H Configuration register. The primary clock operation is further defined by these Configuration and register bits:

1. PCLKEN (CONFIG1H<5>)
2. PRISD (OSCCON2<2>)
3. CFGPLLLEN (CONFIG1L<1>)
4. PLLLEN (OSCCON2<4>)
5. IRCF<2:0> (OSCCON<6:4>)
6. INTSRC (OSCCON2<5>)

The HFINTOSC and INTRC are factory calibrated high and low-frequency oscillators, respectively, which are used as the internal clock sources.

FIGURE 3-1: SIMPLIFIED OSCILLATOR SYSTEM BLOCK DIAGRAM



3.2 Oscillator Control

The OSCCON, OSCCON2 and OSCTUNE registers ([Register 3-1](#) to [Register 3-3](#)) control several aspects of the device clock's operation, both in full-power operation and in power-managed modes.

- Main System Clock Selection (SCS)
- Primary Oscillator Circuit Shutdown (PRISD)
- Secondary Oscillator Enable (SOSCGO)
- Primary Clock Frequency multiplier (PLLEN)
- Internal Frequency selection bits (IRCF, INTSRC)
- Clock Status bits (OSTS, HFIOFS, LFIOFS, SOSCRUN, PLLRDY)
- Power management selection (IDLEN)

3.2.1 MAIN SYSTEM CLOCK SELECTION

The System Clock Select bits, SCS<1:0>, select the main clock source. The available clock sources are:

- Primary clock defined by the FOSC<3:0> bits of CONFIG1H. The primary clock can be the primary oscillator, an external clock, or the internal oscillator block.
- Secondary clock (secondary oscillator)
- Internal oscillator block (HFINTOSC and INTRC).

The clock source changes immediately after one or more of the bits is written to, following a brief clock transition interval. The SCS bits are cleared to select the primary clock on all forms of Reset.

3.2.2 INTERNAL FREQUENCY SELECTION

The Internal Oscillator Frequency Select bits (IRCF<2:0>) select the frequency output of the internal oscillator block. The choices are the INTRC source (31.25 kHz) and the HFINTOSC source (16 MHz) or one of the frequencies derived from the HFINTOSC postscaler (31.25 kHz to 16 MHz). If the internal oscillator block is supplying the main clock, changing the states of these bits will have an immediate change on the internal oscillator's output. On device Resets, the output frequency of the internal oscillator is set to the default frequency of 1 MHz.

3.2.3 LOW-FREQUENCY SELECTION

When a nominal output frequency of 31.25 kHz is selected (IRCF<2:0> = 000), users may choose which internal oscillator acts as the source. This is done with the INTSRC bit of the OSCCON2<5> register. See [Figure 3-2](#) and [Register 3-1](#) for specific 31.25 kHz selection. This option allows users to select a 31.25 kHz clock (based on HFINTOSC) that can be tuned using the TUN<6:0> bits in the OSCTUNE register, while maintaining power savings with a very low clock speed. INTRC always remains the clock source for features such as the Watchdog Timer and the Fail-Safe Clock Monitor, regardless of the setting of the INTSRC bit.

This option allows users to select the tunable and more precise HFINTOSC as a clock source, while maintaining power savings with a very low clock speed.

3.2.4 POWER MANAGEMENT

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

FIGURE 3-2: INTERNAL OSCILLATOR MUX BLOCK DIAGRAM

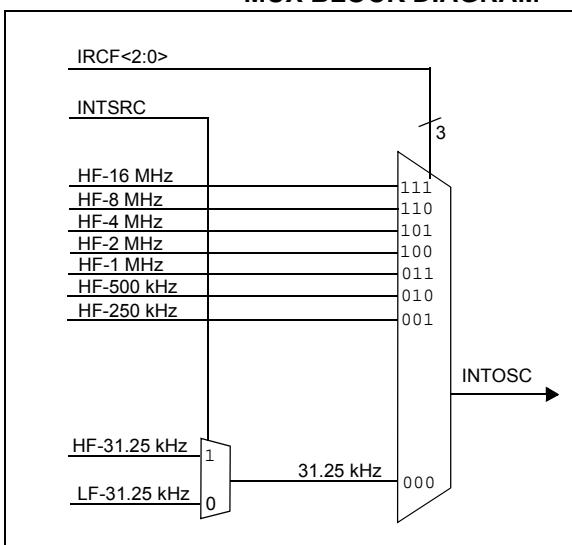


TABLE 3-1: PLL_SELECT TRUTH TABLE

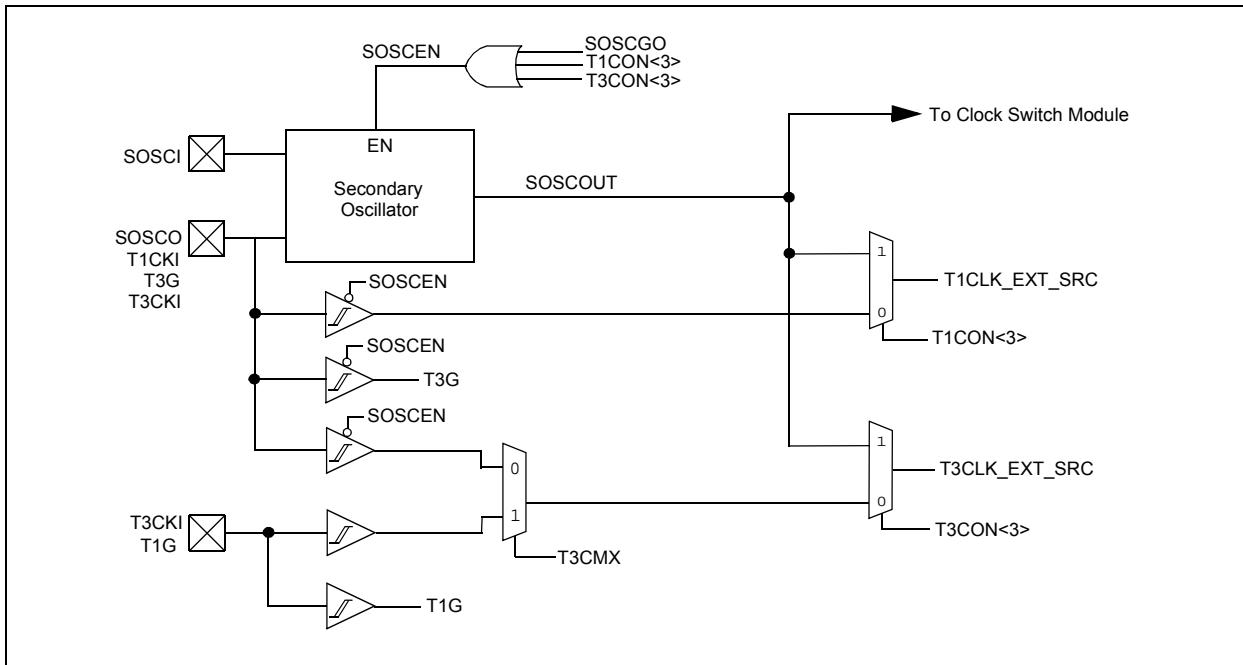
| Primary Clock MUX Source | FOSC<3:0> | CFGPLLLEN | PLLSEL | PLLEN | SPLLMULT | PLL_Select |
|--------------------------------|-----------|-----------|--------|-------|----------|----------------------|
| External Clock (ECHIO/ECHCLKO) | 010x | 1 | 1 | x | x | 3xPLL ⁽¹⁾ |
| | | | 0 | x | x | 4xPLL ⁽²⁾ |
| HS Crystal (HSH) | 0010 | 0 | x | 1 | 1 | 3xPLL ⁽¹⁾ |
| | | | | 0 | 0 | 4xPLL ⁽²⁾ |
| INTOSC (INTOSCI, INTOSCCLKO) | 100x | | | 0 | x | OFF |
| Fosc (all other modes) | xxxx | x | x | x | x | OFF |

Note 1: The input clock source must be 16 MHz when 3xPLL is used.

2: The input clock source must be 8 MHz to 12 MHz when 4xPLL is used.

PIC18(L)F2X/45K50

FIGURE 3-3: SECONDARY OSCILLATOR AND EXTERNAL CLOCK INPUTS



3.3 Register Definitions: Oscillator Control

REGISTER 3-1: 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 | | IRCF<2:0> | | OSTS ⁽¹⁾ | HFIofs | | SCS<1:0> |
| 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 RC Oscillator Frequency Select bits
 111 = HFINTOSC – (16 MHz)
 110 = HFINTOSC/2 – (8 MHz)
 101 = HFINTOSC/4 – (4 MHz)
 100 = HFINTOSC/8 – (2 MHz)
 011 = HFINTOSC/16 – (1 MHz)⁽²⁾
 010 = HFINTOSC/32 – (500 kHz)
 001 = HFINTOSC/64 – (250 kHz)
- If INTSRC = 1:
 000 = HFINTOSC/512 – (31.25 kHz)
- If INTSRC = 0:
 000 = INTRC – (31.25 kHz)
- bit 3 **OSTS:** Oscillator Start-up Time-out Status bit
 1 = Device is running from the clock defined by FOSC<3:0> of the CONFIG1H register
 0 = Device is running from the internal oscillator (HFINTOSC or INTRC)
- 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 bit
 1x = Internal oscillator block
 01 = Secondary (SOSC) oscillator
 00 = Primary clock (determined by FOSC<3:0> in CONFIG1H).

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

2: Default output frequency of HFINTOSC on Reset.

PIC18(L)F2X/45K50

REGISTER 3-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2

| R-0/0 | R-0/q | R/W-0 | R/W-0/0 | R/W-0/u | R/W-1/1 | R-0/0 | R-0/0 |
|--------|---------|--------|---------|-----------------------|---------|--------|--------|
| PLLRDY | SOSCRUN | INTSRC | PLLEN | SOSCGO ⁽¹⁾ | PRISD | HFIOFR | LFIOFS |
| bit 7 | | | | | | | bit 0 |

Legend:

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

bit 7

PLLRDY: PLL Run Status bit

- 1 = System clock comes from PLL
0 = System clock comes from an oscillator, other than PLL

bit 6

SOSCRUN: SOSC Run Status bit

- 1 = System clock comes from secondary SOSC
0 = System clock comes from an oscillator, other than SOSC

bit 5

INTSRC: HFINTOSC Divided by 512 Enable bit

- 1 = HFINTOSC used as the 31.25 kHz system clock reference – high accuracy
0 = INTRC used as the 31.25 kHz system clock reference – low power.

bit 4

PLLEN: Software PLL Enable bit

- If FOSC<3:0> = 100x, 010x or 001x
1 = PLL enabled
0 = PLL disabled

Else,

No effect on PLL operation.

bit 3

SOSCGO⁽¹⁾: Secondary Oscillator Start Control bit

- 1 = Secondary oscillator is enabled.
0 = Secondary oscillator is shut off if no other sources are requesting it.

bit 2

PRISD: Primary Oscillator Drive Circuit Shutdown bit

- 1 = Oscillator drive circuit on
0 = Oscillator drive circuit off (zero power)

bit 1

HFIOFR: HFINTOSC Status bit

- 1 = HFINTOSC is running
0 = HFINTOSC is not running

bit 0

LFIOFS: INTRC Frequency Stable bit

- 1 = INTRC is stable
0 = INTRC is not stable

Note 1: The SOSCGO bit is only reset on a POR Reset.

3.4 Clock Source Modes

Clock source modes can be classified as external or internal.

- External Clock modes rely on external circuitry for the clock source. Examples are: Clock modules (EC mode), quartz crystal resonators or ceramic resonators (LP, XT and HS modes) and Resistor-Capacitor (RC mode) circuits.
- Internal clock sources are contained internally within the Oscillator block. The Oscillator block has two internal oscillators: the 16 MHz High-Frequency Internal Oscillator (HFINTOSC) and the 31.25 kHz Low-Frequency Internal Oscillator (INTRC).

The system clock can be selected between external or internal clock sources via the System Clock Select (SCS<1:0>) bits of the OSCCON register. See [Section 3.11 “Clock Switching”](#) for additional information.

TABLE 3-2: OSCILLATOR DELAY EXAMPLES

| Switch From | Switch To | Frequency | Oscillator Delay |
|-------------------|-------------------|----------------------------------|----------------------------------|
| Sleep/POR | INTRC HFINTOSC | 31.25 kHz 31.25 kHz to 16 MHz | Oscillator Warm-Up Delay (TWARM) |
| Sleep/POR | EC, RC | DC – 48 MHz | 2 instruction cycles |
| INTRC (31.25 kHz) | EC, RC | DC – 48 MHz | 1 cycle of each |
| Sleep/POR | LP, XT, HS | 32 kHz to 25 MHz | 1024 Clock Cycles (OST) |
| Sleep/POR | PLL | 32 MHz to 48 MHz | 1024 Clock Cycles (OST) + 2 ms |
| INTRC (31.25 kHz) | INTRC HFINTOSC | 31.25 kHz to 16 MHz | 1 µs (approx.) |

3.5.2 EC MODE

The External Clock (EC) mode allows an externally generated logic level as the system clock source. When operating in this mode, an external clock source is connected to the OSC1 input and the OSC2 is available for general purpose I/O. [Figure 3-4](#) shows the pin connections for EC mode.

The External Clock (EC) offers different power modes, Low Power (ECL), Medium Power (ECM) and High Power (ECH), selectable by the FOSC<3:0> bits. Each mode is best suited for a certain range of frequencies. The ranges are:

- ECL – below 4 MHz
- ECM – between 4 MHz and 16 MHz
- ECH – above 16 MHz

The Oscillator Start-up Timer (OST) is disabled when EC mode is selected. Therefore, there is no delay in operation after a Power-on Reset (POR) or Wake-up from Sleep. Because the PIC® MCU design is fully static, stopping the external clock input will have the effect of halting the device while leaving all data intact. Upon restarting the external clock, the device will resume operation as if no time had elapsed.

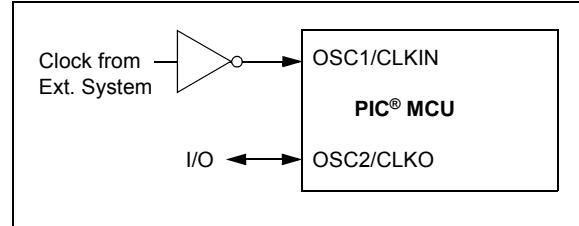
3.5 External Clock Modes

3.5.1 OSCILLATOR START-UP TIMER (OST)

When the oscillator module is configured for LP, XT or HS modes, the Oscillator Start-up Timer (OST) counts 1024 oscillations from OSC1. This occurs following a Power-on Reset (POR) and when the Power-up Timer (PWRT) has expired (if configured), or a wake-up from Sleep. During this time, the program counter does not increment and program execution is suspended. The OST ensures that the oscillator circuit, using a quartz crystal resonator or ceramic resonator, has started and is providing a stable system clock to the oscillator module. When switching between clock sources, a delay is required to allow the new clock to stabilize. These oscillator delays are shown in [Table 3-2](#).

In order to minimize latency between external oscillator start-up and code execution, the Two-Speed Clock Start-up mode can be selected (see [Section 3.12 “Two-Speed Clock Start-up Mode”](#)).

FIGURE 3-4: EXTERNAL CLOCK (EC) MODE OPERATION



PIC18(L)F2X/45K50

3.5.3 LP, XT, HS MODES

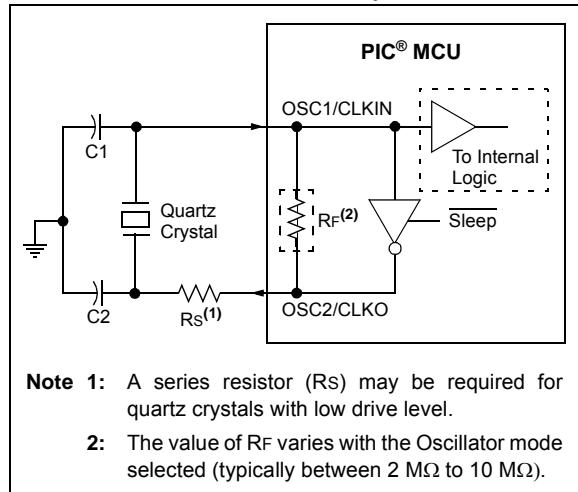
The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to OSC1 and OSC2 (Figure 3-5). The mode selects a low, medium or high gain setting of the internal inverter-amplifier to support various resonator types and speed.

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 offers a Medium Power (MP) and a High Power (HP) option selectable by the FOSC<3:0> bits. The MP selections are best suited for oscillator frequencies between 4 and 16 MHz. The HP selection has the highest gain setting of the internal inverter-amplifier and is best suited for frequencies above 16 MHz. HS mode is best suited for resonators that require a high drive setting.

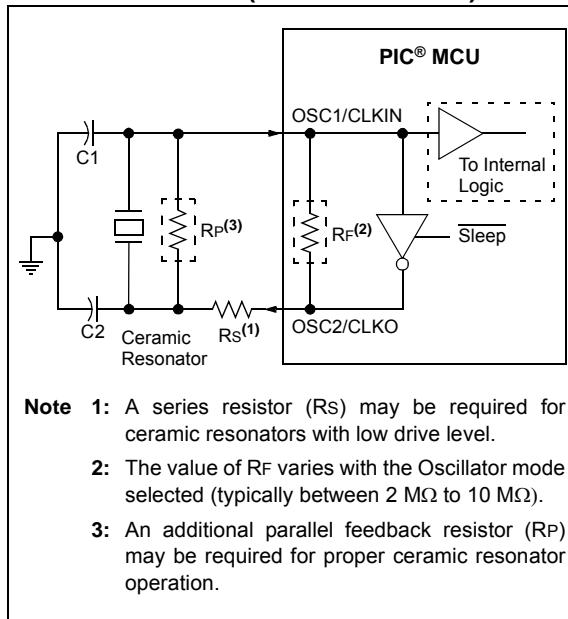
FIGURE 3-5: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)



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

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

FIGURE 3-6: CERAMIC RESONATOR OPERATION (XT OR HS MODE)



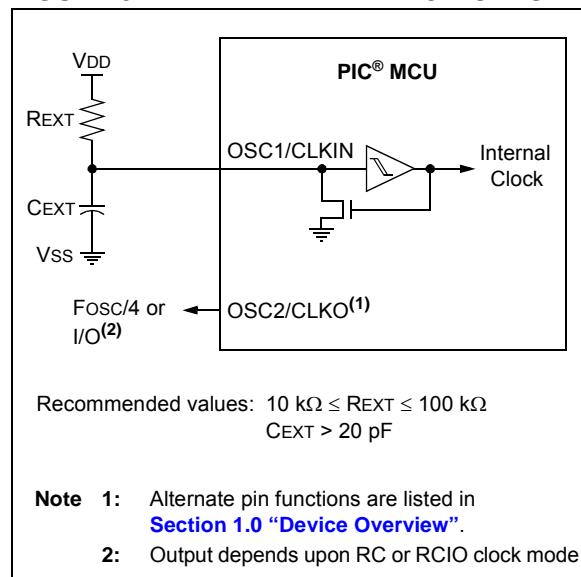
3.5.4 EXTERNAL RC MODES

The external Resistor-Capacitor (RC) modes support 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. There are two modes: RC and RCIO.

3.5.4.1 RC Mode

In RC mode, the RC circuit connects to OSC1. OSC2/CLKO outputs the RC oscillator frequency divided by 4. This signal may be used to provide a clock for external circuitry, synchronization, calibration, test or other application requirements. [Figure 3-7](#) shows the external RC mode connections.

FIGURE 3-7: EXTERNAL RC MODES



3.5.4.2 RCIO Mode

In RCIO mode, the RC circuit is connected to OSC1. OSC2 becomes a general purpose I/O pin.

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

- input threshold voltage variation
- component tolerances
- packaging variations in capacitance

The user also needs to take into account variation due to tolerance of external RC components used.

3.6 Internal Clock Modes

The oscillator module has two independent, internal oscillators that can be configured or selected as the system clock source.

1. The **HFINTOSC** (High-Frequency Internal Oscillator) is factory calibrated and operates at 16 MHz. The frequency of the HFINTOSC can be user-adjusted via software using the OSCTUNE register ([Register 3-3](#)).
2. The **INTRC** (Low-Frequency Internal Oscillator) is factory calibrated and operates at 31.25 kHz. The INTRC cannot be user-adjusted, but is designed to be stable over temperature and voltage.

The system clock speed can be selected via software using the Internal Oscillator Frequency select bits IRCF<2:0> of the OSCCON register. The INTSRC bit allows users to select which internal oscillator provides the clock source for the 31.25 kHz frequency option. This is covered in greater detail in [Section 3.2.3 "Low-Frequency Selection"](#).

The system clock can be selected between external or internal clock sources via the System Clock Selection (SCS<1:0>) bits of the OSCCON register. See [Section 3.11 "Clock Switching"](#) for more information.

3.6.1 INTOSC WITH I/O OR CLOCKOUT

Two of the clock modes selectable with the FOSC<3:0> bits of the CONFIG1H Configuration register configure the internal oscillator block as the primary oscillator. Mode selection determines whether OSC2/CLKO/RA6 will be configured as general purpose I/O (RA6) or Fosc/4 (CLKO). In both modes, OSC1/CLKIN/RA6 is configured as general purpose I/O. See [Section 26.0 "Special Features of the CPU"](#) for more information.

The CLKO signal may be used to provide a clock for external circuitry, synchronization, calibration, test or other application requirements.

PIC18(L)F2X/45K50

3.6.1.1 OSCTUNE Register

The HFINTOSC oscillator circuits are factory calibrated but can be adjusted in software by writing to the TUN<6:0> bits of the OSCTUNE register ([Register 3-3](#)).

The default value of the TUN<6:0> is '0'. The value is a 7-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. There is no indication that the shift has occurred.

The TUN<6:0> bits in OSCTUNE do not affect the INTRC frequency. Operation of features that depend on the INTRC 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 SPLLMULT bit, which controls whether 3x or 4xPLL clock multiplication is used when the PLL is enabled dynamically in software. For more details about the function of the SPLLMULT bit see [Section 3.8.2 "PLL in HFINTOSC Modes"](#).

3.7 Register Definitions: Oscillator Tuning

REGISTER 3-3: 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 |
|-------|-------|-------|-------|----------|-------|-------|-------|
| SPLL | MULT | | | TUN<6:0> | | | |
| 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

SPLLMULT: Software PLL Multiplier Select bit

If PLL Enabled, SPLL

MULT changes are ignored.

Else,

Selects which PLL multiplier will be used:

1 = 3xPLL is selected

0 = 4xPLL is selected

bit 6-0

TUN<6:0>: Frequency Tuning bits – affects HFINTOSC⁽¹⁾

0111111 = Maximum frequency

0111110 =

•••

0000001 =

0000000 = Center frequency. Oscillator module is running at the factory calibrated frequency.

1111111 =

•••

1000000 = Minimum frequency

Note 1: The TUN<6:0> bits may be supplied and controlled by the Active Clock Tuning module (see [Section 3.15 "Active Clock Tuning \(ACT\) Module"](#)) When the Active Clock Tuning is enabled, the TUN<6:0> bits are read-only.

3.7.1 INTRC

The Low-Frequency Internal Oscillator (INTRC) is a 31.25 kHz internal clock source. The INTRC is not tunable, but is designed to be stable across temperature and voltage. See [Section 29.0 “Electrical Specifications”](#) for the INTRC accuracy specifications.

The output of the INTRC can be a clock source to the primary clock or the INTOSC clock (see [Figure 3-1](#)). The INTRC is also the clock source for the Power-up Timer (PWRT), Watchdog Timer (WDT) and Fail-Safe Clock Monitor (FSCM).

3.7.2 FREQUENCY SELECT BITS (IRCF)

The HFINTOSC (16 MHz) outputs to a divide circuit that provides frequencies of 16 MHz to 31.25 kHz. These divide circuit frequencies, along with the 31.25 kHz INTRC output, are multiplexed to provide a single INTOSC clock output (see [Figure 3-1](#)). The IRCF<2:0> bits of the OSCCON register and the INTSRC bit of the OSCCON2 register select the output frequency of the internal oscillators. One of eight frequencies can be selected via software:

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

3.7.3 INTOSC FREQUENCY DRIFT

The factory calibrates the internal oscillator block outputs (HFINTOSC) for 16 MHz. However, this frequency may drift as VDD or temperature changes. It is possible to automatically tune the HFINTOSC frequency using USB or secondary oscillator sources using the active clock tuning module (see [Section 3.15 “Active Clock Tuning \(ACT\) Module”](#)). The HFINTOSC frequency may be manually adjusted using the TUN<6:0> bits in the OSCTUNE register. This has no effect on the INTRC clock source frequency.

Manually tuning the HFINTOSC source requires knowing when to make the adjustment, in which direction it should be made and, in some cases, how large a change is needed. Three possible compensation techniques are discussed in the following sections. However, other techniques may be used.

3.7.3.1 Compensating with the EUSART

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

3.7.3.2 Compensating with the Timers

This technique compares device clock speed to some reference clock. Two timers may be used; one timer is clocked by the peripheral clock, while the other is clocked by a fixed reference source, such as the Timer1 oscillator.

Both timers are cleared, but the timer clocked by the reference generates interrupts. When an interrupt occurs, the internally clocked timer is read and both timers are cleared. If the internally clocked timer value is greater than expected, then the internal oscillator block is running too fast. To adjust for this, decrement the OSCTUNE register.

3.7.3.3 Compensating with the CCP Module in Capture Mode

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

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

3.8 PLL Frequency Multiplier

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

3.8.1 PLL IN EXTERNAL OSCILLATOR MODES

The PLL can be enabled for any of the external oscillator modes using the OSC1/OSC2 pins. Medium-power and low-power oscillator mode selections in CONFIG1H<3:0> (FOSC) should not be used with the PLL. The PLL can be enabled using the CFGPLLLEN/PLLSEL configuration bits in the CONFIG1L register, or by software using the PLLEN/SPLLMULT special function register bits in OSCCON2 and OSCTUNE, respectively.

A selectable 3x or 4x frequency multiplier circuit is provided. This gives greater flexibility in source clock frequencies that can be used. Source clock frequencies between 8 and 12 MHz may use the 4x frequency multiplier to achieve operating speeds of 32 through 48 MHz. A source clock frequency of 16 MHz may use the 3x frequency multiplier to achieve 48 MHz operating speed.

3.8.2 PLL IN HFINTOSC MODES

The PLL can be enabled using the HFINTOSC internal oscillator block. The frequency select bits (IRCF<2:0> in the OSCCON register) should be configured for 16 MHz when using the HFINTOSC with 3x frequency multiplier. The IRCF bits should be configured for 8 MHz when using HFINTOSC with 4x frequency multiplier.

3.9 Effects of Power-Managed Modes on the Various Clock Sources

For more information about the modes discussed in this section see [Section 4.0 “Power-Managed Modes”](#). A quick reference list is also available in [Table 4-1](#).

When PRI_IDLE mode is selected, the designated primary oscillator continues to run without interruption. For all other power-managed modes, the oscillator using the OSC1 pin is disabled. The OSC1 pin (and OSC2 pin, if used by the oscillator) will stop oscillating.

In secondary clock modes (SEC_RUN and SEC_IDLE), the secondary oscillator (SOSC) is operating and providing the device clock. The secondary oscillator may also run in all power-managed modes if required to clock Timer1 or Timer3.

In internal oscillator modes (INTOSC_RUN and INTOSC_IDLE), the internal oscillator block provides the device clock source. The 31.25 kHz INTRC output can be used directly to provide the clock and may be enabled to support various special features, regardless of the power-managed mode (see [Section 26.3 “Watchdog Timer \(WDT\)”](#), [Section 3.12 “Two-Speed Clock Start-up Mode”](#) and [Section 3.13 “Fail-Safe Clock Monitor”](#) for more information on WDT, Fail-Safe Clock Monitor and Two-Speed Start-up). The HFINTOSC output may be used directly to clock the device or may be divided down by the postscaler. The HFINTOSC output is disabled when the clock is provided directly from the INTRC output.

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

Enabling any on-chip feature that will operate during Sleep will increase the current consumed during Sleep. The INTRC is required to support WDT operation. Other features may be operating that do not require a device clock source (i.e., SSP slave, INTn pins and others). Peripherals that may add significant current consumption are listed in [Table 29-8](#).

3.10 Power-up Delays

Power-up delays are controlled by two timers, so that no external Reset circuitry is required for most applications. The delays ensure that the device is kept in Reset until the device power supply is stable under normal circumstances and the primary clock is operating and stable. For additional information on power-up delays, see [Section 5.7 “Device Reset Timers”](#).

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up. It is enabled by clearing (= 0) the PWRTE Configuration bit.

The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable (LP, XT and HS modes). The OST does this by counting 1024 oscillator cycles before allowing the oscillator to clock the device.

When the PLL is enabled with external oscillator modes, the device runs off of the base external oscillator for 2 ms, following the OST delay, so the PLL can lock to the incoming clock frequency.

There is a delay of interval Tcsd, following POR, while the controller becomes ready to execute instructions. This delay runs concurrently with any other delays. This may be the only delay that occurs when any of the EC, RC or INTOSC modes are used as the primary clock source.

When the HFINTOSC is selected as the primary clock, the main system clock can be delayed until the HFINTOSC is stable. This is user selectable by the HFOFST bit of the CONFIG3H Configuration register. When the HFOFST bit is cleared, the main system clock is delayed until the HFINTOSC is stable. When the HFOFST bit is set, the main system clock starts immediately.

In either case, the HFIOFS bit of the OSCCON register can be read to determine whether the HFINTOSC is operating and stable.

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

| OSC Mode | OSC1 Pin | OSC2 Pin |
|----------------------|---|---|
| RC, INTOSC with CLKO | Floating, external resistor should pull high | At logic low (clock/4 output) |
| RC with IO | Floating, external resistor should pull high | Configured as PORTA, bit 6 |
| INTOSC with IO | Configured as PORTA, bit 7 | Configured as PORTA, bit 6 |
| EC with IO | Floating, pulled by external clock | Configured as PORTA, bit 6 |
| EC with CLKO | Floating, pulled by external clock | At logic low (clock/4 output) |
| LP, XT, HS | Feedback inverter disabled at quiescent voltage level | Feedback inverter disabled at quiescent voltage level |

Note: See [Table 5-2](#) in [Section 5.0 “Reset”](#) for time-outs due to Sleep and $\overline{\text{MCLR}}$ Reset.

3.11 Clock Switching

The system clock source can be switched between external and internal clock sources via software using the System Clock Select (SCS<1:0>) bits of the OSCCON register.

PIC18(L)F2X/45K50 devices contain circuitry to prevent clock “glitches” when switching between clock sources. A short pause in the device clock occurs during the clock switch. The length of this pause is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Clock transitions are discussed in greater detail in [Section 4.1.2 “Entering Power-Managed Modes”](#).

3.11.1 SYSTEM CLOCK SELECT (SCS<1:0>) BITS

The System Clock Select (SCS<1:0>) bits of the OSCCON register select the system clock source that is used for the CPU and peripherals.

- When SCS<1:0> = 00, the system clock source is determined by configuration of the FOSC<3:0> bits in the CONFIG1H Configuration register.
- When SCS<1:0> = 10, the system clock source is chosen by the internal oscillator frequency selected by the INTSRC bit of the OSCCON2<5> register and the IRCF<2:0> bits of the OSCCON register.
- When SCS<1:0> = 01, the system clock source is the 32.768 kHz secondary oscillator shared with Timer1 and Timer3.

After a Reset, the SCS<1:0> bits of the OSCCON register are always cleared.

Note: Any automatic clock switch, which may occur from Two-Speed Start-up or Fail-Safe Clock Monitor, does not update the SCS<1:0> bits of the OSCCON register. The user can monitor the SOSCRUN and LFIOFS bits of the OSCCON2 register, and the HFIOFS and OSTS bits of the OSCCON register to determine the current system clock source.

3.11.2 OSCILLATOR START-UP TIME-OUT STATUS (OSTS) BIT

The Oscillator Start-up Time-out Status (OSTS) bit of the OSCCON register indicates whether the system clock is running from the external clock source, as defined by the FOSC<3:0> bits in the CONFIG1H Configuration register, or from the internal clock source. In particular, when the primary oscillator is the source of the primary clock, OSTS indicates that the Oscillator Start-up Timer (OST) has timed out for LP, XT or HS modes.

3.11.3 CLOCK SWITCH TIMING

When switching between one oscillator and another, the new oscillator may not be operating which saves power (see [Figure 3-8](#)). If this is the case, there is a delay after the SCS<1:0> bits of the OSCCON register are modified before the frequency change takes place. The OSTS and HFIOFR, LFIOFS bits of the OSCCON and OSCCON2 registers will reflect the current active status of the external and HFINTOSC oscillators. The timing of a frequency selection is as follows:

1. SCS<1:0> bits of the OSCCON register are modified.
2. The old clock continues to operate until the new clock is ready.
3. Clock switch circuitry waits for two consecutive rising edges of the old clock after the new clock ready signal goes true.
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 released and new clock is switched in as the system clock.
7. Clock switch is complete.

See [Figure 3-1](#) for more details.

If the HFINTOSC is the source of both the old and new frequency, there is no start-up delay before the new frequency is active. This is because the old and new frequencies are derived from the HFINTOSC via the postscaler and multiplexer.

Start-up delay specifications are located in [Section 29.0 “Electrical Specifications”](#), under AC Specifications (Oscillator Module).

3.12 Two-Speed Clock Start-up Mode

Two-Speed Start-up mode provides additional power savings by minimizing the latency between external oscillator start-up and code execution. In applications that make heavy use of the Sleep mode, Two-Speed Start-up will remove the external oscillator start-up time from the time spent awake and can reduce the overall power consumption of the device.

This mode allows the application to wake-up from Sleep, perform a few instructions using the HFINTOSC as the clock source and go back to Sleep without waiting for the primary oscillator to become stable.

Note: Executing a SLEEP instruction will abort the oscillator start-up time and will cause the OSTS bit of the OSCCON register to remain clear.

When the oscillator module is configured for LP, XT or HS modes, the Oscillator Start-up Timer (OST) is enabled (see [Section 3.5.1 “Oscillator Start-up Timer \(OST\)”](#)). 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 as the OST is counting. When the OST count reaches 1024 and the OSTS bit of the OSCCON register is set, program execution switches to the external oscillator.

3.12.1 TWO-SPEED START-UP MODE CONFIGURATION

Two-Speed Start-up mode is enabled when all of the following settings are configured as noted:

- Two-Speed Start-up mode is enabled when the IESO of the CONFIG1H Configuration register is set.
- SCS<1:0> (of the OSCCON register) = 00.
- FOSC<2:0> bits of the CONFIG1H Configuration register are configured for LP, XT or HS mode.

Two-Speed Start-up mode becomes active after:

- Power-on Reset (POR) and, if enabled, after Power-up Timer (PWRT) has expired, or
- Wake-up from Sleep.

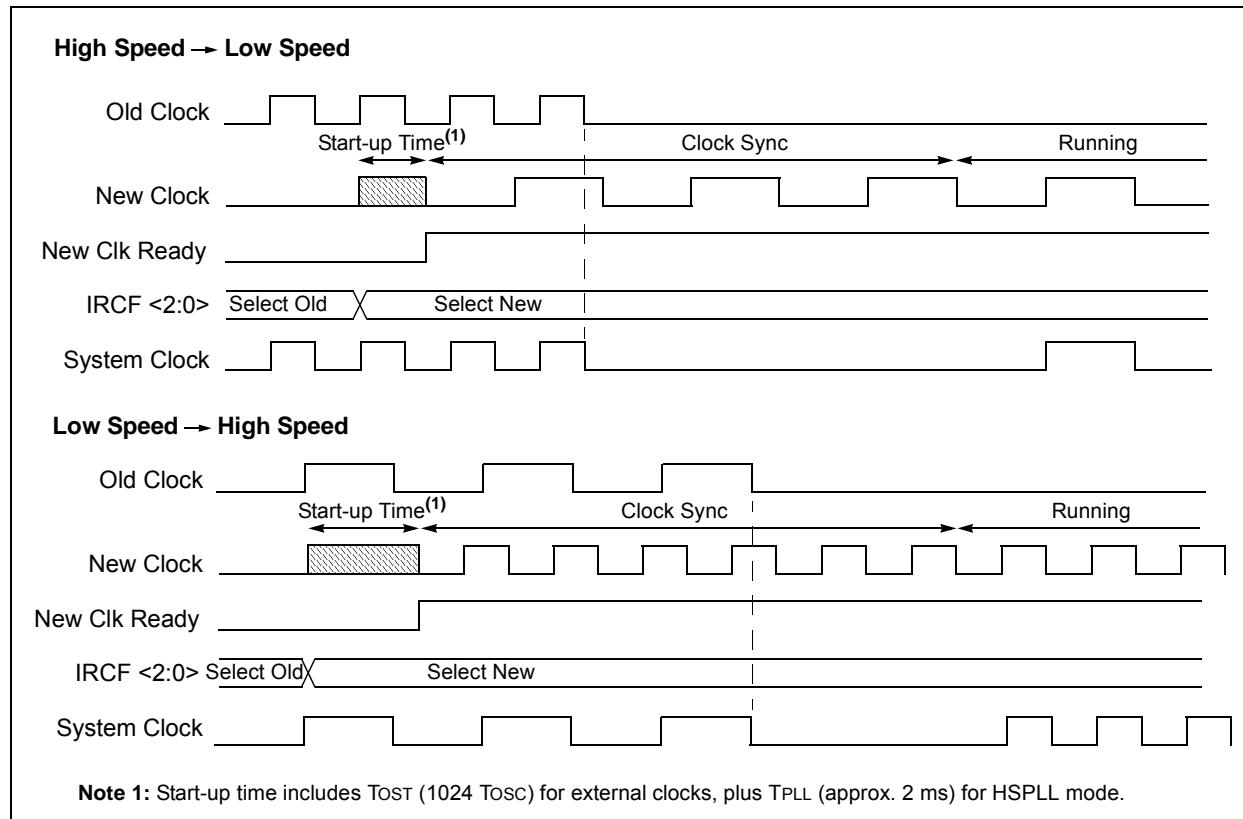
3.12.2 TWO-SPEED START-UP SEQUENCE

1. Wake-up from Power-on Reset or Sleep.
2. Instructions begin executing by the internal oscillator at the frequency set in the IRCF<2:0> bits of the OSCCON register.
3. OST enabled to count 1024 external clock cycles.
4. OST timed out. External clock is ready.
5. OSTS is set.
6. Clock switch finishes according to [Figure 3-8](#)

3.12.3 CHECKING TWO-SPEED CLOCK STATUS

Checking the state of the OSTS bit of the OSCCON register will confirm if the microcontroller is running from the external clock source, as defined by the FOSC<2:0> bits in CONFIG1H Configuration register, or the internal oscillator. OSTS = 0 when the external oscillator is not ready, which indicates that the system is running from the internal oscillator.

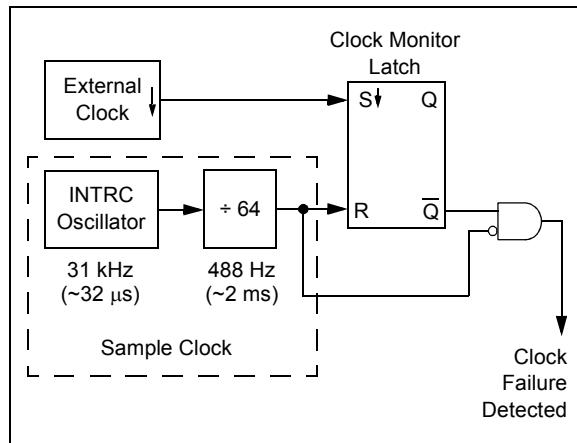
FIGURE 3-8: CLOCK SWITCH TIMING



3.13 Fail-Safe Clock Monitor

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

FIGURE 3-9: FSCM BLOCK DIAGRAM



3.13.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the INTRC by 64 (see [Figure 3-9](#)). 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.

3.13.2 FAIL-SAFE OPERATION

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

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

3.13.3 FAIL-SAFE CONDITION CLEARING

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

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

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

3.13.4 RESET OR WAKE-UP FROM SLEEP

The FSCM is designed to detect an oscillator failure after the Oscillator Start-up Timer (OST) has expired. The OST is used after waking up from Sleep and after any type of Reset. The OST is not used with the EC or RC Clock modes so that the FSCM will be active as soon as the Reset or wake-up has completed.

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.

Note: When the device is configured for Fail-Safe clock monitoring in either HS, XT, or LS oscillator modes then the IESO configuration bit should also be set so that the clock will automatically switch from the internal clock to the external oscillator when the OST times out.

PIC18(L)F2X/45K50

FIGURE 3-10: FSCM TIMING DIAGRAM

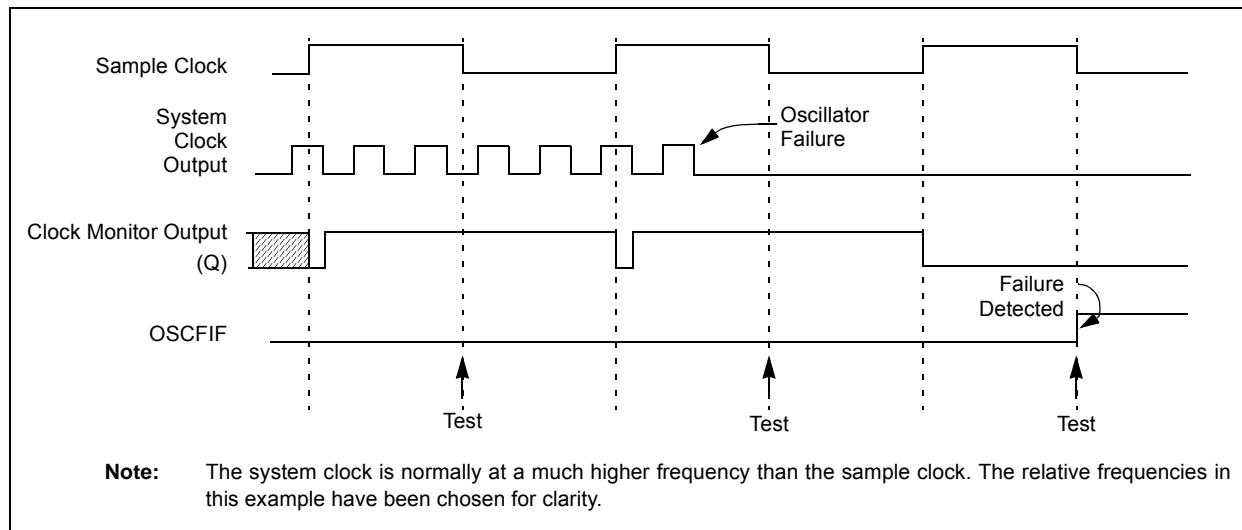


TABLE 3-4: REGISTERS ASSOCIATED WITH CLOCK SOURCES

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|---------|----------|-----------|--------|--------|---------|--------|----------|--------|---------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 |
| OSCCON | IDLEN | IRCF<2:0> | | | OSTS | HFIOPS | SCS<1:0> | | 33 |
| OSCCON2 | PLLRDY | SOSCRUN | INTSRC | PLLEN | SOSCIGO | PRISD | HFIOPR | LFIOFS | 34 |
| OSCTUNE | SPLLMULT | TUN<6:0> | | | | | | | 38 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used by clock sources.

TABLE 3-5: CONFIGURATION REGISTERS ASSOCIATED WITH CLOCK SOURCES

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|--------|-----------|-----------|------------|-------|--------|---------------------|
| CONFIG1H | IESO | FCMEN | PCLKEN | — | FOSC<3:0> | | | | 373 |
| CONFIG2L | — | LPBOR | — | BORV<1:0> | | BOREN<1:0> | | PWRREN | 374 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for clock sources.

3.14 Oscillator Settings for USB

When the PIC18(L)F2X/45K50 family devices are used for USB connectivity, a 6 MHz or 48 MHz clock must be provided to the USB module for operation in either Low-Speed or Full-Speed modes, respectively. This may require some forethought in selecting an oscillator frequency and programming the device.

The full range of possible oscillator configurations compatible with USB operation is shown in [Table 3-7](#).

TABLE 3-6: CLOCK FOR LOW-SPEED USB

| System Clock | CPUDIV<1:0> | Microcontroller Clock | LS48MHZ | USB Clock |
|--------------|-------------|-----------------------|---------|--------------|
| 48 | 11 | 48/6 = 8 MHz | 1 | 48/8 = 6 MHz |
| 48 | 10 | 48/3 = 16 MHz | 1 | 48/8 = 6 MHz |
| 48 | 01 | 48/2 = 24 MHz | 1 | 48/8 = 6 MHz |
| 48 | 00 | 48 MHz | 1 | 48/8 = 6 MHz |
| 24 | 11 | 24/6 = 4 MHz | 0 | 24/4 = 6 MHz |
| 24 | 10 | 24/3 = 8 MHz | 0 | 24/4 = 6 MHz |
| 24 | 01 | 24/2 = 12 MHz | 0 | 24/4 = 6 MHz |
| 24 | 00 | 24 MHz | 0 | 24/4 = 6 MHz |

TABLE 3-7: OSCILLATOR CONFIGURATION OPTIONS FOR USB OPERATION

| Input Oscillator Frequency | Clock Mode (FOSC<3:0>) | MCU Clock Division (CPUDIV<1:0>) | Microcontroller Clock Frequency |
|----------------------------|-----------------------------|----------------------------------|---------------------------------|
| 48 MHz | EC | ÷6 (11) | 8 MHz |
| | | ÷3 (10) | 16 MHz |
| | | ÷2 (01) | 24 MHz |
| | | None (00) | 48 MHz |
| 16 MHz | EC, HS or INTOSC with 3xPLL | ÷6 (11) | 8 MHz |
| | | ÷3 (10) | 16 MHz |
| | | ÷2 (01) | 24 MHz |
| | | None (00) | 48 MHz |
| 12 MHz | EC or HS with 4xPLL | ÷6 (11) | 8 MHz |
| | | ÷3 (10) | 16 MHz |
| | | ÷2 (01) | 24 MHz |
| | | None (00) | 48 MHz |
| 24 MHz | EC or HS ⁽¹⁾ | ÷6 (11) | 4 MHz |
| | | ÷3 (10) | 8 MHz |
| | | ÷2 (01) | 12 MHz |
| | | None (00) | 24 MHz |

Note 1: The 24 MHz mode (without PLL) is only compatible with low-speed USB. Full-speed USB requires a 48 MHz system clock.

3.14.1 LOW-SPEED OPERATION

The USB clock for Low-Speed mode is derived from the primary oscillator or from the PLL. In order to operate the USB module in Low-Speed mode, a 6 MHz clock must be provided to the USB module.

See [Table 3-6](#) and [Table 3-7](#) for possible combinations which can be used for low-speed USB operation.

PIC18(L)F2X/45K50

3.15 Active Clock Tuning (ACT) Module

The Active Clock Tuning (ACT) module continuously adjusts the 16 MHz internal oscillator, using an available external reference, to achieve $\pm 0.20\%$ accuracy. This eliminates the need for a high-speed, high-accuracy external crystal when the system has an available lower speed, lower power, high-accuracy clock source available.

Systems implementing a Real-Time Clock Calendar (RTCC) or a full-speed USB application can take full advantage of the ACT module.

3.16 Active Clock Tuning Operation

The ACT module defaults to the disabled state after any Reset. When the ACT module is disabled, the user can write to the TUN<6:0> bits in the OSCTUNE register to manually adjust the 16 MHz internal oscillator.

The module is enabled by setting the ACTEN bit of the ACTCON register. When enabled, the ACT module takes control of the OSCTUNE register. The ACT module uses the selected ACT reference clock to tune the 16 MHz internal oscillator to an accuracy of 16 MHz $\pm 0.2\%$. The tuning automatically adjusts the OSCTUNE register every reference clock cycle.

- Note 1:** When the ACT module is enabled, the OSCTUNE register is only updated by the module. Writes to the OSCTUNE register by the user are inhibited, but reading the register is permitted.
- 2:** After disabling the ACT module, the user should wait three instructions before writing to the OSCTUNE register.

3.17 Active Clock Tuning Source Selection

The ACT reference clock is selected with the ACTSRC bit of the ACTCON register. The reference clock sources are provided by the:

- USB module in full-speed operation (ACT_clk)
- Secondary clock at 32.768 kHz (SOSC_clk)

3.18 ACT Lock Status

The ACTLOCK bit will be set to '1', when the 16 MHz internal oscillator is successfully tuned.

The bit will be cleared by the following conditions:

- Out of Lock condition
- Device Reset
- Module is disabled

3.19 ACT Out-of-Range Status

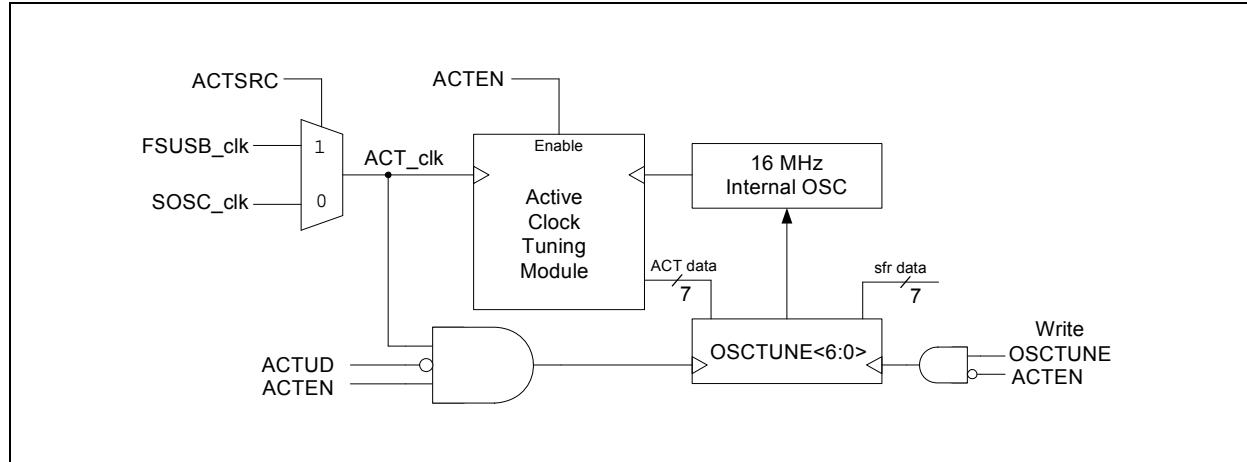
If the ACT module requires an OSCTUNE value outside the range to achieve $\pm 0.20\%$ accuracy, then the ACT Out-of-Range (ACTORS) Status bit will be set to '1'.

An out-of-range status can occur:

- When the 16 MHZ internal oscillator is tuned to its lowest frequency and the next ACT_clk event requests a lower frequency.
- When the 16 MHZ internal oscillator is tuned to its highest frequency and the next ACT_clk event requests a higher frequency.

When the ACT out-of-range event occurs, the 16 MHz internal oscillator will continue to use the last written OSCTUNE value. When the OSCTUNE value moves back within the tunable range and ACTLOCK is established, the ACTORS bit is cleared to '0'.

FIGURE 3-11: ACTIVE CLOCK TUNING BLOCK DIAGRAM



3.20 Active Clock Tuning Update Disable

When the ACT module is enabled, the OSCTUNE register is continuously updated every ACT_clk period. Setting the ACT Update Disable bit can be used to suspend updates to the OSCTUNE register, without disabling the module. If the 16 MHz internal oscillator drifts out of the accuracy range, the ACT Status bits will change and an interrupt can be generated to notify the application.

Clearing the ACTUD bit will engage the ACT updates to OSCTUNE and an interrupt can be generated to notify the application.

3.21 Interrupts

The ACT module will set the ACT module Interrupt Flag, (ACTIF) when either of the ACT module Status bits (ACTLOCK or ACTORS) change state, regardless if the interrupt is enabled, (ACTIE = 1). The ACTIF and ACTIE bits are in the PIR1 and PIE1 registers, respectively. When ACTIE = 1, an interrupt will be generated whenever the ACT module Status bits change.

The ACTIF bit must be cleared in software, regardless of the interrupt enable setting.

3.22 Operation during Sleep

This ACT module does not run during Sleep and will not generate interrupts during Sleep.

PIC18(L)F2X/45K50

3.23 Register Definitions: Active Clock Tuning Control

REGISTER 3-4: ACTCON: ACTIVE CLOCK TUNING (ACT) CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R-0/0 | U-0 | R-0/0 | U-0 |
|---------|---------|-----|-----------------------|---------|-----|--------|-----|
| ACTEN | ACTUD | — | ACTSRC ⁽¹⁾ | ACTLOCK | — | ACTORS | — |
| 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 **ACTEN:** Active Clock Tuning Selection bit
1 = ACT module is enabled, updates to OSCTUNE are exclusive to the ACT module.
0 = ACT module is disabled
- bit 6 **ACTUD:** Active Clock Tuning Update Disable bit
1 = Updates to the OSCTUNE register from ACT module are disabled.
0 = Updates to the OSCTUNE register from ACT module are enabled.
- bit 5 **Unimplemented:** Read as '0'
- bit 4 **ACTSRC:** Active Clock Tuning Source Selection bit
1 = The HFINTOSC Oscillator is tuned to approximately match the USB host clock tolerance
0 = The HFINTOSC Oscillator is tuned to approximately match the 32.768 kHz SOSC tolerance
- bit 3 **ACTLOCK:** Active Clock Tuning Lock Status bit
1 = Locked; 16 MHz internal oscillator is within $\pm 0.20\%$.Locked
0 = Not locked; 16 MHz internal oscillator tuning has not stabilized within $\pm 0.20\%$
- bit 2 **Unimplemented:** Read as '0'
- bit 1 **ACTORS:** Active Clock Tuning Out-of-Range Status bit
1 = Out-of-range; oscillator frequency is outside of the OSCTUNE range
0 = In-range; oscillator frequency is within the OSCTUNE range
- bit 0 **Unimplemented:** Read as '0'

Note 1: The ACTSRC bit should only be changed when ACTEN = 0.

TABLE 3-8: SUMMARY OF REGISTERS ASSOCIATED WITH ACT SOURCES

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|---------|-------------|---------|-------------|----------|---------|--------|--------|----------|------------------|
| ACTCON | ACTEN | ACTUD | — | ACTSRC | ACTLOCK | — | ACTORS | — | 50 |
| OSCCON | IDLEN | | IRCF<2:0> | | OSTS | HFIOFS | | SCS<1:0> | 33 |
| OSCTUNE | SPLLMULT | | | TUN<6:0> | | | | | 38 |
| OSCCON2 | PLL RDY | SOSCRUN | INTSRC | PLLEN | SOSC GO | PRISD | HFIOFR | LFIOFS | 34 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | SOSCEN | T1SYNC | RD16 | TMR1ON | 165 |

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

TABLE 3-9: SUMMARY OF CONFIGURATION WORD WITH ACT SOURCES

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|--------|-------|-----------|-------|-------|-------|------------------|
| CONFIG1H | IESO | FCMEN | PCLKEN | — | FOSC<3:0> | | | | 373 |

4.0 POWER-MANAGED MODES

PIC18(L)F2X/45K50 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 of the clock switching features allows the controller to use the secondary oscillator (SOSC) in place of the primary oscillator. Also included is the Sleep mode, offered by all PIC microcontroller devices, where all device clocks are stopped.

4.1 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 (OSCCON<7>) controls CPU clocking, while the SCS<1:0> bits (OSCCON<1:0>) select the clock source. The individual modes, bit settings, clock sources and affected modules are summarized in [Table 4-1](#).

TABLE 4-1: 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 – SOSC 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 – SOSC 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 INTRC source.

4.1.1 CLOCK SOURCES

The SCS<1:0> bits allow the selection of one of three clock sources for power-managed modes. They are:

- the primary clock, as defined by the FOSC<3:0> Configuration bits
- the secondary clock (the SOSC oscillator)
- the internal oscillator block

4.1.2 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 3.11 “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.

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.

4.1.3 MULTIPLE FUNCTIONS OF THE SLEEP COMMAND

The power-managed mode that is invoked with the `SLEEP` instruction is determined by the value of the `IDLEN` bit at the time the instruction is executed. If `IDLEN = 0`, when `SLEEP` is executed, the device enters the sleep mode and all clocks stop and minimum power is consumed. If `IDLEN = 1`, when `SLEEP` is executed, the device enters the IDLE mode and the system clock continues to supply a clock to the peripherals but is disconnected from the CPU.

4.2 Run Modes

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

4.2.1 PRI_RUN MODE

The `PRI_RUN` mode is the normal, full-power execution mode of the microcontroller. This is also the default mode upon a device Reset, unless Two-Speed Start-up is enabled (see [Section 3.12 “Two-Speed Clock Start-up Mode”](#) for details). In this mode, the device is operated off the oscillator defined by the `FOSC<3:0>` bits of the `CONFIG1H` Configuration register.

4.2.2 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 to ‘01’. When `SEC_RUN` mode is active, all of the following are true:

- The device clock source is switched to the `SOSC` oscillator (see [Figure 4-1](#))
- The primary oscillator is shut down
- The `SOSCRUN` bit (`OSCCON2<6>`) is set
- The `OSTS` bit (`OSCCON<3>`) is cleared

Note: The secondary external oscillator should already be running prior to entering `SEC_RUN` mode. If the `SOSCGO` bit or any of the `SOSCEN` bits are not set when the `SCS<1:0>` bits are set to ‘01’, entry to `SEC_RUN` mode will not occur until the `SOSCGO` bit is set and secondary external oscillator is ready.

On transitions from `SEC_RUN` mode to `PRI_RUN` mode, the peripherals and CPU continue to be clocked from the `SOSC` oscillator, while the primary clock is started. When the primary clock becomes ready, a clock switch back to the primary clock occurs (see

[Figure 4-2](#)). When the clock switch is complete, the `SOSCRUN` bit is cleared, the `OSTS` bit is set and the primary clock is providing the clock. The `IDLEN` and `SCS` bits are not affected by the wake-up and the `SOSC` oscillator continues to run.

4.2.3 RC_RUN MODE

In `RC_RUN` mode, the CPU and peripherals are clocked from the internal oscillator block using the `INTOSC` multiplexer. In this mode, the primary clock is shut down. When using the `INTRC` source, this mode provides the best power conservation of all the Run modes, while still executing code. It works well for user applications which are not highly timing-sensitive or do not require high-speed clocks at all times. If the primary clock source is the internal oscillator block – either `INTRC` or `HFINTOSC` – there are no distinguishable differences between the `PRI_RUN` and `RC_RUN` modes during execution. Entering or exiting `RC_RUN` mode, however, causes a clock switch delay. Therefore, if the primary clock source is the internal oscillator block, using `RC_RUN` mode is not recommended.

This mode is entered by setting the `SCS1` bit to ‘1’. To maintain software compatibility with future devices, it is recommended that the `SCS0` bit also be cleared, even though the bit is ignored. When the clock source is switched to the `INTOSC` multiplexer (see [Figure 4-1](#)), the primary oscillator is shut down and the `OSTS` bit is cleared. The `IRCF<2:0>` bits (`OSCCON<6:4>`) may be modified at any time to immediately change the clock speed.

When the `IRCF` bits and the `INTSRC` bit are all clear, the `INTOSC` output (`HFINTOSC`) is not enabled and the `HFIOFS` bit will remain clear. There will be no indication of the current clock source. The `INTRC` source is providing the device clocks.

If the `IRCF` bits are changed from all clear (thus, enabling the `INTOSC` output) or if `INTSRC` is set, then the `HFIOFS` bit is set after the `INTOSC` output becomes stable. For details, see [Table 4-2](#).

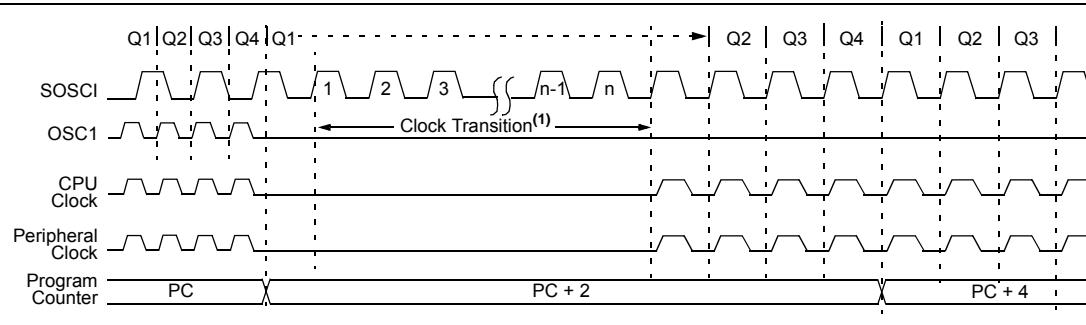
Clocks to the device continue while the `INTOSC` source stabilizes after an interval of `TIOB1`.

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

PIC18(L)F2X/45K50

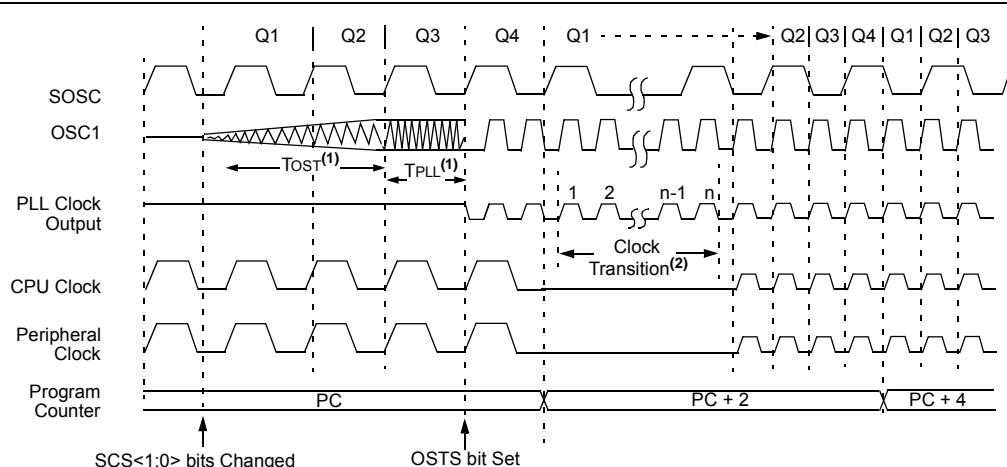
On transitions from RC_RUN mode to PRI_RUN mode, the device continues to be clocked from the INTOSC multiplexer while the primary clock is started. When the primary clock becomes ready, a clock switch to the primary clock occurs (see Figure 4-3). When the clock switch is complete, the HFIOFS bit is cleared,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

FIGURE 4-1: TRANSITION TIMING FOR ENTRY TO SEC_RUN MODE



Note 1: Clock transition typically occurs within 2-4 Tosc.

FIGURE 4-2: TRANSITION TIMING FROM SEC_RUN MODE TO PRI_RUN MODE (HSPLL)



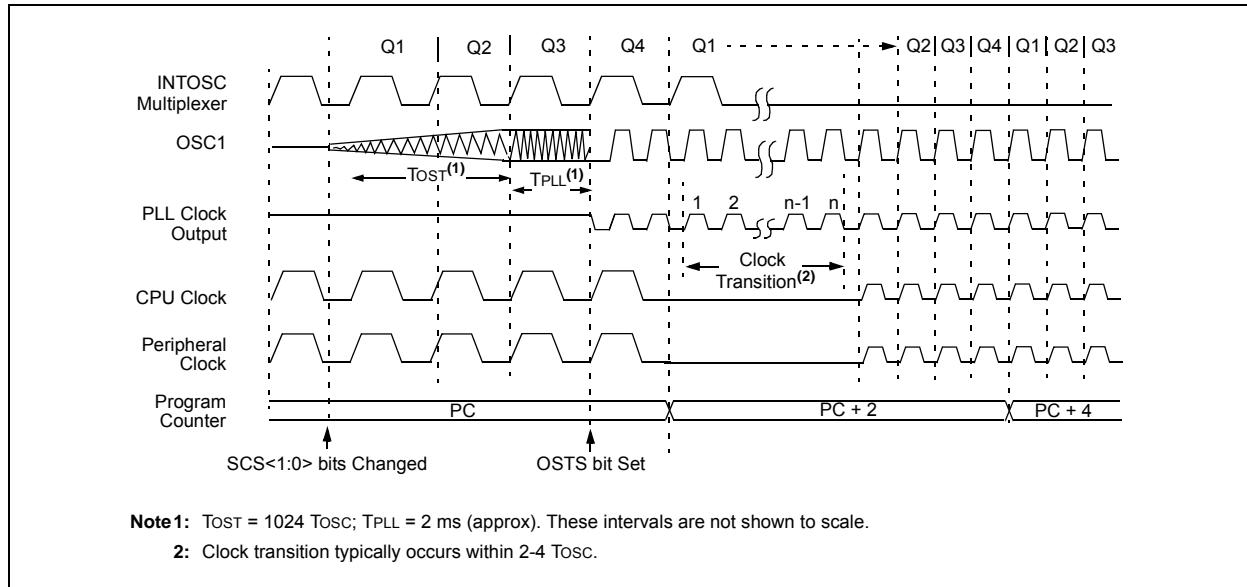
Note1: TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

2: Clock transition typically occurs within 2-4 Tosc.

TABLE 4-2: INTERNAL OSCILLATOR FREQUENCY STABILITY BITS

| IRCF<2:0> | INTSRC | Selected Oscillator | Selected Oscillator Stable when: |
|-----------|--------|---------------------|----------------------------------|
| 000 | 0 | INTRC | LFIOFS = 1 |
| 000 | 1 | HFINTOSC | HFIOFS = 1 |
| 001-111 | x | HFINTOSC | HFIOFS = 1 |

FIGURE 4-3: TRANSITION TIMING FROM RC_RUN MODE TO PRI_RUN MODE



PIC18(L)F2X/45K50

4.3 Sleep Mode

The Power-Managed Sleep mode in the PIC18(L)F2X/45K50 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 4-4) 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 INT RC source will continue to operate. If the SOSC 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 4-5), 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 26.0 “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.

REGISTER 4-1: VREGCON – VOLTAGE REGULATOR POWER CONTROL REGISTER⁽¹⁾

| U-0 | U-0 | U-0 | U-0 | U-0 | R-0 | R/W-0/0 | R/W-0/0 |
|-------|-----|-----|-----|-----|-----|-------------|---------|
| — | — | — | — | — | — | VREGPM<1:0> | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

-n/n = Value at POR and BOR/Value at all other resets

U = Unimplemented bit, read as ‘0’

W = Writable bit

‘0’ = Bit is cleared

x = Bit is unknown

‘1’ = bit is set

bit 7-2

Unimplemented: Read as ‘0’

bit 1-0

VREGPM<1:0>: Voltage Regulator Power mode bits

11 = Band gap not forced in Sleep; LDO off in Sleep; ULP Regulator active

10 = Band gap forced in Sleep; LDO off in Sleep; ULP Regulator active

01 = LDO in Low-Power mode in Sleep, if no peripherals require High-Power mode.

00 = LDO in High-Power mode – always

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

2: Default output frequency of HFINTOSC on Reset.

4.4 Idle Modes

The Idle modes allow the controller's CPU to be selectively shut down while the peripherals continue to operate. Selecting a particular Idle mode allows users to further manage power consumption.

If the IDLEN bit is set to a '1' when a SLEEP instruction is executed, the peripherals will be clocked from the clock source selected 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 INTRC source will continue to operate. If the SOSC 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 4-4: TRANSITION TIMING FOR ENTRY TO SLEEP MODE

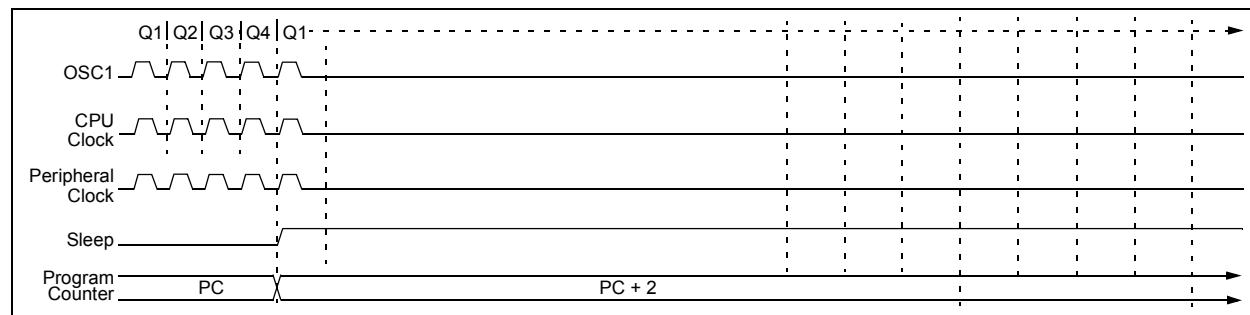
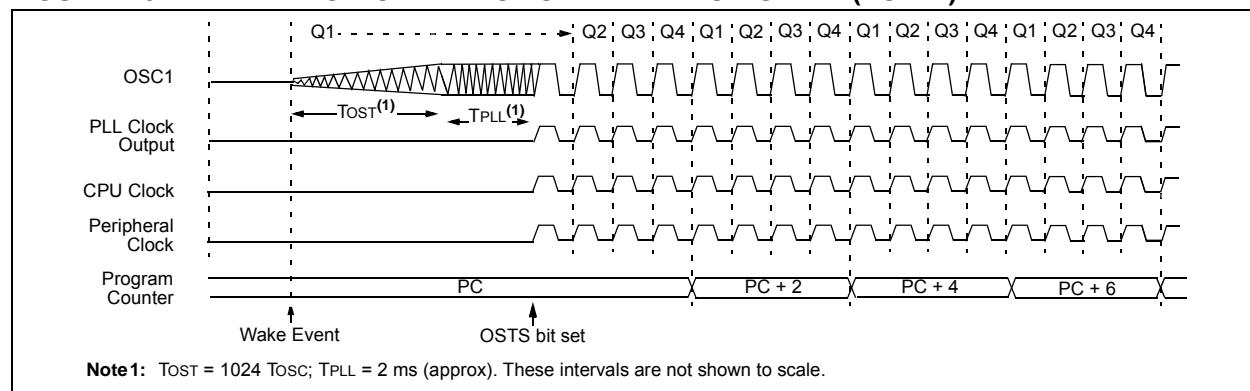


FIGURE 4-5: TRANSITION TIMING FOR WAKE FROM SLEEP (HSPLL)



4.4.1 PRI_IDLE MODE

This mode is unique among the three low-power Idle modes, in that it does not disable the primary device clock. For timing sensitive applications, this allows for the fastest resumption of device operation with its more accurate primary clock source, since the clock source does not have to "warm-up" or transition from another oscillator.

PRI_IDLE mode is entered from PRI_RUN mode by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set IDLEN first, then clear the SCS bits and execute SLEEP. Although the CPU is disabled, the peripherals continue

to be clocked from the primary clock source specified by the FOSC<3:0> Configuration bits. The OSTS bit remains set (see [Figure 4-6](#)).

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 4-7](#)).

PIC18(L)F2X/45K50

4.4.2 SEC_IDLE MODE

In SEC_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the SOSC 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 SOSC oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the SOSCRUN bit is set.

When a wake event occurs, the peripherals continue to be clocked from the SOSC oscillator. After an interval of TcSD following the wake event, the CPU begins executing code being clocked by the SOSC oscillator. The IDLEN and SCS bits are not affected by the wake-up; the SOSC oscillator continues to run (see Figure 4-7).

Note: The SOSC oscillator should already be running prior to entering SEC_IDLE mode. At least one of the secondary oscillator enable bits (SOSCEN, T1CON<3> or T3CON<3>) must be set when the SLEEP instruction is executed. Otherwise, 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 4-6: TRANSITION TIMING FOR ENTRY TO IDLE MODE

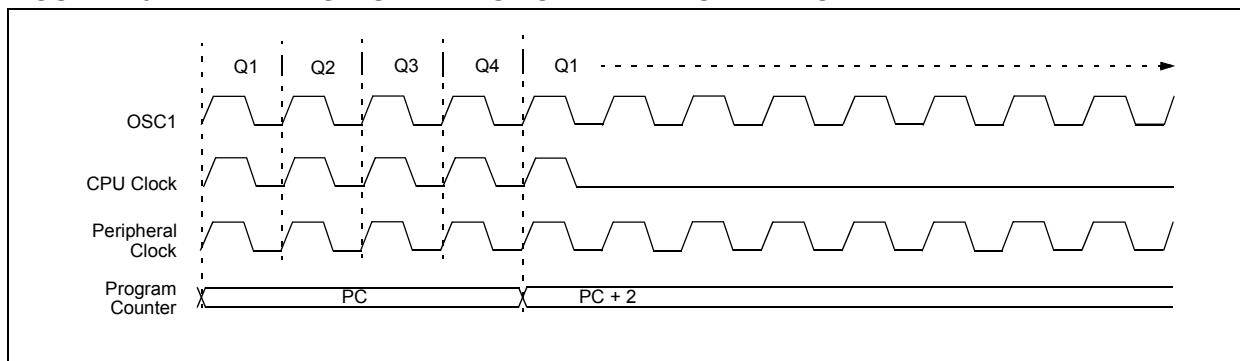
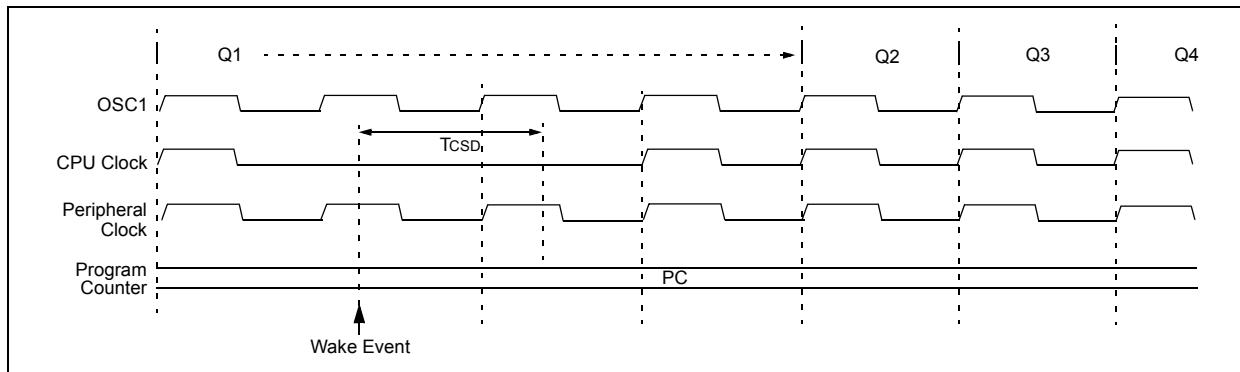


FIGURE 4-7: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE



4.4.3 RC_IDLE MODE

In RC_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the internal oscillator block 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 bits are set, the HFINTOSC output is enabled. The HFIOFS bit becomes set after the HFINTOSC output stabilizes after an interval of TIOBST. For information on the HFIOFS bit, see [Table 4-2](#).

Clocks to the peripherals continue while the HFINTOSC source stabilizes. The HFIOFS bit will remain set if the IRCF bits were previously set at a non-zero value or if INTSRC was set before the SLEEP instruction was executed and the HFINTOSC source was already stable. If the IRCF bits and INTSRC are all clear, the HFINTOSC output will not be enabled, the HFIOFS 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 INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

4.5 Exiting Idle and Sleep Modes

An exit from Sleep mode or any of the Idle modes is triggered by 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 4.2 “Run Modes”, Section 4.3 “Sleep Mode”](#) and [Section 4.4 “Idle Modes”](#)).

4.5.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit from an Idle 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 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 10.0 “Interrupts”](#)).

A fixed delay of interval Tcsd following the wake event is required when leaving Sleep and Idle modes. This delay is required for the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

4.5.2 EXIT BY WDT TIME-OUT

A WDT time-out will cause different actions depending on which power-managed mode the device is in when the time-out occurs.

If the device is not executing code (all Idle modes and Sleep mode), the time-out will result in an exit from the power-managed mode (see [Section 4.2 “Run Modes”](#) and [Section 4.3 “Sleep Mode”](#)). If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see [Section 26.3 “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

4.5.3 EXIT BY RESET

Exiting Sleep and Idle modes by Reset causes code execution to restart at address 0. See [Section 5.0 "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.

4.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

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

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

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

4.6 Selective Peripheral Module Control

Idle mode allows users to substantially reduce power consumption by stopping the CPU clock. Even so, peripheral modules still remain clocked, and thus, consume power. There may be cases where the application needs what IDLE mode does not provide: the allocation of power resources to the CPU processing with minimal power consumption from the peripherals. PIC18(L)F2X/45K50 family devices address this requirement by allowing peripheral modules to be selectively disabled, reducing or eliminating their power consumption. This can be done with control bits in the Peripheral Module Disable (PMD) registers. These bits generically named XXXMD are located in control registers PMD0 or PMD1.

Setting the PMD bit for a module disables all clock sources to that module, reducing its power consumption to an absolute minimum. In this state, power to the control and status registers associated with the peripheral is removed. Writes to these registers have no effect and read values are invalid. Clearing a set PMD bit restores power to the associated control and status registers, thereby setting those registers to their default values.

4.7 Register Definitions: Peripheral Module Disable

REGISTER 4-2: PMD0: PERIPHERAL MODULE DISABLE REGISTER 0

| U-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|--------|-------|-------|-----|--------|--------|--------|
| — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD |
| 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 | UARTMD: UART Peripheral Module Disable Control bit 1 = Module is disabled, clock source is disconnected, module does not draw digital power 0 = Module is enabled, clock source is connected, module draws digital power |
| bit 5 | USBMD: USB Peripheral Module Disable Control bit 1 = Module is disabled, clock source is disconnected, module does not draw digital power 0 = Module is enabled, clock source is connected, module draws digital power |
| bit 4 | ACTMD: Active Clock Tuning Peripheral Module Disable Control bit 1 = Module is disabled and does not draw any digital power 0 = Module is enabled and available for use; will draw digital power |
| bit 3 | Unimplemented: Read as '0'. |
| bit 2 | TMR3MD: Timer3 Peripheral Module Disable Control bit 1 = Module is disabled, clock source is disconnected, module does not draw digital power 0 = Module is enabled, clock source is connected, module draws digital power |
| bit 1 | TMR2MD: Timer2 Peripheral Module Disable Control bit 1 = Module is disabled, clock source is disconnected, module does not draw digital power 0 = Module is enabled, clock source is connected, module draws digital power |
| bit 0 | TMR1MD: Timer1 Peripheral Module Disable Control bit 1 = Module is disabled, clock source is disconnected, module does not draw digital power 0 = Module is enabled, clock source is connected, module draws digital power |

PIC18(L)F2X/45K50

REGISTER 4-3: PMD1: PERIPHERAL MODULE DISABLE 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 |
|-------|--------|--------|--------|--------|-------|--------|--------|
| — | MSSPMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD |
| 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

MSSPMD: MSSP Peripheral Module Disable Control bit

1 = Module is disabled, clock source is disconnected, module does not draw digital power
0 = Module is enabled, clock source is connected, module draws digital power

bit 5

CTMUMD: CTMU Peripheral Module Disable Control bit

1 = Module is disabled, clock source is disconnected, module does not draw digital power
0 = Module is enabled, clock source is connected, module draws digital power

bit 4

CMP2MD: Comparator 2 Peripheral Module Disable Control bit

1 = Module is disabled, clock source is disconnected, module does not draw digital power
0 = Module is enabled, clock source is connected, module draws digital power

bit 3

CMP1MD: Comparator 1 Peripheral Module Disable Control bit

1 = Module is disabled, clock source is disconnected, module does not draw digital power
0 = Module is enabled, clock source is connected, module draws digital power

bit 2

ADCMD: Analog-to-Digital Converter Peripheral Module Disable Control bit

1 = Module is disabled, clock source is disconnected, module does not draw digital power
0 = Module is enabled, clock source is connected, module draws digital power

bit 1

CCP2MD: CCP2 Peripheral Module Disable Control bit

1 = Module is disabled, clock source is disconnected, module does not draw digital power
0 = Module is enabled, clock source is connected, module draws digital power

bit 0

CCP1MD: CCP1 Peripheral Module Disable Control bit

1 = Module is disabled, clock source is disconnected, module does not draw digital power
0 = Module is enabled, clock source is connected, module draws digital power

5.0 RESET

The PIC18(L)F2X/45K50 devices differentiate between various kinds of Reset:

- Power-on Reset (POR)
- MCLR Reset during normal operation
- MCLR Reset during power-managed modes
- Watchdog Timer (WDT) Reset (during execution)
- Programmable Brown-out Reset (BOR)
- RESET Instruction
- Stack Full Reset
- Stack Underflow Reset

This section discusses Resets generated by MCLR, POR and BOR and covers the operation of the various start-up timers. Stack Reset events are covered in [Section 6.2.1 “Stack Full and Underflow Resets”](#). WDT Resets are covered in [Section 26.3 “Watchdog Timer \(WDT\)”](#).

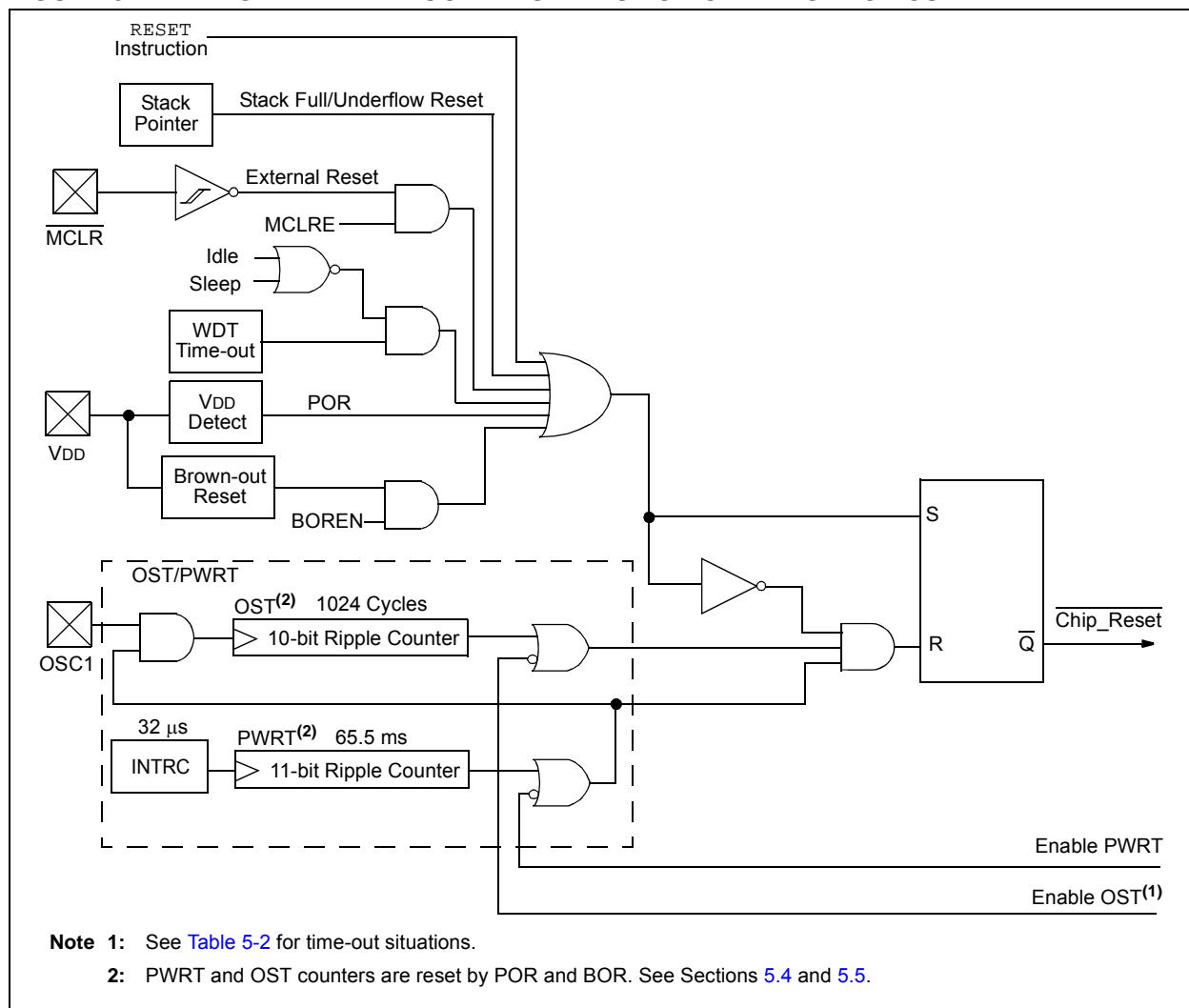
A simplified block diagram of the On-Chip Reset Circuit is shown in [Figure 5-1](#).

5.1 RCON Register

Device Reset events are tracked through the RCON register ([Register 5-1](#)). The lower five bits of the register indicate that a specific Reset event has occurred. In most cases, these bits can only be 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 5.8 “Reset State of Registers”](#).

The RCON register also has control bits for setting interrupt priority (IPEN) and software control of the BOR (SBOREN). Interrupt priority is discussed in [Section 10.0 “Interrupts”](#). BOR is covered in [Section 5.5 “Brown-out Reset \(BOR\)”](#).

FIGURE 5-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: See [Table 5-2](#) for time-out situations.

2: PWRT and OST counters are reset by POR and BOR. See Sections [5.4](#) and [5.5](#).

PIC18(L)F2X/45K50

5.2 Register Definitions: Reset Control

REGISTER 5-1: RCON: RESET CONTROL REGISTER

| R/W-0/0 | R/W-q/u | U-0 | R/W-1/q | R-1/q | R-1/q | R/W-q/u | R/W-0/q |
|---------|-----------------------|-----|---------|-------|-------|--------------------|---------|
| IPEN | SBOREN ⁽¹⁾ | — | RI | TO | PD | POR ⁽²⁾ | BOR |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

-n/n = Value at POR and BOR/Value at all other Resets

x = Bit is unknown

u = unchanged

q = depends on condition

- 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 **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 **TO:** Watchdog Time-out Flag bit
1 = Set by power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred
- bit 2 **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 **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 **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: When CONFIG2L[2:1] = 01, then the SBOREN Reset state is '1'; otherwise, it is '0'.

2: The actual Reset value of POR is determined by the type of device Reset. See the notes following this register and [Section 5.8 “Reset State of Registers”](#) for additional information.

3: See [Table 5-1](#).

Note 1: Brown-out Reset is indicated when BOR is '0' and POR is '1' (assuming that both POR and BOR were set to '1' by firmware immediately after POR).

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

5.3 Master Clear (MCLR)

The MCLR pin provides a method for triggering an external Reset of the device. A Reset is generated by holding the pin low. These devices have a noise filter in the MCLR Reset path which detects and ignores small pulses. An internal weak pull-up is enabled when the pin is configured as the MCLR input.

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

In PIC18(L)F2X/45K50 devices, the MCLR input can be disabled with the MCLRE Configuration bit. When MCLR is disabled, the pin becomes a digital input. See [Section 11.6 “PORTE Registers”](#) for more information.

5.4 Power-on Reset (POR)

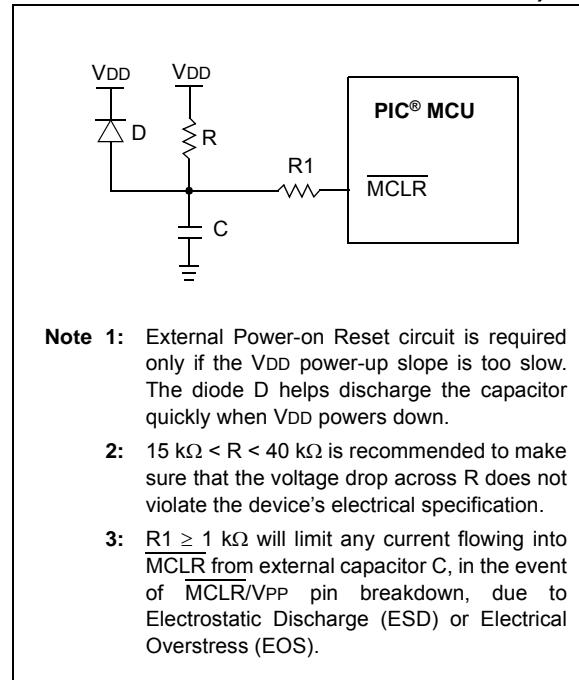
A Power-on Reset pulse is generated on-chip whenever VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.

To take advantage of the POR circuitry either leave the pin floating, or tie the MCLR pin through a resistor to VDD. This will eliminate external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for VDD is specified. For a slow rise time, see [Figure 5-2](#).

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

POR events are captured by the POR bit of the RCON register. The state of the bit is set to '0' whenever a POR occurs; it does not change for any other Reset event. POR is not reset to '1' by any hardware event. To capture multiple events, the user must manually set the bit to '1' by software following any POR.

FIGURE 5-2: 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:** $15\text{ k}\Omega < R < 40\text{ k}\Omega$ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.
- 3:** $R1 \geq 1\text{ k}\Omega$ will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

5.5 Brown-out Reset (BOR)

PIC18(L)F2X/45K50 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 5-1](#).

The BOR threshold is set by the BORV<1:0> bits. If BOR is enabled (any values of BOREN<1:0>, except '00'), any drop of VDD below VBOR for greater than TBOR will reset the device. A Reset may or may not occur if VDD falls below VBOR for less than TBOR. The chip will remain in Brown-out Reset until VDD rises above VBOR.

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

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

The BOR circuit has an output that feeds into the POR circuit and rearms the POR within the operating range of the BOR. This early rearming of the POR ensures that the device will remain in Reset in the event that VDD falls below the operating range of the BOR circuitry.

5.5.1 DETECTING BOR

When BOR is enabled, the BOR bit always resets to '0' on any BOR or POR event. This makes it difficult to determine if a BOR event has occurred just by reading the state of BOR alone. A more reliable method is to simultaneously check the state of both POR and BOR. This assumes that the POR and BOR bits are reset to '1' by software immediately after any POR event. If BOR is '0' while POR is '1', it can be reliably assumed that a BOR event has occurred.

5.5.2 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 the 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.

5.5.3 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.

5.5.4 MINIMUM BOR ENABLE TIME

Enabling the BOR also enables the Fixed Voltage Reference (FVR) when no other peripheral requiring the FVR is active. The BOR becomes active only after the FVR stabilizes. Therefore, to ensure BOR protection, the FVR settling time must be considered when enabling the BOR in software or when the BOR is automatically enabled after waking from Sleep. If the BOR is disabled, in software or by reentering Sleep before the FVR stabilizes, the BOR circuit will not sense a BOR condition. The FVRST bit of the VREFCON0 register can be used to determine FVR stability.

TABLE 5-1: 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. |

5.6 Low-Power BOR (LPBOR)

PIC18(L)F2X/45K50 devices implement a low-power Brown-out Reset circuit (LPBOR). The LPBOR is used to monitor the external VDD pin.

When low voltage is detected, the device is held in Reset. When this occurs, the RCON<0> (BOR) bit is changed to indicate that a BOR reset has occurred. This is the same bit in the RCON register that is set for the traditional BOR.

LPBOR provides the user with a lower power BOR option. In exchange for the lower power, the LPBOR circuit trips at a loose voltage range compared to the traditional BOR voltage trip point options.

LPBOR is enabled by the Configuration bit CONFIG2L<6> (LPBOR). The threshold of the LPBOR is not configurable and its range is specified as parameter D006.

5.7 Device Reset Timers

PIC18(L)F2X/45K50 devices incorporate three separate on-chip timers that help regulate the Power-on Reset process. Their main function is to ensure that the device clock is stable before code is executed. These timers are:

- Power-up Timer (PWRT)
- Oscillator Start-up Timer (OST)
- PLL Lock Time-out

5.7.1 POWER-UP TIMER (PWRT)

The Power-up Timer (PWRT) of PIC18(L)F2X/45K50 devices is an 11-bit counter which uses the INTRC source as the clock input. This yields an approximate time interval of $2048 \times 32 \mu\text{s} = 65.6 \text{ ms}$. While the PWRT is counting, the device is held in Reset.

The power-up time delay depends on the INTRC clock and will vary from chip-to-chip due to temperature and process variation.

The PWRT is enabled by clearing the PWRTE Configuration bit.

5.7.2 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 and HS modes and only on Power-on Reset, or on exit from all power-managed modes that stop the external oscillator.

5.7.3 PLL LOCK TIME-OUT

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

5.7.4 TIME-OUT SEQUENCE

On power-up, the time-out sequence is as follows:

1. After the POR pulse has cleared, PWRT time-out is invoked (if enabled).
2. Then, the OST is activated.

The total time-out will vary based on oscillator configuration and the status of the PWRT. [Figure 5-3](#), [Figure 5-4](#), [Figure 5-5](#), [Figure 5-6](#) and [Figure 5-7](#) all depict time-out sequences on power-up, with the Power-up Timer enabled and the device operating in HS Oscillator mode. Figures [5-3](#) through [5-6](#) also apply to devices operating in XT or LP modes. For devices in RC mode and with the PWRT disabled, on the other hand, there will be no time-out at all.

Since the time-outs occur from the POR pulse, if MCLR is kept low long enough, all time-outs will expire, after which, bringing MCLR high will allow program execution to begin immediately ([Figure 5-5](#)). This is useful for testing purposes or to synchronize more than one PIC MCU device operating in parallel.

PIC18(L)F2X/45K50

TABLE 5-2: TIME-OUT IN VARIOUS SITUATIONS

| Oscillator Configuration | Power-up ⁽²⁾ and Brown-out | | Exit from Power-Managed Mode |
|--------------------------|--|---------------------------------|---------------------------------|
| | PWRTE ^N = 0 | PWRTE ^N = 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⁾ | — | — |
| INTOSC, INTOSCO | 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 5-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD, VDD RISE < TPWRT)

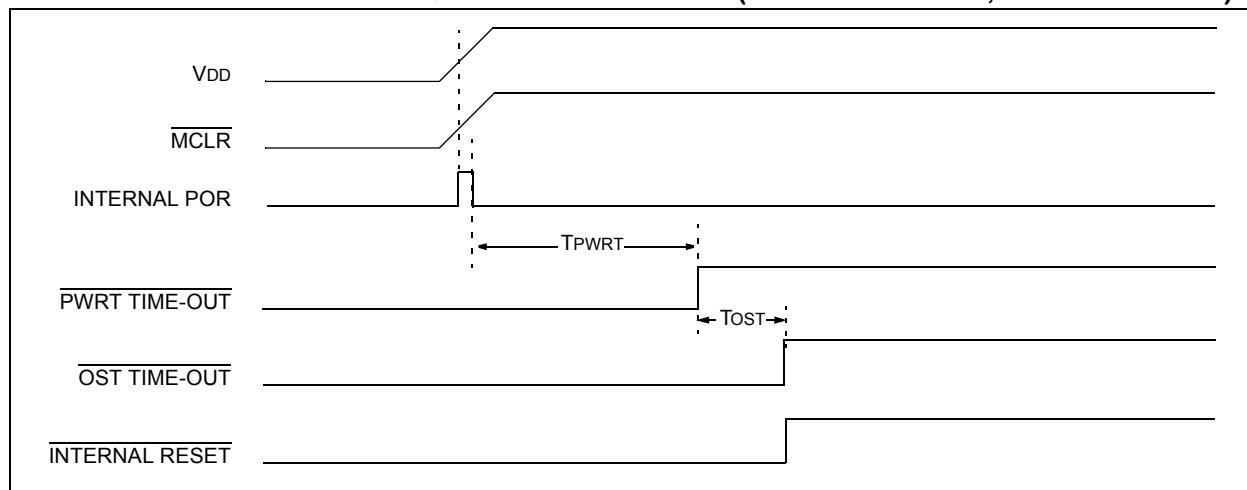


FIGURE 5-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

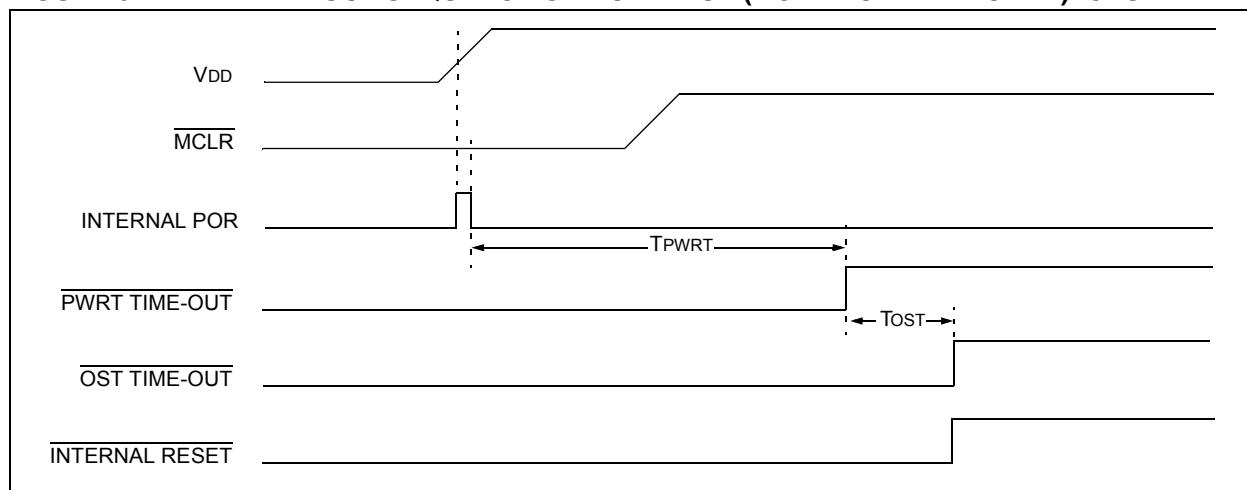


FIGURE 5-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2

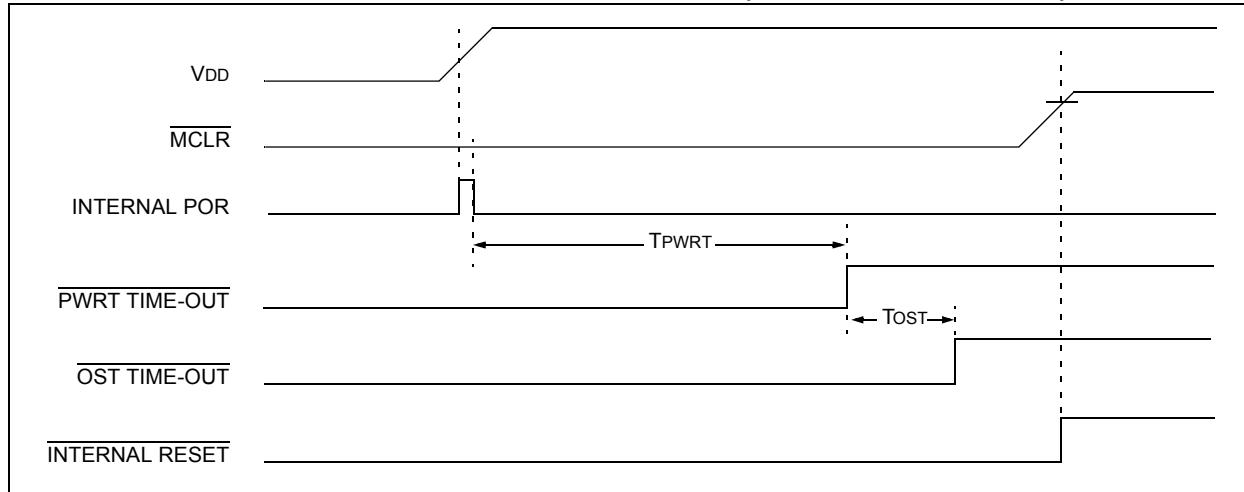
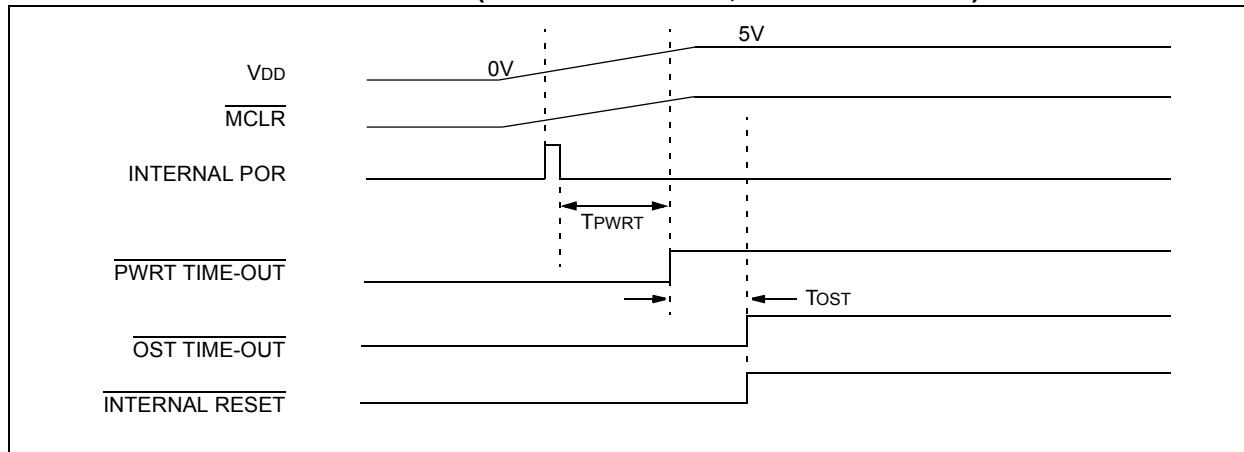
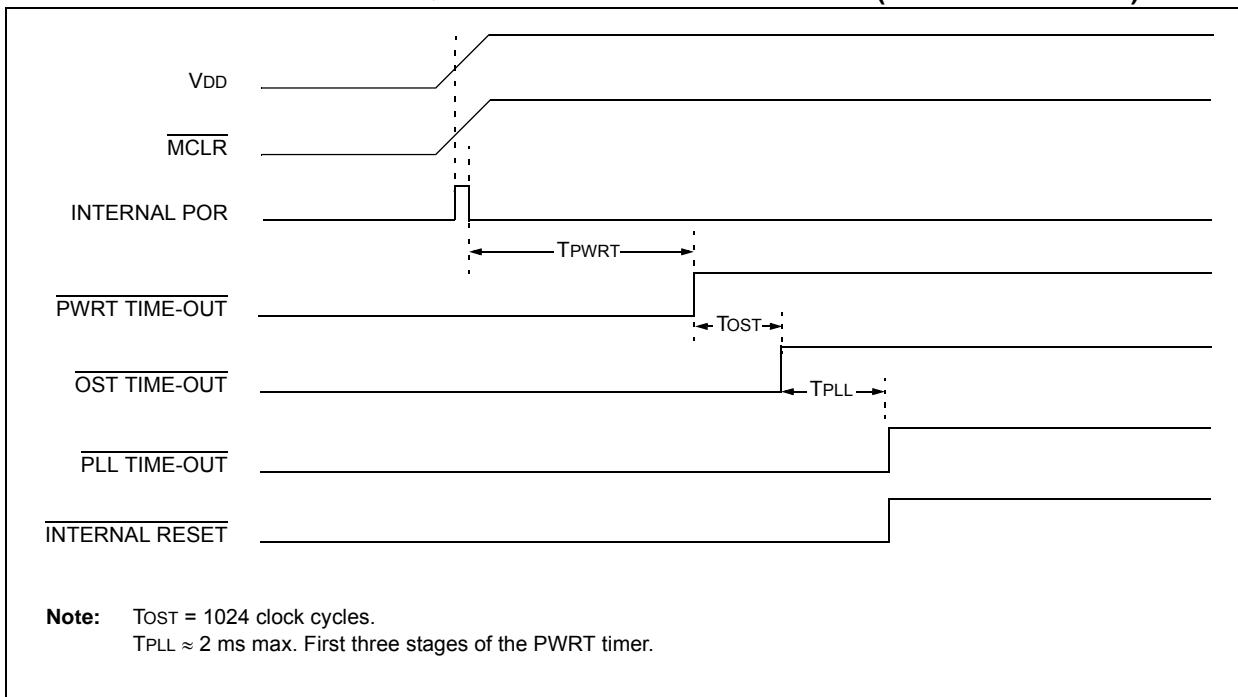


FIGURE 5-6: SLOW RISE TIME (MCLR TIED TO VDD, VDD RISE > TPWRT)



PIC18(L)F2X/45K50

FIGURE 5-7: TIME-OUT SEQUENCE ON POR W/PLL ENABLED (MCLR TIED TO VDD)



5.8 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, RI, \overline{TO} , PD, POR and BOR, are set or cleared differently in different Reset situations, as indicated in [Table 5-3](#). These bits are used by software to determine the nature of the Reset.

TABLE 5-3: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER

| Condition | Program Counter | RCON Register | | | | | | STKPTR Register | |
|--|-----------------------|------------------|----|-----------------|----|-----|-----|-----------------|--------|
| | | SBOREN | RI | \overline{TO} | PD | POR | BOR | STKFUL | 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 |
| MCLR during Power-Managed Run Modes | 0000h | u ⁽²⁾ | u | 1 | u | u | u | u | u |
| 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 |
| 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 SBOREN and unchanged for all other Resets when software BOR is enabled (BOREN<1:0> Configuration bits = 01). Otherwise, the Reset state is ‘0’.

TABLE 5-4: REGISTERS ASSOCIATED WITH RESETS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|--------|--------|--------|-------|-----------------|-----------------|-----------------|------------------|------------------|------------------|
| RCON | IPEN | SBOREN | — | \overline{RI} | \overline{TO} | \overline{PD} | \overline{POR} | \overline{BOR} | 64 |
| STKPTR | STKFUL | STKUNF | — | STKPTR<4:0> | | | | | 76 |

Legend: — = unimplemented locations, read as ‘0’. Shaded bits are not used for Resets.

[Table 6-2](#) describes the Reset states for all of the Special Function Registers. The table identifies differences between Power-On Reset (POR)/Brown-Out Reset (BOR) and all other Resets, (i.e., Master Clear, WDT Resets, STKFUL, STKUNF, etc.). Additionally, the table identifies register bits that are changed when the device receives a wake-up from WDT or other interrupts.

PIC18(L)F2X/45K50

TABLE 5-5: CONFIGURATION REGISTERS ASSOCIATED WITH RESETS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|------------|------------|-------|------------|--------|------------------|
| CONFIG2L | — | LPBOR | — | BORV<1:0> | BOREN<1:0> | | PWRDEN | 374 | |
| CONFIG2H | — | — | | WDTPS<3:0> | | | WDTEN<1:0> | 375 | |
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |
| CONFIG4L | DEBUG | XINST | ICPRT | — | — | LVP | — | STRVEN | 377 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for Resets.

6.0 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 buses; this allows for concurrent access of the two memory spaces. The data EEPROM, for practical purposes, can be regarded as a peripheral device, since it is addressed and accessed through a set of control registers.

Additional detailed information on the operation of the Flash program memory is provided in [Section 7.0 “Flash Program Memory”](#). Data EEPROM is discussed separately in [Section 8.0 “Data EEPROM Memory”](#).

6.1 Program Memory Organization

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

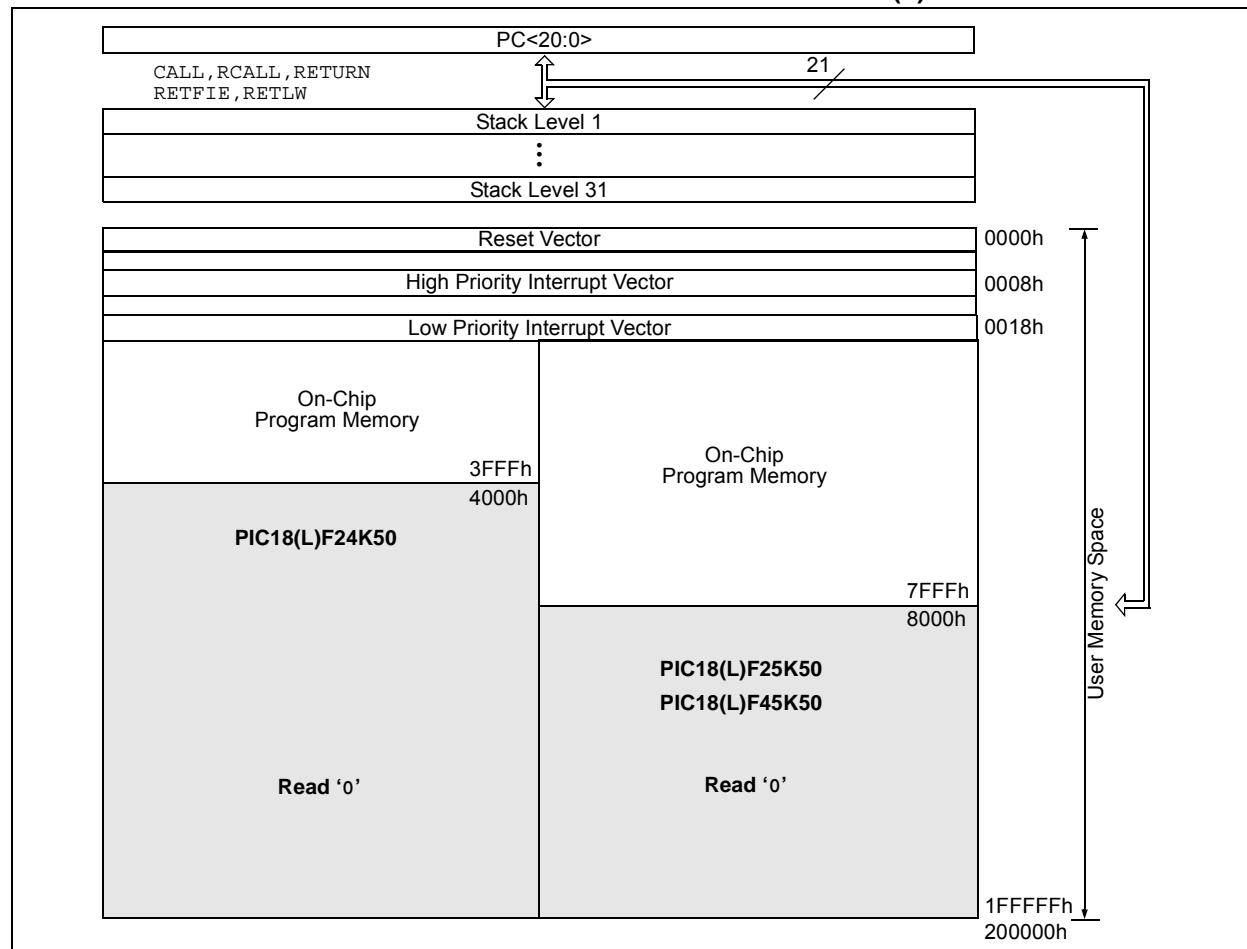
This family of devices contain the following:

- PIC18(L)F24K50: 16 Kbytes of Flash memory, up to 8,192 single-word instructions
- PIC18(L)F25K50, PIC18(L)F45K50: 32 Kbytes of Flash memory, up to 16,384 single-word instructions

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

The program memory map for PIC18(L)F2X/45K50 devices is shown in [Figure 6-1](#). Memory block details are shown in [Figure 21-2](#).

FIGURE 6-1: PROGRAM MEMORY MAP AND STACK FOR PIC18(L)F2X/45K50 DEVICES



PIC18(L)F2X/45K50

6.1.1 PROGRAM COUNTER

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

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see [Section 6.2.3.1 “Computed GOTO”](#)).

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

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

6.1.2 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.

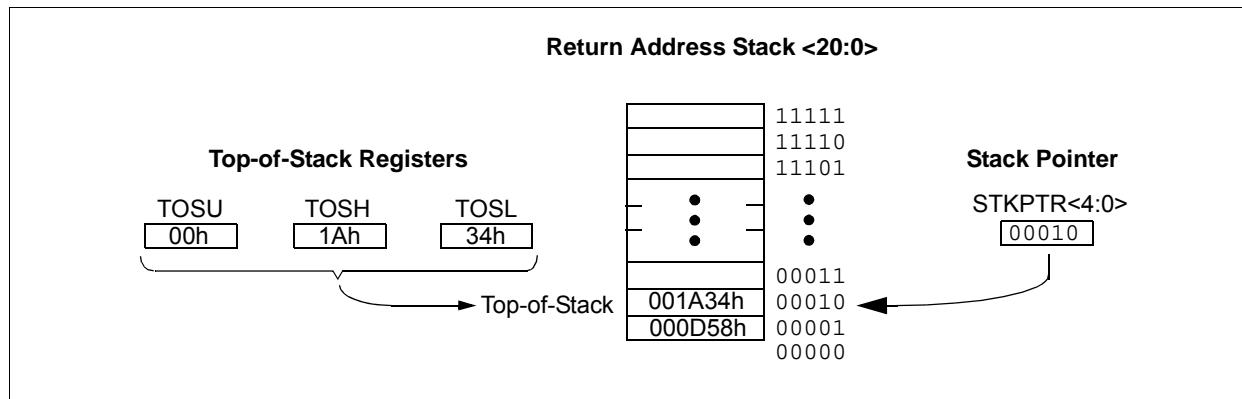
The Stack Pointer is initialized to ‘00000’ after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of ‘00000’; this is only a Reset value. Status bits indicate if the stack is full or has overflowed or has underflowed.

6.1.2.1 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 6-2](#)). 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 (GIE) bits while accessing the stack to prevent inadvertent stack corruption.

FIGURE 6-2: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



6.1.2.2 Return Stack Pointer (STKPTR)

The STKPTR register ([Register 6-1](#)) contains the Stack Pointer value, the STKFUL (stack full) Status bit and the STKUNF (Stack Underflow) Status bits. The value of the Stack Pointer can be 0 through 31. The Stack Pointer increments before values are pushed onto the stack and decrements after values are popped off the stack. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for return stack maintenance.

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

The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to [Section 26.1 “Configuration Bits”](#) for a description of the device Configuration bits.) If STVREN is set (default), the 31st push will push the $(PC + 2)$ value onto the stack, set the STKFUL bit and reset the device. The STKFUL bit will remain set and the Stack Pointer will be set to zero.

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

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

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

6.1.2.3 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.

6.2 Register Definitions: Stack Pointer

REGISTER 6-1: STKPTR: STACK POINTER REGISTER

| R/C-0 | R/C-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-----------------------|-----------------------|-----|-------|-------------|-------|-------|-------|
| STKFUL ⁽¹⁾ | STKUNF ⁽¹⁾ | — | | STKPTR<4:0> | | | |
| 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 **STKFUL:** Stack Full 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 **STKPTR<4:0>:** Stack Pointer Location bits

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

6.2.1 STACK FULL AND UNDERFLOW RESETS

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

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 6-1 shows a source code example that uses the fast register stack during a subroutine call and return.

6.2.2 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.

EXAMPLE 6-1: FAST REGISTER STACK CODE EXAMPLE

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

6.2.3 LOOK-UP TABLES IN PROGRAM MEMORY

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

- Computed GOTO
- Table Reads

6.2.3.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in [Example 6-2](#).

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

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

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

EXAMPLE 6-2: COMPUTED GOTO USING AN OFFSET VALUE

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

6.2.3.2 Table Reads and Table Writes

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

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

Table read and table write operations are discussed further in [Section 7.1 “Table Reads and Table Writes”](#).

PIC18(L)F2X/45K50

6.3 PIC18 Instruction Cycle

6.3.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the program counter is incremented on every Q1; the instruction is fetched from the program memory and latched into the instruction register during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in [Figure 6-3](#).

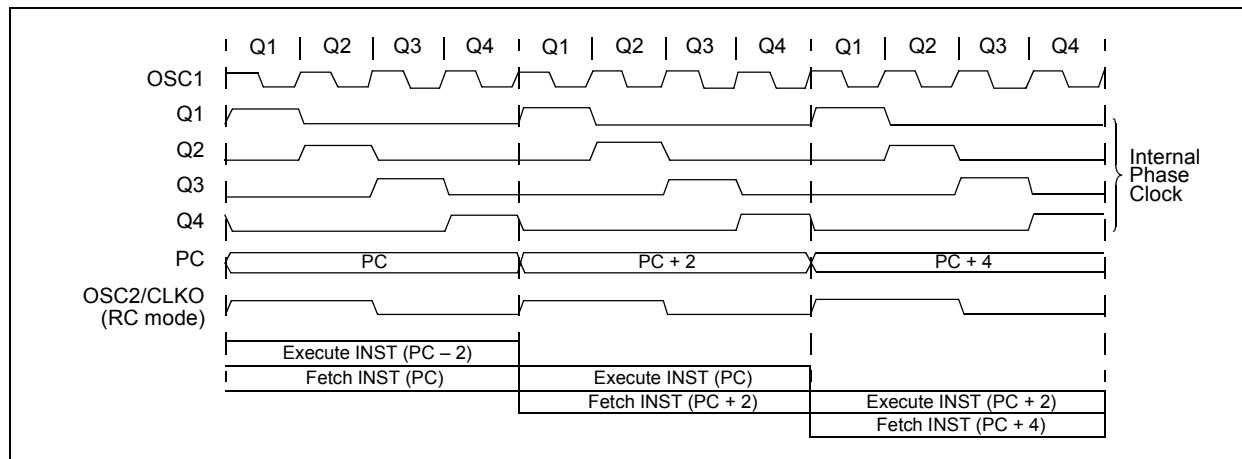
6.3.2 INSTRUCTION FLOW/PIPELINING

An “Instruction Cycle” consists of four Q cycles: Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), then two cycles are required to complete the instruction ([Example 6-3](#)).

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

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

FIGURE 6-3: CLOCK/INSTRUCTION CYCLE



EXAMPLE 6-3: 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.

6.3.3 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 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 two and the LSb will always read '0' (see [Section 6.1.1 "Program Counter"](#)).

[Figure 6-4](#) 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 6-4](#) 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 27.0 "Instruction Set Summary"](#) provides further details of the instruction set.

FIGURE 6-4: INSTRUCTIONS IN PROGRAM MEMORY

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

6.3.4 TWO-WORD INSTRUCTIONS

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

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

and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a NOP is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. [Example 6-4](#) shows how this works.

Note: See [Section 6.8 "PIC18 Instruction Execution and the Extended Instruction Set"](#) for information on two-word instructions in the extended instruction set.

EXAMPLE 6-4: TWO-WORD INSTRUCTIONS

CASE 1:

| Object Code | Source Code |
|---------------------|---------------------------------------|
| 0110 0110 0000 0000 | TSTFSZ REG1 ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF REG1, REG2 ; No, skip this word |
| 1111 0100 0101 0110 | ; Execute this word as a NOP |
| 0010 0100 0000 0000 | ADDWF REG3 ; continue code |

CASE 2:

| Object Code | Source Code |
|---------------------|---|
| 0110 0110 0000 0000 | TSTFSZ REG1 ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF REG1, REG2 ; Yes, execute this word |
| 1111 0100 0101 0110 | ; 2nd word of instruction |
| 0010 0100 0000 0000 | ADDWF REG3 ; continue code |

6.4 Data Memory Organization

Note: The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See [Section 6.7 “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. Figures [6-5](#) through [6-7](#) show the data memory organization for the PIC18(L)F2X/45K50 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 6.4.3 “Access Bank”](#) provides a detailed description of the Access RAM.

6.4.1 USB RAM

Banks 4 through 7 of the data memory are actually mapped to special dual port RAM. When the USB module is disabled, the GPRs in these banks are used like any other GPR in the data memory space.

When the USB module is enabled, the memory in these banks is allocated as buffer RAM for USB operation. This area is shared between the microcontroller core and the USB Serial Interface Engine (SIE) and is used to transfer data directly between the two. It is theoretically possible to use the areas of USB RAM that are not allocated as USB buffers for normal scratchpad memory or other variable storage. In practice, the dynamic nature of buffer allocation makes this risky, at best. Additionally, Bank 4 is used for USB buffer descriptor tables when the module is enabled and should not be used for any other purposes during that time. Additional information on USB RAM and buffer operation is provided in [Section 24.0 “Universal Serial Bus \(USB\)”](#).

6.4.2 BANK SELECT REGISTER (BSR)

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

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the four Most Significant bits of a location's address; the instruction itself includes the eight Least Significant bits. Only the four lower bits of the BSR are implemented ($\text{BSR}\langle 3:0 \rangle$). 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.

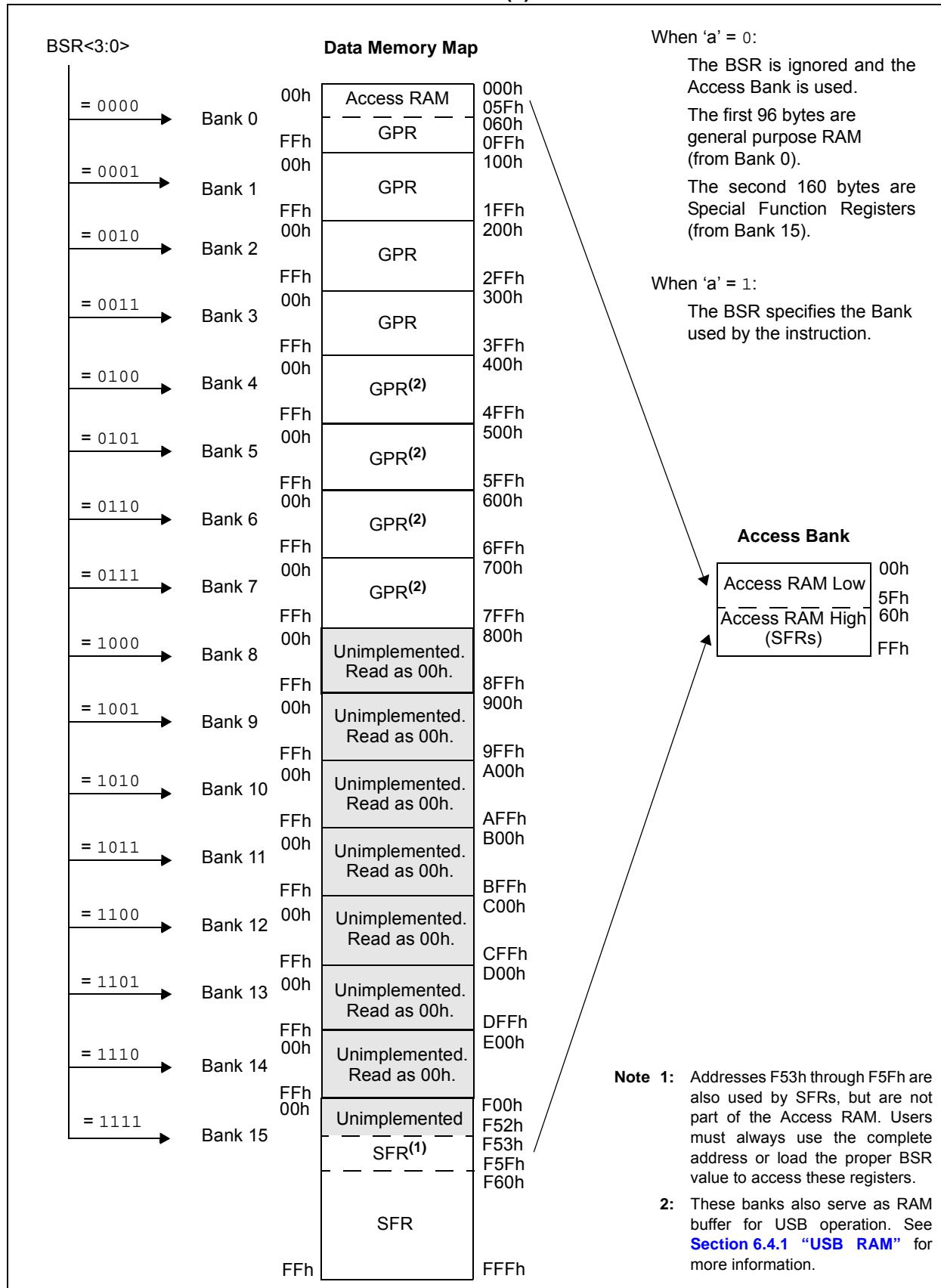
The value of the BSR indicates the bank in data memory; the eight bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figures [6-5](#) through [6-7](#).

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

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory maps in Figures [6-5](#) through [6-7](#) indicate which banks are implemented.

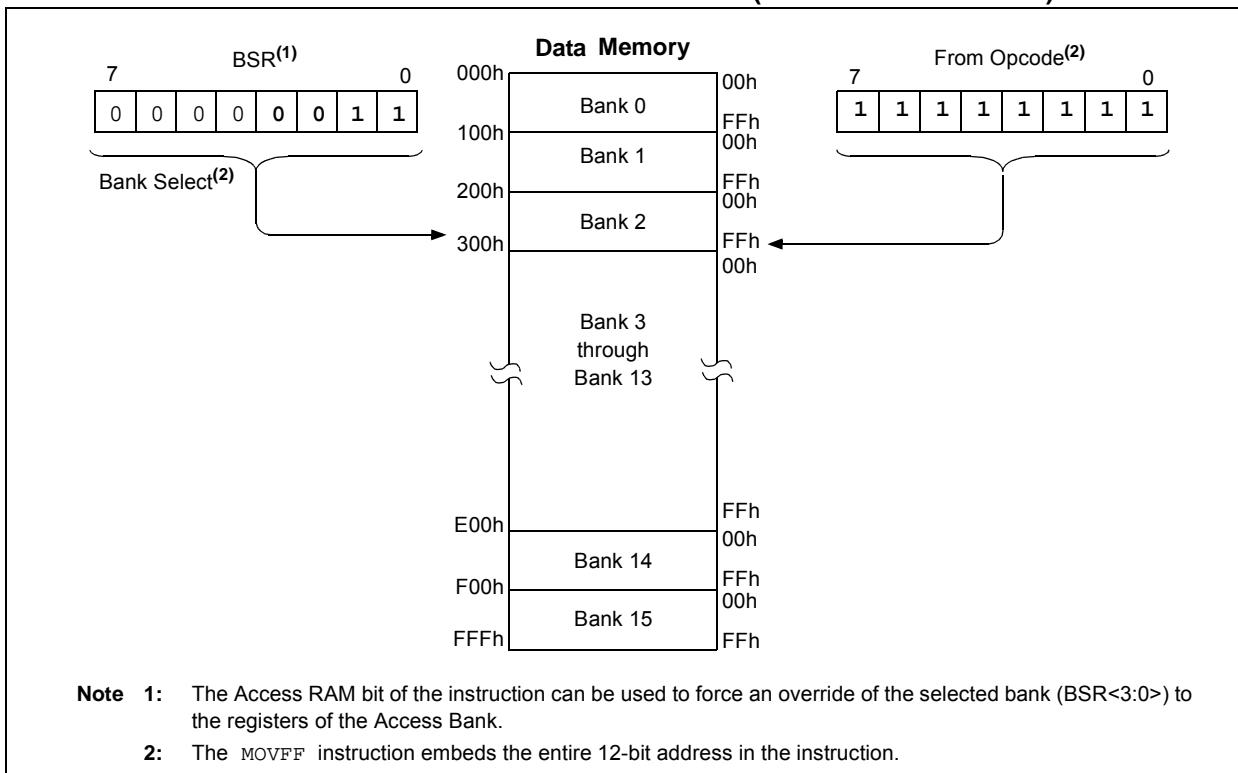
In the core PIC18 instruction set, only the `MOVFF` instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the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

FIGURE 6-5: DATA MEMORY MAP FOR PIC18(L)F2X/45K50 DEVICES



PIC18(L)F2X/45K50

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



6.4.3 ACCESS BANK

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

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Block 15. The lower half is known as the “Access RAM” and is composed of GPRs. This upper half is also where the device’s SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figures 6-5 through 6-7).

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 6.7.3 “Mapping the Access Bank in Indexed Literal Offset Mode”](#).

6.4.4 GENERAL PURPOSE REGISTER FILE

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

6.4.5 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top portion of Bank 15 (F53h to FFFh). A list of these registers is given in [Table 6-1](#) and [Table 6-2](#).

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.

PIC18(L)F2X/45K50

TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F2X/45K50 DEVICES

| Address | Name | Address | Name | Address | Name | Address | Name | Address | Name |
|---------|-------------------------|---------|------------------|---------|-----------------------|---------|----------------------|---------|------------------------|
| FFFh | TOSU | FD7h | TMR0H | FAFh | SPBRG1 | F87h | IOCC | F5Fh | ANSELE ⁽³⁾ |
| FFEh | TOSH | FD6h | TMR0L | FAEh | RCREG1 | F86h | IOCB | F5Eh | ANSEL(3) |
| FFDh | TOSL | FD5h | T0CON | FADh | TXREG1 | F85h | WPUB | F5Dh | ANSELC |
| FFCh | STKPTR | FD4h | — ⁽²⁾ | FACh | TXSTA1 | F84h | PORTE | F5Ch | ANSELB |
| FFBh | PCLATU | FD3h | OSCCON | FABh | RCSTA1 | F83h | PORTD ⁽³⁾ | F5Bh | ANSELA |
| FFAh | PCLATH | FD2h | OSCCON2 | FAAh | — | F82h | PORTC | F5Ah | VREGCON ⁽⁴⁾ |
| FF9h | PCL | FD1h | WDTCON | FA9h | EEADR | F81h | PORTB | F59h | CCPTMRS |
| FF8h | TBLPTRU | FD0h | RCON | FA8h | EEDATA | F80h | PORTA | F58h | SRCON0 |
| FF7h | TBLPTRH | FCFh | TMR1H | FA7h | EECON2 ⁽¹⁾ | F7Fh | PMD1 | F57h | SRCON1 |
| FF6h | TBLPTRL | FCEh | TMR1L | FA6h | EECON1 | F7Eh | PMD0 | F56h | — |
| FF5h | TABLAT | FCDh | T1CON | FA5h | IPR3 | F7Dh | VREFCON0 | F55h | — |
| FF4h | PRODH | FCCh | T1GCON | FA4h | PIR3 | F7Ch | VREFCON1 | F54h | — |
| FF3h | PRODL | FCBh | SSP1CON3 | FA3h | PIE3 | F7Bh | VREFCON2 | F53h | — |
| FF2h | INTCON | FCAh | SSP1MSK | FA2h | IPR2 | F7Ah | SLRCON | F52h | |
| FF1h | INTCON2 | FC9h | SSP1BUF | FA1h | PIR2 | F79h | UEP15 | F51h | |
| FF0h | INTCON3 | FC8h | SSP1ADD | FA0h | PIE2 | F78h | UEP14 | F50h | |
| FEFh | INDF0 ⁽¹⁾ | FC7h | SSP1STAT | F9Fh | IPR1 | F77h | UEP13 | F4Fh | |
| FEEh | POSTINC0 ⁽¹⁾ | FC6h | SSP1CON1 | F9Eh | PIR1 | F76h | UEP12 | F4Eh | |
| FEDh | POSTDEC0 ⁽¹⁾ | FC5h | SSP1CON2 | F9Dh | PIE1 | F75h | UEP11 | F4Dh | |
| FECh | PREINC0 ⁽¹⁾ | FC4h | ADRESH | F9Ch | HLVDCON | F74h | UEP10 | F4Ch | |
| FEBh | PLUSW0 ⁽¹⁾ | FC3h | ADRESL | F9Bh | OSCTUNE | F73h | UEP9 | F4Bh | |
| FEAh | FSR0H | FC2h | ADCON0 | F9Ah | CM2CON1 | F72h | UEP8 | F4Ah | |
| FE9h | FSR0L | FC1h | ADCON1 | F99h | CM2CON0 | F71h | UEP7 | F49h | |
| FE8h | WREG | FC0h | ADCON2 | F98h | CM1CON0 | F70h | UEP6 | F48h | |
| FE7h | INDF1 ⁽¹⁾ | FBFh | CCPR1H | F97h | CCP2CON | F6Fh | UEP5 | F47h | General Purpose RAM |
| FE6h | POSTINC1 ⁽¹⁾ | FBEh | CCPR1L | F96h | TRISE ⁽³⁾ | F6Eh | UEP4 | F46h | |
| FE5h | POSTDEC1 ⁽¹⁾ | FBDh | CCP1CON | F95h | TRISD ⁽³⁾ | F6Dh | UEP3 | F45h | |
| FE4h | PREINC1 ⁽¹⁾ | FBCh | TMR2 | F94h | TRISC | F6Ch | UEP2 | F44h | |
| FE3h | PLUSW1 ⁽¹⁾ | FBBh | PR2 | F93h | TRISB | F6Bh | UEP1 | F43h | |
| FE2h | FSR1H | FBAh | T2CON | F92h | TRISA | F6Ah | UEP0 | F42h | |
| FE1h | FSR1L | FB9h | PSTR1CON | F91h | CCPR2H | F69h | UFRMH | F41h | |
| FE0h | BSR | FB8h | BAUDCON1 | F90h | CCPR2L | F68h | UFRML | F40h | |
| FDfh | INDF2 ⁽¹⁾ | FB7h | PWM1CON | F8Fh | CTMUCONH | F67h | UEIR | F3Fh | |
| FDEh | POSTINC2 ⁽¹⁾ | FB6h | ECCP1AS | F8Eh | CTMUCONL | F66h | UEIE | F3Eh | |
| FDDh | POSTDEC2 ⁽¹⁾ | FB5h | STCON | F8Dh | LATE ⁽³⁾ | F65h | UIR | F3Dh | |
| FDCh | PREINC2 ⁽¹⁾ | FB4h | T3GCON | F8Ch | LATD ⁽³⁾ | F64h | UIE | F3Ch | |
| FDBh | PLUSW2 ⁽¹⁾ | FB3h | TMR3H | F8Bh | LATC | F63h | UADDR | F3Bh | |
| FDAh | FSR2H | FB2h | TMR3L | F8Ah | LATB | F62h | UCNFG | F3Ah | |
| FD9h | FSR2L | FB1h | T3CON | F89h | LATA | F61h | USTAT | F39h | |
| FD8h | STATUS | FB0h | SPBRGH1 | F88h | CTMUICONH | F60h | UCTRL | F38h | |

Note 1: This is not a physical register.

2: Unimplemented registers are read as '0'.

3: PIC18(L)F45K50 device only.

4: F devices only.

TABLE 6-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/45K50 DEVICES

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | |
|---------|----------|--|-----------|---------|---|---|-----------|----------|--------|-------------------|-----------|
| FFFh | TOSU | — | — | — | | Top-of-Stack, Upper Byte (TOS<20:16>) | | | | | |
| FFEh | TOSH | Top-of-Stack, High Byte (TOS<15:8>) | | | | | | | | 0000 0000 | |
| FFDh | Tosl | Top-of-Stack, Low Byte (TOS<7:0>) | | | | | | | | 0000 0000 | |
| FFCh | STKPTR | STKFUL | STKUNF | — | | STKPTR<4:0> | | | | | |
| FFBh | PCLATU | — | — | — | | Holding Register for PC<20:16> | | | | | |
| FFAh | PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000 0000 | |
| FF9h | PCL | Holding Register for PC<7:0> | | | | | | | | 0000 0000 | |
| FF8h | TBLPTRU | — | — | | Program Memory Table Pointer Upper Byte (TBLPTR<21:16>) | | | | | --00 0000 | |
| FF7h | TBLPTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 | |
| FF6h | TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 | |
| FF5h | TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 | |
| FF4h | PRODH | Product Register, High Byte | | | | | | | | xxxxx xxxx | |
| FF3h | PRODL | Product Register, Low Byte | | | | | | | | xxxxx xxxx | |
| FF2h | INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 0000 000x | |
| FF1h | INTCON2 | RBPU | INTEGD0 | INTEGD1 | INTEGD2 | — | TMR0IP | — | IOCIP | 1111 -1-1 | |
| FF0h | INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 11-0 0-00 | |
| FEFh | INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) | | | | | | | | ----- | |
| FEEh | POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | ----- | |
| FEDh | POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) | | | | | | | | ----- | |
| FECh | PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) | | | | | | | | ----- | |
| FEBh | PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W | | | | | | | | ----- | |
| FEAh | FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0, High Byte | | | | | |
| FE9h | FSR0L | Indirect Data Memory Address Pointer 0, Low Byte | | | | | | | | xxxxx xxxx | |
| FE8h | WREG | Working Register | | | | | | | | xxxxx xxxx | |
| FE7h | INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) | | | | | | | | ----- | |
| FE6h | POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | ----- | |
| FE5h | POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | ----- | |
| FE4h | PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) | | | | | | | | ----- | |
| FE3h | PLUSW1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W | | | | | | | | ----- | |
| FE2h | FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1, High Byte | | | | | |
| FE1h | FSR1L | Indirect Data Memory Address Pointer 1, Low Byte | | | | | | | | xxxxx xxxx | |
| FE0h | BSR | — | — | — | — | Bank Select Register | | | | | |
| FDfh | INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) | | | | | | | | ----- | |
| FDEh | POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | ----- | |
| FDDh | POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) | | | | | | | | ----- | |
| FDCh | PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) | | | | | | | | ----- | |
| FDBh | PLUSW2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W | | | | | | | | ----- | |
| FDAh | FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2, High Byte | | | | | |
| FD9h | FSR2L | Indirect Data Memory Address Pointer 2, Low Byte | | | | | | | | xxxxx xxxx | |
| FD8h | STATUS | — | — | — | N | OV | Z | DC | C | ---x xxxx | |
| FD7h | TMR0H | Timer0 Register, High Byte | | | | | | | | 0000 0000 | |
| FD6h | TMR0L | Timer0 Register, Low Byte | | | | | | | | xxxxx xxxx | |
| FD5h | TOCON | TMR0ON | T08BIT | TOCS | T0SE | PSA | T0PS<2:0> | | | | |
| FD3h | OSCCON | IDLEN | IRCF<2:0> | | | OSTS | HFIOPS | SCS<1:0> | | | 0011 q000 |
| FD2h | OSCCON2 | PLLRDY | SOSCRUN | INTSRC | PLLEN | SOSCIGO | PRISD | HFIOPFR | LFIOPS | | 0000 0100 |
| FD1h | WDTCON | — | — | — | — | — | — | — | SWDTEN | | ----- |
| FD0h | RCON | IPEN | SBOREN | — | RI | TO | PD | POR | BOR | | 01-1 1100 |

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

Note 1: PIC18(L)F45K50 devices only.

2: PIC18(L)F2XK50 devices only.

PIC18(L)F2X/45K50

TABLE 6-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/45K50 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | |
|---------|----------|---|--------------|-------------|----------|----------------|---------------------|-------------|---------|-------------------|-----------|
| FCFh | TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | |
| FCEh | TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | |
| FCDh | T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | SOSCEN | $\overline{T1SYNC}$ | RD16 | TMR1ON | 0000 0000 | |
| FCCh | T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/ DONE | T1GVAL | T1GSS<1:0> | | 0000 0x00 | |
| FCBh | SSP1CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 0000 0000 | |
| FCAh | SSP1MSK | SSP1 Mask Register bits | | | | | | | | 1111 1111 | |
| FC9h | SSP1BUF | SSP1 Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | |
| FC8h | SSP1ADD | SSP1 Address Register in I ² C™ Slave Mode. SSP1 Baud Rate Reload Register in I ² C Master Mode | | | | | | | | 0000 0000 | |
| FC7h | SSP1STAT | SMP | CKE | D/A | P | S | R/W | UA | BF | 0000 0000 | |
| FC6h | SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 0000 0000 | |
| FC5h | SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | |
| FC4h | ADRESH | A/D Result, High Byte | | | | | | | | xxxx xxxx | |
| FC3h | ADRESL | A/D Result, Low Byte | | | | | | | | xxxx xxxx | |
| FC2h | ADCON0 | — | CHS<4:0> | | | | GO/DONE | ADON | | -000 0000 | |
| FC1h | ADCON1 | TRIGSEL | — | — | — | PVCFG<1:0> | | NVCFG<1:0> | | 0--- 0000 | |
| FC0h | ADCON2 | ADFM | — | ACQT<2:0> | | | ADCS<2:0> | | | 0-00 0000 | |
| FBFh | CCPR1H | Capture/Compare/PWM Register 1, High Byte | | | | | | | | xxxx xxxx | |
| FBEh | CCPR1L | Capture/Compare/PWM Register 1, Low Byte | | | | | | | | xxxx xxxx | |
| FBDh | CCP1CON | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | | 0000 0000 | |
| FBCh | TMR2 | Timer2 Register | | | | | | | | 0000 0000 | |
| FBBh | PR2 | Timer2 Period Register | | | | | | | | 1111 1111 | |
| FBAh | T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | -000 0000 | |
| FB9h | PSTR1CON | — | — | — | STR1SYNC | STR1D | STR1C | STR1B | STR1A | ---0 0001 | |
| FB8h | BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 0100 0-00 | |
| FB7h | PWM1CON | P1RSEN | P1DC<6:0> | | | | | | | | 0000 0000 |
| FB6h | ECCP1AS | ECCP1ASE | ECCP1AS<2:0> | | | PSS1AC<1:0> | | PSS1BD<1:0> | | | 0000 0000 |
| FB5h | ACTCON | ACTEN | ACTUD | — | ACTSRC | ACTLOCK | — | ACTORS | — | 00-0 0-0- | |
| FB4h | T3GCON | TMR3GE | T3GPOL | T3GTM | T3GSPM | T3GGO/ DONE | T3GVAL | T3GSS<1:0> | | 0000 0x00 | |
| FB3h | TMR3H | Holding Register for the Most Significant Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | |
| FB2h | TMR3L | Holding Register for the Least Significant Byte of the 16-bit TMR3 Register | | | | | | | | xxxx xxxx | |
| FB1h | T3CON | TMR3CS<1:0> | | T3CKPS<1:0> | | SOSCEN | $\overline{T3SYNC}$ | RD16 | TMR3ON | 0000 0000 | |
| FB0h | SPBRGH1 | EUSART Baud Rate Generator, High Byte | | | | | | | | 0000 0000 | |
| FAFh | SPBRG1 | EUSART Baud Rate Generator, Low Byte | | | | | | | | 0000 0000 | |
| FAEh | RCREG1 | EUSART Receive Register | | | | | | | | 0000 0000 | |
| FADh | TXREG1 | EUSART Transmit Register | | | | | | | | 0000 0000 | |
| FACh | TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDDB | BRGH | TRMT | TX9D | 0000 0010 | |
| FABh | RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | |
| FA9h | EEADR | EEADR<7:0> | | | | | | | | 0000 0000 | |
| FA8h | EEDATA | EEPROM Data Register | | | | | | | | 0000 0000 | |
| FA7h | EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | ----- ----- | |
| FA6h | EECON1 | EEPGD | CFGs | — | FREE | WRERR | WREN | WR | RD | xx-0 x000 | |
| FA5h | IPR3 | — | — | — | — | CTMU1P | USBIP | TMR3GIP | TMR1GIP | 0000 1111 | |
| FA4h | PIR3 | — | — | — | — | CTMU1F | USBIF | TMR3GIF | TMR1GIF | 0000 0000 | |
| FA3h | PIE3 | — | — | — | — | CTMU1E | USBIE | TMR3GIE | TMR1GIE | 0000 0000 | |
| FA2h | IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 1111 1111 | |
| FA1h | PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 0000 0000 | |
| FA0h | PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 0000 0000 | |

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

Note 1: PIC18(L)F45K50 devices only.

2: PIC18(L)F2XK50 devices only.

TABLE 6-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/45K50 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|----------------------|---|--------------|-----------|---------|--------------|-----------------------|-----------------------|-----------------------|-------------------|
| F9Fh | IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 |
| F9Eh | PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 |
| F9Dh | PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 |
| F9Ch | HLVDCON | VDIRMAG | BGVST | IRVST | HLVDEN | | | | | 0000 0000 |
| F9Bh | OSCTUNE | SPLLMULT | | | | TUN<6:0> | | | | 0000 0000 |
| F9Ah | CM2CON1 | MC1OUT | MC2OUT | C1RSEL | C2RSEL | C1HYS | C2HYS | C1SYNC | C2SYNC | 0000 0000 |
| F99h | CM2CON0 | C2ON | C2OUT | C2OE | C2POL | C2SP | C2R | C2CH<1:0> | | 0000 1000 |
| F98h | CM1CON0 | C1ON | C1OUT | C1OE | C1POL | C1SP | C1R | C1CH<1:0> | | 0000 1000 |
| F97h | CCP2CON | — | — | DC2B<1:0> | | | CCP2M<3:0> | | | --00 0000 |
| F96h | TRISE | WPUE3 | — | — | — | — | TRISE2 ⁽¹⁾ | TRISE1 ⁽¹⁾ | TRISE0 ⁽¹⁾ | 1--- -111 |
| F95h | TRISD ⁽¹⁾ | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 1111 1111 |
| F94h | TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISCO | 1111 -111 |
| F93h | TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 1111 1111 |
| F92h | TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 1111 1111 |
| F91h | CCPR2H | Capture/Compare/PWM Register 2, High Byte | | | | | | | | xxxxx xxxx |
| F90h | CCPR2L | Capture/Compare/PWM Register 2, Low Byte | | | | | | | | xxxxx xxxx |
| F8Fh | CTMUCONH | CTMUEN | — | CTMUSIDL | TGEN | EDGEN | EDGSEQEN | IDISSEN | CTTRIG | 0-00 0000 |
| F8Eh | CTMUCONL | EDG2POL | EDG2SEL<1:0> | | EDG1POL | EDG1SEL<1:0> | | EDG2STAT | EDG1STAT | 0000 00xx |
| F8Dh | LATE ⁽¹⁾ | — | — | — | — | — | LATE2 | LATE1 | LATE0 | ---- -xxx |
| F8Ch | LATD ⁽¹⁾ | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | xxxxx xxxx |
| F8Bh | LATC | LATC7 | LATC6 | — | — | — | LATC2 | LATC1 | LATC0 | xxxxx -xxx |
| F8Ah | LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | xxxxx xxxx |
| F89h | LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | xxxxx xxxx |
| F88h | CTMUICON | ITRIM<5:0> | | | | | | IRNG<1:0> | | 0000 0000 |
| F87h | IOCC | IOCC7 | IOCC6 | IOCC5 | IOCC4 | — | IOCC2 | IOCC1 | IOCC0 | 0000 -000 |
| F86h | IOCB | IOCB7 | IOCB6 | IOCB5 | IOCB4 | — | — | — | — | 0000 ---- |
| F85h | WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 | 1111 1111 |
| F84h | PORTE ⁽²⁾ | — | — | — | — | RE3 | — | — | — | ---- x--- |
| | PORTE ⁽¹⁾ | — | — | — | — | RE3 | RE2 | RE1 | RE0 | ---- xxxx |
| F83h | PORTD ⁽¹⁾ | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxxx xxxx |
| F82h | PORTC | RC7 | RC6 | — | — | — | RC2 | RC1 | RC0 | xx-- -xxx |
| F81h | PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | xxxxx xxxx |
| F80h | PORTA | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | xxxxx xxxx |
| F7Fh | PMD1 | — | MSSPM | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | -000 0000 |
| F7Eh | PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | -000 -000 |
| F7Dh | VREFCON0 | FVREN | FVRST | FVRS<1:0> | | — | — | — | — | 0001 00-- |
| F7Ch | VREFCON1 | DACEN | DACLPS | DACOE | — | DACPSS<1:0> | | — | DACNSS | 000- 00-0 |
| F7Bh | VREFCON2 | — | — | — | | DACR<4:0> | | | | ---0 0000 |
| F7Ah | SLRCON | — | — | — | SLRE | SLRD | SLRC | SLRB | SLRA | ---1 1111 |
| F79h | UEP15 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F78h | UEP14 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F77h | UEP13 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F76h | UEP12 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F75h | UEP11 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F74h | UEP10 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F73h | UEP9 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F72h | UEP8 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F71h | UEP7 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F70h | UEP6 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |

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

Note 1: PIC18(L)F45K50 devices only.

2: PIC18(L)F2XK50 devices only.

PIC18(L)F2X/45K50

TABLE 6-2: REGISTER FILE SUMMARY FOR PIC18(L)F2X/45K50 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|---------|-------|--------|------------|-----------|-----------|-----------|-------------|---------|-------------------|
| F6Fh | UEP5 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F6Eh | UEP4 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F6Dh | UEP3 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F6Ch | UEP2 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F6Bh | UEP1 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F6Ah | UEP0 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | ---0 0000 |
| F69h | UFRMH | — | — | — | — | — | FRM<10:8> | | | ---- -xxx |
| F68h | UFRML | | | | | FRM<7:0> | | | | xxxx xxxx |
| F67h | UEIR | BTSEF | — | — | BTOEF | DFN8EF | CRC16EF | CRC5EF | PIDEF | 0--0 0000 |
| F66h | UEIE | BTSEE | — | — | BTOEE | DFN8EE | CRC16EE | CRC5EE | PIDEE | 0--0 0000 |
| F65h | UIR | — | SOFIF | STALLIF | IDLEIF | TRNIF | ACTVIF | UERRIF | URSTIF | -000 0000 |
| F64h | UIE | — | SOFIE | STALLIE | IDLEIE | TRNIE | ACTVIE | UERRIE | URSTIE | -000 0000 |
| F63h | UADDR | — | | | | ADDR<6:0> | | | | -000 0000 |
| F62h | UCFG | UTEYE | UOEMON | — | UPUEN | UTRDIS | FSEN | PPB<1:0> | | 00-0 0000 |
| F61h | USTAT | — | | | ENDP<3:0> | | DIR | PPBI | — | -xxx xxxx- |
| F60h | UCON | — | PPBRST | SE0 | PKTDIS | USBEN | RESUME | SUSPND | — | -0x0 000- |
| F5Fh | ANSELE | — | — | — | — | — | ANSE2 | ANSE1 | ANSE0 | ---- -111 |
| F5Eh | ANSEL0 | ANS07 | ANS06 | ANS05 | ANS04 | ANS03 | ANS02 | ANS01 | ANS00 | 1111 1111 |
| F5Dh | ANSEL0C | ANS07 | ANS06 | ANS05 | ANS04 | ANS03 | ANS02 | ANS01 | ANS00 | 11-- -1-- |
| F5Ch | ANSEL0B | — | — | ANS05 | ANS04 | ANS03 | ANS02 | ANS01 | ANS00 | --11 1111 |
| F5Bh | ANSEL0A | — | — | ANS05 | — | ANS03 | ANS02 | ANS01 | ANS00 | --1- 1111 |
| F5Ah | VREGCON | — | — | — | — | — | — | VREGPM<1:0> | | ---- --01 |
| F59h | CCPTMRS | — | — | — | — | C2TSEL | — | — | C1TSEL | ---- 0--0 |
| F58h | SRCON0 | SRLEN | | SRCLK<2:0> | | SRQEN | SRNQEN | SRPS | SRPR | 0000 0000 |
| F57h | SRCON1 | SRSPE | SRSCKE | SRSC2E | SRSC1E | SRRPE | SRRCKE | SRRC2E | SRRC1E | 0000 0000 |
| F56h | — | — | — | — | — | — | — | — | — | ----- |
| F55h | — | — | — | — | — | — | — | — | — | ----- |
| F54h | — | — | — | — | — | — | — | — | — | ----- |
| F53h | — | — | — | — | — | — | — | — | — | ----- |

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

Note 1: PIC18(L)F45K50 devices only.

2: PIC18(L)F2XK50 devices only.

6.4.6 STATUS REGISTER

The STATUS register, shown in [Register 6-2](#), 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 uluu').

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 [Section 27.2 "Extended Instruction Set"](#) and [Table 27-3](#).

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

6.5 Register Definitions: Status

REGISTER 6-2: 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/Digit 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/Borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) ⁽²⁾ 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 Digit 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 bit 4 or bit 3 of the source register.

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

PIC18(L)F2X/45K50

6.6 Data Addressing Modes

Note: The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See [Section 6.7 “Data Memory and the Extended Instruction Set”](#) for more information.

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

The addressing modes are:

- Inherent
 - Literal
 - Direct
 - Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in [Section 6.7.1 “Indexed Addressing with Literal Offset”](#).

6.6.1 INHERENT AND LITERAL ADDRESSING

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

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

6.6.2 DIRECT ADDRESSING

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

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of direct addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 6.4.4 “General Purpose Register File”**) or a location in the Access Bank (**Section 6.4.3 “Access Bank”**) as the data source for the instruction.

The Access RAM bit 'a' determines how the address is interpreted. When 'a' is '1', the contents of the BSR (**Section 6.4.2 “Bank Select Register (BSR)”**) are used with the address to determine the complete 12-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

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

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

6.6.3 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 6-5](#).

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

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

6.6.3.1 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.

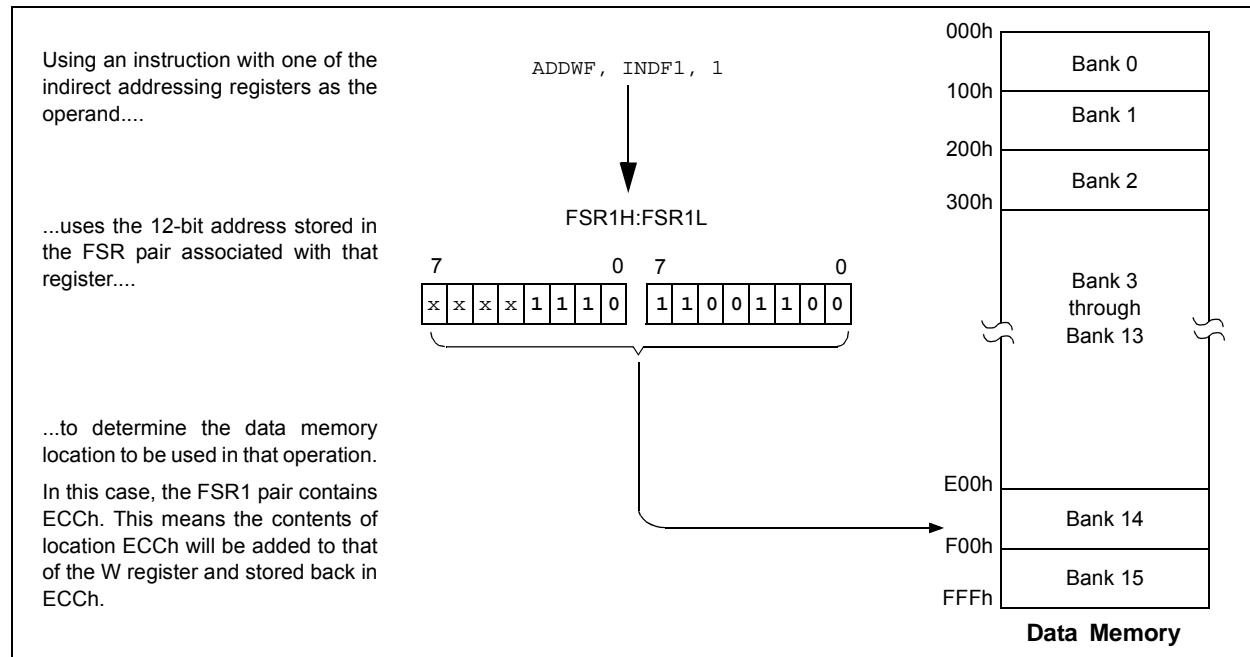
6.6.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

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 -128 to +127) 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 6-7: INDIRECT ADDRESSING



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

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

6.6.3.3 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.

6.7 Data Memory and the Extended Instruction Set

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

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

6.7.1 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.

6.7.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

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

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is '1'), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in [Figure 6-8](#).

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 27.2.1 "Extended Instruction Syntax"](#).

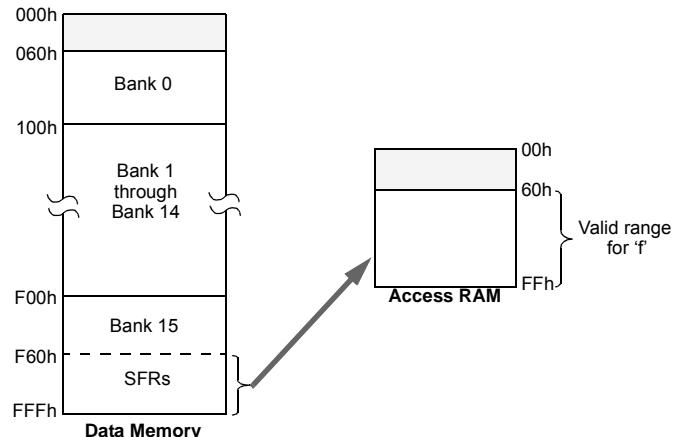
FIGURE 6-8: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

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

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

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

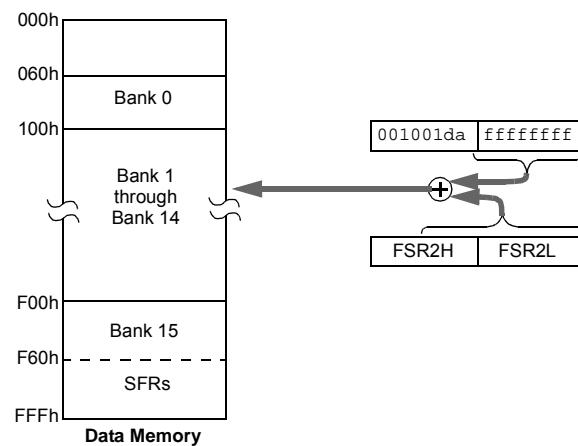
Locations below 60h are not available in this addressing mode.



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

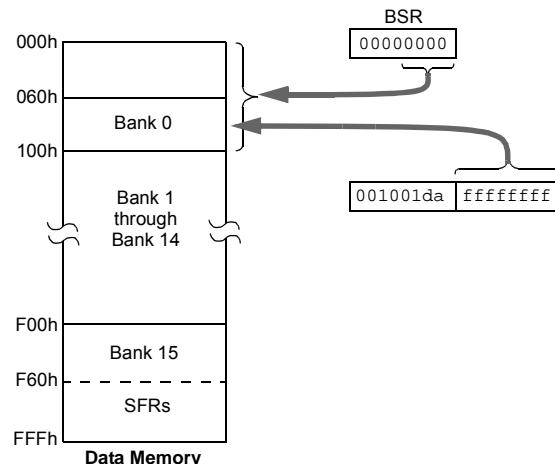
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

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



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

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

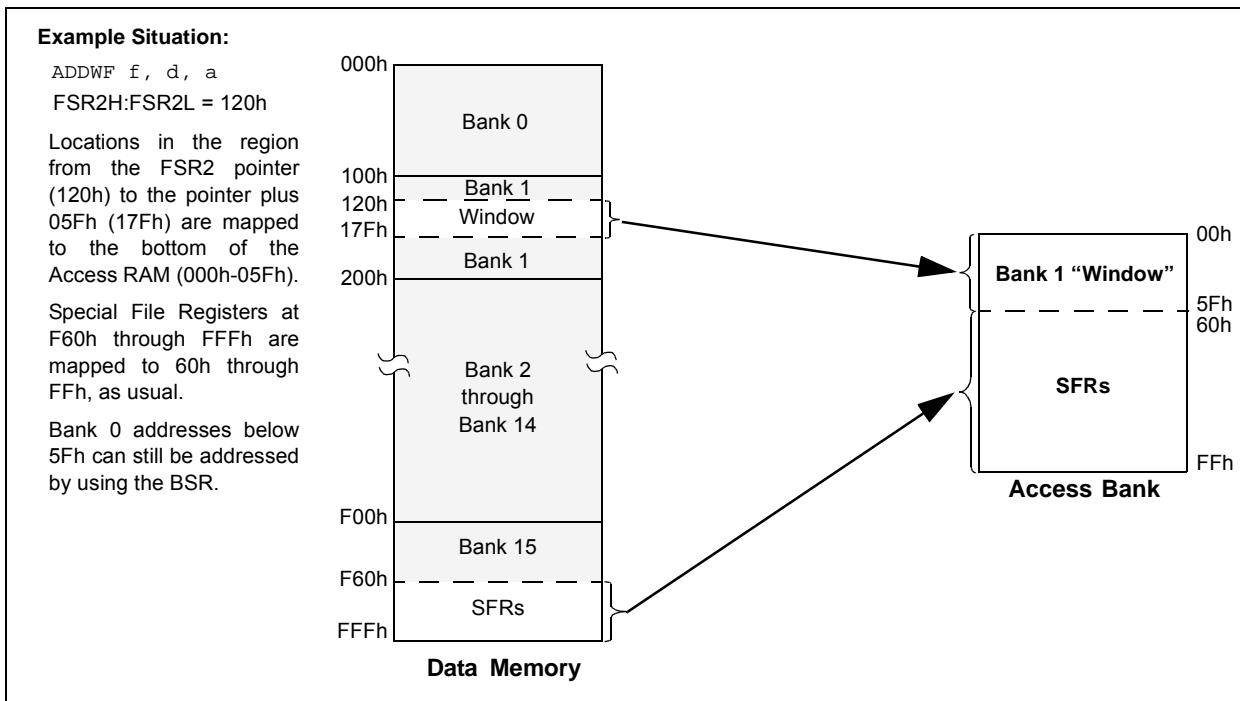


6.7.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

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 6.4.3 “Access Bank”](#)). An example of Access Bank remapping in this addressing mode is shown in [Figure 6-9](#).

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.

FIGURE 6-9: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING



6.8 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 27.2 “Extended Instruction Set”](#).

7.0 FLASH PROGRAM MEMORY

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

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

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

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

7.1 Table Reads and Table Writes

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

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

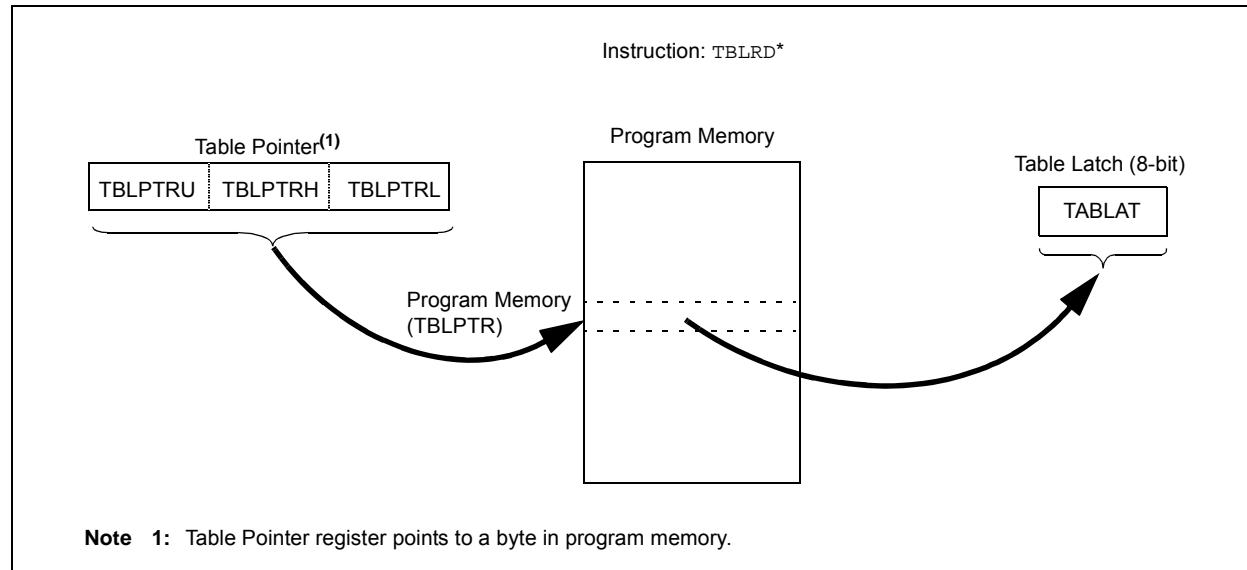
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

The table read operation retrieves one byte of data directly from program memory and places it into the TABLAT register. [Figure 7-1](#) 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 7.6 “Writing to Flash Program Memory”](#). [Figure 7-2](#) shows the operation of a table write with program memory and data RAM.

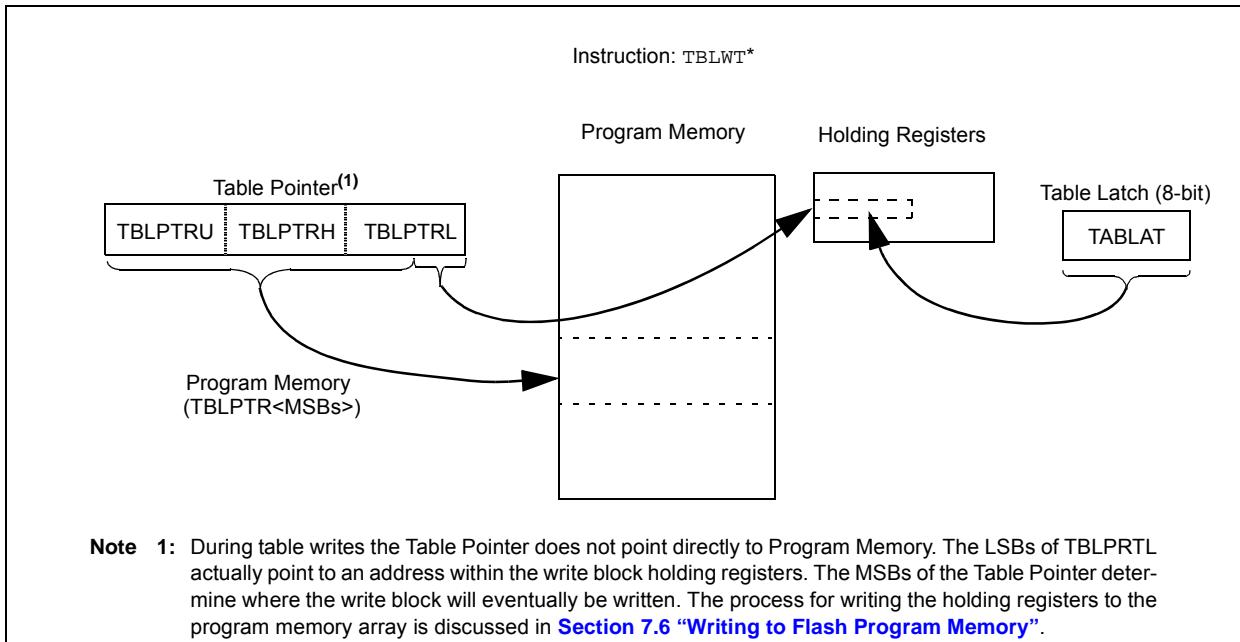
Table operations work with byte entities. Tables containing data, rather than program instructions, are not required to be word aligned. Therefore, a table can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need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

FIGURE 7-1: TABLE READ OPERATION



PIC18(L)F2X/45K50

FIGURE 7-2: TABLE WRITE OPERATION



7.2 Control Registers

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

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

7.2.1 EECON1 AND EECON2 REGISTERS

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

The EEPGD control bit determines if the access will be a program or data EEPROM memory access. When 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 26.0 "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. The 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.

7.3 Register Definitions: Memory Control

REGISTER 7-1: 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.

PIC18(L)F2X/45K50

7.3.1 TABLAT – TABLE LATCH REGISTER

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

7.3.2 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 on the TBLPTR affect only the low-order 21 bits.

7.3.3 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 7-1](#)). The six LSbs of the TBLPTR 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 (WR = 1), the entire holding register block is written to the Flash memory at the address determined by the MSbs of the TBLPTR. The six LSbs are ignored during Flash memory writes. For more detail, see [Section 7.6 “Writing to Flash Program Memory”](#).

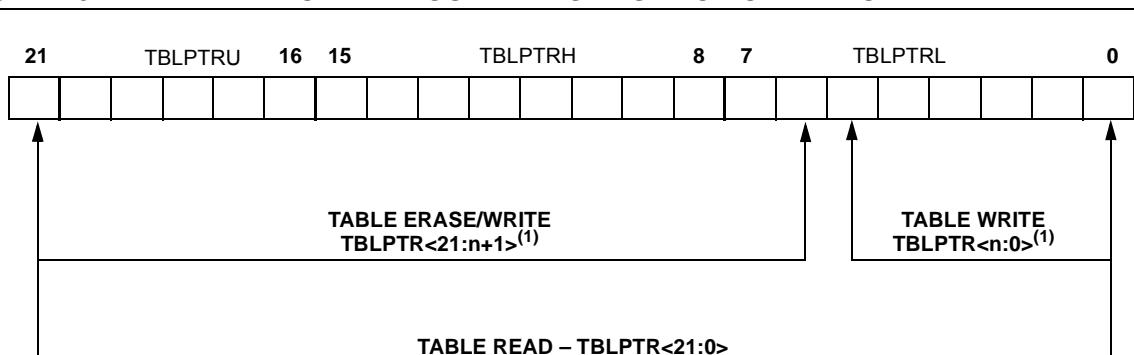
When an erase of program memory is executed, the 16 MSbs of the Table Pointer register (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

[Figure 7-3](#) describes the relevant boundaries of TBLPTR based on Flash program memory operations.

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

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

FIGURE 7-3: TABLE POINTER BOUNDARIES BASED ON OPERATION



Note 1: n = 5 for block sizes of 64 bytes.

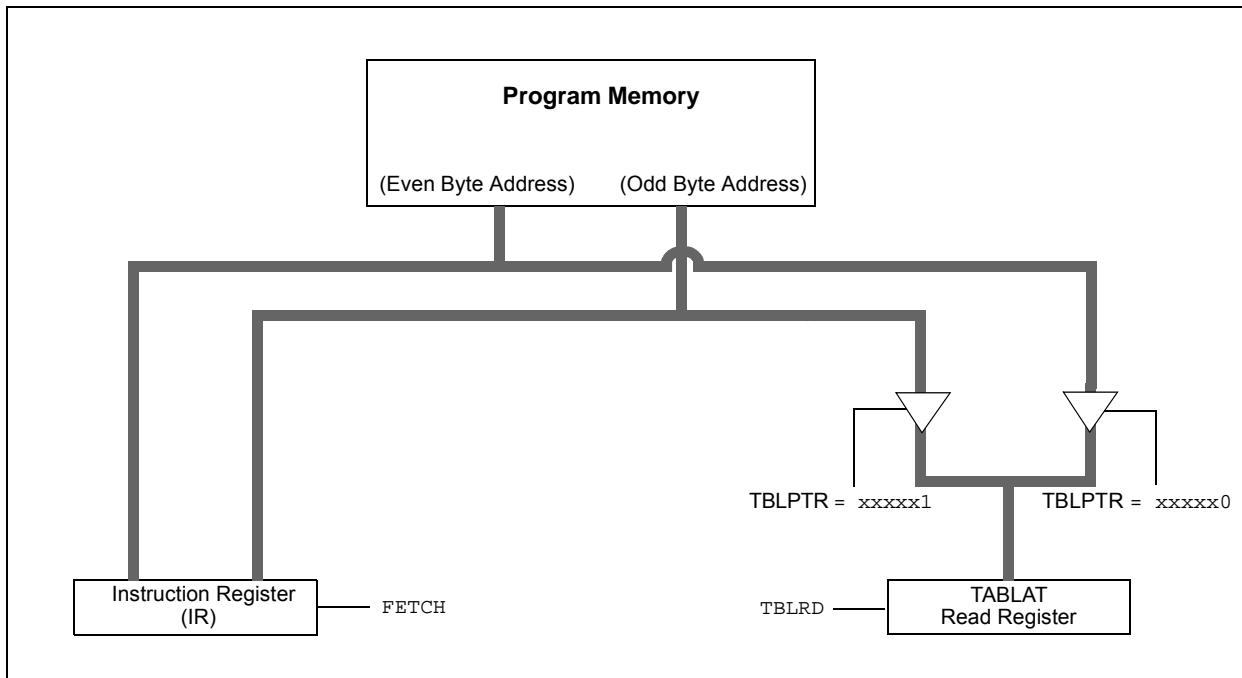
7.4 Reading the Flash Program Memory

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 7-4](#) shows the interface between the internal program memory and the TABLAT.

FIGURE 7-4: READS FROM FLASH PROGRAM MEMORY



EXAMPLE 7-1: READING A FLASH PROGRAM MEMORY WORD

```

MOVLW   CODE_ADDR_UPPER      ; Load TBLPTR with the base
MOVWF   TBLPTRU             ; 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

```

7.5 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 7.5.1 "Flash Program Memory Erase Sequence"](#), is used to guard against accidental writes. This is sometimes referred to as a long write.

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

EXAMPLE 7-2: ERASING A FLASH PROGRAM MEMORY BLOCK

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

7.6 Writing to Flash Program Memory

The programming block size is 64 bytes. Word or byte programming is not supported.

Table writes are used internally to load the holding registers needed to program the Flash memory. There are only as many holding registers as there are bytes in a write block (64 bytes).

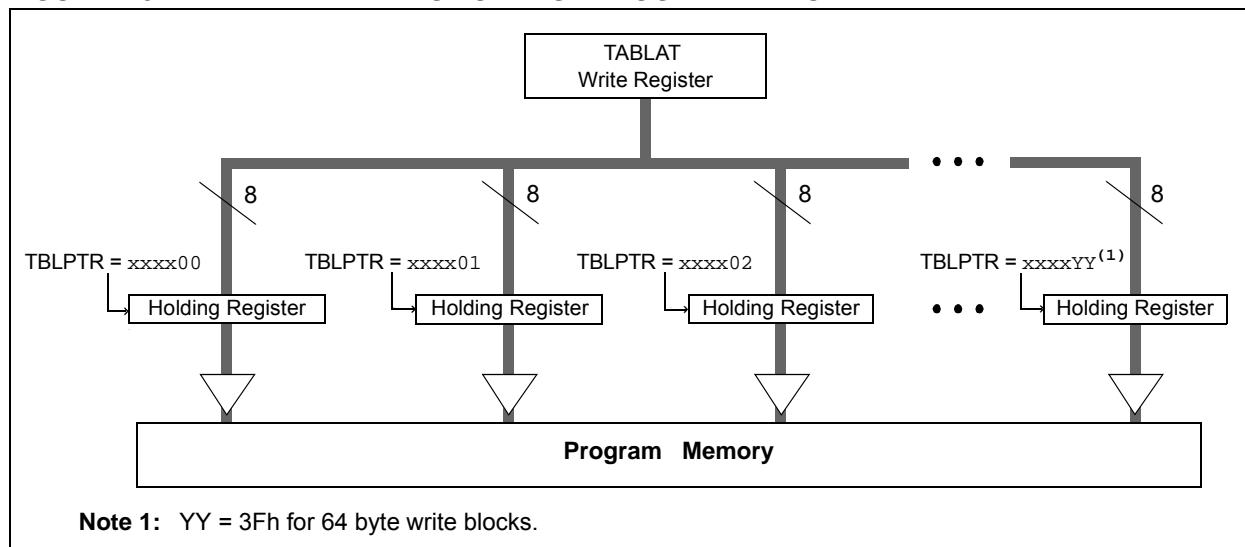
Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction needs to be executed 64 times for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. After all the holding registers have been written, 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 7-5: TABLE WRITES TO FLASH PROGRAM MEMORY



7.6.1 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 64-byte block into the holding registers with auto-increment (TBLWT*+ or TBLWT+*).
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 writes.

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

This procedure will require about 6 ms to update each write block of memory. An example of the required code is given in [Example 7-3](#).

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

PIC18(L)F2X/45K50

EXAMPLE 7-3: WRITING TO FLASH PROGRAM MEMORY

```
        MOVLW    D'64'                      ; number of bytes in erase block
        MOVWF    COUNTER
        MOVLW    BUFFER_ADDR_HIGH          ; point to buffer
        MOVWF    FSROH
        MOVLW    BUFFER_ADDR_LOW
        MOVWF    FSR0L
        MOVLW    CODE_ADDR_UPPER          ; Load TBLPTR with the base
        MOVWF    TBLPTRU                 ; address of the memory block
        MOVLW    CODE_ADDR_HIGH
        MOVWF    TBLPTRH
        MOVLW    CODE_ADDR_LOW
        MOVWF    TBLPTRL

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

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

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

Required Sequence
        MOVLW    55h
        MOVWF    EECON2                  ; write 55h
        MOVLW    0AAh
        MOVWF    EECON2
        BSF      EECON1, WR              ; start erase (CPU stall)
        BSF      INTCON, GIE
        TBLRD*-
        MOVLW    BUFFER_ADDR_HIGH          ; point to buffer
        MOVWF    FSROH
        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.
```

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

| | | | | | |
|-------------------|--------|---------------------|---|--|--|
| PROGRAM_MEMORY | DECFSZ | COUNTER | ; loop until holding registers are full | | |
| | BRA | WRITE_WORD_TO_HREGS | | | |
| | BSF | EECON1, EEPGD | | | |
| | BCF | EECON1, CFGS | | | |
| | BSF | EECON1, WREN | | | |
| | BCF | INTCON, GIE | | | |
| | MOVLW | 55h | | | |
| | MOVWF | EECON2 | | | |
| | MOVLW | 0AAh | | | |
| | MOVWF | EECON2 | | | |
| Required Sequence | BSF | EECON1, WR | | | |
| | DCFSZ | COUNTER2 | | | |
| | BRA | WRITE_BYTE_TO_HREGS | | | |
| | BSF | INTCON, GIE | | | |
| | BCF | EECON1, WREN | | | |
| | | | | | |
| | | | | | |

7.6.2 WRITE VERIFY

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

7.6.3 UNEXPECTED TERMINATION OF WRITE OPERATION

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

TABLE 7-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page | | | | | | |
|---------|---|-----------|---|--------|-------|--------|--------|--------|------------------|--|--|--|--|--|--|
| TBLPTRU | — | — | Program Memory Table Pointer Upper Byte (TBLPTR<21:16>) | | | | | | — | | | | | | |
| TBPLTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | — | | | | | | |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | — | | | | | | |
| TABLAT | Program Memory Table Latch | | | | | | | | — | | | | | | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 | | | | | | |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | — | | | | | | |
| EECON1 | EEPGD | CFGs | — | FREE | WRERR | WREN | WR | RD | 97 | | | | | | |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 | | | | | | |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 | | | | | | |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 | | | | | | |

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

7.6.4 PROTECTION AGAINST SPURIOUS WRITES

To protect against spurious writes to Flash program memory, the write initiate sequence must also be followed. See [Section 26.0 "Special Features of the CPU"](#) for more detail.

7.7 Flash Program Operation During Code Protection

See [Section 26.5 "Program Verification and Code Protection"](#) for details on code protection of Flash program memory.

8.0 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 specified V_{DD} 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 the Data EEPROM Memory parameters in [Section 29.0 “Electrical Specifications”](#) for limits.

8.1 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).

8.2 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 8-1](#)) 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 7.1 “Table Reads and Table Writes”](#) regarding table reads.

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

REGISTER 8-1: 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.

PIC18(L)F2X/45K50

8.3 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 8-1](#).

8.4 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 8-2](#) 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, EIF, is set. The user may either enable this interrupt or poll this bit. EIF must be cleared by software.

8.5 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 8-1: 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 8-2: DATA EEPROM WRITE

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

8.6 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 26.0 “Special Features of the CPU”](#) for additional information.

8.7 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).

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

8.8 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. Refer to the Data EEPROM Memory parameters in [Section 29.0 “Electrical Specifications”](#) for write cycle limits. 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 8-3](#).

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 8-3: DATA EEPROM REFRESH ROUTINE

```
CLRF    EEADR          ; Start at address 0
BCF     EECON1, CFGS   ; Set for memory
BCF     EECON1, EEPGD   ; Set for Data EEPROM
BCF     INTCON, GIE    ; Disable interrupts
BSF     EECON1, WREN   ; Enable writes
Loop
  BSF    EECON1, RD    ; Loop to refresh array
  MOVLW  55h           ; Read current address
  MOVWF  EECON2         ; 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
```

PIC18(L)F2X/45K50

TABLE 8-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|--------|---|-----------|--------|--------|--------|--------|--------|--------|------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| EEADR | EEADR7 | EEADR6 | EEADR5 | EEADR4 | EEADR3 | EEADR2 | EEADR1 | EEADR0 | — |
| EEDATA | EEPROM Data Register | | | | | | | | — |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | — |
| EECON1 | EEPGD | CFGSD | — | FREE | WRERR | WREN | WR | RD | 105 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 |

Legend: — = unimplemented, read as '0'. Shaded bits are not used during EEPROM access.

9.0 8 x 8 HARDWARE MULTIPLIER

9.1 Introduction

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

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in [Table 9-1](#).

9.2 Operation

[Example 9-1](#) shows the instruction sequence for an 8 x 8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

[Example 9-2](#) shows the sequence to do an 8 x 8 signed multiplication. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

TABLE 9-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

| Routine | Multiply Method | Program Memory (Words) | Cycles (Max) | Time | | | |
|------------------|---------------------------|------------------------|--------------|--------------|--------------|---------------|-------------|
| | | | | @ 48 MHz | @ 40 MHz | @ 10 MHz | @ 4 MHz |
| 8 x 8 unsigned | Without hardware multiply | 13 | 69 | 5.7 μ s | 6.9 μ s | 27.6 μ s | 69 μ s |
| | Hardware multiply | 1 | 1 | 83.3 ns | 100 ns | 400 ns | 1 μ s |
| 8 x 8 signed | Without hardware multiply | 33 | 91 | 7.5 μ s | 9.1 μ s | 36.4 μ s | 91 μ s |
| | Hardware multiply | 6 | 6 | 500 ns | 600 ns | 2.4 μ s | 6 μ s |
| 16 x 16 unsigned | Without hardware multiply | 21 | 242 | 20.1 μ s | 24.2 μ s | 96.8 μ s | 242 μ s |
| | Hardware multiply | 28 | 28 | 2.3 μ s | 2.8 μ s | 11.2 μ s | 28 μ s |
| 16 x 16 signed | Without hardware multiply | 52 | 254 | 21.6 μ s | 25.4 μ s | 102.6 μ s | 254 μ s |
| | Hardware multiply | 35 | 40 | 3.3 μ s | 4.0 μ s | 16.0 μ s | 40 μ s |

EXAMPLE 9-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

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

EXAMPLE 9-2: 8 x 8 SIGNED MULTIPLY ROUTINE

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

PIC18(L)F2X/45K50

[Example 9-3](#) shows the sequence to do a 16×16 unsigned multiplication. [Equation 9-1](#) shows the algorithm that is used. The 32-bit result is stored in four registers (RES<3:0>).

EQUATION 9-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

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

EXAMPLE 9-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```

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

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

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

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

```

[Example 9-4](#) shows the sequence to do a 16×16 signed multiply. [Equation 9-2](#) 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 9-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

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

EXAMPLE 9-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```

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

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

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

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

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

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

CONT_CODE
:

```

10.0 INTERRUPTS

The PIC18(L)F2X/45K50 devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high or low priority level (INT0 does not have a priority bit, it is always a high priority). 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 13 registers used to control interrupt operation.

These registers are:

- INTCON, INTCON2, INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3
- RCON

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

10.1 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/GIEL bit of the INTCON register is the global interrupt enable for the peripherals. The PEIE/GIEL bit disables only the peripheral interrupt sources and enables the peripheral interrupt sources when the GIE/GIEH bit is also set. The GIE/GIEH 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.

10.2 Interrupt Priority

The interrupt priority feature is enabled by setting the IPEN bit of the RCON register. When interrupt priority is enabled the GIE/GIEH and PEIE/GIEL 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 GIEH 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 (GIE) 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.

10.3 Interrupt Response

When an interrupt is responded to, the Global Interrupt Enable bit is cleared to disable further interrupts. The GIE/GIEH 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.

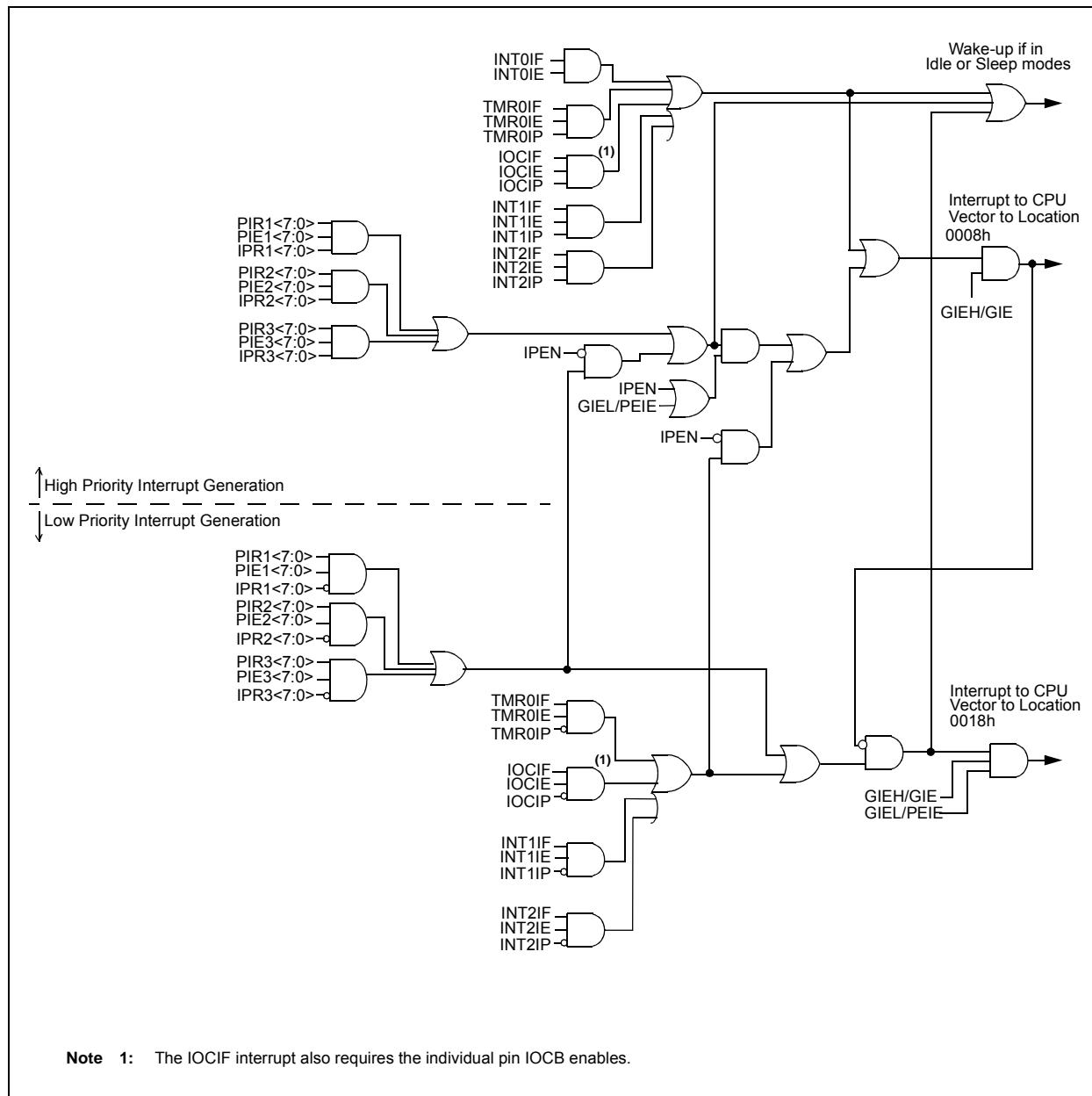
The “return from interrupt” instruction, RETFIE, exits the interrupt routine and sets the GIE/GIEH bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

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.

PIC18(L)F2X/45K50

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 10-1: PIC18 INTERRUPT LOGIC



10.4 INTCON Registers

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

10.5 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Request Flag registers (PIR1, PIR2 and PIR3).

10.6 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Enable registers (PIE1, PIE2 and PIE3). When IPEN = 0, the PEIE/GIEL bit must be set to enable any of these peripheral interrupts.

10.7 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Priority registers (IPR1, IPR2 and IPR3). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

PIC18(L)F2X/45K50

10.8 Register Definitions: Interrupt Control

REGISTER 10-1: INTCON: INTERRUPT CONTROL REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-x |
|----------|-----------|--------|--------|-------|--------|--------|-------|
| GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF |
| 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 bitWhen 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 bitWhen 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 **IOCIE:** Interrupt-On-Change (IOC_x) Interrupt Enable bit⁽²⁾

- 1 = Enables the IOC_x port change interrupt
- 0 = Disables the IOC_x 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 **IOCIF:** Interrupt-On-Change (IOC_x) Interrupt Flag bit⁽¹⁾

- 1 = At least one of the IOC pins changed state (must be cleared by software)
- 0 = None of the IOC pins have changed state

Note 1: A mismatch condition will continue to set the IOCIF bit. Reading PORTB/PORTC will end the mismatch condition and allow the bit to be cleared.

2: Port change interrupts also require the individual pins IOCB_x/IOCC_x enables.

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 10-2: 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 |
|-------|---------|---------|---------|-----|--------|-----|-------|
| RBPU | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | IOCIP |
| bit 7 | | | | — | | — | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

| | |
|-------|---|
| bit 7 | RBPU: PORTB Pull-up Enable bit 1 = All PORTB pull-ups are disabled 0 = PORTB pull-ups are enabled provided that the pin is an input and the corresponding WPUB bit is 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 | IOCIP: Port Change Interrupt Priority bit 1 = High priority 0 = Low priority |

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

PIC18(L)F2X/45K50

REGISTER 10-3: 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
-n = Value at POR

W = Writable bit
'1' = Bit is set

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

- bit 7 **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 should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

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

| R/W-0 | R/W-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|------|------|-------|--------|--------|--------|
| ACTIF | 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 | ACTIF: Active Clock Tuning Interrupt Flag bit 1 = An Active Clock Tuning Event generated an interrupt (must be cleared in software) 0 = No Active Clock Tuning interrupt is pending |
| 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, RCREG1, is full (cleared when RCREG1 is read) 0 = The EUSART receive buffer is empty |
| bit 4 | TXIF: EUSART Transmit Interrupt Flag bit 1 = The EUSART transmit buffer, TXREG1, is empty (cleared when TXREG1 is written) 0 = The EUSART transmit buffer is full |
| bit 3 | SSPIF: Master Synchronous Serial Port Interrupt Flag bit 1 = The transmission/reception is complete (must be cleared by software) 0 = Waiting to transmit/receive |
| bit 2 | CCP1IF: CCP1 Interrupt Flag bit <u>Capture mode:</u> 1 = A TMR1 register capture occurred (must be cleared by software) 0 = No TMR1 register capture occurred <u>Compare mode:</u> 1 = A TMR1 register compare match occurred (must be cleared by software) 0 = No TMR1 register compare match occurred <u>PWM mode:</u> 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:** 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/GIEH of the INTCON register.
- 2:** User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

PIC18(L)F2X/45K50

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

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|--------|-------|-------|-------|-------|--------|--------|--------|
| OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit
-n = Value at POR

W = Writable bit
'1' = Bit is set

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

- bit 7 **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:** MSSP Bus Collision Interrupt Flag bit
1 = A bus collision occurred (must be cleared by software)
0 = No bus collision occurred
- bit 2 **HLVDIF:** Low-Voltage Detect Interrupt Flag bit
1 = A low-voltage condition occurred (direction determined by the VDIRMAG bit of the HLVDCON register)
0 = A low-voltage condition has not occurred
- 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 **CCP2IF:** CCP2 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.

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

| U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-----|-----|--------|-------|---------|---------|
| — | — | — | — | CTMUIF | USBIF | TMR3GIF | TMR1GIF |
| 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 **CTMUIF:** CTMU Interrupt Flag bit
1 = CTMU interrupt occurred (must be cleared in software)
0 = No CTMU interrupt occurred
- bit 2 **USBIF:** USB Interrupt Flag bit
1 = USB requested an interrupt (must be cleared in software)
0 = No USB interrupt request
- bit 1 **TMR3GIF:** TMR3 Gate Interrupt Flag bit
1 = TMR gate interrupt occurred (must be cleared in software)
0 = No TMR gate occurred
- bit 0 **TMR1GIF:** TMR1 Gate Interrupt Flag bit
1 = TMR gate interrupt occurred (must be cleared in software)
0 = No TMR gate occurred

PIC18(L)F2X/45K50

REGISTER 10-7: PIE1: PERIPHERAL INTERRUPT ENABLE (FLAG) 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 |
|-------|-------|-------|-------|-------|--------|--------|--------|
| ACTIE | 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 **ACTIE:** Active Clock Tuning Interrupt Enable bit

1 = Enables Active Clock Tuning interrupt
0 = Disables Active Clock Tuning interrupt

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

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

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

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

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

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

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

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

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

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

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

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

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

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

REGISTER 10-8: PIE2: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 2

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|--------|-------|-------|-------|-------|--------|--------|--------|
| OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- | | |
|-------|--|
| bit 7 | OSCFIE: Oscillator Fail Interrupt Enable bit 1 = Enabled 0 = Disabled |
| bit 6 | 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: MSSP Bus Collision Interrupt Enable bit 1 = Enabled 0 = Disabled |
| bit 2 | HLVDIE: Low-Voltage Detect Interrupt Enable bit 1 = Enabled 0 = Disabled |
| bit 1 | TMR3IE: TMR3 Overflow Interrupt Enable bit 1 = Enabled 0 = Disabled |
| bit 0 | CCP2IE: CCP2 Interrupt Enable bit 1 = Enabled 0 = Disabled |

PIC18(L)F2X/45K50

REGISTER 10-9: PIE3: PERIPHERAL INTERRUPT ENABLE (FLAG) REGISTER 3

| U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-----|-----|--------|-------|---------|---------|
| — | — | — | — | CTMUIE | USBIE | TMR3GIE | TMR1GIE |
| 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 **CTMUIE:** CTMU Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 2 **USBIE:** USB Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 1 **TMR3GIE:** TMR3 Gate Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 0 **TMR1GIE:** TMR1 Gate Interrupt Enable bit

1 = Enabled

0 = Disabled

REGISTER 10-10: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|-------|-------|-------|-------|-------|--------|--------|--------|
| ACTIP | 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 | ACTIP: Active Clock Tuning Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 6 | ADIP: A/D Converter Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 5 | 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 |

PIC18(L)F2X/45K50

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

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|--------|-------|-------|-------|-------|--------|--------|--------|
| OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit
-n = Value at POR

W = Writable bit
'1' = Bit is set

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

- bit 7 **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:** MSSP Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 2 **HLVDIP:** Low-Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 1 **TMR3IP:** TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 0 **CCP2IP:** CCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

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

| U-0 | U-0 | U-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|-------|-------|-----|-----|--------|-------|---------|---------|
| — | — | — | — | CTMUIP | USBIP | TMR3GIP | TMR1GIP |
| 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 **CTMUIP:** CTMU Interrupt Priority bit

1 = High priority

0 = Low priority

bit 2 **USBIP:** USB Interrupt Priority bit

1 = High priority

0 = Low priority

bit 1 **TMR3GIP:** TMR3 Gate Interrupt Priority bit

1 = High priority

0 = Low priority

bit 0 **TMR1GIP:** TMR1 Gate Interrupt Priority bit

1 = High priority

0 = Low priority

10.9 INTn Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1 and RB2/INT2 pins are edge-triggered. If the corresponding INTEDG_x bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. When a valid edge appears on the RB_x/INT_x 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/GIEH, 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.

10.10 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 12.0 “Timer0 Module”](#) for further details on the Timer0 module.

10.11 PORTB/PORTC Interrupt-on-Change

An input change on PORTB<7:4> or PORTC<2:0> sets flag bit, IOCIF of the INTCON register. The interrupt can be enabled/disabled by setting/clearing enable bit, IOCIE of the INTCON register. Pins must also be individually enabled with the IOCB/IOCC register. Interrupt priority for interrupt-on-change is determined by the value contained in the interrupt priority bit, IOCIP of the INTCON2 register.

10.12 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 6.2.2 “Fast Register Stack”](#)), the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user's application, other registers may also need to be saved. [Example 10-1](#) saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

EXAMPLE 10-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

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

TABLE 10-1: REGISTERS ASSOCIATED WITH INTERRUPTS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|---------|----------|-----------|---------|---------|--------|--------|---------|---------|------------------|
| ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 148 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| INTCON2 | RBU | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | IOCIP | 115 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 116 |
| IOCB | IOCB7 | IOCB6 | IOCB5 | IOCB4 | — | — | — | — | 151 |
| IOCC | IOCC7 | IOCC6 | IOCC5 | IOCC4 | — | IOCC2 | IOCC1 | IOCC0 | 151 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 |
| IPR3 | — | — | — | — | CTMUIP | USBIP | TMR3GIP | TMR1GIP | 125 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 |
| PIE3 | — | — | — | — | CTMUIE | USBI | TMR3GIE | TMR1GIE | 122 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 |
| PIR3 | — | — | — | — | CTMUIF | USBF | TMR3GIF | TMR1GIF | 119 |
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | 146 |
| RCON | IPEN | SBOREN | — | RI | TO | PD | POR | BOR | 64 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for interrupts.

TABLE 10-2: CONFIGURATION REGISTERS ASSOCIATED WITH INTERRUPTS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|-------|-------|-------|--------|--------|------------------|
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |
| CONFIG4L | DEBUG | XINST | ICPRT | — | — | LVP | — | STRVEN | 377 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for interrupts.

PIC18(L)F2X/45K50

11.0 I/O PORTS

Depending on the device selected and features enabled, there are up to five ports available. All pins of the I/O ports are multiplexed with one or more alternate functions 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.

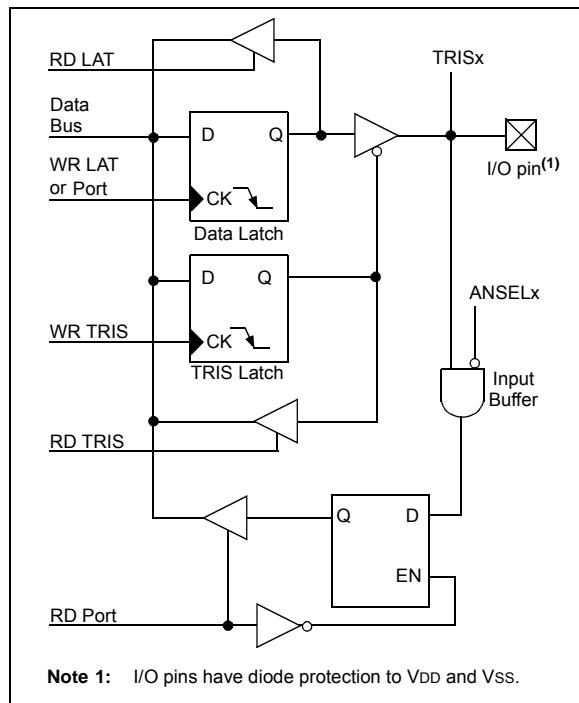
Each port has five registers for its operation. These registers are:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)
- ANSEL register (analog input control)
- SLRCON register (port slew rate control)

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

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

FIGURE 11-1: GENERIC I/O PORT OPERATION



11.1 PORTA Registers

PORTA is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., 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 Data Latch (LATA) register is also memory mapped. Read-modify-write operations on the LATA register read and write the latched output value for PORTA.

The RA4 pin is multiplexed with the Timer0 module clock input and one of the comparator outputs to become the RA4/T0CKI/C1OUT pin. Pins RA6 and RA7 are multiplexed with the main oscillator pins; they are enabled as oscillator or I/O pins by the selection of the main oscillator in the Configuration register (see [Section 26.1 “Configuration Bits”](#) for details). When they are not used as port pins, RA6 and RA7 and their associated TRIS and LAT bits are read as ‘0’.

The other PORTA pins are multiplexed with analog inputs, the analog VREF+ and VREF- inputs, and the comparator voltage reference output. The operation of pins RA<3:0> and RA5 as analog is selected by setting the ANSELA<5, 3:0> bits in the ANSELA register which is the default setting after a Power-on Reset.

Pins RA0 through RA5 may also be used as comparator inputs or outputs by setting the appropriate bits in the CM1CON0 and CM2CON0 registers.

Note: On a Power-on Reset, RA5 and RA<3:0> are configured as analog inputs and read as ‘0’. RA4 is configured as a digital input.

The RA4/T0CKI/C1OUT pin is a Schmitt Trigger input. All other PORTA pins have TTL input levels and full CMOS output drivers.

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

EXAMPLE 11-1: INITIALIZING PORTA

```
MOVLB 0xF      ; Set BSR for banked SFRs
CLRF  LATA     ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF  ANSELA   ; Configure I/O
                ; for digital inputs
MOVLW 0CFh     ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISA    ; Set RA<3:0> as inputs
                ; RA<5:4> as outputs
```

TABLE 11-1: PORTA I/O SUMMARY

| Pin Name | Function | TRIS Setting | ANSEL Setting | Pin Type | Buffer Type | Description |
|---------------------------------|---------------------|--------------|---------------|----------|-------------|---|
| RA0/C12IN0-/AN0 | RA0 | 0 | x | O | DIG | LATA<0> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTA<0> data input; disabled when analog input enabled. |
| | C12IN0- | 1 | 1 | I | AN | Comparators C1 and C2 inverting input. |
| | AN0 | 1 | 1 | I | AN | Analog input 0. |
| RA1/C12IN1-/AN1 | RA1 | 0 | x | O | DIG | LATA<1> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTA<1> data input; disabled when analog input enabled. |
| | C12IN1- | 1 | 1 | I | AN | Comparators C1 and C2 inverting input. |
| | AN1 | 1 | 1 | I | AN | Analog input 1. |
| RA2/C2IN+/AN2/DACOUT/VREF- | RA2 | 0 | x | O | DIG | LATA<2> data output; not affected by analog input; disabled when DACOUT enabled. |
| | | 1 | 0 | I | TTL | PORTA<2> data input; disabled when analog input enabled; disabled when DACOUT enabled. |
| | C2IN+ | 1 | 1 | I | AN | Comparator C2 non-inverting input. |
| | AN2 | 1 | 1 | I | AN | Analog output 2. |
| | DACOUT | x | 1 | O | AN | DAC Reference output. |
| | VREF- | 1 | 1 | I | AN | A/D reference voltage (low) input. |
| | RA3/C1IN+/AN3/VREF+ | 0 | x | O | DIG | LATA<3> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTA<3> data input; disabled when analog input enabled. |
| | C1IN+ | 1 | 1 | I | AN | Comparator C1 non-inverting input. |
| | AN3 | 1 | 1 | I | AN | Analog input 3. |
| RA4/C1OUT/SRQ/T0CKI | RA4 | 0 | — | O | DIG | LATA<4> data output. |
| | | 1 | — | I | ST | PORTA<4> data input; default configuration on POR. |
| | C1OUT | 0 | — | O | DIG | Comparator C1 output. |
| | SRQ | 0 | — | O | DIG | SR latch Q output; take priority over CCP 5 output. |
| | T0CKI | 1 | — | I | ST | Timer0 external clock input. |
| RA5/C2OUT/SRNQ/SS1/HLDV DIN/AN4 | RA5 | 0 | x | O | DIG | LATA<5> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTA<5> data input; disabled when analog input enabled. |
| | C2OUT | 0 | 0 | O | DIG | Comparator C2 output. |
| | SRNQ | 0 | 0 | O | DIG | SR latch \bar{Q} output. |
| | SS1 | 1 | 0 | I | TTL | SPI slave select input (MSSP). |
| | HLDV DIN | 1 | 1 | I | AN | High/Low-Voltage Detect input. |
| | AN4 | 1 | 1 | I | AN | A/D input 4. |
| RA6/CLKO/OSC2 | RA6 | 0 | — | O | DIG | LATA<6> data output; enabled in INTOSC modes when CLKO is not enabled. |
| | | 1 | — | I | TTL | PORTA<6> data input; enabled in INTOSC modes when CLKO is not enabled. |
| | CLKO | x | — | O | DIG | In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate. |
| | OSC2 | x | — | O | XTAL | Oscillator crystal output; connects to crystal or resonator in Crystal Oscillator mode. |

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I²C™ = Schmitt Trigger input with I²C.

PIC18(L)F2X/45K50

TABLE 11-1: PORTA I/O SUMMARY (CONTINUED)

| Pin Name | Function | TRIS Setting | ANSEL Setting | Pin Type | Buffer Type | Description | | |
|---------------|----------|--------------|---------------|----------|-------------|---|--|--|
| RA7/CLKI/OSC1 | RA7 | 0 | — | O | DIG | LATA<7> data output; disabled in external oscillator modes. | | |
| | | 1 | — | I | TTL | PORTA<7> data input; disabled in external oscillator modes. | | |
| | CLKI | x | — | I | AN | External clock source input; always associated with pin function OSC1. | | |
| | OSC1 | x | — | I | XTAL | Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise. | | |

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; $\text{I}^2\text{C}^{\text{TM}}$ = Schmitt Trigger input with I^2C .

TABLE 11-2: REGISTERS ASSOCIATED WITH PORTA

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|---------|------------|--------|-----------|-------------|-----------|-----------|--------|---------------------|
| ANSELA | — | — | ANSA5 | — | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 147 |
| CM1CON0 | C1ON | C1OUT | C1OE | C1POL | C1SP | C1R | C1CH<1:0> | | 307 |
| CM2CON0 | C2ON | C2OUT | C2OE | C2POL | C2SP | C2R | C2CH<1:0> | | 307 |
| VREFCON1 | DACEN | DACLPS | DACOE | — | DACPSS<1:0> | | — | DACNSS | 334 |
| VREFCON2 | — | — | — | Dacr<4:0> | | | | | 335 |
| HLVDCON | VDIRMAG | BGVST | IRVST | HLVDEN | HLVLDL<3:0> | | | | 364 |
| PORTA | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 146 |
| LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | 150 |
| SLRCON | — | — | — | SLRE | SLRD | SLRC | SLRB | SLRA | 152 |
| SRCON0 | SRLEN | SRCLK<2:0> | | | SRQEN | SRNQEN | SRPS | SRPR | 328 |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 252 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS<2:0> | | | 153 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 149 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTA.

TABLE 11-3: CONFIGURATION REGISTERS ASSOCIATED WITH PORTA

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|--------|-------|-----------|-------|-------|-------|---------------------|
| CONFIG1H | IESO | FCMEN | PCLKEN | — | FOSC<3:0> | | | | 373 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTA.

11.1.1 PORTA OUTPUT PRIORITY

Each PORTA pin is multiplexed with other functions. The pins, their combined functions and their output priorities are briefly described here. For additional information, refer to the appropriate section in this data sheet.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the higher priority. [Table 11-4](#) lists the PORTA pin functions from the highest to the lowest priority.

Analog input functions, such as ADC and comparator, are not shown in the priority lists.

These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown below.

PIC18(L)F2X/45K50

TABLE 11-4: PORT PIN FUNCTION PRIORITY

| Port bit | Port Function Priority by Port Pin | | | | |
|----------|------------------------------------|---------------------|---------------------|----------------------|----------------------|
| | PORTA | PORTB | PORTC | PORTD ⁽²⁾ | PORTE ⁽²⁾ |
| 0 | RA0 | SDA | SOSCO | | |
| | | RB0 | | | |
| | | | RC0 | RD0 | RE0 |
| 1 | RA1 | SCL | SOSCI | | |
| | | SCK | CCP2 ⁽³⁾ | | RE1 |
| | | P1C ⁽¹⁾ | | RD1 | |
| | | RB1 | RC1 | | |
| 2 | DACOUT | | CCP1 | | |
| | RA2 | P1B ⁽¹⁾ | P1A | RD2 | RE2 |
| | | RB2 | CTPLS | | |
| | | | RC2 | | |
| 3 | RA3 | SDO ⁽³⁾ | | | MCLR |
| | | CCP2 ⁽⁴⁾ | | RD3 | VPP |
| | | | | | RE3 |
| | | RB3 | | | |
| 4 | SRQ | P1D ⁽¹⁾ | D- | | |
| | C1OUT | RB4 | | | |
| | | | | RD4 | |
| | RA4 | | | | |
| 5 | SRNQ | | D+ | P1B | |
| | C2OUT | | | RD5 | |
| | RA5 | | | | |
| | | RB5 | | | |
| 6 | OSC2 | PGC | TX/CK | | |
| | CLKO | ICDCK | | P1C | |
| | RA6 | RB6 | | RD6 | |
| | | | RC6 | | |
| 7 | OSC1 | PGD | RX/DT | | |
| | RA7 | ICDDT | | P1D | |
| | | RB7 | RC7 | RD7 | |
| | | | | | |

Note 1: PIC18(L)F2XK50 devices.

2: PIC18(L)F45K50 devices.

3: Function default pin.

4: Function alternate pin.

11.2 PORTB Registers

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

The 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 11-2: INITIALIZING PORTB

```

MOVLB 0xF      ; Set BSR for banked SFRs
CLRF  LATB    ; Initialize PORTB by
               ; clearing output
               ; data latches
MOVLW 0F0h    ; Value for init
MOVWF ANSELB  ; Enable RB<3:0> for
               ; digital input pins
               ; (not required if config bit
               ; PBADEN is clear)
MOVLW 0CFh    ; Value used to
               ; initialize data
               ; direction
MOVWF TRISB   ; Set RB<3:0> as inputs
               ; RB<5:4> as outputs
               ; RB<7:6> as inputs

```

11.2.1 PORTB OUTPUT PRIORITY

Each PORTB pin is multiplexed with other functions. The pins, their combined functions and their output priorities are briefly described here. For additional information, refer to the appropriate section in this data sheet.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the higher priority. **Table 11-4** lists the PORTB pin functions from the highest to the lowest priority.

Analog input functions, such as ADC, comparator and SR latch inputs, are not shown in the priority lists.

These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown below.

11.3 Additional PORTB Pin Functions

PORTB pins RB<7:4> have an interrupt-on-change option. All PORTB pins have a weak pull-up option.

11.3.1 WEAK PULL-UPS

Each of the PORTB pins has an individually controlled weak internal pull-up. When set, each bit of the WPUB register enables the corresponding pin pull-up. When cleared, the RBPU bit of the INTCON2 register enables pull-ups on all pins which also have their corresponding WPUB bit set. When set, the RBPU 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<5:0> are configured as analog inputs by default and read as '0'; RB<7:6> are configured as digital inputs.

When the PBADEN Configuration bit is set to '0', RB<5:0> will alternatively be configured as digital inputs on POR.

11.3.2 INTERRUPT-ON-CHANGE

Four of the PORTB pins (RB<7:4>) 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 IOCE bit of the INTCON register enables interrupts on all pins which also have their corresponding IOCB bit set. When clear, the IOCE bit disables all interrupt-on-changes.

Only pins configured as inputs can cause this interrupt to occur (i.e., any RB<7:4> pin configured as an output is excluded from the interrupt-on-change comparison).

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 interrupt-on-change Interrupt flag bit (IOCIF) in the INTCON register.

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

- Any read or write of PORTB to clear the mismatch condition (except when PORTB is the source or destination of a MOVFF instruction).
- Execute at least one instruction after reading or writing PORTB, then clear the flag bit, IOCIF.

PIC18(L)F2X/45K50

A mismatch condition will continue to set the IOCIF flag bit. Reading or writing PORTB will end the mismatch condition and allow the IOCIF 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 IOCIF 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 IOCIF 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.

11.3.3 ALTERNATE FUNCTIONS

PORTB is multiplexed with several peripheral functions (Table 11-5). The pins have TTL input buffers. Some of these pin functions can be relocated to alternate pins using the Control fuse bits in CONFIG3H. RB3 is the default pin for SDO. Clearing the SDOMX bit moves the SDO pin function to RC7.

Two other pin functions, T3CKI and CCP2, can be relocated from their default pins to PORTB pins by clearing the control fuses in CONFIG3H. Clearing T3CMX and CCP2MX moves the pin functions to RB5 and RB3, respectively.

TABLE 11-5: PORTB I/O SUMMARY

| Pin | Function | TRIS Setting | ANSEL Setting | Pin Type | Buffer Type | Description |
|---|--------------------|--------------|---------------|----------|-------------------|--|
| RB0/INT0/FLT0/ SRI/SDA/SDI/AN12 | RB0 | 0 | x | O | DIG | LATB<0> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTB<0> data input; disabled when analog input enabled. |
| | INT0 | 1 | 0 | I | ST | External interrupt 0. |
| | FLT0 | 1 | 0 | I | ST | PWM Fault input for ECCP auto-shutdown. |
| | SRI | 1 | 0 | I | ST | SR latch input. |
| | SDA | 1 | 0 | I/O | I ² C™ | I ² C Data I/O (MSSP). |
| | SDI | 1 | 0 | I | ST | SPI Data in (MSSP). |
| RB1/INT1/P1C/ SCK/SCL/C12IN3-/ AN10 | AN12 | 1 | 1 | I | AN | Analog input 12. |
| | RB1 | 0 | x | O | DIG | LATB<1> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTB<1> data input; disabled when analog input enabled. |
| | INT1 | 1 | 0 | I | ST | External Interrupt 1. |
| | P1C ⁽³⁾ | 0 | 0 | O | DIG | Enhanced CCP1 PWM output 3. |
| | SCK | 0 | 0 | O | DIG | MSSP SPI Clock output. |
| | | 1 | 0 | I | ST | MSSP SPI Clock input. |
| | SCL | 0 | 0 | O | DIG | MSSP I ² C Clock output. |
| | | 1 | 0 | I | I ² C | MSSP I ² C Clock input. |
| | C12IN3- | 1 | 1 | I | AN | Comparators C1 and C2 inverting input. |
| | AN10 | 1 | 1 | I | AN | Analog input 10. |

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I²C = Schmitt Trigger input with I²C.

- Note 1:** Default pin assignment for SDO when Configuration bit SDOMX is set.
2: Alternate pin assignment for T3CKI and CCP2 when Configuration bits T3CMX and CCP2MX are clear.
3: Function is on PORTD/PORTE for PIC18(L)F45K50 devices.

TABLE 11-5: PORTB I/O SUMMARY (CONTINUED)

| Pin | Function | TRIS Setting | ANSEL Setting | Pin Type | Buffer Type | Description |
|--------------------------------|----------------------|--------------|---------------|----------|-------------|--|
| RB2/INT2/CTED1/P1B/AN8 | RB2 | 0 | x | O | DIG | LATB<2> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTB<2> data input; disabled when analog input enabled. |
| | INT2 | 1 | 0 | I | ST | External interrupt 2. |
| | CTED1 | 1 | 0 | I | ST | CTMU Edge 1 input. |
| | P1B ⁽³⁾ | 0 | 0 | O | DIG | Enhanced CCP1 PWM output 2. |
| | AN8 | 1 | 1 | I | AN | Analog input 8. |
| RB3/CTED2/CCP2/SDO/C12IN2-/AN9 | RB3 | 0 | x | O | DIG | LATB<3> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTB<3> data input; disabled when analog input enabled. |
| | CTED2 | 1 | 0 | I | ST | CTMU Edge 2 input. |
| | CCP2 ⁽²⁾ | 0 | 0 | O | DIG | Compare 2 output/PWM 2 output. |
| | | 1 | 0 | I | ST | Capture 2 input. |
| | SDO ⁽¹⁾ | 0 | 0 | O | DIG | MSSP SPI data output. |
| | C12IN2- | 1 | 1 | I | AN | Comparators C1 and C2 inverting input. |
| RB4/IOCB4/P1D/AN11 | RB4 | 0 | x | O | DIG | LATB<4> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTB<4> data input; disabled when analog input enabled. |
| | IOCB4 | 1 | 0 | I | TTL | Interrupt-on-change pin. |
| | P1D ⁽³⁾ | 0 | 0 | O | DIG | Enhanced CCP1 PWM output 4. |
| | AN11 | 1 | 1 | I | AN | Analog input 11. |
| RB5/IOCB5/T3CKI/T1G/AN13 | RB5 | 0 | x | O | DIG | LATB<5> data output; not affected by analog input. |
| | | 1 | 0 | I | TTL | PORTB<5> data input; disabled when analog input enabled. |
| | IOCB5 | 1 | 0 | I | TTL | Interrupt-on-change pin 1. |
| | T3CKI ⁽²⁾ | 1 | 0 | I | ST | Timer3 clock input. |
| | T1G | 1 | 0 | I | ST | Timer1 external clock gate input. |
| | AN13 | 1 | 1 | I | AN | Analog input 13. |
| RB6/IOCB6/PGC | RB6 | 0 | — | O | DIG | LATB<6> data output; not affected by analog input. |
| | | 1 | — | I | TTL | PORTB<6> data input; disabled when analog input enabled. |
| | IOCB6 | 1 | — | I | TTL | Interrupt-on-change pin. |
| | PGC | x | — | I | ST | In-Circuit Debugger and ICSP™ programming clock input. |
| RB7/IOCB7/PGD | RB7 | 0 | — | O | DIG | LATB<7> data output; not affected by analog input. |
| | | 1 | — | I | TTL | PORTB<7> data input; disabled when analog input enabled. |
| | IOCB7 | 1 | — | I | TTL | Interrupt-on-change pin. |
| | PGD | x | — | O | DIG | In-Circuit Debugger and ICSP™ programming data output. |
| | | x | — | I | ST | In-Circuit Debugger and ICSP™ programming data input. |

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal;
CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I²C = Schmitt Trigger input with I²C.

Note 1: Default pin assignment for SDO when Configuration bit SDOMX is set.

2: Alternate pin assignment for T3CKI and CCP2 when Configuration bits T3CMX and CCP2MX are clear.

3: Function is on PORTD/PORTE for PIC18(L)F45K50 devices.

PIC18(L)F2X/45K50

TABLE 11-6: REGISTERS ASSOCIATED WITH PORTB

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|---------|----------|-------------|--------------|---------------------|---------------------|------------|-------------|--------|------------------|
| ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 148 |
| ECCP1AS | ECCP1ASE | | ECCP1AS<2:0> | | PSS1AC<1:0> | | PSS1BD<1:0> | | 201 |
| CCP1CON | | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | 197 |
| CCP2CON | — | — | DC2B<1:0> | | CCP2M<3:0> | | | | 197 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| INTCON2 | RBPU | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | IOCIP | 115 |
| INTCON3 | INT2IP | INT1IP | — | INT2IE | INT1IE | — | INT2IF | INT1IF | 116 |
| IOCB | IOCB7 | IOCB6 | IOCB5 | IOCB4 | — | — | — | — | 151 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | 150 |
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | 146 |
| SLRCON | — | — | — | SLRE ⁽¹⁾ | SLRD ⁽¹⁾ | SLRC | SLRB | SLRA | 152 |
| T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/DONE | T1GVAL | T1GSS<1:0> | | 166 |
| T3CON | | TMR3CS<1:0> | | T3CKPS<1:0> | SOSCEN | T3SYNC | RD16 | TMR3ON | 165 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 |
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 | 150 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTB.

Note 1: Available on PIC18(L)F45K50 devices only.

TABLE 11-7: CONFIGURATION REGISTERS ASSOCIATED WITH PORTB

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|-------|-------|--------------------|--------|--------|------------------|
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |
| CONFIG4L | DEBUG | XINST | ICPRT | — | — | LVP ⁽¹⁾ | — | STRVEN | 377 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTB.

Note 1: Can only be changed when in high voltage programming mode.

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

PORTC is multiplexed with several peripheral functions ([Table 11-8](#)). The pins have Schmitt Trigger input buffers.

Some of these pin functions can be relocated to alternate pins using the Control fuse bits in CONFIG3H. RC0 is the default pin for T3CKI. Clearing the T3CMX bit moves the pin function to RB5. RC1 is the default pin for the CCP2 peripheral pin. Clearing the CCP2MX bit moves the pin function to the RB3 pin.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. The EUSART and MSSP peripherals override the TRIS bit to make a pin an output or an input, depending on the peripheral configuration. Refer to the corresponding peripheral section for additional information.

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

The contents of the TRISC register are affected by peripheral overrides. Reading TRISC always returns the current contents, even though a peripheral device may be overriding one or more of the pins.

EXAMPLE 11-3: INITIALIZING PORTC

```
MOVLB 0xF      ; Set BSR for banked SFRs
CLRF   LATC    ; Initialize PORTC by
                ; clearing 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
MOVLW  30h    ; Value used to
                ; enable digital inputs
MOVWF  ANSELC  ; RC<3:2> dig input enable
                ; No ANSEL bits for RC<1:0>
                ; RC<7:6> dig input enable
```

11.4.1 PORTC OUTPUT PRIORITY

Each PORTC pin is multiplexed with other functions. The pins, their combined functions and their output priorities are briefly described here. For additional information, refer to the appropriate section in this data sheet.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the higher priority. [Table 11-4](#) lists the PORTC pin functions from the highest to the lowest priority.

Analog input functions, such as ADC, comparator and SR latch inputs, are not shown in the priority lists.

These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown below.

11.4.2 INTERRUPT-ON-CHANGE

All of the PORTC pins (RC<7:4> and RC<2:0>) are individually configurable as interrupt-on-change pins. Control bits in the IOCC register enable (when set) or disable (when clear) the interrupt function for each pin. See [Section 11.3.2 “Interrupt-on-Change”](#) for details on operation of interrupt-on-change.

PIC18(L)F2X/45K50

TABLE 11-8: PORTC I/O SUMMARY

| Pin Name | Function | TRIS Setting | ANSEL setting | Pin Type | Buffer Type | Description |
|---------------------------------|----------------------|--------------|---------------|----------|-------------|--|
| RC0/IOCC0/T3CKI/T3G/T1CKI/SOSCO | RC0 | 0 | — | O | DIG | LATC<0> data output; not affected by analog input. |
| | | 1 | — | I | ST | PORTC<0> data input; disabled when analog input enabled. |
| | IOCC0 | 1 | — | I | TTL | Interrupt-on-change pin. |
| | T3CKI ⁽¹⁾ | 1 | — | I | ST | Timer3 clock input. |
| | T3G | 1 | — | I | ST | Timer3 external clock gate input. |
| | T1CKI | 1 | — | I | ST | Timer1 clock input. |
| | SOSCO | x | — | O | XTAL | Secondary oscillator output. |
| RC1/IOCC1/CCP2/SOSCI | RC1 | 0 | — | O | DIG | LATC<1> data output; not affected by analog input. |
| | | 1 | — | I | ST | PORTC<1> data input; disabled when analog input enabled. |
| | IOCC1 | 1 | — | I | TTL | Interrupt-on-change pin. |
| | CCP2 ⁽¹⁾ | 0 | — | O | DIG | Compare 2 output/PWM 2 output. |
| | | 1 | — | I | ST | Capture 2 input. |
| | SOSCI | x | — | I | XTAL | Secondary oscillator input. |
| RC2/CTPLS/P1A/CCP1/IOCC2/AN14 | RC2 | 0 | 0 | O | DIG | LATC<2> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTC<2> data input; disabled when analog input enabled. |
| | CTPLS | 0 | 0 | O | DIG | CTMU pulse generator output. |
| | P1A | 0 | 0 | O | DIG | Enhanced CCP1 PWM output 1. |
| | CCP1 | 0 | 0 | O | DIG | Compare 1 output/PWM 1 output. |
| | | 1 | 0 | I | ST | Capture 1 input. |
| | IOCC2 | 1 | — | I | TTL | Interrupt-on-change pin. |
| | AN14 | 1 | 1 | I | AN | Analog input 14. |
| D-/IOCC4 | D- | — | — | I | XCVR | USB bus differential minus line input. |
| | | — | — | O | XCVR | USB bus differential minus line output. |
| | IOCC4 | — | — | I | ST | Interrupt-on-change pin. |
| D+/IOCC5 | D+ | — | — | I | XCVR | USB bus differential minus line input. |
| | | — | — | O | XCVR | USB bus differential minus line output. |
| | IOCC5 | — | — | I | ST | Interrupt-on-change pin. |

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I²C™ = Schmitt Trigger input with I²C.

Note 1: Default pin assignment for T3CKI and CCP2 when Configuration bits T3CMX and CCP2MX are set.

2: Alternate pin assignment for SDO when Configuration bit SDOMX is clear.

3: Function is on PORTD/PORTE for PIC18(L)F45K50 devices.

TABLE 11-8: PORTC I/O SUMMARY (CONTINUED)

| Pin Name | Function | TRIS Setting | ANSEL setting | Pin Type | Buffer Type | Description |
|------------------------------|--------------------|--------------|---------------|----------|-------------|--|
| RC6/IOCC6/TX/CK/ AN18 | RC6 | 0 | 0 | O | DIG | LATC<6> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTC<6> data input; disabled when analog input enabled. |
| | IOCC6 | 1 | 0 | I | TTL | Interrupt-on-change pin. |
| | TX | 1 | 0 | O | DIG | EUSART asynchronous transmit data output. |
| | CK | 1 | 0 | O | DIG | EUSART synchronous serial clock output. |
| | | 1 | 0 | I | ST | EUSART synchronous serial clock input. |
| | AN18 | 1 | 1 | I | AN | Analog input 18. |
| RC7/IOCC7/SDO/RX/ DT/AN19 | RC7 | 0 | 0 | O | DIG | LATC<7> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTC<7> data input; disabled when analog input enabled. |
| | IOCC7 | 1 | 0 | I | TTL | Interrupt-on-change pin. |
| | SDO ⁽²⁾ | 1 | 0 | O | DIG | Alternate MSSP SPI data output. |
| | RX | 1 | 0 | I | ST | EUSART asynchronous receive data in. |
| | DT | 1 | 0 | O | DIG | EUSART synchronous serial data output. |
| | | 1 | 0 | I | ST | EUSART synchronous serial data input. |
| | AN19 | 1 | 1 | I | AN | Analog input 19. |

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal;
CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I²C™ = Schmitt Trigger input with I²C.

Note 1: Default pin assignment for T3CKI and CCP2 when Configuration bits T3CMX and CCP2MX are set.

2: Alternate pin assignment for SDO when Configuration bit SDOMX is clear.

3: Function is on PORTD/PORTE for PIC18(L)F45K50 devices.

PIC18(L)F2X/45K50

TABLE 11-9: REGISTERS ASSOCIATED WITH PORTC

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page | | |
|----------------------|-------------|--------------|-------------|---------------------|---------------------|---------------|-------------|---------------------|------------------|--|--|
| ANSEL _C | ANSC7 | ANSC6 | — | — | — | ANSC2 | — | — | 148 | | |
| ECCP1AS | ECCP1ASE | ECCP1AS<2:0> | | | PSS1AC<1:0> | | PSS1BD<1:0> | | 201 | | |
| CCP1CON | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | | | | |
| CCP2CON | — | — | DC2B<1:0> | | CCP2M<3:0> | | | | | | |
| CTMUONH | CTMUEN | — | CTMUSIDL | TGEN | EDGEN | EDGSEQEN | IDISSEN | CTTRIG | 322 | | |
| LATC | LATC7 | LATC6 | — | — | — | LATC2 | LATC1 | LATC0 | 150 | | |
| PORTC | RC7 | RC6 | — | — | — | RC2 | RC1 | RC0 | 146 | | |
| RCSTA ₁ | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 269 | | |
| SLRCON | — | — | — | SLRE ⁽¹⁾ | SLRD ⁽¹⁾ | SLRC | SLRB | SLRA | 152 | | |
| SSP1CON ₁ | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | | | |
| T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | SOSCEN | <u>T1SYNC</u> | RD16 | TMR1ON | 165 | | |
| T3CON | TMR3CS<1:0> | | T3CKPS<1:0> | | SOSCEN | <u>T3SYNC</u> | RD16 | TMR3ON | 165 | | |
| T3GCON | TMR3GE | T3GPOL | T3GTM | T3GSPM | T3GGO/DONE | T3GVAL | T3GSS | | 166 | | |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISCO ₀ | 149 | | |
| TXSTA ₁ | CSRC | TX9 | TXEN | SYNC | SEND _B | BRGH | TRMT | RX9D | 268 | | |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTC.

Note 1: Available on PIC18(L)F45K50 devices only.

TABLE 11-10: CONFIGURATION REGISTERS ASSOCIATED WITH PORTC

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|-------|-------|-------|--------|--------|------------------|
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTC.

11.5 PORTD Registers

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

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

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

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

All of the PORTD pins are multiplexed with analog and digital peripheral modules. See [Table 11-11](#).

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

EXAMPLE 11-4: INITIALIZING PORTD

```
MOVLB 0xF      ; Set BSR for banked SFRs
CLRF  LATD    ; Initialize PORTD by
                ; clearing output
                ; data latches
MOVLW 0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISD   ; Set RD<3:0> as inputs
                ; RD<5:4> as outputs
                ; RD<7:6> as inputs
MOVLW 30h     ; Value used to
                ; enable digital inputs
MOVWF  ANSELD  ; RD<3:0> dig input enable
                ; RC<7:6> dig input enable
```

11.5.1 PORTD OUTPUT PRIORITY

Each PORTD pin is multiplexed with other functions. The pins, their combined functions and their output priorities are briefly described here. For additional information, refer to the appropriate section in this data sheet.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the higher priority. [Table 11-4](#) lists the PORTD pin functions from the highest to the lowest priority.

Analog input functions, such as ADC, comparator and SR latch inputs, are not shown in the priority lists.

These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown below.

PIC18(L)F2X/45K50

TABLE 11-11: PORTD I/O SUMMARY

| Pin Name | Function | TRIS Setting | ANSEL setting | Pin Type | Buffer Type | Description |
|--------------|----------|--------------|---------------|----------|-------------|--|
| RD0/AN20 | RD0 | 0 | 0 | O | DIG | LATD<0> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTD<0> data input; disabled when analog input enabled. |
| | AN20 | 1 | 1 | I | AN | Analog input 20. |
| RD1/AN21 | RD1 | 0 | 0 | O | DIG | LATD<1> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTD<1> data input; disabled when analog input enabled. |
| | AN21 | 1 | 1 | I | AN | Analog input 21. |
| RD2/AN22 | RD2 | 0 | 0 | O | DIG | LATD<2> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTD<2> data input; disabled when analog input enabled. |
| | AN22 | 1 | 1 | I | AN | Analog input 22. |
| RD3/AN23 | RD3 | 0 | 0 | O | DIG | LATD<3> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTD<3> data input; disabled when analog input enabled. |
| | AN23 | 1 | 1 | I | AN | Analog input 23. |
| RD4/AN24 | RD4 | 0 | 0 | O | DIG | LATD<4> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTD<4> data input; disabled when analog input enabled. |
| | AN24 | 1 | 1 | I | AN | Analog input 24. |
| RD5/P1B/AN25 | RD5 | 0 | 0 | O | DIG | LATD<5> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTD<5> data input; disabled when analog input enabled. |
| | P1B | 0 | 0 | O | DIG | Enhanced CCP1 PWM output 2. |
| | AN25 | 1 | 1 | I | AN | Analog input 25. |
| RD6/P1C/AN26 | RD6 | 0 | 0 | O | DIG | LATD<6> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTD<6> data input; disabled when analog input enabled. |
| | P1C | 0 | 0 | O | DIG | Enhanced CCP1 PWM output 3. |
| | AN26 | 1 | 1 | I | AN | Analog input 26. |
| RD7/P1D/AN27 | RD7 | 0 | 0 | O | DIG | LATD<7> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTD<7> data input; disabled when analog input enabled. |
| | P1D | 0 | 0 | O | DIG | Enhanced CCP1 PWM output 4. |
| | AN27 | 1 | 1 | I | AN | Analog input 27. |

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal;
CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I²C™ = Schmitt Trigger input with I²C.

TABLE 11-12: REGISTERS ASSOCIATED WITH PORTD

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|-----------------------|----------|--------|-----------|--------|------------|--------|--------|--------|------------------|
| ANSELD ⁽¹⁾ | ANSD7 | ANSD6 | ANSD5 | ANSD4 | ANSD3 | ANSD2 | ANSD1 | ANSD0 | 148 |
| CCP1CON | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | | 197 |
| LATD ⁽¹⁾ | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | 150 |
| PORTD ⁽¹⁾ | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | 146 |
| SLRCON ⁽¹⁾ | — | — | — | SLRE | SLRD | SLRC | SLRB | SLRA | 152 |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 252 |
| TRISD ⁽¹⁾ | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 149 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTD.

Note 1: Available on PIC18(L)F45K50 devices only.

PIC18(L)F2X/45K50

11.6 PORTE Registers

Depending on the particular PIC18(L)F2X/45K50 device selected, PORTE is implemented in two different ways.

11.6.1 PORTE ON 40/44-PIN DEVICES

For PIC18(L)F2X/45K50 devices, PORTE is a 4-bit wide port. Three pins (RE0/AN5, RE1/AN6 and RE2/AN7) are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. When selected as an analog input, these pins will read as '0's.

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

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

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

Note: On a Power-on Reset, RE<2:0> are configured as analog inputs.

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

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

EXAMPLE 11-5: INITIALIZING PORTE

```
CLRF    LATE      ; Initialize PORTE by
                  ; clearing output
                  ; data latches
CLRF    ANSELE   ; Configure analog pins
                  ; for digital only
MOVlw  05h      ; Value used to
                  ; initialize data
                  ; direction
MOVwf  TRISE    ; Set RE<0> as input
                  ; RE<1> as output
                  ; RE<2> as input
```

11.6.2 PORTE ON 28-PIN DEVICES

For PIC18(L)F2XK50 devices, PORTE is only available when Master Clear functionality is disabled (MCLR = 0). In these cases, PORTE is a single bit, input-only port comprised of RE3 only. The pin operates as previously described.

11.6.3 RE3 WEAK PULL-UP

The port RE3 pin has an individually controlled weak internal pull-up. When set, the WPUE3 (TRISE<7>) bit enables the RE3 pin pull-up. The RBPU bit of the INTCON2 register controls pull-ups on both PORTB and PORTE. When RBPU = 0, the weak pull-ups become active on all pins which have the WPUE3 or WPUBx bits set. When set, the RBPU bit disables all weak pull-ups. The pull-ups are disabled on a Power-on Reset. When the RE3 port pin is configured as MCLR, (CONFIG3H<7>, MCLRE = 1 and CONFIG4L<2>, LVP = 0), or configured for Low-Voltage Programming, (MCLRE = x and LVP = 1), the pull-up is always enabled and the WPUE3 bit has no effect.

11.6.4 PORTE OUTPUT PRIORITY

Each PORTE pin is multiplexed with other functions. The pins, their combined functions and their output priorities are briefly described here. For additional information, refer to the appropriate section in this data sheet.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the higher priority. Table 11-4 lists the PORTE pin functions from the highest to the lowest priority.

Analog input functions, such as ADC, comparator and SR latch inputs, are not shown in the priority lists.

These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown below.

TABLE 11-13: PORTE I/O SUMMARY

| Pin | Function | TRIS Setting | ANSEL Setting | Pin Type | Buffer Type | Description |
|--------------|----------|--------------|---------------|----------|-------------|---|
| RE0/AN5 | RE0 | 0 | 0 | O | DIG | LATE<0> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTE<0> data input; disabled when analog input enabled. |
| | AN5 | 1 | 1 | I | AN | Analog input 5. |
| RE1/AN6 | RE1 | 0 | 0 | O | DIG | LATE<1> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTE<1> data input; disabled when analog input enabled. |
| | AN6 | 1 | 1 | I | AN | Analog input 6. |
| RE2/AN7 | RE2 | 0 | 0 | O | DIG | LATE<2> data output; not affected by analog input. |
| | | 1 | 0 | I | ST | PORTE<2> data input; disabled when analog input enabled. |
| | AN7 | 1 | 1 | I | AN | Analog input 7. |
| RE3/VPP/MCLR | RE3 | — | — | I | ST | PORTE<3> data input; enabled when Configuration bit MCLRE = 0. |
| | VPP | — | — | P | AN | Programming voltage input; always available |
| | MCLR | — | — | I | ST | Active-low Master Clear (device Reset) input; enabled when configuration bit MCLRE = 1. |

Legend: AN = Analog input or output; TTL = TTL compatible input; HV = High Voltage; OD = Open Drain; XTAL = Crystal; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I²C™ = Schmitt Trigger input with I²C.

TABLE 11-14: REGISTERS ASSOCIATED WITH PORTE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|-----------------------|-------|---------|---------|---------------------|---------------------|-----------------------|-----------------------|-----------------------|------------------|
| ANSELE ⁽¹⁾ | — | — | — | — | — | ANSE2 | ANSE1 | ANSE0 | 149 |
| INTCON2 | RBPU | INTEDG0 | INTEDG1 | INTEDG2 | — | TMR0IP | — | IOCIP | 115 |
| LATE ⁽¹⁾ | — | — | — | — | — | LATE2 | LATE1 | LATE0 | 150 |
| PORTE | — | — | — | — | RE3 | RE2 ⁽¹⁾ | RE1 ⁽¹⁾ | RE0 ⁽¹⁾ | 147 |
| SLRCON | — | — | — | SLRE ⁽¹⁾ | SLRD ⁽¹⁾ | SLRC | SLRB | SLRA | 152 |
| TRISE | WPUE3 | — | — | — | — | TRISE2 ⁽¹⁾ | TRISE1 ⁽¹⁾ | TRISE0 ⁽¹⁾ | 149 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for PORTE.

Note 1: Available on PIC18(L)F45K50 devices only.

TABLE 11-15: CONFIGURATION REGISTERS ASSOCIATED WITH PORTE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|-------|-------|--------------------|--------|--------|------------------|
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |
| CONFIG4L | DEBUG | XINST | — | — | — | LVP ⁽¹⁾ | — | STRVEN | 377 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for interrupts.

Note 1: Can only be changed when in high-voltage programming mode.

PIC18(L)F2X/45K50

11.7 Port Analog Control

Most 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 ANSELA, ANSELB, ANSELC, ANSELD and ANSELE 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. This can cause unexpected behavior when performing read-modify-write operations on the affected port.

All ANSEL register bits default to '1' upon POR and BOR, disabling digital inputs for their associated port pins. All TRIS register bits default to '1' upon POR or BOR, disabling digital outputs for their associated port pins. As a result, all port pins that have an ANSEL register will default to analog inputs upon POR or BOR.

11.9 Register Definitions – Port Control

REGISTER 11-1: PORTx⁽¹⁾: PORTx REGISTER

| R/W-u/x |
|---------|---------|---------|---------|---------|---------|---------|---------|
| Rx7 | Rx6 | Rx5 | Rx4 | Rx3 | Rx2 | Rx1 | Rx0 |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 Rx<7:0>: PORTx I/O bit values⁽²⁾

Note 1: Register Description for PORTA, PORTB, PORTC and PORTD.

2: Writes to PORTx are written to corresponding LATx register. Reads from PORTx register is return of I/O pin values.

REGISTER 11-2: PORTE: PORTE REGISTER

| U-0 | U-0 | U-0 | U-0 | R/W-u/x | R/W-u/x | R/W-u/x | R/W-u/x |
|-------|-----|-----|-----|--------------------|-------------------------|-------------------------|-------------------------|
| — | — | — | — | RE3 ⁽¹⁾ | RE2 ^{(2), (3)} | RE1 ^{(2), (3)} | RE0 ^{(2), (3)} |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

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

bit 3 **RE3:** PORTE Input bit value⁽¹⁾

bit 2-0 **RE<2:0>:** PORTE I/O bit values^{(2), (3)}

Note 1: Port is available as input-only when MCLRE = 0.

2: Writes to PORTx are written to corresponding LATx register. Reads from PORTx register is return of I/O pin values.

3: Available on PIC18(L)F45K50 devices.

REGISTER 11-3: ANSELA – PORTA ANALOG SELECT REGISTER

| U-0 | U-0 | R/W-1 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|-------|-----|-------|-----|-------|-------|-------|-------|
| — | — | ANSA5 | — | ANSA3 | ANSA2 | ANSA1 | ANSA0 |
| 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 **ANSA5:** RA5 Analog Select bit

1 = Digital input buffer disabled

0 = Digital input buffer enabled

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

bit 3-0 **ANSA<3:0>:** RA<3:0> Analog Select bit

1 = Digital input buffer disabled

0 = Digital input buffer enabled

PIC18(L)F2X/45K50

REGISTER 11-4: ANSELB – PORTB ANALOG SELECT REGISTER

| U-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 |
| 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 **ANSB<5:0>:** RB<5:0> Analog Select bit

1 = Digital input buffer disabled

0 = Digital input buffer enabled

REGISTER 11-5: ANSELC – PORTC ANALOG SELECT REGISTER

| R/W-1 | R/W-1 | U-0 | U-0 | U-0 | R/W-1 | U-0 | U-0 |
|-------|-------|-----|-----|-----|-------|-----|-----|
| ANSC7 | ANSC6 | — | — | — | ANSC2 | — | — |
| 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 **ANSC<7:6>:** RC<7:6> Analog Select bit

1 = Digital input buffer disabled

0 = Digital input buffer enabled

bit 5-3 **Unimplemented:** Read as '0'

bit 2 **ANSC<2>:** RC<2> Analog Select bit

1 = Digital input buffer disabled

0 = Digital input buffer enabled

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

REGISTER 11-6: ANSELD – PORTD ANALOG SELECT REGISTER

| R/W-1 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| ANSD7 | ANSD6 | ANSD5 | ANSD4 | ANSD3 | ANSD2 | ANSD1 | ANSD0 |
| 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 **ANSD<7:0>:** RD<7:0> Analog Select bit

1 = Digital input buffer disabled

0 = Digital input buffer enabled

REGISTER 11-7: ANSELE – PORTE ANALOG SELECT REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----|----------------------|----------------------|----------------------|
| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-1 | R/W-1 | R/W-1 |
| — | — | — | — | — | ANSE2 ⁽¹⁾ | ANSE1 ⁽¹⁾ | ANSE0 ⁽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 7-3 **Unimplemented:** Read as '0'

bit 2-0 **ANSE<2:0>:** RE<2:0> Analog Select bit⁽¹⁾

1 = Digital input buffer disabled

0 = Digital input buffer enabled

Note 1: Available on PIC18(L)F45K50 devices only.

REGISTER 11-8: TRISx: PORTx TRI-STATE REGISTER⁽¹⁾

| R/W-1 |
|--------|--------|--------|--------|--------|--------|--------|--------|
| TRISx7 | TRISx6 | TRISx5 | TRISx4 | TRISx3 | TRISx2 | TRISx1 | TRISx0 |
| 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 **TRISx<7:0>:** PORTx Tri-State Control bit

1 = PORTx pin configured as an input (tri-stated)

0 = PORTx pin configured as an output

Note 1: Register description for TRISA, TRISB, TRISC and TRISD.

REGISTER 11-9: TRISE: PORTE TRI-STATE REGISTER

| R/W-1 | U-0 | U-0 | U-0 | U-0 | R/W-1 | R/W-1 | R/W-1 |
|-------|-----|-----|-----|-----|-----------------------|-----------------------|-----------------------|
| WPUE3 | — | — | — | — | TRISE2 ⁽¹⁾ | TRISE1 ⁽¹⁾ | TRISE0 ⁽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 **WPUE3:** Weak Pull-up Register bits

1 = Pull-up enabled on PORT pin

0 = Pull-up disabled on PORT pin

bit 6-3 **Unimplemented:** Read as '0'

bit 2-0 **TRISE<2:0>:** PORTE Tri-State Control bit⁽¹⁾

1 = PORTE pin configured as an input (tri-stated)

0 = PORTE pin configured as an output

Note 1: Available on PIC18(L)F45K50 devices only.

PIC18(L)F2X/45K50

REGISTER 11-10: LATx: PORTx OUTPUT LATCH REGISTER⁽¹⁾

| R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| LATx7 | LATx6 | LATx5 | LATx4 | LATx3 | LATx2 | LATx1 | LATx0 |
| 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 **LATx<7:0>**: PORTx Output Latch bit value⁽²⁾

Note 1: Register Description for LATA, LATB, LATC and LATD.

2: Writes to PORTA are written to corresponding LATA register. Reads from PORTA register is return of I/O pin values.

REGISTER 11-11: LATE: PORTE OUTPUT LATCH REGISTER⁽¹⁾

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-x/u | R/W-x/u | R/W-x/u |
|-------|-------|-----|-----|-----|---------|---------|---------|
| — | — | — | — | — | LATE2 | LATE1 | LATE0 |
| 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 2-0 **LATE<2:0>**: PORTE Output Latch bit value⁽²⁾

Note 1: Available on PIC18(L)F45K50 devices only.

2: Writes to PORTE are written to corresponding LATE register. Reads from PORTE register is return of I/O pin values.

REGISTER 11-12: WPUB: WEAK PULL-UP PORTB REGISTER

| R/W-1 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 |
| 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 **WPUB<7:0>**: Weak Pull-up Register bits

1 = Pull-up enabled on PORTB pin

0 = Pull-up disabled on PORTB pin

REGISTER 11-13: IOCB: INTERRUPT-ON-CHANGE PORTB CONTROL 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 PORTB control bits

1 = Interrupt-on-change enabled⁽¹⁾

0 = Interrupt-on-change disabled

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

Note 1: Interrupt-on-change requires that the IOCTIE bit (INTCON<3>) is set.

REGISTER 11-14: IOCC: INTERRUPT-ON-CHANGE PORTC CONTROL REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-------|-------|-----|-------|-------|-------|
| IOCC7 | IOCC6 | IOCC5 | IOCC4 | — | IOCC2 | IOCC1 | IOCC0 |
| 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 **IOCC<7:4>**: Interrupt-on-Change PORTC control bits

1 = Interrupt-on-change enabled⁽¹⁾

0 = Interrupt-on-change disabled

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

bit 2-0 **IOCC<2:0>**: Interrupt-on-Change PORTC control bits

1 = Interrupt-on-change enabled⁽¹⁾

0 = Interrupt-on-change disabled

Note 1: Interrupt-on-change requires that the IOCTIE bit (INTCON<3>) is set.

PIC18(L)F2X/45K50

REGISTER 11-15: SLRCON: SLEW RATE CONTROL REGISTER

| U-0 | U-0 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|-------|-----|-----|---------------------|---------------------|-------|-------|-------|
| — | — | — | SLRE ⁽¹⁾ | SLRD ⁽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 7-5 **Unimplemented:** Read as '0'

bit 4 **SLRE:** PORTE Slew Rate Control bit⁽¹⁾

1 = All outputs on PORTE slew at a limited rate

0 = All outputs on PORTE slew at the standard rate

bit 3 **SLRD:** PORTD Slew Rate Control bit⁽¹⁾

1 = All outputs on PORTD slew at a limited rate

0 = All outputs on PORTD slew at the standard rate

bit 2 **SLRC:** PORTC Slew Rate Control bit

1 = All outputs on PORTC slew at a limited 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 a limited 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 a limited rate⁽²⁾

0 = All outputs on PORTA slew at the standard rate

Note 1: These bits are available on PIC18(L)F45K50 devices.

2: The slew rate of RA6 defaults to standard rate when the pin is used as CLKO.

12.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

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

The T0CON register ([Register 12-1](#)) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in [Figure 12-1](#). [Figure 12-2](#) shows a simplified block diagram of the Timer0 module in 16-bit mode.

12.1 Register Definitions: Timer0 Control

REGISTER 12-1: 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 | TOPS<2:0> | | |
| bit 7 | | | | | | | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

| | |
|---------|---|
| bit 7 | 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 (Fosc/4) |
| 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 | TOPS<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 |

PIC18(L)F2X/45K50

12.2 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 12.4 "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 pin RA4/T0CKI. 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 29-22](#)) to ensure that the external clock can be synchronized with the internal phase clock ($Tosc$). There is a delay between synchronization and the onset of incrementing the timer/counter.

12.3 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 12-2](#)). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without 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 12-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)

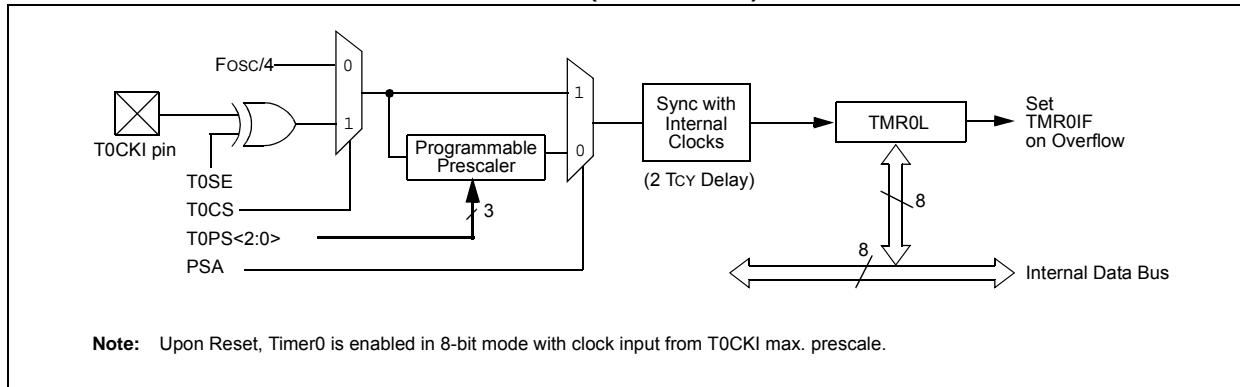
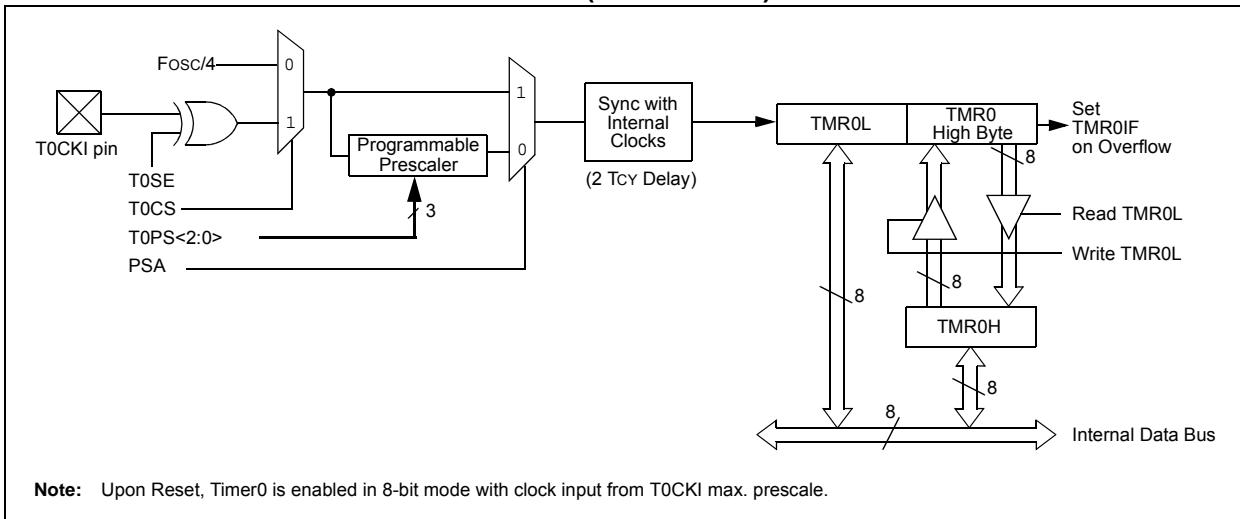


FIGURE 12-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)



12.4 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.

12.4.1 SWITCHING PRESCALER ASSIGNMENT

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

12.5 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 12-1: REGISTERS ASSOCIATED WITH TIMER0

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|---------|----------------------------|-----------|---------|---------|--------|-----------|--------|--------|---------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| INTCON2 | RBPU | INTEGD0 | INTEGD1 | INTEGD2 | — | TMR0IP | — | IOCIP | 115 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | TOPS<2:0> | | | |
| TMR0H | Timer0 Register, High Byte | | | | | | | | — |
| TMR0L | Timer0 Register, Low Byte | | | | | | | | — |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 149 |

Legend: — = unimplemented locations, read as ‘0’. Shaded bits are not used by Timer0.

13.0 TIMER1/3 MODULE WITH GATE CONTROL

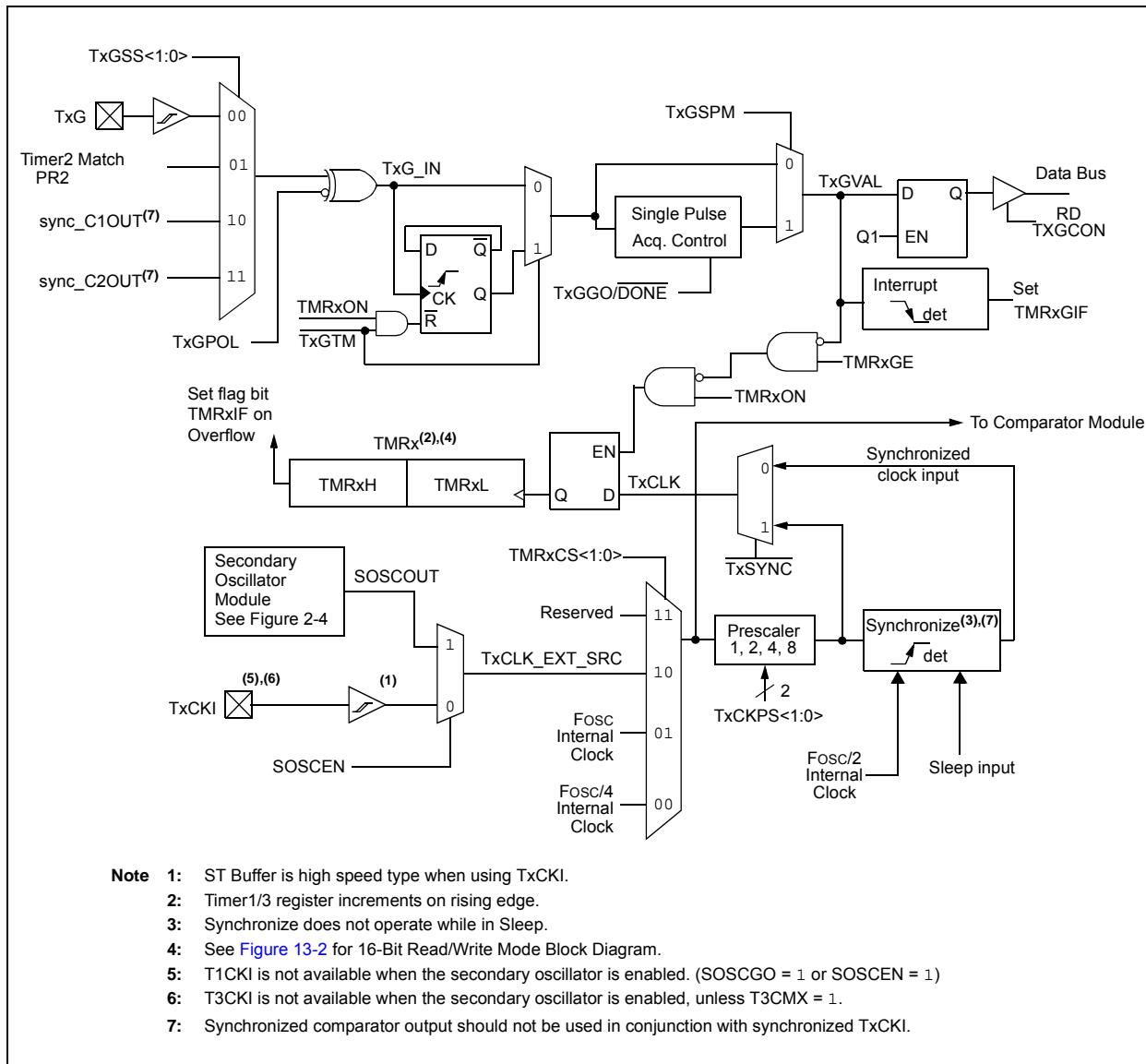
The Timer1/3 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMRxH:TMRxL)
- Programmable internal or external clock source
- 2-bit prescaler
- Dedicated Secondary 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1/3 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- 16-Bit Read/Write Operation
- Time base for the Capture/Compare function

- Special Event Trigger (with CCP/ECCP)
- Selectable Gate Source Polarity
- Gate Toggle mode
- Gate Single-pulse mode
- Gate Value Status
- Gate Event Interrupt

[Figure 13-1](#) is a block diagram of the Timer1/3 module.

FIGURE 13-1: TIMER1/3 BLOCK DIAGRAM



13.1 Timer1/3 Operation

The Timer1/3 module is a 16-bit incrementing counter which is accessed through the TMRxH:TMRxL register pair. Writes to TMRxH or TMRxL directly update the counter.

When used with an internal clock source, the module is a timer and increments on every instruction cycle. When used with an external clock source, the module can be used as either a timer or counter and increments on every selected edge of the external source.

Timer1/3 is enabled by configuring the TMRxON and TMRxGE bits in the TxCON and TxGCON registers, respectively. [Table 13-1](#) displays the Timer1/3 enable selections.

TABLE 13-1: TIMER1/3 ENABLE SELECTIONS

| TMRxON | TMRxGE | Timer1/3 Operation |
|--------|--------|--------------------|
| 0 | 0 | Off |
| 0 | 1 | Off |
| 1 | 0 | Always On |
| 1 | 1 | Count Enabled |

13.2 Clock Source Selection

The TMRxCS<1:0> and SOSCEN bits of the TxCON register are used to select the clock source for Timer1/3. The dedicated secondary oscillator circuit can be used as the clock source for Timer1 and Timer3, simultaneously. Any of the SOSCEN bits will enable the secondary oscillator circuit and select it as the clock source for that particular timer. [Table 13-2](#) displays the clock source selections.

13.2.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected the TMRxH:TMRxL register pair will increment on multiples of Fosc as determined by the Timer1/3 prescaler.

When the Fosc internal clock source is selected, the Timer1/3 register value will increment by four counts every instruction clock cycle. Due to this condition, a 2 LSB error in resolution will occur when reading the Timer1/3 value. To utilize the full resolution of Timer1/3, an asynchronous input signal must be used to gate the Timer1/3 clock input.

The following asynchronous sources may be used:

- Asynchronous event on the TxG pin to Timer1/3 gate
- C1 or C2 comparator input to Timer1/3 gate

13.2.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1/3 module may work as a timer or a counter.

When enabled to count, Timer1/3 is incremented on the rising edge of the external clock input of the TxCKI pin. This external clock source can be synchronized to the microcontroller system clock or it can run asynchronously.

When used as a timer with a clock oscillator, an external 32.768 kHz crystal can be used in conjunction with the dedicated secondary internal oscillator circuit.

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:

- Timer1/3 enabled after POR
- Write to TMRxH or TMRxL
- Timer1/3 is disabled
- Timer1/3 is disabled (TMRxON = 0) when TxCKI is high then Timer1/3 is enabled (TMRxON=1) when TxCKI is low.

TABLE 13-2: CLOCK SOURCE SELECTIONS

| TMRxCS1 | TMRxCS0 | SOSCEN | Clock Source |
|---------|---------|--------|--|
| 0 | 1 | x | System Clock (Fosc) |
| 0 | 0 | x | Instruction Clock (Fosc/4) |
| 1 | 0 | 0 | External Clocking on TxCKI Pin |
| 1 | 0 | 1 | Oscillator Circuit on SOSCI/SOSCO Pins |

13.3 Timer1/3 Prescaler

Timer1/3 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The TxCKPS bits of the TxCON register control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMRxH or TMRxL.

13.4 Secondary Oscillator

A dedicated secondary low-power 32.768 kHz oscillator circuit is built-in between pins SOSCI (input) and SOSCO (amplifier output). This internal circuit is to be used in conjunction with an external 32.768 kHz crystal.

The oscillator circuit is enabled by setting the SOSCEN bit of the TxCON register, the SOSCGO bit of the OSCCON2 register or by selecting the secondary oscillator as the system clock by setting SCS<1:0> = 01 in the OSCCON register. The oscillator will continue to run during Sleep.

Note: The oscillator requires a start-up and stabilization time before use. Thus, SOSCEN should be set and a suitable delay observed prior to enabling Timer1/3.

13.5 Timer1/3 Operation in Asynchronous Counter Mode

If control bit TxSYNC of the TxCON register is set, the external clock input is not synchronized. The timer increments asynchronously to the internal phase clocks. If external clock source is selected then the timer will continue to run during Sleep and can generate an interrupt on overflow, which will wake-up the processor. However, special precautions in software are needed to read/write the timer (see [Section 13.5.1 “Reading and Writing Timer1/3 in Asynchronous Counter Mode”](#)).

Note: When switching from synchronous to asynchronous operation, it is possible to skip an increment. When switching from asynchronous to synchronous operation, it is possible to produce an additional increment.

13.5.1 READING AND WRITING TIMER1/3 IN ASYNCHRONOUS COUNTER MODE

Reading TMRxH or TMRxL while the timer is running from an external asynchronous clock will ensure a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since the timer may overflow between the reads. For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers, while the register is incrementing. This may produce an unpredictable value in the TMRxH:TMRxL register pair.

13.6 Timer1/3 16-Bit Read/Write Mode

Timer1/3 can be configured to read and write all 16 bits of data, to and from, the 8-bit TMRxL and TMRxH registers, simultaneously. The 16-bit read and write operations are enabled by setting the RD16 bit of the TxCON register.

To accomplish this function, the TMRxH register value is mapped to a buffer register called the TMRxH buffer register. While in 16-Bit mode, the TMRxH register is not directly readable or writable and all read and write operations take place through the use of this TMRxH buffer register.

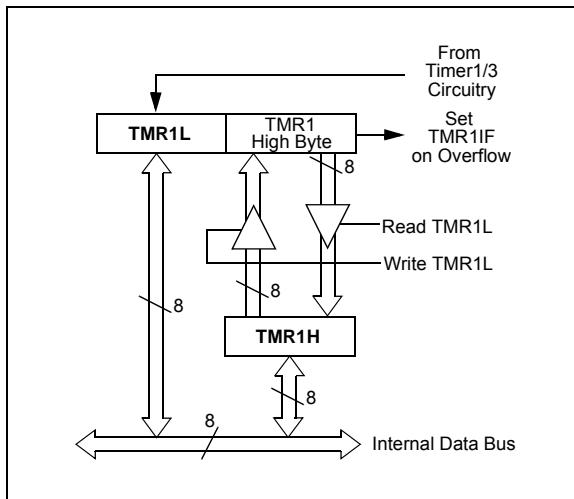
When a read from the TMRxL register is requested, the value of the TMRxH register is simultaneously loaded into the TMRxH buffer register. When a read from the TMRxH register is requested, the value is provided from the TMRxH buffer register instead. This provides the user with the ability to accurately read all 16 bits of the Timer1/3 value from a single instance in time.

In contrast, when not in 16-Bit mode, the user must read each register separately and determine if the values have become invalid due to a rollover that may have occurred between the read operations.

When a write request of the TMRxL register is requested, the TMRxH buffer register is simultaneously updated with the contents of the TMRxH register. The value of TMRxH must be preloaded into the TMRxH buffer register prior to the write request for the TMRxL register. This provides the user with the ability to write all 16 bits to the TMRxL:TMRxH register pair at the same time.

Any requests to write to the TMRxH directly does not clear the Timer1/3 prescaler value. The prescaler value is only cleared through write requests to the TMRxL register.

FIGURE 13-2: TIMER1/3 16-BIT READ/WRITE MODE BLOCK DIAGRAM



13.7 Timer1/3 Gate

Timer1/3 can be configured to count freely or the count can be enabled and disabled using Timer1/3 gate circuitry. This is also referred to as Timer1/3 Gate Enable.

Timer1/3 gate can also be driven by multiple selectable sources.

13.7.1 TIMER1/3 GATE ENABLE

The Timer1/3 Gate Enable mode is enabled by setting the TMRxGE bit of the TxGCON register. The polarity of the Timer1/3 Gate Enable mode is configured using the TxGPOL bit of the TxGCON register.

When Timer1/3 Gate Enable mode is enabled, Timer1/3 will increment on the rising edge of the Timer1/3 clock source. When Timer1/3 Gate Enable mode is disabled, no incrementing will occur and Timer1/3 will hold the current count. See [Figure 13-4](#) for timing details.

TABLE 13-3: TIMER1/3 GATE ENABLE SELECTIONS

| TxCLK | TxGPOL | TxG | Timer1/3 Operation |
|-------|--------|-----|--------------------|
| ↑ | 0 | 0 | Counts |
| ↑ | 0 | 1 | Holds Count |
| ↑ | 1 | 0 | Holds Count |
| ↑ | 1 | 1 | Counts |

13.7.2 TIMER1/3 GATE SOURCE SELECTION

The Timer1/3 gate source can be selected from one of four different sources. Source selection is controlled by the TxGSS bits of the TxGCON register. The polarity for each available source is also selectable. Polarity selection is controlled by the TxGPOL bit of the TxGCON register.

TABLE 13-4: TIMER1/3 GATE SOURCES

| TxGSS | Timer1/3 Gate Source |
|-------|--|
| 00 | Timer1/3 Gate Pin (TxG) |
| 01 | Timer2 Match to PR2 (TMR2 increments to match PR2) |
| 10 | Comparator 1 Output sync_C1OUT (optionally Timer1/3 synchronized output) |
| 11 | Comparator 2 Output sync_C2OUT (optionally Timer1/3 synchronized output) |

13.7.2.1 TxG Pin Gate Operation

The TxG pin is one source for Timer1/3 gate control. It can be used to supply an external source to the Timer1/3 gate circuitry.

13.7.2.2 Timer2 Match Gate Operation

The TMR2 register will increment until it matches the value in the PR2 register. On the very next increment cycle, TMR2 will be reset to 00h. When this Reset occurs, a low-to-high pulse will automatically be generated and internally supplied to the Timer1/3 gate circuitry. See [Section 13.7.2 “Timer1/3 Gate Source Selection”](#) for more information.

13.7.2.3 Comparator C1 Gate Operation

The output resulting from a Comparator 1 operation can be selected as a source for Timer1/3 gate control. The Comparator 1 output (sync_C1OUT) can be synchronized to the Timer1/3 clock or left asynchronous. For more information see [Section 19.8.3 “Synchronizing Comparator Output to Timer1”](#).

13.7.2.4 Comparator C2 Gate Operation

The output resulting from a Comparator 2 operation can be selected as a source for Timer1/3 gate control. The Comparator 2 output (sync_C2OUT) can be synchronized to the Timer1/3 clock or left asynchronous. For more information see [Section 19.8.3 “Synchronizing Comparator Output to Timer1”](#).

13.7.3 TIMER1/3 GATE TOGGLE MODE

When Timer1/3 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1/3 gate signal, as opposed to the duration of a single level pulse.

The Timer1/3 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See [Figure 13-5](#) for timing details.

Timer1/3 Gate Toggle mode is enabled by setting the TxGTM bit of the TxGCON register. When the TxGTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

Note: Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

13.7.4 TIMER1/3 GATE SINGLE-PULSE MODE

When Timer1/3 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1/3 Gate Single-Pulse mode is first enabled by setting the TxGSPM bit in the TxGCON register. Next, the TxGGO/DONE bit in the TxGCON register must be set. The Timer1/3 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the TxGGO/DONE bit will automatically be cleared. No other gate events will be allowed to increment Timer1/3 until the TxGGO/DONE bit is once again set in software.

Clearing the TxGSPM bit of the TxGCON register will also clear the TxGGO/DONE bit. See [Figure 13-6](#) for timing details.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1/3 gate source to be measured. See [Figure 13-7](#) for timing details.

13.7.5 TIMER1/3 GATE VALUE STATUS

When Timer1/3 Gate Value Status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the TxGVAL bit in the TxGCON register. The TxGVAL bit is valid even when the Timer1/3 gate is not enabled (TMRxGE bit is cleared).

13.7.6 TIMER1/3 GATE EVENT INTERRUPT

When Timer1/3 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of TxGVAL occurs, the TMRxGIF flag bit in the PIR3 register will be set. If the TMRxGIE bit in the PIE3 register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer1/3 gate is not enabled (TMRxGE bit is cleared).

For more information on selecting high or low priority status for the Timer1/3 Gate Event Interrupt see [Section 10.0 “Interrupts”](#).

13.8 Timer1/3 Interrupt

The Timer1/3 register pair (TMRxH:TMRxL) increments to FFFFh and rolls over to 0000h. When Timer1/3 rolls over, the Timer1/3 interrupt flag bit of the PIR1/2 register is set. To enable the interrupt on rollover, you must set these bits:

- TMRxON bit of the TxCON register
- TMRxIE bits of the PIE1 or PIE2 registers
- PEIE/GIEL bit of the INTCON register
- GIE/GIEH bit of the INTCON register

The interrupt is cleared by clearing the TMRxIF bit in the Interrupt Service Routine.

For more information on selecting high or low priority status for the Timer1/3 Overflow Interrupt, see [Section 10.0 "Interrupts"](#).

Note: The TMRxH:TMRxL register pair and the TMRxIF bit should be cleared before enabling interrupts.

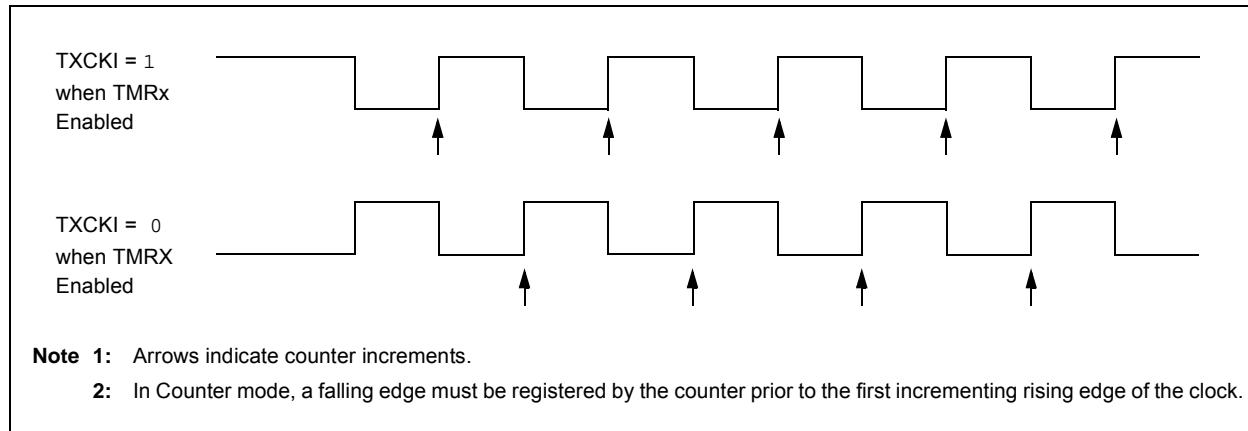
13.9 Timer1/3 Operation During Sleep

Timer1/3 can only operate during Sleep when set up in Asynchronous Counter mode. In this mode, an external crystal or clock source can be used to increment the counter. To set up the timer to wake the device:

- TMRxON bit of the TxCON register must be set
- TMRxIE bit of the PIE1/2 register must be set
- PEIE/GIEL bit of the INTCON register must be set
- TxSYNC bit of the TxCON register must be set
- TMRxCS bits of the TxCON register must be configured
- SOSCEN bit of the TxCON register must be configured

The device will wake-up on an overflow and execute the next instruction. If the GIE/GIEH bit of the INTCON register is set, the device will call the Interrupt Service Routine.

FIGURE 13-3: TIMER1/3 INCREMENTING EDGE



The secondary oscillator will continue to operate in Sleep regardless of the TxSYNC bit setting.

13.10 CCP/CAPTURE/COMPARE TIME BASE

The CCP modules use the TMRxH:TMRxL register pair as the time base when operating in Capture or Compare mode.

In Capture mode, the value in the TMRxH:TMRxL register pair is copied into the CCPRxH:CCPRxL register pair on a configured event.

In Compare mode, an event is triggered when the value CCPRxH:CCPRxL register pair matches the value in the TMRxH:TMRxL register pair. This event can be a Special Event Trigger.

For more information, see [Section 15.0 "Capture/Compare/PWM Modules"](#).

13.11 CCP/CAPTURE/SPECIAL EVENT TRIGGER

When any of the CCP's are configured to trigger a special event, the trigger will clear the TMRxH:TMRxL register pair. This special event does not cause a Timer1/3 interrupt. The CCP module may still be configured to generate a CCP interrupt.

In this mode of operation, the CCPRxH:CCPRxL register pair becomes the period register for Timer1/3.

Timer1/3 should be synchronized and Fosc/4 should be selected as the clock source in order to utilize the Special Event Trigger. Asynchronous operation of Timer1/3 can cause a Special Event Trigger to be missed.

In the event that a write to TMRxH or TMRxL coincides with a Special Event Trigger from the CCP, the write will take precedence.

For more information, see [Section 18.2.8 "Special Event Trigger"](#).

PIC18(L)F2X/45K50

FIGURE 13-4: TIMER1/3 GATE ENABLE MODE

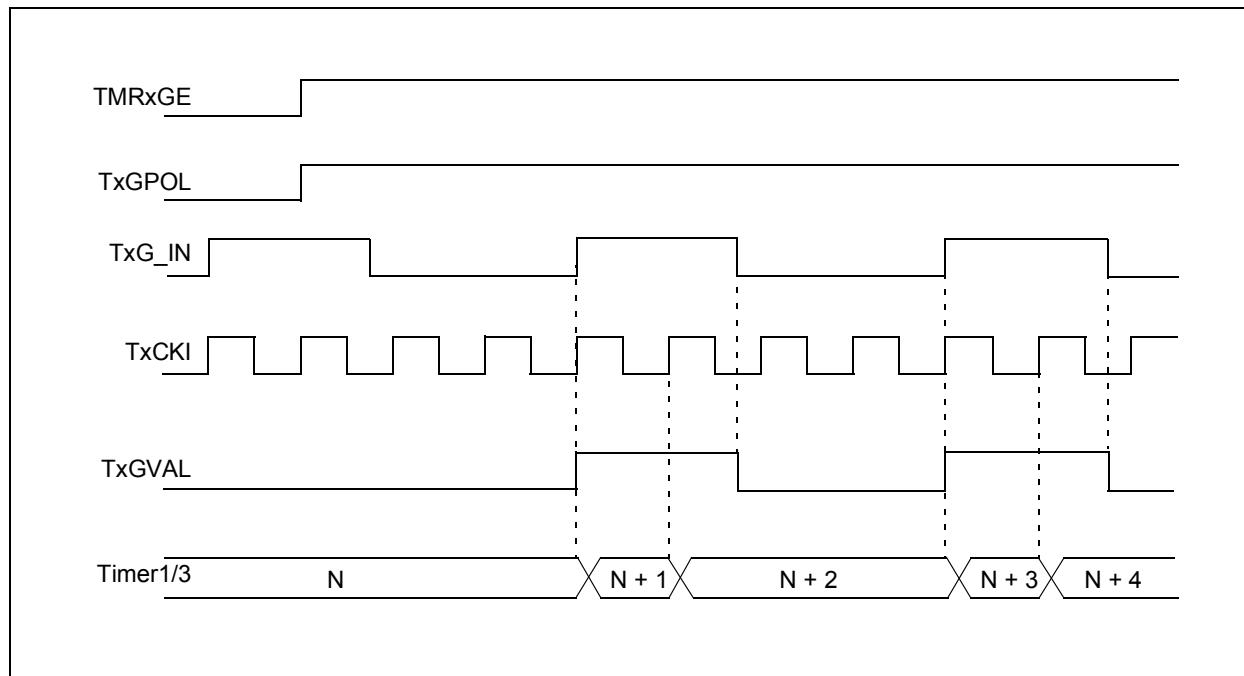


FIGURE 13-5: TIMER1/3 GATE TOGGLE MODE

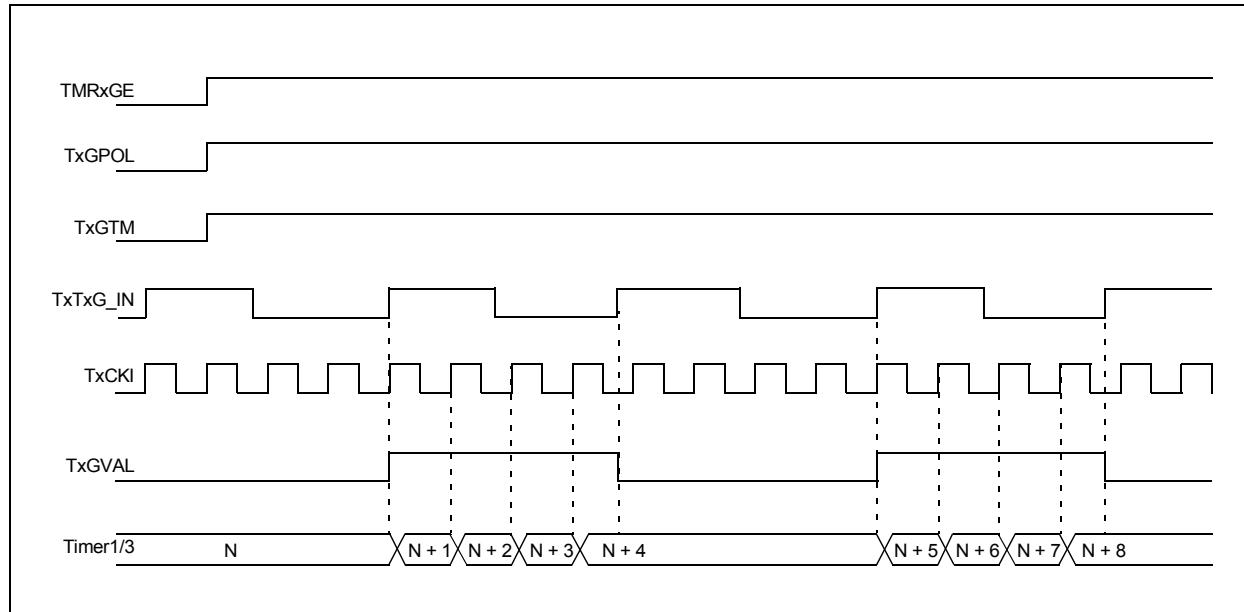
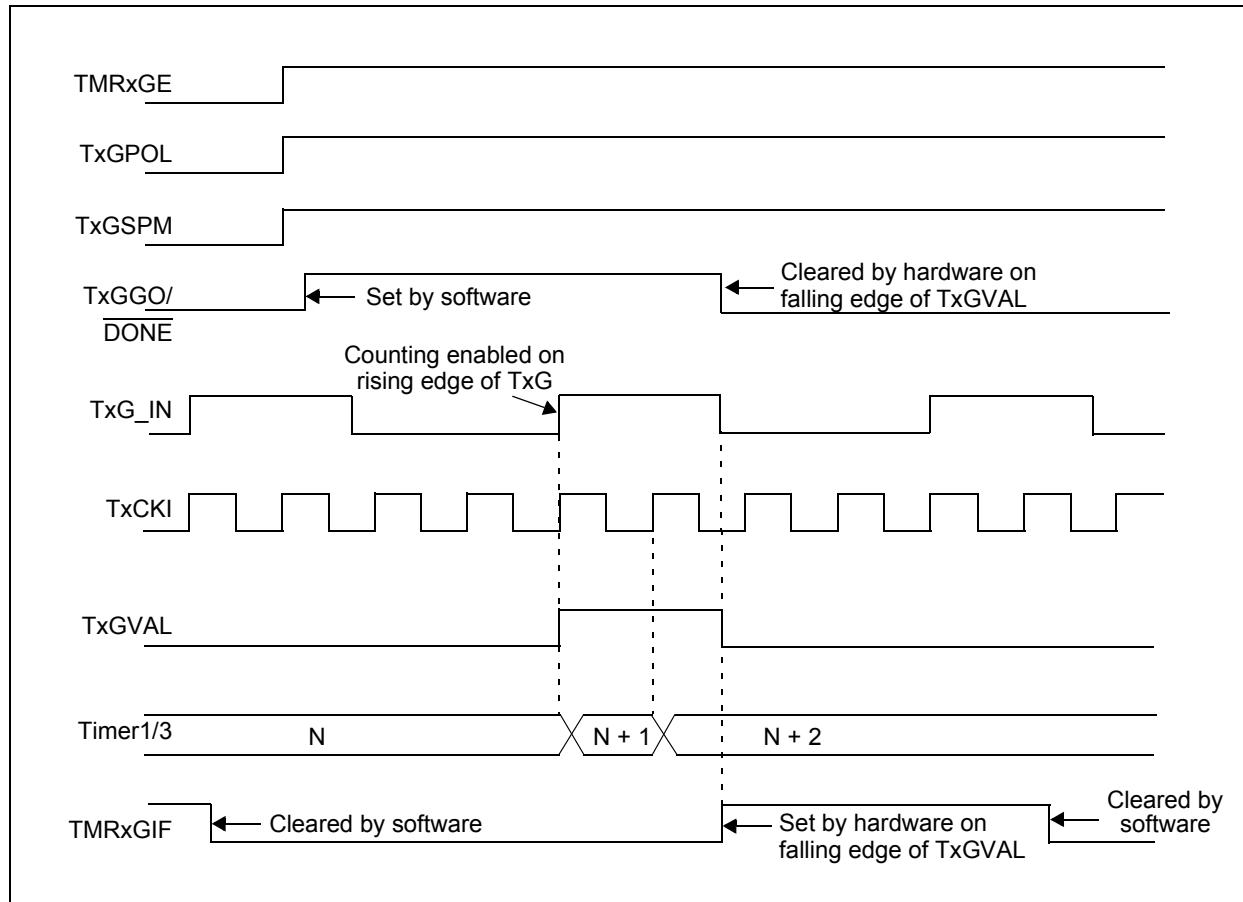
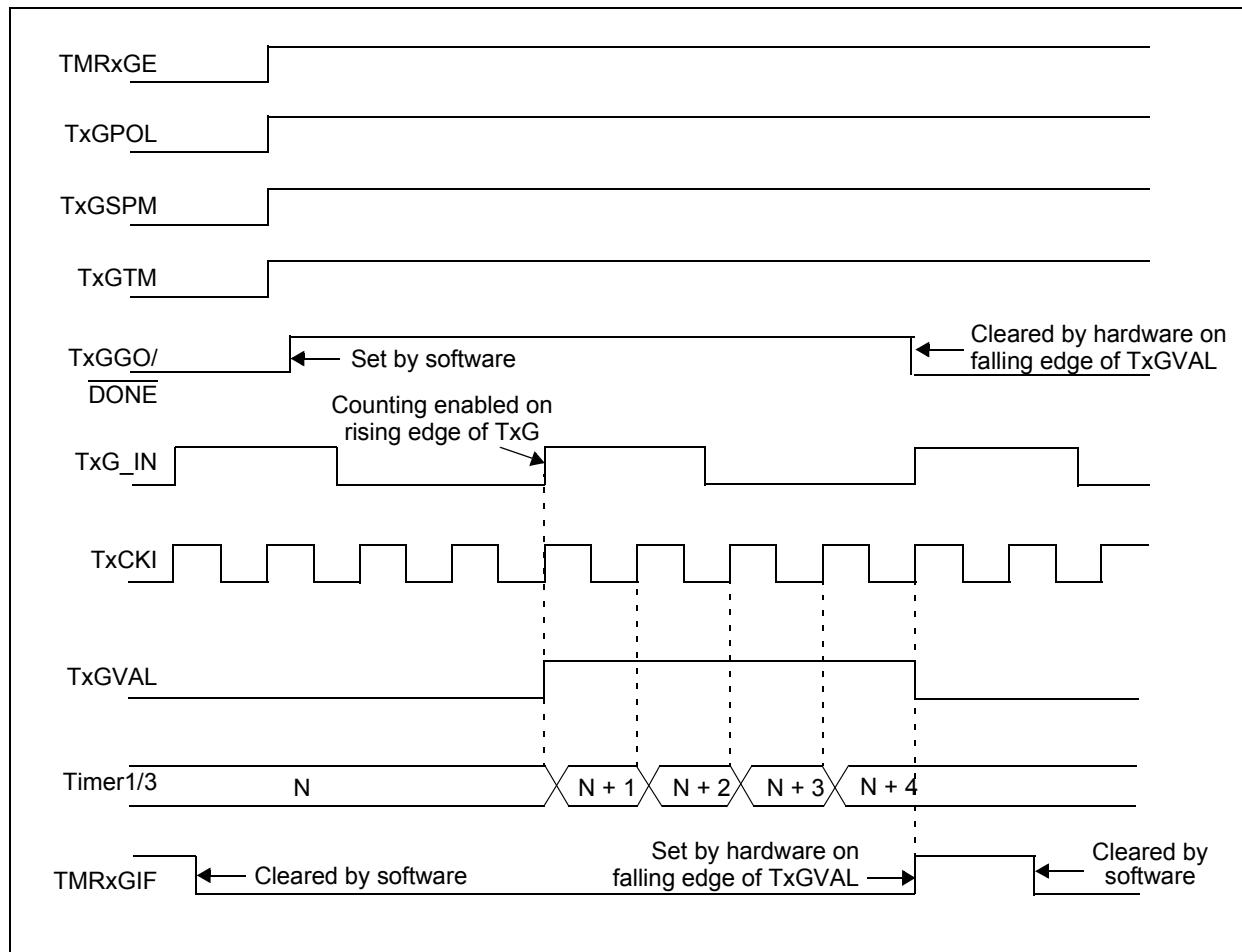


FIGURE 13-6: TIMER1/3 GATE SINGLE-PULSE MODE



PIC18(L)F2X/45K50

FIGURE 13-7: TIMER1/3 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE



13.12 Peripheral Module Disable

When a peripheral module is not used or inactive, the module can be disabled by setting the Module Disable bit in the PMD registers. This will reduce power consumption to an absolute minimum. Setting the PMD bits holds the module in Reset and disconnects the module's clock source. The Module Disable bits for Timer1 (TMR1MD) and Timer3 (TMR3MD) are in the PMD0 Register. See [Section 4.0 “Power-Managed Modes”](#) for more information.

13.13 Register Definitions: Timer1/3 Control

REGISTER 13-1: TxCON: TIMER1/3 CONTROL REGISTER

| R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/0 | R/W-0/u |
|-------------|---------|-------------|---------|---------|---------|---------|---------|
| TMRxCS<1:0> | | TxCKPS<1:0> | | SOSCEN | TxSYNC | RD16 | TMRxON |
| 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 7-6 **TMRxCS<1:0>**: Timer1/3 Clock Source Select bits

11 = Reserved. Do not use.

10 = Timer1/3 clock source is pin or oscillator:

If SOSCEN = 0:

External clock from TxCKI pin (on the rising edge)

If SOSCEN = 1:

Crystal oscillator on SOSCI/SOSCO pins

01 = Timer1/3 clock source is system clock (Fosc)

00 = Timer1/3 clock source is instruction clock (Fosc/4)

bit 5-4 **TxCKPS<1:0>**: Timer1/3 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 **SOSCEN**: Secondary Oscillator Enable Control bit

1 = Dedicated secondary oscillator circuit enabled

0 = Dedicated secondary oscillator circuit disabled

bit 2 **TxSYNC**: Timer1/3 External Clock Input Synchronization Control bit

TMRxCS<1:0> = 1X

1 = Do not synchronize external clock input

0 = Synchronize external clock input with system clock (Fosc)

TMRxCS<1:0> = 0X

This bit is ignored. Timer1/3 uses the internal clock when TMRxCS<1:0> = 0x.

bit 1 **RD16**: 16-Bit Read/Write Mode Enable bit

1 = Enables register read/write of Timer1/3 in one 16-bit operation

0 = Enables register read/write of Timer1/3 in two 8-bit operation

bit 0 **TMRxON**: Timer1/3 On bit

1 = Enables Timer1/3

0 = Stops Timer1/3

Clears Timer1/3 Gate flip-flop

PIC18(L)F2X/45K50

REGISTER 13-2: TxGCON: TIMER1/3 GATE CONTROL REGISTER

| R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W/HC-0/u | R-x/x | R/W-0/u | R/W-0/u |
|---------|---------|---------|---------|------------|--------|------------|---------|
| TMRxGE | TxGPOL | TxGTM | TxGSPM | TxGGO/DONE | TxGVAL | TxGSS<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

HC = Bit is cleared by hardware

- bit 7 **TMRxGE:** Timer1/3 Gate Enable bit
If TMRxON = 0:
This bit is ignored
If TMRxON = 1:
1 = Timer1/3 counting is controlled by the Timer1/3 gate function
0 = Timer1/3 counts regardless of Timer1/3 gate function
- bit 6 **TxGPOL:** Timer1/3 Gate Polarity bit
1 = Timer1/3 gate is active-high (Timer1/3 counts when gate is high)
0 = Timer1/3 gate is active-low (Timer1/3 counts when gate is low)
- bit 5 **TxGTM:** Timer1/3 Gate Toggle Mode bit
1 = Timer1/3 Gate Toggle mode is enabled
0 = Timer1/3 Gate Toggle mode is disabled and toggle flip-flop is cleared
Timer1/3 gate flip-flop toggles on every rising edge.
- bit 4 **TxGSPM:** Timer1/3 Gate Single-Pulse Mode bit
1 = Timer1/3 gate Single-Pulse mode is enabled and is controlling Timer1/3 gate
0 = Timer1/3 gate Single-Pulse mode is disabled
- bit 3 **TxGGO/DONE:** Timer1/3 Gate Single-Pulse Acquisition Status bit
1 = Timer1/3 gate single-pulse acquisition is ready, waiting for an edge
0 = Timer1/3 gate single-pulse acquisition has completed or has not been started
This bit is automatically cleared when TxGSPM is cleared.
- bit 2 **TxGVAL:** Timer1/3 Gate Current State bit
Indicates the current state of the Timer1/3 gate that could be provided to TMRxH:TMRxL.
Unaffected by Timer1/3 Gate Enable (TMRxGE).
- bit 1-0 **TxGSS<1:0>:** Timer1/3 Gate Source Select bits
00 = Timer1/3 Gate pin
01 = Timer2 Match PR2 output
10 = Comparator 1 optionally synchronized output (sync_C1OUT)
11 = Comparator 2 optionally synchronized output (sync_C2OUT)

TABLE 13-5: REGISTERS ASSOCIATED WITH TIMER1/3 AS A TIMER/COUNTER

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page | | |
|--------|------------------------------|-----------|-------------|--------|------------|---------------|------------|---------|------------------|--|--|
| ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 148 | | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 | | |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 | | |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 | | |
| IPR3 | — | — | — | — | CTMUIP | USBIP | TMR3GIP | TMR1GIP | 125 | | |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 | | |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 | | |
| PIE3 | — | — | — | — | CTMUIE | USBIE | TMR3GIE | TMR1GIE | 122 | | |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 | | |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 | | |
| PIR3 | — | — | — | — | CTMUIF | USBIF | TMR3GIF | TMR1GIF | 119 | | |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 | | |
| T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | SOSCEN | <u>T1SYNC</u> | RD16 | TMR1ON | 165 | | |
| T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/DONE | T1GVAL | T1GSS<1:0> | | 166 | | |
| T3CON | TMR3CS<1:0> | | T3CKPS<1:0> | | SOSCEN | <u>T3SYNC</u> | RD16 | TMR3ON | 165 | | |
| T3GCON | TMR3GE | T3GPOL | T3GTM | T3GSPM | T3GGO/DONE | T3GVAL | T3GSS<1:0> | | 166 | | |
| TMRxH | Timer1/3 Register, High Byte | | | | | | | | — | | |
| TMRxL | Timer1/3 Register, Low Byte | | | | | | | | — | | |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 | | |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 149 | | |

TABLE 13-6: CONFIGURATION REGISTERS ASSOCIATED WITH TIMER1/3

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|-------|-------|-------|--------|--------|------------------|
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |

PIC18(L)F2X/45K50

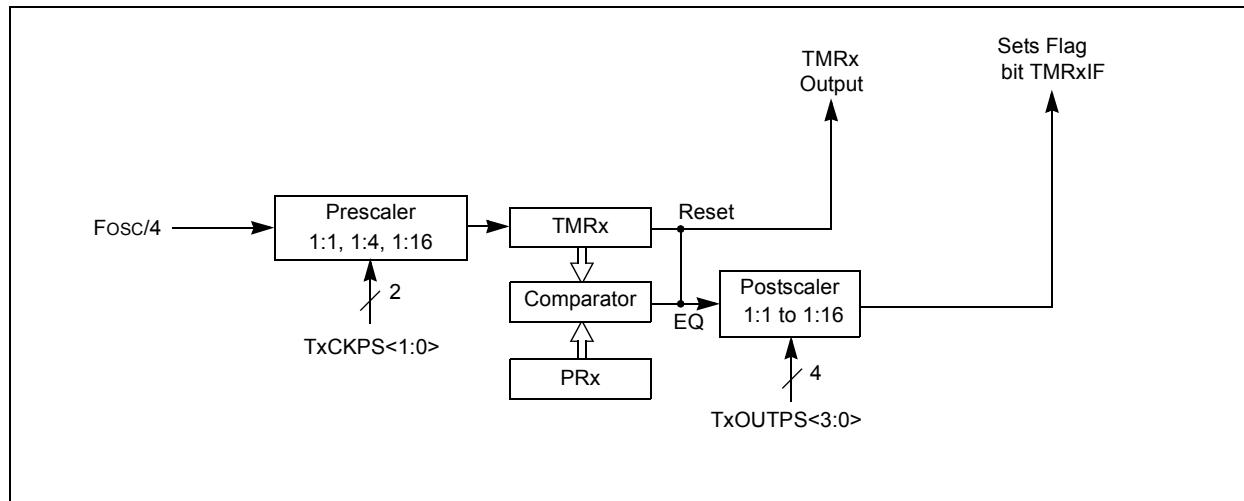
14.0 TIMER2 MODULE

The Timer2 module incorporates the following features:

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

See [Figure 14-1](#) for a block diagram of Timer2.

FIGURE 14-1: TIMER2 BLOCK DIAGRAM



14.1 Timer2 Operation

The clock input to the Timer2 module is the system instruction clock ($F_{osc}/4$).

TMR2 increments from 00h on each clock edge.

A 4-bit counter/prescaler on the clock input allows direct input, divide-by-4 and divide-by-16 prescale options. These options 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 14.2 “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
- Power-on Reset (POR)
- Brown-out Reset (BOR)
- MCLR Reset
- Watchdog Timer (WDT) Reset
- Stack Overflow Reset
- Stack Underflow Reset
- RESET Instruction

Note: TMR2 is not cleared when T2CON is written.

14.2 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 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. Interrupt Priority is selected with the TMR2IP bit in the IPR1 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.

14.3 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 by setting SSPM $<3:0>$ = 0011 in the SSPxCON1 register. Additional information is provided in [Section 16.0 “Master Synchronous Serial Port \(MSSP\) Module”](#).

14.4 Timer2 Operation During Sleep

The Timer2 timers cannot be operated while the processor is in Sleep mode. The contents of the TMR2 and PR2 registers will remain unchanged while the processor is in Sleep mode.

14.5 Peripheral Module Disable

When a peripheral module is not used or inactive, the module can be disabled by setting the Module Disable bit in the PMD registers. This will reduce power consumption to an absolute minimum. Setting the PMD bits holds the module in Reset and disconnects the module's clock source. The Module Disable bit for Timer2 (TMR2MD) is in the PMD0 register. See [Section 4.0 “Power-Managed Modes”](#) for more information.

PIC18(L)F2X/45K50

14.6 Register Definitions: Timer2 Control

REGISTER 14-1: T2CON: TIMER2 CONTROL REGISTER

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|--------------|-------|--------|-------|-------------|-------|
| — | | T2OUTPS<3:0> | | TMR2ON | | T2CKPS<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 **Unimplemented:** Read as '0'

bit 6-3 **T2OUTPS<3:0>:** Timer2 Output Postscaler Select bits

0000 = 1:1 Postscaler

0001 = 1:2 Postscaler

0010 = 1:3 Postscaler

0011 = 1:4 Postscaler

0100 = 1:5 Postscaler

0101 = 1:6 Postscaler

0110 = 1:7 Postscaler

0111 = 1:8 Postscaler

1000 = 1:9 Postscaler

1001 = 1:10 Postscaler

1010 = 1:11 Postscaler

1011 = 1:12 Postscaler

1100 = 1:13 Postscaler

1101 = 1:14 Postscaler

1110 = 1:15 Postscaler

1111 = 1:16 Postscaler

bit 2 **TMR2ON:** Timer2 On bit

1 = Timer2 is on

0 = Timer2 is off

bit 1-0 **T2CKPS<1:0>:** Timer2-type Clock Prescale Select bits

00 = Prescaler is 1

01 = Prescaler is 4

1x = Prescaler is 16

TABLE 14-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|--------|------------------------|--------------|--------|--------|-------|--------|-------------|--------|------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 |
| PR2 | Timer2 Period Register | | | | | | | | — |
| T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | 165 |
| TMR2 | Timer2 Register | | | | | | | | — |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used by Timer2.

15.0 CAPTURE/COMPARE/PWM MODULES

The Capture/Compare/PWM module is a peripheral which allows the user to time and control different events, and to generate Pulse-Width Modulation (PWM) signals. In Capture mode, the peripheral allows the timing of the duration of an event. The Compare mode allows the user to trigger an external event when a predetermined amount of time has expired. The PWM mode can generate Pulse-Width Modulated signals of varying frequency and duty cycle.

This family of devices contains one Enhanced Capture/Compare/PWM module (ECCP1) and one standard Capture/Compare/PWM module (CCP2).

The Capture and Compare functions are identical for the CCP/ECCP modules. The difference between CCP and ECCP modules are in the Pulse-Width Modulation (PWM) function. In CCP modules, the standard PWM function is identical. In ECCP modules, the Enhanced PWM function has either full-bridge or half-bridge PWM output. Full-bridge ECCP modules have four available I/O pins while half-bridge ECCP modules only have two available I/O pins. ECCP PWM modules are backward compatible with CCP PWM modules and can be configured as standard PWM modules.

Note 1: In devices with more than one CCP module, it is very important to pay close attention to the register names used. A number placed after the module acronym is used to distinguish between separate modules. For example, the CCP1CON and CCP2CON control the same operational aspects of two completely different CCP modules.

2: Throughout this section, generic references to a CCP module in any of its operating modes may be interpreted as being equally applicable to ECCP1 and CCP2. Register names, module signals, I/O pins and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module, when required.

15.1 Capture Mode

The Capture mode function described in this section is identical for all CCP and ECCP modules available on this device family.

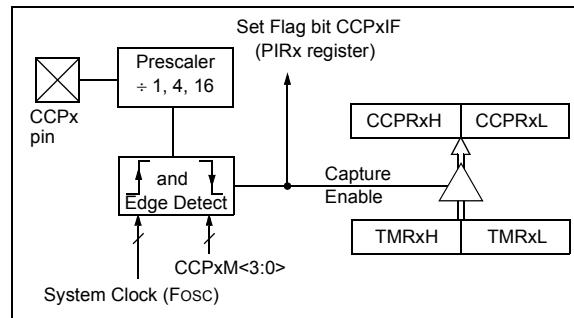
Capture mode makes use of the 16-bit Timer resources, Timer1 and Timer3. The timer resources for each CCP capture function are independent and are selected using the CCPTMRS register. When an event occurs on the CCPx pin, the 16-bit CCPRxH:CCPRxL register pair captures and stores the 16-bit value of the TMRxH:TMRxL register pair, respectively. An event is defined as one of the following and is configured by the CCPxM<3:0> bits of the CCPxCON register:

- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

When a capture is made, the corresponding Interrupt Request Flag bit CCPxIF of the PIR1 and PIR2 register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRxH:CCPRxL register pair is read, the old captured value is overwritten by the new captured value.

Figure 15-1 shows a simplified diagram of the Capture operation.

FIGURE 15-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



15.1.1 CCP PIN CONFIGURATION

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

Some CCPx outputs are multiplexed on a couple of pins. [Table 15-1](#) shows the CCP output pin multiplexing. Selection of the output pin is determined by the CCPxMX bits in Configuration register 3H (CONFIG3H). Refer to [Register 26-5](#) for more details.

Note: If the CCPx pin is configured as an output, a write to the port can cause a capture condition.

TABLE 15-1: CCP PIN MULTIPLEXING

| CCP OUTPUT | CONFIG 3H Control Bit | Bit Value | I/O pin |
|------------|-----------------------|-----------|---------|
| CCP2 | CCP2MX | 0 | RB3 |
| | | 1(*) | RC1 |

Legend: * = Default

15.1.2 TIMER1 MODE RESOURCE

The 16-bit Timer resource must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

See [Section 13.0 “Timer1/3 Module with Gate Control”](#) for more information on configuring the 16-bit Timers.

15.1.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE interrupt enable bit of the PIE register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the PIR register following any change in Operating mode.

Note: Clocking the 16-bit Timer resource from the system clock (Fosc) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, the Timer resource must be clocked from the instruction clock (Fosc/4) or from an external clock source.

PIC18(L)F2X/45K50

15.1.4 CCP PRESCALER

There are four prescaler settings specified by the CCPxM<3:0> bits of the CCPxCON register. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another does not clear the prescaler and may generate a false interrupt. To avoid this unexpected operation, turn the module off by clearing the CCPxCON register before changing the prescaler. [Example 15-1](#) demonstrates the code to perform this function.

EXAMPLE 15-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
#define NEW_CAPT_PS 0x06      //Capture
                           // Prescale 4th
...
CCPxCON = 0;              // rising edge
                           // Turn the CCP
                           // Module Off
CCPxCON = NEW_CAPT_PS;    // Turn CCP module
                           // on with new
                           // prescale value
```

15.1.5 CAPTURE DURING SLEEP

Capture mode requires a 16-bit TimerX module for use as a time base. There are four options for driving the 16-bit TimerX module in Capture mode. It can be driven by the system clock (Fosc), the instruction clock (Fosc/4), or by the external clock sources, the Secondary Oscillator (Sosc), or the TxCKI clock input. When the 16-bit TimerX resource is clocked by Fosc or Fosc/4, TimerX will not increment during Sleep. When the device wakes from Sleep, TimerX will continue from its previous state. Capture mode will operate during Sleep when the 16-bit TimerX resource is clocked by one of the external clock sources (Sosc or the TxCKI pin).

TABLE 15-2: REGISTERS ASSOCIATED WITH CAPTURE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page | | |
|---------|---|-----------|-------------|--------|------------|--------|------------|--------|------------------|-----|--|
| CCP1CON | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | | | 197 | |
| CCP2CON | — | — | DC2B<1:0> | | CCP2M<3:0> | | | | | 197 | |
| CCPR1H | Capture/Compare/PWM Register 1, High Byte (MSB) | | | | | | | | — | | |
| CCPR1L | Capture/Compare/PWM Register 1, Low Byte (LSB) | | | | | | | | — | | |
| CCPR2H | Capture/Compare/PWM Register 2, High Byte (MSB) | | | | | | | | — | | |
| CCPR2L | Capture/Compare/PWM Register 2, Low Byte (LSB) | | | | | | | | — | | |
| CCPTMRS | — | — | — | — | C2TSEL | — | — | C1TSEL | 200 | | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 | | |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 | | |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 | | |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 | | |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 | | |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 | | |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 | | |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 | | |
| PMD1 | — | MSSPMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | 62 | | |
| T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | SOSCEN | T1SYNC | RD16 | TMR1ON | 165 | | |
| T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/DONE | T1GVAL | T1GSS<1:0> | | 166 | | |
| T3CON | TMR3CS<1:0> | | T3CKPS<1:0> | | SOSCEN | T3SYNC | RD16 | TMR3ON | 165 | | |
| T3GCON | TMR3GE | T3GPOL | T3GTM | T3GSPM | T3GGO/DONE | T3GVAL | T3GSS<1:0> | | 166 | | |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | — | | |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | — | | |
| TMR3H | Holding Register for the Most Significant Byte of the 16-bit TMR3 Register | | | | | | | | — | | |
| TMR3L | Holding Register for the Least Significant Byte of the 16-bit TMR3 Register | | | | | | | | — | | |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 | | |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 149 | | |

Legend: — = Unimplemented location, read as '0'. Shaded bits are not used by Capture mode.

TABLE 15-3: CONFIGURATION REGISTERS ASSOCIATED WITH CAPTURE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|-------|-------|-------|--------|--------|------------------|
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |

Legend: — = Unimplemented location, read as '0'. Shaded bits are not used by Capture mode.

15.2 Compare Mode

The Compare mode function described in this section is identical for all CCP and ECCP modules available on this device family.

Compare mode makes use of the 16-bit Timer resources, Timer1 and Timer3. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared against the 16-bit value of the TMRxH:TMRxL register pair. When a match occurs, one of the following events can occur:

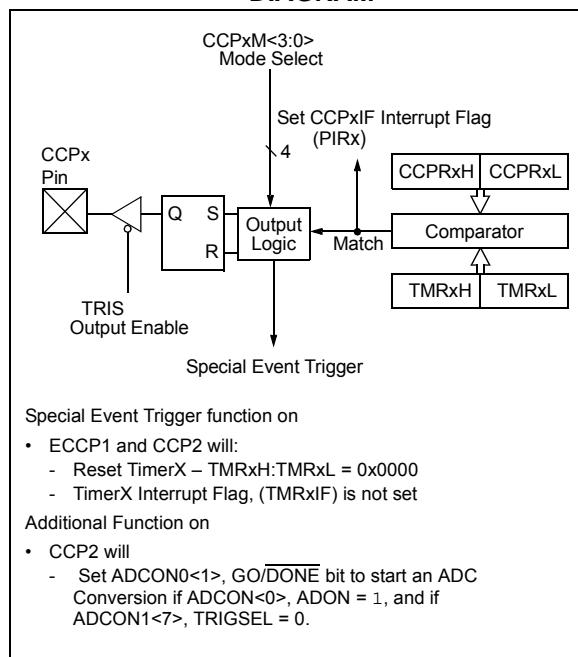
- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Generate a Special Event Trigger
- Generate a Software Interrupt

The action on the pin is based on the value of the CCPxM<3:0> control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set.

All Compare modes can generate an interrupt.

[Figure 15-2](#) shows a simplified diagram of the Compare operation.

FIGURE 15-2: COMPARE MODE OPERATION BLOCK DIAGRAM



15.2.1 CCP PIN CONFIGURATION

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

Some CCPx outputs are multiplexed on a couple of pins. [Table 15-1](#) shows the CCP output pin multiplexing. Selection of the output pin is determined by the CCPxMX bits in Configuration register 3H (CONFIG3H). Refer to [Register 26-5](#) for more details.

Note: Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

15.2.2 TimerX MODE RESOURCE

In Compare mode, 16-bit TimerX resource must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See [Section 13.0 “Timer1/3 Module with Gate Control”](#) for more information on configuring the 16-bit TimerX resources.

Note: Clocking TimerX from the system clock (Fosc) should not be used in Compare mode. In order for Compare mode to recognize the trigger event on the CCPx pin, TimerX must be clocked from the instruction clock (Fosc/4) or from an external clock source.

15.2.3 SOFTWARE INTERRUPT MODE

When Generate Software Interrupt mode is chosen (CCPxM<3:0> = 1010), the CCPx module does not assert control of the CCPx pin (see the CCPxCON register).

15.2.4 SPECIAL EVENT TRIGGER

When Special Event Trigger mode is selected ($\text{CCPxM}\langle 3:0 \rangle = 1011$), and a match of the TMRxH:TMRxL and the CCPRxH:CCPRxL registers occurs, all CCPx and ECCPx modules will immediately:

- Set the CCP interrupt flag bit – CCPxIF
- CCP2 will start an ADC conversion, if the ADC is enabled and TRIGSEL is configured for CCP2.

On the next TimerX rising clock edge:

- A Reset of TimerX register pair occurs –
 $\text{TMRxH:TMRxL} = 0x0000$,

This Special Event Trigger mode does not:

- Assert control over the CCPx or ECCPx pins.
- Set the TMRxIF interrupt bit when the TMRxH:TMRxL register pair is reset. (TMRxIF gets set on a TimerX overflow.)

If the value of the CCPRxH:CCPRxL registers are modified when a match occurs, the user should be aware that the automatic reset of TimerX occurs on the next rising edge of the clock. Therefore, modifying the CCPRxH:CCPRxL registers before this reset occurs will allow the TimerX to continue without being reset, inadvertently resulting in the next event being advanced or delayed.

The Special Event Trigger mode allows the CCPRxH:CCPRxL register pair to effectively provide a 16-bit programmable period register for TimerX.

15.2.5 COMPARE DURING SLEEP

The Compare mode is dependent upon the system clock (Fosc) for proper operation. Since Fosc is shut down during Sleep mode, the Compare mode will not function properly during Sleep.

PIC18(L)F2X/45K50

TABLE 15-4: REGISTERS ASSOCIATED WITH COMPARE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page | | |
|---------|---|-----------|-------------|--------|------------|---------------|------------|--------|------------------|-----|--|
| CCP1CON | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | | | 197 | |
| CCP2CON | — | — | DC2B<1:0> | | CCP2M<3:0> | | | | | 197 | |
| CCPR1H | Capture/Compare/PWM Register 1, High Byte (MSB) | | | | | | | | — | | |
| CCPR1L | Capture/Compare/PWM Register 1, Low Byte (LSB) | | | | | | | | — | | |
| CCPR2H | Capture/Compare/PWM Register 2, High Byte (MSB) | | | | | | | | — | | |
| CCPR2L | Capture/Compare/PWM Register 2, Low Byte (LSB) | | | | | | | | — | | |
| CCPTMRS | — | — | — | — | C2TSEL | — | — | C1TSEL | 200 | | |
| ADCON1 | TRIGSEL | — | — | — | PVCFG<1:0> | | NVCFG<1:0> | | 295 | | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 | | |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 | | |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 | | |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 | | |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 | | |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 | | |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 | | |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 | | |
| PMD1 | — | MSSPMMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | 62 | | |
| T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | SOSCEN | <u>T1SYNC</u> | RD16 | TMR1ON | 165 | | |
| T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/DONE | T1GVAL | T1GSS<1:0> | | 166 | | |
| T3CON | TMR3CS<1:0> | | T3CKPS<1:0> | | SOSCEN | <u>T3SYNC</u> | RD16 | TMR3ON | 165 | | |
| T3GCON | TMR3GE | T3GPOL | T3GTM | T3GSPM | T3GGO/DONE | T3GVAL | T3GSS<1:0> | | 166 | | |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | — | | |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | — | | |
| TMR3H | Holding Register for the Most Significant Byte of the 16-bit TMR3 Register | | | | | | | | — | | |
| TMR3L | Holding Register for the Least Significant Byte of the 16-bit TMR3 Register | | | | | | | | — | | |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 | | |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 149 | | |

Legend: — = Unimplemented location, read as '0'. Shaded bits are not used by Capture mode.

TABLE 15-5: CONFIGURATION REGISTERS ASSOCIATED WITH CAPTURE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|-------|-------|-------|--------|--------|------------------|
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |

Legend: — = Unimplemented location, read as '0'. Shaded bits are not used by Capture mode.

15.3 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides power to a load by switching quickly between fully on and fully off states. The PWM signal resembles a square wave where the high portion of the signal is considered the on state and the low portion of the signal is considered the off state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of on and off time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the on time to the off time and is expressed in percentages, where 0% is fully off and 100% is fully on. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied.

Figure 15-3 shows a typical waveform of the PWM signal.

15.3.1 STANDARD PWM OPERATION

The standard PWM function described in this section is available and identical for CCP and ECCP modules.

The standard PWM mode generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to 10 bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- PR2 register
- T2CON register
- CCPRxL registers
- CCPxCON registers

Figure 15-4 shows a simplified block diagram of PWM operation.

Note 1: The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

2: Clearing the CCPxCON register will relinquish control of the CCPx pin.

FIGURE 15-3: CCP PWM OUTPUT SIGNAL

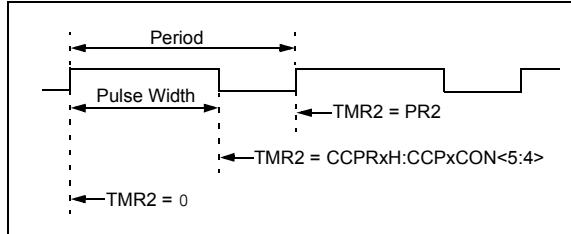
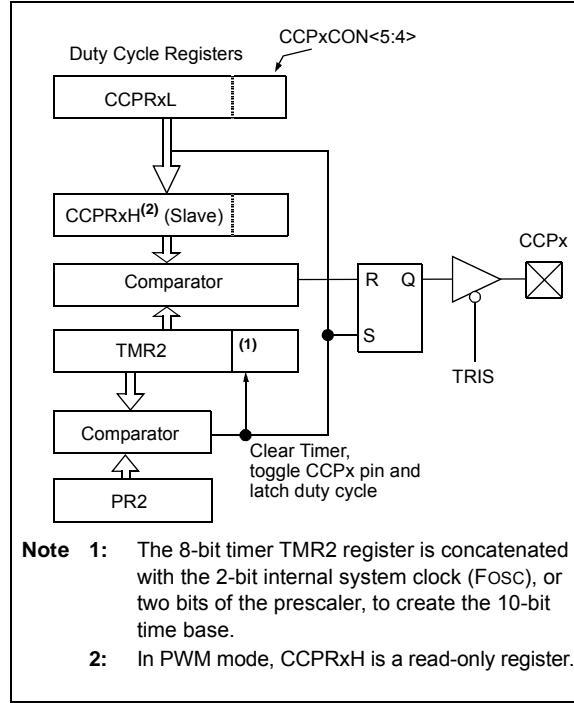


FIGURE 15-4: SIMPLIFIED PWM BLOCK DIAGRAM



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

2: In PWM mode, CCPRxH is a read-only register.

15.3.2 SETUP FOR PWM OPERATION

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

1. Disable the CCPx pin output driver by setting the associated TRIS bit.
2. Load the PR2 register for Timer2 with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRxL register and the DCxB<1:0> bits of the CCPxCON register, with the PWM duty cycle value.

5. Configure and start the 8-bit Timer2:
 - Clear the TMR2IF interrupt flag bit of the PIR1 register. See [Note 1](#) below.
 - Configure the T2CKPS bits of the T2CON register with the Timer prescale value.
 - Enable the Timer by setting the TMR2ON bit of the T2CON register.
6. Enable PWM output pin:
 - Wait until the Timer overflows and the TMR2IF bit of the PIR1 register is set. See [Note 1](#) below.
 - Enable the CCPx pin output driver by clearing the associated TRIS bit.

Note 1: In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 5 may be ignored.

15.3.3 PWM PERIOD

The PWM period is specified by the PR2 register of 8-bit Timer2. The PWM period can be calculated using the formula of [Equation 15-1](#).

EQUATION 15-1: PWM PERIOD

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

Note 1: $TOSC = 1/Fosc$

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is latched from CCPRxL into CCPRxH.

Note: The Timer postscaler (see [Section 14.0 “Timer2 Module”](#)) is not used in the determination of the PWM frequency.

15.3.4 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to multiple registers: CCPRxL register and DCxB<1:0> bits of the CCPxCON register. The CCPRxL contains the eight MSbs and the DCxB<1:0> bits of the CCPxCON register contain the two LSbs. CCPRxL and DCxB<1:0> bits of the CCPxCON register can be written to at any time. The duty cycle value is not latched into CCPRxH until after the period completes (i.e., a match between PR2 and TMR2 registers occurs). While using the PWM, the CCPRxH register is read-only.

[Equation 15-2](#) is used to calculate the PWM pulse width.

[Equation 15-3](#) is used to calculate the PWM duty cycle ratio.

EQUATION 15-2: PULSE WIDTH

$$\text{Pulse Width} = (CCPRxL:CCPxCON<5:4>) \cdot TOSC \cdot (TMR2 \text{ Prescale Value})$$

EQUATION 15-3: DUTY CYCLE RATIO

$$\text{Duty Cycle Ratio} = \frac{(CCPRxL:CCPxCON<5:4>)}{4(PR2 + 1)}$$

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

The 8-bit timer TMR2 register is concatenated with either the 2-bit internal system clock (Fosc), or two bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

When the 10-bit time base matches the CCPRxH and 2-bit latch, then the CCPx pin is cleared (see [Figure 15-4](#)).

15.3.5 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is 10 bits when PR2 is 255. The resolution is a function of the PR2 register value as shown by [Equation 15-4](#).

EQUATION 15-4: PWM RESOLUTION

$$\text{Resolution} = \frac{\log[4(PR2 + 1)]}{\log(2)} \text{ bits}$$

Note: If the pulse width value is greater than the period the assigned PWM pin(s) will remain unchanged.

TABLE 15-6: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 32 MHz)

| PWM Frequency | 1.95 kHz | 7.81 kHz | 31.25 kHz | 125 kHz | 250 kHz | 333.3 kHz |
|---------------------------|----------|----------|-----------|---------|---------|-----------|
| Timer Prescale (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.6 |

TABLE 15-7: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 20 MHz)

| PWM Frequency | 1.22 kHz | 4.88 kHz | 19.53 kHz | 78.12 kHz | 156.3 kHz | 208.3 kHz |
|---------------------------|----------|----------|-----------|-----------|-----------|-----------|
| Timer Prescale (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.6 |

TABLE 15-8: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 8 MHz)

| PWM Frequency | 1.22 kHz | 4.90 kHz | 19.61 kHz | 76.92 kHz | 153.85 kHz | 200.0 kHz |
|---------------------------|----------|----------|-----------|-----------|------------|-----------|
| Timer Prescale (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | 0x65 | 0x65 | 0x65 | 0x19 | 0x0C | 0x09 |
| Maximum Resolution (bits) | 8 | 8 | 8 | 6 | 5 | 5 |

15.3.6 OPERATION IN SLEEP MODE

In Sleep mode, the TMR2 register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, TMR2 will continue from its previous state.

15.3.7 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency. Any changes in the system clock frequency will result in changes to the PWM frequency. See [Section 3.0 “Oscillator Module \(With Fail-Safe Clock Monitor\)”](#) for additional details.

15.3.8 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

PIC18(L)F2X/45K50

TABLE 15-9: REGISTERS ASSOCIATED WITH STANDARD PWM

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page | |
|---------|------------------------|--------------|-----------|--------|------------|--------|-------------|--------|------------------|-----|
| CCP1CON | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | | | 197 |
| CCP2CON | — | — | DC2B<1:0> | | CCP2M<3:0> | | | | | 197 |
| CCPTMRS | — | — | — | — | C2TSEL | — | — | C1TSEL | 200 | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 | |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 | |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 | |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 | |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 | |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 | |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 | |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 | |
| PMD1 | — | MSSPMMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | 62 | |
| PR2 | Timer2 Period Register | | | | | | | | — | |
| T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | 170 | |
| TMR2 | Timer2 Period Register | | | | | | | | — | |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 | |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 149 | |

Legend: — = Unimplemented location, read as '0'. Shaded bits are not used by Capture mode.

TABLE 15-10: CONFIGURATION REGISTERS ASSOCIATED WITH CAPTURE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|-------|-------|-------|--------|--------|------------------|
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |

Legend: — = Unimplemented location, read as '0'. Shaded bits are not used by Capture mode.

15.4 PWM (Enhanced Mode)

The enhanced PWM function described in this section is available for CCP module ECCP1.

The enhanced PWM mode generates a Pulse-Width Modulation (PWM) signal on up to four different output pins with up to 10 bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- PR2 register
 - T2CON register
 - CCPRxL registers
 - CCPxCON registers

The ECCP modules have the following additional PWM registers which control Auto-shutdown, Auto-restart, Dead-band Delay and PWM Steering modes:

- ECCPxAS registers
 - PSTRxCON registers
 - PWMxCON registers

The enhanced PWM module can generate the following five PWM Output modes:

- Single PWM
 - Half-Bridge PWM
 - Full-Bridge PWM, Forward mode
 - Full-Bridge PWM, Reverse mode
 - Single PWM with PWM Steering mode

To select an Enhanced PWM Output mode, the PxM<1:0> bits of the CCPxCON register must be configured appropriately.

The PWM outputs are multiplexed with I/O pins and are designated Px_A, Px_B, Px_C and Px_D. The polarity of the PWM pins is configurable and is selected by setting the CCPxM bits in the CCPxCON register appropriately.

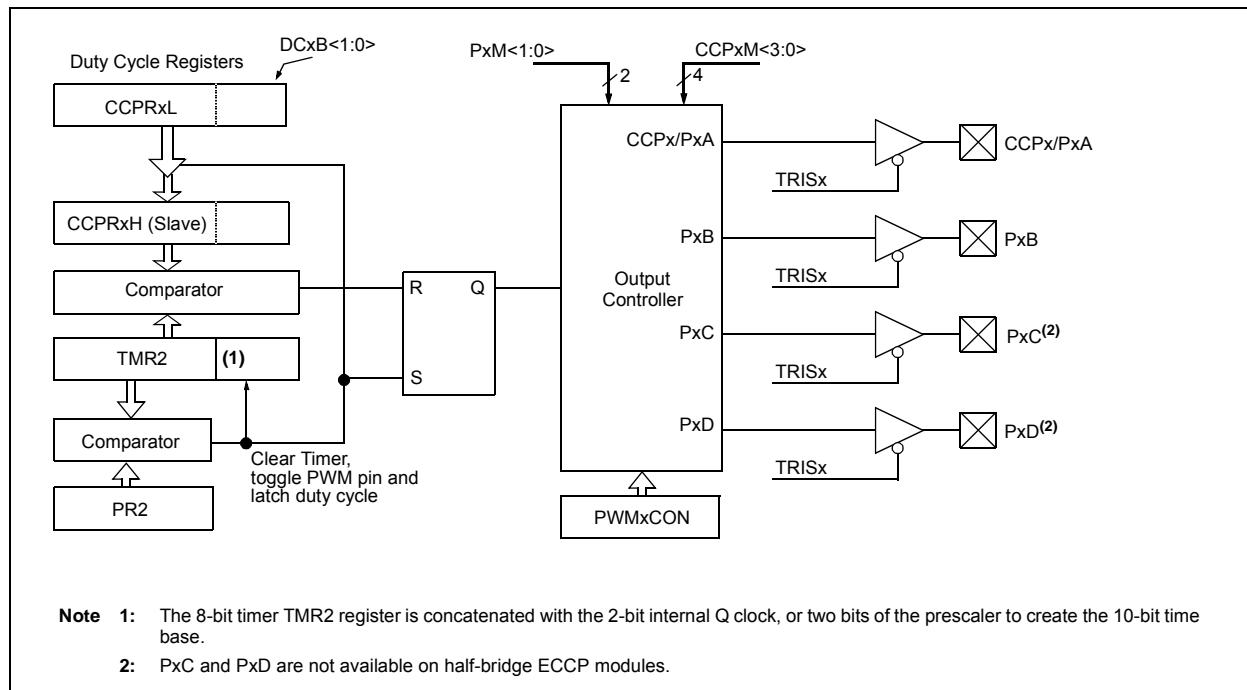
Figure 15-5 shows an example of a simplified block diagram of the Enhanced PWM module.

Table 15-11 shows the pin assignments for various Enhanced PWM modes.

- Note 1:** The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

 - 2: Clearing the CCPxCON register will relinquish control of the CCPx pin.
 - 3: Any pin not used in the enhanced PWM mode is available for alternate pin functions, if applicable.
 - 4: 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 15-5: EXAMPLE SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODE



PIC18(L)F2X/45K50

TABLE 15-11: EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES

| ECCP Mode | PxM<1:0> | CCPx/PxA | PxB | PxC | PxD |
|----------------------|----------|--------------------|--------------------|--------------------|--------------------|
| 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: PWM Steering enables outputs in Single mode.

FIGURE 15-6: EXAMPLE PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)

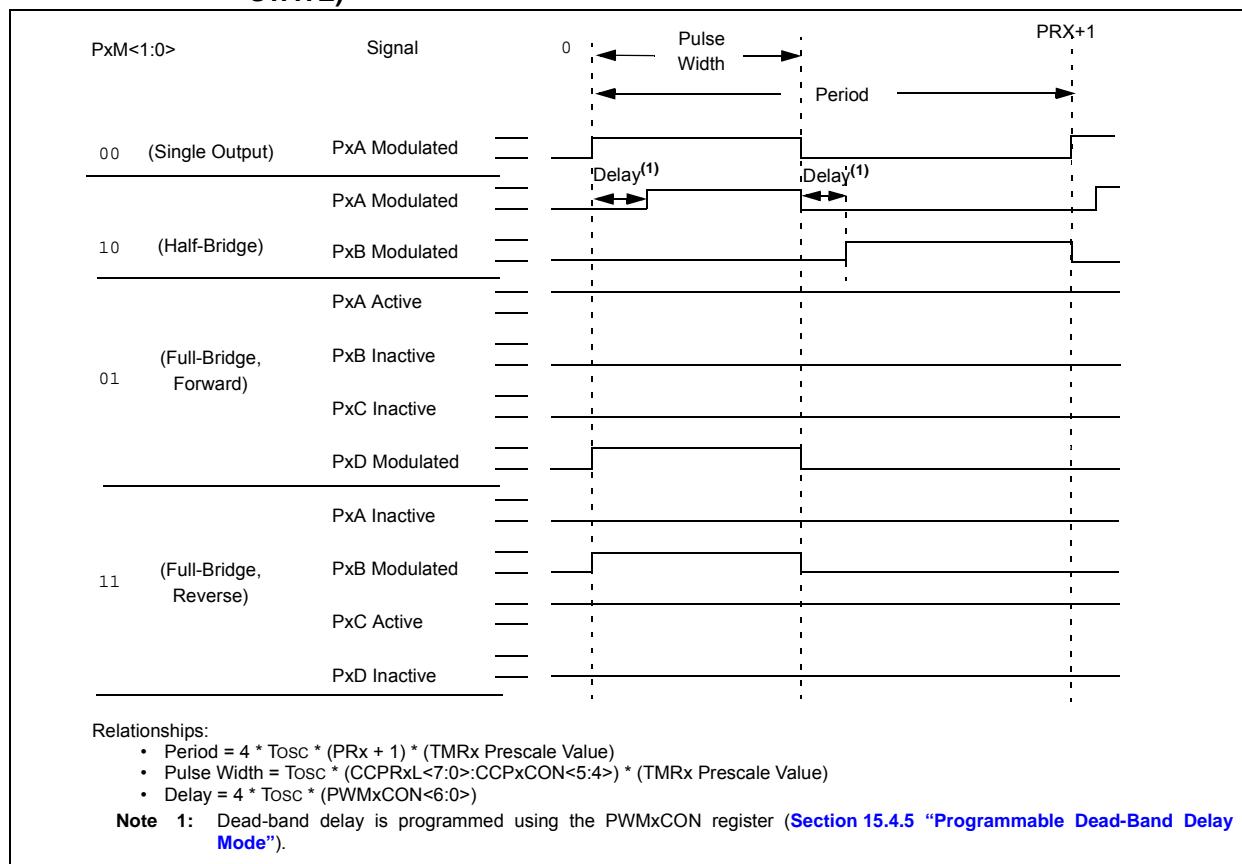
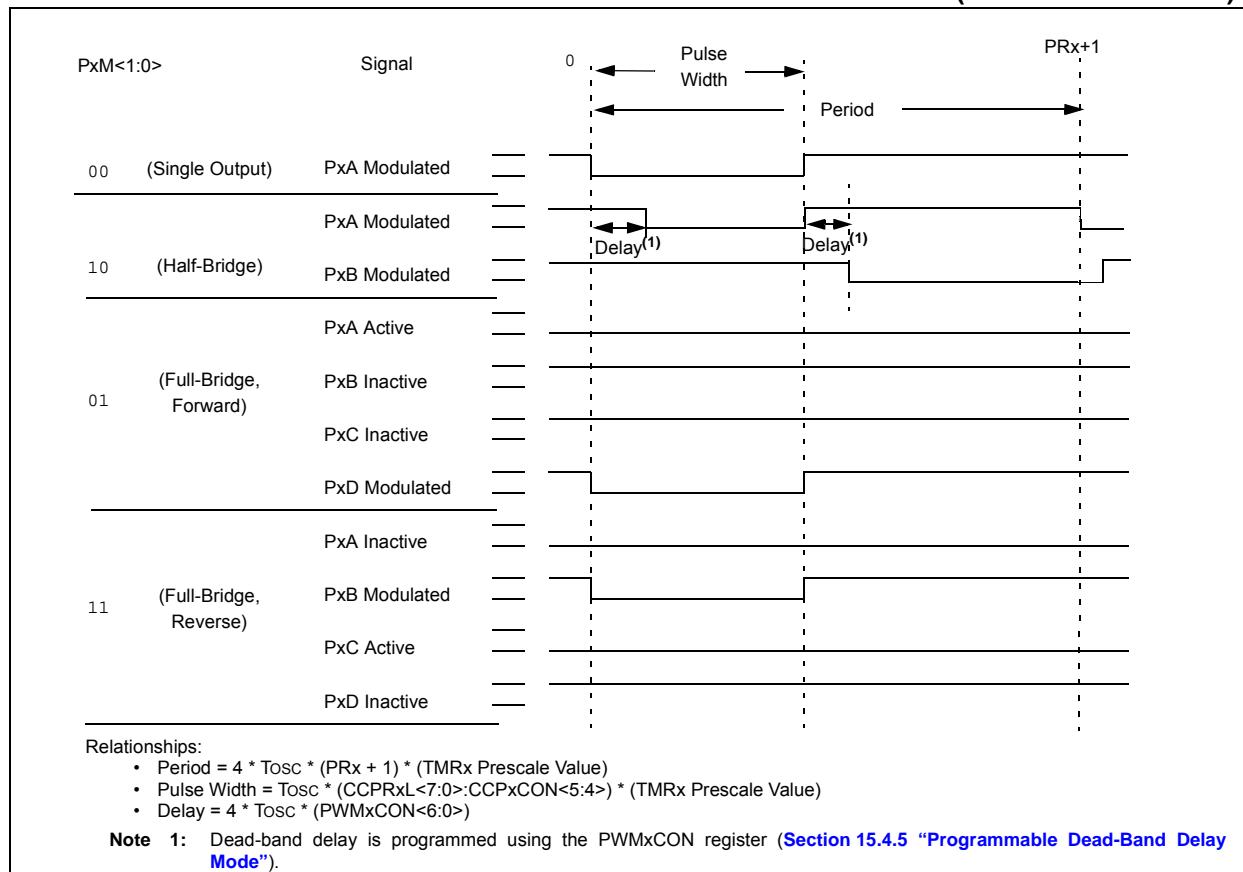


FIGURE 15-7: EXAMPLE ENHANCED PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)



15.4.1 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 CCPx/PxA pin, while the complementary PWM output signal is output on the PxB pin (see Figure 15-9). This mode can be used for half-bridge applications, as shown in Figure 15-9, 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 PWMxCON 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 15.4.5 “Programmable Dead-Band Delay Mode” for more details of the dead-band delay operations.

Since the PxA and PxB outputs are multiplexed with the PORT data latches, the associated TRIS bits must be cleared to configure PxA and PxB as outputs.

FIGURE 15-8: EXAMPLE OF HALF-BRIDGE PWM OUTPUT

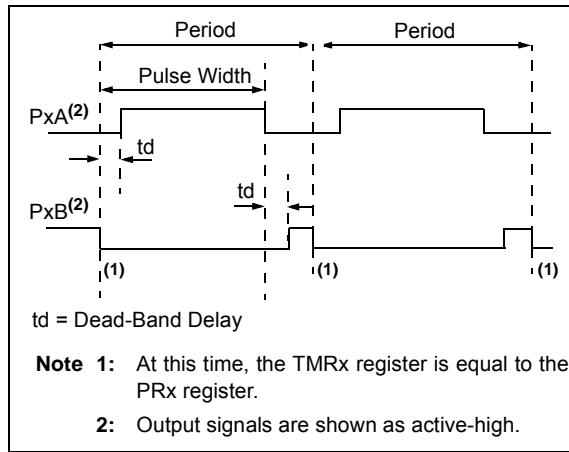
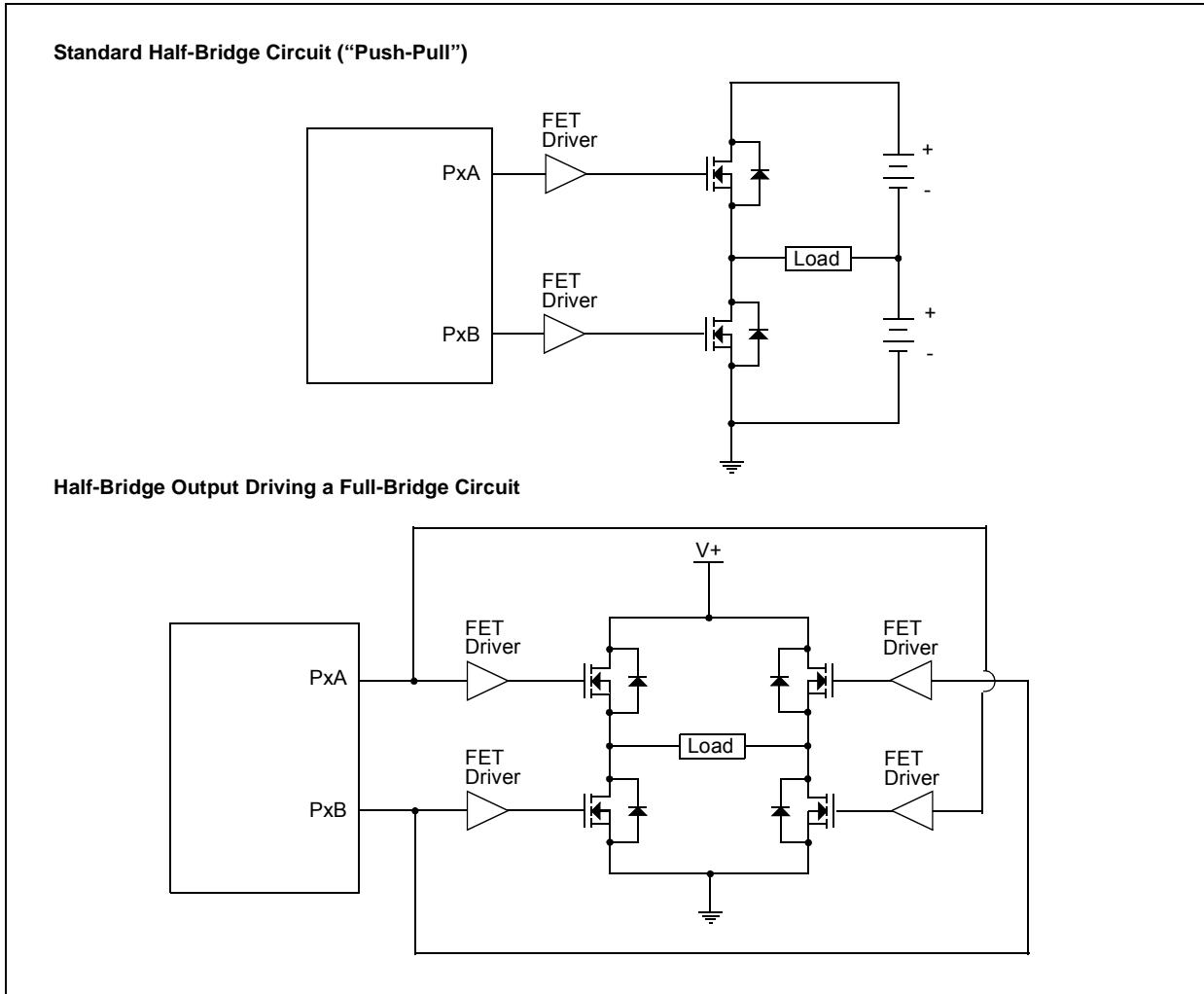


FIGURE 15-9: EXAMPLE OF HALF-BRIDGE APPLICATIONS



15.4.2 FULL-BRIDGE MODE

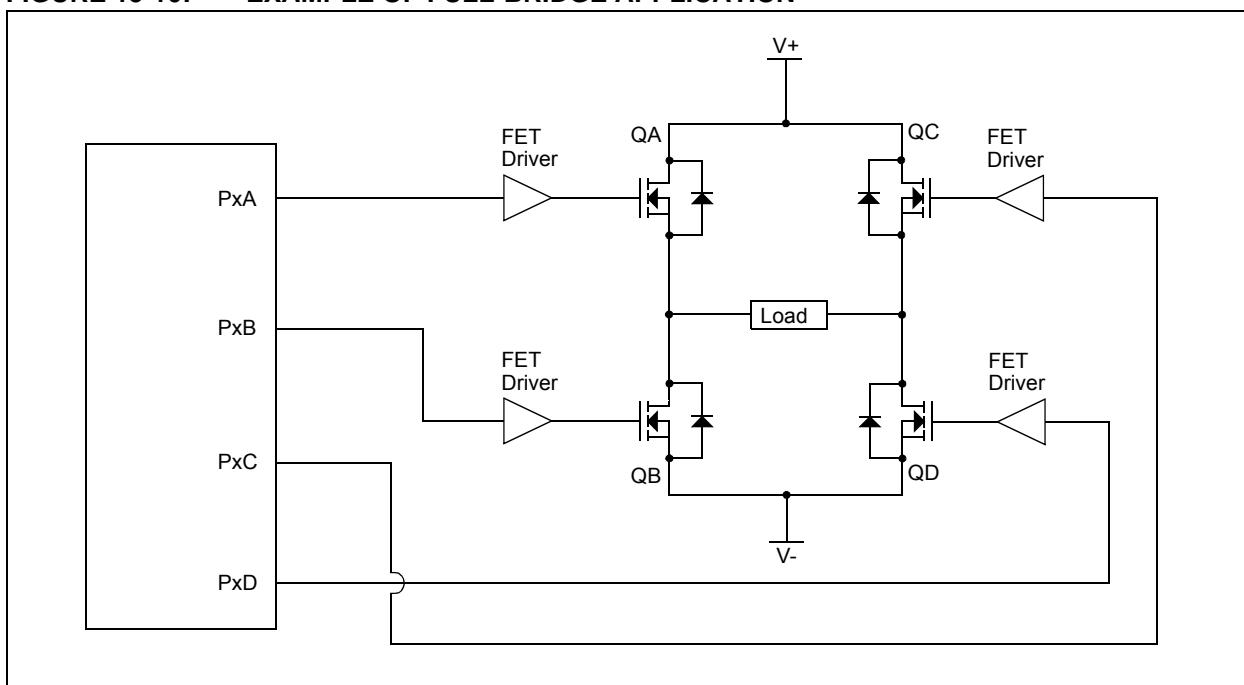
In Full-Bridge mode, all four pins are used as outputs. An example of full-bridge application is shown in [Figure 15-10](#).

In the Forward mode, pin CCPx/PxA is driven to its active state, pin PxD is modulated, while PxB and PxC will be driven to their inactive state as shown in [Figure 15-11](#).

In the Reverse mode, PxC is driven to its active state, pin PxB is modulated, while PxA and PxD will be driven to their inactive state as shown [Figure 15-11](#).

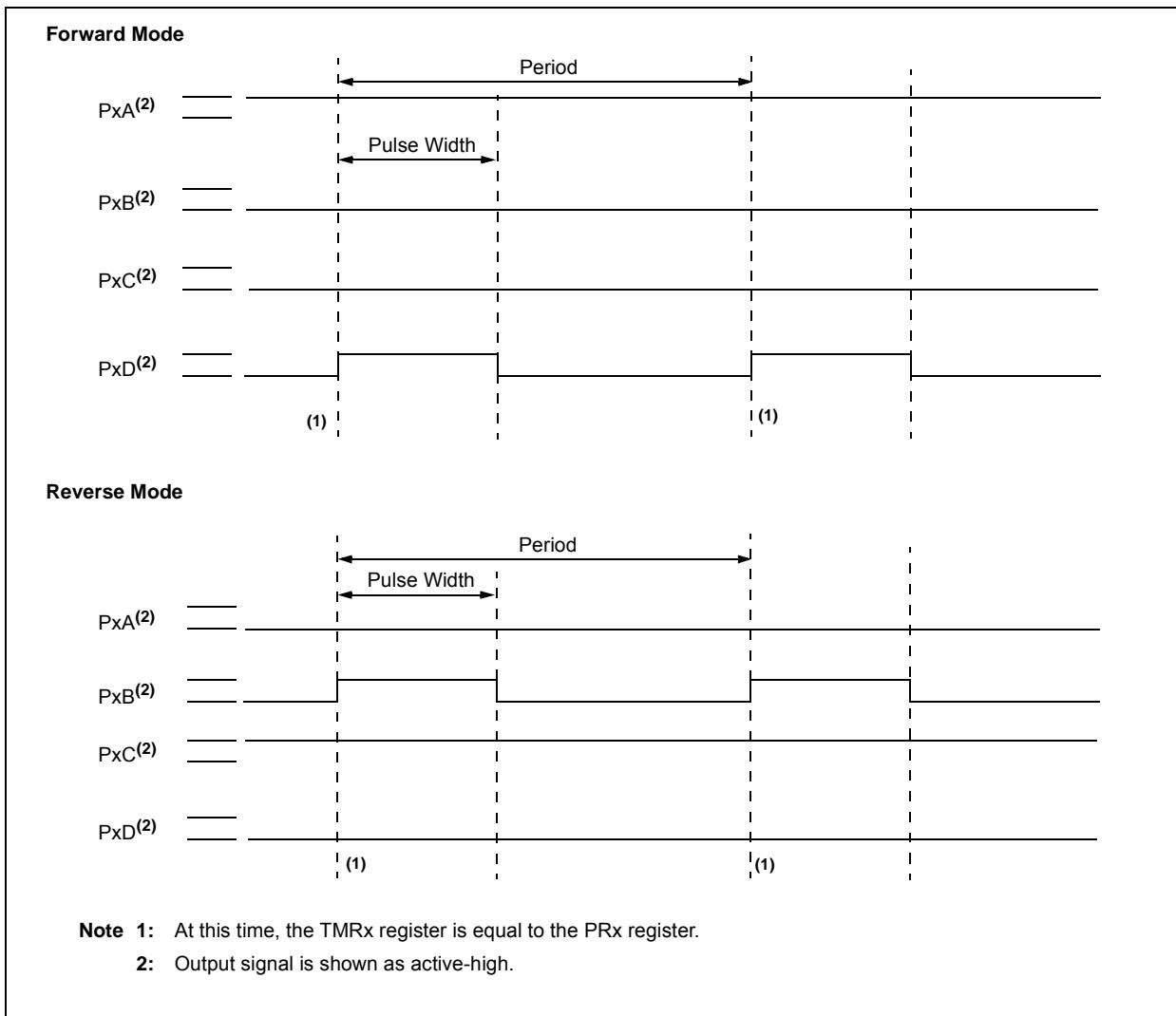
PxA, PxB, PxC and PxD outputs are multiplexed with the PORT data latches. The associated TRIS bits must be cleared to configure the PxA, PxB, PxC and PxD pins as outputs.

FIGURE 15-10: EXAMPLE OF FULL-BRIDGE APPLICATION



PIC18(L)F2X/45K50

FIGURE 15-11: EXAMPLE OF FULL-BRIDGE PWM OUTPUT



15.4.2.1 Direction Change in Full-Bridge Mode

In the Full-Bridge mode, the PxM1 bit in the CCPxCON 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 PxM1 bit of the CCPxCON register. The following sequence occurs four Timer cycles prior to the end of the current PWM period:

- The modulated outputs (Px_B and Px_D) are placed in their inactive state.
- The associated unmodulated outputs (Px_A and Px_C) are switched to drive in the opposite direction.
- PWM modulation resumes at the beginning of the next period.

See [Figure 15-12](#) 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.

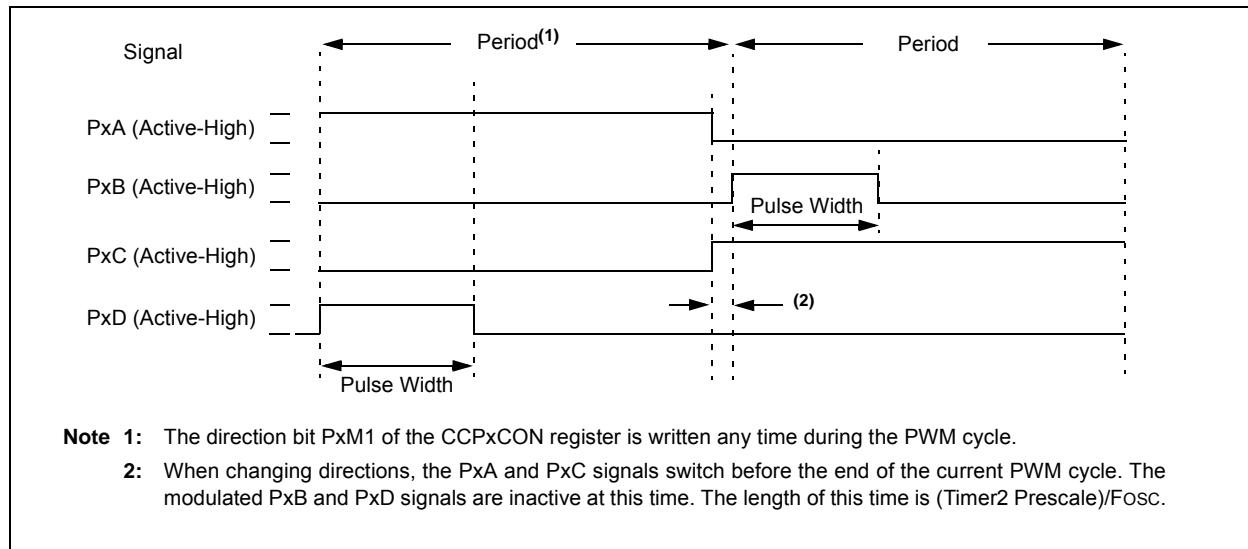
[Figure 15-13](#) shows an example of the PWM direction changing from forward to reverse, at a near 100% duty cycle. In this example, at time t₁, the output Px_A and Px_D become inactive, while output Px_C becomes active. Since the turn off time of the power devices is longer than the turn on time, a shoot-through current will flow through power devices QC and QD (see [Figure 15-10](#)) 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.

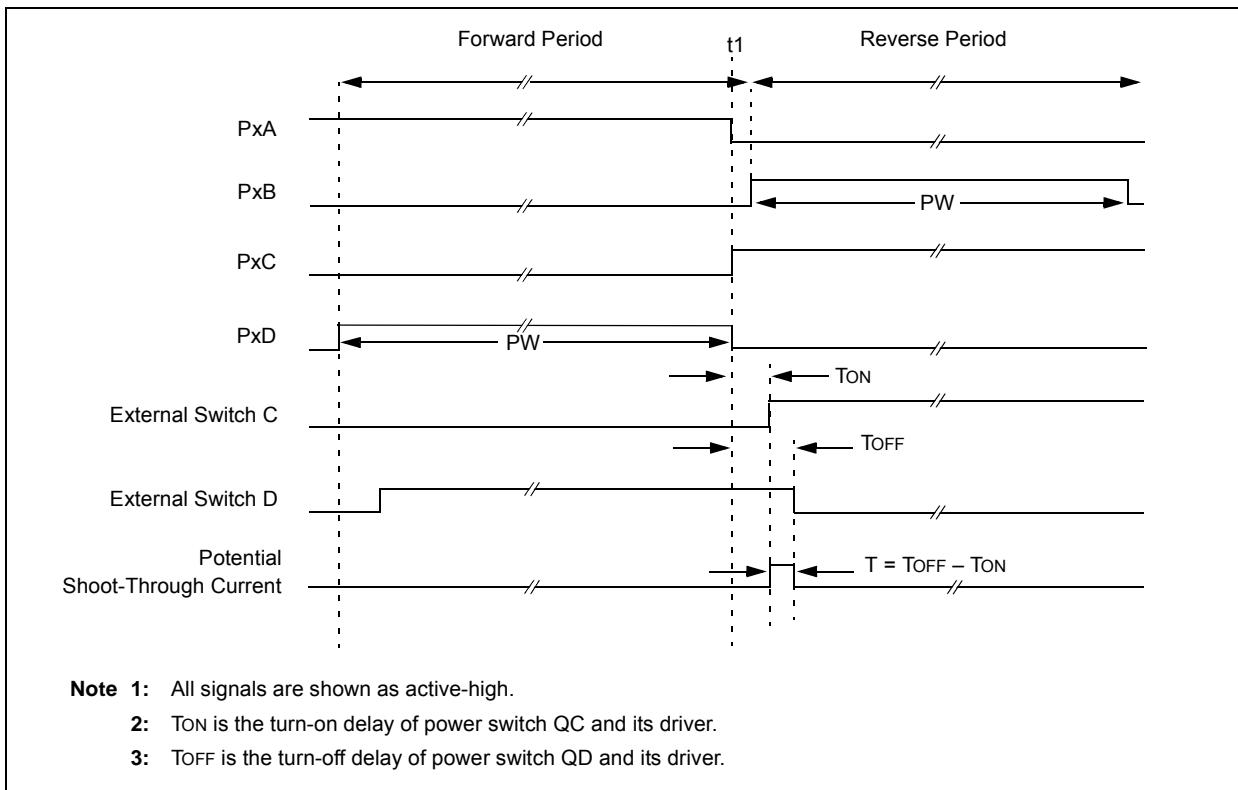
Other options to prevent shoot-through current may exist.

FIGURE 15-12: EXAMPLE OF PWM DIRECTION CHANGE



PIC18(L)F2X/45K50

FIGURE 15-13: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



15.4.3 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 ECCPxAS register. A shutdown event may be generated by:

- A logic ‘0’ on the FLTx pin
- Comparator Cx (async_CxOUT)
- Setting the ECCPxASE bit in firmware

A shutdown condition is indicated by the ECCPxASE (Auto-Shutdown Event Status) bit of the ECCPxAS 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 ECCPxASE bit is set to ‘1’. The ECCPxASE will remain set until cleared in firmware or an auto-restart occurs (see [Section 15.4.4 “Auto-Restart Mode”](#)).

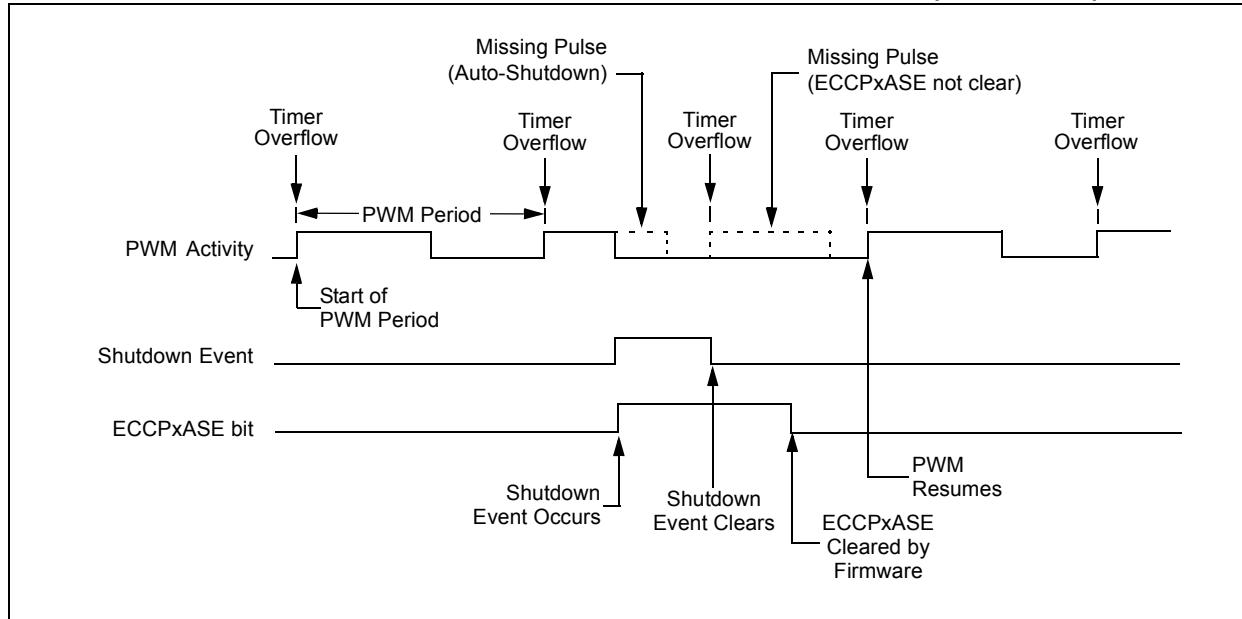
The enabled PWM pins are asynchronously placed in their shutdown states. The PWM output pins are grouped into pairs [Px A/Px C] and [Px B/Px D]. The state

of each pin pair is determined by the PSSxAC<1:0> and PSSxBD<1:0> bits of the ECCPxAS register. Each pin pair may be placed into one of three states:

- Drive logic ‘1’
- Drive logic ‘0’
- Tri-state (high-impedance)

- 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 ECCPxASE bit is disabled while an auto-shutdown condition persists.
- 3:** Once the auto-shutdown condition has been removed and the PWM restarted (either through firmware or auto-restart), the PWM signal will always restart at the beginning of the next PWM period.

FIGURE 15-14: PWM AUTO-SHUTDOWN WITH FIRMWARE RESTART (PxRSEN = 0)

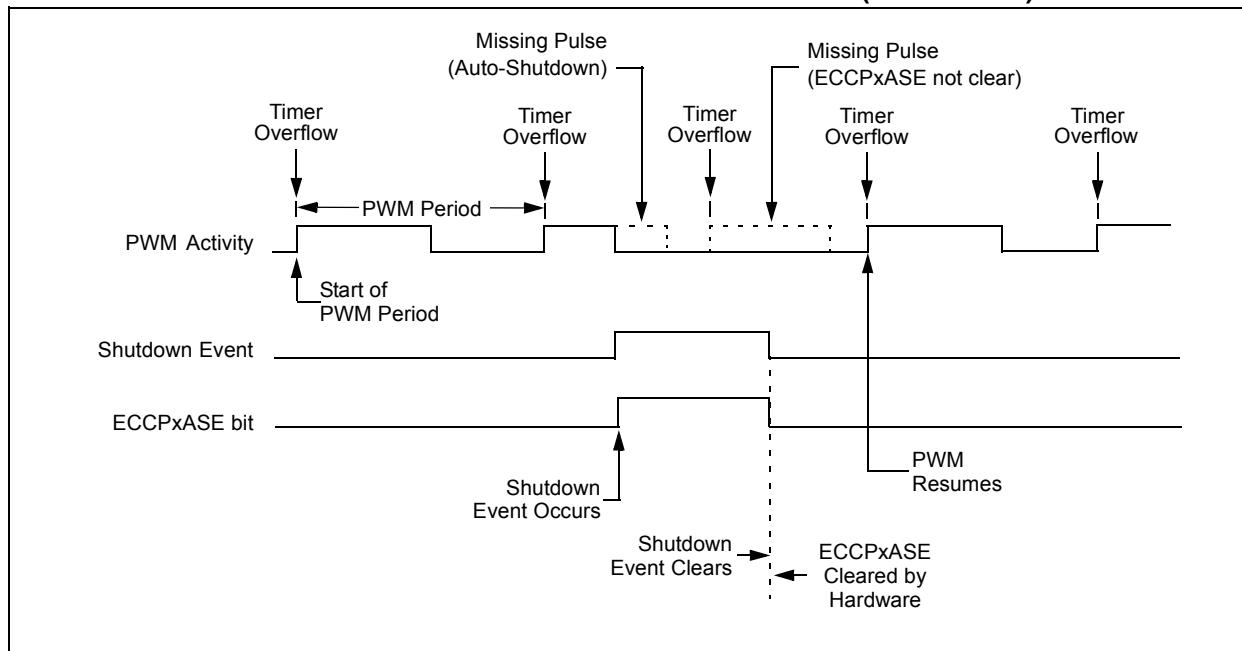


15.4.4 AUTO-RESTART MODE

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 PxRSEN bit in the PWMxCON register.

If auto-restart is enabled, the ECCPxASE bit will remain set as long as the auto-shutdown condition is active. When the auto-shutdown condition is removed, the ECCPxASE bit will be cleared via hardware and normal operation will resume.

FIGURE 15-15: PWM AUTO-SHUTDOWN WITH AUTO-RESTART (PxRSEN = 1)



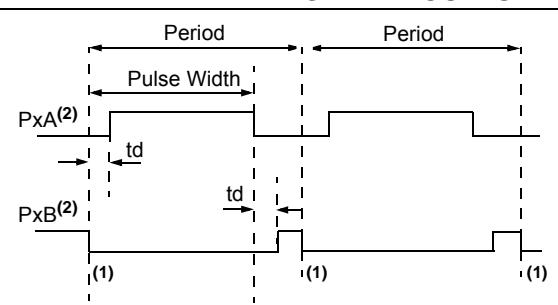
PIC18(L)F2X/45K50

15.4.5 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 15-16](#) for illustration. The lower seven bits of the associated PWMxCON register ([Register 15-5](#)) sets the delay period in terms of microcontroller instruction cycles (T_{CY} or 4 T_{osc}).

FIGURE 15-16: EXAMPLE OF HALF-BRIDGE PWM OUTPUT

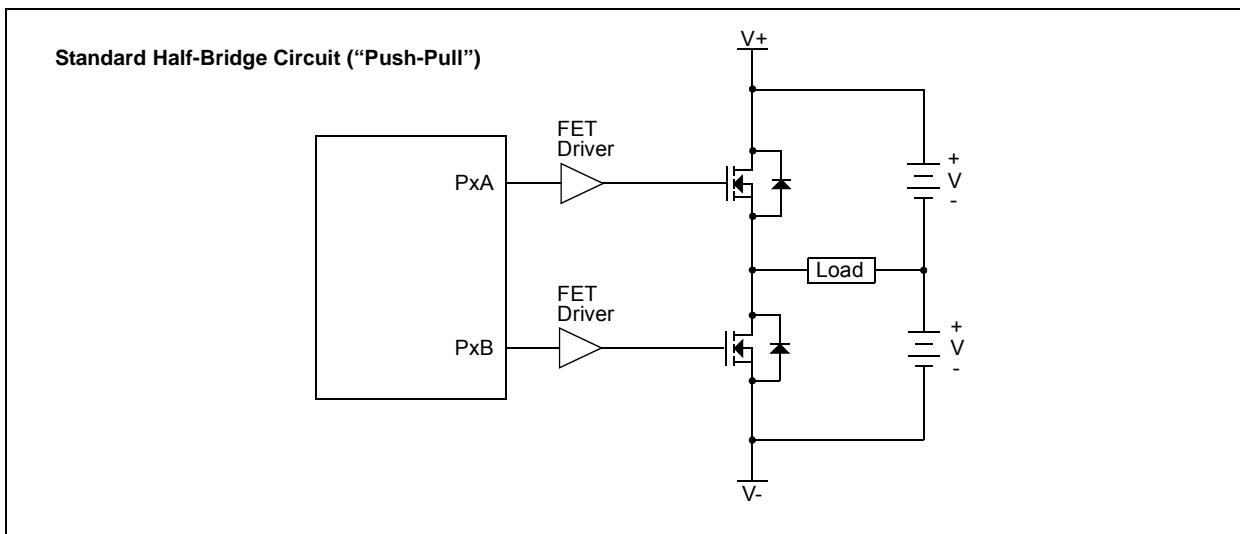


td = Dead-Band Delay

Note 1: At this time, the TMRx register is equal to the PRx register.

2: Output signals are shown as active-high.

FIGURE 15-17: EXAMPLE OF HALF-BRIDGE APPLICATIONS



15.4.6 PWM STEERING MODE

In Single Output mode, PWM steering allows any of the PWM pins to be the modulated signal. Additionally, the same PWM signal can be simultaneously available on multiple pins.

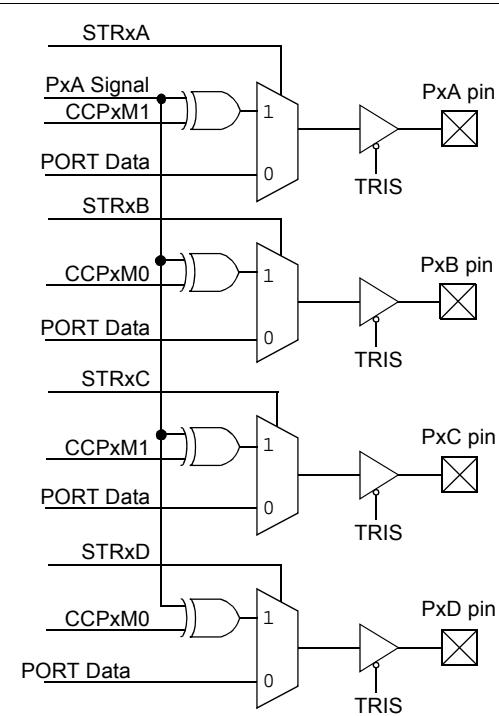
Once the Single Output mode is selected ($\text{CCPxM}<3:2> = 11$ and $\text{PxM}<1:0> = 00$ of the CCPxCON register), the user firmware can bring out the same PWM signal to one, two, three or four output pins by setting the appropriate Steering Enable bits (STRxA, STRxB, STRxC and/or STRxD) of the PSTRxCON register, as shown in [Table 15-12](#).

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, CCPxM<1:0> bits of the CCPxCON register select the PWM output polarity for the PxD, PxC, PxB and PxA pins.

The PWM auto-shutdown operation also applies to PWM Steering mode as described in [Section 15.4.3 "Enhanced PWM Auto-Shutdown Mode"](#). An auto-shutdown event will only affect pins that have PWM outputs enabled.

FIGURE 15-18: SIMPLIFIED STEERING BLOCK DIAGRAM



Note 1: Port outputs are configured as shown when the CCPxCON register bits $\text{PxM}<1:0> = 00$ and $\text{CCPxM}<3:2> = 11$.

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

15.4.6.1 Steering Synchronization

The STRxSYNC bit of the PSTRxCON register gives the user two selections of when the steering event will happen. When the STRxSYNC bit is '0', the steering event will happen at the end of the instruction that writes to the PSTRxCON register. In this case, the output signal at the Px A, Px B, Px C and Px D 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 STRxSYNC 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 [15-19](#) and [15-20](#) illustrate the timing diagrams of the PWM steering depending on the STRxSYNC setting.

PIC18(L)F2X/45K50

15.4.7 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.

The CCPxM<1:0> bits of the CCPxCON register allow the user to choose whether the PWM output signals are active-high or active-low for each pair of PWM output pins (Px A/PxC and Px B/PxD). 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 enabled is not recommended since it may result in damage to the application circuits.

The Px A, Px B, Px C and Px D 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.

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).

FIGURE 15-19: EXAMPLE OF STEERING EVENT AT END OF INSTRUCTION (STRxSYNC = 0)

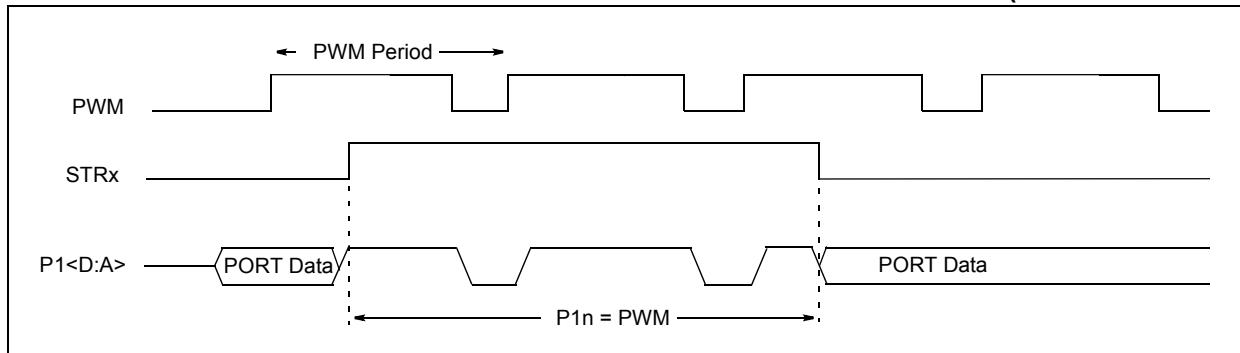
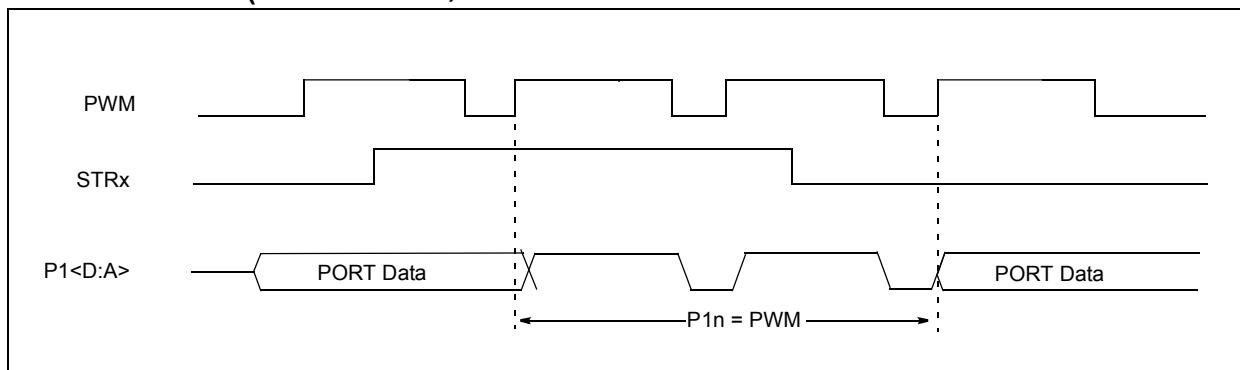


FIGURE 15-20: EXAMPLE OF STEERING EVENT AT BEGINNING OF INSTRUCTION (STRxSYNC = 1)



15.4.8 SETUP FOR ECCP PWM OPERATION USING ECCP1 AND TIMER2

The following steps should be taken when configuring the ECCP1 module for PWM operation using Timer2:

1. Configure the PWM pins to be used (P1A, P1B, P1C, and P1D):
 - Configure PWM outputs to be used as inputs by setting the corresponding TRIS bits. This prevents spurious outputs during setup.
 - Set the PSTR1CON bits for each PWM output to be used.
2. Set the PWM period by loading the PR2 register.
3. Configure auto-shutdown as OFF or select the source with the CCP1AS<2:0> bits of the ECCP1AS register.
4. Configure the auto-shutdown sources as needed:
 - Configure each comparator used.
 - Configure the comparator inputs as analog.
 - Configure the $\overline{FLT0}$ input pin and clear ANSB0.
5. Force a shutdown condition (OFF included):
 - Configure safe starting output levels by setting the default shutdown drive states with the PSS1AC<1:0> and PSS1BD<1:0> bits of the ECCP1AS register.
 - Clear the P1RSEN bit of the PWM1CON register.
 - Set the CCP1AS bit of the ECCP1AS register.
6. Configure the ECCP1 module for the desired PWM mode and configuration by loading the CCP1CON register with the appropriate values:
 - Select one of the available output configurations and direction with the P1M<1:0> bits.
 - Select the polarities of the PWM output signals with the CCP1M<3:0> bits.
7. Set the 10-bit PWM duty cycle:
 - Load the eight MS bits into the CCPR1L register.
 - Load the two LS bits into the DC<1:0> bits of the CCP1CON register.
8. For Half-Bridge Output mode, set the dead-band delay by loading P1DC<6:0> bits of the PWM1CON register with the appropriate value.
9. Configure and start TMR2:
 - Set the TMR2 prescale value by loading the T2CKPS bits of the T2CON register.
 - Start Timer2 by setting the TMR2ON bit.
10. Enable the ECCP1/P1A, P1B, P1C and/or P1D pin outputs by clearing the respective TRIS bits.
11. Start the PWM:
 - If shutdown auto-restart is used, then set the P1RSEN bit of the PWM1CON register.
 - If shutdown auto-restart is not used, then clear the CCP1ASE bit of the ECCP1AS register.

PIC18(L)F2X/45K50

TABLE 15-12: REGISTERS ASSOCIATED WITH ENHANCED PWM

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page | |
|----------------------|------------------------|--------------|-----------|----------|--------|-------------|-------------|-------------|------------------|-----|
| ECCP1AS | ECCP1ASE | ECCP1AS<2:0> | | | | PSS1AC<1:0> | | PSS1BD<1:0> | | 201 |
| CCP1CON | P1M<1:0> | | DC1B<1:0> | | | CCP1M<3:0> | | | | 197 |
| CCPTMRS | — | — | — | — | C2TSEL | — | — | C1TSEL | 200 | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 | |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 | |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 | |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 | |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 | |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 | |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 | |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 | |
| PMD1 | — | MSSPMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | 62 | |
| PR2 | Timer2 Period Register | | | | | | | | — | |
| PSTR1CON | — | — | — | STR1SYNC | STR1D | STR1C | STR1B | STR1A | 202 | |
| PWM1CON | P1RSEN | P1DC<6:0> | | | | | | | 202 | |
| T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | 170 | |
| TMR2 | Timer2 Period Register | | | | | | | | — | |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 | |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 149 | |
| TRISD ⁽¹⁾ | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 149 | |

Legend: — = Unimplemented location, read as '0'. Shaded bits are not used by Capture mode.

Note 1: These registers/bits are available on PIC18(L)F45K50 devices.

15.5 Register Definitions: ECCP Control

REGISTER 15-1: CCPxCON: STANDARD CCPx CONTROL REGISTER

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-----|-----------|-------|-------|------------|-------|-------|
| — | — | DCxB<1:0> | | | CCPxM<3:0> | | |
| 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 Reset

'1' = Bit is set

'0' = Bit is cleared

bit 7-6 **Unused**

bit 5-4 **DCxB<1:0>**: PWM Duty Cycle Least Significant bits

Capture mode:

Unused

Compare mode:

Unused

PWM mode:

These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM<3:0>**: ECCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets the 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: set output on compare match (CCPx pin is set, CCPxIF is set)

1001 = Compare mode: clear output on compare match (CCPx pin is cleared, CCPxIF is set)

1010 = Compare mode: generate software interrupt on compare match (CCPx pin is unaffected, CCPxIF is set)

1011 = Compare mode: Special Event Trigger (CCPx pin is unaffected, CCPxIF is set)

TimerX (selected by CxTSEL bits) is reset

ADON is set, starting A/D conversion if A/D module is enabled and TRIGSEL is clear⁽¹⁾

11xx =: PWM mode

Note 1: This feature is available on CCP2 only.

PIC18(L)F2X/45K50

REGISTER 15-2: CCPxCON: ENHANCED CCPx CONTROL REGISTER

| R/x-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | | | |
|-------|----------|-------|-----------|------------|-------|-------|-------|--|--|--|
| | PxM<1:0> | | DCxB<1:0> | CCPxM<3:0> | | | | | | |
| 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 Reset

'1' = Bit is set

'0' = Bit is cleared

bit 7-6

PxM<1:0>: Enhanced PWM Output Configuration bits

If CCPxM<3:2> = 00, 01, 10: (Capture/Compare modes)

$_{xx}$ = PxA assigned as Capture/Compare input; Px B, Px C, Px D assigned as port pins

Half-Bridge ECCP Modules⁽¹⁾:

If CCPxM<3:2> = 11: (PWM modes)

$_{0x}$ = Single output; Px A modulated; Px B assigned as port pin

$_{1x}$ = Half-bridge output; Px A, Px B modulated with dead-band control

Full-Bridge ECCP Modules⁽¹⁾:

If CCPxM<3:2> = 11: (PWM modes)

$_{00}$ = Single output; Px A modulated; Px B, Px C, Px D assigned as port pins

$_{01}$ = Full-bridge output forward; Px D modulated; Px A active; Px B, Px C inactive

$_{10}$ = Half-bridge output; Px A, Px B modulated with dead-band control; Px C, Px D assigned as port pins

$_{11}$ = Full-bridge output reverse; Px B modulated; Px C active; Px A, Px D inactive

bit 5-4

DCxB<1:0>: PWM Duty Cycle Least Significant bits

Capture mode:

Unused

Compare mode:

Unused

PWM mode:

These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

Note 1: See [Table 15-1](#) to determine full-bridge and half-bridge ECCPs for the device being used.

REGISTER 15-2: CCPxCON: ENHANCED CCPx CONTROL REGISTER (CONTINUED)

bit 3-0

CCPxM<3:0>: ECCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets the 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: set output on compare match (CCPx pin is set, CCPxIF is set)

1001 = Compare mode: clear output on compare match (CCPx pin is cleared, CCPxIF is set)

1010 = Compare mode: generate software interrupt on compare match (CCPx pin is unaffected, CCPxIF is set)

1011 = Compare mode: Special Event Trigger (CCPx pin is unaffected, CCPxIF is set)
TimerX is reset

Half-Bridge ECCP Modules⁽¹⁾:

1100 = PWM mode: Px A active-high; Px B active-high

1101 = PWM mode: Px A active-high; Px B active-low

1110 = PWM mode: Px A active-low; Px B active-high

1111 = PWM mode: Px A active-low; Px B active-low

Full-Bridge ECCP Modules⁽¹⁾:

1100 = PWM mode: Px A, Px C active-high; Px B, Px D active-high

1101 = PWM mode: Px A, Px C active-high; Px B, Px D active-low

1110 = PWM mode: Px A, Px C active-low; Px B, Px D active-high

1111 = PWM mode: Px A, Px C active-low; Px B, Px D active-low

Note 1: See [Table 15-1](#) to determine full-bridge and half-bridge ECCPs for the device being used.

PIC18(L)F2X/45K50

REGISTER 15-3: CCPTMRS: PWM TIMER SELECTION CONTROL REGISTER 0

| U-0 | U-0 | U-0 | U-0 | R/W-0 | U-0 | U-0 | R/W-0 |
|-------|-----|-----|-----|--------|-----|-----|--------|
| — | — | — | — | C2TSEL | — | — | C1TSEL |
| 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 7-4 **Unimplemented:** Read as '0'

bit 3 **C2TSEL:** CCP2 Timer Selection bit

0 = CCP2 – Capture/Compare modes use TMR1, PWM modes use TMR2

1 = CCP2 – Capture/Compare modes use TMR3, PWM modes use TMR2

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **C1TSEL:** ECCP1 Timer Selection bit

0 = ECCP1 – Capture/Compare modes use TMR1, PWM modes use TMR2

1 = ECCP1 – Capture/Compare modes use TMR3, PWM modes use TMR2

REGISTER 15-4: ECCPxAS: CCPX 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 |
|----------|--------------|-------|-------|-------------|-------|-------------|-------|
| ECCPxASE | ECCPxAS<2:0> | | | PSSxAC<1:0> | | PSSxBD<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 | ECCPxASE: CCPx Auto-shutdown Event Status bit if PxRSEN = 1; 1 = An Auto-shutdown event occurred; ECCPxASE bit will automatically clear when event goes away; CCPx outputs in shutdown state 0 = CCPx outputs are operating if PxRSEN = 0; 1 = An Auto-shutdown event occurred; bit must be cleared in software to restart PWM; CCPx outputs in shutdown state 0 = CCPx outputs are operating |
| bit 6-4 | ECCPxAS<2:0>: CCPx Auto-Shutdown Source Select bits (1) 000 = Auto-shutdown is disabled 001 = Comparator C1 (async_C1OUT) – output high will cause shutdown event 010 = Comparator C2 (async_C2OUT) – output high will cause shutdown event 011 = Either Comparator C1 or C2 – output high will cause shutdown event 100 = <u>FLT0</u> pin - low level will cause shutdown event 101 = <u>FLT0</u> pin or Comparator C1 (async_C1OUT) – low level will cause shutdown event 110 = <u>FLT0</u> pin or Comparator C2 (async_C2OUT) – low level will cause shutdown event 111 = <u>FLT0</u> pin or Comparators C1 or C2 – low level will cause shutdown event |
| bit 3-2 | PSSxAC<1:0>: Pins Px A and Px C Shutdown State Control bits 00 = Drive pins Px A and Px C to '0' 01 = Drive pins Px A and Px C to '1' 1x = Pins Px A and Px C tri-state |
| bit 1-0 | PSSxBD<1:0>: Pins Px B and Px D Shutdown State Control bits 00 = Drive pins Px B and Px D to '0' 01 = Drive pins Px B and Px D to '1' 1x = Pins Px B and Px D tri-state |

Note 1: If C1SYNC or C2SYNC bits in the CM2CON1 register are enabled, the shutdown will be delayed by Timer1.

PIC18(L)F2X/45K50

REGISTER 15-5: PWMxCON: 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 |
|--------|-------|-------|-------|-----------|-------|-------|-------|
| PxRSEN | | | | PxDC<6:0> | | | |
| 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 **PxRSEN:** PWM Restart Enable bit
1 = Upon auto-shutdown, the ECCPxASE bit clears automatically once the shutdown event goes away; the PWM restarts automatically
0 = Upon auto-shutdown, ECCPxASE must be cleared in software to restart the PWM
- bit 6-0 **PxDC<6:0>:** PWM Delay Count bits
PxDCx = 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

REGISTER 15-6: PSTRxCON: PWM 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 |
|-------|-----|-----|----------|-------|-------|-------|-------|
| — | — | — | STRxSYNC | STRxD | STRxC | STRxB | STRxA |
| 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 **STRxSYNC:** 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 **STRxD:** Steering Enable bit D
1 = PxD pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxD pin is assigned to port pin
- bit 2 **STRxC:** Steering Enable bit C
1 = PxC pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxC pin is assigned to port pin
- bit 1 **STRxB:** Steering Enable bit B
1 = PxB pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxB pin is assigned to port pin
- bit 0 **STRxA:** Steering Enable bit A
1 = PxA pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxA pin is assigned to port pin

Note 1: The PWM Steering mode is available only when the CCPxCON register bits CCPxM<3:2> = 11 and PxM<1:0> = 00.

16.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

16.1 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:

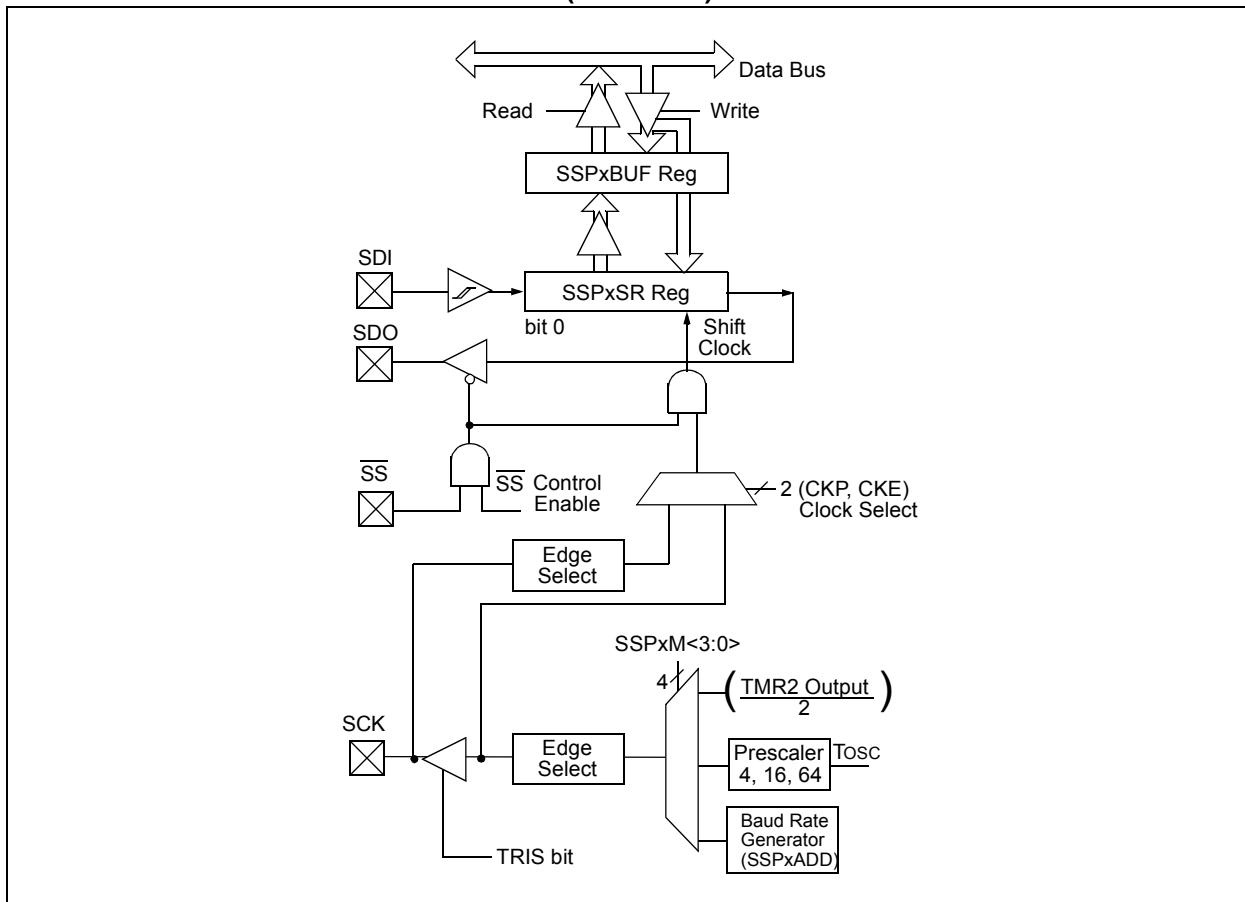
- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)

The SPI interface supports the following modes and features:

- Master mode
- Slave mode
- Clock Parity
- Slave Select Synchronization (Slave mode only)
- Daisy chain connection of slave devices

[Figure 16-1](#) is a block diagram of the SPI interface module.

FIGURE 16-1: MSSP BLOCK DIAGRAM (SPI MODE)



PIC18(L)F2X/45K50

The I²C interface supports the following modes and features:

- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited Multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection

- General call address matching
- Address masking
- Address Hold and Data Hold modes
- Selectable SDA hold times

Figure 16-2 is a block diagram of the I²C interface module in Master mode. Figure 16-3 is a diagram of the I²C interface module in Slave mode.

FIGURE 16-2: MSSP BLOCK DIAGRAM (I²C™ MASTER MODE)

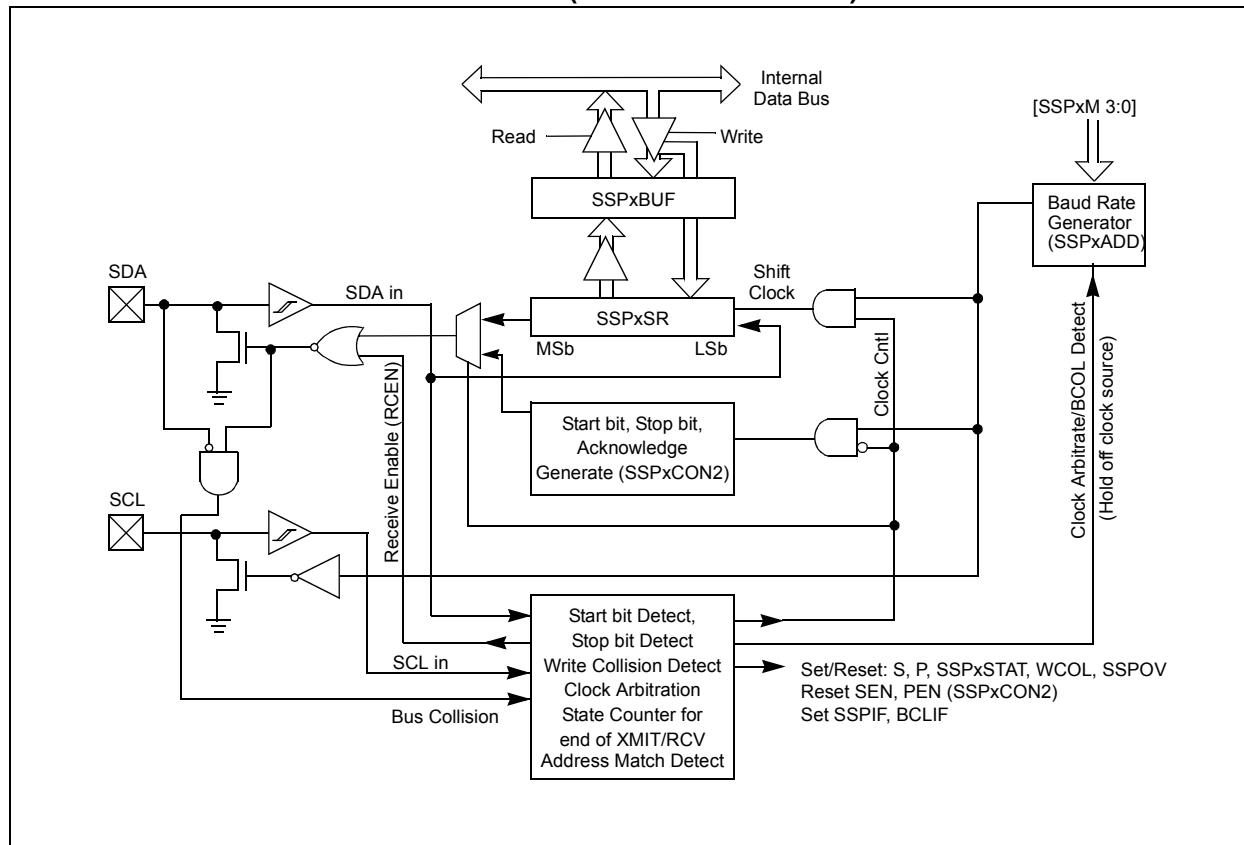
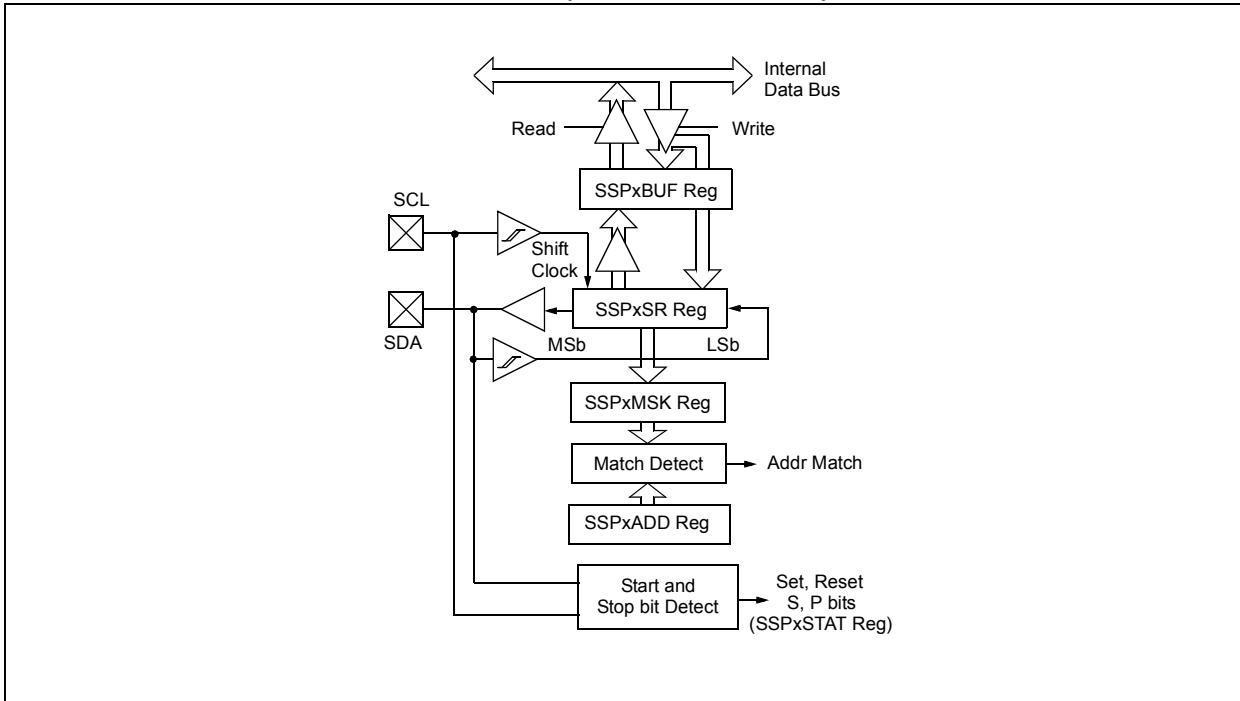


FIGURE 16-3: MSSP BLOCK DIAGRAM (I²C™ SLAVE MODE)



16.2 SPI Mode Overview

The Serial Peripheral Interface (SPI) bus is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a chip select known as Slave Select.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Slave Select (\overline{SS})

[Figure 16-1](#) shows the block diagram of the MSSP module when operating in SPI Mode.

The SPI bus operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

[Figure 16-4](#) shows a typical connection between a master device and multiple slave devices.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

Transmissions involve two shift registers, eight bits in size, one in the master and one in the slave. With either the master or the slave device, data is always shifted out one bit at a time, with the Most Significant bit (MSb) shifted out first. At the same time, a new Least Significant bit (LSb) is shifted into the same register.

[Figure 16-5](#) shows a typical connection between two processors configured as master and slave devices.

Data is shifted out of both shift registers on the programmed clock edge and latched on the opposite edge of the clock.

The master device transmits information out on its SDO output pin which is connected to, and received by, the slave's SDI input pin. The slave device transmits information out on its SDO output pin, which is connected to, and received by, the master's SDI input pin.

To begin communication, the master device first sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

The master device starts a transmission by sending out the MSb from its shift register. The slave device reads this bit from that same line and saves it into the LSb position of its shift register.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that at the same time, the slave device is sending out the MSb from its shift

register and the master device is reading this bit from that same line and saving it as the LSb of its shift register.

After eight bits have been shifted out, the master and slave have exchanged register values.

If there is more data to exchange, the shift registers are loaded with new data and the process repeats itself.

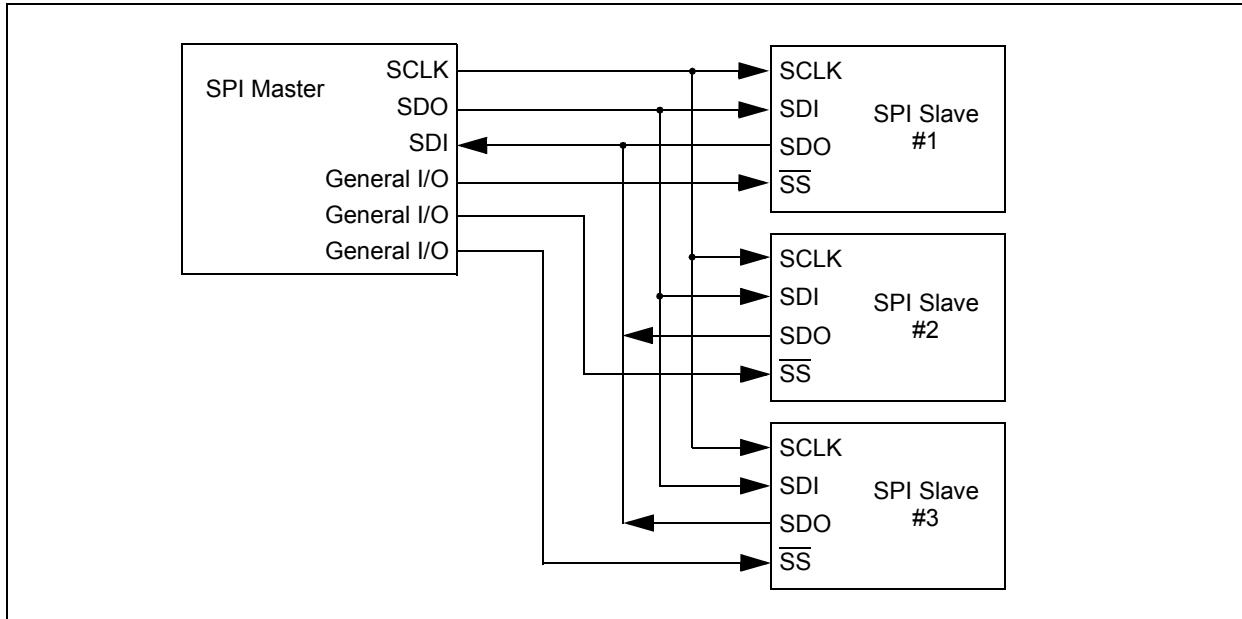
Whether the data is meaningful or not (dummy data), depends on the application software. This leads to three scenarios for data transmission:

- Master sends useful data and slave sends dummy data.
- Master sends useful data and slave sends useful data.
- Master sends dummy data and slave sends useful data.

Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops sending the clock signal and it deselects the slave.

Every slave device connected to the bus that has not been selected through its slave select line must disregard the clock and transmission signals and must not transmit out any data of its own.

FIGURE 16-4: SPI MASTER AND MULTIPLE SLAVE CONNECTION



16.2.1 SPI MODE REGISTERS

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

- MSSP STATUS register (SSPxSTAT)
- MSSP Control register 1 (SSPxCON1)
- MSSP Control register 3 (SSPxCON3)
- MSSP Data Buffer register (SSPxBUF)
- MSSP Address register (SSPxADD)
- MSSP Shift register (SSPxSR)
(Not directly accessible)

SSPxCON1 and SSPxSTAT are the control and STATUS registers in SPI mode operation. The SSPxCON1 register is readable and writable. The lower six bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

In one SPI Master mode, SSPxADD can be loaded with a value used in the Baud Rate Generator. More information on the Baud Rate Generator is available in [Section 16.7 “Baud Rate Generator”](#).

SSPxSR is the shift register used for shifting data in and out. SSPxBUF provides indirect access to the SSPxSR register. SSPxBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPxSR and SSPxBUF together create a buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPIF interrupt is set.

During transmission, the SSPxBUF is not buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

16.2.2 SPI MODE OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPxCON1<5:0> and SSPxSTAT<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)

To enable the serial port, SSPx Enable bit, SSPxEN of the SSPxCON1 register, must be set. To reset or reconfigure SPI mode, clear the SSPxEN bit, re-initialize the SSPxCONx registers and then set the SSPxEN bit. This configures the SDI, SDO, SCK and SS pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI must have corresponding TRIS bit set
- SDO must have corresponding TRIS bit cleared
- SCK (Master mode) must have corresponding TRIS bit cleared
- SCK (Slave mode) must have corresponding TRIS bit set
- SS must have corresponding TRIS bit set

PIC18(L)F2X/45K50

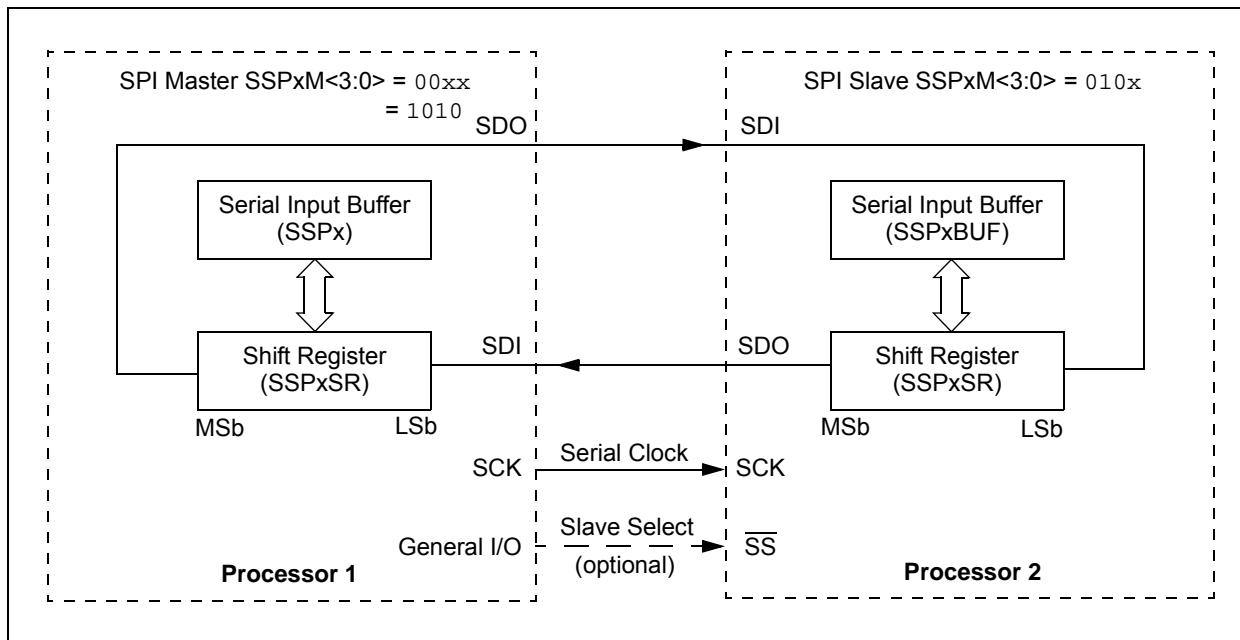
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

The MSSP consists of a transmit/receive shift register (SSPxSR) and a buffer register (SSPxBUF). The SSPxSR shifts the data in and out of the device, MSb first. The SSPxBUF holds the data that was written to the SSPxSR until the received data is ready. Once the eight bits of data have been received, that byte is moved to the SSPxBUF register. Then, the Buffer Full Detect bit, BF of the SSPxSTAT register, indicates when SSPxBUF has been loaded with the received data (transmission is complete). When the SSPxBUF 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.

register, will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPxBUF register to complete successfully.

When the application software is expecting to receive valid data, the SSPxBUF should be read before the next byte of data to transfer is written to the SSPxBUF. The Buffer Full bit, BF of the SSPxSTAT register, indicates when SSPxBUF has been loaded with the received data (transmission is complete). When the SSPxBUF 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.

FIGURE 16-5: SPI MASTER/SLAVE CONNECTION



16.2.3 SPI MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK line. The master determines when the slave (Processor 2, Figure 16-5) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPxBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPxSR 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 SSPxBUF register as if a normal received byte (interrupts and Status bits appropriately set).

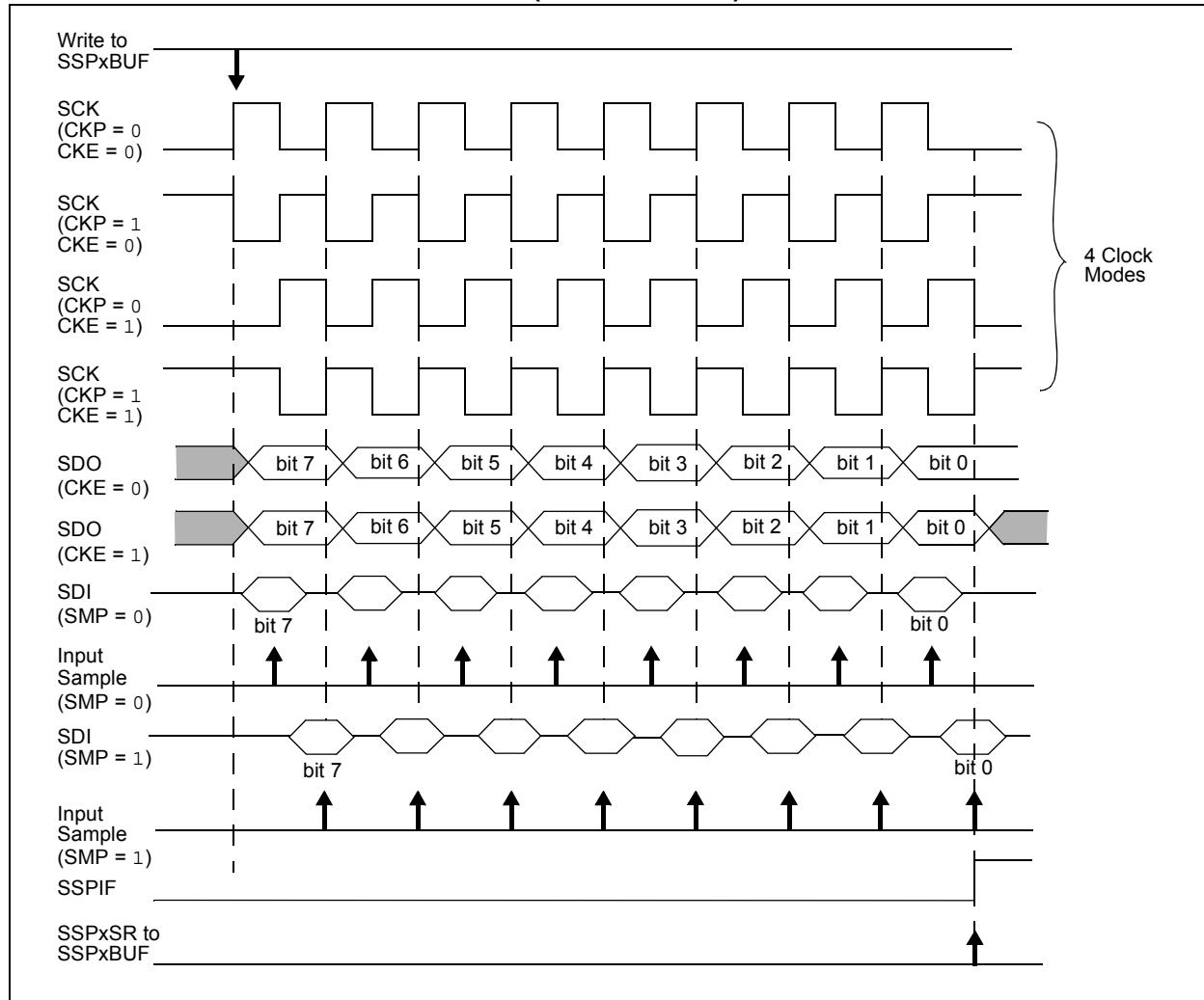
The clock polarity is selected by appropriately programming the CKP bit of the SSPxCON1 register and the CKE bit of the SSPxSTAT register. This then, would give waveforms for SPI communication as shown in Figure 16-6, Figure 16-8 and Figure 16-9, where the MSB is transmitted first. In Master mode, the 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
- Fosc/(4 * (SSPxADD + 1))

Figure 16-6 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 SSPxBUF is loaded with the received data is shown.

FIGURE 16-6: SPI MODE WAVEFORM (MASTER MODE)



16.2.4 SPI 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 SSPxCON1 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. The shift register is clocked from the SCK pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

16.2.4.1 Daisy-Chain Configuration

The SPI bus can sometimes be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole chain acts as one large communication shift register. The daisy-chain feature only requires a single Slave Select line from the master device.

Figure 16-7 shows the block diagram of a typical daisy-chain connection when operating in SPI mode.

In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the BOEN bit of the SSPxCON3 register will enable writes to the SSPxBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

16.2.5 SLAVE SELECT SYNCHRONIZATION

The Slave Select can also be used to synchronize communication. The Slave Select line is held high until the master device is ready to communicate. When the Slave Select line is pulled low, the slave knows that a new transmission is starting.

If the slave fails to receive the communication properly, it will be reset at the end of the transmission, when the Slave Select line returns to a high state. The slave is then ready to receive a new transmission when the Slave Select line is pulled low again. If the Slave Select line is not used, there is a risk that the slave will eventually become out of sync with the master. If the slave misses a bit, it will always be one bit off in future transmissions. Use of the Slave Select line allows the slave and master to align themselves at the beginning of each transmission.

The $\overline{\text{SS}}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{\text{SS}}$ pin control enabled ($\text{SSPxCON1<3:0>} = 0100$).

When the $\overline{\text{SS}}$ pin is low, transmission and reception are enabled and the SDO pin is driven.

When the $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ pin control enabled ($\text{SSPxCON1<3:0>} = 0100$), the SPI module will reset if the $\overline{\text{SS}}$ pin is set to VDD.
- 2:** When the SPI is used in Slave mode with CKE set; the user must enable $\overline{\text{SS}}$ pin control.
- 3:** While operated in SPI Slave mode the SMP bit of the SSPxSTAT register must remain clear.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the $\overline{\text{SS}}$ pin to a high level or clearing the SSPxEN bit.

FIGURE 16-7: SPI DAISY-CHAIN CONNECTION

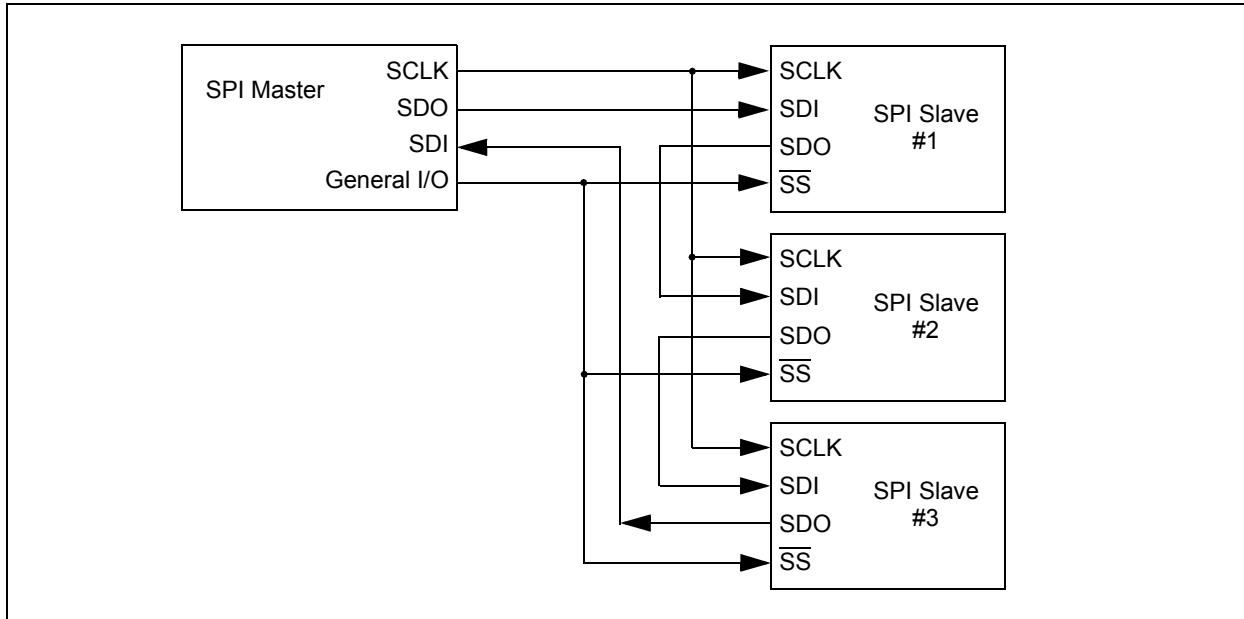
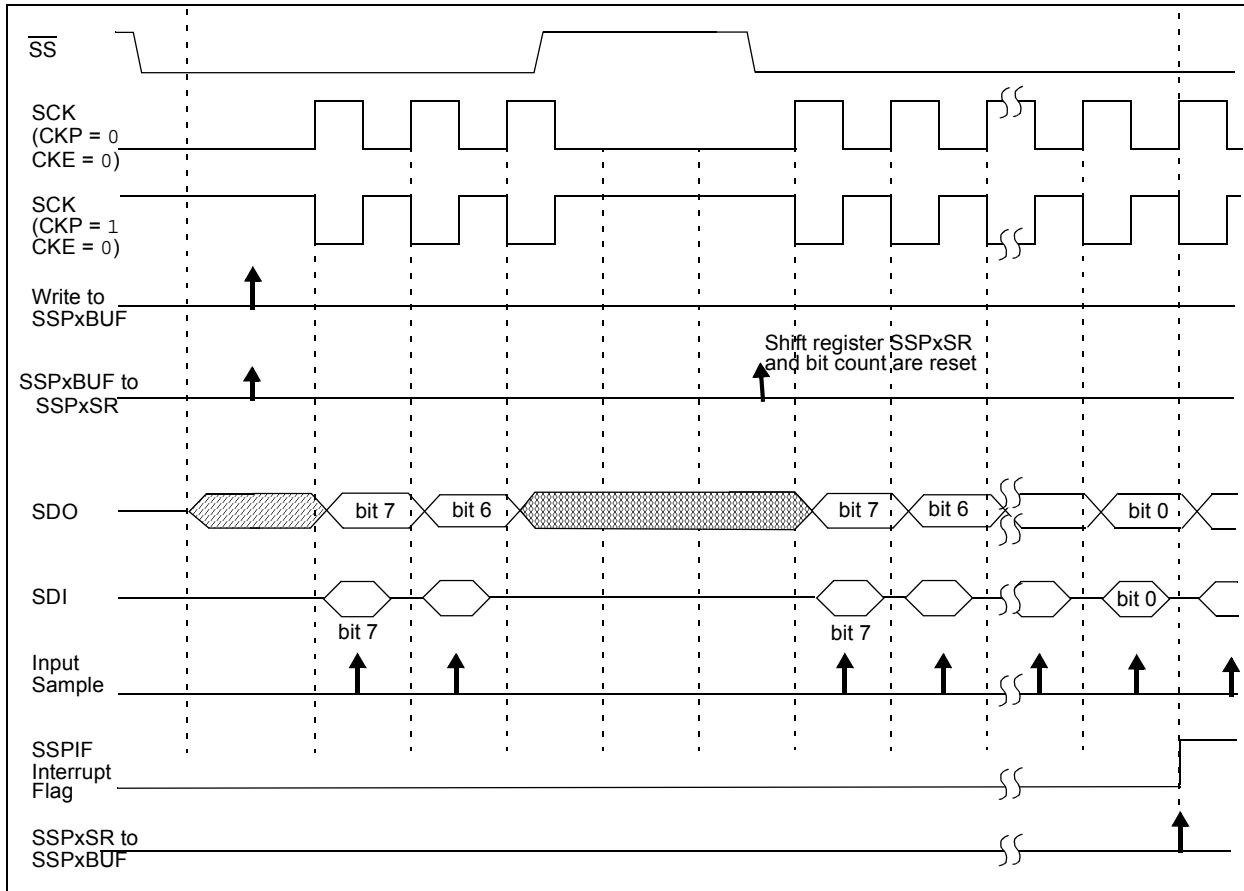


FIGURE 16-8: SLAVE SELECT SYNCHRONOUS WAVEFORM



PIC18(L)F2X/45K50

FIGURE 16-9: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

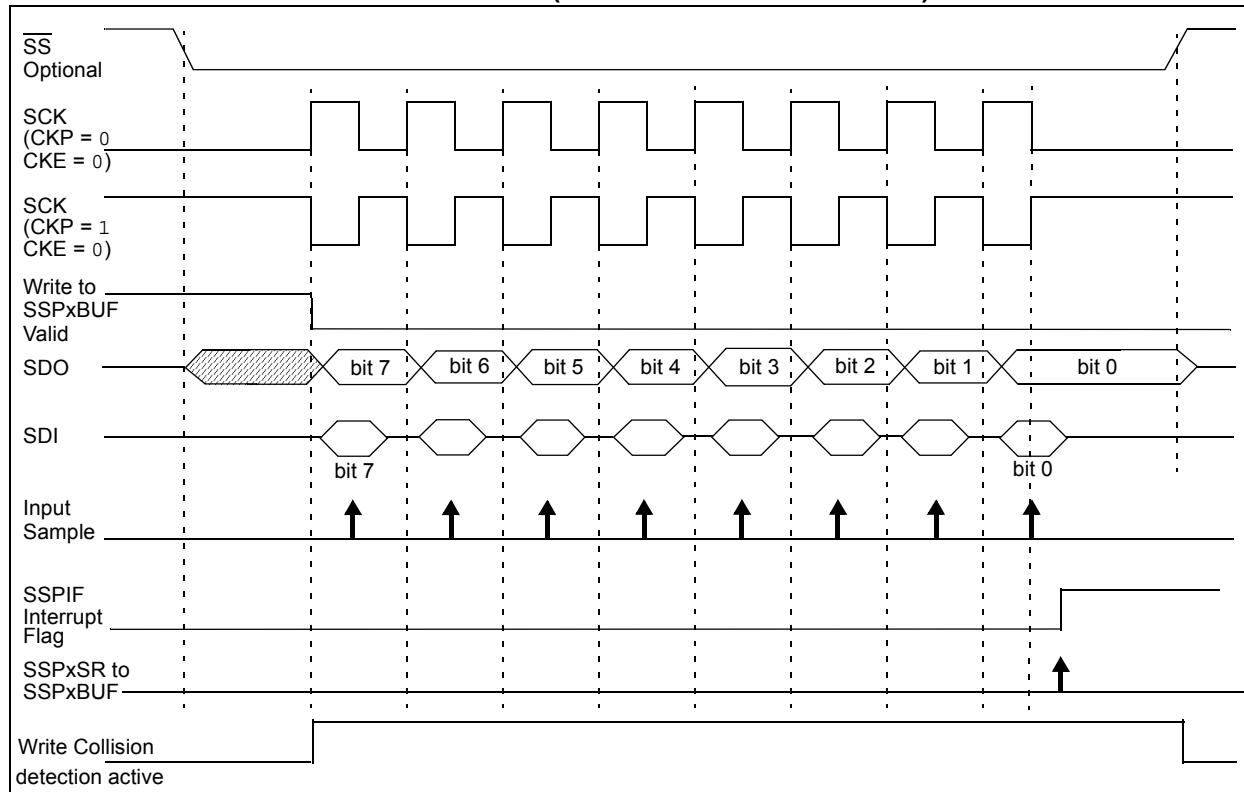
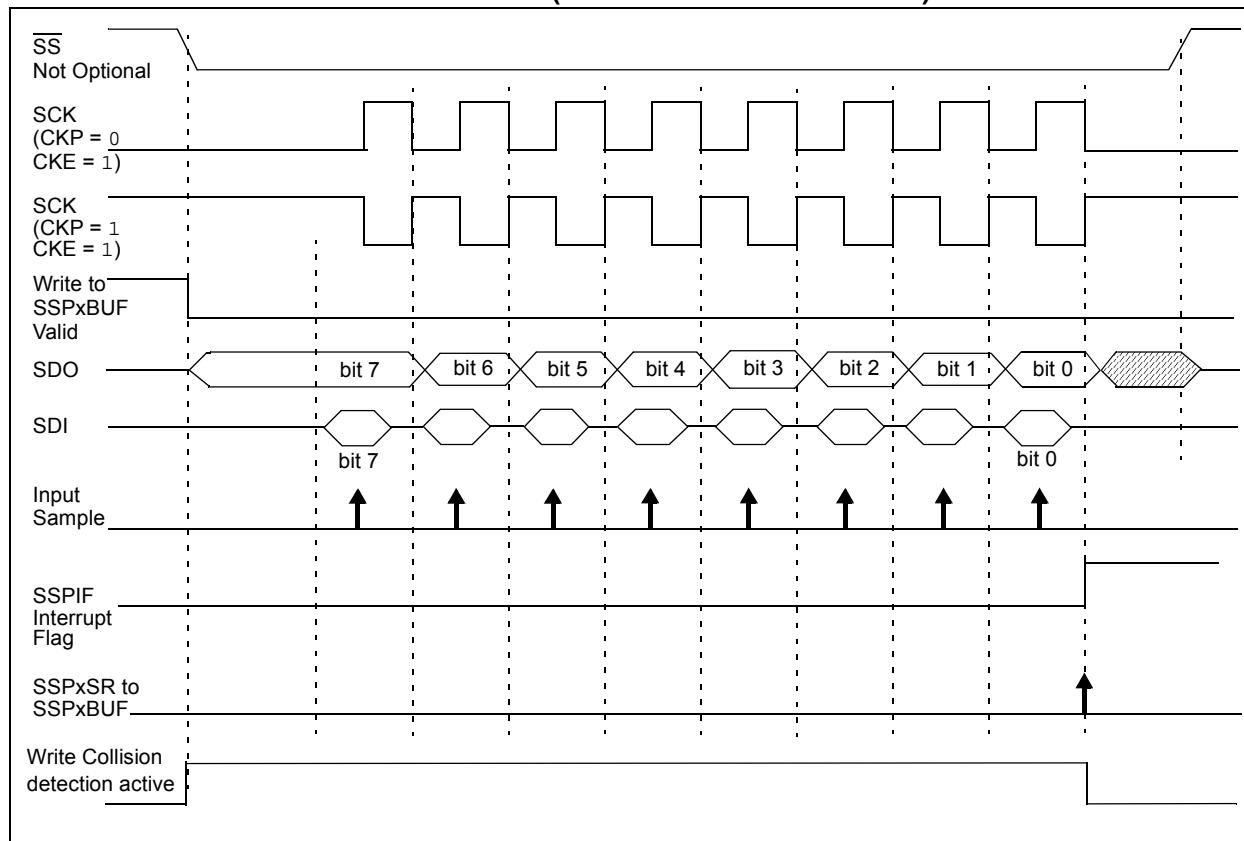


FIGURE 16-10: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



16.2.6 SPI OPERATION IN SLEEP MODE

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.

Special care must be taken by the user when the MSSP clock is much faster than the system clock.

In Slave mode, when MSSP interrupts are enabled, after the master completes sending data, an MSSP interrupt will wake the controller from Sleep.

If an exit from Sleep 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 Sleep 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.

TABLE 16-1: REGISTERS ASSOCIATED WITH SPI OPERATION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|---------------------------------------|-----------|--------|--------|-----------|--------|--------|--------|---------------------|
| ANSELA | — | — | ANSA5 | — | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 147 |
| ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 148 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| PMD1 | — | MSSPMMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | 62 |
| SSP1BUF | SSP1 Receive Buffer/Transmit Register | | | | | | | | — |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 252 |
| SSP1CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 255 |
| SSP1STAT | SMP | CKE | D/A | P | S | R/W | UA | BF | 251 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 149 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 149 |

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

16.3 I²C Mode Overview

The Inter-Integrated Circuit Bus (I²C) is a multi-master serial data communication bus. Devices communicate in a master/slave environment where the master devices initiate the communication. A slave device is controlled through addressing.

The I²C bus specifies two signal connections:

- Serial Clock (SCL)
- Serial Data (SDA)

[Figure 16-11](#) shows the block diagram of the MSSP module when operating in I²C mode.

Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors for the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one.

[Figure 16-11](#) shows a typical connection between two processors configured as master and slave devices.

The I²C bus can operate with one or more master devices and one or more slave devices.

There are four potential modes of operation for a given device:

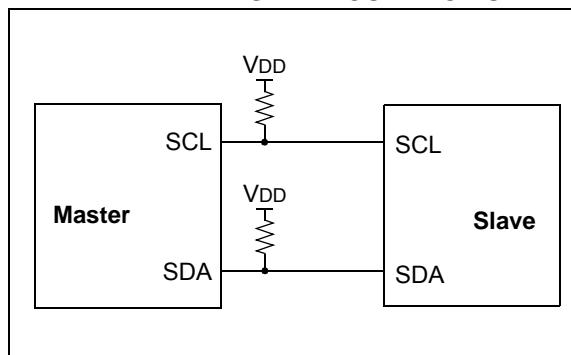
- Master Transmit mode
(master is transmitting data to a slave)
- Master Receive mode
(master is receiving data from a slave)
- Slave Transmit mode
(slave is transmitting data to a master)
- Slave Receive mode
(slave is receiving data from the master)

To begin communication, a master device starts out in Master Transmit mode. The master device sends out a Start bit followed by the address byte of the slave it intends to communicate with. This is followed by a single Read/Write bit, which determines whether the master intends to transmit to or receive data from the slave device.

If the requested slave exists on the bus, it will respond with an Acknowledge bit, otherwise known as an ACK. The master then continues in either Transmit mode or Receive mode and the slave continues in the complement, either in Receive mode or Transmit mode, respectively.

A Start bit is indicated by a high-to-low transition of the SDA line while the SCL line is held high. Address and data bytes are sent out, Most Significant bit (MSB) first. The Read/Write bit is sent out as a logical one when the master intends to read data from the slave, and is sent out as a logical zero when it intends to write data to the slave.

FIGURE 16-11: I²C™ MASTER/SLAVE CONNECTION



The Acknowledge bit (ACK) is an active-low signal, which holds the SDA line low to indicate to the transmitter that the slave device has received the transmitted data and is ready to receive more.

The transition of data bits is always performed while the SCL line is held low. Transitions that occur while the SCL line is held high are used to indicate Start and Stop bits.

If the master intends to write to the slave, then it repeatedly sends out a byte of data, with the slave responding after each byte with an ACK bit. In this example, the master device is in Master Transmit mode and the slave is in Slave Receive mode.

If the master intends to read from the slave, then it repeatedly receives a byte of data from the slave, and responds after each byte with an ACK bit. In this example, the master device is in Master Receive mode and the slave is Slave Transmit mode.

On the last byte of data communicated, the master device may end the transmission by sending a Stop bit. If the master device is in Receive mode, it sends the Stop bit in place of the last ACK bit. A Stop bit is indicated by a low-to-high transition of the SDA line while the SCL line is held high.

In some cases, the master may want to maintain control of the bus and re-initiate another transmission. If so, the master device may send another Start bit in place of the Stop bit or last ACK bit when it is in receive mode.

The I²C bus specifies three message protocols;

- Single message where a master writes data to a slave.
- Single message where a master reads data from a slave.
- Combined message where a master initiates a minimum of two writes, or two reads, or a combination of writes and reads, to one or more slaves.

When one device is transmitting a logical one, or letting the line float, and a second device is transmitting a logical zero, or holding the line low, the first device can detect that the line is not a logical one. This detection, when used on the SCL line, is called clock stretching. Clock stretching give slave devices a mechanism to control the flow of data. When this detection is used on the SDA line, it is called arbitration. Arbitration ensures that there is only one master device communicating at any single time.

16.3.1 CLOCK STRETCHING

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCL clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master that is communicating with the slave will attempt to raise the SCL line in order to transfer the next bit, but will detect that the clock line has not yet been released. Because the SCL connection is open-drain, the slave has the ability to hold that line low until it is ready to continue communicating.

Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

16.3.2 ARBITRATION

Each master device must monitor the bus for Start and Stop bits. If the device detects that the bus is busy, it cannot begin a new message until the bus returns to an Idle state.

However, two master devices may try to initiate a transmission on or about the same time. When this occurs, the process of arbitration begins. Each transmitter checks the level of the SDA data line and compares it to the level that it expects to find. The first transmitter to observe that the two levels do not match, loses arbitration, and must stop transmitting on the SDA line.

For example, if one transmitter holds the SDA line to a logical one (lets it float) and a second transmitter holds it to a logical zero (pulls it low), the result is that the SDA line will be low. The first transmitter then observes that the level of the line is different than expected and concludes that another transmitter is communicating.

The first transmitter to notice this difference is the one that loses arbitration and must stop driving the SDA line. If this transmitter is also a master device, it also must stop driving the SCL line. It then can monitor the lines for a Stop condition before trying to reissue its transmission. In the meantime, the other device that has not noticed any difference between the expected and actual levels on the SDA line continues with its original transmission. It can do so without any complications, because so far, the transmission appears exactly as expected with no other transmitter disturbing the message.

Slave Transmit mode can also be arbitrated, when a master addresses multiple slaves, but this is less common.

If two master devices are sending a message to two different slave devices at the address stage, the master sending the lower slave address always wins arbitration. When two master devices send messages to the same slave address, and addresses can sometimes refer to multiple slaves, the arbitration process must continue into the data stage.

Arbitration usually occurs very rarely, but it is a necessary process for proper multi-master support.

16.4 I²C Mode Operation

All MSSP I²C communication is byte oriented and shifted out MSb first. Six SFR registers and two interrupt flags interface the module with the PIC microcontroller and user software. Two pins, SDA and SCL, are exercised by the module to communicate with other external I²C devices.

16.4.1 BYTE FORMAT

All communication in I²C is done in 9-bit segments. A byte is sent from a master to a slave or vice-versa, followed by an Acknowledge bit sent back. After the 8th falling edge of the SCL line, the device outputting data on the SDA changes that pin to an input and reads in an acknowledge value on the next clock pulse.

The clock signal, SCL, is provided by the master. Data is valid to change while the SCL signal is low, and sampled on the rising edge of the clock. Changes on the SDA line while the SCL line is high define special conditions on the bus, explained below.

16.4.2 DEFINITION OF I²C TERMINOLOGY

There is language and terminology in the description of I²C communication that have definitions specific to I²C. That word usage is defined below and may be used in the rest of this document without explanation. This table was adapted from the Phillips I²C specification.

16.4.3 SDA AND SCL PINS

Selection of any I²C mode with the SSPxEN bit set, forces the SCL and SDA pins to be open-drain. These pins should be set by the user to inputs by setting the appropriate TRIS bits.

Note: Data is tied to output zero when an I²C™ mode is enabled.

16.4.4 SDA HOLD TIME

The hold time of the SDA pin is selected by the SDAHT bit of the SSPxCON3 register. Hold time is the time SDA is held valid after the falling edge of SCL. Setting the SDAHT bit selects a longer 300 ns minimum hold time and may help on buses with large capacitance.

TABLE 16-2: I²C™ BUS TERMS

| TERM | Description |
|------------------|---|
| Transmitter | The device which shifts data out onto the bus. |
| Receiver | The device which shifts data in from the bus. |
| Master | The device that initiates a transfer, generates clock signals and terminates a transfer. |
| Slave | The device addressed by the master. |
| Multi-master | A bus with more than one device that can initiate data transfers. |
| Arbitration | Procedure to ensure that only one master at a time controls the bus. Winning arbitration ensures that the message is not corrupted. |
| Synchronization | Procedure to synchronize the clocks of two or more devices on the bus. |
| Idle | No master is controlling the bus, and both SDA and SCL lines are high. |
| Active | Any time one or more master devices are controlling the bus. |
| Addressed Slave | Slave device that has received a matching address and is actively being clocked by a master. |
| Matching Address | Address byte that is clocked into a slave that matches the value stored in SSPxADD. |
| Write Request | Slave receives a matching address with R/W bit clear, and is ready to clock in data. |
| Read Request | Master sends an address byte with the R/W bit set, indicating that it wishes to clock data out of the Slave. This data is the next and all following bytes until a Restart or Stop. |
| Clock Stretching | When a device on the bus holds SCL low to stall communication. |
| Bus Collision | Any time the SDA line is sampled low by the module while it is outputting and expected high state. |

16.4.5 START CONDITION

The I²C specification defines a Start condition as a transition of SDA from a high-to-low state while SCL line is high. A Start condition is always generated by the master and signifies the transition of the bus from an Idle to an active state. [Figure 16-10](#) shows waveforms for Start and Stop conditions.

A bus collision can occur on a Start condition if the module samples the SDA line low before asserting it low. This does not conform to the I²C specification that states no bus collision can occur on a Start.

16.4.6 STOP CONDITION

A Stop condition is a transition of the SDA line from a low-to-high state while the SCL line is high.

Note: At least one SCL low time must appear before a Stop is valid, therefore, if the SDA line goes low then high again while the SCL line stays high, only the Start condition is detected.

16.4.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave.

In 10-bit Addressing Slave mode a Restart is required for the master to clock data out of the addressed slave. Once a slave has been fully addressed, matching both high and low address bytes, the master can issue a Restart and the high address byte with the R/W bit set. The slave logic will then hold the clock and prepare to clock out data.

After a full match with R/W clear in 10-bit mode, a prior match flag is set and maintained. Until a Stop condition, a high address with R/W clear, or high address match fails.

16.4.8 START/STOP CONDITION INTERRUPT MASKING

The SCIE and PCIE bits of the SSPxCON3 register can enable the generation of an interrupt in Slave modes that do not typically support this function. Slave modes where interrupt on Start and Stop detect are already enabled, these bits will have no effect.

FIGURE 16-12: I²C™ START AND STOP CONDITIONS

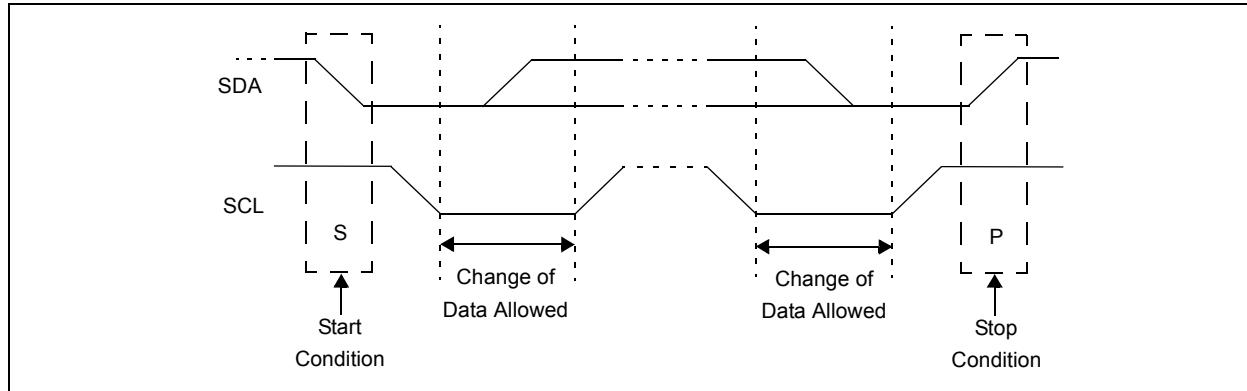
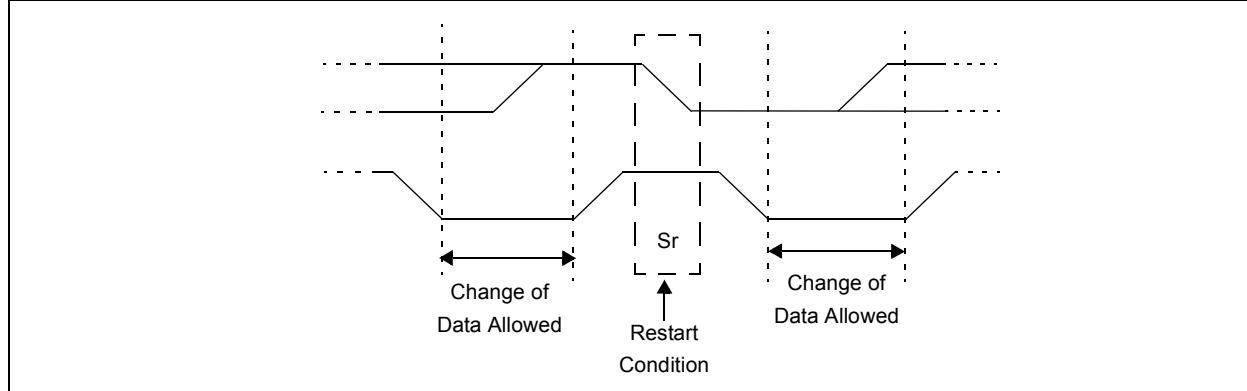


FIGURE 16-13: I²C™ RESTART CONDITION



16.4.9 ACKNOWLEDGE SEQUENCE

The 9th SCL pulse for any transferred byte in I²C is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDA line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge (ACK) is an active-low signal, pulling the SDA line low indicated to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an ACK is placed in the ACKSTAT bit of the SSPxCON2 register.

Slave software, when the AHEN and DHEN bits are set, allow the user to set the ACK value sent back to the transmitter. The ACKDT bit of the SSPxCON2 register is set/cleared to determine the response.

Slave hardware will generate an ACK response if the AHEN and DHEN bits of the SSPxCON3 register are clear.

There are certain conditions where an ACK will not be sent by the slave. If the BF bit of the SSPxSTAT register or the SSPOV bit of the SSPxCON1 register are set when a byte is received.

When the module is addressed, after the 8th falling edge of SCL on the bus, the ACKTIM bit of the SSPxCON3 register is set. The ACKTIM bit indicates the acknowledge time of the active bus.

The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is enabled.

16.5 I²C Slave Mode Operation

The MSSP Slave mode operates in one of four modes selected in the SSPxM bits of SSPxCON1 register. The modes can be divided into 7-bit and 10-bit Addressing mode. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

Modes with Start and Stop bit interrupts operated the same as the other modes with SSPIF additionally getting set upon detection of a Start, Restart, or Stop condition.

16.5.1 SLAVE MODE ADDRESSES

The SSPxADD register ([Register 16-6](#)) contains the Slave mode address. The first byte received after a Start or Restart condition is compared against the value stored in this register. If the byte matches, the value is loaded into the SSPxBUF register and an interrupt is generated. If the value does not match, the module goes Idle and no indication is given to the software that anything happened.

The SSPx Mask register ([Register 16-5](#)) affects the address matching process. See [Section 16.5.9 "SSPx Mask Register"](#) for more information.

16.5.1.1 I²C Slave 7-Bit Addressing Mode

In 7-bit Addressing mode, the LSb of the received data byte is ignored when determining if there is an address match.

16.5.1.2 I²C Slave 10-Bit Addressing Mode

In 10-bit Addressing mode, the first received byte is compared to the binary value of '1 1 1 1 0 A9 A8 0'. A9 and A8 are the two MSb of the 10-bit address and stored in bits 2 and 1 of the SSPxADD register.

After the acknowledge of the high byte the UA bit is set and SCL is held low until the user updates SSPxADD with the low address. The low address byte is clocked in and all eight bits are compared to the low address value in SSPxADD. Even if there is not an address match; SSPIF and UA are set, and SCL is held low until SSPxADD is updated to receive a high byte again. When SSPxADD is updated the UA bit is cleared. This ensures the module is ready to receive the high address byte on the next communication.

A high and low address match as a write request is required at the start of all 10-bit addressing communication. A transmission can be initiated by issuing a Restart once the slave is addressed, and clocking in the high address with the R/W bit set. The slave hardware will then acknowledge the read request and prepare to clock out data. This is only valid for a slave after it has received a complete high and low address byte match.

16.5.2 SLAVE RECEPTION

When the R/W bit of a matching received address byte is clear, the R/W bit of the SSPxSTAT register is cleared. The received address is loaded into the SSPxBUF register and acknowledged.

When the overflow condition exists for a received address, then no Acknowledge is given. An overflow condition is defined as either bit BF of the SSPxSTAT register is set, or bit SSPOV of the SSPxCON1 register is set. The BOEN bit of the SSPxCON3 register modifies this operation. For more information see [Register 16-4](#).

An MSSP interrupt is generated for each transferred data byte. Flag bit, SSPIF, must be cleared by software.

When the SEN bit of the SSPxCON2 register is set, SCL will be held low (clock stretch) following each received byte. The clock must be released by setting the CKP bit of the SSPxCON1 register, except sometimes in 10-bit mode. See [Section 16.2.3 "SPI Master Mode"](#) for more detail.

16.5.2.1 7-Bit Addressing Reception

This section describes a standard sequence of events for the MSSP module configured as an I²C slave in 7-bit Addressing mode. All decisions made by hardware or software and their effect on reception. [Figure 16-13](#) and [Figure 16-14](#) is used as a visual reference for this description.

This is a step by step process of what typically must be done to accomplish I²C communication.

1. Start bit detected.
2. S bit of SSPxSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Matching address with R/W bit clear is received.
4. The slave pulls SDA low sending an ACK to the master, and sets SSPIF bit.
5. Software clears the SSPIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. If SEN = 1; Slave software sets CKP bit to release the SCL line.
8. The master clocks out a data byte.
9. Slave drives SDA low sending an ACK to the master, and sets SSPIF bit.
10. Software clears SSPIF.
11. Software reads the received byte from SSPxBUF clearing BF.
12. Steps 8-12 are repeated for all received bytes from the master.
13. Master sends Stop condition, setting P bit of SSPxSTAT, and the bus goes Idle.

16.5.2.2 7-Bit Reception with AHEN and DHEN

Slave device reception with AHEN and DHEN set operate the same as without these options with extra interrupts and clock stretching added after the 8th falling edge of SCL. These additional interrupts allow the slave software to decide whether it wants to ACK the receive address or data byte, rather than the hardware. This functionality adds support for PMBus™ that was not present on previous versions of this module.

This list describes the steps that need to be taken by slave software to use these options for I²C communication. [Figure 16-15](#) displays a module using both address and data holding. [Figure 16-16](#) includes the operation with the SEN bit of the SSPxCON2 register set.

1. S bit of SSPxSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
2. Matching address with R/W bit clear is clocked in. SSPIF is set and CKP cleared after the 8th falling edge of SCL.
3. Slave clears the SSPIF.
4. Slave can look at the ACKTIM bit of the SSPxCON3 register to determine if the SSPIF was after or before the ACK.
5. Slave reads the address value from SSPxBUF, clearing the BF flag.
6. Slave sets ACK value clocked out to the master by setting ACKDT.
7. Slave releases the clock by setting CKP.
8. SSPIF is set after an ACK, not after a NACK.
9. If SEN = 1 the slave hardware will stretch the clock after the ACK.
10. Slave clears SSPIF.

Note: SSPIF is still set after the 9th falling edge of SCL even if there is no clock stretching and BF has been cleared. Only if NACK is sent to master is SSPIF not set.

11. SSPIF set and CKP cleared after 8th falling edge of SCL for a received data byte.
12. Slave looks at ACKTIM bit of SSPxCON3 to determine the source of the interrupt.
13. Slave reads the received data from SSPxBUF clearing BF.
14. Steps 7-14 are the same for each received data byte.
15. Communication is ended by either the slave sending an ACK = 1, or the master sending a Stop condition. If a Stop is sent and Interrupt on Stop detect is disabled, the slave will only know by polling the P bit of the SSTSTAT register.

PIC18(L)F2X/45K50

FIGURE 16-14: I²C™ SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 0, DHEN = 0)

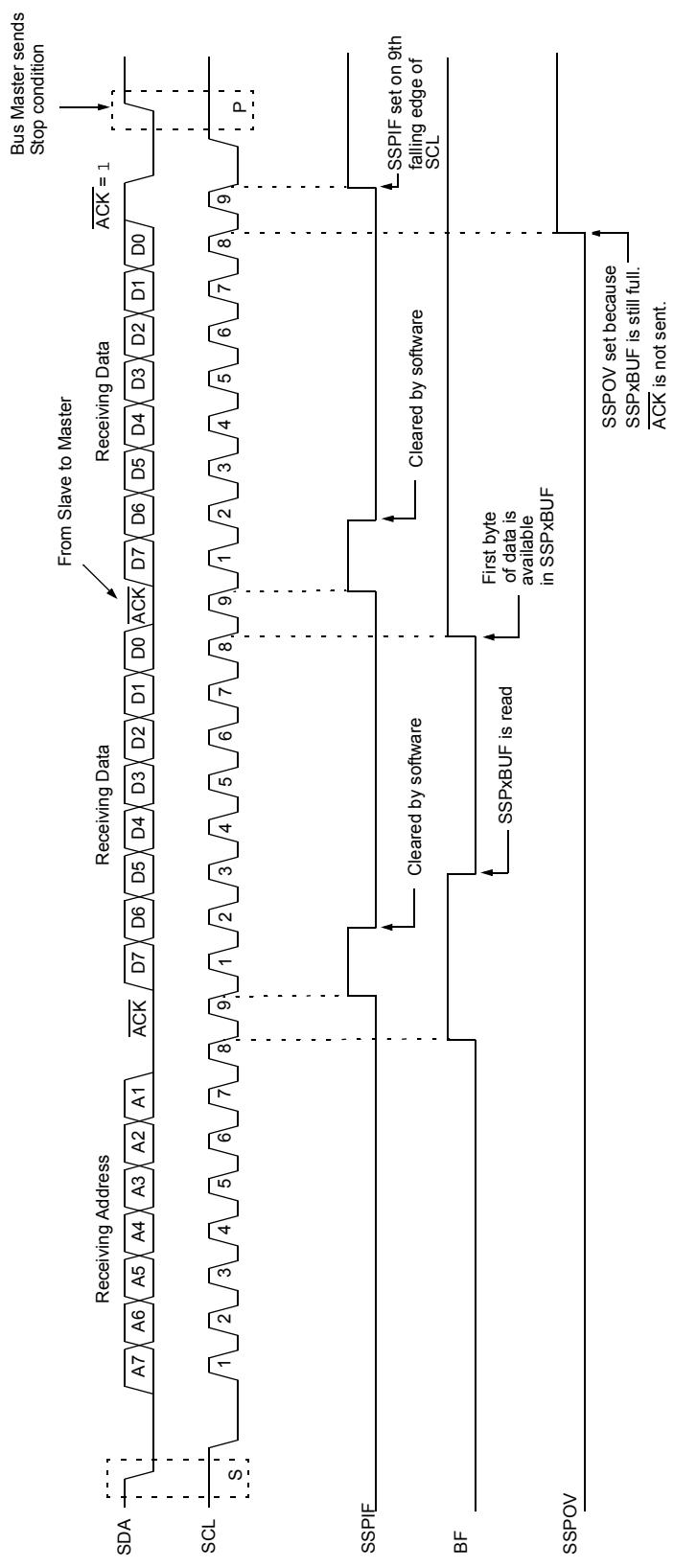
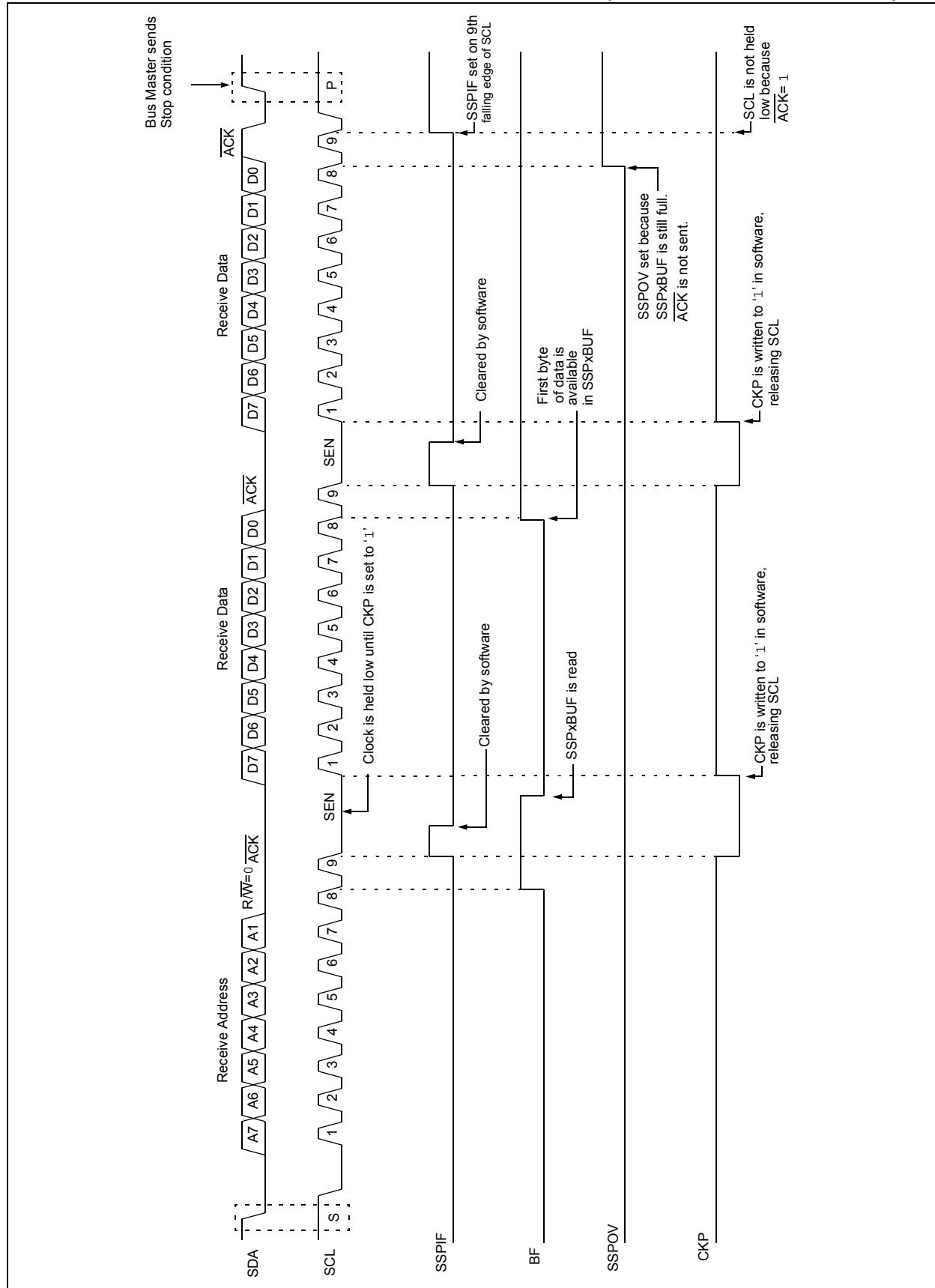


FIGURE 16-15: I²C™ SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)



PIC18(L)F2X/45K50

FIGURE 16-16: I²C™ SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)

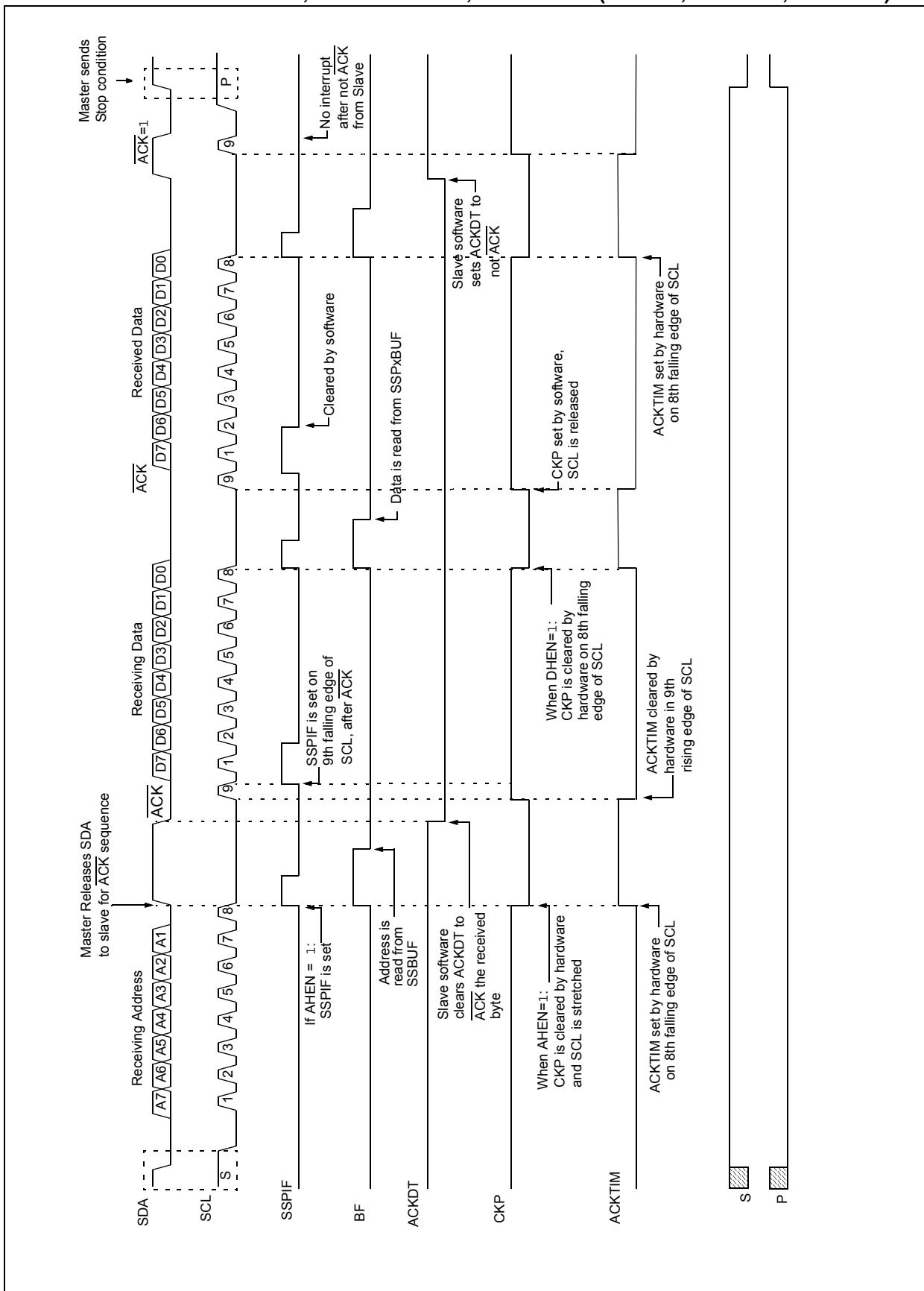
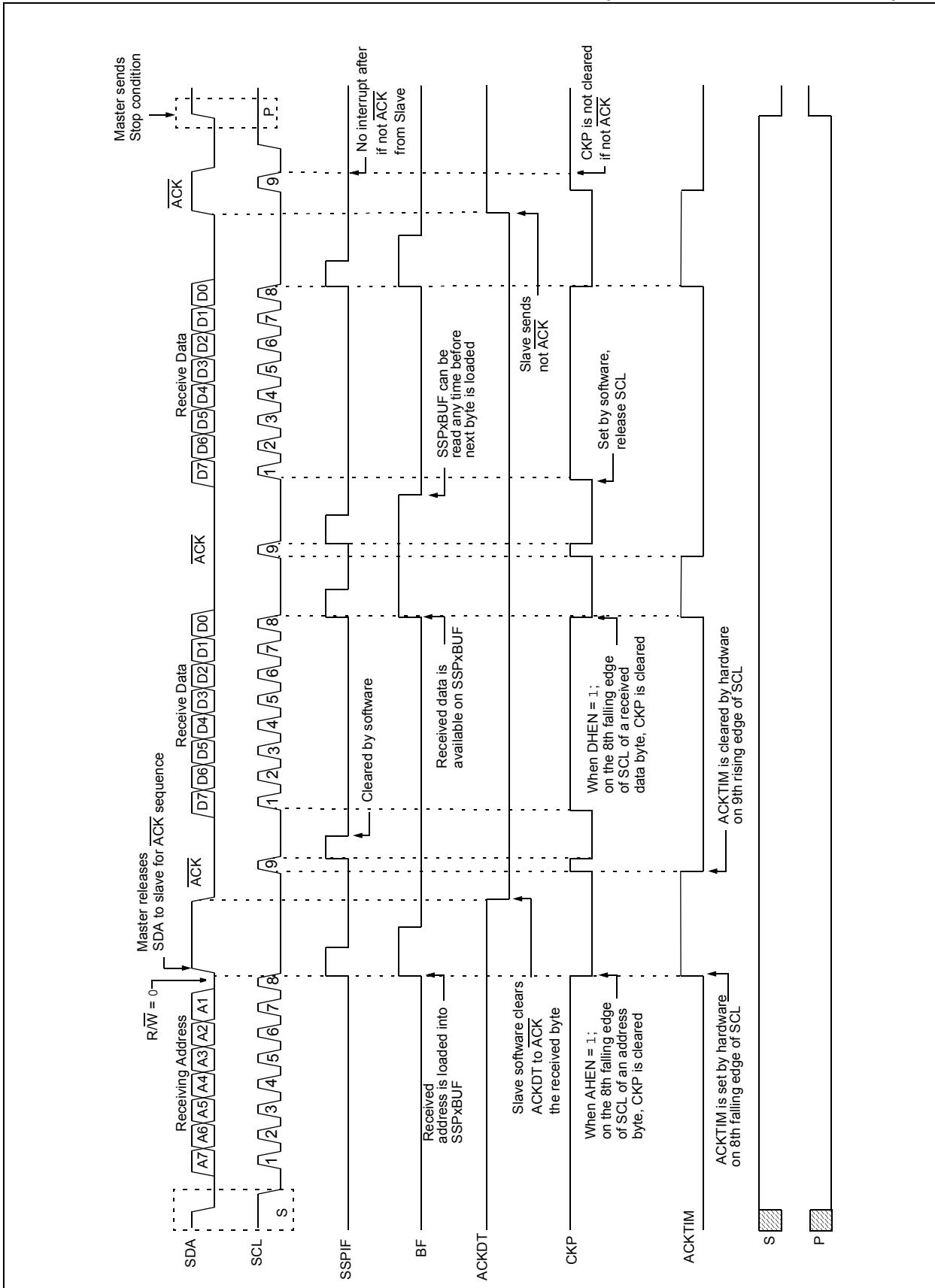


FIGURE 16-17: I²C™ SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 1, DHEN = 1)



16.5.3 SLAVE TRANSMISSION

When the R/W bit of the incoming address byte is set and an address match occurs, the R/W bit of the SSPxSTAT register is set. The received address is loaded into the SSPxBUF register, and an ACK pulse is sent by the slave on the ninth bit.

Following the ACK, slave hardware clears the CKP bit and the SCL pin is held low (see [Section 16.5.6 "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 SSPxBUF register which also loads the SSPxSR register. Then the SCL pin should be released by setting the CKP bit of the SSPxCON1 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.

The ACK pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. This ACK value is copied to the ACKSTAT bit of the SSPxCON2 register. If ACKSTAT is set (not ACK), then the data transfer is complete. In this case, when the not ACK is latched by the slave, the slave goes Idle and waits for another occurrence of the Start bit. If the SDA line was low (ACK), the next transmit data must be loaded into the SSPxBUF register. Again, the SCL pin 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 SSPxSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

16.5.3.1 Slave Mode Bus Collision

A slave receives a Read request and begins shifting data out on the SDA line. If a bus collision is detected and the SBCDE bit of the SSPxCON3 register is set, the BCLIF bit of the PIRx register is set. Once a bus collision is detected, the slave goes Idle and waits to be addressed again. User software can use the BCLIF bit to handle a slave bus collision.

16.5.3.2 7-Bit Transmission

A master device can transmit a read request to a slave, and then clock data out of the slave. The list below outlines what software for a slave will need to do to accomplish a standard transmission. [Figure 16-17](#) can be used as a reference to this list.

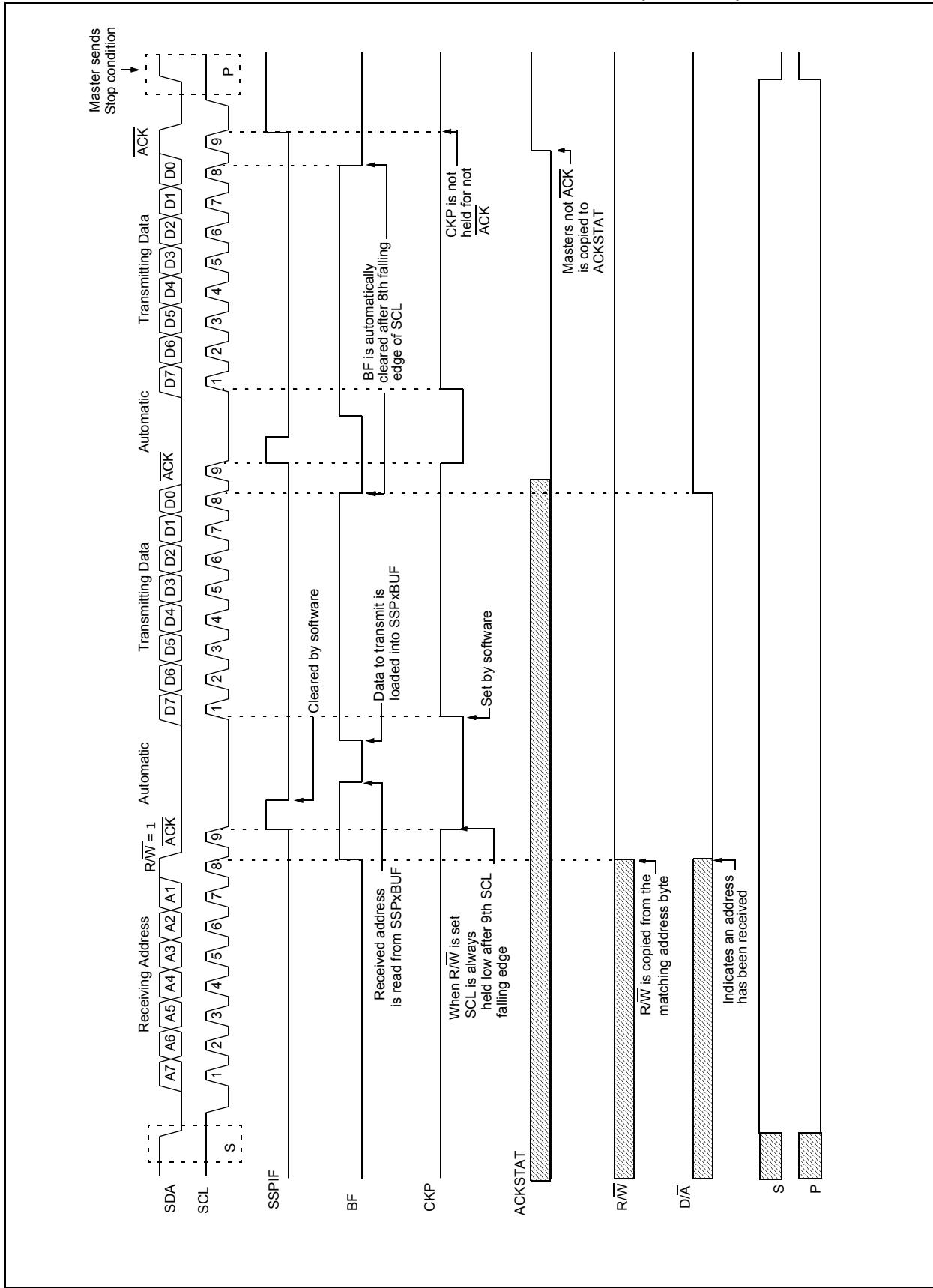
1. Master sends a Start condition on SDA and SCL.
2. S bit of SSPxSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Matching address with R/W bit set is received by the slave setting SSPIF bit.
4. Slave hardware generates an ACK and sets SSPIF.
5. SSPIF bit is cleared by user.
6. Software reads the received address from SSPxBUF, clearing BF.
7. R/W is set so CKP was automatically cleared after the ACK.
8. The slave software loads the transmit data into SSPxBUF.
9. CKP bit is set releasing SCL, allowing the master to clock the data out of the slave.
10. SSPIF is set after the ACK response from the master is loaded into the ACKSTAT register.
11. SSPIF bit is cleared.
12. The slave software checks the ACKSTAT bit to see if the master wants to clock out more data.

Note 1: If the master ACKs the clock will be stretched.

2: ACKSTAT is the only bit updated on the rising edge of SCL (9th) rather than the falling.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not ACK; the clock is not held, but SSPIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.

FIGURE 16-18: I²C™ SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 0)



PIC18(L)F2X/45K50

16.5.3.3 7-Bit Transmission with Address Hold Enabled

Setting the AHEN bit of the SSPxCON3 register enables additional clock stretching and interrupt generation after the 8th falling edge of a received matching address. Once a matching address has been clocked in, CKP is cleared and the SSPIF interrupt is set.

Figure 16-18 displays a standard waveform of a 7-bit Address Slave Transmission with AHEN enabled.

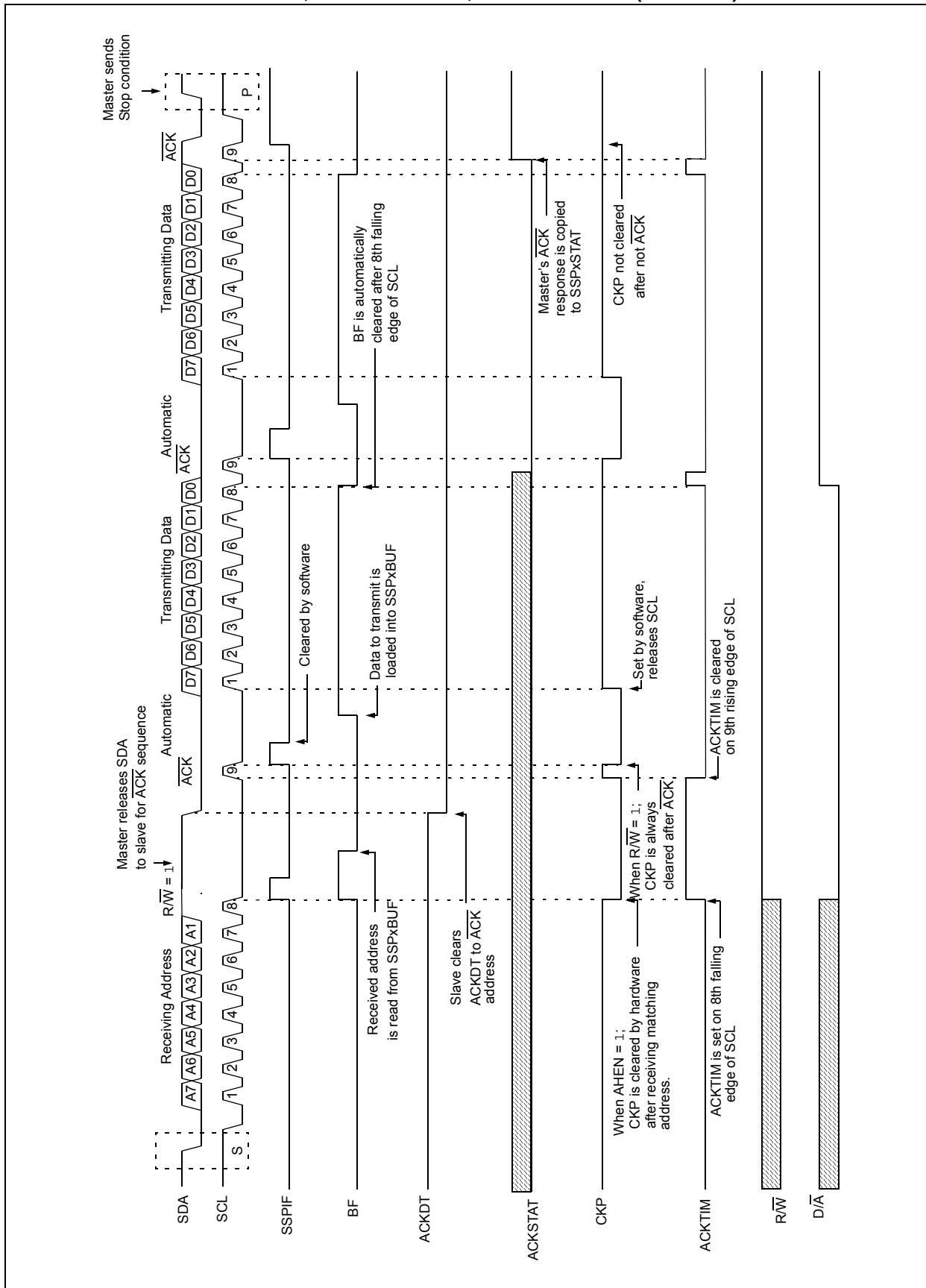
1. Bus starts Idle.
2. Master sends Start condition; the S bit of SSPxSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Master sends matching address with R/W bit set. After the 8th falling edge of the SCL line the CKP bit is cleared and SSPIF interrupt is generated.
4. Slave software clears SSPIF.
5. Slave software reads ACKTIM bit of SSPxCON3 register, and R/W and D/A of the SSPxSTAT register to determine the source of the interrupt.
6. Slave reads the address value from the SSPxBUF register clearing the BF bit.
7. Slave software decides from this information if it wishes to ACK or not ACK and sets ACKDT bit of the SSPxCON2 register accordingly.
8. Slave sets the CKP bit releasing SCL.
9. Master clocks in the ACK value from the slave.
10. Slave hardware automatically clears the CKP bit and sets SSPIF after the ACK if the R/W bit is set.
11. Slave software clears SSPIF.
12. Slave loads value to transmit to the master into SSPxBUF setting the BF bit.

Note: SSPxBUF cannot be loaded until after the ACK.

13. Slave sets CKP bit releasing the clock.
14. Master clocks out the data from the slave and sends an ACK value on the 9th SCL pulse.
15. Slave hardware copies the ACK value into the ACKSTAT bit of the SSPxCON2 register.
16. Steps 10-15 are repeated for each byte transmitted to the master from the slave.
17. If the master sends a not ACK the slave releases the bus allowing the master to send a Stop and end the communication.

Note: Master must send a not ACK on the last byte to ensure that the slave releases the SCL line to receive a Stop.

FIGURE 16-19: I²CTM SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)



16.5.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSP module configured as an I²C slave in 10-bit Addressing mode.

[Figure 16-19](#) and is used as a visual reference for this description.

This is a step by step process of what must be done by slave software to accomplish I²C communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit of SSPxSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Master sends matching high address with R/W bit clear; UA bit of the SSPxSTAT register is set.
4. Slave sends ACK and SSPIF is set.
5. Software clears the SSPIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. Slave loads low address into SSPxADD, releasing SCL.
8. Master sends matching low address byte to the slave; UA bit is set.

Note: Updates to the SSPxADD register are not allowed until after the ACK sequence.

9. Slave sends ACK and SSPIF is set.

Note: If the low address does not match, SSPIF and UA are still set so that the slave software can set SSPxADD back to the high address. BF is not set because there is no match. CKP is unaffected.

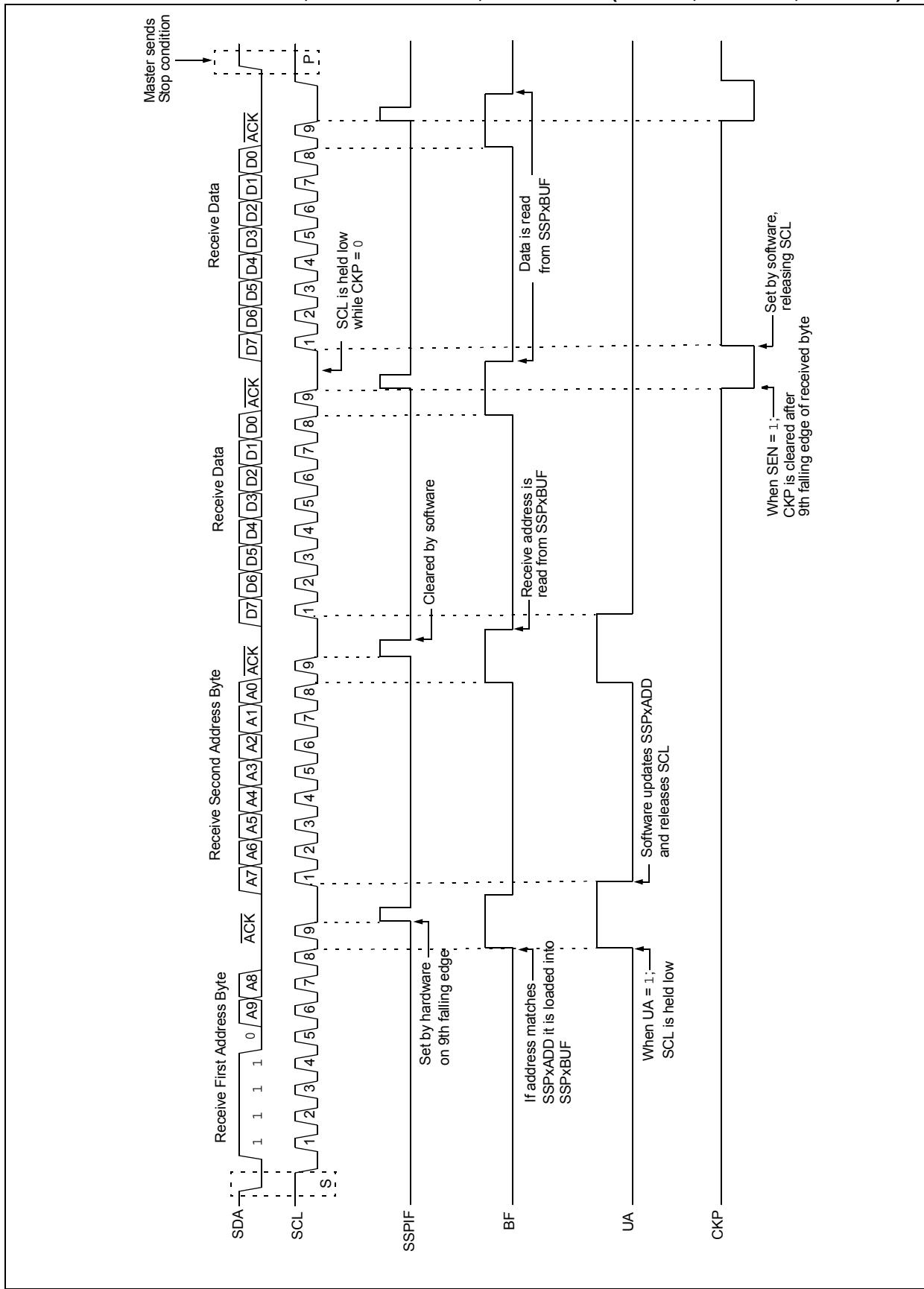
10. Slave clears SSPIF.
11. Slave reads the received matching address from SSPxBUF clearing BF.
12. Slave loads high address into SSPxADD.
13. Master clocks a data byte to the slave and clocks out the slaves ACK on the 9th SCL pulse; SSPIF is set.
14. If SEN bit of SSPxCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPIF.
16. Slave reads the received byte from SSPxBUF clearing BF.
17. If SEN is set the slave sets CKP to release the SCL.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

16.5.5 10-BIT ADDRESSING WITH ADDRESS OR DATA HOLD

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPxADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCL line is held low are the same. [Figure 16-20](#) can be used as a reference of a slave in 10-bit addressing with AHEN set.

[Figure 16-21](#) shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

FIGURE 16-20: I²C™ SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)



PIC18(L)F2X/45K50

FIGURE 16-21: I²C™ SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 0)

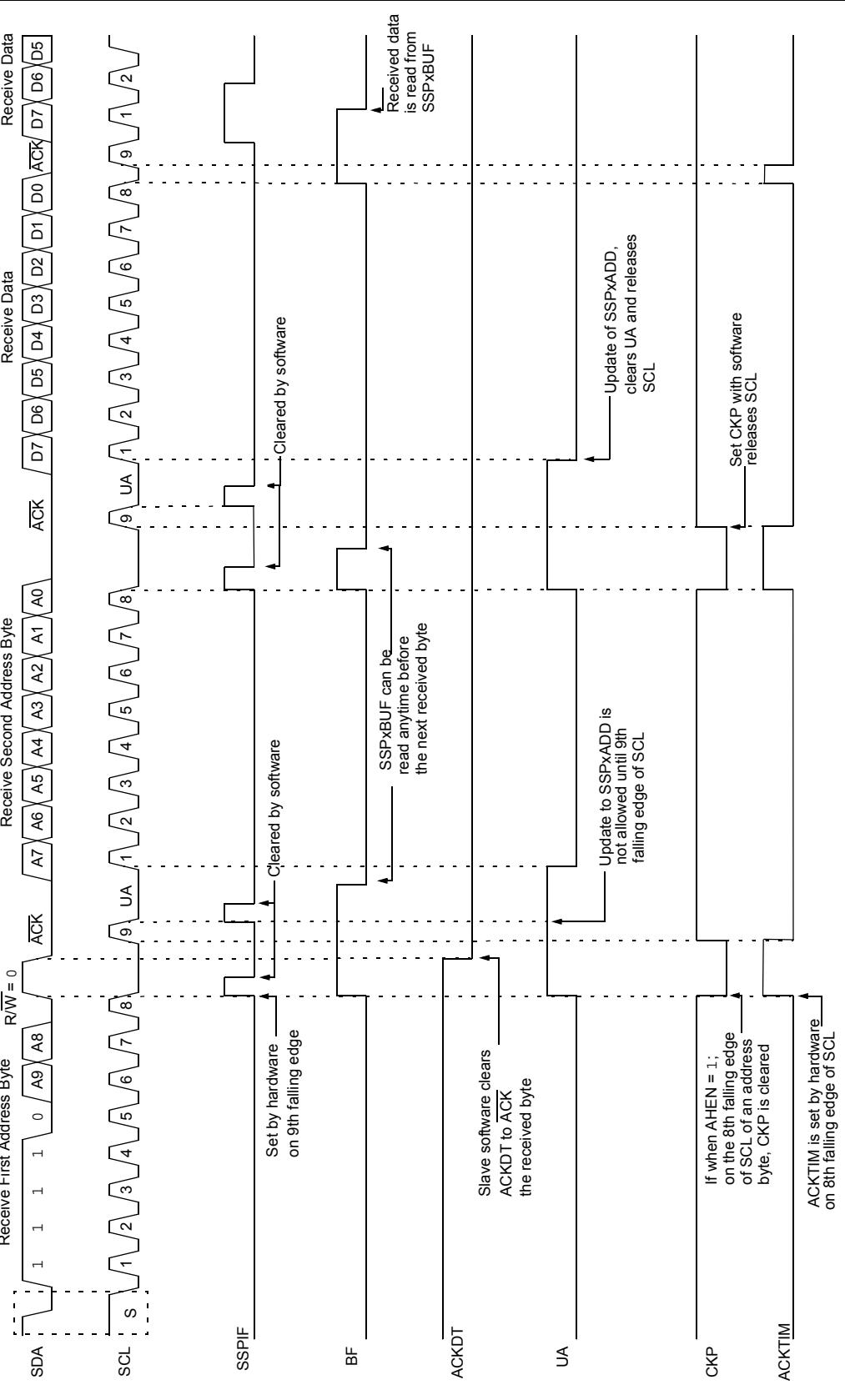
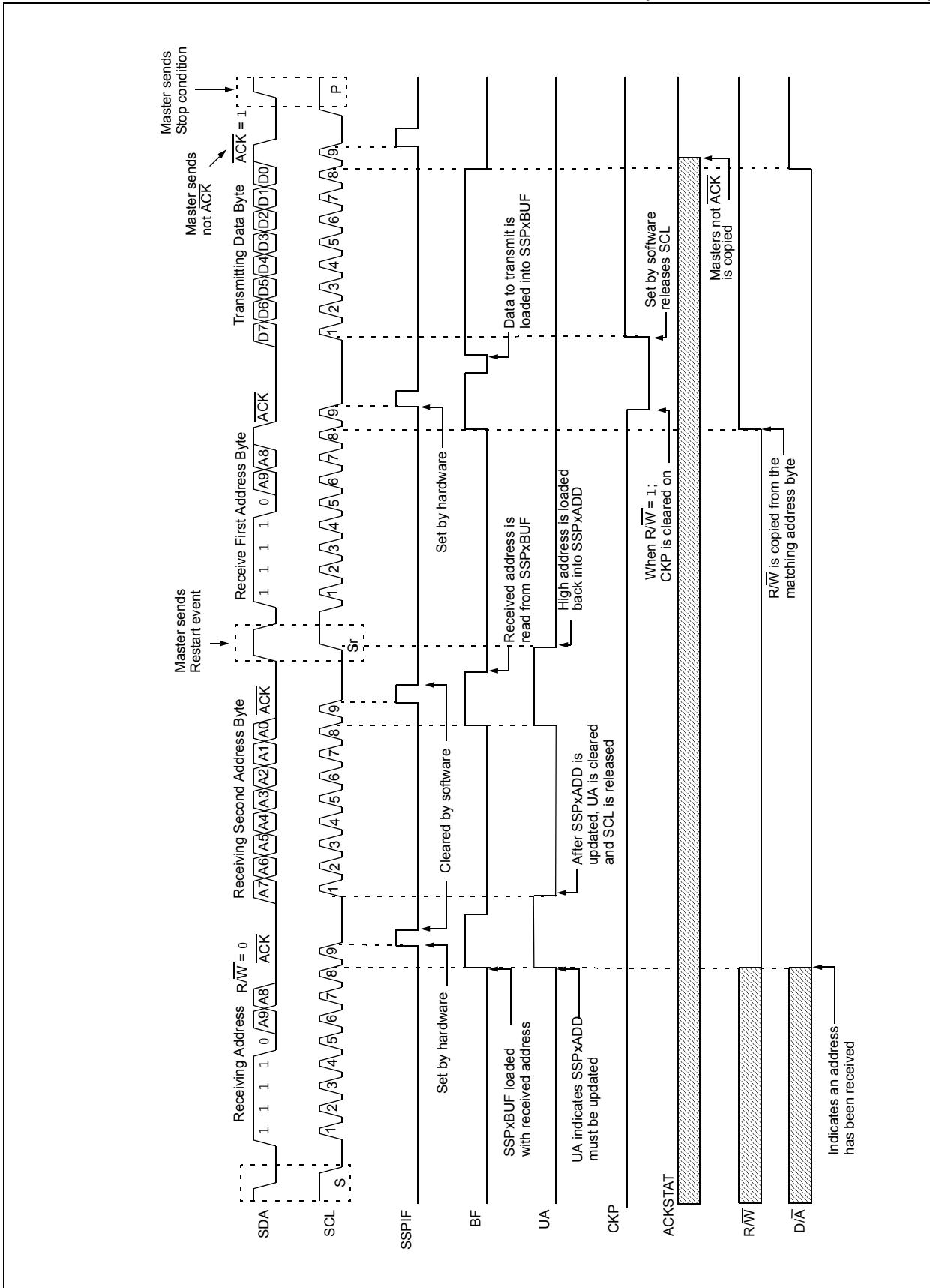


FIGURE 16-22: I²C™ SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)



16.5.6 CLOCK STRETCHING

Clock stretching occurs when a device on the bus holds the SCL line low effectively pausing communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. A master device is not concerned with stretching as anytime it is active on the bus and not transferring data it is stretching. Any stretching done by a slave is invisible to the master software and handled by the hardware that generates SCL.

The CKP bit of the SSPxCON1 register is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. Setting CKP will release SCL and allow more communication.

16.5.6.1 Normal Clock Stretching

Following an ACK if the R/W bit of SSPxSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPxBUF with data to transfer to the master. If the SEN bit of SSPxCON2 is set, the slave hardware will always stretch the clock after the ACK sequence. Once the slave is ready; CKP is set by software and communication resumes.

Note 1: The BF bit has no effect on whether the clock will be stretched or not. This is different than previous versions of the module that would not stretch the clock, clear CKP, if SSPxBUF was read before the 9th falling edge of SCL.

2: Previous versions of the module did not stretch the clock for a transmission if SSPxBUF was loaded before the 9th falling edge of SCL. It is now always cleared for read requests.

16.5.6.2 10-Bit Addressing Mode

In 10-bit Addressing mode, when the UA bit is set, the clock is always stretched. This is the only time the SCL is stretched without CKP being cleared. SCL is released immediately after a write to SSPxADD.

Note: Previous versions of the module did not stretch the clock if the second address byte did not match.

16.5.6.3 Byte NACKing

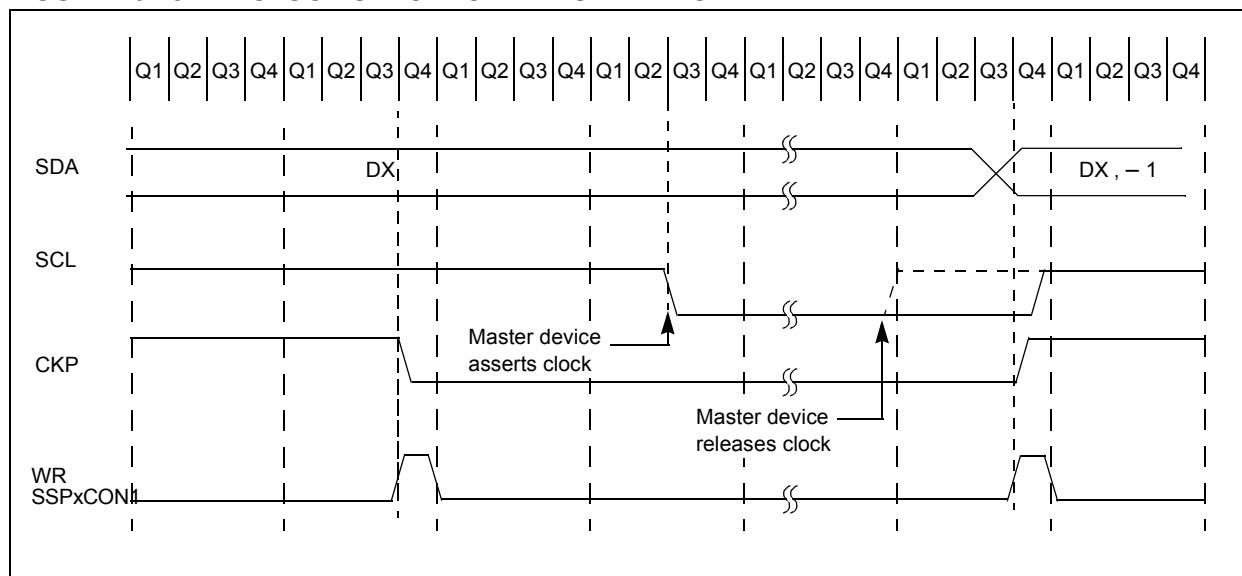
When the AHEN bit of SSPxCON3 is set; CKP is cleared by hardware after the 8th falling edge of SCL for a received matching address byte. When the DHEN bit of SSPxCON3 is set; CKP is cleared after the 8th falling edge of SCL for received data.

Stretching after the 8th falling edge of SCL allows the slave to look at the received address or data and decide if it wants to ACK the received data.

16.5.7 CLOCK SYNCHRONIZATION AND THE CKP BIT

Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. 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 I²C 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 I²C bus have released SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 16-22).

FIGURE 16-23: CLOCK SYNCHRONIZATION TIMING



16.5.8 GENERAL CALL ADDRESS SUPPORT

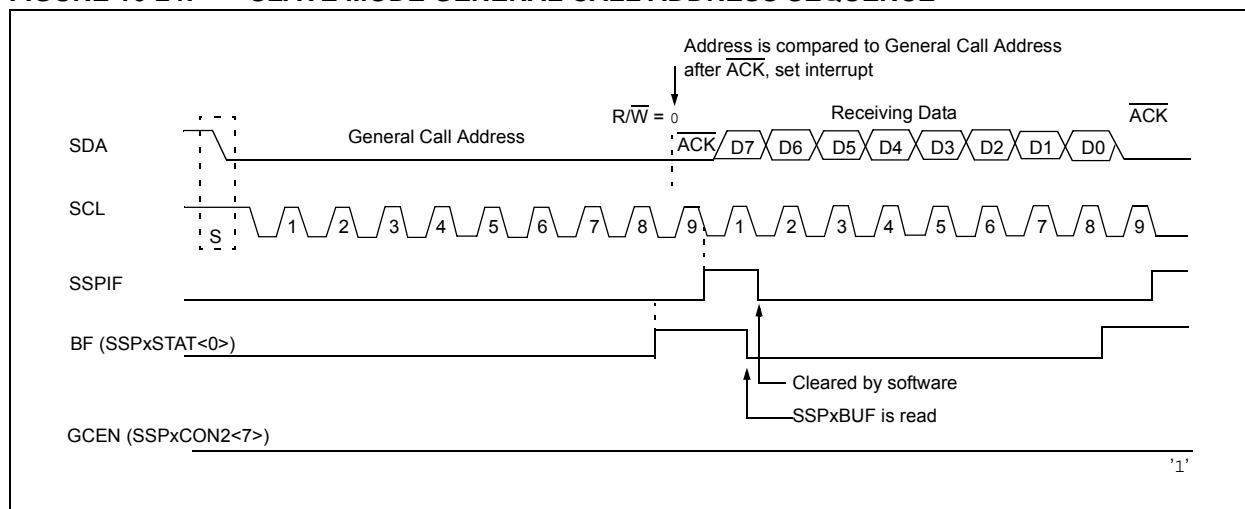
The addressing procedure for the I²C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master device. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge.

The general call address is a reserved address in the I²C protocol, defined as address 0x00. When the GCEN bit of the SSPxCON2 register is set, the slave module will automatically ACK the reception of this address regardless of the value stored in SSPxBUF. After the slave clocks in an address of all zeros with the R/W bit clear, an interrupt is generated and slave software can read SSPxBUF and respond. [Figure 16-23](#) shows a general call reception sequence.

In 10-bit Address mode, the UA bit will not be set on the reception of the general call address. The slave will prepare to receive the second byte as data, just as it would in 7-bit mode.

If the AHEN bit of the SSPxCON3 register is set, just as with any other address reception, the slave hardware will stretch the clock after the 8th falling edge of SCL. The slave must then set its ACKDT value and release the clock with communication progressing as it would normally.

FIGURE 16-24: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE



16.5.9 SSPx MASK REGISTER

An SSPx Mask (SSPxMSK) register ([Register 16-5](#)) is available in I²C Slave mode as a mask for the value held in the SSPxSR register during an address comparison operation. A zero ('0') bit in the SSPxMSK register has the effect of making the corresponding bit of the received address a "don't care".

This register is reset to all '1's upon any Reset condition and, therefore, has no effect on standard SSPx operation until written with a mask value.

The SSPx 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 SSPx mask has no effect during the reception of the first (high) byte of the address.

16.6 I²C Master Mode

Master mode is enabled by setting and clearing the appropriate SSPxM bits in the SSPxCON1 register and by setting the SSPxEN bit. In Master mode, the SCL and SDA lines are set as inputs and 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 I²C bus may be taken when the P bit is set, or the bus is Idle.

In Firmware Controlled Master mode, user code conducts all I²C bus operations based on Start and Stop bit condition detection. Start and Stop condition detection is the only active circuitry in this mode. All other communication is done by the user software directly manipulating the SDA and SCL lines.

The following events will cause the SSPx Interrupt Flag bit, SSPIF, to be set (SSPx interrupt, if enabled):

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

Note 1: The MSSP module, when configured in I²C Master mode, does not allow queuing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur

2: When in Master mode, Start/Stop detection is masked and an interrupt is generated when the SEN/PEN bit is cleared and the generation is complete.

16.6.1 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 (7 bits) and the Read/Write (R/W) bit. In this case, the R/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.

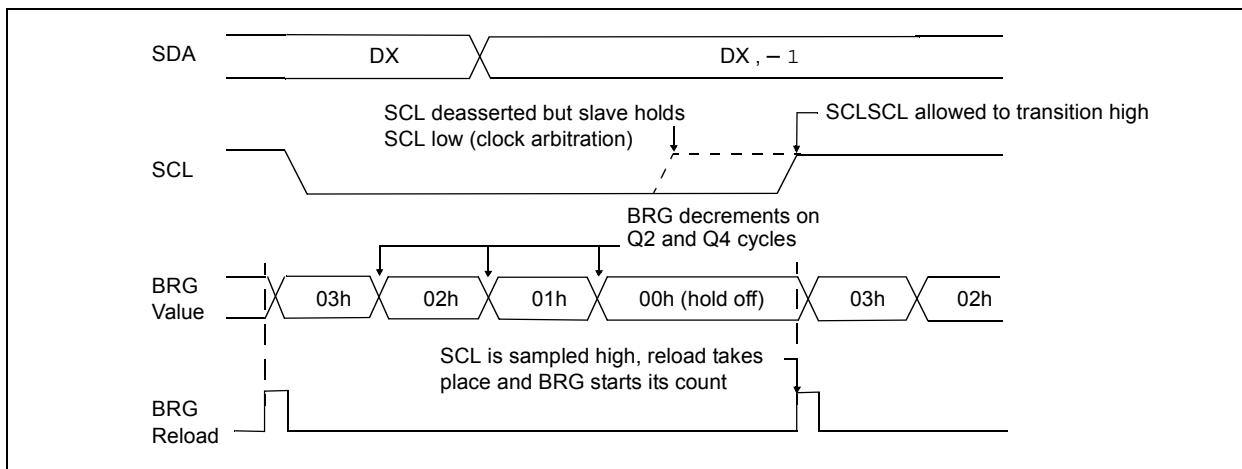
In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/W bit. In this case, the R/W bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received eight 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 16.7 "Baud Rate Generator"](#) for more detail.

16.6.2 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, releases 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 SSPxADD<7:0> 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 16-25).

FIGURE 16-25: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION



16.6.3 WCOL STATUS FLAG

If the user writes the SSPxBUF when a Start, Restart, Stop, Receive or Transmit sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write does not occur). Any time the WCOL bit is set it indicates that an action on SSPxBUF was attempted while the module was not Idle.

Note: Because queuing of events is not allowed, writing to the lower five bits of SSPxCON2 is disabled until the Start condition is complete.

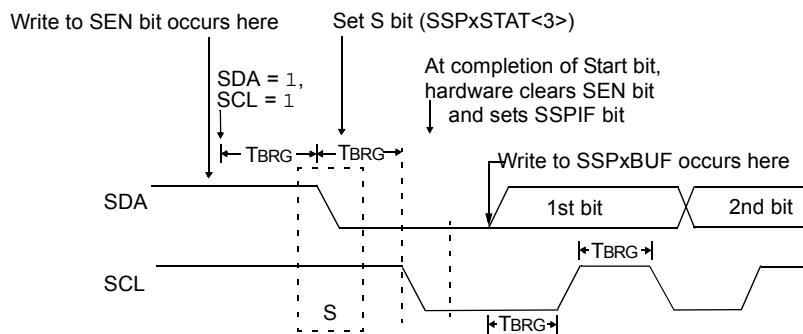
16.6.4 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Enable bit, SEN, of the SSPxCON2 register. If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> 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 SSPxSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit of the SSPxCON2 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 1: 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.

2: The Philips I²C Specification states that a bus collision cannot occur on a Start.

FIGURE 16-26: FIRST START BIT TIMING

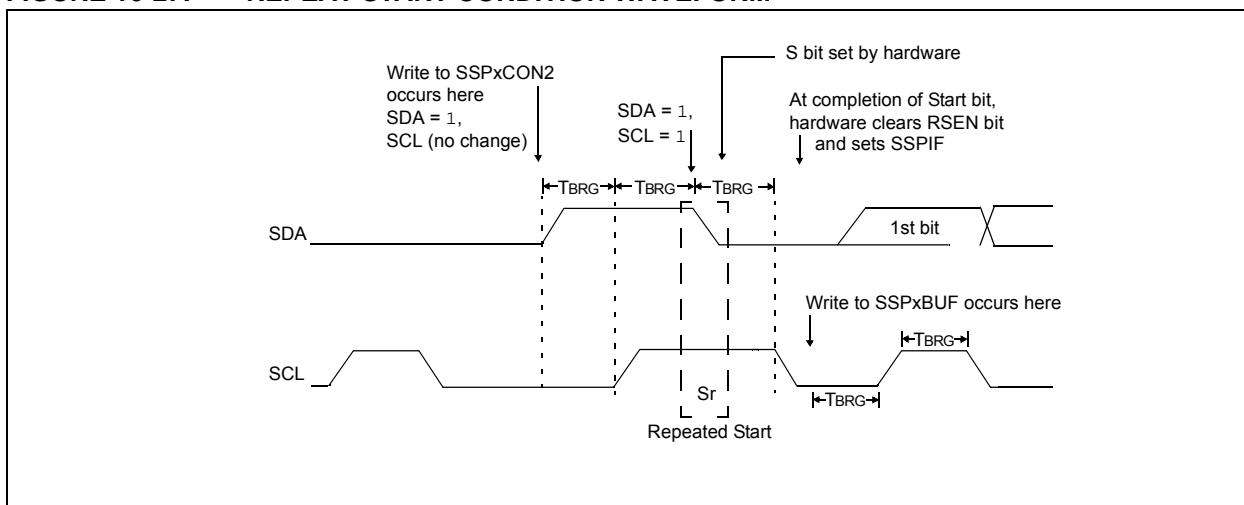


16.6.5 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit of the SSPxCON2 register is programmed high and the master state machine is no longer active. 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. SCL is asserted low. Following this, the RSEN bit of the SSPxCON2 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 SSPxSTAT 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'.

FIGURE 16-27: REPEAT START CONDITION WAVEFORM



16.6.6 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 SSPxBUF 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. SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high. 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 ACKSTAT bit on the rising 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 SSPxBUF, leaving SCL low and SDA unchanged ([Figure 16-27](#)).

After the write to the SSPxBUF, 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 release 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 SSPxCON2 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 SSPxBUF takes place, holding SCL low and allowing SDA to float.

16.6.6.1 BF Status Flag

In Transmit mode, the BF bit of the SSPxSTAT register is set when the CPU writes to SSPxBUF and is cleared when all eight bits are shifted out.

16.6.6.2 WCOL Status Flag

If the user writes the SSPxBUF when a transmit is already in progress (i.e., SSPxSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write does not occur).

WCOL must be cleared by software before the next transmission.

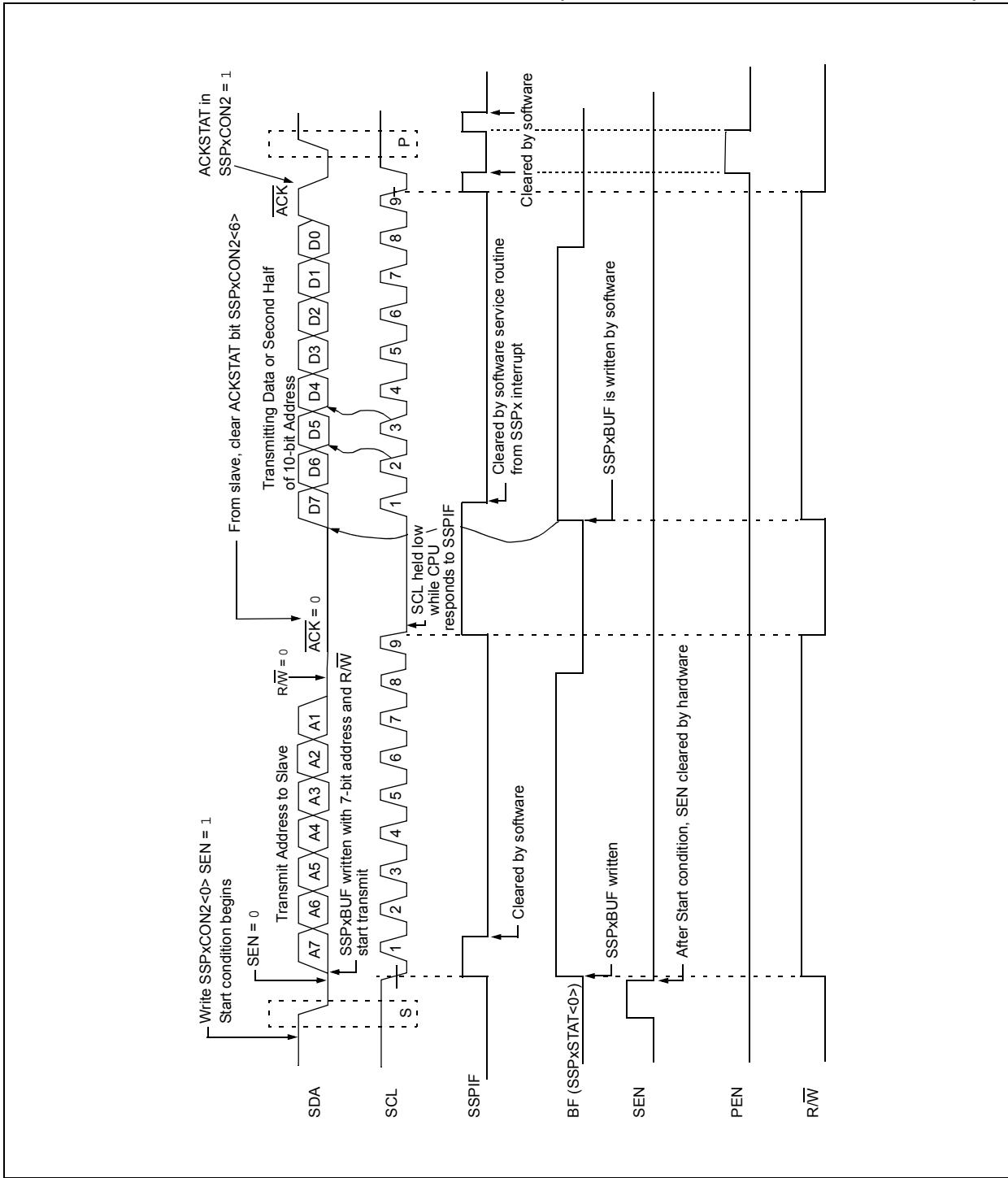
16.6.6.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit of the SSPxCON2 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.

16.6.6.4 Typical Transmit Sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPxCON2 register.
2. SSPIF is set by hardware on completion of the Start.
3. SSPIF is cleared by software.
4. The MSSP module will wait the required start time before any other operation takes place.
5. The user loads the SSPxBUF with the slave address to transmit.
6. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
7. The MSSP module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
8. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
9. The user loads the SSPxBUF with eight bits of data.
10. Data is shifted out the SDA pin until all eight bits are transmitted.
11. The MSSP module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
12. Steps 8-11 are repeated for all transmitted data bytes.
13. The user generates a Stop or Restart condition by setting the PEN or RSEN bits of the SSPxCON2 register. Interrupt is generated once the Stop/Restart condition is complete.

FIGURE 16-28: I²CTM MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)



16.6.7 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN, of the SSPxCON2 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 SSPxSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPxSR are loaded into the SSPxBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN, of the SSPxCON2 register.

16.6.7.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPxBUF from SSPxSR. It is cleared when the SSPxBUF register is read.

16.6.7.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when eight bits are received into the SSPxSR and the BF flag bit is already set from a previous reception.

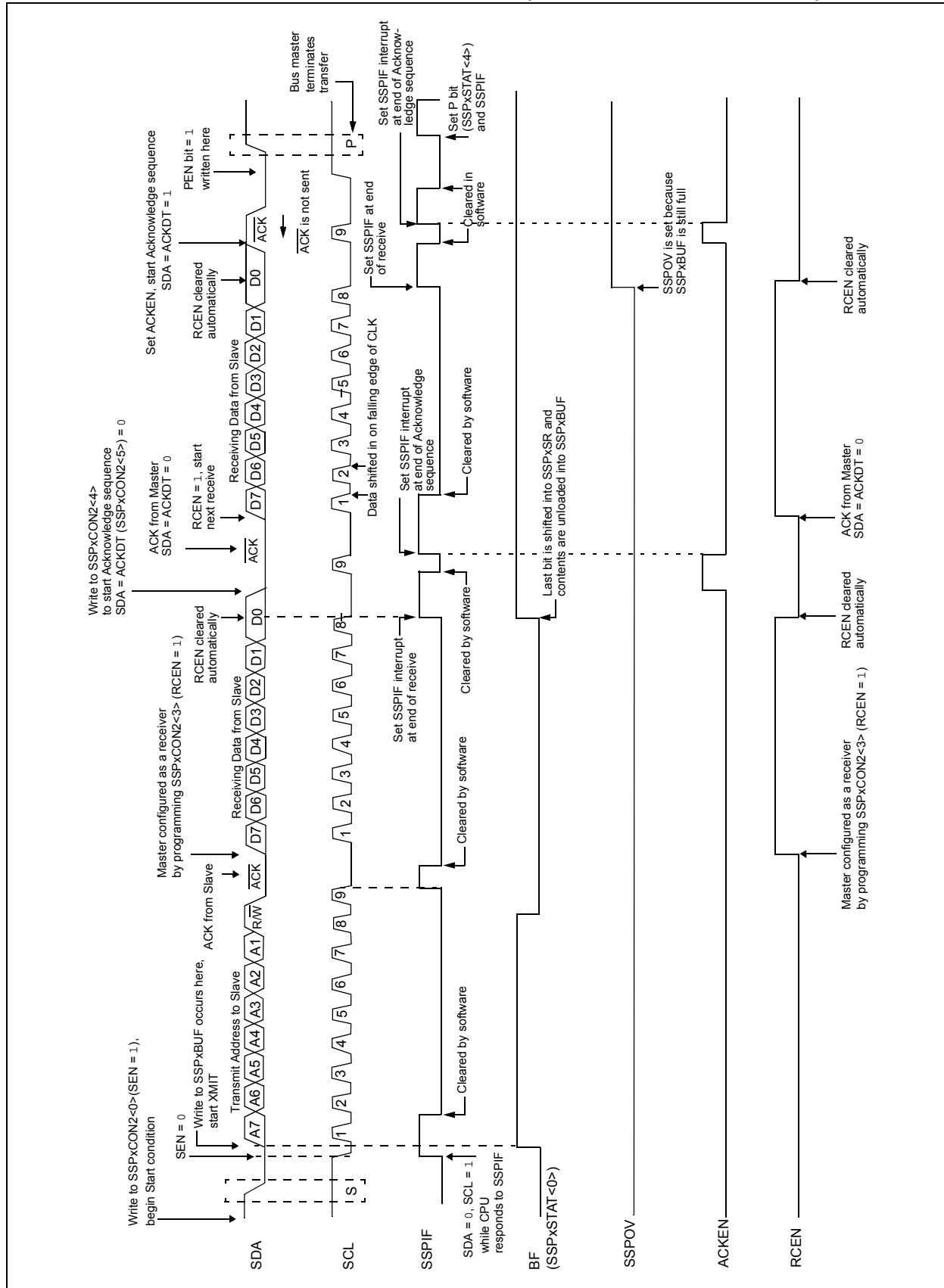
16.6.7.3 WCOL Status Flag

If the user writes the SSPxBUF when a receive is already in progress (i.e., SSPxSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

16.6.7.4 Typical Receive Sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPxCON2 register.
2. SSPIF is set by hardware on completion of the Start.
3. SSPIF is cleared by software.
4. User writes SSPxBUF with the slave address to transmit and the R/W bit set.
5. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
6. The MSSP module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
7. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
8. User sets the RCEN bit of the SSPxCON2 register and the Master clocks in a byte from the slave.
9. After the 8th falling edge of SCL, SSPIF and BF are set.
10. Master clears SSPIF and reads the received byte from SSPxUF, clears BF.
11. Master sets ACK value sent to slave in ACKDT bit of the SSPxCON2 register and initiates the ACK by setting the ACKEN bit.
12. Masters ACK is clocked out to the slave and SSPIF is set.
13. User clears SSPIF.
14. Steps 8-13 are repeated for each received byte from the slave.
15. Master sends a not ACK or Stop to end communication.

FIGURE 16-29: I²C™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



16.6.8 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN, of the SSPxCON2 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 16-29).

16.6.8.1 WCOL Status Flag

If the user writes the SSPxBUF when an Acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write does not occur).

16.6.9 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, of the SSPxCON2 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 SSPxSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 16-30).

16.6.9.1 WCOL Status Flag

If the user writes the SSPxBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

FIGURE 16-30: ACKNOWLEDGE SEQUENCE WAVEFORM

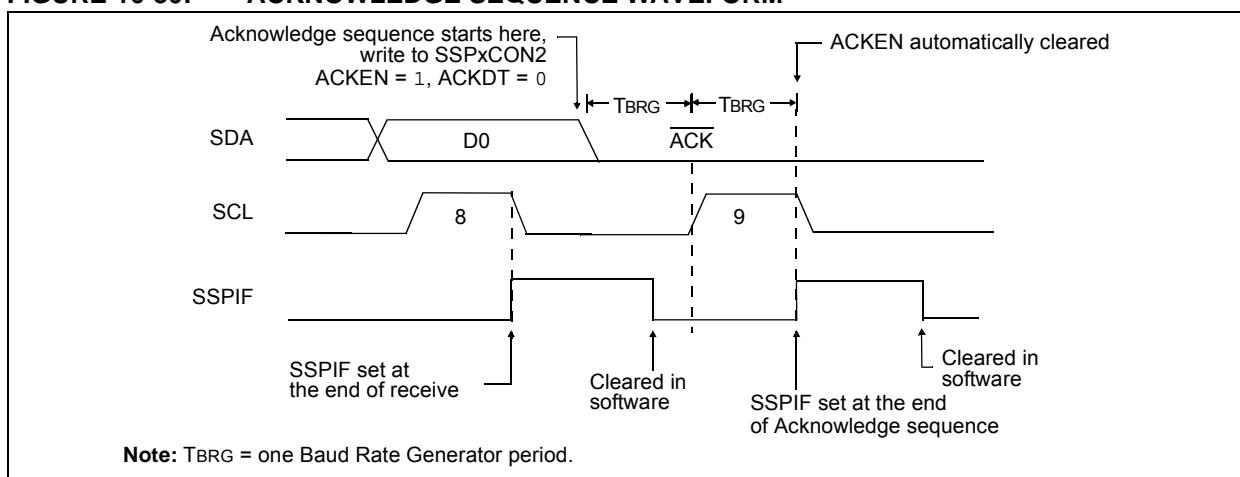
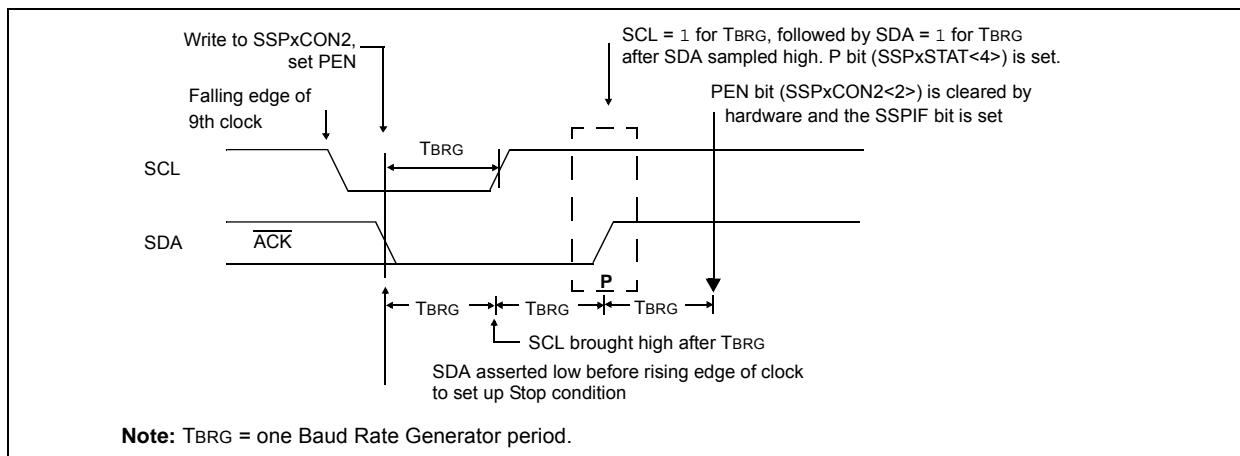


FIGURE 16-31: STOP CONDITION RECEIVE OR TRANSMIT MODE



16.6.10 SLEEP OPERATION

While in Sleep mode, the I²C 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).

16.6.11 EFFECTS OF A RESET

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

16.6.12 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 I²C bus may be taken when the P bit of the SSPxSTAT 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

16.6.13 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 is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF, and reset the I²C port to its Idle state (Figure 16-31).

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 SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I²C bus is free, the user can resume communication by asserting a Start condition.

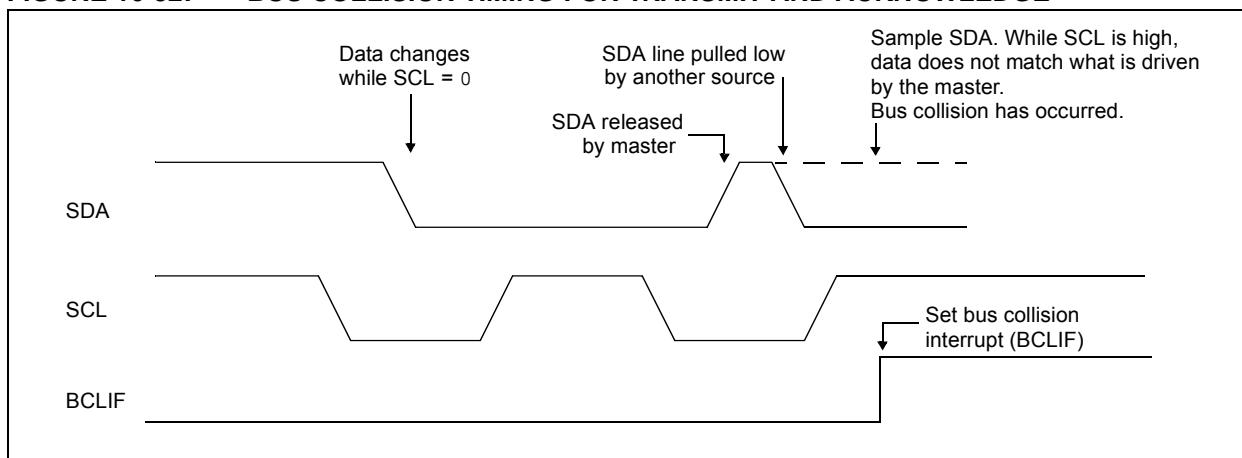
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I²C 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 SSPxBUF 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 I²C bus can be taken when the P bit is set in the SSPxSTAT register, or the bus is Idle and the S and P bits are cleared.

FIGURE 16-32: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



16.6.13.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

- SDA or SCL are sampled low at the beginning of the Start condition ([Figure 16-32](#)).
- SCL is sampled low before SDA is asserted low ([Figure 16-33](#)).

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:

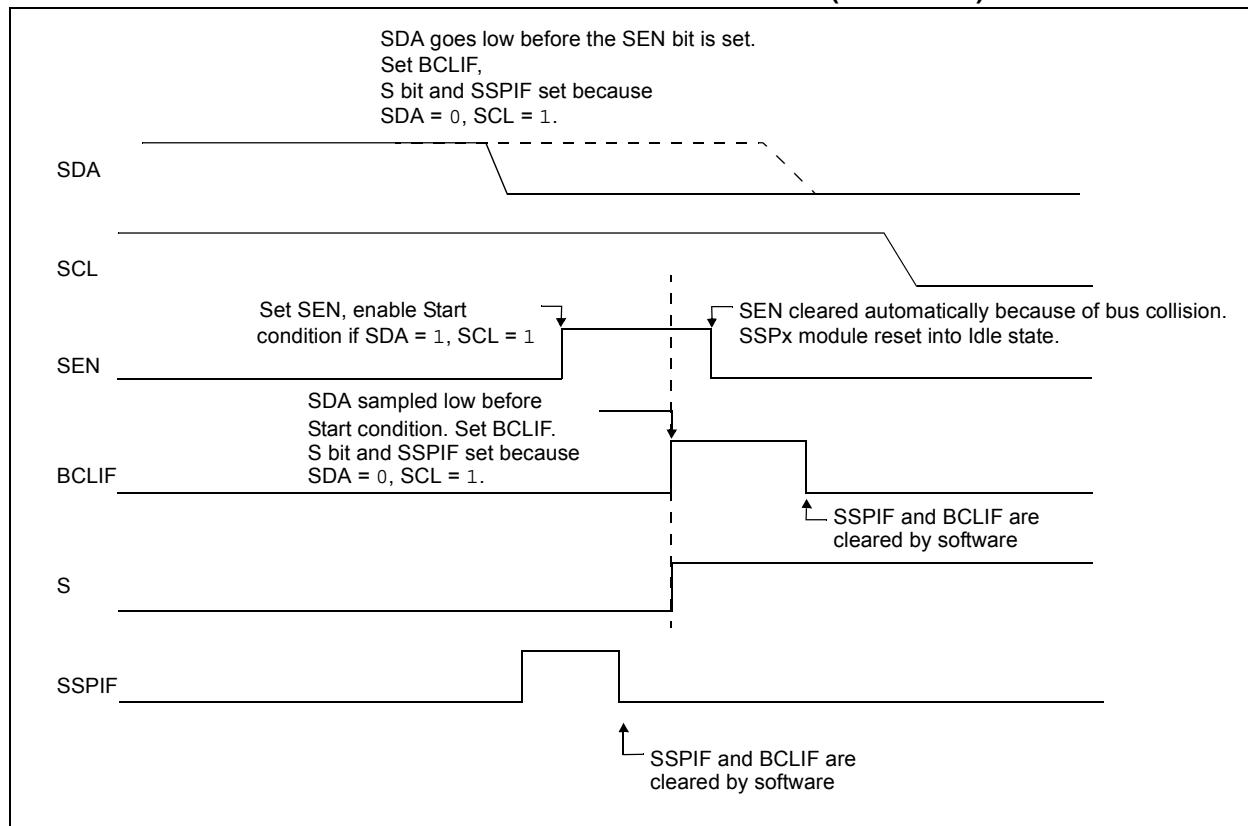
- the Start condition is aborted,
- the BCLIF flag is set and
- the MSSP module is reset to its Idle state ([Figure 16-32](#)).

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 16-34](#)). 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 zero; 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 16-33: BUS COLLISION DURING START CONDITION (SDA ONLY)



PIC18(L)F2X/45K50

FIGURE 16-34: BUS COLLISION DURING START CONDITION (SCL = 0)

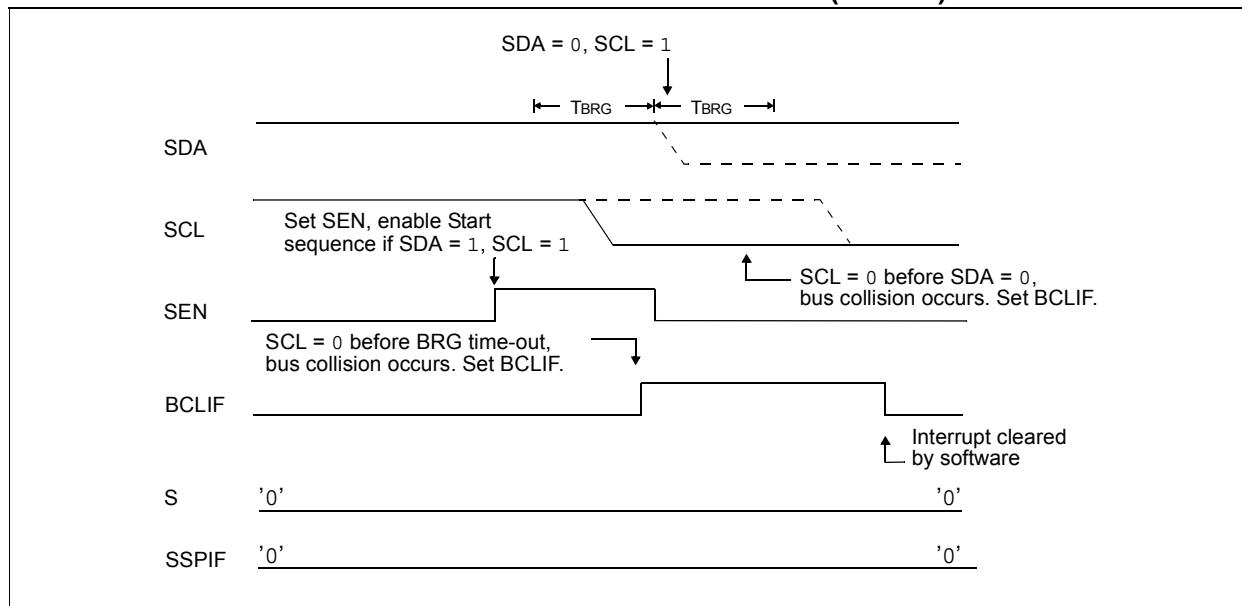
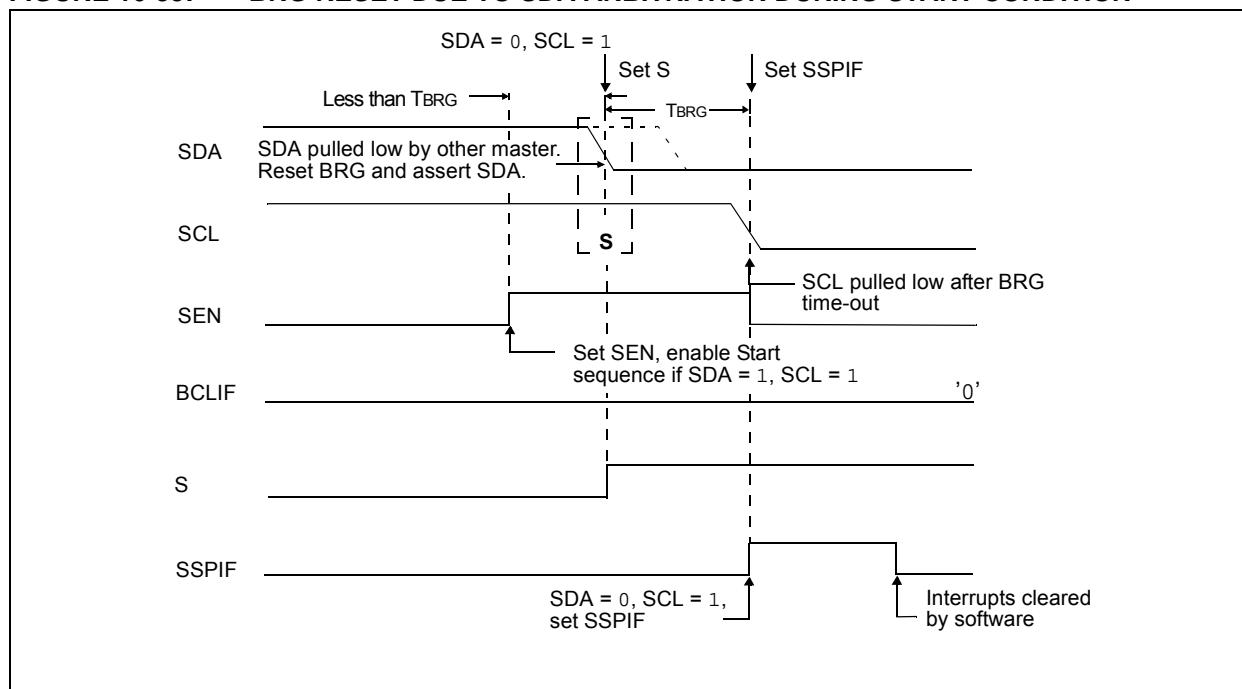


FIGURE 16-35: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



16.6.13.2 Bus Collision During a Repeated Start Condition

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

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

When the user releases SDA and the pin is allowed to float high, the BRG is loaded with SSPxADD and counts down to zero. 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 16-35](#)). 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 16-36](#).

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 16-36: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

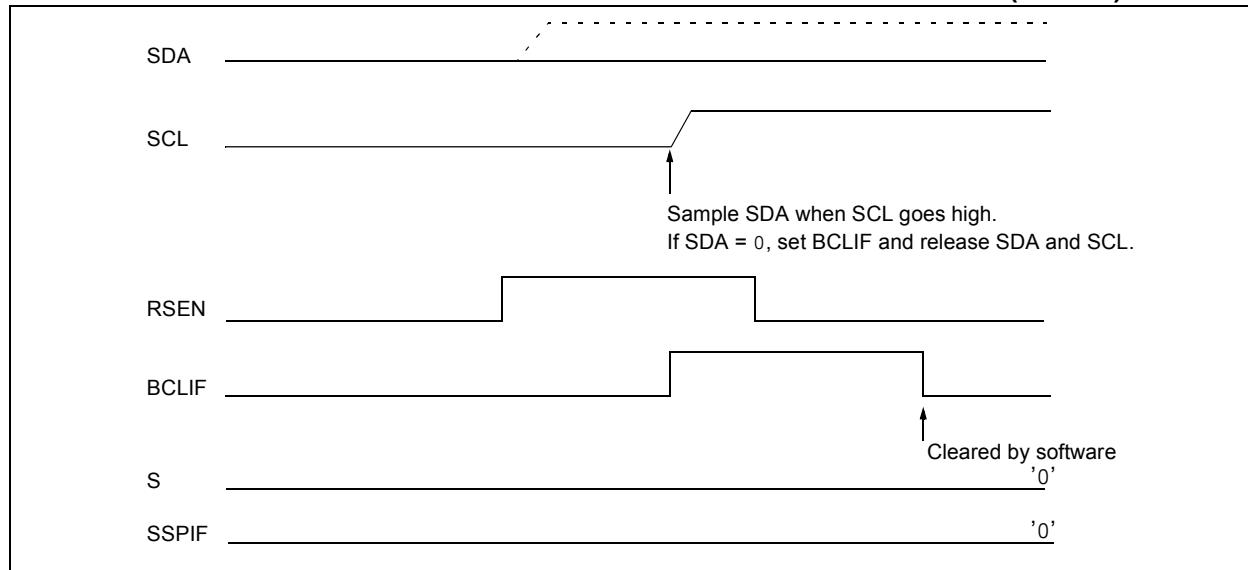
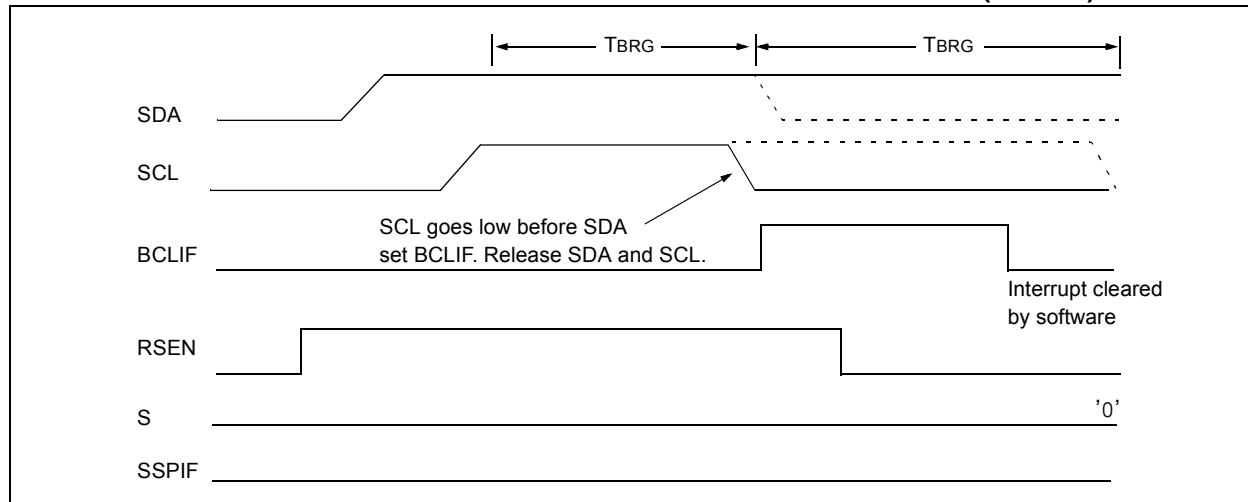


FIGURE 16-37: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



16.6.13.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

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

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPxADD 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 16-37). 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 16-38).

FIGURE 16-38: BUS COLLISION DURING A STOP CONDITION (CASE 1)

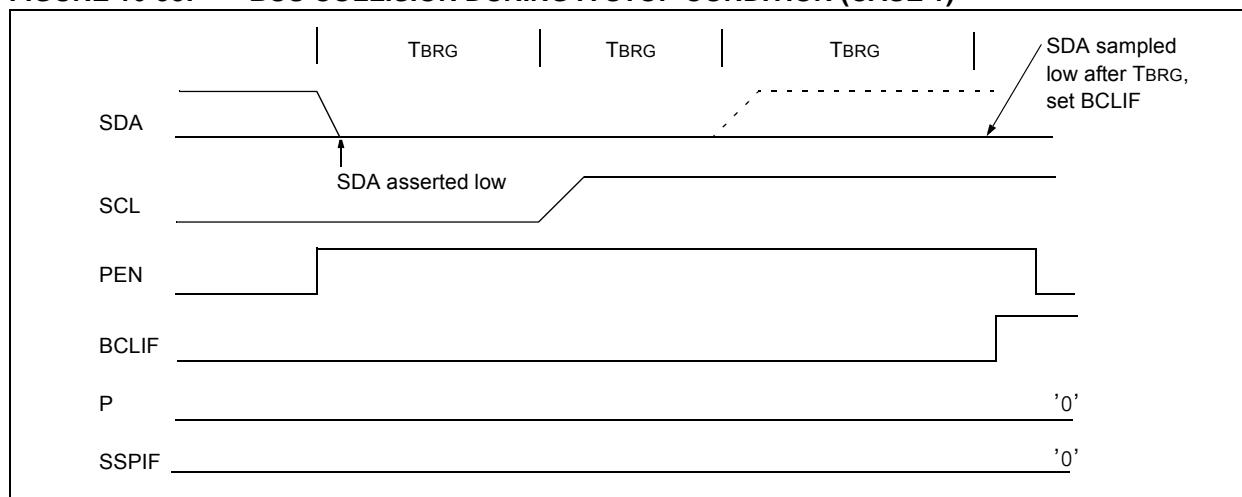


FIGURE 16-39: BUS COLLISION DURING A STOP CONDITION (CASE 2)

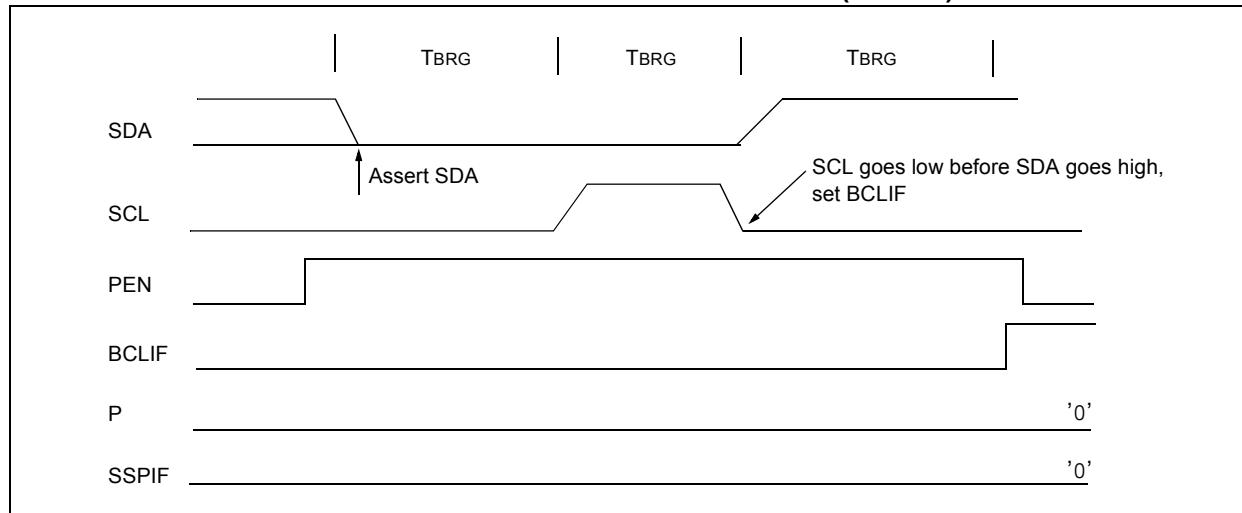


TABLE 16-3: REGISTERS ASSOCIATED WITH I²C™ OPERATION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|--|-----------|--------|--------|-----------|--------|--------|--------|------------------|
| ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 148 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 |
| PMD1 | — | MSSPMMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | 62 |
| SSP1ADD | SSP1 Address Register in I ² C™ Slave Mode. SSP1 Baud Rate Reload Register in I ² C Master mode. | | | | | | | | 257 |
| SSP1BUF | SSP1 Receive Buffer/Transmit Register | | | | | | | | — |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 252 |
| SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 254 |
| SSP1CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 255 |
| SSP1MSK | SSP1 MASK Register bits | | | | | | | | 256 |
| SSP1STAT | SMP | CKE | D/A | P | S | R/W | UA | BF | 251 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 |

Legend: Shaded bits are not used by the MSSP in I²C mode.

PIC18(L)F2X/45K50

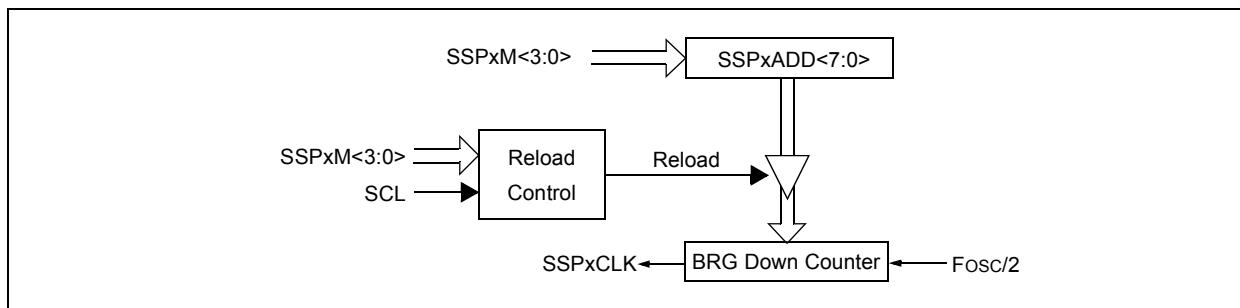
16.7 Baud Rate Generator

The MSSP module has a Baud Rate Generator available for clock generation in both I²C and SPI Master modes. The Baud Rate Generator (BRG) reload value is placed in the SSPxADD register ([Register 16-6](#)). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting down.

Once the given operation is complete, the internal clock will automatically stop counting and the clock pin will remain in its last state.

An internal signal “Reload” in [Figure 16-39](#) triggers the value from SSPxADD to be loaded into the BRG counter. This occurs twice for each oscillation of the module clock line. The logic dictating when the reload signal is asserted depends on the mode the MSSP is being operated in.

FIGURE 16-40: BAUD RATE GENERATOR BLOCK DIAGRAM



Note: Values of 0x00, 0x01 and 0x02 are not valid for SSPxADD when used as a Baud Rate Generator for I²C™. This is an implementation limitation.

TABLE 16-4: MSSP CLOCK RATE W/BRG

| Fosc | Fcy | BRG Value | FCLOCK (2 Rollovers of BRG) |
|--------|-------|-----------|--------------------------------|
| 32 MHz | 8 MHz | 13h | 400 kHz ⁽¹⁾ |
| 32 MHz | 8 MHz | 19h | 308 kHz |
| 32 MHz | 8 MHz | 4Fh | 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 | 09h | 100 kHz |

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.

[Table 16-4](#) demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD.

EQUATION 16-1:

$$FCLOCK = \frac{FOSC}{(SSPxADD + 1)(4)}$$

16.8 Register Definitions: MSSP Control

REGISTER 16-1: SSPxSTAT: SSPx STATUS REGISTER

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

Legend:

R = Readable bit

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 | SMP: SPI Data Input Sample bit <u>SPI Master mode:</u> 1 = Input data sampled at end of data output time 0 = Input data sampled at middle of data output time <u>SPI Slave mode:</u> SMP must be cleared when SPI is used in Slave mode <u>In I²C Master or Slave mode:</u> 1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz) 0 = Slew rate control enabled for high speed mode (400 kHz) |
| bit 6 | CKE: SPI Clock Edge Select bit (SPI mode only) <u>In SPI Master or Slave mode:</u> 1 = Transmit occurs on transition from active to Idle clock state 0 = Transmit occurs on transition from Idle to active clock state <u>In I²C mode only:</u> 1 = Enable input logic so that thresholds are compliant with SMBus specification 0 = Disable SMBus specific inputs |
| bit 5 | D/A: Data/Address bit (I ² C mode only) 1 = Indicates that the last byte received or transmitted was data 0 = Indicates that the last byte received or transmitted was address |
| bit 4 | P: Stop bit (I ² C mode only. This bit is cleared when the MSSP module is disabled, SSPxEN is cleared.) 1 = Indicates that a Stop bit has been detected last (this bit is '0' on Reset) 0 = Stop bit was not detected last |
| bit 3 | S: Start bit (I ² C mode only. This bit is cleared when the MSSP module is disabled, SSPxEN is cleared.) 1 = Indicates that a Start bit has been detected last (this bit is '0' on Reset) 0 = Start bit was not detected last |
| bit 2 | R/W: Read/Write bit information (I ² C mode only) This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit, or not ACK bit. <u>In I²C Slave mode:</u> 1 = Read 0 = Write <u>In I²C Master mode:</u> 1 = Transmit is in progress 0 = Transmit is not in progress OR-ing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Idle mode. |
| bit 1 | UA: Update Address bit (10-bit I ² C mode only) 1 = Indicates that the user needs to update the address in the SSPxADD register 0 = Address does not need to be updated |
| bit 0 | BF: Buffer Full Status bit <u>Receive (SPI and I²C modes):</u> 1 = Receive complete, SSPxBUF is full 0 = Receive not complete, SSPxBUF is empty <u>Transmit (I²C mode only):</u> 1 = Data transmit in progress (does not include the ACK and Stop bits), SSPxBUF is full 0 = Data transmit complete (does not include the ACK and Stop bits), SSPxBUF is empty |

PIC18(L)F2X/45K50

REGISTER 16-2: SSPxCON1: SSPx CONTROL REGISTER 1

| R/C/HS-0 | R/C/HS-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|----------|----------|-------|-------|-------|-----------|-------|-------|
| WCOL | SSPOV | SSPEN | CKP | | SSPM<3:0> | | |
| bit 7 | | | | | | | |

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

HS = Bit is set by hardware

C = User cleared

bit 7

WCOL: Write Collision Detect bit

Master mode:

1 = A write to the SSPxBUF register was attempted while the I²C™ conditions were not valid for a transmission to be started

0 = No collision

Slave mode:

1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)

0 = No collision

bit 6

SSPOV: Receive Overflow Indicator bit⁽¹⁾

In SPI mode:

1 = A new byte is received while the SSPxBUF register is still holding the previous data. In case of overflow, the data in SSPxSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPxBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register (must be cleared in software).

0 = No overflow

In I²C mode:

1 = A byte is received while the SSPxBUF register is still holding the previous byte. SSPOV is a "don't care" in Transmit mode (must be cleared in software).

0 = No overflow

bit 5

SSPEN: Synchronous Serial Port Enable bit

In both modes, when enabled, these pins must be properly configured as input or output

In SPI mode:

1 = Enables serial port and configures SCK, SDO, SDI and \overline{SS} as the source of the serial port pins⁽²⁾

0 = Disables serial port and configures these pins as I/O port pins

In I²C mode:

1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins⁽³⁾

0 = Disables serial port and configures these pins as I/O port pins

bit 4

CKP: Clock Polarity Select bit

In SPI mode:

1 = Idle state for clock is a high level

0 = Idle state for clock is a low level

In I²C Slave mode:

SCL release control

1 = Enable clock

0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In I²C Master mode:

Unused in this mode

REGISTER 16-2: SSPxCON1: SSPx CONTROL REGISTER 1 (CONTINUED)

| | |
|---------|--|
| bit 3-0 | SSPM<3:0> : Synchronous Serial Port Mode Select bits 0000 = SPI Master mode, clock = Fosc/4 0001 = SPI Master mode, clock = Fosc/16 0010 = SPI Master mode, clock = Fosc/64 0011 = SPI Master mode, clock = TMR2 output/2 0100 = SPI Slave mode, clock = SCK pin, <u>SS</u> pin control enabled 0101 = SPI Slave mode, clock = SCK pin, <u>SS</u> pin control disabled, <u>SS</u> can be used as I/O pin 0110 = I ² C Slave mode, 7-bit address 0111 = I ² C Slave mode, 10-bit address 1000 = I ² C Master mode, clock = Fosc / (4 * (SSPxADD+1)) ⁽⁴⁾ 1001 = Reserved 1010 = SPI Master mode, clock = Fosc/(4 * (SSPxADD+1)) 1011 = I ² C firmware controlled Master mode (slave idle) 1100 = Reserved 1101 = Reserved 1110 = I ² C Slave mode, 7-bit address with Start and Stop bit interrupts enabled 1111 = I ² C Slave mode, 10-bit address with Start and Stop bit interrupts enabled |
|---------|--|

- Note**
- 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register.
 - 2: When enabled, these pins must be properly configured as input or output.
 - 3: When enabled, the SDA and SCL pins must be configured as inputs.
 - 4: SSPxADD values of 0, 1 or 2 are not supported for I²C mode.

PIC18(L)F2X/45K50

REGISTER 16-3: SSPxCON2: SSPx CONTROL REGISTER 2

| R/W-0 | R-0 | R/W-0 | R/S/HC-0 | R/S/HC-0 | R/S/HC-0 | R/S/HC-0 | R/W/HC-0 |
|-------|---------|-------|----------------------|---------------------|--------------------|---------------------|--------------------|
| GCEN | ACKSTAT | ACKDT | 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' |
| 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 | HC = Cleared by hardware S = User set |

- bit 7 **GCEN:** General Call Enable bit (in I²C™ Slave mode only)
1 = Enable interrupt when a general call address (0x00 or 00h) is received in the SSPxSR
0 = General call address disabled
- bit 6 **ACKSTAT:** Acknowledge Status bit (in I²C mode only)
1 = Acknowledge was not received
0 = Acknowledge was received
- bit 5 **ACKDT:** Acknowledge Data bit (in I²C mode only)
In Receive mode:
Value transmitted when the user initiates an Acknowledge sequence at the end of a receive
1 = Not Acknowledge
0 = Acknowledge
- bit 4 **ACKEN⁽¹⁾:** Acknowledge Sequence Enable bit (in I²C Master mode only)
In Master Receive mode:
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⁽¹⁾:** Receive Enable bit (in I²C Master mode only)
1 = Enables Receive mode for I²C
0 = Receive idle
- bit 2 **PEN⁽¹⁾:** Stop Condition Enable bit (in I²C Master mode only)
SCK Release Control:
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle
- bit 1 **RSEN⁽¹⁾:** Repeated Start Condition Enabled bit (in I²C Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle
- bit 0 **SEN⁽¹⁾:** Start Condition Enabled bit (in I²C Master mode only)
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, this bit may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).

REGISTER 16-4: SSPxCON3: SSPx CONTROL REGISTER 3

| R-0 | R/W-0 |
|--------|-------|-------|-------|-------|-------|-------|-------|
| ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN |
| 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 | ACKTIM: Acknowledge Time Status bit (I^2C mode only) ⁽³⁾ 1 = Indicates the I^2C bus is in an Acknowledge sequence, set on 8 th falling edge of SCL clock 0 = Not an Acknowledge sequence, cleared on 9 th rising edge of SCL clock |
| bit 6 | PCIE: Stop Condition Interrupt Enable bit (I^2C mode only) 1 = Enable interrupt on detection of Stop condition 0 = Stop detection interrupts are disabled ⁽²⁾ |
| bit 5 | SCIE: Start Condition Interrupt Enable bit (I^2C mode only) 1 = Enable interrupt on detection of Start or Restart conditions 0 = Start detection interrupts are disabled ⁽²⁾ |
| bit 4 | BOEN: Buffer Overwrite Enable bit <u>In SPI Slave mode:</u> ⁽¹⁾ 1 = SSPxBUF updates every time that a new data byte is shifted in ignoring the BF bit 0 = If new byte is received with BF bit of the SSPxSTAT register already set, SSPOV bit of the SSPxCON1 register is set, and the buffer is not updated <u>In I^2C Master mode:</u> This bit is ignored. <u>In I^2C Slave mode:</u> 1 = SSPxBUF is updated and \overline{ACK} is generated for a received address/data byte, ignoring the state of the SSPOV bit only if the BF bit = 0. 0 = SSPxBUF is only updated when SSPOV is clear |
| bit 3 | SDAHT: SDA Hold Time Selection bit (I^2C mode only) 1 = Minimum of 300 ns hold time on SDA after the falling edge of SCL 0 = Minimum of 100 ns hold time on SDA after the falling edge of SCL |
| bit 2 | SBCDE: Slave Mode Bus Collision Detect Enable bit (I^2C Slave mode only) If on the rising edge of SCL, SDA is sampled low when the module is outputting a high state, the BCLIF bit of the PIR2 register is set, and bus goes idle 1 = Enable slave bus collision interrupts 0 = Slave bus collision interrupts are disabled |
| bit 1 | AHEN: Address Hold Enable bit (I^2C Slave mode only) 1 = Following the 8th falling edge of SCL for a matching received address byte; CKP bit of the SSPxCON1 register will be cleared and the SCL will be held low. 0 = Address holding is disabled |

- Note 1:** For daisy-chained SPI operation; allows the user to ignore all but the last received byte. SSPOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPxBUF.
- 2:** This bit has no effect in Slave modes for which Start and Stop condition detection is explicitly listed as enabled.
- 3:** The ACKTIM Status bit is active only when the AHEN bit or DHEN bit is set.

PIC18(L)F2X/45K50

REGISTER 16-4: SSPxCON3: SSPx CONTROL REGISTER 3 (CONTINUED)

- bit 0 **DHEN:** Data Hold Enable bit (I^2C Slave mode only)
1 = Following the 8th falling edge of SCL for a received data byte; slave hardware clears the CKP bit of the SSPxCON1 register and SCL is held low.
0 = Data holding is disabled

- Note 1:** For daisy-chained SPI operation; allows the user to ignore all but the last received byte. SSPOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPxBUF.
- 2:** This bit has no effect in Slave modes for which Start and Stop condition detection is explicitly listed as enabled.
- 3:** The ACKT1M Status bit is active only when the AHEN bit or DHEN bit is set.

REGISTER 16-5: SSPxMSK: SSPx MASK REGISTER

| R/W-1 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0 |
| 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 7-1 **MSK<7:1>:** Mask bits
1 = The received address bit n is compared to SSPxADD<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
 I^2C Slave mode, 10-bit address (SSPxM<3:0> = 0111 or 1111):
1 = The received address bit 0 is compared to SSPxADD<0> to detect I^2C address match
0 = The received address bit 0 is not used to detect I^2C address match
 I^2C Slave mode, 7-bit address, the bit is ignored

REGISTER 16-6: SSPxADD: 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 |
| ADD<7:0> | | | | | | | |
| 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 | |

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<2:1>:** 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<7:1>:** 7-bit address
 bit 0 **Not used:** Unused in this mode. Bit state is a “don’t care”.

17.0 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 17-1](#) and [Figure 17-2](#).

FIGURE 17-1: EUSART TRANSMIT BLOCK DIAGRAM

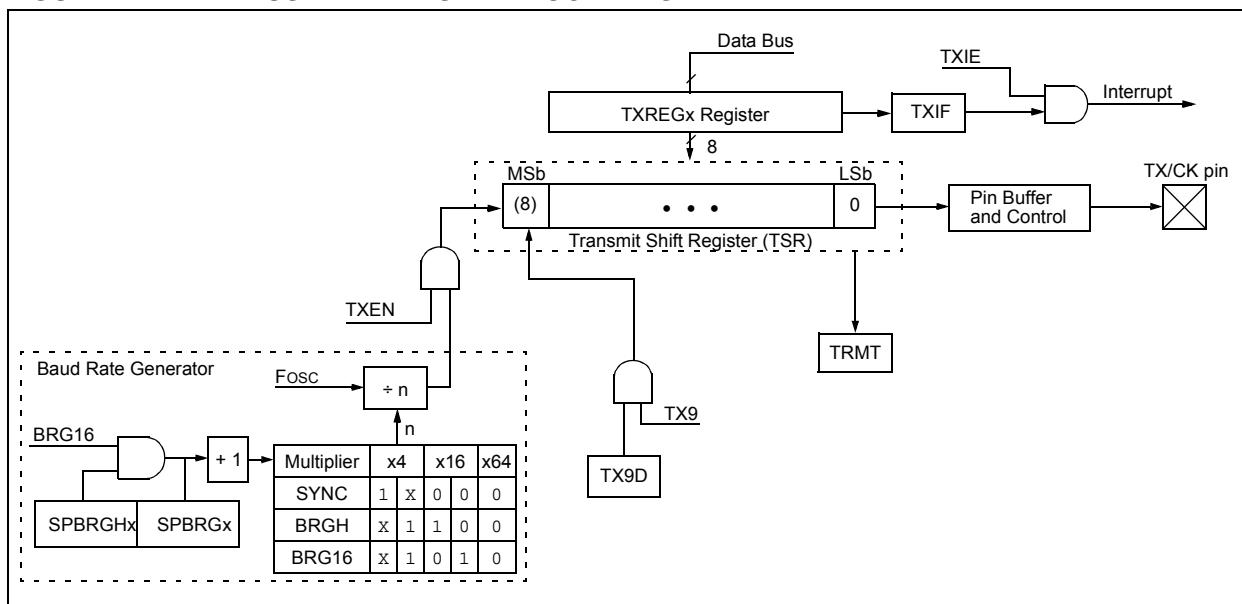
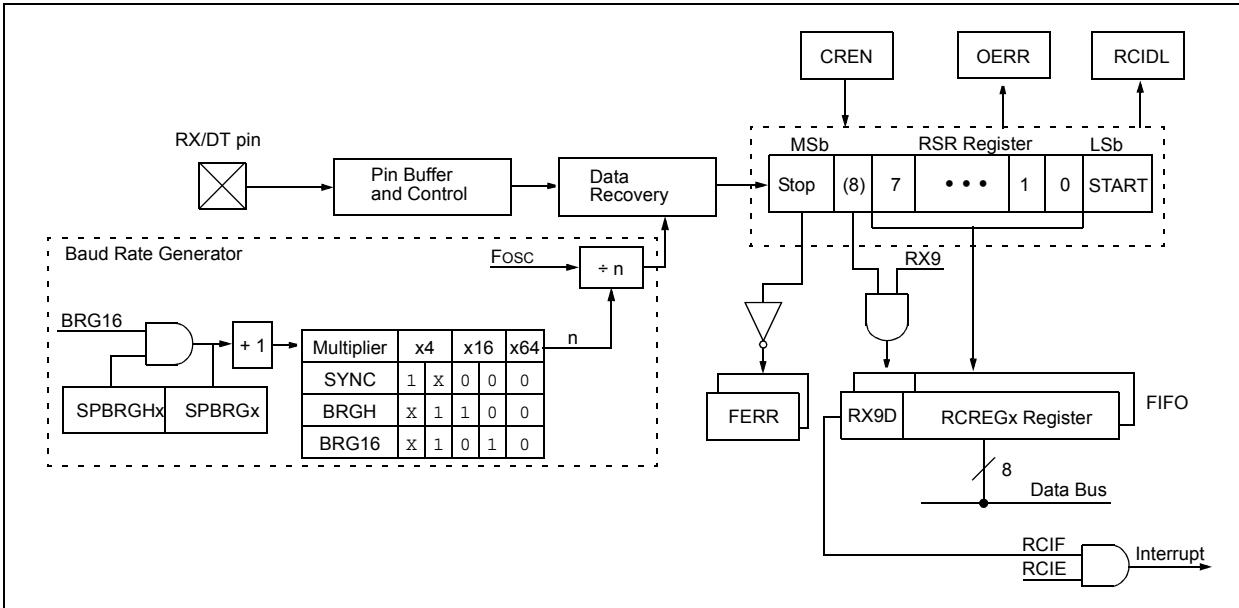


FIGURE 17-2: EUSART RECEIVE BLOCK DIAGRAM



The operation of the EUSART module is controlled through three registers:

- Transmit Status and Control (TXSTAx)
- Receive Status and Control (RCSTAx)
- Baud Rate Control (BAUDCONx)

These registers are detailed in [Register 17-1](#), [Register 17-2](#) and [Register 17-3](#), 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.

When the receiver or transmitter section is not enabled then the corresponding RX/DT or TX/CK pin may be used for general purpose input and output.

17.1 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 17-5](#) 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.

17.1.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in [Figure 17-1](#). 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 TXREGx register.

17.1.1.1 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 TXSTAx register enables the transmitter circuitry of the EUSART. Clearing the SYNC bit of the TXSTAx register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTAx 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: The TXIF transmitter interrupt flag is set when the TXEN enable bit is set.

17.1.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXREGx register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREGx 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 TXREGx until the Stop bit of the previous character has been transmitted. The pending character in the TXREGx 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 TXREGx.

17.1.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the TXCKP bit of the BAUDCONx register. The default state of this bit is '0' which selects high true transmit idle and data bits. Setting the TXCKP bit to '1' will invert the transmit data resulting in low true idle and data bits. The TXCKP bit controls transmit data polarity only in Asynchronous mode. In Synchronous mode the TXCKP bit has a different function.

17.1.1.4 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 TXREGx. 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 TXREGx. The TXIF flag bit is not cleared immediately upon writing TXREGx. TXIF becomes valid in the second instruction cycle following the write execution. Polling TXIF immediately following the TXREGx 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 TXREGx 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 TXREGx.

17.1.1.5 TSR Status

The TRMT bit of the TXSTAx 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 TXREGx. 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.

17.1.1.6 Transmitting 9-Bit Characters

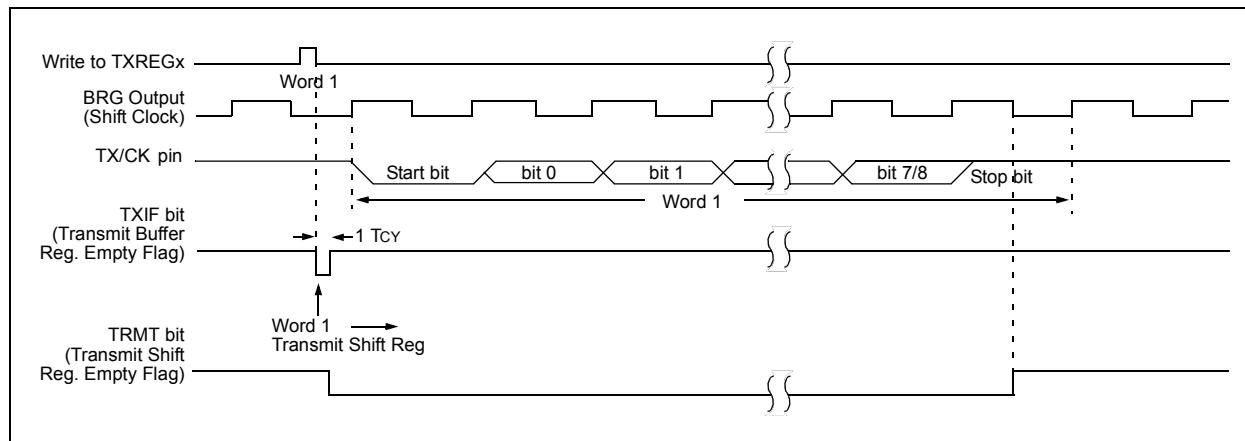
The EUSART supports 9-bit character transmissions. When the TX9 bit of the TXSTAx register is set the EUSART will shift 9 bits out for each character transmitted. The TX9D bit of the TXSTAx register is the ninth, and Most Significant, data bit. When transmitting 9-bit data, the TX9D data bit must be written before writing the 8 Least Significant bits into the TXREGx. All nine bits of data will be transferred to the TSR shift register immediately after the TXREGx is written.

A special 9-bit Address mode is available for use with multiple receivers. See [Section 17.1.2.8 “Address Detection”](#) for more information on the Address mode.

17.1.1.7 Asynchronous Transmission Setup

1. Initialize the SPBRGHx:SPBRGx register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see [Section 17.4 “EUSART Baud Rate Generator \(BRG\)”](#)).
2. Set the RX/DT and TX/CK TRIS controls to ‘1’.
3. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
4. If 9-bit transmission is desired, set the TX9 control bit. A set ninth data bit will indicate that the eight Least Significant data bits are an address when the receiver is set for address detection.
5. Set the TXCKP control bit if inverted transmit data polarity is desired.
6. Enable the transmission by setting the TXEN control bit. This will cause the TXIF interrupt bit to be set.
7. If interrupts are desired, set the TXIE interrupt enable bit. An interrupt will occur immediately provided that the GIE/GIEH and PEIE/GIEL bits of the INTCON register are also set.
8. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
9. Load 8-bit data into the TXREGx register. This will start the transmission.

FIGURE 17-3: ASYNCHRONOUS TRANSMISSION



PIC18(L)F2X/45K50

FIGURE 17-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)

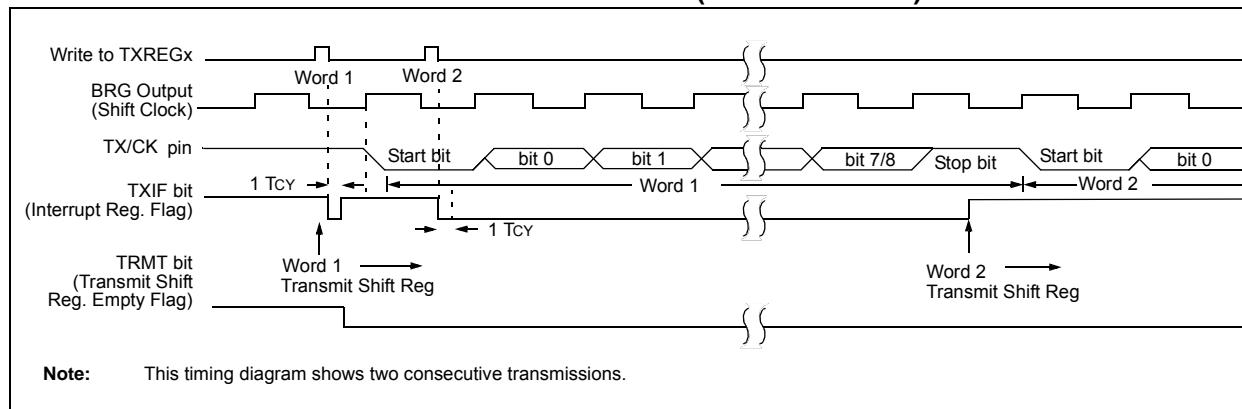


TABLE 17-1: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|---------------------------------------|-----------|--------|--------|--------|--------|--------|--------|---------------------|
| BAUDCON1 | ABDOVF | RIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 270 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 269 |
| SPBRG1 | EUSART Baud Rate Generator, Low Byte | | | | | | | — | |
| SPBRGH1 | EUSART Baud Rate Generator, High Byte | | | | | | | — | |
| TXREG1 | EUSART Transmit Register | | | | | | | — | |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDDB | BRGH | TRMT | TX9D | 268 |

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

17.1.2 EUSART ASYNCHRONOUS RECEIVER

The Asynchronous mode would typically be used in RS-232 systems. The receiver block diagram is shown in [Figure 17-2](#). 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 eight or nine 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 RCREGx register.

17.1.2.1 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_x register enables the receiver circuitry of the EUSART. Clearing the SYNC bit of the TXSTA_x register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTA_x 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.

17.1.2.2 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 17.1.2.5 “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 RCREGx register.

Note: If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See [Section 17.1.2.6 “Receive Overrun Error”](#) for more information on overrun errors.

17.1.2.3 Receive Data Polarity

The polarity of the receive data can be controlled with the RXDTP bit of the BAUDCON_x register. The default state of this bit is '0' which selects high true receive Idle and data bits. Setting the RXDTP bit to '1' will invert the receive data resulting in low true Idle and data bits. The RXDTP bit controls receive data polarity only in Asynchronous mode. In Synchronous mode the RXDTP bit has a different function.

17.1.2.4 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/GIEL peripheral interrupt enable bit of the INTCON register
- GIE/GIEH 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.

17.1.2.5 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 RCSTAx 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 RCREGx.

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 RCSTAx register which resets the EUSART. Clearing the CREN bit of the RCSTAx 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 RCREGx will not clear the FERR bit.

17.1.2.6 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 RCSTAx 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 RCSTAx register or by resetting the EUSART by clearing the SPEN bit of the RCSTAx register.

17.1.2.7 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTAx register is set, the EUSART will shift nine bits into the RSR for each character received. The RX9D bit of the RCSTAx 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 RCREGx.

17.1.2.8 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 RCSTAx 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.

17.1.2.9 Asynchronous Reception Setup:

1. Initialize the SPBRGHx:SPBRGx register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see [Section 17.4 “EUSART Baud Rate Generator \(BRG\)”](#)).
2. Set the RX/DT and TX/CK TRIS controls to ‘1’.
3. 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.
4. If interrupts are desired, set the RCIE interrupt enable bit and set the GIE/GIEH and PEIE/GIEL bits of the INTCON register.
5. If 9-bit reception is desired, set the RX9 bit.
6. Set the RXDTP 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 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 RCSTAx register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
10. Get the received eight Least Significant data bits from the receive buffer by reading the RCREGx register.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

17.1.2.10 9-Bit Address Detection Mode Setup

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

1. Initialize the SPBRGHx, SPBRGx register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see [Section 17.4 “EUSART Baud Rate Generator \(BRG\)”](#)).
2. Set the RX/DT and TX/CK TRIS controls to ‘1’.
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCIE interrupt enable bit and set the GIE/GIEH and PEIE/GIEL bits of the INTCON register.
5. Enable 9-bit reception by setting the RX9 bit.
6. Enable address detection by setting the ADDEN bit.
7. Set the RXDTP if inverted receive polarity is desired.
8. Enable reception by setting the CREN bit.
9. 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.
10. Read the RCSTAx register to get the error flags. The ninth data bit will always be set.
11. Get the received eight Least Significant data bits from the receive buffer by reading the RCREGx register. Software determines if this is the device’s address.
12. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
13. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

PIC18(L)F2X/45K50

FIGURE 17-5: ASYNCHRONOUS RECEPTION

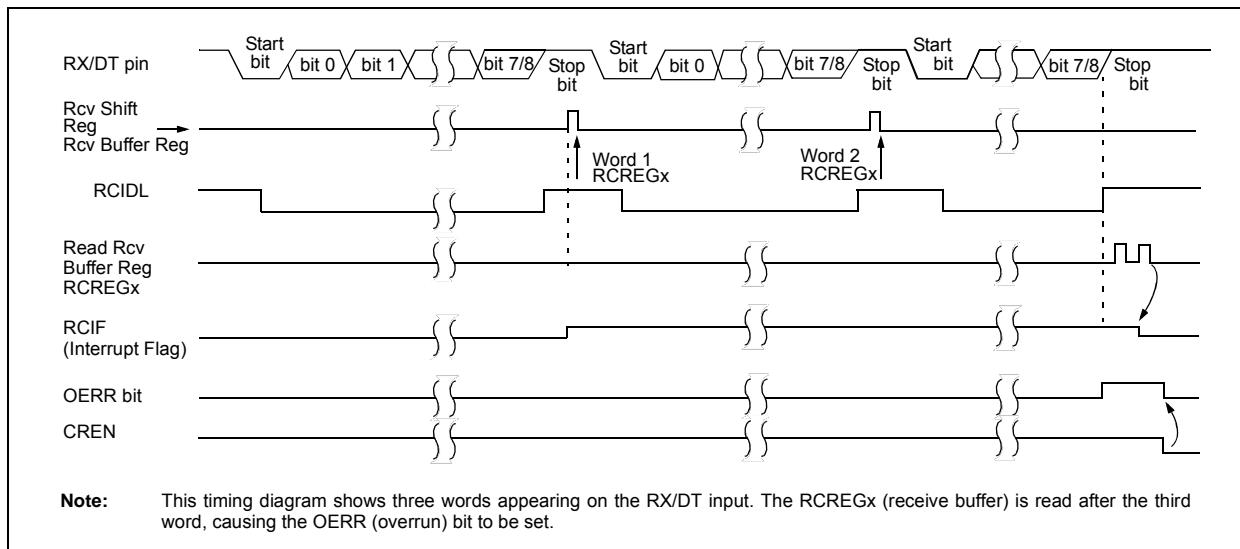


TABLE 17-2: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|---------------------------------------|-----------|--------|--------|-------|--------|--------|--------|---------------------|
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 270 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 |
| RCREG1 | EUSART Receive Register | | | | | | | | — |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 269 |
| SPBRG1 | EUSART Baud Rate Generator, Low Byte | | | | | | | | — |
| SPBRGH1 | EUSART Baud Rate Generator, High Byte | | | | | | | | — |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISCO | 149 |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 268 |

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

17.2 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 [3.6 “Internal Clock Modes”](#) 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 17.4.1 “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.

PIC18(L)F2X/45K50

17.3 Register Definitions: EUSART Control

REGISTER 17-1: TXSTAx: 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 | SEND B | 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 **SEND B:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed
Synchronous mode:
Don't care
- bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode
- bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full
- bit 0 **TX9D:** Ninth bit of Transmit Data
Can be address/data bit or a parity bit.

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

REGISTER 17-2: RCSTAx: 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-0 |
|-------|-------|-------|-------|-------|------|------|-------|
| 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 <u>Asynchronous mode:</u> Don't care <u>Synchronous mode – Master:</u> 1 = Enables single receive 0 = Disables single receive This bit is cleared after reception is complete. <u>Synchronous mode – Slave</u> Don't care |
| bit 4 | CREN: Continuous Receive Enable bit <u>Asynchronous mode:</u> 1 = Enables receiver 0 = Disables receiver <u>Synchronous mode:</u> 1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN) 0 = Disables continuous receive |
| bit 3 | ADDEN: Address Detect Enable bit <u>Asynchronous mode 9-bit (RX9 = 1):</u> 1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set 0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit <u>Asynchronous mode 8-bit (RX9 = 0):</u> Don't care |
| bit 2 | FERR: Framing Error bit 1 = Framing error (can be updated by reading RCREGx 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. |

PIC18(L)F2X/45K50

REGISTER 17-3: BAUDCONx: BAUD RATE CONTROL REGISTER

| R/W-0 | R-1 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
|--------|-------|-------|-------|-------|-----|-------|-------|
| ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **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 **RXDTP**: 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 **TXCKP**: 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 (SPBRGHx:SPBRGx)

0 = 8-bit Baud Rate Generator is used (SPBRGx)

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

WUE: Wake-up Enable bit

Asynchronous mode:

1 = Receiver is waiting for a falling edge. No character will be received but RCIF will be set on the falling edge. WUE will automatically clear on the rising edge.

0 = Receiver is operating normally

Synchronous mode:

Don't care

bit 0 **ABDEN**: Auto-Baud Detect Enable bit

Asynchronous mode:

1 = Auto-Baud Detect mode is enabled (clears when auto-baud is complete)

0 = Auto-Baud Detect mode is disabled

Synchronous mode:

Don't care

17.4 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 BAUDCONx register selects 16-bit mode.

The SPBRGHx:SPBRGx 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 TXSTAx register and the BRG16 bit of the BAUDCONx register. In Synchronous mode, the BRGH bit is ignored.

Table 17-3 contains the formulas for determining the baud rate. **Example 17-1** 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 17-5**. 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 SPBRGHx, SPBRGx 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 17-1: CALCULATING BAUD RATE ERROR

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

$$\text{Desired Baud Rate} = \frac{F_{OSC}}{64([SPBRGHx:SPBRGx] + 1)}$$

Solving for SPBRGHx:SPBRGx:

$$X = \frac{F_{OSC}}{64 \cdot \text{Desired Baud Rate}} - 1$$

$$= \frac{16000000}{64 \cdot 9600} - 1$$

$$= [25.042] = 25$$

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

$$= 9615$$

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

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

TABLE 17-3: 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 SPBRGHx, SPBRGx register pair.

PIC18(L)F2X/45K50

TABLE 17-4: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|---------------------------------------|--------|-------|-------|--------|--------|--------|--------|------------------|
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 270 |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 269 |
| SPBRG1 | EUSART Baud Rate Generator, Low Byte | | | | | | | | — |
| SPBRGH1 | EUSART Baud Rate Generator, High Byte | | | | | | | | — |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDDB | BRGH | TRMT | TX9D | 268 |

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

TABLE 17-5: BAUD RATES FOR ASYNCHRONOUS MODES

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|-----------|-------------------------------|---------|------------------------|-------------------|---------|------------------------|-------------------|---------|------------------------|--------------------|---------|------------------------|
| | Fosc = 48.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 16.000 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx value (decimal) |
| 300 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1200 | — | — | — | 1200 | 0.00 | 239 | 1202 | 0.16 | 207 | 1200 | 0.00 | 143 |
| 2400 | — | — | — | 2400 | 0.00 | 119 | 2404 | 0.16 | 103 | 2400 | 0.00 | 71 |
| 9600 | 9615 | 0.16 | 77 | 9600 | 0.00 | 29 | 9615 | 0.16 | 25 | 9600 | 0.00 | 17 |
| 10417 | 10417 | 0.00 | 71 | 10286 | -1.26 | 27 | 10417 | 0.00 | 23 | 10165 | -2.42 | 16 |
| 19.2k | 19.23k | 0.16 | 38 | 19.20k | 0.00 | 14 | 19.23k | 0.16 | 12 | 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 | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx 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 | — | — | — | — | — | — | — | — | — | — | — | — |

PIC18(L)F2X/45K50

TABLE 17-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | | | | |
|-----------|-------------------------------|---------|------------------------|-------------------|---------|------------------------|-------------------|---------|------------------------|--------------------|---------|------------------------|
| | Fosc = 48.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 16.000 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx value (decimal) |
| 300 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1200 | — | — | — | — | — | — | — | — | — | — | — | — |
| 2400 | — | — | — | — | — | — | — | — | — | — | — | — |
| 9600 | — | — | — | 9600 | 0.00 | 119 | 9615 | 0.16 | 103 | 9600 | 0.00 | 71 |
| 10417 | — | — | — | 10378 | -0.37 | 110 | 10417 | 0.00 | 95 | 10473 | 0.53 | 65 |
| 19.2k | 19.23k | 0.16 | 155 | 19.20k | 0.00 | 59 | 19.23k | 0.16 | 51 | 19.20k | 0.00 | 35 |
| 57.6k | 57.97k | 0.16 | 51 | 57.60k | 0.00 | 19 | 58.82k | 2.12 | 16 | 57.60k | 0.00 | 11 |
| 115.2k | 115.39k | 0.16 | 25 | 115.2k | 0.00 | 9 | 111.1k | -3.55 | 8 | 115.2k | 0.00 | 5 |

| 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 | SPBRGx value (decimal) | Actual Rate | % Error | SPBRGx value (decimal) | Actual Rate | % Error | SxBRGx value (decimal) | Actual Rate | % Error | SPBRGx 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 = 16.000 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) |
| 300 | 300.0 | 0.00 | 9999 | 300.0 | 0.00 | 3839 | 300.03 | 0.01 | 3332 | 300.0 | 0.00 | 2303 |
| 1200 | 1200 | 0.00 | 2499 | 1200 | 0.00 | 959 | 1200.5 | 0.04 | 832 | 1200 | 0.00 | 575 |
| 2400 | 2400 | 0.00 | 1249 | 2400 | 0.00 | 479 | 2398 | -0.08 | 416 | 2400 | 0.00 | 287 |
| 9600 | 9585 | -0.16 | 312 | 9600 | 0.00 | 119 | 9615 | 0.16 | 103 | 9600 | 0.00 | 71 |
| 10417 | 10417 | 0.00 | 287 | 10378 | -0.37 | 110 | 10417 | 0.00 | 95 | 10473 | 0.53 | 65 |
| 19.2k | 19.23k | 0.16 | 155 | 19.20k | 0.00 | 59 | 19.23k | 0.16 | 51 | 19.20k | 0.00 | 35 |
| 57.6k | 57.69k | 0.16 | 51 | 57.60k | 0.00 | 19 | 58.82k | 2.12 | 16 | 57.60k | 0.00 | 11 |
| 115.2k | 115.39k | 0.16 | 25 | 115.2k | 0.00 | 9 | 111.1k | -3.55 | 8 | 115.2k | 0.00 | 5 |

PIC18(L)F2X/45K50

TABLE 17-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

| 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 | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (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 | — | — | — |

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1 | | | | | | | | | | | |
|-----------|--|---------|---------------------------------|-------------------|---------|---------------------------------|-------------------|---------|---------------------------------|--------------------|---------|---------------------------------|
| | Fosc = 48.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 16.000 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) |
| 300 | 300 | 0.00 | 39999 | 300.0 | 0.00 | 15359 | 300.0 | 0.00 | 13332 | 300.0 | 0.00 | 9215 |
| 1200 | 1200 | 0.00 | 9999 | 1200 | 0.00 | 3839 | 1200.1 | 0.01 | 3332 | 1200 | 0.00 | 2303 |
| 2400 | 2400 | 0.00 | 4999 | 2400 | 0.00 | 1919 | 2399.5 | -0.02 | 1666 | 2400 | 0.00 | 1151 |
| 9600 | 9600 | 0.00 | 1249 | 9600 | 0.00 | 479 | 9592 | -0.08 | 416 | 9600 | 0.00 | 287 |
| 10417 | 10417 | 0.00 | 1151 | 10425 | 0.08 | 441 | 10417 | 0.00 | 383 | 10433 | 0.16 | 264 |
| 19.2k | 19.2k | 0.00 | 624 | 19.20k | 0.00 | 239 | 19.23k | 0.16 | 207 | 19.20k | 0.00 | 143 |
| 57.6k | 57.69k | 0.16 | 207 | 57.60k | 0.00 | 79 | 57.97k | 0.64 | 68 | 57.60k | 0.00 | 47 |
| 115.2k | 115.39k | 0.16 | 103 | 115.2k | 0.00 | 39 | 114.29k | -0.79 | 34 | 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 | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (decimal) | Actual Rate | % Error | SPBRGHx: SPBRGx (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 | — | — | — |

17.4.1 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 BAUDCONx register starts the auto-baud calibration sequence ([Section 17.4.2 "Auto-Baud Overflow"](#)). 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 SPBRGx begins counting up using the BRG counter clock as shown in [Table 17-6](#). The fifth rising edge will occur on the RX/DT pin at the end of the eighth bit period. At that time, an accumulated value totaling the proper BRG period is left in the SPBRGHx:SPBRGx register pair, the ABDEN bit is automatically cleared, and the RCIF interrupt flag is set. A read operation on the RCREGx needs to be performed to clear the RCIF interrupt. RCREGx content should be discarded. When calibrating for modes that do not use the SPBRGHx register the user can verify that the SPBRGx register did not overflow by checking for 00h in the SPBRGHx register.

The BRG auto-baud clock is determined by the BRG16 and BRGH bits as shown in [Table 17-6](#). During ABD, both the SPBRGHx and SPBRGx registers are used as a 16-bit counter, independent of the BRG16 bit setting. While calibrating the baud rate period, the SPBRGHx

and SPBRGx 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 17.4.3 "Auto-Wake-up on Break"](#)).

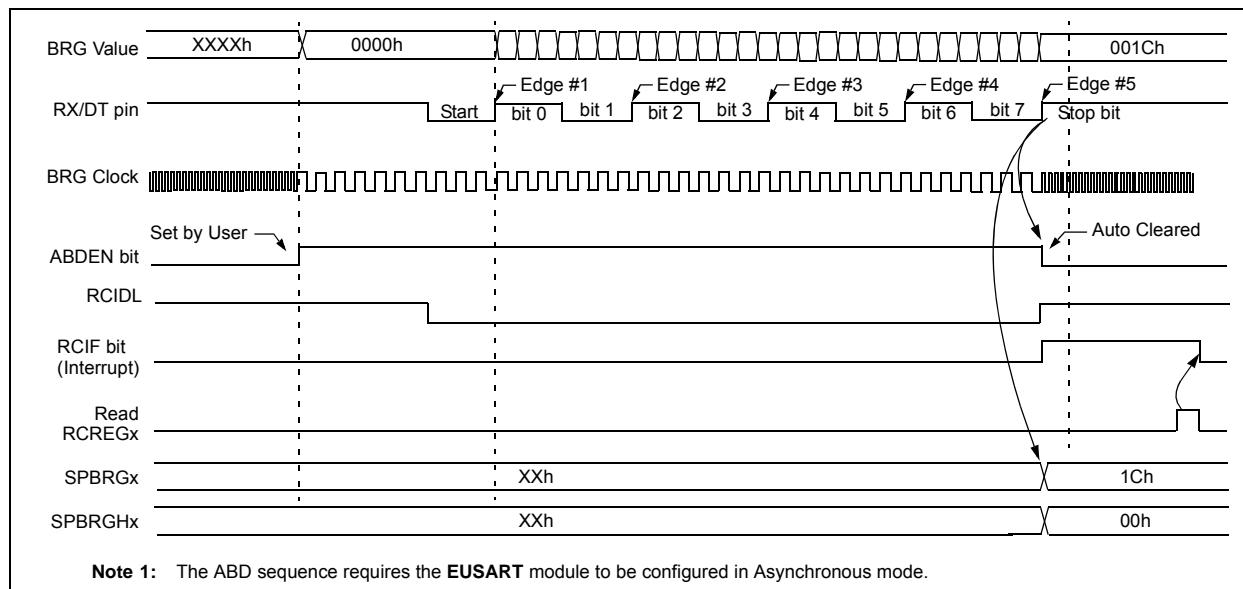
- 2: It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible.
- 3: During the auto-baud process, the auto-baud counter starts counting at 1. Upon completion of the auto-baud sequence, to achieve maximum accuracy, subtract 1 from the SPBRGHx:SPBRGx register pair.

TABLE 17-6: 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, SPBRGx and SPBRGHx registers are both used as a 16-bit counter, independent of BRG16 setting.

FIGURE 17-6: AUTOMATIC BAUD RATE CALIBRATION



17.4.2 AUTO-BAUD OVERFLOW

During the course of automatic baud detection, the ABDOVF bit of the BAUDCONx 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 SPBRGHx:SPBRGx register pair. After the ABDOVF has been set, the counter continues to count until the fifth rising edge is detected on the RX/DT pin. Upon detecting the fifth RX/DT edge, the hardware will set the RCIF interrupt flag and clear the ABDEN bit of the BAUDCONx register. The RCIF flag can be subsequently cleared by reading the RCREGx. The ABDOVF flag 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. The ABDOVF bit will remain set if the ABDEN bit is not cleared first.

17.4.3 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 BAUDCONx 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 17-7](#)), and asynchronously if the device is in Sleep mode ([Figure 17-8](#)). The interrupt condition is cleared by reading the RCREGx 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.

17.4.3.1 Special Considerations

Break Character

To avoid character errors or character fragments during a wake-up event, the wake-up character must be all zeros.

When the wake-up is enabled the function works independent of the low time on the data stream. If the WUE bit is set and a valid non-zero character is received, the low time from the Start bit to the first rising edge will be interpreted as the wake-up event. The remaining bits in the character will be received as a fragmented character and subsequent characters can result in framing or overrun errors.

Therefore, the initial character in the transmission must be all '0's. This must be 10 or more bit times, 13-bit times recommended for LIN bus, or any number of bit times for standard RS-232 devices.

Oscillator Startup Time

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

WUE Bit

The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared by hardware by a rising edge on RX/DT. The interrupt condition is then cleared by software by reading the RCREGx 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 17-7: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION

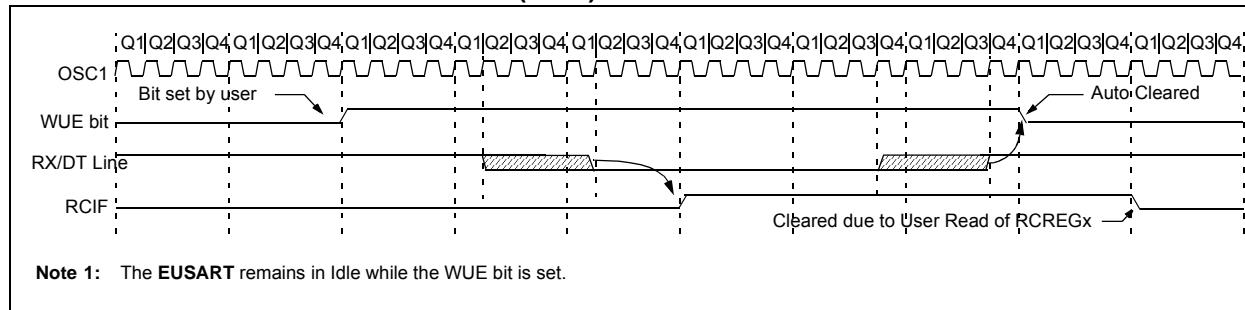
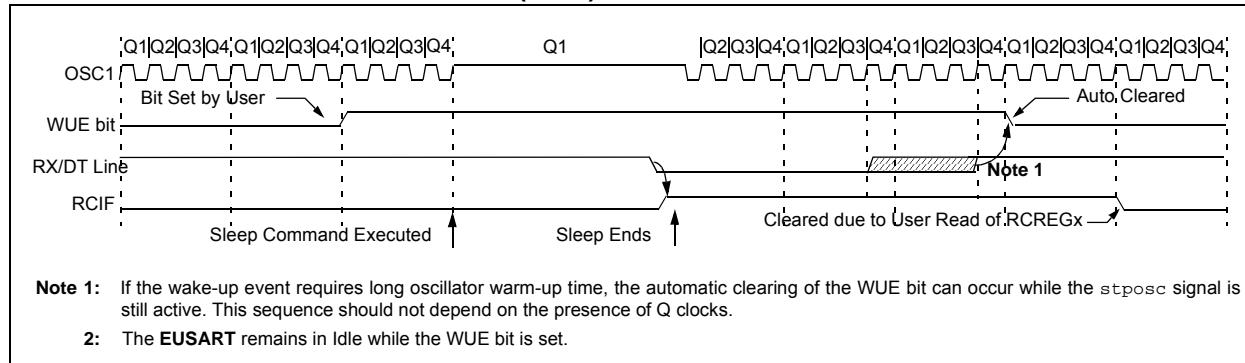


FIGURE 17-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP



17.4.4 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 TXSTAx register. The Break character transmission is then initiated by a write to the TXREGx. The value of data written to TXREGx 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 TXSTAx register indicates when the transmit operation is active or Idle, just as it does during normal transmission. See [Figure 17-9](#) for the timing of the Break character sequence.

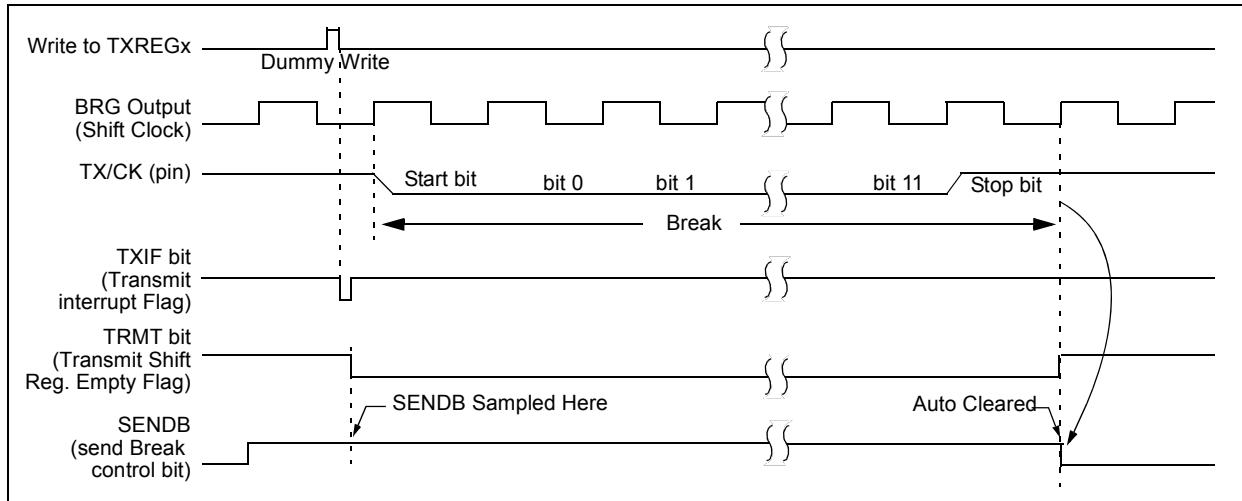
17.4.4.1 Break and Sync Transmit Sequence

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

FIGURE 17-9: SEND BREAK CHARACTER SEQUENCE



17.4.5 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 RCSTAx register and the Received data as indicated by RCREGx. 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
- RCREGx = 00h

The second method uses the Auto-Wake-up feature described in [Section 17.4.3 "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 BAUDCONx register before placing the EUSART in Sleep mode.

17.5 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.

17.5.1 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 TXSTAx register configures the device for synchronous operation. Setting the CSRC bit of the TXSTAx register configures the device as a master. Clearing the SREN and CREN bits of the RCSTAx register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTAx 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.

17.5.1.1 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.

17.5.1.2 Clock Polarity

A clock polarity option is provided for Microwire compatibility. Clock polarity is selected with the TXCKP bit of the BAUDCONx register. Setting the TXCKP bit sets the clock Idle state as high. When the TXCKP 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 TXCKP bit sets the Idle state as low. When the TXCKP bit is cleared, the data changes on the rising edge of each clock and is sampled on the falling edge of each clock.

17.5.1.3 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 TXREGx register. If the TSR still contains all or part of a previous character the new character data is held in the TXREGx 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 TXREGx is immediately transferred to the TSR. The transmission of the character commences immediately following the transfer of the data to the TSR from the TXREGx.

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.

17.5.1.4 Data Polarity

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

PIC18(L)F2X/45K50

17.5.1.5 Synchronous Master Transmission Setup

1. Initialize the SPBRGH_x, SPBRG_x register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see [Section 17.4 “EUSART Baud Rate Generator \(BRG\)](#)”).
2. Set the RX/DT and TX/CK TRIS controls to ‘1’.
3. 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.

4. Disable Receive mode by clearing bits SREN and CREN.
5. Enable Transmit mode by setting the TXEN bit.
6. If 9-bit transmission is desired, set the TX9 bit.
7. If interrupts are desired, set the TXIE, GIE/GIEH and PEIE/GIEL interrupt enable bits.
8. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
9. Start transmission by loading data to the TXREG_x register.

FIGURE 17-10: SYNCHRONOUS TRANSMISSION

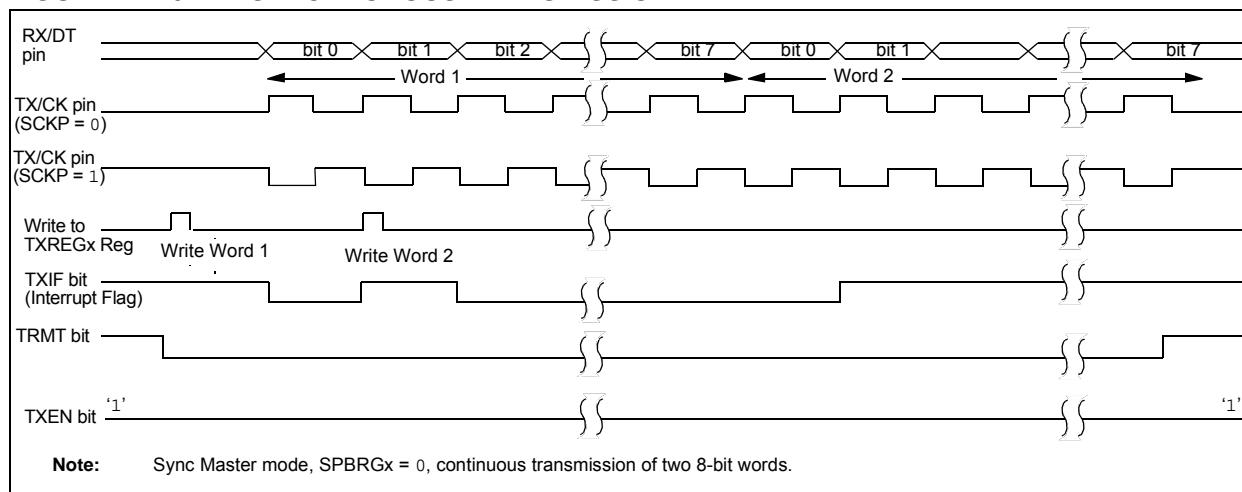


FIGURE 17-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

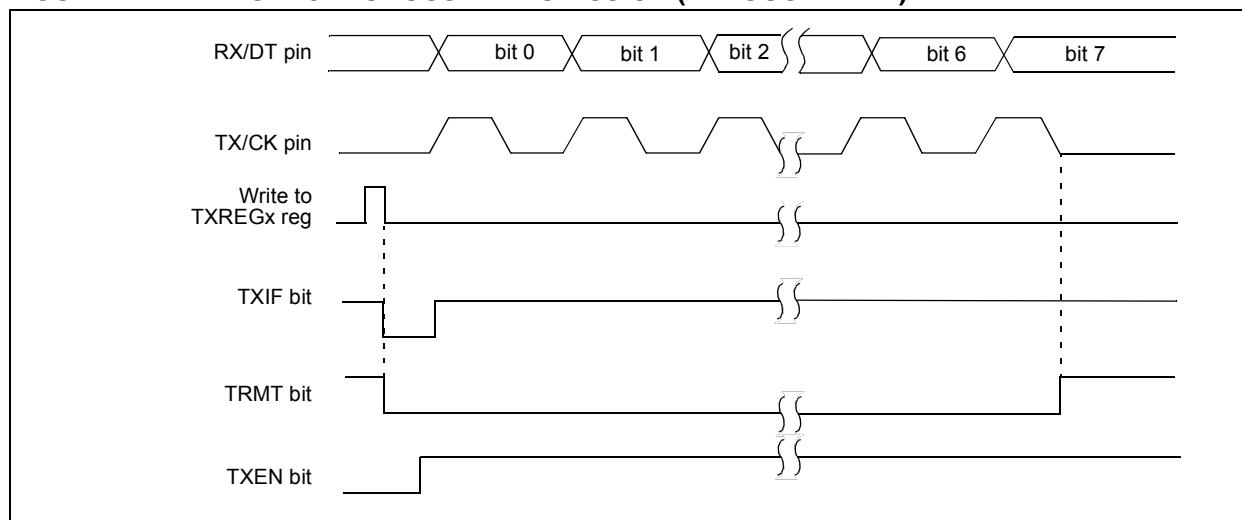


TABLE 17-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|---------------------------------------|-----------|--------|--------|--------|--------|--------|--------|------------------|
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 270 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 269 |
| SPBRG1 | EUSART Baud Rate Generator, Low Byte | | | | | | | | — |
| SPBRGH1 | EUSART Baud Rate Generator, High Byte | | | | | | | | — |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 149 |
| TXREG1 | EUSART Transmit Register | | | | | | | | — |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDDB | BRGH | TRMT | TX9D | 268 |

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

17.5.1.6 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 RCSTAx register) or the Continuous Receive Enable bit (CREN of the RCSTAx 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 RCREGx. The RCIF bit remains set as long as there are un-read characters in the receive FIFO.

17.5.1.7 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.

17.5.1.8 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 RCREGx is read to access the FIFO. When this happens the OERR bit of the RCSTAx 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 RCREGx.

If the overrun occurred when the CREN bit is set then the error condition is cleared by either clearing the CREN bit of the RCSTAx register or by clearing the SPEN bit which resets the EUSART.

17.5.1.9 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTAx register is set the EUSART will shift 9-bits into the RSR for each character received. The RX9D bit of the RCSTAx 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 RCREGx.

17.5.1.10 Synchronous Master Reception Setup

1. Initialize the SPBRGH_x, SPBRG_x register pair for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Set the RX/DT and TX/CK TRIS controls to '1'.
3. 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.
4. Ensure bits CREN and SREN are clear.
5. If using interrupts, set the GIE/GIEH and PEIE/GIEL bits of the INTCON register and set RCIE.
6. If 9-bit reception is desired, set bit RX9.
7. Start reception by setting the SREN bit or for continuous reception, set the CREN bit.
8. 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.
9. Read the RCSTAx register to get the ninth bit (if enabled) and determine if any error occurred during reception.
10. Read the 8-bit received data by reading the RCREGx register.
11. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTAx register or by clearing the SPEN bit which resets the EUSART.

FIGURE 17-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)

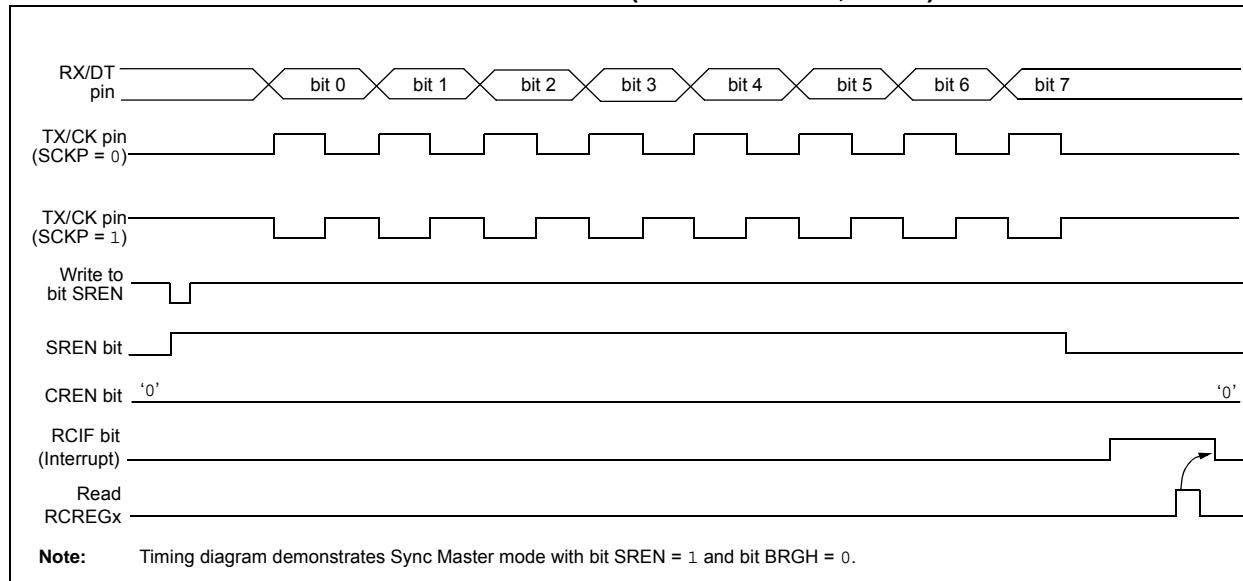


TABLE 17-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|---------------------------------------|-----------|--------|--------|-------|--------|--------|--------|------------------|
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 270 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 |
| RCREG1 | EUSART Receive Register | | | | | | | | — |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 269 |
| SPBRG1 | EUSART Baud Rate Generator, Low Byte | | | | | | | | — |
| SPBRGH1 | EUSART Baud Rate Generator, High Byte | | | | | | | | — |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 268 |

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

17.5.2 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 TXSTAx register configures the device for synchronous operation. Clearing the CSRC bit of the TXSTAx register configures the device as a slave. Clearing the SREN and CREN bits of the RCSTAx register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTAx 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.

17.5.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see [Section 17.5.1.3 "Synchronous Master Transmission"](#)), except in the case of the Sleep mode.

If two words are written to the TXREGx 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 TXREGx register.
3. The TXIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXREGx register will transfer the second character to the TSR and the TXIF bit will now be set.
5. If the PEIE/GIEL and TXIE bits are set, the interrupt will wake the device from Sleep and execute the next instruction. If the GIE/GIEH bit is also set, the program will call the Interrupt Service Routine.

17.5.2.2 Synchronous Slave Transmission Setup:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Set the RX/DT and TX/CK TRIS controls to '1'.
3. Clear the CREN and SREN bits.
4. If using interrupts, ensure that the GIE/GIEH and PEIE/GIEL bits of the INTCON register are set and set the TXIE bit.
5. If 9-bit transmission is desired, set the TX9 bit.
6. Enable transmission by setting the TXEN bit.
7. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
8. Start transmission by writing the Least Significant eight bits to the TXREGx register.

TABLE 17-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|---------------------------------------|-----------|--------|--------|-------------------|--------|--------|--------|------------------|
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 270 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSP1F | CCP1IF | TMR2IF | TMR1IF | 117 |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 269 |
| SPBRG1 | EUSART Baud Rate Generator, Low Byte | | | | | | | | — |
| SPBRGH1 | EUSART Baud Rate Generator, High Byte | | | | | | | | — |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 149 |
| TXREG1 | EUSART Transmit Register | | | | | | | | — |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SEND _B | BRGH | TRMT | TX9D | 268 |

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

PIC18(L)F2X/45K50

17.5.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical ([Section 17.5.1.6 “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 RCREGx 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/GIEH bit is also set, the program will branch to the interrupt vector.

17.5.2.4 Synchronous Slave Reception Setup

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Set the RX/DT and TX/CK TRIS controls to ‘1’.
3. If using interrupts, ensure that the GIE/GIEH and PEIE/GIEL bits of the INTCON register are set and set the RCIE bit.
4. If 9-bit reception is desired, set the RX9 bit.
5. Set the CREN bit to enable reception.
6. The RCIF bit will be set when reception is complete. An interrupt will be generated if the RCIE bit was set.
7. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCSTAx register.
8. Retrieve the eight Least Significant bits from the receive FIFO by reading the RCREGx register.
9. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTAx register or by clearing the SPEN bit which resets the EUSART.

TABLE 17-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|---------------------------------------|-----------|--------|--------|-------|--------|--------|--------|---------------------|
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 270 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 |
| PMD0 | — | UARTMD | USBMD | ACTMD | — | TMR3MD | TMR2MD | TMR1MD | 61 |
| RCREG1 | EUSART Receive Register | | | | | | | | — |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 269 |
| SPBRG1 | EUSART Baud Rate Generator, Low Byte | | | | | | | | — |
| SPBRGH1 | EUSART Baud Rate Generator, High Byte | | | | | | | | — |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 268 |

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

18.0 ANALOG-TO-DIGITAL CONVERTER (ADC) MODULE

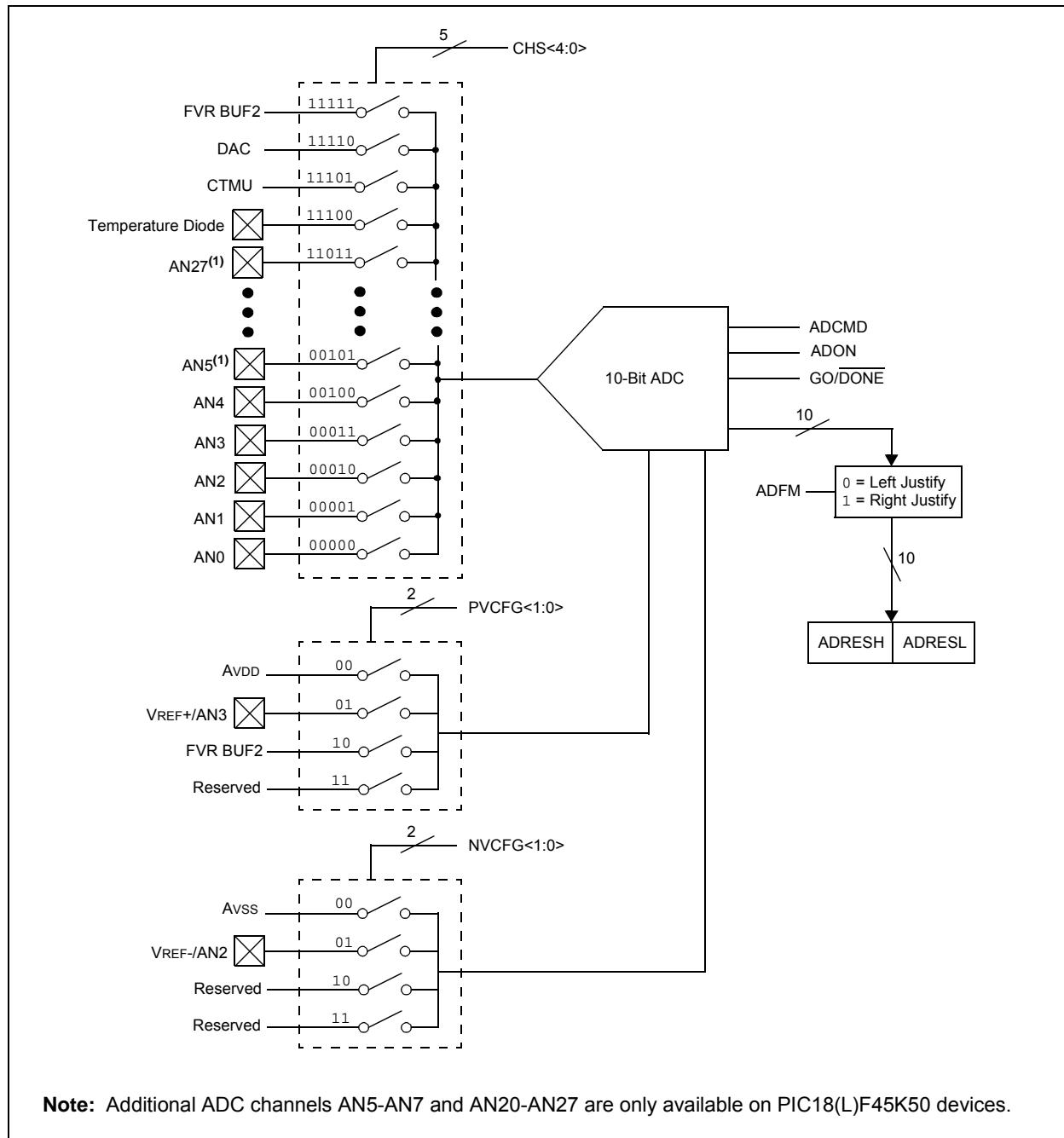
The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESL and ADRESH).

The ADC voltage reference is software selectable to either VDD or a voltage applied to the external reference pins.

The ADC can generate an interrupt upon completion of a conversion. This interrupt can be used to wake-up the device from Sleep.

Figure 18-1 shows the block diagram of the ADC.

FIGURE 18-1: ADC BLOCK DIAGRAM



18.1 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

18.1.1 PORT CONFIGURATION

The ANSELx and TRISx registers configure the A/D port pins. Any port pin needed as an analog input should have its corresponding ANSx bit set to disable the digital input buffer and TRISx bit set to disable the digital output driver. If the TRISx bit is cleared, the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the ANSx bits and the TRIS bits.

- Note 1:** When reading the PORT register, all pins with their corresponding ANSx bit set read as cleared (a low level). However, analog conversion of pins configured as digital inputs (ANSx bit cleared and TRISx bit set) will be accurately converted.
- 2:** Analog levels on any pin with the corresponding ANSx bit cleared may cause the digital input buffer to consume current out of the device's specification limits.
- 3:** The PBADEN bit in Configuration Register 3H configures PORTB pins to reset as analog or digital pins by controlling how the bits in ANSELB are reset.

18.1.2 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 18.2 "ADC Operation"](#) for more information.

18.1.3 ADC VOLTAGE REFERENCE

The PVCFG<1:0> and NVCFG<1:0> bits of the ADCON1 register provide independent control of the positive and negative voltage references.

The positive voltage reference can be:

- VDD
- the fixed voltage reference (FVR BUF2)
- an external voltage source (VREF+)

The negative voltage reference can be:

- Vss
- an external voltage source (VREF-)

18.1.4 SELECTING AND CONFIGURING ACQUISITION TIME

The ADCON2 register allows the user to select an acquisition time that occurs each time the GO/DONE bit is set.

Acquisition time is set with the ACQT<2:0> bits of the ADCON2 register. Acquisition delays cover a range of 2 to 20 TAD. When the GO/DONE bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Since the acquisition time is programmed, there is no need to wait for an acquisition time between selecting a channel and setting the GO/DONE bit.

Manual acquisition is selected when ACQT<2:0> = 000. When the GO/DONE bit is set, sampling is stopped and a conversion begins. The user is responsible for ensuring the required acquisition time has passed between selecting the desired input channel and setting the GO/DONE bit. This option is also the default Reset state of the ACQT<2:0> bits and is compatible with devices that do not offer programmable acquisition times.

In either case, when the conversion is completed, the GO/DONE bit is cleared, the ADIF flag is set and the A/D begins sampling the currently selected channel again. When an acquisition time is programmed, there is no indication of when the acquisition time ends and the conversion begins.

18.1.5 CONVERSION CLOCK

The source of the conversion clock is software selectable via the ADCS bits of the ADCON2 register. There are seven possible clock options:

- Fosc/2
- Fosc/4
- Fosc/8
- Fosc/16
- Fosc/32
- Fosc/64
- FRC (dedicated internal oscillator)

The time to complete one bit conversion is defined as TAD. One full 10-bit conversion requires 11 TAD periods as shown in [Figure 18-3](#).

For correct conversion, the appropriate TAD specification must be met. See A/D conversion requirements in [Table 29-33](#) for more information. [Table 18-1](#) gives examples of appropriate ADC clock selections.

Note: Unless using the FRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.

18.1.6 INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital Conversion. The ADC interrupt enable is the ADIE bit in the PIE1 register and the interrupt priority is the ADIP bit in the IPR1 register. The ADC interrupt flag is the ADIF bit in the PIR1 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.

TABLE 18-1: ADC CLOCK PERIOD (TAD) VS. DEVICE OPERATING FREQUENCIES

| AD Clock Period (TAD) | | Device Frequency (Fosc) | | | |
|-----------------------|-----------|-------------------------|-------------------------|-------------------------|-------------------------|
| ADC Clock Source | ADCS<2:0> | 48 MHz | 16 MHz | 4 MHz | 1 MHz |
| Fosc/2 | 000 | 41.17 ns ⁽²⁾ | 125 ns ⁽²⁾ | 500 ns ⁽²⁾ | 2.0 µs |
| Fosc/4 | 100 | 83.3 ns ⁽²⁾ | 250 ns ⁽²⁾ | 1.0 µs | 4.0 µs ⁽³⁾ |
| Fosc/8 | 001 | 166.7 ns ⁽²⁾ | 500 ns ⁽²⁾ | 2.0 µs | 8.0 µs ⁽³⁾ |
| Fosc/16 | 101 | 333.3 ns ⁽²⁾ | 1.0 µs | 4.0 µs ⁽³⁾ | 16.0 µs ⁽³⁾ |
| Fosc/32 | 010 | 666.7 ns ⁽²⁾ | 2.0 µs | 8.0 µs ⁽³⁾ | 32.0 µs ⁽³⁾ |
| Fosc/64 | 110 | 1.3 µs | 4.0 µs ⁽³⁾ | 16.0 µs ⁽³⁾ | 64.0 µs ⁽³⁾ |
| F _{RC} | 011 | 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 the recommended range.

Note 1: The FRC source has a typical TAD time of 1.7 µs.

2: These values violate the minimum required TAD time.

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

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

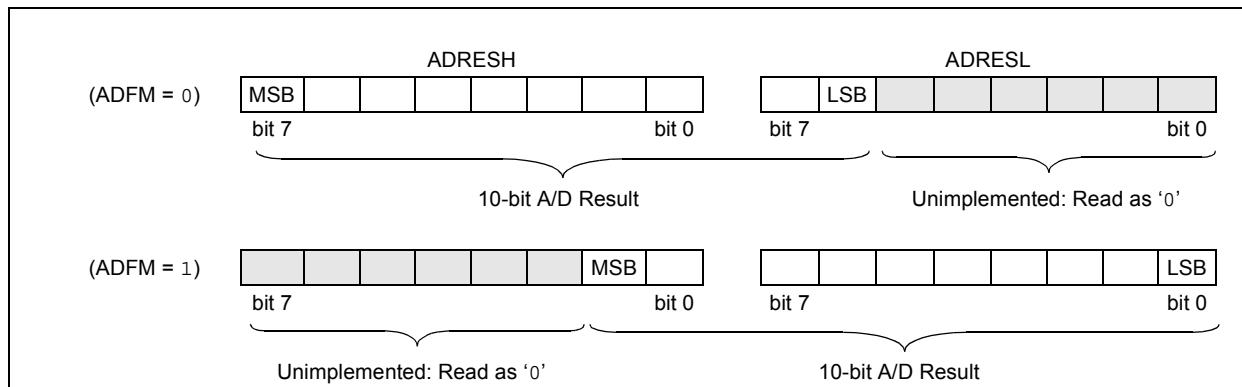
PIC18(L)F2X/45K50

18.1.7 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 18-2 shows the two output formats.

FIGURE 18-2: 10-BIT A/D CONVERSION RESULT FORMAT



18.2 ADC Operation

18.2.1 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 18-3 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 18-4 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 18.2.10 "A/D Conversion Procedure"](#).

FIGURE 18-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)

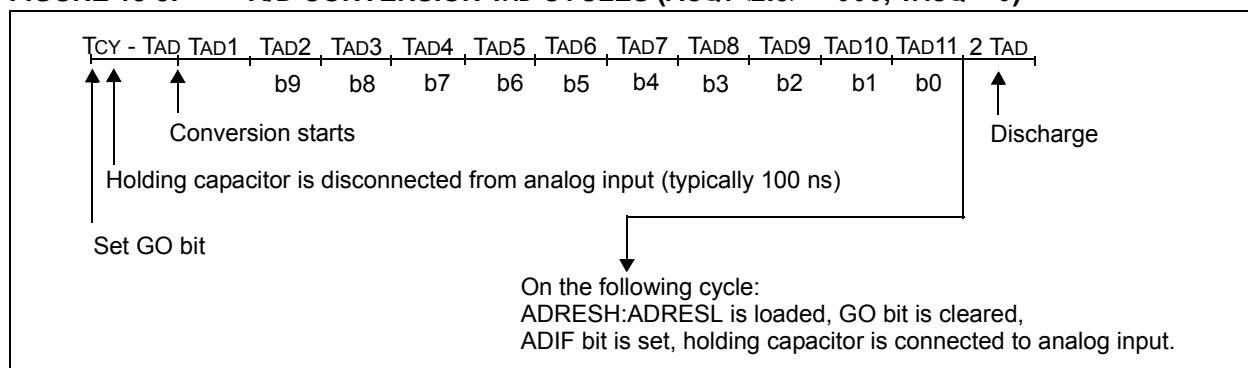
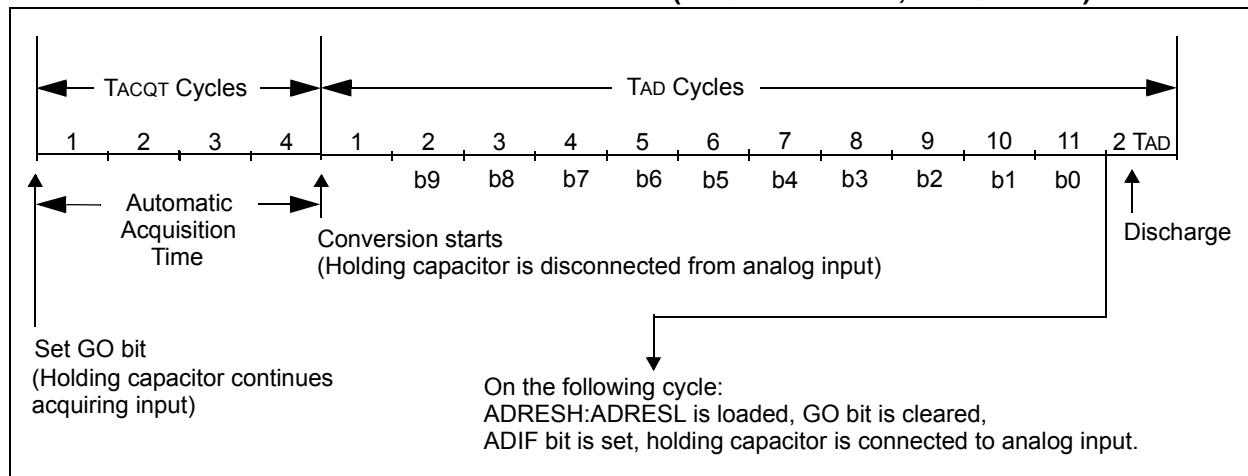


FIGURE 18-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)



18.2.2 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

18.2.3 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.

18.2.4 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 not be updated with the partially complete Analog-to-Digital conversion sample. Instead, the ADRESH:ADRESL register pair will retain the value of the previous conversion.

Note: A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated.

18.2.5 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.

18.2.6 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.

18.2.7 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.

18.2.8 SPECIAL EVENT TRIGGER

Two Special Event Triggers are available to start an A/D conversion: CTMU and CCP2. The Special Event Trigger source is selected using the TRIGSEL bit in ADCON1.

When TRIGSEL = 0, the CCP2 module is selected as the Special Event Trigger source. To enable the Special Event Trigger in the CCP module, set CCP2M_{3:0} = 1011, in the CCP2CON register.

When TRIGSEL = 1, the CTMU module is selected. The CTMU module requires that the CTTRIG bit in CTMUCONH is set to enable the Special Event Trigger.

In addition to the TRIGSEL bit, the following steps are required to start an A/D conversion:

- The A/D module must be enabled (ADON = 1)
- The appropriate analog input channel selected
- The minimum acquisition period set one of these ways:
 - Timing provided by the user
 - Selection made of an appropriate TACQ time

With these conditions met, the trigger sets the GO/DONE bit and the A/D acquisition starts.

If the A/D module is not enabled (ADON = 0), the module ignores the Special Event Trigger.

18.2.9 PERIPHERAL MODULE DISABLE

When a peripheral module is not used or inactive, the module can be disabled by setting the Module Disable bit in the PMD registers. This will reduce power consumption to an absolute minimum. Setting the PMD bits holds the module in Reset and disconnects the module's clock source. The Module Disable bit for the ADC module is ADCMD in the PMD1 Register. See [Section 4.0 "Power-Managed Modes"](#) for more information.

18.2.10 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/DONE bit.
6. Wait for ADC conversion to complete by one of the following:
 - Polling the GO/DONE bit
 - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result
8. Clear the ADC interrupt flag (required if interrupt is enabled).

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.
2: Software delay required if ACQT bits are set to zero delay. See [Section 18.4 “A/D Acquisition Requirements”](#).

EXAMPLE 18-1: A/D CONVERSION

```
;This code block configures the ADC  
;for polling, Vdd and Vss as reference, Frc  
clock and AN0 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      TRISA,0     ;Set RA0 to input  
BSF      ANSEL,0     ;Set RA0 to analog  
MOVLW    B'00000001' ;AN0, ADC on  
MOVWF    ADCON0      ;  
BSF      ADCON0,GO   ;Start conversion  
ADCPoll:  
BTFSR   ADCON0,GO    ;Is conversion done?  
BRA     ADCPoll      ;No, test again  
; Result is complete - store 2 MSbits in  
; RESULTHI and 8 LSbits in RESULTLO  
MOVFF    ADRESH,RESULTHI  
MOVFF    ADRESL,RESULTLO
```

PIC18(L)F2X/45K50

18.3 Register Definitions: ADC Control

Note: Analog pin control is determined by the ANSELx registers (see [Register 11-2](#))

REGISTER 18-1: ADCON0: A/D CONTROL REGISTER 0

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-------|----------|-------|---------|-------|-------|
| — | | | CHS<4:0> | | GO/DONE | | ADON |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

bit 6-2 **CHS<4:0>: Analog Channel Select bits**

00000 = AN0

00001 = AN1

00010 = AN2

00011 = AN3

00100 = AN4

00101 = AN5⁽¹⁾

00110 = AN6⁽¹⁾

00111 = AN7⁽¹⁾

01000 = AN8

01001 = AN9

01010 = AN10

01011 = AN11

01100 = AN12

01101 = AN13

01110 = AN14

01111 = AN15

10000 = AN16

10001 = AN17

10010 = AN18

10011 = AN19

10100 = AN20⁽¹⁾

10101 = AN21⁽¹⁾

10110 = AN22⁽¹⁾

10111 = AN23⁽¹⁾

11000 = AN24⁽¹⁾

11001 = AN25⁽¹⁾

11010 = AN26⁽¹⁾

11011 = AN27⁽¹⁾

11100 = Temperature Diode

11101 = CTMU

11110 = DAC

11111 = FVR BUF2 (1.024V/2.048V/4.096V Fixed Voltage Reference)⁽²⁾

bit 1 **GO/DONE:** A/D Conversion Status bit

1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.

This bit is automatically cleared by hardware when the A/D conversion has completed.

0 = A/D conversion completed/not in progress

bit 0 **ADON:** ADC Enable bit

1 = ADC is enabled

0 = ADC is disabled and consumes no operating current

Note 1: Available on PIC18(L)F45K50 devices only.

2: Allow greater than 15 μ s acquisition time when measuring the Fixed Voltage Reference.

REGISTER 18-2: ADCON1: A/D CONTROL REGISTER 1

| R/W-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---------|-----|-----|-----|------------|------------|-------|-------|
| TRIGSEL | — | — | — | PVCFG<1:0> | NVCFG<1:0> | | |
| 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 **TRIGSEL:** Special Trigger Select bit
 1 = Selects the special trigger from CTMU
 0 = Selects the special trigger from CCP2

bit 6-4 **Unimplemented:** Read as '0'

bit 3-2 **PVCFG<1:0>:** Positive Voltage Reference Configuration bits
 00 = A/D VREF+ connected to internal signal, AVDD
 01 = A/D VREF+ connected to external pin, VREF+
 10 = A/D VREF+ connected to internal signal, FVR BUF2
 11 = Reserved

bit 1-0 **NVCFG<1:0>:** Negative Voltage Reference Configuration bits
 00 = A/D VREF- connected to internal signal, AVss
 01 = A/D VREF- connected to external pin, VREF-
 10 = Reserved
 11 = Reserved

PIC18(L)F2X/45K50

REGISTER 18-3: 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 | — | | ACQT<2:0> | | | ADCS<2:0> | |
| 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 18-4: 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 |
|------------|-------|-------|-------|-------|-------|-------|-------|
| ADRES<9:2> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

-n = Value at POR

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7-0 **ADRES<9:2>**: ADC Result Register bits
Upper eight bits of 10-bit conversion result

REGISTER 18-5: 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 |
|-------|------------|-------|-------|-------|-------|-------|-------|
| | ADRES<1:0> | r | r | r | r | r | r |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

-n = Value at POR

r = Reserved bit

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **ADRES<1:0>**: ADC Result Register bits
Lower two bits of 10-bit conversion result

bit 5-0 **Reserved**: Do not use.

REGISTER 18-6: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 1

| R/W-x |
|-------|-------|-------|-------|-------|-------|-------|------------|
| r | r | r | r | r | r | r | ADRES<9:8> |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

-n = Value at POR

r = Reserved bit

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7-2 **Reserved**: Do not use.

bit 1-0 **ADRES<9:8>**: ADC Result Register bits
Upper two bits of 10-bit conversion result

REGISTER 18-7: 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 |
|------------|-------|-------|-------|-------|-------|-------|-------|
| ADRES<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

-n = Value at POR

W = Writable bit

'1' = Bit is set

U = Unimplemented bit, read as '0'

'0' = Bit is cleared

x = Bit is unknown

bit 7-0 **ADRES<7:0>**: ADC Result Register bits
Lower eight bits of 10-bit conversion result

18.4 A/D Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (C_{HOLD}) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in [Figure 18-5](#). The source impedance (R_s) and the internal sampling switch (R_{ss}) impedance directly affect the time required to charge the capacitor C_{HOLD} . The sampling switch (R_{ss}) impedance varies over the device voltage (V_{DD}), see [Figure 18-5](#). **The maximum recommended impedance for analog sources is 10 k Ω .** 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 18-1](#) 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 18-1: ACQUISITION TIME EXAMPLE

Assumptions: Temperature = 50°C and external impedance of 10k Ω 3.0V V_{DD}

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \\ &= 5\mu s + T_C + [(Temperature - 25^\circ C)(0.05\mu s/\text{ }^\circ C)] \end{aligned}$$

The value for T_C can be approximated with the following equations:

$$\begin{aligned} V_{APPLIED} \left(1 - \frac{1}{2047} \right) &= V_{CHOLD} && ;[1] V_{CHOLD} \text{ charged to within 1/2 lsb} \\ V_{APPLIED} \left(1 - e^{-\frac{T_C}{RC}} \right) &= V_{CHOLD} && ;[2] V_{CHOLD} \text{ charge response to } V_{APPLIED} \\ V_{APPLIED} \left(1 - e^{-\frac{T_C}{RC}} \right) &= V_{APPLIED} \left(1 - \frac{1}{2047} \right) && ;\text{combining [1] and [2]} \end{aligned}$$

Solving for T_C :

$$\begin{aligned} T_C &= -C_{HOLD}(R_{IC} + R_{SS} + R_s) \ln(1/2047) \\ &= -13.5\text{pF}(1k\Omega + 700\Omega + 10k\Omega) \ln(0.0004885) \\ &= 1.20\mu s \end{aligned}$$

Therefore:

$$\begin{aligned} T_{ACQ} &= 5\mu s + 1.20\mu s + [(50^\circ C - 25^\circ C)(0.05\mu s/\text{ }^\circ C)] \\ &= 7.45\mu s \end{aligned}$$

Note 1: The reference voltage (V_{REF}) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (C_{HOLD}) is discharged after each conversion.

3: The maximum recommended impedance for analog sources is 10 k Ω . This is required to meet the pin leakage specification.

FIGURE 18-5: ANALOG INPUT MODEL

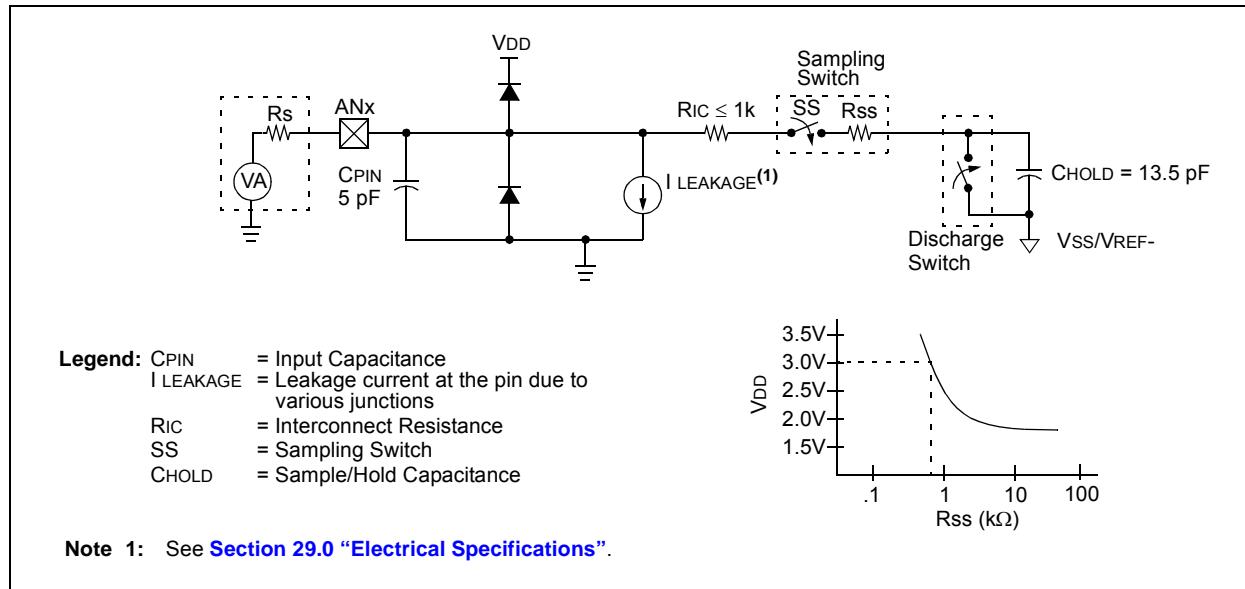
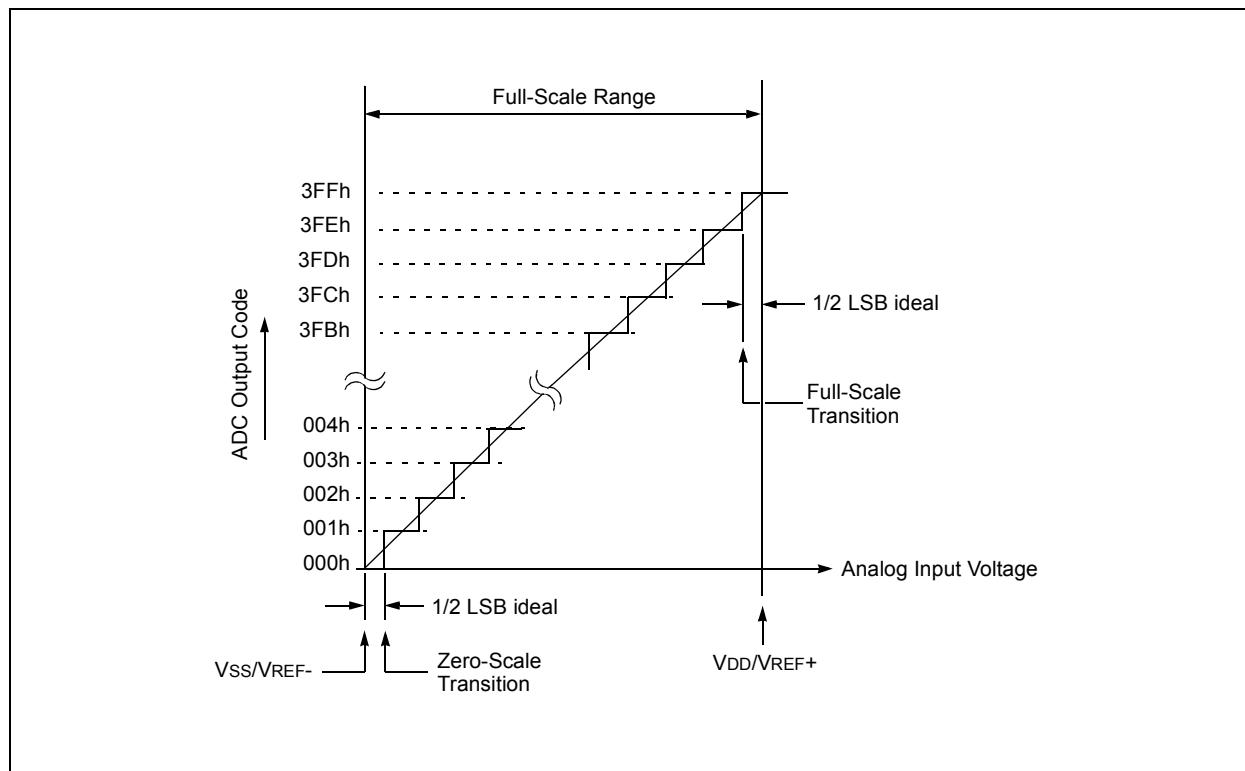


FIGURE 18-6: ADC TRANSFER FUNCTION



PIC18(L)F2X/45K50

TABLE 18-2: REGISTERS ASSOCIATED WITH A/D OPERATION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page | | | | | | | | |
|-----------------------|-----------------------|-----------|-----------|--------|------------|-----------------------|-----------------------|-----------------------|------------------|--|--|--|--|--|--|--|--|
| ADCON0 | — | | CHS<4:0> | | | | GO/DONE | ADON | 294 | | | | | | | | |
| ADCON1 | TRIGSEL | — | — | — | PVCFG<1:0> | | NVCFG<1:0> | | 295 | | | | | | | | |
| ADCON2 | ADFM | — | ACQT<2:0> | | | ADCS<2:0> | | | 296 | | | | | | | | |
| ADRESH | A/D Result, High Byte | | | | | | | | 297 | | | | | | | | |
| ADRESL | A/D Result, Low Byte | | | | | | | | 297 | | | | | | | | |
| ANSELA | — | — | ANSA5 | — | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 147 | | | | | | | | |
| ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 148 | | | | | | | | |
| ANSELC | ANSC7 | ANSC6 | — | — | — | ANSC2 | — | — | 148 | | | | | | | | |
| ANSELD ⁽¹⁾ | ANSD7 | ANSD6 | ANSD5 | ANSD4 | ANSD3 | ANSD2 | ANSD1 | ANSD0 | 148 | | | | | | | | |
| ANSELE ⁽¹⁾ | — | — | — | — | — | ANSE2 | ANSE1 | ANSE0 | 149 | | | | | | | | |
| CTMUCONH | CTMUEN | — | CTMUSIDL | TGEN | EDGEN | EDGSEQEN | IDISSEN | CTTRIG | 322 | | | | | | | | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 | | | | | | | | |
| IPR1 | ACTIP | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 123 | | | | | | | | |
| PIE1 | ACTIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 120 | | | | | | | | |
| PIR1 | ACTIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 117 | | | | | | | | |
| PMD1 | — | MSSPMMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | 62 | | | | | | | | |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 149 | | | | | | | | |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 | | | | | | | | |
| TRISC | TRISC7 | TRISC6 | — | — | — | TRISC2 | TRISC1 | TRISC0 | 149 | | | | | | | | |
| TRISD ⁽¹⁾ | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 149 | | | | | | | | |
| TRISE | WPUE3 | — | — | — | — | TRISE2 ⁽¹⁾ | TRISE1 ⁽¹⁾ | TRISE0 ⁽¹⁾ | 149 | | | | | | | | |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used by this module.

Note 1: Available on PIC18(L)F45K50 devices.

TABLE 18-3: CONFIGURATION REGISTERS ASSOCIATED WITH THE ADC MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|-------|-------|-------|-------|--------|--------|------------------|
| CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 376 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used by the ADC module.

19.0 COMPARATOR MODULE

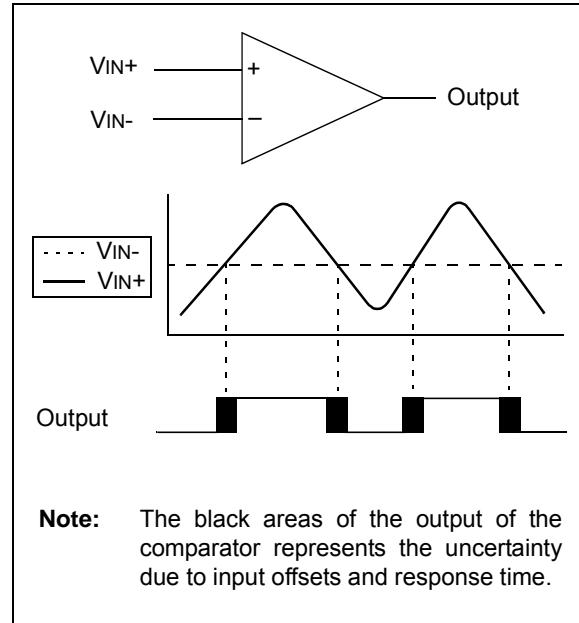
Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. The comparators are very useful mixed signal building blocks because they provide analog functionality independent of the program execution. 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

19.1 Comparator Overview

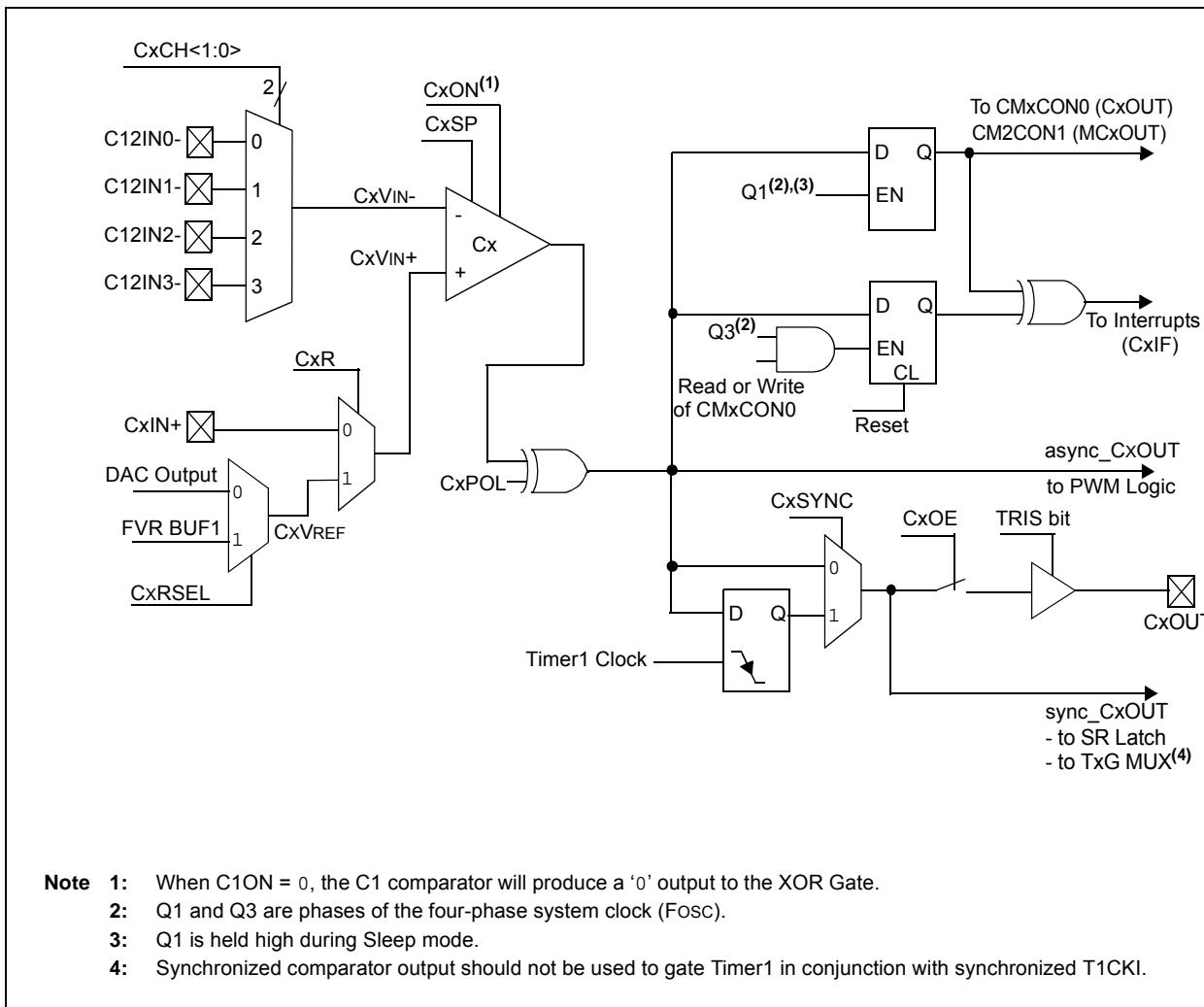
A single comparator is shown in [Figure 19-1](#) along with the relationship between the analog input levels and the digital output. When the analog voltage at V_{IN+} is less than the analog voltage at V_{IN-} , the output of the comparator is a digital low level. When the analog voltage at V_{IN+} is greater than the analog voltage at V_{IN-} , the output of the comparator is a digital high level.

FIGURE 19-1: SINGLE COMPARATOR



PIC18(L)F2X/45K50

FIGURE 19-2: COMPARATOR C1/C2 SIMPLIFIED BLOCK DIAGRAM



19.2 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 [Register 19-1](#)) contain the control and status bits for the following:

- Enable
- Input selection
- Reference selection
- Output selection
- Output polarity
- Speed selection

19.2.1 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.

19.2.2 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.

19.2.3 COMPARATOR REFERENCE SELECTION

Setting the CxR bit of the CMxCON0 register directs an internal voltage reference or an analog input pin to the non-inverting input of the comparator. See [Section 22.0 “Fixed Voltage Reference \(FVR\)”](#) for more information on the Internal Voltage Reference module.

19.2.4 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.

19.2.5 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 19-1](#) shows the output state versus input conditions, including polarity control.

TABLE 19-1: 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 |

19.2.6 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’.

19.3 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 29.0 “Electrical Specifications”](#) for more details.

19.4 Comparator Interrupt Operation

The comparator interrupt flag will 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 19-2). The first latch is updated with the comparator output value, when the CMxCON0 register is read or written. The value is latched on the third cycle of the system clock, also known as Q3. This first latch retains the comparator value until another read or write of the CMxCON0 register occurs or a Reset takes place. The second latch is updated with the comparator output value on every first cycle of the system clock, also known as Q1. When the output value of the comparator changes, the second latch is updated and the output values of both latches no longer match one another, resulting in a mismatch condition. The latch outputs are fed directly into the inputs of an exclusive-or gate. This mismatch condition is detected by the exclusive-or gate and sent to the interrupt circuitry. The mismatch condition will persist until the first latch value is updated by performing a read of the CMxCON0 register 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.

When the mismatch condition occurs, the comparator interrupt flag is set. The interrupt flag is triggered by the edge of the changing value coming from the exclusive-or gate. This means that the interrupt flag can be reset once it is triggered without the additional step of reading or writing the CMxCON0 register to clear the mismatch latches. 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 19-3 and 19-4.

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/GIEL and GIE/GIEH 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.

19.4.1 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 19-3: COMPARATOR INTERRUPT TIMING W/O CMxCON0 READ

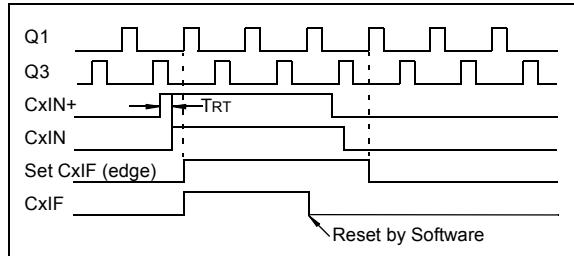
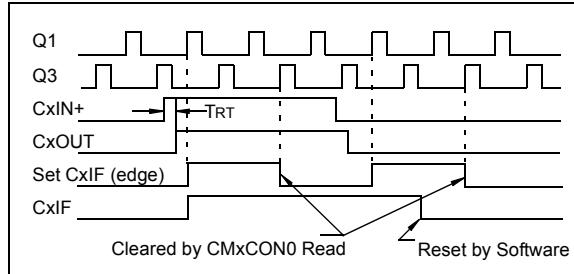


FIGURE 19-4: COMPARATOR INTERRUPT TIMING WITH CMxCON0 READ



- 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 μ s for bias settling then clear the mismatch condition and interrupt flags before enabling comparator interrupts.

19.5 Operation During Sleep

The comparator, if enabled before entering Sleep mode, remains active during Sleep mode. The additional current consumed by the comparator is shown separately in [Section 29.0 “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/GIEL bit of the INTCON register must be set. The instruction following the *SLEEP* instruction always executes following a wake from Sleep. If the GIE/GIEH bit of the INTCON register is also set, the device will then execute the Interrupt Service Routine.

19.6 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. Comparator Control Registers.

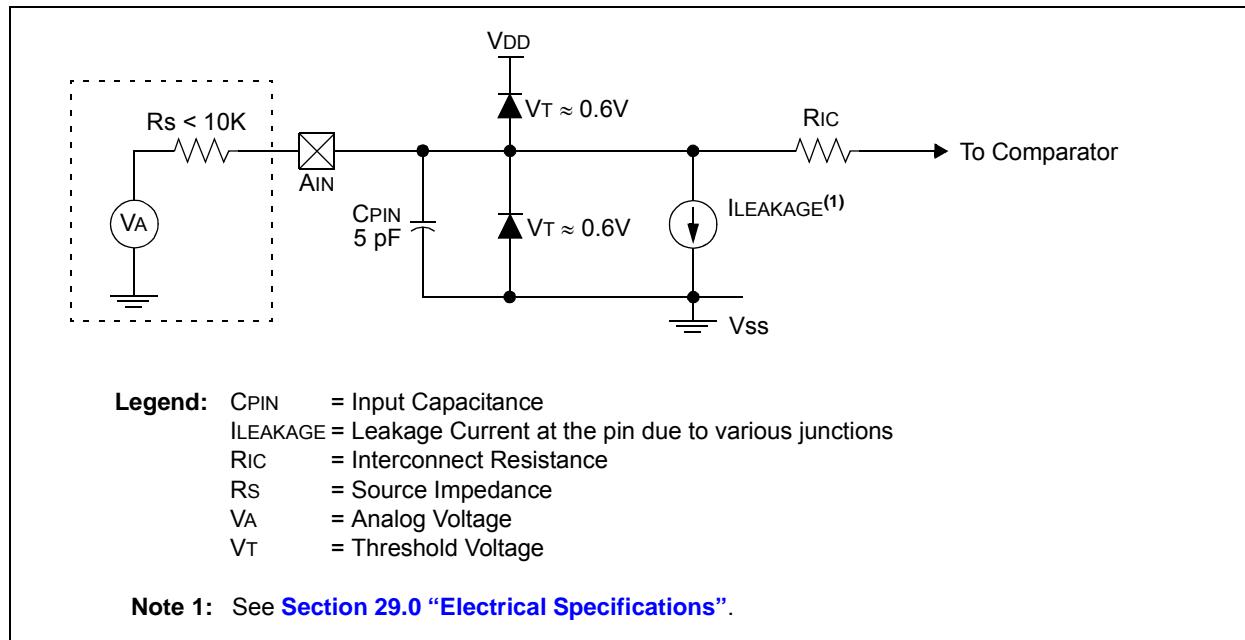
19.7 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in [Figure 19-5](#). 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 k Ω 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 19-5: ANALOG INPUT MODEL



19.8 Additional Comparator Features

There are four additional comparator features:

- Simultaneous read of comparator outputs
- Internal reference selection
- Hysteresis selection
- Output Synchronization

19.8.1 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.

19.8.2 INTERNAL REFERENCE SELECTION

There are two internal voltage references available to the non-inverting input of each comparator. One of these is the Fixed Voltage Reference (FVR) and the other is the variable Digital-to-Analog Converter (DAC). The CxRSEL bit of the CM2CON1 register determines which of these references is routed to the Comparator Voltage reference output (CxVREF). Further routing to the comparator is accomplished by the CxR bit of the CMxCON0 register. See [Section 22.0 “Fixed Voltage Reference \(FVR\)”](#) and [Figure 19-2](#) for more detail.

19.8.3 SYNCHRONIZING COMPARATOR OUTPUT TO TIMER1

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 falling edge of the Timer1 source clock. To prevent a race condition when gating Timer1 clock with the comparator output, Timer1 increments on the rising edge of its clock source, and the falling edge latches the comparator output. See the Comparator Block Diagram ([Figure 19-2](#)) and the Timer1 Block Diagram ([Figure 13-1](#)) for more information.

Note 1: The comparator synchronized output should not be used to gate the external Timer1 clock when the Timer1 synchronizer is enabled.

2: The Timer1 prescale should be set to 1:1 when synchronizing the comparator output as unexpected results may occur with other prescale values.

19.9 Register Definitions: Comparator Control

REGISTER 19-1: CMxCON0: COMPARATOR x CONTROL REGISTER

| R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | | |
|-------|-------|-------|-------|-------|-------|-----------|-------|--|--|
| CxON | CxOUT | CxOE | CxPOL | CxSP | CxR | CxCH<1:0> | | | |
| 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 | CxON: Comparator Cx Enable bit 1 = Comparator Cx is enabled 0 = Comparator Cx is disabled |
| bit 6 | CxOUT: Comparator Cx Output bit <u>If CxPOL = 1 (inverted polarity):</u> CxOUT = 0 when CxVIN+ > CxVIN- CxOUT = 1 when CxVIN+ < CxVIN- <u>If CxPOL = 0 (non-inverted polarity):</u> CxOUT = 1 when CxVIN+ > CxVIN- CxOUT = 0 when CxVIN+ < CxVIN- |
| bit 5 | CxOE: Comparator Cx Output Enable bit 1 = CxOUT is present on the CxOUT pin ⁽¹⁾ 0 = CxOUT is internal only |
| bit 4 | CxPOL: Comparator Cx Output Polarity Select bit 1 = CxOUT logic is inverted 0 = CxOUT logic is not inverted |
| bit 3 | CxSP: Comparator Cx Speed/Power Select bit 1 = Cx operates in normal power, higher speed mode 0 = Cx operates in low-power, low-speed mode |
| bit 2 | CxR: Comparator Cx Reference Select bit (non-inverting input) 1 = CxVIN+ connects to CxVREF output 0 = CxVIN+ connects to C12IN+ pin |
| bit 1-0 | CxCH<1:0>: Comparator Cx Channel Select bit 00 = C12IN0- pin of Cx connects to CxVIN- 01 = C12IN1- pin of Cx connects to CxVIN- 10 = C12IN2- pin of Cx connects to CxVIN- 11 = C12IN3- pin of Cx connects to CxVIN- |

Note 1: Comparator output requires the following three conditions: CxOE = 1, CxON = 1 and corresponding port TRIS bit = 0.

PIC18(L)F2X/45K50

REGISTER 19-2: CM2CON1: COMPARATOR 1 AND 2 CONTROL REGISTER

| R-0 | R-0 | R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 |
|--------|--------|--------|--------|-----|-----|--------|--------|
| MC1OUT | MC2OUT | C1RSEL | C2RSEL | — | — | 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 BUF1 routed to C1VREF input
 0 = DAC routed to C1VREF input
bit 4 **C2RSEL:** Comparator C2 Reference Select bit
 1 = FVR BUF1 routed to C2VREF input
 0 = DAC routed to C2VREF input
bit 3-2 **Reserved:** Maintain these bits clear
bit 1 **C1SYNC:** C1 Output Synchronous Mode bit
 1 = C1 output is synchronized to rising edge of TMR1 clock (T1CLK)
 0 = C1 output is asynchronous
bit 0 **C2SYNC:** C2 Output Synchronous Mode bit
 1 = C2 output is synchronized to rising edge of TMR1 clock (T1CLK)
 0 = C2 output is asynchronous

TABLE 19-2: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|----------|-----------|-----------|--------|-------------|--------|-----------|--------|------------------|
| ANSELA | — | — | ANS A5 | — | ANS A3 | ANS A2 | ANS A1 | ANS A0 | 147 |
| ANSELB | — | — | ANS B5 | ANS B4 | ANS B3 | ANS B2 | ANS B1 | ANS B0 | 148 |
| CM2CON1 | MC1OUT | MC2OUT | C1RSEL | C2RSEL | C1HYS | C2HYS | C1SYNC | C2SYNC | 308 |
| CM1CON0 | C1ON | C1OUT | C1OE | C1POL | C1SP | C1R | C1CH<1:0> | | 307 |
| CM2CON0 | C2ON | C2OUT | C2OE | C2POL | C2SP | C2R | C2CH<1:0> | | 307 |
| VREFCON1 | DACEN | DACLPS | DACOE | — | DACPSS<1:0> | — | DACNSS | | 334 |
| VREFCON2 | — | — | — | | DACR<4:0> | | | | 335 |
| VREFCON0 | FVREN | FVRST | FVRS<1:0> | — | — | — | — | | 331 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 |
| PMD1 | — | MSSPMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | 62 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 149 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used by the comparator module.

20.0 CHARGE TIME MEASUREMENT UNIT (CTMU)

The Charge Time Measurement Unit (CTMU) is a flexible analog module that provides accurate differential time measurement between pulse sources, as well as asynchronous pulse generation. By working with other on-chip analog modules, the CTMU can be used to precisely measure time, measure capacitance, measure relative changes in capacitance or generate output pulses with a specific time delay. The CTMU is ideal for interfacing with capacitive-based sensors.

The module includes the following key features:

- Up to 28⁽¹⁾ channels available for capacitive or time measurement input
- On-chip precision current source
- Four-edge input trigger sources
- Polarity control for each edge source
- Control of edge sequence
- Control of response to edges

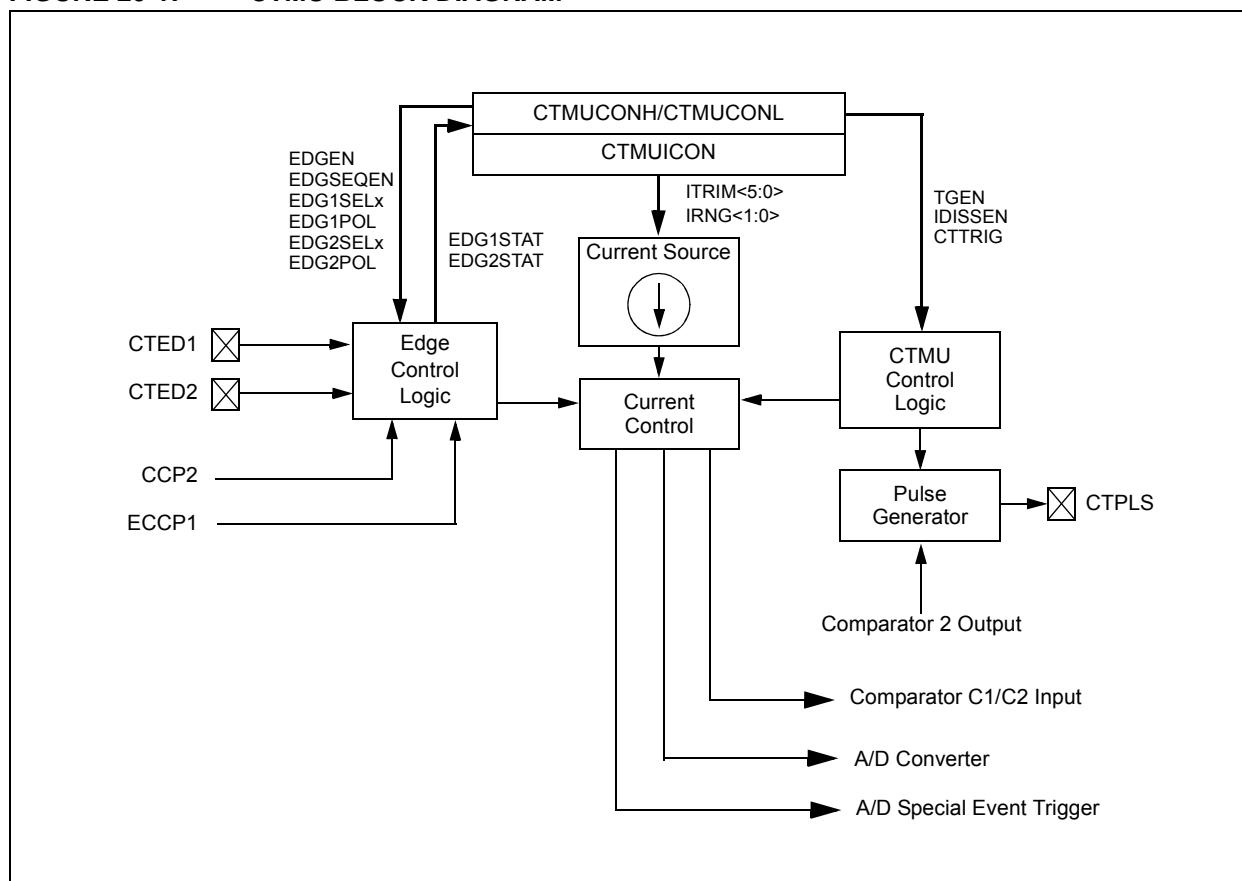
- High precision time measurement
- Time delay of external or internal signal asynchronous to system clock
- Accurate current source suitable for capacitive measurement

The CTMU works in conjunction with the A/D Converter to provide up to 28⁽¹⁾ channels for time or charge measurement, depending on the specific device and the number of A/D channels available. When configured for time delay, the CTMU is connected to the C12IN1- input of Comparator 2. The level-sensitive input edge sources can be selected from four sources: two external input pins (CTED1/CTED2) or the CCP1/CCP2 Special Event Triggers.

Figure 20-1 provides a block diagram of the CTMU.

Note 1: PIC18(L)F2XK50 devices have up to 17 channels available.

FIGURE 20-1: CTMU BLOCK DIAGRAM



20.1 CTMU Operation

The CTMU works by using a fixed current source to charge a circuit. The type of circuit depends on the type of measurement being made. In the case of charge measurement, the current is fixed and the amount of time the current is applied to the circuit is fixed. The amount of voltage read by the A/D is then a measurement of the capacitance of the circuit. In the case of time measurement, the current, as well as the capacitance of the circuit, is fixed. In this case, the voltage read by the A/D is then representative of the amount of time elapsed from the time the current source starts and stops charging the circuit.

If the CTMU is being used as a time delay, both capacitance and current source are fixed, as well as the voltage supplied to the comparator circuit. The delay of a signal is determined by the amount of time it takes the voltage to charge to the comparator threshold voltage.

20.1.1 THEORY OF OPERATION

The operation of the CTMU is based on the following equation for charge:

$$I = C \cdot \frac{dV}{dT}$$

More simply, the amount of charge measured in coulombs in a circuit is defined as current in amperes (I) multiplied by the amount of time in seconds that the current flows (t). Charge is also defined as the capacitance in farads (C) multiplied by the voltage of the circuit (V). It follows that:

$$I \cdot t = C \cdot V.$$

The CTMU module provides a constant, known current source. The A/D Converter is used to measure (V) in the equation, leaving two unknowns: capacitance (C) and time (t). The above equation can be used to calculate capacitance or time, by either the relationship using the known fixed capacitance of the circuit:

$$t = (C \cdot V)/I$$

or by:

$$C = (I \cdot t)/V$$

using a fixed time that the current source is applied to the circuit.

20.1.2 CURRENT SOURCE

At the heart of the CTMU is a precision current source, designed to provide a constant reference for measurements. The level of current is user selectable across three ranges or a total of two orders of magnitude, with the ability to trim the output in $\pm 2\%$ increments (nominal). The current range is selected by the IRNG_{<1:0>} bits (CTMUICON_{<1:0>}), with a value of '01' representing the lowest range.

Current trim is provided by the ITRIM_{<5:0>} bits (CTMUICON_{<7:2>}). These six bits allow trimming of the current source in steps of approximately 2% per step. Note that half of the range adjusts the current source positively and the other half reduces the current source. A value of '000000' is the neutral position (no change). A value of '100001' is the maximum negative adjustment (approximately -62%) and '011111' is the maximum positive adjustment (approximately +62%).

20.1.3 EDGE SELECTION AND CONTROL

CTMU measurements are controlled by edge events occurring on the module's two input channels. Each channel, referred to as Edge 1 and Edge 2, can be configured to receive input pulses from one of the edge input pins (CTED1 and CTED2), Timer1 or Output Compare Module 1. The input channels are level-sensitive, responding to the instantaneous level on the channel rather than a transition between levels. The inputs are selected using the EDG1SEL and EDG2SEL bit pairs (CTMUCONL_{<3:2>} and _{<6:5>}).

In addition to source, each channel can be configured for event polarity using the EDGE2POL and EDGE1POL bits (CTMUCONL_{<7,4>}). The input channels can also be filtered for an edge event sequence (Edge 1 occurring before Edge 2) by setting the EDGSEQEN bit (CTMUCONH_{<2>}).

20.1.4 EDGE STATUS

The CTMUCONL register also contains two status bits: EDG2STAT and EDG1STAT (CTMUCONL<1:0>). Their primary function is to show if an edge response has occurred on the corresponding channel. The CTMU automatically sets a particular bit when an edge response is detected on its channel. The level-sensitive nature of the input channels also means that the status bits become set immediately if the channel's configuration is changed and is the same as the channel's current state.

The module uses the edge status bits to control the current source output to external analog modules (such as the A/D Converter). Current is only supplied to external modules when only one (but not both) of the status bits is set, and shuts current off when both bits are either set or cleared. This allows the CTMU to measure current only during the interval between edges. After both status bits are set, it is necessary to clear them before another measurement is taken. Both bits should be cleared simultaneously, if possible, to avoid re-enabling the CTMU current source.

In addition to being set by the CTMU hardware, the edge status bits can also be set by software. This is also the user's application to manually enable or disable the current source. Setting either one (but not both) of the bits enables the current source. Setting or clearing both bits at once disables the source.

20.1.5 INTERRUPTS

The CTMU sets its interrupt flag (PIR3<3>) whenever the current source is enabled, then disabled. An interrupt is generated only if the corresponding interrupt enable bit (PIE3<3>) is also set. If edge sequencing is not enabled (i.e., Edge 1 must occur before Edge 2), it is necessary to monitor the edge Status bits and determine which edge occurred last and caused the interrupt.

20.2 CTMU Module Initialization

The following sequence is a general guideline used to initialize the CTMU module:

1. Select the current source range using the IRNG bits (CTMUICON<1:0>).
2. Adjust the current source trim using the ITRIM bits (CTMUICON<7:2>).
3. Configure the edge input sources for Edge 1 and Edge 2 by setting the EDG1SEL and EDG2SEL bits (CTMUCONL<3:2 and 6:5>).
4. Configure the input polarities for the edge inputs using the EDG1POL and EDG2POL bits (CTMUCONL<4,7>). The default configuration is for negative edge polarity (high-to-low transitions).
5. Enable edge sequencing using the EDGSEQEN bit (CTMUCONH<2>). By default, edge sequencing is disabled.
6. Select the operating mode (Measurement or Time Delay) with the TGEN bit. The default mode is Time/Capacitance Measurement.
7. Discharge the connected circuit by setting the IDISSEN bit (CTMUCONH<1>); after waiting a sufficient time for the circuit to discharge, clear IDISSEN.
8. Disable the module by clearing the CTMUEEN bit (CTMUCONH<7>).
9. Enable the module by setting the CTMUEEN bit.
10. Clear the Edge Status bits: EDG2STAT and EDG1STAT (CTMUCONL<1:0>).
11. Enable both edge inputs by setting the EDGEN bit (CTMUCONH<3>).

Depending on the type of measurement or pulse generation being performed, one or more additional modules may also need to be initialized and configured with the CTMU module:

- Edge Source Generation: In addition to the external edge input pins, both Timer1 and the Output Compare/PWM1 module can be used as edge sources for the CTMU.
- Capacitance or Time Measurement: The CTMU module uses the A/D Converter to measure the voltage across a capacitor that is connected to one of the analog input channels.
- Pulse Generation: When generating system clock independent output pulses, the CTMU module uses Comparator 2 and the associated comparator voltage reference.

20.3 Calibrating the CTMU Module

The CTMU requires calibration for precise measurements of capacitance and time, as well as for accurate time delay. If the application only requires measurement of a relative change in capacitance or time, calibration is usually not necessary. An example of this type of application would include a capacitive touch switch, in which the touch circuit has a baseline capacitance, and the added capacitance of the human body changes the overall capacitance of a circuit.

If actual capacitance or time measurement is required, two hardware calibrations must take place: the current source needs calibration to set it to a precise current, and the circuit being measured needs calibration to measure and/or nullify all other capacitance other than that to be measured.

20.3.1 CURRENT SOURCE CALIBRATION

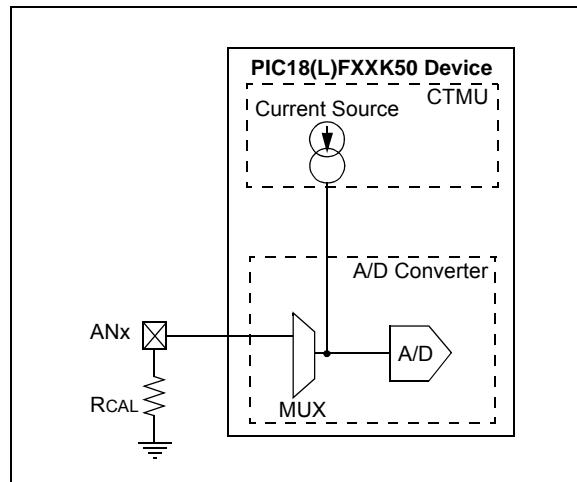
The current source on the CTMU module is trimmable. Therefore, for precise measurements, it is possible to measure and adjust this current source by placing a high precision resistor, R_{CAL} , onto an unused analog channel. An example circuit is shown in [Figure 20-2](#). The current source measurement is performed using the following steps:

1. Initialize the A/D Converter.
2. Initialize the CTMU.
3. Enable the current source by setting EDG1STAT (CTMUCONL<0>).
4. Issue settling time delay.
5. Perform A/D conversion.
6. Calculate the current source current using $I = V/R_{CAL}$, where R_{CAL} is a high precision resistance and V is measured by performing an A/D conversion.

The CTMU current source may be trimmed with the trim bits in CTMUICON using an iterative process to get an exact desired current. Alternatively, the nominal value without adjustment may be used; it may be stored by the software for use in all subsequent capacitive or time measurements.

To calculate the value for R_{CAL} , the nominal current must be chosen, and then the resistance can be calculated. For example, if the A/D Converter reference voltage is 3.3V, use 70% of full scale, or 2.31V as the desired approximate voltage to be read by the A/D Converter. If the range of the CTMU current source is selected to be 0.55 μ A, the resistor value needed is calculated as $R_{CAL} = 2.31V/0.55 \mu$ A, for a value of 4.2 M Ω . Similarly, if the current source is chosen to be 5.5 μ A, R_{CAL} would be 420,000 Ω , and 42,000 Ω if the current source is set to 55 μ A.

FIGURE 20-2: CTMU CURRENT SOURCE CALIBRATION CIRCUIT



A value of 70% of full-scale voltage is chosen to make sure that the A/D Converter was in a range that is well above the noise floor. Keep in mind that if an exact current is chosen, that is to incorporate the trimming bits from CTMUICON, the resistor value of R_{CAL} may need to be adjusted accordingly. R_{CAL} may also be adjusted to allow for available resistor values. R_{CAL} should be of the highest precision available, keeping in mind the amount of precision needed for the circuit that the CTMU will be used to measure. A recommended minimum would be 0.1% tolerance.

The following examples show one typical method for performing a CTMU current calibration. [Example 20-1](#) demonstrates how to initialize the A/D Converter and the CTMU; this routine is typical for applications using both modules. [Example 20-2](#) demonstrates one method for the actual calibration routine.

PIC18(L)F2X/45K50

EXAMPLE 20-1: SETUP FOR CTMU CALIBRATION ROUTINES

```
#include "p18cxx.h"
/*********************************************************************
/*Set up CTMU ****
/*********************************************************************
void setup(void)

{ //CTMUCONH/1 - CTMU Control registers

    CTMUCONH = 0x00;      //make sure CTMU is disabled
    CTMUCONL = 0x90;
    //CTMU continues to run when emulator is stopped,CTMU continues
    //to run in idle mode,Time Generation mode disabled, Edges are blocked
    //No edge sequence order, Analog current source not grounded, trigger
    //output disabled, Edge2 polarity = positive level, Edge2 source =
    //source 0, Edge1 polarity = positive level, Edge1 source = source 0,

    //CTMUIICON - CTMU Current Control Register
    CTMUIICON = 0x01;      //0.55uA, Nominal - No Adjustment

/*********************************************************************
//Set up AD converter;
/*********************************************************************


    TRISA=0x04;           //set channel 2 as an input

    // Configure AN2 as an analog channel
    ANSELAbits.ANSA2=1;
    TRISAbits.TRISA2=1;

    // ADCON2
    ADCON2bits.ADFM=1;        // Results format 1= Right justified
    ADCON2bits.ACQT=1;        // Acqution time 7 = 20TAD 2 = 4TAD 1=2TAD
    ADCON2bits.ADCS=2;        // Clock conversion bits 6= FOSC/64 2=FOSC/32

    // ADCON1
    ADCON1bits.PVCFG0 =0;     // Vref+ = AVdd
    ADCON1bits.NVCFG1 =0;     // Vref- = AVss
    // ADCON0
    ADCON0bits.CHS=2;         // Select ADC channel

    ADCON0bits.ADON=1;        // Turn on ADC

}
```

EXAMPLE 20-2: CURRENT CALIBRATION ROUTINE

```
#include "p18cxx.h"

#define COUNT 500                                // @ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define RCAL .027
#define ADSCALE 1023
#define ADREF 3.3

int main(void)
{
    int i;
    int j = 0;                                     // index for loop
    unsigned int Vread = 0;
    double VTot = 0;
    float Vavg=0, Vcal=0, CTMUISrc = 0;           // float values stored for calcs

    //assume CTMU and A/D have been set up correctly
    //see Example 25-1 for CTMU & A/D setup
    setup();

    CTMUCONHbits.CTMUEN = 1;                      //Enable the CTMU
    CTMUCONLbits.EDG1STAT = 0;                     // Set Edge status bits to zero
    CTMUCONLbits.EDG2STAT = 0;

    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;                 //drain charge on the circuit
        DELAY;                                    //wait 125us
        CTMUCONHbits.IDISSEN = 0;                  //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;                 //Begin charging the circuit
        //using CTMU current source
        DELAY;                                    //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;                  //Stop charging circuit

        PIR1bits.ADIF = 0;                         //make sure A/D Int not set
        ADCON0bits.GO=1;                          //and begin A/D conv.
        while(!PIR1bits.ADIF);                    //Wait for A/D convert complete

        Vread = ADRES;                            //Get the value from the A/D
        PIR1bits.ADIF = 0;                         //Clear A/D Interrupt Flag
        VTot += Vread;                           //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);                  //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);          //CTMUISrc is in 1/100ths of uA
    CTMUISrc = Vcal/RCAL;
}
```

PIC18(L)F2X/45K50

20.3.2 CAPACITANCE CALIBRATION

There is a small amount of capacitance from the internal A/D Converter sample capacitor as well as stray capacitance from the circuit board traces and pads that affect the precision of capacitance measurements. A measurement of the stray capacitance can be taken by making sure the desired capacitance to be measured has been removed. The measurement is then performed using the following steps:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT (= 1).
3. Wait for a fixed delay of time t .
4. Clear EDG1STAT.
5. Perform an A/D conversion.
6. Calculate the stray and A/D sample capacitances:

$$C_{\text{OFFSET}} = C_{\text{STRAY}} + C_{\text{AD}} = (I \cdot t) / V$$

where I is known from the current source measurement step, t is a fixed delay and V is measured by performing an A/D conversion.

This measured value is then stored and used for calculations of time measurement or subtracted for capacitance measurement. For calibration, it is expected that the capacitance of $C_{\text{STRAY}} + C_{\text{AD}}$ is approximately known. C_{AD} is approximately 4 pF.

An iterative process may need to be used to adjust the time, t , that the circuit is charged to obtain a reasonable voltage reading from the A/D Converter. The value of t may be determined by setting C_{OFFSET} to a theoretical value, then solving for t . For example, if C_{STRAY} is theoretically calculated to be 11 pF, and V is expected to be 70% of V_{DD} , or 2.31V, then t would be:

$$(4 \text{ pF} + 11 \text{ pF}) \cdot 2.31V / 0.55 \mu\text{A}$$

or 63 μs .

See [Example 20-3](#) for a typical routine for CTMU capacitance calibration.

EXAMPLE 20-3: CAPACITANCE CALIBRATION ROUTINE

```
#include "p18cxx.h"

#define COUNT 25                                // @ 8MHz INTFRC = 62.5 us.
#define ETIME COUNT*2.5                         // time in uS
#define DELAY for(i=0;i<COUNT;i++)             //for unsigned conversion 10 sig
#define ADScale 1023                            //for unsigned conversion 10 sig
bits
#define ADREF 3.3                               //Vdd connected to A/D Vr+
#define RCAL .027                               //R value is 4200000 (4.2M)
                                              //scaled so that result is in
                                              //1/100th of uA

int main(void)
{
    int i;
    int j = 0;                                  //index for loop
    unsigned int Vread = 0;
    float CTMUISrc, CTMUCap, Vavg, VTot, Vcal;

    //assume CTMU and A/D have been set up correctly
    //see Example 25-1 for CTMU & A/D setup
    setup();

    CTMUCONHbits.CTMUEN = 1;                   //Enable the CTMU
    CTMUCONLbits.EDG1STAT = 0;                 // Set Edge status bits to zero
    CTMUCONLbits.EDG2STAT = 0;
    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;              //drain charge on the circuit
        DELAY;                                 //wait 125us
        CTMUCONHbits.IDISSEN = 0;              //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;              //Begin charging the circuit
                                                //using CTMU current source
        DELAY;                                 //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;              //Stop charging circuit

        PIR1bits.ADIF = 0;                     //make sure A/D Int not set
        ADCON0bits.GO=1;                      //and begin A/D conv.
        while(!PIR1bits.ADIF);                //Wait for A/D convert complete

        Vread = ADRES;                        //Get the value from the A/D
        PIR1bits.ADIF = 0;                    //Clear A/D Interrupt Flag
        VTot += Vread;                       //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);            //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);     //CTMUISrc is in 1/100ths of uA
    CTMUISrc = Vcal/RCAL;
    CTMUCap = (CTMUISrc*ETIME/Vcal)/100;
}
```

20.4 Measuring Capacitance with the CTMU

There are two separate methods of measuring capacitance with the CTMU. The first is the absolute method, in which the actual capacitance value is desired. The second is the relative method, in which the actual capacitance is not needed, rather an indication of a change in capacitance is required.

20.4.1 ABSOLUTE CAPACITANCE MEASUREMENT

For absolute capacitance measurements, both the current and capacitance calibration steps found in [Section 20.3 “Calibrating the CTMU Module”](#) should be followed. Capacitance measurements are then performed using the following steps:

1. Initialize the A/D Converter.
2. Initialize the CTMU.
3. Set EDG1STAT.
4. Wait for a fixed delay, T .
5. Clear EDG1STAT.
6. Perform an A/D conversion.
7. Calculate the total capacitance, $C_{TOTAL} = (I * T)/V$, where I is known from the current source measurement step (see [Section 20.3.1 “Current Source Calibration”](#)), T is a fixed delay and V is measured by performing an A/D conversion.
8. Subtract the stray and A/D capacitance (C_{OFFSET} from [Section 20.3.2 “Capacitance Calibration”](#)) from C_{TOTAL} to determine the measured capacitance.

20.4.2 RELATIVE CHARGE MEASUREMENT

An application may not require precise capacitance measurements. For example, when detecting a valid press of a capacitance-based switch, detecting a relative change of capacitance is of interest. In this type of application, when the switch is open (or not touched), the total capacitance is the capacitance of the combination of the board traces, the A/D Converter, etc. A larger voltage will be measured by the A/D Converter. When the switch is closed (or is touched), the total capacitance is larger due to the addition of the capacitance of the human body to the above listed capacitances, and a smaller voltage will be measured by the A/D Converter.

Detecting capacitance changes is easily accomplished with the CTMU using these steps:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT.
3. Wait for a fixed delay.
4. Clear EDG1STAT.
5. Perform an A/D conversion.

The voltage measured by performing the A/D conversion is an indication of the relative capacitance. Note that in this case, no calibration of the current source or circuit capacitance measurement is needed. See [Example 20-4](#) for a sample software routine for a capacitive touch switch.

EXAMPLE 20-4: ROUTINE FOR CAPACITIVE TOUCH SWITCH

```
#include "p18cxx.h"

#define COUNT 500                      // @ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define OPENSW 1000                     // Un-pressed switch value
#define TRIP 300                        // Difference between pressed
                                         // and un-pressed switch
#define HYST 65                         // amount to change
                                         // from pressed to un-pressed
#define PRESSED 1
#define UNPRESSED 0

int main(void)
{
    unsigned int Vread;                // storage for reading
    unsigned int switchState;
    int i;

    //assume CTMU and A/D have been set up correctly
    //see Example 25-1 for CTMU & A/D setup
    setup();

    CTMUCONHbits.CTMUEN = 1;           // Enable the CTMU
    CTMUCONLbits.EDG1STAT = 0;         // Set Edge status bits to zero
    CTMUCONLbits.EDG2STAT = 0;
    CTMUCONHbits.IDISSEN = 1;          // drain charge on the circuit
    DELAY;                            // wait 125us
    CTMUCONHbits.IDISSEN = 0;          // end drain of circuit

    CTMUCONLbits.EDG1STAT = 1;          // Begin charging the circuit
                                         // using CTMU current source
    DELAY;                            // wait for 125us
    CTMUCONLbits.EDG1STAT = 0;          // Stop charging circuit

    PIR1bits.ADIF = 0;                // make sure A/D Int not set
    ADCON0bits.GO=1;                  // and begin A/D conv.
    while(!PIR1bits.ADIF);            // Wait for A/D convert complete

    Vread = ADRES;                   // Get the value from the A/D

    if(Vread < OPENSW - TRIP)
    {
        switchState = PRESSED;
    }
    else if(Vread > OPENSW - TRIP + HYST)
    {
        switchState = UNPRESSED;
    }
}
```

PIC18(L)F2X/45K50

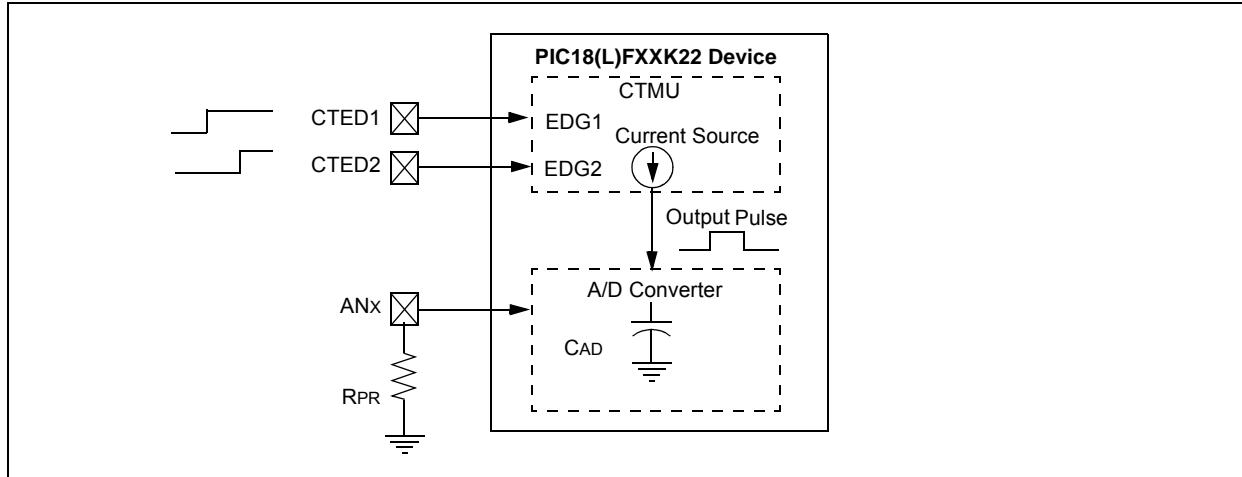
20.5 Measuring Time with the CTMU Module

Time can be precisely measured after the ratio (C/I) is measured from the current and capacitance calibration step by following these steps:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT.
3. Set EDG2STAT.
4. Perform an A/D conversion.
5. Calculate the time between edges as $T = (C/I) * V$, where I is calculated in the current calibration step ([Section 20.3.1 “Current Source Calibration”](#)), C is calculated in the capacitance calibration step ([Section 20.3.2 “Capacitance Calibration”](#)) and V is measured by performing the A/D conversion.

It is assumed that the time measured is small enough that the capacitance, C_{OFFSET} , provides a valid voltage to the A/D Converter. For the smallest time measurement, always set the A/D Channel Select register (AD1CHS) to an unused A/D channel; the corresponding pin for which is not connected to any circuit board trace. This minimizes added stray capacitance, keeping the total circuit capacitance close to that of the A/D Converter itself (4-5 pF). To measure longer time intervals, an external capacitor may be connected to an A/D channel and this channel selected when making a time measurement.

FIGURE 20-3: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR TIME MEASUREMENT



20.6 Creating a Delay with the CTMU Module

A unique feature on board the CTMU module is its ability to generate system clock independent output pulses based on an external capacitor value. This is accomplished using the internal comparator voltage reference module, Comparator 2 input pin and an external capacitor. The pulse is output onto the CTPLS pin. To enable this mode, set the TGEN bit.

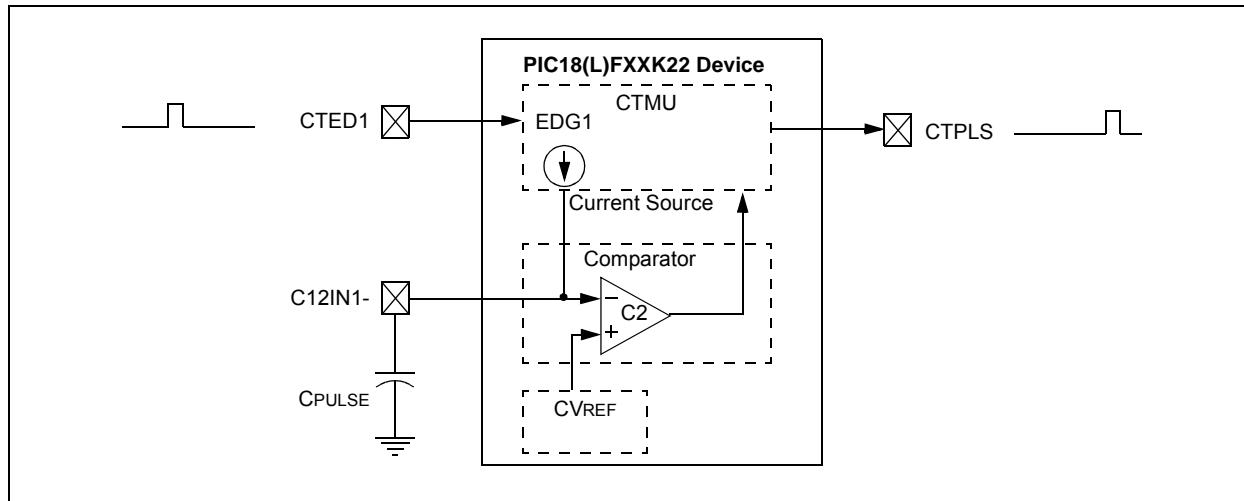
See [Figure 20-4](#) for an example circuit. CPULSE is chosen by the user to determine the output pulse width on CTPLS. The pulse width is calculated by $T = (CPULSE/I) * V$, where I is known from the current source measurement step ([Section 20.3.1 “Current Source Calibration”](#)) and V is the internal reference voltage (CVREF).

An example use of this feature is for interfacing with variable capacitive-based sensors, such as a humidity sensor. As the humidity varies, the pulse width output on CTPLS will vary. The CTPLS output pin can be connected to an input capture pin and the varying pulse width is measured to determine the humidity in the application.

Follow these steps to use this feature:

1. Initialize Comparator 2.
2. Initialize the comparator voltage reference.
3. Initialize the CTMU and enable time delay generation by setting the TGEN bit.
4. Set EDG1STAT.
5. When CPULSE charges to the value of the voltage reference trip point, an output pulse is generated on CTPLS.

FIGURE 20-4: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR PULSE DELAY GENERATION



20.7 Operation During Sleep/Idle Modes

20.7.1 SLEEP MODE

When the device enters any Sleep mode, the CTMU module current source is always disabled. If the CTMU is performing an operation that depends on the current source when Sleep mode is invoked, the operation may not terminate correctly. Capacitance and time measurements may return erroneous values.

20.7.2 IDLE MODE

The behavior of the CTMU in Idle mode is determined by the CTMUSIDL bit (CTMUCONH<5>). If CTMUSIDL is cleared, the module will continue to operate in Idle mode. If CTMUSIDL is set, the module's current source is disabled when the device enters Idle mode. If the module is performing an operation when Idle mode is invoked, in this case, the results will be similar to those with Sleep mode.

20.8 CTMU Peripheral Module Disable (PMD)

When this peripheral is not used, the Peripheral Module Disable bit can be set to disconnect all clock sources to the module, reducing power consumption to an absolute minimum. See [Section 4.6 “Selective Peripheral Module Control”](#).

20.9 Effects of a Reset on CTMU

Upon Reset, all registers of the CTMU are cleared. This leaves the CTMU module disabled, its current source is turned off and all configuration options return to their default settings. The module needs to be re-initialized following any Reset.

If the CTMU is in the process of taking a measurement at the time of Reset, the measurement will be lost. A partial charge may exist on the circuit that was being measured, and should be properly discharged before the CTMU makes subsequent attempts to make a measurement. The circuit is discharged by setting and then clearing the IDISSEN bit (CTMUCONH<1>) while the A/D Converter is connected to the appropriate channel.

20.11 Register Definitions: CTMU Control

REGISTER 20-1: CTMUCONH: CTMU CONTROL REGISTER 0

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
|-------|-------|----------|-------|-------|----------|---------|--------|
| CTMUE | — | CTMUSIDL | TGEN | EDGEN | EDGSEQEN | IDISSEN | CTTRIG |
| 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 | CTMUE : CTMU Enable bit 1 = Module is enabled 0 = Module is disabled |
| bit 6 | Unimplemented : Read as '0' |
| bit 5 | CTMUSIDL : Stop in Idle Mode bit 1 = Discontinue module operation when device enters Idle mode 0 = Continue module operation in Idle mode |
| bit 4 | TGEN : Time Generation Enable bit 1 = Enables edge delay generation 0 = Disables edge delay generation |
| bit 3 | EDGEN : Edge Enable bit 1 = Edges are not blocked 0 = Edges are blocked |
| bit 2 | EDGSEQEN : Edge Sequence Enable bit 1 = Edge 1 event must occur before Edge 2 event can occur 0 = No edge sequence is needed |
| bit 1 | IDISSEN : Analog Current Source Control bit 1 = Analog current source output is grounded 0 = Analog current source output is not grounded |
| bit 0 | CTTRIG : CTMU Special Event Trigger Control Bit 1 = CTMU Special Event Trigger is enabled 0 = CTMU Special Event Trigger is disabled |

20.10 Registers

There are three control registers for the CTMU:

- CTMUCONH
- CTMUCONL
- CTMUICON

The CTMUCONH and CTMUCONL registers ([Register 20-1](#) and [Register 20-2](#)) contain control bits for configuring the CTMU module edge source selection, edge source polarity selection, edge sequencing, A/D trigger, analog circuit capacitor discharge and enables. The CTMUICON register ([Register 20-3](#)) has bits for selecting the current source range and current source trim.

REGISTER 20-2: CTMUCONL: CTMU 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 |
|---------|--------------|-------|---------|--------------|-------|----------|----------|
| EDG2POL | EDG2SEL<1:0> | | EDG1POL | EDG1SEL<1:0> | | EDG2STAT | EDG1STAT |
| 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 **EDG2POL:** Edge 2 Polarity Select bit
 1 = Edge 2 programmed for a positive edge response
 0 = Edge 2 programmed for a negative edge response
- bit 6-5 **EDG2SEL<1:0>:** Edge 2 Source Select bits
 11 = CTED1 pin
 10 = CTED2 pin
 01 = ECCP1 Special Event Trigger
 00 = CCP2 Special Event Trigger
- bit 4 **EDG1POL:** Edge 1 Polarity Select bit
 1 = Edge 1 programmed for a positive edge response
 0 = Edge 1 programmed for a negative edge response
- bit 3-2 **EDG1SEL<1:0>:** Edge 1 Source Select bits
 11 = CTED1 pin
 10 = CTED2 pin
 01 = ECCP1 Special Event Trigger
 00 = CCP2 Special Event Trigger
- bit 1 **EDG2STAT:** Edge 2 Status bit
 1 = Edge 2 event has occurred
 0 = Edge 2 event has not occurred
- bit 0 **EDG1STAT:** Edge 1 Status bit
 1 = Edge 1 event has occurred
 0 = Edge 1 event has not occurred

PIC18(L)F2X/45K50

REGISTER 20-3: CTMUICON: CTMU CURRENT 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 |
| ITRIM<5:0> | | | | | | IRNG<1:0> | |
| 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 **ITRIM<5:0>**: Current Source Trim bits

011111 = Maximum positive change from nominal current

011110

.

.

000001 = Minimum positive change from nominal current

000000 = Nominal current output specified by IRNG<1:0>

111111 = Minimum negative change from nominal current

.

.

100010

100001 = Maximum negative change from nominal current

bit 1-0 **IRNG<1:0>**: Current Source Range Select bits (see [Table 29-14](#))

11 = 100 × Base current

10 = 10 × Base current

01 = Base current level

00 = Current source disabled

TABLE 20-1: REGISTERS ASSOCIATED WITH CTMU MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|-----------|------------|--------------|----------|---------|--------------|----------|-----------|----------|---------------------|
| CTMUICONH | CTMUEEN | — | CTMUSIDL | TGEN | EDGEN | EDGSEQEN | IDISSEN | CTTRIG | 322 |
| CTMUICONL | EDG2POL | EDG2SEL<1:0> | | EDG1POL | EDG1SEL<1:0> | | EDG2STAT | EDG1STAT | 323 |
| CTMUICON | ITRIM<5:0> | | | | | | IRNG<1:0> | | 324 |
| IPR3 | — | — | — | — | CTMUIP | USBIP | TMR3GIP | TMR1GIP | 125 |
| PIE3 | — | — | — | — | CTMUIE | USBIE | TMR3GIE | TMR1GIE | 122 |
| PIR3 | — | — | — | — | CTMUIF | USBIF | TMR3GIF | TMR1GIF | 119 |
| PMD1 | — | MSSPMMD | CTMUMD | CMP2MD | CMP1MD | ADCMD | CCP2MD | CCP1MD | 62 |

Legend: — = Unimplemented, read as '0'. Shaded bits are not used during CTMU operation.

21.0 SR LATCH

The module consists of a single SR latch with multiple Set and Reset inputs as well as separate latch outputs. The SR latch module includes the following features:

- Programmable input selection
- SR latch output is available internally/externally
- Selectable Q and \bar{Q} output
- Firmware Set and Reset

The SR latch can be used in a variety of analog applications, including oscillator circuits, one-shot circuit, hysteretic controllers, and analog timing applications.

21.1 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:

- Software control (SRPS and SRPR bits)
- Comparator C1 output (sync_C1OUT)
- Comparator C2 output (sync_C2OUT)
- SRI Pin
- Programmable clock (DIVSRCLK)

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.

The output from Comparator C1 or C2 can be used as the Set or Reset inputs of the SR latch. The output of either Comparator can be synchronized to the Timer1 clock source. See [Section 19.0 “Comparator Module”](#) and [Section 13.0 “Timer1/3 Module with Gate Control”](#) for more information.

An external source on the SRI pin can be used as the Set or Reset inputs of the SR latch.

An internal clock source, DIVSRCLK, is available and it can periodically set or reset the SR latch. The SRCLK<2:0> bits in the SRCON0 register are used to select the clock source period. The SRSCKE and SRRCKE bits of the SRCON1 register enable the clock source to set or reset the SR latch, respectively.

21.2 Latch Output

The SRQEN and SRNQEN bits of the SRCON0 register control the Q and \bar{Q} latch outputs. Both of the SR latch outputs may be directly output to I/O pins at the same time. Control is determined by the state of bits SRQEN and SRNQEN in the SRCON0 register.

The applicable TRIS bit of the corresponding port must be cleared to enable the port pin output driver.

21.3 DIVSRCLK Clock Generation

The DIVSRCLK clock signal is generated from the peripheral clock which is pre-scaled by a value determined by the SRCLK<2:0> bits. See [Figure 21-1](#) and [Table 21-1](#) for additional detail.

21.4 Effects of a Reset

Upon any device Reset, the SR latch is not initialized, and the SRQ and SRNQ outputs are unknown. The user's firmware is responsible to initialize the latch output before enabling it to the output pins.

PIC18(L)F2X/45K50

FIGURE 21-1: DIVSRCLK BLOCK DIAGRAM

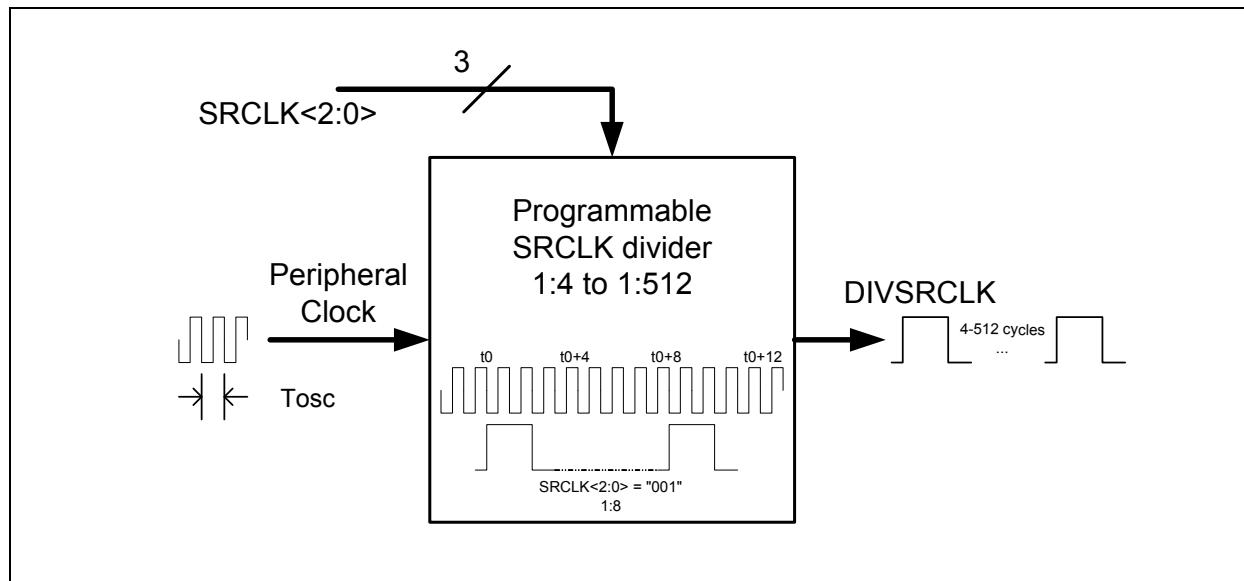


FIGURE 21-2: SR LATCH SIMPLIFIED BLOCK DIAGRAM

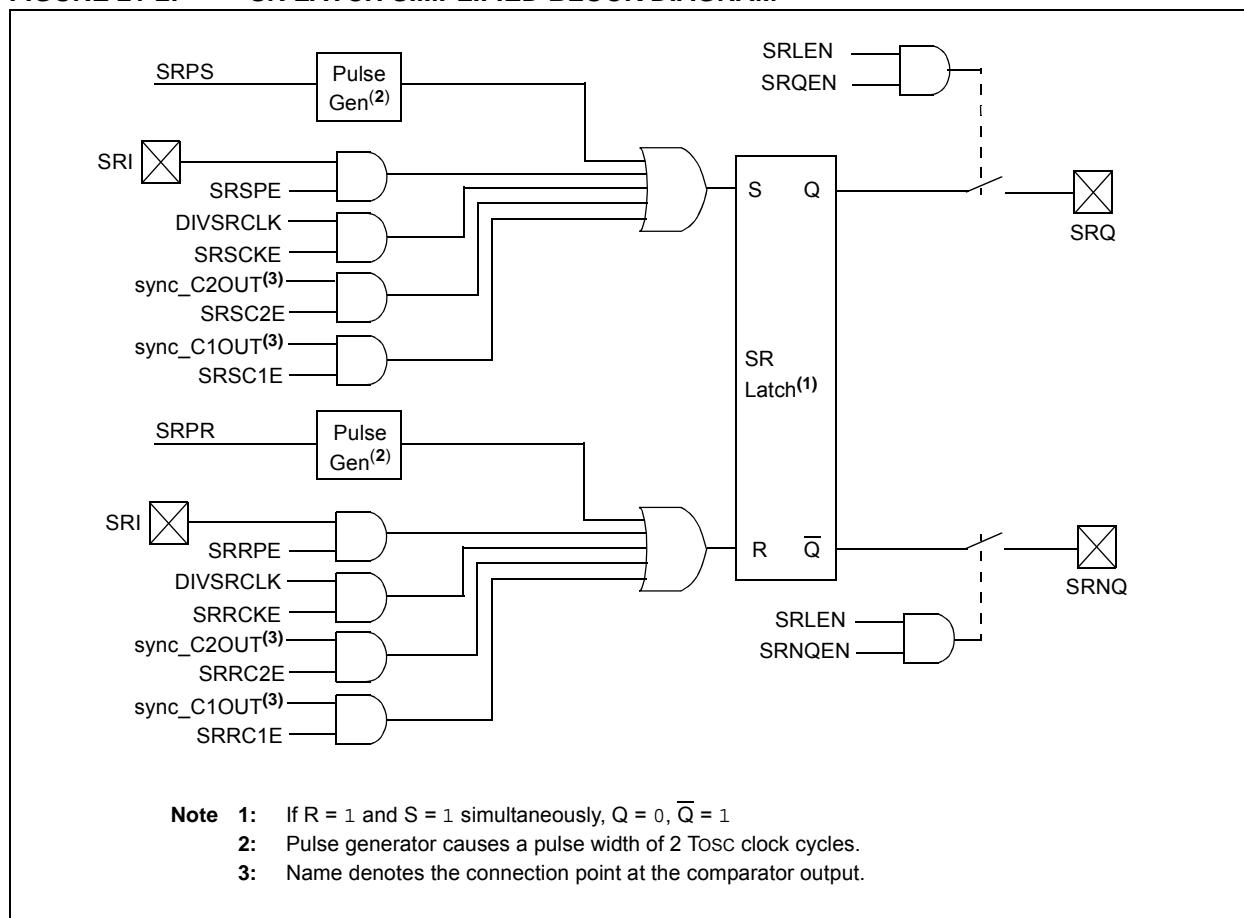


TABLE 21-1: DIVSRCLK FREQUENCY TABLE

| SRCLK<2:0> | 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 |

PIC18(L)F2X/45K50

21.5 Register Definitions: SR Latch Control

REGISTER 21-1: 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 | | SRCLK<2:0> | | SRQEN | SRNQEN | SRPS | SRPR |
| bit 7 | bit 0 | | | | | | |

Legend:

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

bit 7 **SRLEN:** SR Latch Enable bit⁽¹⁾

- 1 = SR latch is enabled
0 = SR latch is disabled

bit 6-4 **SRCLK<2:0>:** SR Latch Clock Divider Bits

- 000 = Generates a 2 Tosc wide pulse on DIVSRCLK every 4 peripheral clock cycles
001 = Generates a 2 Tosc wide pulse on DIVSRCLK every 8 peripheral clock cycles
010 = Generates a 2 Tosc wide pulse on DIVSRCLK every 16 peripheral clock cycles
011 = Generates a 2 Tosc wide pulse on DIVSRCLK every 32 peripheral clock cycles
100 = Generates a 2 Tosc wide pulse on DIVSRCLK every 64 peripheral clock cycles
101 = Generates a 2 Tosc wide pulse on DIVSRCLK every 128 peripheral clock cycles
110 = Generates a 2 Tosc wide pulse on DIVSRCLK every 256 peripheral clock cycles
111 = Generates a 2 Tosc wide pulse on DIVSRCLK every 512 peripheral clock cycles

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 = Q is internal only

bit 1 **SRPS:** Pulse Set Input of the SR Latch bit⁽²⁾

- 1 = Pulse set input for 2 Tosc clock cycles
0 = No effect on set input

bit 0 **SRPR:** Pulse Reset Input of the SR Latch bit⁽²⁾

- 1 = Pulse Reset input for 2 TOSC clock cycles
0 = No effect on Reset input

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.

2: Set only, always reads back '0'.

REGISTER 21-2: 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 -n = Value at POR | W = Writable bit '1' = Bit is set | U = Unimplemented '0' = Bit is cleared | C = Clearable only bit x = Bit is unknown |
|---------------------------------------|--------------------------------------|---|--|

| | |
|-------|--|
| bit 7 | SRSPE: SR Latch Peripheral Set Enable bit 1 = SRI pin status sets SR latch 0 = SRI 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 DIVSRCLK 0 = Set input of SR latch is not pulsed with DIVSRCLK |
| 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 = SRI pin resets SR latch 0 = SRI 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 DIVSRCLK 0 = Reset input of SR latch is not pulsed with DIVSRCLK |
| 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 21-2: REGISTERS ASSOCIATED WITH THE SR LATCH

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|--------|--------|------------|--------|--------|--------|--------|--------|--------|------------------|
| SRCON0 | SRLEN | SRCLK<2:0> | | | SRQEN | SRNQEN | SRPS | SRPR | 328 |
| SRCON1 | SRSPE | SRSCKE | SRSC2E | SRSC1E | SRRPE | SRRCKE | SRRC2E | SRRC1E | 329 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 149 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 149 |
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 | 150 |

Legend: Shaded bits are not used with this module.

22.0 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 FVREN bit of the VREFCON0 register.

22.1 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.

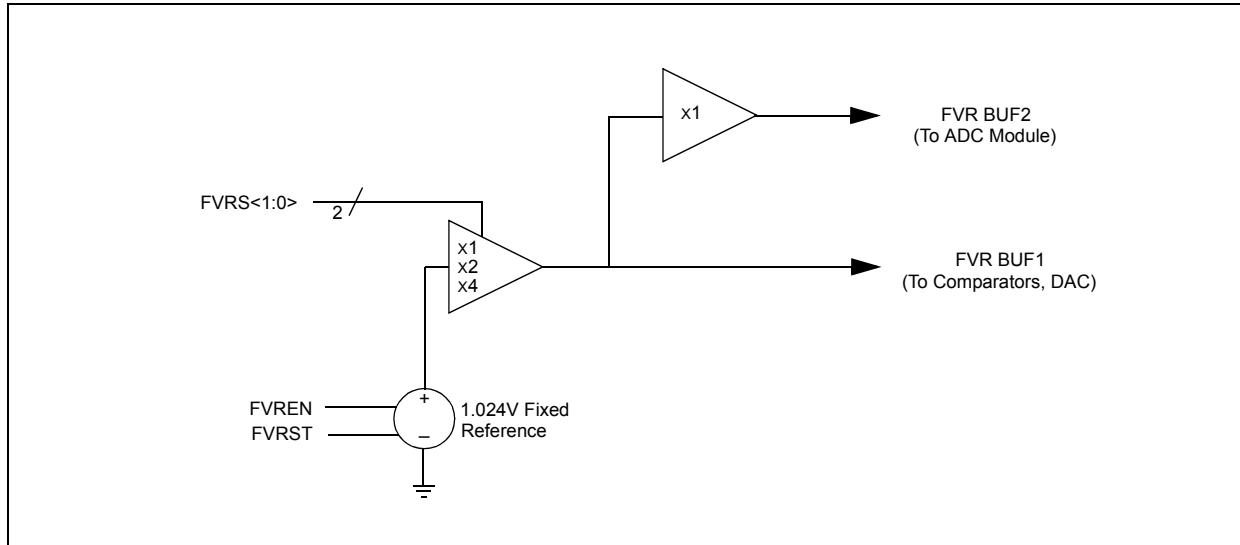
The FVRS<1:0> bits of the VREFCON0 register are used to enable and configure the gain amplifier settings for the reference supplied to the DAC and Comparator modules. When the ADC module is configured to use the FVR output, (FVR BUF2) the reference is buffered through an additional unity gain amplifier. 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 18.0 “Analog-to-Digital Converter \(ADC\) Module”](#), [Section 23.0 “Digital-to-Analog Converter \(DAC\) Module”](#) and [Section 19.0 “Comparator Module”](#).

22.2 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 FVRST bit of the VREFCON0 register will be set. See [Table 29-13](#) for the minimum delay requirement.

FIGURE 22-1: VOLTAGE REFERENCE BLOCK DIAGRAM



22.3 Register Definitions: FVR Control

REGISTER 22-1: 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 |
|-------|-------|-------|-----------|-----|-----|-----|-----|
| FVREN | FVRST | | FVRS<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 | FVREN: Fixed Voltage Reference Enable bit 0 = Fixed Voltage Reference is disabled 1 = Fixed Voltage Reference is enabled |
| bit 6 | FVRST: 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 | FVRS<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 22-1: 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 | Register on page |
|----------|-------|-------|-----------|-------|-------|-------|-------|-------|------------------|
| VREFCON0 | FVREN | FVRST | FVRS<1:0> | | — | — | — | — | 331 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used by the FVR module.

23.0 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
- DACOUT pin

The Digital-to-Analog Converter (DAC) can be enabled by setting the DACEN bit of the VREFCON1 register.

23.1 Output Voltage Selection

The DAC has 32 voltage level ranges. The 32 levels are set with the DACR<4:0> bits of the VREFCON2 register.

The DAC output voltage is determined by the following equations:

EQUATION 23-1: DAC OUTPUT VOLTAGE

$$V_{OUT} = \left((V_{SRC+} - V_{SRC-}) \times \frac{DACR<4:0>}{2^5} \right) + V_{SRC-}$$

$V_{SRC+} = V_{DD}, V_{REF+}$ or FVR_1

$V_{SRC-} = V_{SS}$ or V_{REF-}

23.2 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 29.0 "Electrical Specifications"](#).

23.3 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, (V_{SRC+}), or the negative voltage source, (V_{SRC-}) can be disabled.

The negative voltage source is disabled by setting the DACLPS bit in the VREFCON1 register. Clearing the DACLPS bit in the VREFCON1 register disables the positive voltage source.

23.4 Output Clamped to Positive Voltage Source

The DAC output voltage can be set to V_{SRC+} with the least amount of power consumption by performing the following:

- Clearing the DACEN bit in the VREFCON1 register.
- Setting the DACLPS bit in the VREFCON1 register.
- Configuring the DACPSS bits to the proper positive source.
- Configuring the DACRx 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 23.6 "DAC Voltage Reference Output"](#) for more information.

23.5 Output Clamped to Negative Voltage Source

The DAC output voltage can be set to V_{SRC-} with the least amount of power consumption by performing the following:

- Clearing the DACEN bit in the VREFCON1 register.
- Clearing the DACLPS bit in the VREFCON1 register.
- Configuring the DACPSS bits to the proper negative source.
- Configuring the DACRx 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.

23.6 DAC Voltage Reference Output

The DAC can be output to the DACOUT pin by setting the DACOE bit of the VREFCON1 register to '1'. Selecting the DAC reference voltage for output on the DACOUT pin automatically overrides the digital output buffer and digital input threshold detector functions of that pin. Reading the DACOUT 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 DACOUT. [Figure 23-2](#) shows an example buffering technique.

FIGURE 23-1: DIGITAL-TO-ANALOG CONVERTER BLOCK DIAGRAM

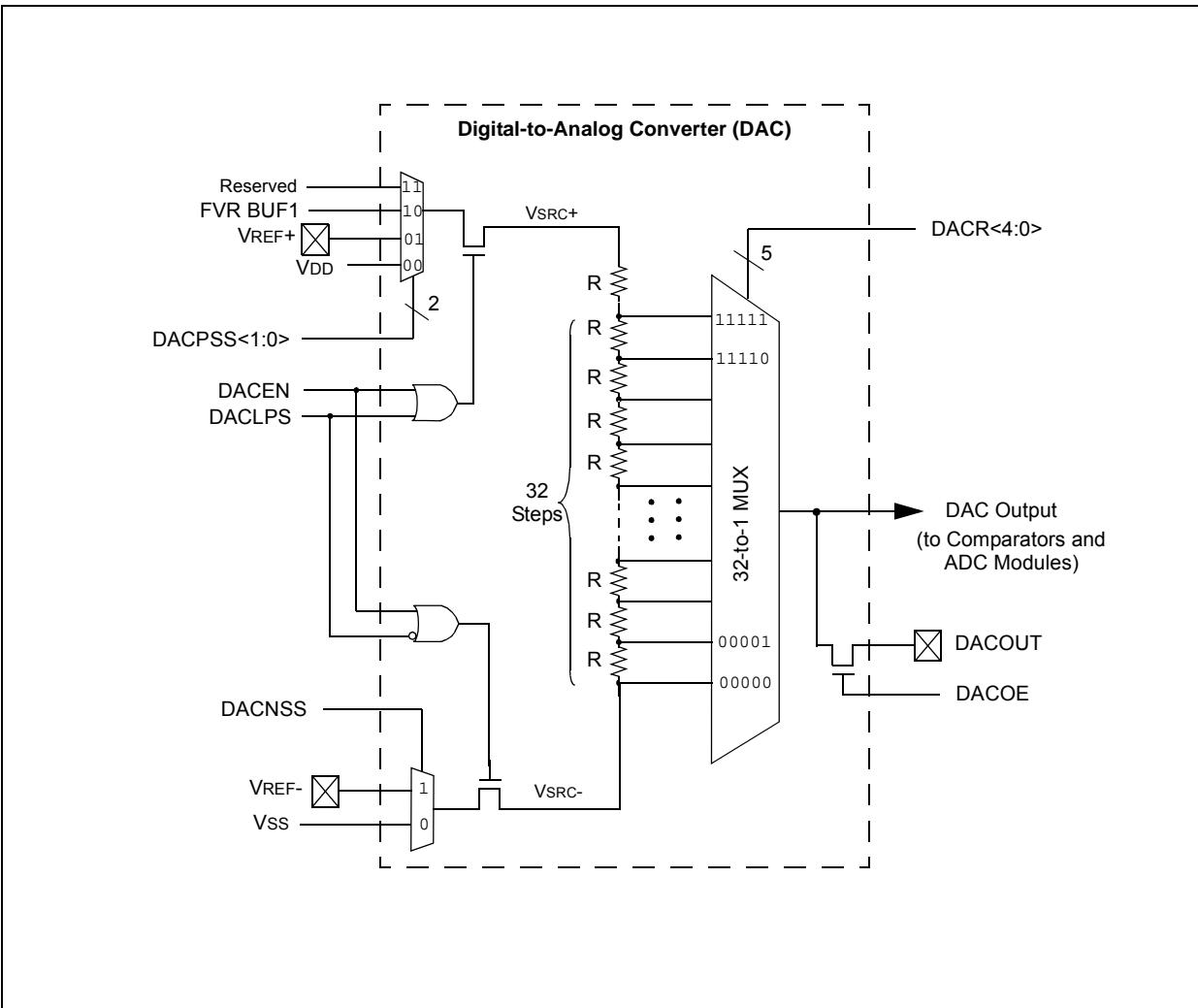
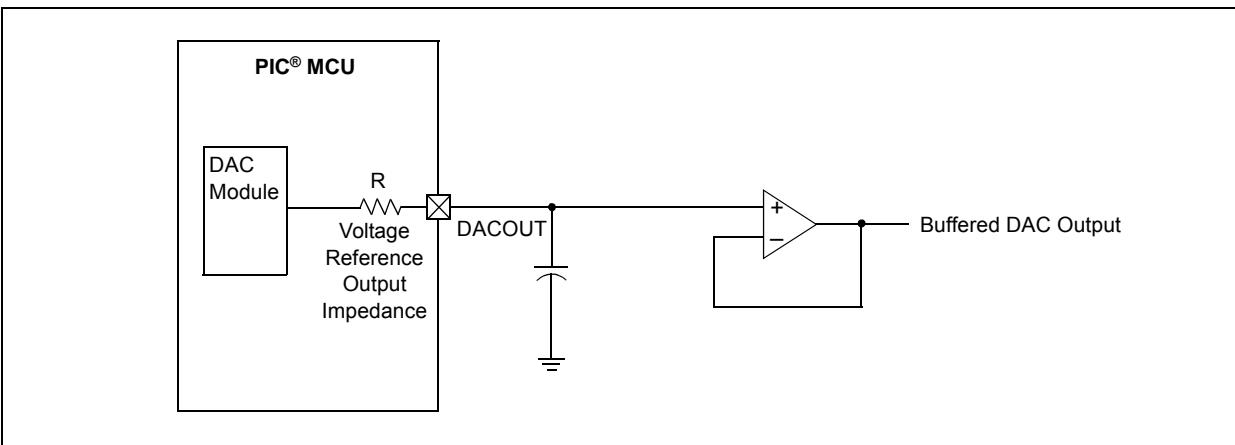


FIGURE 23-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE



23.7 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.

23.8 Effects of a Reset

A device Reset affects the following:

- DAC is disabled
- DAC output voltage is removed from the DACOUT pin
- The DACR<4:0> range select bits are cleared

23.9 Register Definitions: DAC Control

REGISTER 23-1: 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 |
|-------|--------|-------|-----|-------------|-------|-----|--------|
| DACEN | DACLPS | DACOE | — | DACPSS<1:0> | — | — | DACNSS |
| 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 | DACEN: DAC Enable bit 1 = DAC is enabled 0 = DAC is disabled |
| bit 6 | DACLPS: DAC Low-Power Voltage Source Select bit 1 = DAC Positive reference source selected 0 = DAC Negative reference source selected |
| bit 5 | DACOE: DAC Voltage Output Enable bit 1 = DAC voltage level is also an output on the DACOUT pin 0 = DAC voltage level is disconnected from the DACOUT pin |
| bit 4 | Unimplemented: Read as '0' |
| bit 3-2 | DACPSS<1:0>: DAC Positive Source Select bits 00 = VDD 01 = VREF+ 10 = FVR BUF1 output 11 = Reserved, do not use |
| bit 1 | Unimplemented: Read as '0' |
| bit 0 | DACNSS: DAC Negative Source Select bits 1 = VREF- 0 = VSS |

REGISTER 23-2: 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 | | |
| — | — | — | DACR<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 **DACR<4:0>:** DAC Voltage Output Select bits

$$V_{OUT} = ((V_{SRC+}) - (V_{SRC-})) * (DACR<4:0> / (2^5)) + V_{SRC-}$$

TABLE 23-1: REGISTERS ASSOCIATED WITH DAC MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|--------|-------|-----------|-------------|-------|--------|-------|------------------|
| VREFCON1 | DACEN | DACLPS | DACOE | — | DACPSS<1:0> | — | DACNSS | 334 | |
| VREFCON2 | — | — | — | DACR<4:0> | | | | | 335 |

Legend: — = Unimplemented locations, read as '0'. Shaded bits are not used by the DAC module.

24.0 UNIVERSAL SERIAL BUS (USB)

This section describes the details of the USB peripheral. Because of the very specific nature of the module, knowledge of USB is expected. Some high-level USB information is provided in [Section 3.14 “Oscillator Settings for USB”](#) only for application design reference. Designers are encouraged to refer to the official specification published by the USB Implementers Forum (USB-IF) for the latest information.

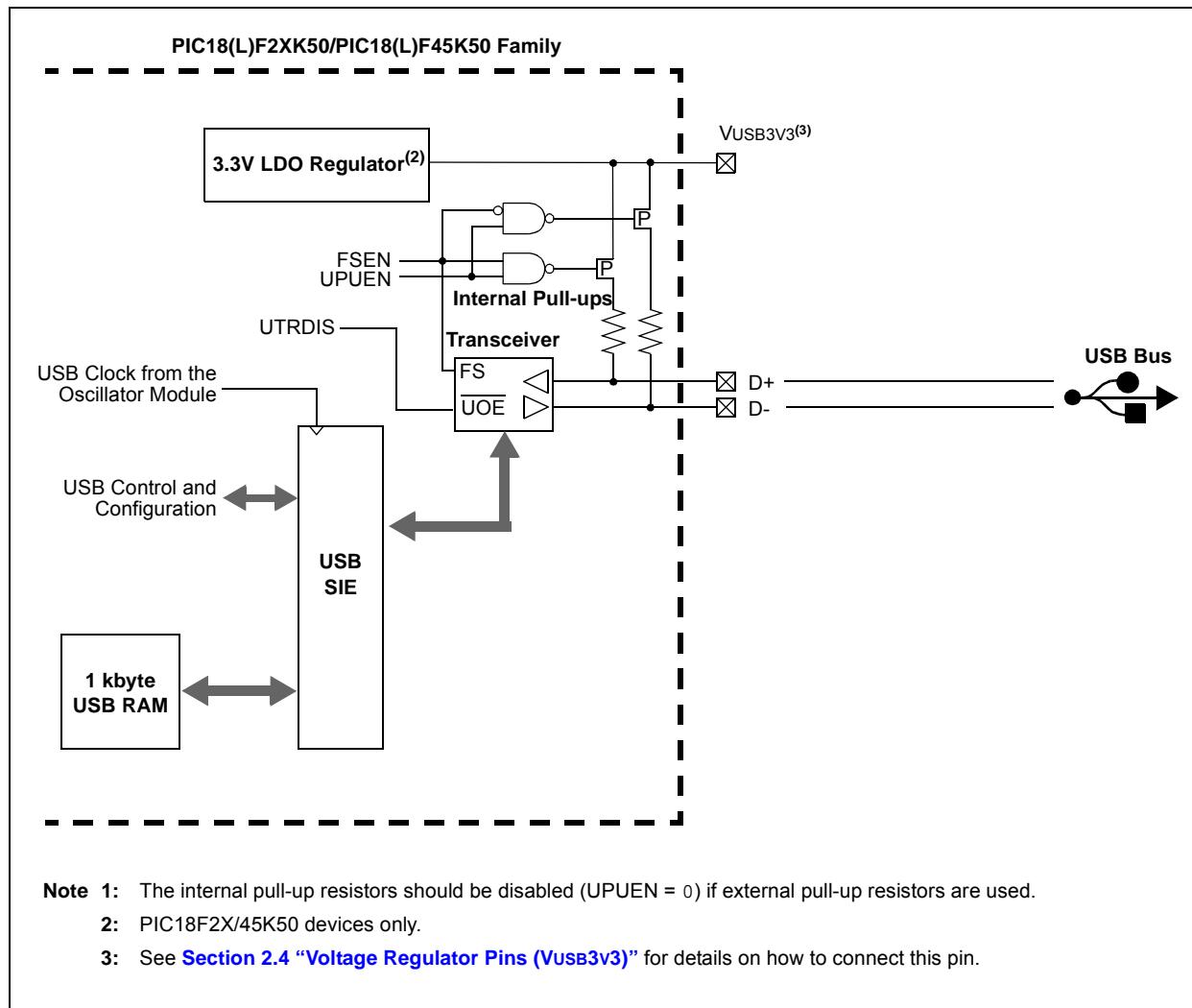
24.1 Overview of the USB Peripheral

PIC18F2X/45K50 devices contain a full-speed and low-speed compatible USB Serial Interface Engine (SIE) that allows fast communication between any USB

host and the PIC microcontroller. The SIE can be interfaced directly to the USB by utilizing the internal transceiver.

Some special hardware features have been included to improve performance. Dual access port memory in the device's data memory space (USB RAM) has been supplied to share direct memory access between the microcontroller core and the SIE. Buffer descriptors are also provided, allowing users to freely program endpoint memory usage within the USB RAM space. [Figure 24-1](#) presents a general overview of the USB peripheral and its features.

FIGURE 24-1: USB PERIPHERAL AND OPTIONS



24.2 USB Status and Control

The operation of the USB module is configured and managed through three control registers. In addition, a total of 14 registers are used to manage the actual USB transactions. The registers are:

- USB Control register (UCON)
- USB Configuration register (UCFG)
- USB Transfer Status register (USTAT)
- USB Device Address register (UADDR)
- Frame Number registers (UFRMH:UFRML)
- Endpoint Enable registers 0 through 7 (UEPn)

24.2.1 USB CONTROL REGISTER (UCON)

The USB Control register ([Register 24-1](#)) contains bits needed to control the module behavior during transfers. The register contains bits that control the following:

- Main USB Peripheral Enable
- Ping-Pong Buffer Pointer Reset
- Control of the Suspend mode
- Packet Transfer Disable

In addition, the USB Control register contains a status bit, SE0 (UCON<5>), which is used to indicate the occurrence of a single-ended zero on the bus. When the USB module is enabled, this bit should be monitored to determine whether the differential data lines have come out of a single-ended zero condition. This helps to differentiate the initial power-up state from the USB Reset signal.

The overall operation of the USB module is controlled by the USBEN bit (UCON<3>). Setting this bit activates the module and resets all of the PPBI bits in the Buffer Descriptor Table to '0'. This bit also activates the internal pull-up resistors, if they are enabled. Thus, this bit can be used as a soft attach/detach to the USB. Although all Status and control bits are ignored when this bit is clear, the module needs to be fully preconfigured prior to setting this bit. This bit cannot be set until the USB module is supplied with an active clock source. If the PLL is being used, it should be enabled at least two milliseconds (enough time for the PLL to lock) before attempting to set the USBEN bit.

Note: When disabling the USB module, make sure the SUSPND bit (UCON<1>) is clear prior to clearing the USBEN bit. Clearing the USBEN bit, when the module is in the suspended state, may prevent the module from fully powering down.

PIC18(L)F2X/45K50

REGISTER 24-1: UCON: USB CONTROL REGISTER

| U-0 | R/W-0 | R-x | R/C-0 | R/W-0 | R/W-0 | R/W-0 | U-0 |
|-------|--------|-----|--------|----------------------|--------|--------|-----|
| — | PPBRST | SE0 | PKTDIS | USBEN ⁽¹⁾ | RESUME | SUSPND | — |
| bit 7 | bit 0 | | | | | | |

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

- bit 7 **Unimplemented:** Read as '0'
- bit 6 **PPBRST:** Ping-Pong Buffers Reset bit
1 = Reset all Ping-Pong Buffer Pointers to the Even Buffer Descriptor (BD) banks
0 = Ping-Pong Buffer Pointers not being reset
- bit 5 **SE0:** Live Single-Ended Zero Flag bit
1 = Single-ended zero active on the USB bus
0 = No single-ended zero detected
- bit 4 **PKTDIS:** Packet Transfer Disable bit
1 = SIE token and packet processing disabled, automatically set when a SETUP token is received
0 = SIE token and packet processing enabled
- bit 3 **USBEN:** USB Module Enable bit⁽¹⁾
1 = USB module and supporting circuitry enabled (device attached)
0 = USB module and supporting circuitry disabled (device detached)
- bit 2 **RESUME:** Resume Signaling Enable bit
1 = Resume signaling activated
0 = Resume signaling disabled
- bit 1 **SUSPND:** Suspend USB bit
1 = USB module and supporting circuitry in Power Conserve mode, SIE clock inactive
0 = USB module and supporting circuitry in normal operation, SIE clock clocked at the configured rate
- bit 0 **Unimplemented:** Read as '0'

Note 1: This bit cannot be set if the USB module does not have an appropriate clock source.

The PPBRST bit (UCON<6>) controls the Reset status when Double-Buffering mode (ping-pong buffering) is used. When the PPBRST bit is set, all Ping-Pong Buffer Pointers are set to the Even buffers. PPBRST has to be cleared by firmware. This bit is ignored in buffering modes not using ping-pong buffering.

The PKTDIS bit (UCON<4>) is a flag indicating that the SIE has disabled packet transmission and reception. This bit is set by the SIE when a SETUP token is received to allow setup processing. This bit cannot be set by the microcontroller, only cleared; clearing it allows the SIE to continue transmission and/or reception. Any pending events within the Buffer Descriptor Table will still be available, indicated within the USTAT register's FIFO buffer.

The RESUME bit (UCON<2>) allows the peripheral to perform a remote wake-up by executing Resume signaling. To generate a valid remote wake-up, firmware must set RESUME for 10 ms and then clear the bit. For more information on "resume signaling", see the **"Universal Serial Bus Specification Revision 2.0"**.

The SUSPND bit (UCON<1>) places the module and supporting circuitry in a Low-Power mode. The input clock to the SIE is also disabled. This bit should be set by the software in response to an IDLEIF interrupt. It should be reset by the microcontroller firmware after an ACTVIF interrupt is observed. When this bit is active, the device remains attached to the bus but the transceiver outputs remain Idle. The voltage on the VUSB3V3 pin may vary depending on the value of this bit. Setting this bit before a IDLEIF request will result in unpredictable bus behavior.

Note: While in Suspend mode, a typical bus-powered USB device is limited to 2.5 mA of current. This is the complete current which may be drawn by the PIC® device and its supporting circuitry. Care should be taken to assure minimum current draw when the device enters Suspend mode.

24.2.2 USB CONFIGURATION REGISTER (UCFG)

Prior to communicating over USB, the module's associated internal and/or external hardware must be configured. Most of the configuration is performed with the UCFG register ([Register 24-2](#)). The UCFG register contains most of the bits that control the system level behavior of the USB module. These include:

- Bus Speed (full speed versus low speed)
- On-Chip Pull-up Resistor Enable
- Ping-Pong Buffer Usage

The UTEYE bit, UCFG<7>, enables eye pattern generation, which aids in module testing, debugging and USB certifications.

Note: The USB speed, transceiver and pull-up should only be configured during the module setup phase. It is not recommended to switch these settings while the module is enabled.

24.2.2.1 Internal Transceiver

The USB peripheral has a built-in, USB 2.0, full-speed and low-speed capable transceiver, internally connected to the SIE. This feature is useful for low-cost, single chip applications. Enabling the USB module (USBEN = 1) will also enable the internal transceiver. The FSEN bit (UCFG<2>) controls the transceiver speed; setting the bit enables full-speed operation.

The on-chip USB pull-up resistors are controlled by the UPUEN bit (UCFG<4>). They can only be selected when the on-chip transceiver is enabled.

The internal USB transceiver obtains power from the VUSB3V3 pin. In order to meet USB signaling level specifications, VUSB3V3 must be supplied with a voltage source between 3.0V and 3.6V. The best electrical signal quality is obtained when a 3.3V supply is used and locally bypassed with a high quality ceramic capacitor. The capacitor should be placed as close as possible to the VUSB3V3 and Vss pins found on the same edge of the package.

The D+ and D- signal lines can be routed directly to their respective pins on the USB connector or cable (for hard-wired applications). No additional resistors, capacitors, or magnetic components are required as the D+ and D- drivers have controlled slew rate and output impedance intended to match with the characteristic impedance of the USB cable.

In order to meet the USB specifications, the traces should be less than 30 cm long. Ideally, these traces should be designed to have a characteristic impedance matching that of the USB cable.

PIC18(L)F2X/45K50

REGISTER 24-2: UCFG: USB CONFIGURATION REGISTER (BANKED F39h)

| R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|--------|-----|------------------------|-------------------------|---------------------|-------|----------|
| UTEYE | UOEMON | — | UPUEN ^(1,2) | UTRDIS ^(1,3) | FSEN ⁽¹⁾ | | PPB<1:0> |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **UTEYE:** USB Eye Pattern Test Enable bit

1 = Eye pattern test is enabled

0 = Eye pattern test is disabled

bit 6 **UOEMON:** USB OE Monitor Enable bit

1 = UOE signal is active, indicating intervals during which the D+/D- lines are driving

0 = UOE signal is inactive

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

UPUEN: USB On-Chip Pull-up Enable bit^(1,2)

1 = On-chip pull-up is enabled (pull-up on D+ with FSEN = 1 or D- with FSEN = 0)

0 = On-chip pull-up is disabled

bit 3 **UTRDIS:** On-Chip Transceiver Disable bit^(1,3)

1 = On-chip transceiver is disabled

0 = On-chip transceiver is active

bit 2 **FSEN:** Full-Speed Enable bit⁽¹⁾

1 = Full-speed device: controls transceiver edge rates; requires input clock at 48 MHz

0 = Low-speed device: controls transceiver edge rates; requires input clock at 6 MHz

bit 1-0 **PPB<1:0>:** Ping-Pong Buffers Configuration bits

11 = Even/Odd ping-pong buffers are enabled for Endpoints 1 to 15

10 = Even/Odd ping-pong buffers are enabled for all endpoints

01 = Even/Odd ping-pong buffer are enabled for OUT Endpoint 0

00 = Even/Odd ping-pong buffers are disabled

Note 1: The UPUEN, UTRDIS and FSEN bits should never be changed while the USB module is enabled. These values must be preconfigured prior to enabling the module.

2: This bit is only valid when the on-chip transceiver is active (UTRDIS = 0); otherwise, it is ignored.

3: If UTRDIS is set, the UOE signal will be active, independent of the UOEMON bit setting.

24.2.2.2 Internal Pull-up Resistors

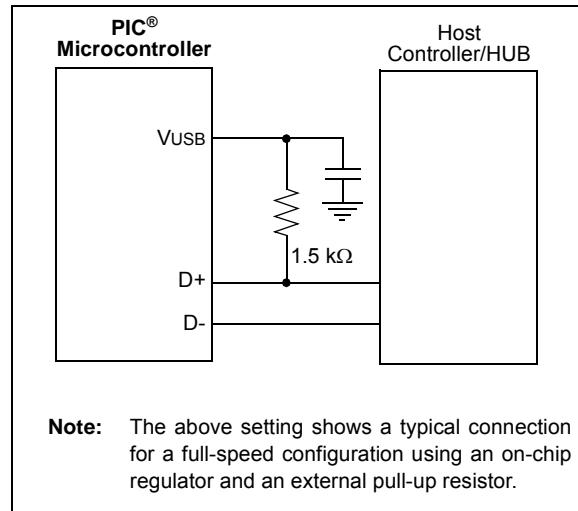
The PIC18F2X/45K50 devices have built-in pull-up resistors designed to meet the requirements for low-speed and full-speed USB. The UPUEN bit (UCFG<4>) enables the internal pull-ups. [Figure 24-1](#) shows the pull-ups and their control.

Note: The official USB specifications require that USB devices must never source any current onto the +5V VBUS line of the USB cable. Additionally, USB devices must never source any current on the D+ and D- data lines whenever the +5V VBUS line is less than 1.17V. In order to meet this requirement, applications which are not purely bus powered should monitor the VBUS line and avoid turning on the USB module and the D+ or D- pull-up resistor until VBUS is greater than 1.17V. VBUS can be connected to a resistive divider and monitored by an analog capable pin.

24.2.2.3 External Pull-up Resistors

External pull-up may also be used. The VUSB3V3 pin may be used to pull up D+ or D-. The pull-up resistor must be 1.5 k Ω ($\pm 5\%$) as required by the USB specifications. [Figure 24-2](#) shows an example.

FIGURE 24-2: EXTERNAL CIRCUITRY



24.2.2.4 Ping-Pong Buffer Configuration

The usage of ping-pong buffers is configured using the PPB<1:0> bits. Refer to [Section 24.4.4 “Ping-Pong Buffering”](#) for a complete explanation of the ping-pong buffers.

24.2.2.5 Eye Pattern Test Enable

An automatic eye pattern test can be generated by the module when the UCFG<7> bit is set. The eye pattern output will be observable based on module settings, meaning that the user is first responsible for configuring the SIE clock settings, pull-up resistor and Transceiver mode. In addition, the module has to be enabled.

Once UTEYE is set, the module emulates a switch from a receive to transmit state and will start transmitting a J-K-J-K bit sequence (K-J-K-J for full speed). The sequence will be repeated indefinitely while the Eye Pattern Test mode is enabled.

Note that this bit should never be set while the module is connected to an actual USB system. This Test mode is intended for board verification to aid with USB certification tests. It is intended to show a system developer the noise integrity of the USB signals which can be affected by board traces, impedance mismatches and proximity to other system components. It does not properly test the transition from a receive to a transmit state. Although the eye pattern is not meant to replace the more complex USB certification test, it should aid during first order system debugging.

PIC18(L)F2X/45K50

24.2.3 USB STATUS REGISTER (USTAT)

The USB Status register reports the transaction status within the SIE. When the SIE issues a USB transfer complete interrupt, USTAT should be read to determine the status of the transfer. USTAT contains the transfer endpoint number, direction and Ping-Pong Buffer Pointer value (if used).

Note: The data in the USB Status register is valid two SIE clocks after the TRNIF interrupt flag is asserted.

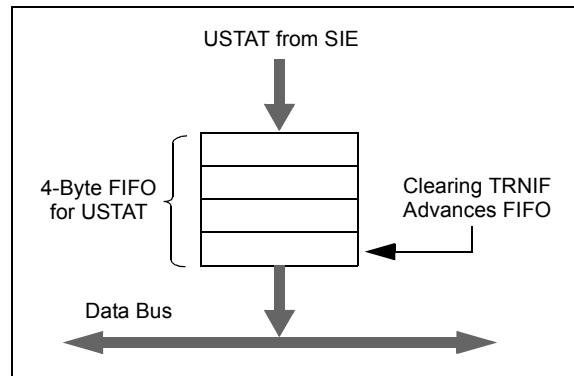
In low-speed operation with the system clock operating at 48 MHz, a delay may be required between receiving the TRNIF interrupt and processing the data in the USTAT register.

The USTAT register is actually a read window into a four-byte status FIFO, maintained by the SIE. It allows the microcontroller to process one transfer while the SIE processes additional endpoints (Figure 24-3). When the SIE completes using a buffer for reading or writing data, it updates the USTAT register. If another USB transfer is performed before a transaction complete interrupt is serviced, the SIE will store the status of the next transfer into the status FIFO.

Clearing the transfer complete flag bit, TRNIF, causes the SIE to advance the FIFO. If the next data in the FIFO holding register is valid, the SIE will reassert the interrupt within 6 TCY of clearing TRNIF. If no additional data is present, TRNIF will remain clear; USTAT data will no longer be reliable.

Note: If an endpoint request is received while the USTAT FIFO is full, the SIE will automatically issue a NAK back to the host.

FIGURE 24-3: USTAT FIFO



REGISTER 24-3: USTAT: USB STATUS REGISTER (ACCESS F64h)

| U-0 | R-x | R-x | R-x | R-x | R-x | R-x | U-0 |
|-------|-----|-----------|-----|-----|---------------------|-----|-------|
| — | | ENDP<3:0> | | DIR | PPBI ⁽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 **Unimplemented:** Read as '0'

bit 6-3 **ENDP<3:0>:** Encoded Number of Last Endpoint Activity bits
(represents the number of the BDT updated by the last USB transfer)

1111 = Endpoint 15

1110 = Endpoint 14

.

.

.

0001 = Endpoint 1

0000 = Endpoint 0

bit 2 **DIR:** Last BD Direction Indicator bit

1 = The last transaction was an IN token

0 = The last transaction was an OUT or SETUP token

bit 1 **PPBI:** Ping-Pong BD Pointer Indicator bit⁽¹⁾

1 = The last transaction was to the Odd BD bank

0 = The last transaction was to the Even BD bank

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

Note 1: This bit is only valid for endpoints with available Even and Odd BD registers.

24.2.4 USB ENDPOINT CONTROL

Each of the 16 possible bidirectional endpoints has its own independent control register, UEPn (where 'n' represents the endpoint number). Each register has an identical complement of control bits. The prototype is shown in [Register 24-4](#).

The EPHSHK bit (UEPn<4>) controls handshaking for the endpoint; setting this bit enables USB handshaking. Typically, this bit is always set except when using isochronous endpoints.

The EPCONDIS bit (UEPn<3>) is used to enable or disable USB control operations (SETUP) through the endpoint. Clearing this bit enables SETUP transactions. Note that the corresponding EPINEN and EPOUTEN bits must be set to enable IN and OUT

transactions. For Endpoint 0, this bit should always be cleared since the USB specifications identify Endpoint 0 as the default control endpoint.

The EPOUTEN bit (UEPn<2>) is used to enable or disable USB OUT transactions from the host. Setting this bit enables OUT transactions. Similarly, the EPINEN bit (UEPn<1>) enables or disables USB IN transactions from the host.

The EPSTALL bit (UEPn<0>) is used to indicate a STALL condition for the endpoint. If a STALL is issued on a particular endpoint, the EPSTALL bit for that endpoint pair will be set by the SIE. This bit remains set until it is cleared through firmware, or until the SIE is reset.

REGISTER 24-4: UEPn: USB ENDPOINT n CONTROL REGISTER (UEP0 THROUGH UEP15)

| U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-----|--------|----------|---------|--------|------------------------|
| — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL ⁽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 | EPHSHK: Endpoint Handshake Enable bit 1 = Endpoint handshake enabled 0 = Endpoint handshake disabled (typically used for isochronous endpoints) |
| bit 3 | EPCONDIS: Bidirectional Endpoint Control bit If EPOUTEN = 1 and EPINEN = 1: 1 = Disable Endpoint n from control transfers; only IN and OUT transfers allowed 0 = Enable Endpoint n for control (SETUP) transfers; IN and OUT transfers also allowed |
| bit 2 | EPOUTEN: Endpoint Output Enable bit 1 = Endpoint n output enabled 0 = Endpoint n output disabled |
| bit 1 | EPINEN: Endpoint Input Enable bit 1 = Endpoint n input enabled 0 = Endpoint n input disabled |
| bit 0 | EPSTALL: Endpoint STALL Enable bit ⁽¹⁾ 1 = Endpoint n is stalled 0 = Endpoint n is not stalled |

Note 1: Valid only if Endpoint n is enabled; otherwise, the bit is ignored.

PIC18(L)F2X/45K50

24.2.5 USB ADDRESS REGISTER (UADDR)

The USB Address register contains the unique USB address that the peripheral will decode when active. UADDR is reset to 00h when a USB Reset is received, indicated by URSTIF, or when a Reset is received from the microcontroller. The USB address must be written by the microcontroller during the USB setup phase (enumeration) as part of the Microchip USB firmware support.

24.2.6 USB FRAME NUMBER REGISTERS (UFRMH:UFRML)

The Frame Number registers contain the 11-bit frame number. The low-order byte is contained in UFRML, while the three high-order bits are contained in UFRMH. The register pair is updated with the current frame number whenever a SOF token is received. For the microcontroller, these registers are read-only. The Frame Number registers are primarily used for isochronous transfers. The contents of the UFRMH and UFRML registers are only valid when the 48 MHz SIE clock is active (i.e., contents are inaccurate when SUSPND (UCON<1>) bit = 1).

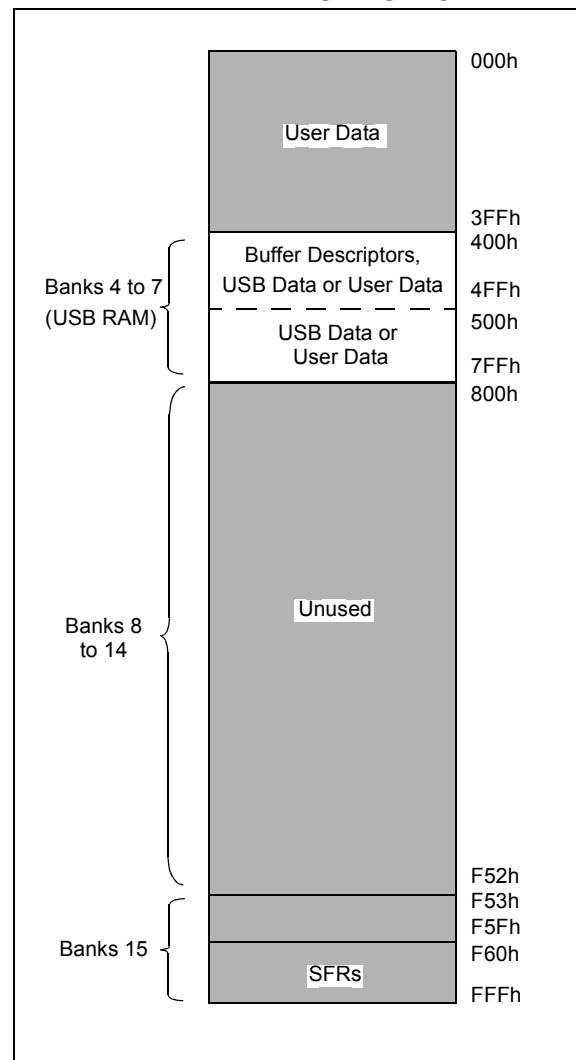
24.3 USB RAM

USB data moves between the microcontroller core and the SIE through a memory space known as the USB RAM. This is a special dual access memory that is mapped into the normal data memory space in Banks 4 through 7 (400h to 7FFh) for a total of 1024 bytes (Figure 24-4).

Bank 4 (400h through 4FFh) is used specifically for endpoint buffer control. Depending on the type of buffering being used, all but eight bytes of Bank 4 may also be available for use as USB buffer space.

Although USB RAM is available to the microcontroller as data memory, the sections that are being accessed by the SIE should not be accessed by the microcontroller. A semaphore mechanism is used to determine the access to a particular buffer at any given time. This is discussed in [Section 24.4.1.1 “Buffer Ownership”](#).

FIGURE 24-4: IMPLEMENTATION OF USB RAM IN DATA MEMORY SPACE



24.4 Buffer Descriptors and the Buffer Descriptor Table

The registers in Bank 4 are used specifically for endpoint buffer control in a structure known as the Buffer Descriptor Table (BDT). This provides a flexible method for users to construct and control endpoint buffers of various lengths and configuration.

The BDT is composed of Buffer Descriptors (BD) which are used to define and control the actual buffers in the USB RAM space. Each BD, in turn, consists of four registers, where n represents one of the 64 possible BDs (range of 0 to 63):

- BDnSTAT: BD Status register
- BDnCNT: BD Byte Count register
- BDnADRL: BD Address Low register
- BDnADRH: BD Address High register

BDs always occur as a four-byte block in the sequence: BDnSTAT:BDnCNT:BDnADRL:BDnADRH. The address of BDnSTAT is always an offset of $(4n - 1)$ (in hexadecimal) from 400h, with n being the buffer descriptor number.

Depending on the buffering configuration used ([Section 24.4.4 “Ping-Pong Buffering”](#)), there are up to 32, 33 or 64 sets of buffer descriptors. At a minimum, the BDT must be at least eight bytes long. This is because the USB specification mandates that every device must have Endpoint 0, with both input and output for initial setup. Depending on the endpoint and buffering configuration, the BDT can be as long as 256 bytes.

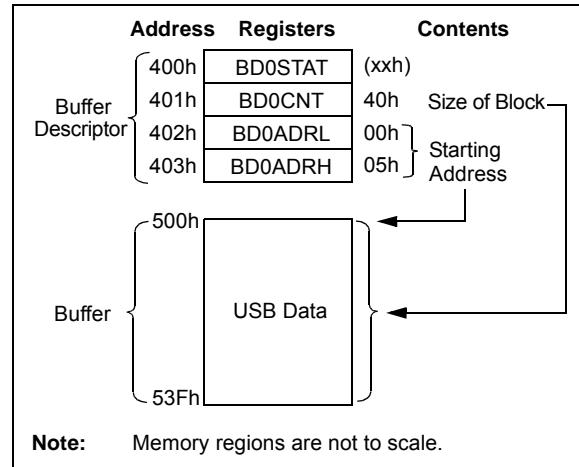
Although they can be thought of as Special Function Registers, the Buffer Descriptor Status and Address registers are not hardware mapped, as conventional microcontroller SFRs in Bank 15 are. If the endpoint corresponding to a particular BD is not enabled, its registers are not used. Instead of appearing as unimplemented addresses, however, they appear as available RAM. Only when an endpoint is enabled by setting the UEPn<1> bit does the memory at those addresses become functional as BD registers. As with any address in the data memory space, the BD registers have an indeterminate value on any device Reset.

[Figure 24-5](#) provides an example of a BD for a 64-byte buffer, starting at 500h. A particular set of BD registers is only valid if the corresponding endpoint has been enabled using the UEPn register. All BD registers are available in USB RAM. The BD for each endpoint should be set up prior to enabling the endpoint.

24.4.1 BD STATUS AND CONFIGURATION

Buffer descriptors not only define the size of an endpoint buffer, but also determine its configuration and control. Most of the configuration is done with the BD Status register, BDnSTAT. Each BD has its own unique and correspondingly numbered BDnSTAT register.

FIGURE 24-5: EXAMPLE OF A BUFFER DESCRIPTOR



Unlike other control registers, the bit configuration for the BDnSTAT register is context sensitive. There are two distinct configurations, depending on whether the microcontroller or the USB module is modifying the BD and buffer at a particular time. Only three bit definitions are shared between the two.

24.4.1.1 Buffer Ownership

Because the buffers and their BDs are shared between the CPU and the USB module, a simple semaphore mechanism is used to distinguish which is allowed to update the BD and associated buffers in memory.

This is done by using the UOWN bit (BDnSTAT<7>) as a semaphore to distinguish which is allowed to update the BD and associated buffers in memory. UOWN is the only bit that is shared between the two configurations of BDnSTAT.

When UOWN is clear, the BD entry is “owned” by the microcontroller core. When the UOWN bit is set, the BD entry and the buffer memory are “owned” by the USB peripheral. The core should not modify the BD or its corresponding data buffer during this time. Note that the microcontroller core can still read BDnSTAT while the SIE owns the buffer and vice versa.

The buffer descriptors have a different meaning based on the source of the register update. Prior to placing ownership with the USB peripheral, the user can configure the basic operation of the peripheral through the BDnSTAT bits. During this time, the byte count and buffer location registers can also be set.

PIC18(L)F2X/45K50

When UOWN is set, the user can no longer depend on the values that were written to the BDs. From this point, the SIE updates the BDs as necessary, overwriting the original BD values. The BDnSTAT register is updated by the SIE with the token PID and the transfer count, BDnCNT, is updated.

Note: The firmware should not set the UOWN bit in the same instruction cycles as any other modifications to the BDnSTAT soft register. The UOWN bit should only be set in a separate instruction cycle, only after all other bits in BDnSTAT (and address/count registers) have been fully updated.

The BDnSTAT byte of the BDT should always be the last byte updated when preparing to arm an endpoint. The SIE will clear the UOWN bit when a transaction has completed.

No hardware mechanism exists to block access when the UOWN bit is set. Thus, unexpected behavior can occur if the microcontroller attempts to modify memory when the SIE owns it. Similarly, reading such memory may produce inaccurate data until the USB peripheral returns ownership to the microcontroller.

24.4.1.2 BDnSTAT Register (CPU Mode)

When UOWN = 0, the microcontroller core owns the BD. At this point, the other seven bits of the register take on control functions.

The Data Toggle Sync Enable bit, DTSEN (BDnSTAT<3>), controls data toggle parity checking. Setting DTSEN enables data toggle synchronization by the SIE. When enabled, it checks the data packet's parity against the value of DTS (BDnSTAT<6>). If a packet arrives with an incorrect synchronization, the data will essentially be ignored. It will not be written to the USB RAM and the USB transfer complete interrupt flag will not be set. The SIE will send an ACK token back to the host to Acknowledge receipt, however. The effects of the DTSEN bit on the SIE are summarized in [Table 24-1](#).

The Buffer Stall bit, BSTALL (BDnSTAT<2>), provides support for control transfers, usually one-time stalls on Endpoint 0. It also provides support for the SET_FEATURE/CLEAR_FEATURE commands specified in Chapter 9 of the USB specification; typically, continuous STALLs to any endpoint other than the default control endpoint.

The BSTALL bit enables buffer stalls. Setting BSTALL causes the SIE to return a STALL token to the host if a received token would use the BD in that location. The EPSTALL bit in the corresponding UEPn control register is set and a STALL interrupt is generated when a STALL is issued to the host. The UOWN bit remains set and the BDs are not changed unless a SETUP token is received. In this case, the STALL condition is cleared and the ownership of the BD is returned to the microcontroller core.

The BD<9:8> bits (BDnSTAT<1:0>) store the two Most Significant digits of the SIE byte count; the lower eight digits are stored in the corresponding BDnCNT register. See [Section 24.4.2 “BD Byte Count”](#) for more information.

TABLE 24-1: EFFECT OF DTSEN BIT ON ODD/EVEN (DATA0/DATA1) PACKET RECEPTION

| OUT Packet from Host | BDnSTAT Settings | | Device Response after Receiving Packet | | | |
|----------------------|------------------|-----|--|------|-------|--------------------------|
| | DTSEN | DTS | Handshake | UOWN | TRNIF | BDnSTAT and USTAT Status |
| DATA0 | 1 | 0 | ACK | 0 | 1 | Updated |
| DATA1 | 1 | 0 | ACK | 1 | 0 | Not Updated |
| DATA0 | 1 | 1 | ACK | 1 | 0 | Not Updated |
| DATA1 | 1 | 1 | ACK | 0 | 1 | Updated |
| Either | 0 | x | ACK | 0 | 1 | Updated |
| Either, with error | x | x | NAK | 1 | 0 | Not Updated |

Legend: x = don't care

REGISTER 24-5: BD_nSTAT: BUFFER DESCRIPTOR _n STATUS REGISTER (BD0STAT THROUGH BD63STAT), CPU MODE (BANKED 4xxh)

| R/W-x | R/W-x | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x |
|---------------------|--------------------|------------------|------------------|-------|--------|-------|-------|
| UOWN ⁽¹⁾ | DTS ⁽²⁾ | — ⁽³⁾ | — ⁽³⁾ | DTSEN | BSTALL | BC9 | BC8 |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit
-n = Value at POR

W = Writable bit
'1' = Bit is set

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

| | |
|---------|---|
| bit 7 | UOWN: USB Own bit ⁽¹⁾ 0 = The microcontroller core owns the BD and its corresponding buffer |
| bit 6 | DTS: Data Toggle Synchronization bit ⁽²⁾ 1 = Data 1 packet 0 = Data 0 packet |
| bit 5-4 | Unimplemented: These bits should always be programmed to '0' ⁽³⁾ . |
| bit 3 | DTSEN: Data Toggle Synchronization Enable bit 1 = Data toggle synchronization is enabled; data packets with incorrect Sync value will be ignored except for a SETUP transaction, which is accepted even if the data toggle bits do not match 0 = No data toggle synchronization is performed |
| bit 2 | BSTALL: Buffer Stall Enable bit 1 = Buffer stall enabled; STALL handshake issued if a token is received that would use the BD in the given location (UOWN bit remains set, BD value is unchanged) 0 = Buffer stall disabled |
| bit 1-0 | BC<9:8>: Byte Count 9 and 8 bits The byte count bits represent the number of bytes that will be transmitted for an IN token or received during an OUT token. Together with BC<7:0>, the valid byte counts are 0-1023. |

- Note 1:** This bit must be initialized by the user to the desired value prior to enabling the USB module.
- 2:** This bit is ignored unless DTSEN = 1.
- 3:** If these bits are set, USB communication may not work. Hence, these bits should always be maintained as '0'.

PIC18(L)F2X/45K50

24.4.1.3 BDnSTAT Register (SIE Mode)

When the BD and its buffer are owned by the SIE, most of the bits in BDnSTAT take on a different meaning. The configuration is shown in [Register 24-6](#). Once the UOWN bit is set, any data or control settings previously written there by the user will be overwritten with data from the SIE.

The BDnSTAT register is updated by the SIE with the token Packet Identifier (PID) which is stored in BDnSTAT<5:3>. The transfer count in the corresponding BDnCNT register is updated. Values that overflow the 8-bit register carry over to the two Most Significant digits of the count, stored in BDnSTAT<1:0>.

24.4.2 BD BYTE COUNT

The byte count represents the total number of bytes that will be transmitted during an IN transfer. After an IN transfer, the SIE will return the number of bytes sent to the host.

For an OUT transfer, the byte count represents the maximum number of bytes that can be received and stored in USB RAM. After an OUT transfer, the SIE will return the actual number of bytes received. If the number of bytes received exceeds the corresponding byte count, the data packet will be rejected and a NAK handshake will be generated. When this happens, the byte count will not be updated.

REGISTER 24-6: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD63STAT), SIE MODE (DATA RETURNED BY THE SIE TO THE MCU)

| R/W-x | U-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|-------|-------|-------|-------|-------|-------|-------|-------|
| UOWN | — | PID3 | PID2 | PID1 | PID0 | BC9 | BC8 |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7

UOWN: USB Own bit

1 = The SIE owns the BD and its corresponding buffer

bit 6

Reserved: Not written by the SIE

bit 5-2

PID<3:0>: Packet Identifier bits

The received token PID value of the last transfer (IN, OUT or SETUP transactions only).

bit 1-0

BC<9:8>: Byte Count 9 and 8 bits

These bits are updated by the SIE to reflect the actual number of bytes received on an OUT transfer and the actual number of bytes transmitted on an IN transfer.

The 10-bit byte count is distributed over two registers. The lower eight bits of the count reside in the BDnCNT register. The upper two bits reside in BDnSTAT<1:0>. This represents a valid byte range of 0 to 1023.

24.4.3 BD ADDRESS VALIDATION

The BD Address register pair contains the starting RAM address location for the corresponding endpoint buffer. No mechanism is available in hardware to validate the BD address.

If the value of the BD address does not point to an address in the USB RAM, or if it points to an address within another endpoint's buffer, data is likely to be lost or overwritten. Similarly, overlapping a receive buffer (OUT endpoint) with a BD location in use can yield unexpected results. When developing USB applications, the user may want to consider the inclusion of software-based address validation in their code.

24.4.4 PING-PONG BUFFERING

An endpoint is defined to have a ping-pong buffer when it has two sets of BD entries: one set for an Even transfer and one set for an Odd transfer. This allows the CPU to process one BD while the SIE is processing the other BD. Double-buffering BDs in this way allows for maximum throughput to/from the USB.

The USB module supports four modes of operation:

- No ping-pong support
- Ping-pong buffer support for OUT Endpoint 0 only
- Ping-pong buffer support for all endpoints
- Ping-pong buffer support for all other Endpoints except Endpoint 0

The ping-pong buffer settings are configured using the PPB<1:0> bits in the UCFG register.

The USB module keeps track of the Ping-Pong Pointer individually for each endpoint. All pointers are initially reset to the Even BD when the module is enabled. After

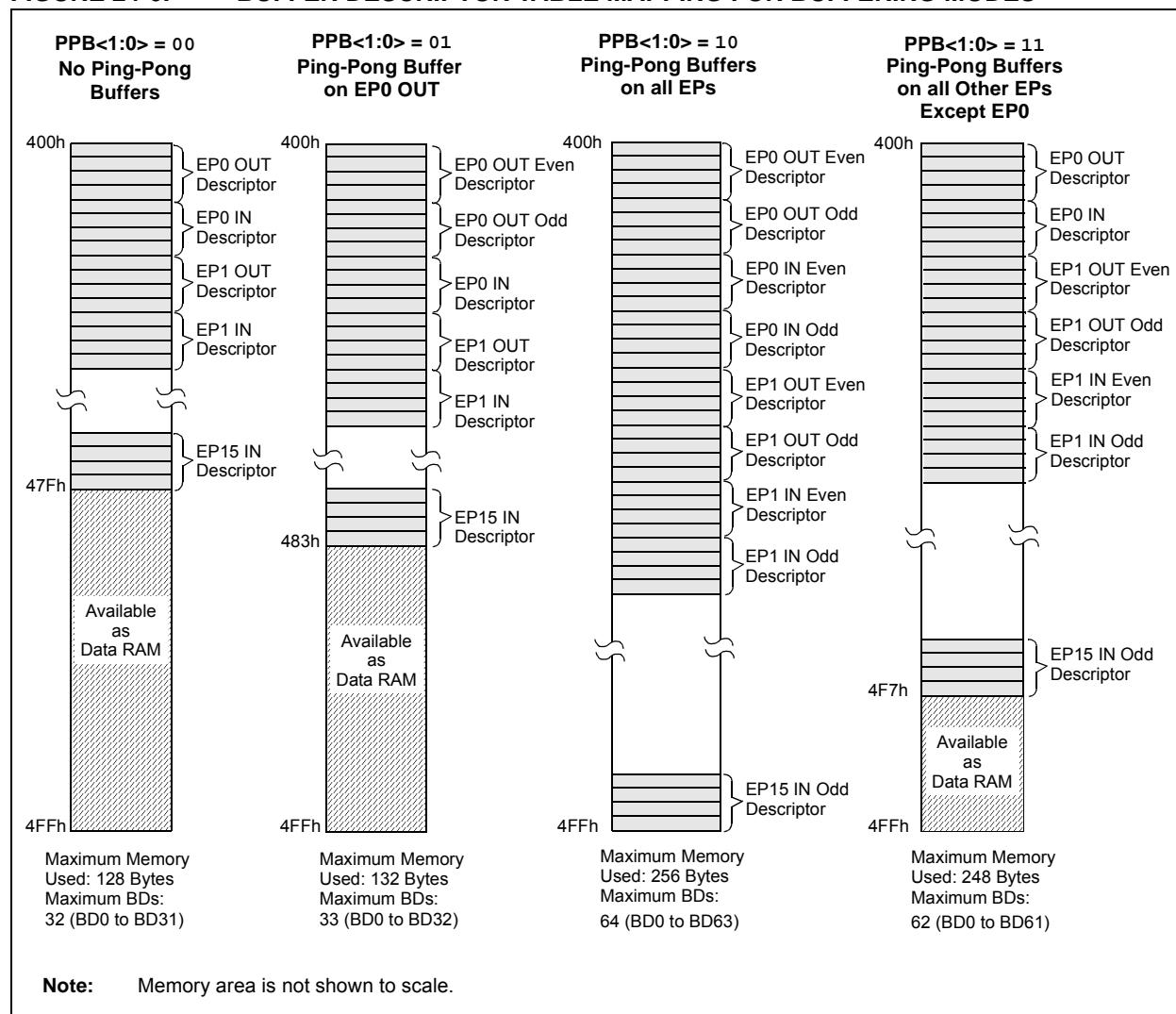
the completion of a transaction (UOWN cleared by the SIE), the pointer is toggled to the Odd BD. After the completion of the next transaction, the pointer is toggled back to the Even BD and so on.

The Even/Odd status of the last transaction is stored in the PPBI bit of the USTAT register. The user can reset all Ping-Pong Pointers to Even using the PPBRST bit.

Figure 24-6 shows the four different modes of operation and how USB RAM is filled with the BDs.

BDs have a fixed relationship to a particular endpoint, depending on the buffering configuration. The mapping of BDs to endpoints is detailed in **Table 24-2**. This relationship also means that gaps may occur in the BDT if endpoints are not enabled contiguously. This theoretically means that the BDs for disabled endpoints could be used as buffer space. In practice, users should avoid using such spaces in the BDT unless a method of validating BD addresses is implemented.

FIGURE 24-6: BUFFER DESCRIPTOR TABLE MAPPING FOR BUFFERING MODES



PIC18(L)F2X/45K50

TABLE 24-2: ASSIGNMENT OF BUFFER DESCRIPTORS FOR THE DIFFERENT BUFFERING MODES

| Endpoint | BDs Assigned to Endpoint | | | | | | | |
|----------|--------------------------|----|----------------------------------|----|----------------------------------|----------------|---|----------------|
| | Mode 0 (No Ping-Pong) | | Mode 1 (Ping-Pong on EP0 OUT) | | Mode 2 (Ping-Pong on all EPs) | | Mode 3 (Ping-Pong on all other EPs, except EP0) | |
| | Out | In | Out | In | Out | In | Out | In |
| 0 | 0 | 1 | 0 (E), 1 (O) | 2 | 0 (E), 1 (O) | 2 (E), 3 (O) | 0 | 1 |
| 1 | 2 | 3 | 3 | 4 | 4 (E), 5 (O) | 6 (E), 7 (O) | 2 (E), 3 (O) | 4 (E), 5 (O) |
| 2 | 4 | 5 | 5 | 6 | 8 (E), 9 (O) | 10 (E), 11 (O) | 6 (E), 7 (O) | 8 (E), 9 (O) |
| 3 | 6 | 7 | 7 | 8 | 12 (E), 13 (O) | 14 (E), 15 (O) | 10 (E), 11 (O) | 12 (E), 13 (O) |
| 4 | 8 | 9 | 9 | 10 | 16 (E), 17 (O) | 18 (E), 19 (O) | 14 (E), 15 (O) | 16 (E), 17 (O) |
| 5 | 10 | 11 | 11 | 12 | 20 (E), 21 (O) | 22 (E), 23 (O) | 18 (E), 19 (O) | 20 (E), 21 (O) |
| 6 | 12 | 13 | 13 | 14 | 24 (E), 25 (O) | 26 (E), 27 (O) | 22 (E), 23 (O) | 24 (E), 25 (O) |
| 7 | 14 | 15 | 15 | 16 | 28 (E), 29 (O) | 30 (E), 31 (O) | 26 (E), 27 (O) | 28 (E), 29 (O) |
| 8 | 16 | 17 | 17 | 18 | 32 (E), 33 (O) | 34 (E), 35 (O) | 30 (E), 31 (O) | 32 (E), 33 (O) |
| 9 | 18 | 19 | 19 | 20 | 36 (E), 37 (O) | 38 (E), 39 (O) | 34 (E), 35 (O) | 36 (E), 37 (O) |
| 10 | 20 | 21 | 21 | 22 | 40 (E), 41 (O) | 42 (E), 43 (O) | 38 (E), 39 (O) | 40 (E), 41 (O) |
| 11 | 22 | 23 | 23 | 24 | 44 (E), 45 (O) | 46 (E), 47 (O) | 42 (E), 43 (O) | 44 (E), 45 (O) |
| 12 | 24 | 25 | 25 | 26 | 48 (E), 49 (O) | 50 (E), 51 (O) | 46 (E), 47 (O) | 48 (E), 49 (O) |
| 13 | 26 | 27 | 27 | 28 | 52 (E), 53 (O) | 54 (E), 55 (O) | 50 (E), 51 (O) | 52 (E), 53 (O) |
| 14 | 28 | 29 | 29 | 30 | 56 (E), 57 (O) | 58 (E), 59 (O) | 54 (E), 55 (O) | 56 (E), 57 (O) |
| 15 | 30 | 31 | 31 | 32 | 60 (E), 61 (O) | 62 (E), 63 (O) | 58 (E), 59 (O) | 60 (E), 61 (O) |

Legend: (E) = Even transaction buffer, (O) = Odd transaction buffer

TABLE 24-3: SUMMARY OF USB BUFFER DESCRIPTOR TABLE REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------------------------|---------------------|--------------------|---------------------|---------------------|---|--|-------|-------|
| BDnSTAT ⁽¹⁾ | UOWN | DTS ⁽⁴⁾ | PID3 ⁽²⁾ | PID2 ⁽²⁾ | PID1 ⁽²⁾ DTSEN ⁽³⁾ | PID0 ⁽²⁾ BSTALL ⁽³⁾ | BC9 | BC8 |
| BDnCNT ⁽¹⁾ | Byte Count | | | | | | | |
| BDnADRL ⁽¹⁾ | Buffer Address Low | | | | | | | |
| BDnADRH ⁽¹⁾ | Buffer Address High | | | | | | | |

- Note 1:** For buffer descriptor registers, n may have a value of 0 to 63. For the sake of brevity, all 64 registers are shown as one generic prototype. All registers have indeterminate Reset values (xxxx xxxx).
- 2:** Bits 5 through 2 of the BDnSTAT register are used by the SIE to return PID<3:0> values once the register is turned over to the SIE (UOWN bit is set). Once the registers have been under SIE control, the values written for DTSEN and BSTALL are no longer valid.
- 3:** Prior to turning the buffer descriptor over to the SIE (UOWN bit is cleared), bits 5 through 2 of the BDnSTAT register are used to configure the DTSEN and BSTALL settings.
- 4:** This bit is ignored unless DTSEN = 1.

24.5 USB Interrupts

The USB module can generate multiple interrupt conditions. To accommodate all of these interrupt sources, the module is provided with its own interrupt logic structure, similar to that of the microcontroller. USB interrupts are enabled with one set of control registers and trapped with a separate set of flag registers. All sources are funneled into a single USB interrupt request, USBIF (PIR3<2>), in the microcontroller's interrupt logic.

Figure 24-7 shows the interrupt logic for the USB module. There are two layers of interrupt registers in the USB module. The top level consists of overall USB Status interrupts; these are enabled and flagged in the UIR and UIE registers, respectively. The second level consists of USB error conditions, which are enabled and flagged in the UEIR and UEIE registers. An interrupt condition in any of these triggers a USB Error Interrupt Flag (UERRIF) in the top level.

Interrupts may be used to trap routine events in a USB transaction. Figure 24-8 shows some common events within a USB frame and their corresponding interrupts.

FIGURE 24-7: USB INTERRUPT LOGIC FUNNEL

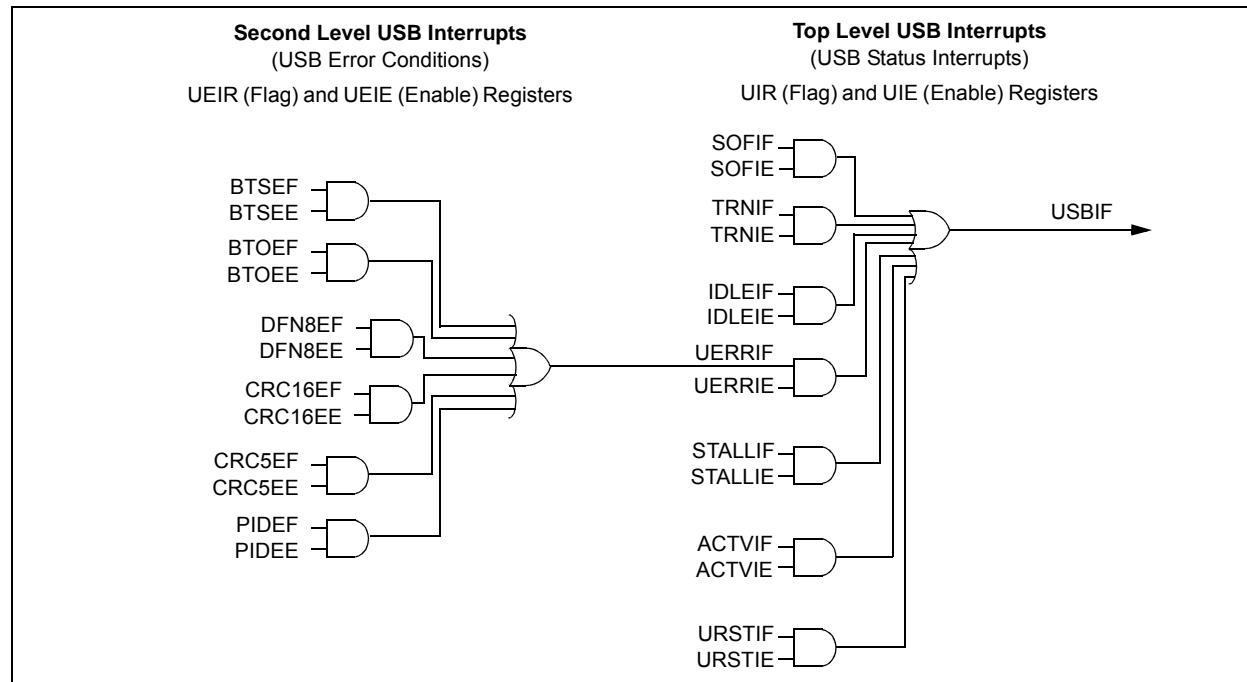
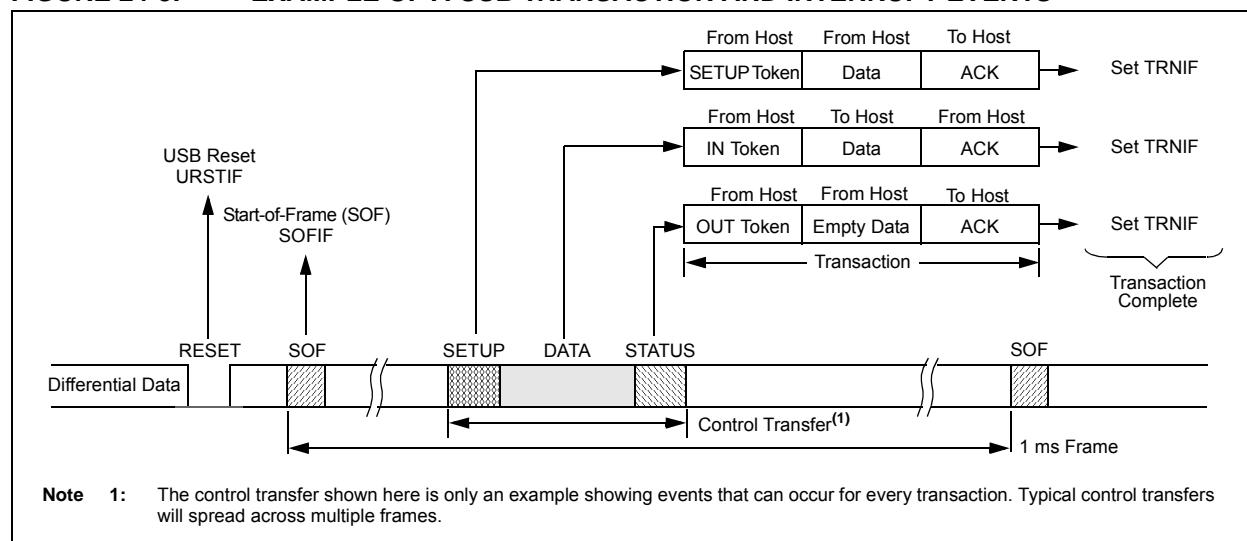


FIGURE 24-8: EXAMPLE OF A USB TRANSACTION AND INTERRUPT EVENTS



PIC18(L)F2X/45K50

24.5.1 USB INTERRUPT STATUS REGISTER (UIR)

The USB Interrupt Status register ([Register 24-7](#)) contains the flag bits for each of the USB Status interrupt sources. Each of these sources has a corresponding interrupt enable bit in the UIE register. All of the USB status flags are ORed together to generate the USBIF interrupt flag for the microcontroller's interrupt funnel.

Once an interrupt bit has been set by the SIE, it must be cleared by software by writing a '0'. The flag bits can also be set in software which can aid in firmware debugging.

When the USB module is in the Low-Power Suspend mode (UCON<1> = 1), the SIE does not get clocked. When in this state, the SIE cannot process packets and, therefore, cannot detect new interrupt conditions other than the Activity Detect Interrupt, ACTVIF. The ACTVIF bit is typically used by USB firmware to detect when the microcontroller should bring the USB module out of the Low-Power Suspend mode (UCON<1> = 0).

REGISTER 24-7: UIR: USB INTERRUPT STATUS REGISTER

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R/W-0 |
|-------|-------|---------|-----------------------|----------------------|-----------------------|-----------------------|--------|
| — | SOFIF | STALLIF | IDLEIF ⁽¹⁾ | TRNIF ⁽²⁾ | ACTVIF ⁽³⁾ | UERRIF ⁽⁴⁾ | URSTIF |
| 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 **SOFIF:** Start-of-Frame Token Interrupt bit
1 = A Start-of-Frame token received by the SIE
0 = No Start-of-Frame token received by the SIE
- bit 5 **STALLIF:** A STALL Handshake Interrupt bit
1 = A STALL handshake was sent by the SIE
0 = A STALL handshake has not been sent
- bit 4 **IDLEIF:** Idle Detect Interrupt bit⁽¹⁾
1 = Idle condition detected (constant Idle state of 3 ms or more)
0 = No Idle condition detected
- bit 3 **TRNIF:** Transaction Complete Interrupt bit⁽²⁾
1 = Processing of pending transaction is complete; read USTAT register for endpoint information
0 = Processing of pending transaction is not complete or no transaction is pending
- bit 2 **ACTVIF:** Bus Activity Detect Interrupt bit⁽³⁾
1 = Activity on the D+/D- lines was detected
0 = No activity detected on the D+/D- lines
- bit 1 **UERRIF:** USB Error Condition Interrupt bit⁽⁴⁾
1 = An unmasked error condition has occurred
0 = No unmasked error condition has occurred
- bit 0 **URSTIF:** USB Reset Interrupt bit
1 = Valid USB Reset occurred; 00h is loaded into UADDR register
0 = No USB Reset has occurred

- Note 1:** Once an Idle state is detected, the user may want to place the USB module in Suspend mode.
- 2:** Clearing this bit will cause the USTAT FIFO to advance (valid only for IN, OUT and SETUP tokens).
- 3:** This bit is typically unmasked only following the detection of a UIDLE interrupt event.
- 4:** Only error conditions enabled through the UEIE register will set this bit. This bit is a status bit only and cannot be set or cleared by the user.

24.5.1.1 Bus Activity Detect Interrupt Bit (ACTVIF)

The ACTVIF bit cannot be cleared immediately after the USB module wakes up from Suspend or while the USB module is suspended. A few clock cycles are required to synchronize the internal hardware state machine before the ACTVIF bit can be cleared by firmware. Clearing the ACTVIF bit before the internal hardware is synchronized may not have an effect on the value of ACTVIF. Additionally, if the USB module uses the clock from the 48 MHz PLL source, then after

clearing the SUSPND bit, the USB module may not be immediately operational while waiting for the 48 MHz PLL to lock. The application code should clear the ACTVIF flag as shown in [Example 24-1](#).

Only one ACTVIF interrupt is generated when resuming from the USB bus Idle condition. If user firmware clears the ACTVIF bit, the bit will not immediately become set again, even when there is continuous bus traffic. Bus traffic must cease long enough to generate another IDLEIF condition before another ACTVIF interrupt can be generated.

EXAMPLE 24-1: CLEARING ACTVIF BIT (UIR<2>)

Assembly:

```
BCF      UCON, SUSPND
LOOP:
    BTFSS   UIR, ACTVIF
    BRA     DONE
    BCF      UIR, ACTVIF
    BRA     LOOP
DONE:
```

C:

```
UCONbits.SUSPND = 0;
while (UIRbits.ACTVIF) { UIRbits.ACTVIF = 0; }
```

PIC18(L)F2X/45K50

24.5.2 USB INTERRUPT ENABLE REGISTER (UIE)

The USB Interrupt Enable register ([Register 24-8](#)) contains the enable bits for the USB Status interrupt sources. Setting any of these bits will enable the respective interrupt source in the UIR register.

The values in this register only affect the propagation of an interrupt condition to the microcontroller's interrupt logic. The flag bits are still set by their interrupt conditions, allowing them to be polled and serviced without actually generating an interrupt.

REGISTER 24-8: UIE: USB INTERRUPT ENABLE REGISTER

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|---------|--------|-------|--------|--------|--------|
| — | SOFIE | STALLIE | IDLEIE | TRNIE | ACTVIE | UERRIE | URSTIE |
| bit 7 | | | | | bit 0 | | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

bit 6 **SOFIE:** Start-of-Frame Token Interrupt Enable bit

 1 = Start-of-Frame token interrupt enabled
 0 = Start-of-Frame token interrupt disabled

bit 5 **STALLIE:** STALL Handshake Interrupt Enable bit

 1 = STALL interrupt enabled
 0 = STALL interrupt disabled

bit 4 **IDLEIE:** Idle Detect Interrupt Enable bit

 1 = Idle detect interrupt enabled
 0 = Idle detect interrupt disabled

bit 3 **TRNIE:** Transaction Complete Interrupt Enable bit

 1 = Transaction interrupt enabled
 0 = Transaction interrupt disabled

bit 2 **ACTVIE:** Bus Activity Detect Interrupt Enable bit

 1 = Bus activity detect interrupt enabled
 0 = Bus activity detect interrupt disabled

bit 1 **UERRIE:** USB Error Interrupt Enable bit

 1 = USB error interrupt enabled
 0 = USB error interrupt disabled

bit 0 **URSTIE:** USB Reset Interrupt Enable bit

 1 = USB Reset interrupt enabled
 0 = USB Reset interrupt disabled

24.5.3 USB ERROR INTERRUPT STATUS REGISTER (UEIR)

The USB Error Interrupt Status register ([Register 24-9](#)) contains the flag bits for each of the error sources within the USB peripheral. Each of these sources is controlled by a corresponding interrupt enable bit in the UEIE register. All of the USB error flags are ORed together to generate the USB Error Interrupt Flag (UERRIF) at the top level of the interrupt logic.

Each error bit is set as soon as the error condition is detected. Thus, the interrupt will typically not correspond with the end of a token being processed.

Once an interrupt bit has been set by the SIE, it must be cleared by software by writing a '0'.

REGISTER 24-9: UEIR: USB ERROR INTERRUPT STATUS REGISTER

| R/C-0 | U-0 | U-0 | R/C-0 | R/C-0 | R/C-0 | R/C-0 | R/C-0 |
|-------|-------|-----|-------|--------|---------|--------|-------|
| BTSEF | — | — | BTOEF | DFN8EF | CRC16EF | CRC5EF | PIDEF |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit

C = Clearable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

| | |
|---------|---|
| bit 7 | BTSEF: Bit Stuff Error Flag bit 1 = A bit stuff error has been detected 0 = No bit stuff error |
| bit 6-5 | Unimplemented: Read as '0' |
| bit 4 | BTOEF: Bus Turnaround Time-out Error Flag bit 1 = Bus turnaround time-out has occurred (more than 16 bit times of Idle from previous EOP elapsed) 0 = No bus turnaround time-out |
| bit 3 | DFN8EF: Data Field Size Error Flag bit 1 = The data field was not an integral number of bytes 0 = The data field was an integral number of bytes |
| bit 2 | CRC16EF: CRC16 Failure Flag bit 1 = The CRC16 failed 0 = The CRC16 passed |
| bit 1 | CRC5EF: CRC5 Host Error Flag bit 1 = The token packet was rejected due to a CRC5 error 0 = The token packet was accepted |
| bit 0 | PIDEF: PID Check Failure Flag bit 1 = PID check failed 0 = PID check passed |

PIC18(L)F2X/45K50

24.5.4 USB ERROR INTERRUPT ENABLE REGISTER (UEIE)

The USB Error Interrupt Enable register ([Register 24-10](#)) contains the enable bits for each of the USB error interrupt sources. Setting any of these bits will enable the respective error interrupt source in the UEIR register to propagate into the UERR bit at the top level of the interrupt logic.

As with the UIE register, the enable bits only affect the propagation of an interrupt condition to the microcontroller's interrupt logic. The flag bits are still set by their interrupt conditions, allowing them to be polled and serviced without actually generating an interrupt.

REGISTER 24-10: UEIE: USB ERROR INTERRUPT ENABLE REGISTER

| R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-----|-------|--------|---------|--------|-------|
| BTSEE | — | — | BTOEE | DFN8EE | CRC16EE | CRC5EE | PIDEE |
| bit 7 | 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 **BTSEE:** Bit Stuff Error Interrupt Enable bit

1 = Bit stuff error interrupt enabled
0 = Bit stuff error interrupt disabled

bit 6-5 **Unimplemented:** Read as '0'

bit 4 **BTOEE:** Bus Turnaround Time-out Error Interrupt Enable bit

1 = Bus turnaround time-out error interrupt enabled
0 = Bus turnaround time-out error interrupt disabled

bit 3 **DFN8EE:** Data Field Size Error Interrupt Enable bit

1 = Data field size error interrupt enabled
0 = Data field size error interrupt disabled

bit 2 **CRC16EE:** CRC16 Failure Interrupt Enable bit

1 = CRC16 failure interrupt enabled
0 = CRC16 failure interrupt disabled

bit 1 **CRC5EE:** CRC5 Host Error Interrupt Enable bit

1 = CRC5 host error interrupt enabled
0 = CRC5 host error interrupt disabled

bit 0 **PIDEE:** PID Check Failure Interrupt Enable bit

1 = PID check failure interrupt enabled
0 = PID check failure interrupt disabled

24.6 USB Power Modes

Many USB applications will likely have several different sets of power requirements and configuration. The most common power modes encountered are Bus Power Only, Self-Power Only and Dual Power with Self-Power Dominance. The most common cases are presented here. Also provided is a means of estimating the current consumption of the USB transceiver.

24.6.1 BUS POWER ONLY

In Bus Power Only mode, all power for the application is drawn from the USB (Figure 24-9). This is effectively the simplest power method for the device.

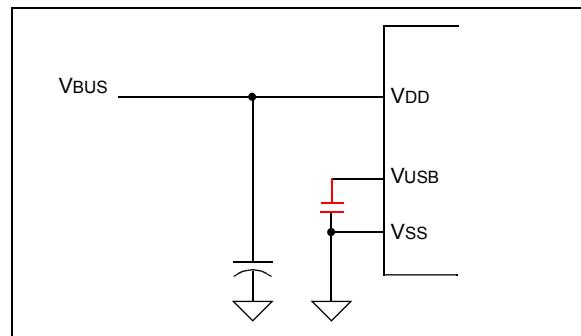
In order to meet the inrush current requirements of the USB 2.0 specifications, the total effective capacitance appearing across VBUS and ground must be no more than 10 μ F. If not, some kind of inrush liming is required. For more details, see section 7.2.4 of the USB 2.0 specification.

According to the USB 2.0 specification, all USB devices must also support a Low-Power Suspend mode. In the USB Suspend mode, devices must consume no more than 2.5 mA from the 5V VBUS line of the USB cable.

The host signals the USB device to enter the Suspend mode by stopping all USB traffic to that device for more than 3 ms. This condition will cause the IDLEIF bit in the UIR register to become set.

During the USB Suspend mode, the D+ or D- pull-up resistor must remain active, which will consume some of the allowed suspend current: 2.5 mA budget.

FIGURE 24-9: BUS POWER ONLY



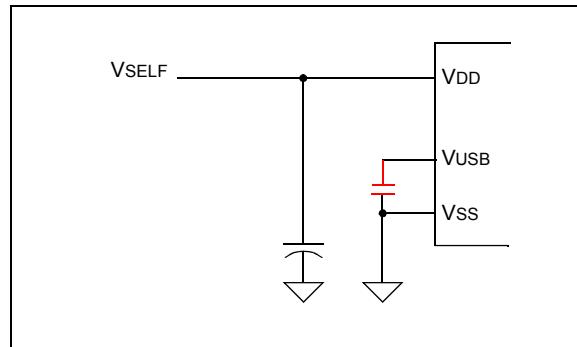
24.6.2 SELF-POWER ONLY

In Self-Power Only mode, the USB application provides its own power, with very little power being pulled from the USB. Figure 24-10 shows an example.

In order to meet compliance specifications, the USB module (and the D+ or D- pull-up resistor) should not be enabled until the host actively drives VBUS high.

The application should never source any current onto the 5V VBUS pin of the USB cable.

FIGURE 24-10: SELF-POWER ONLY

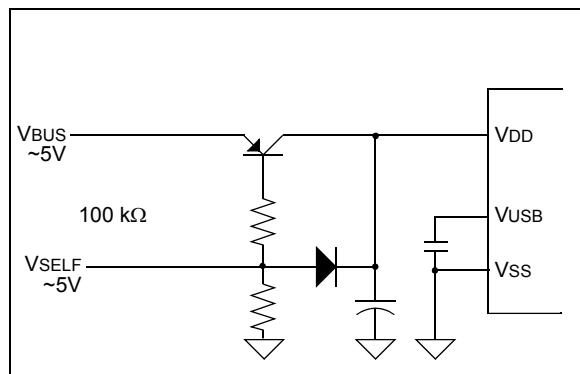


24.6.3 DUAL POWER WITH SELF-POWER DOMINANCE

Some applications may require a dual power option. This allows the application to use internal power primarily, but switch to power from the USB when no internal power is available. [Figure 24-11](#) shows a simple Dual Power with Self-Power Dominance mode example, which automatically switches between Self-Power Only and USB Bus Power Only modes.

Dual power devices must also meet all of the special requirements for inrush current and Suspend mode current and must not enable the USB module until VBUS is driven high. See [Section 24.6.1 “Bus Power Only”](#) and [Section 24.6.2 “Self-Power Only”](#) for descriptions of those requirements. Additionally, dual power devices must never source current onto the 5V VBUS pin of the USB cable.

FIGURE 24-11: DUAL POWER EXAMPLE



Note: Users should keep in mind the limits for devices drawing power from the USB. According to USB Specification 2.0, this cannot exceed 100 mA per low-power device or 500 mA per high-power device.

24.6.4 USB TRANSCEIVER CURRENT CONSUMPTION

The USB transceiver consumes a variable amount of current depending on the characteristic impedance of the USB cable, the length of the cable, the V_{USB3V3} supply voltage and the actual data patterns moving across the USB cable. Longer cables have larger capacitances and consume more total energy when switching output states.

Data patterns that consist of “IN” traffic consume far more current than “OUT” traffic. IN traffic requires the PIC® device to drive the USB cable, whereas OUT traffic requires that the host drive the USB cable.

The data that is sent across the USB cable is NRZI encoded. In the NRZI encoding scheme, ‘0’ bits cause a toggling of the output state of the transceiver (either from a “J” state to a “K” state, or vice versa). With the exception of the effects of bit-stuffing, NRZI encoded ‘1’ bits do not cause the output state of the transceiver to change. Therefore, IN traffic consisting of data bits of value, ‘0’, cause the most current consumption, as the transceiver must charge/discharge the USB cable in order to change states.

More details about NRZI encoding and bit-stuffing can be found in the USB 2.0 specification’s section 7.1, although knowledge of such details is not required to make USB applications using the PIC18F2X/45K50 of microcontrollers. Among other things, the SIE handles bit-stuffing/unstuffing, NRZI encoding/decoding and CRC generation/checking in hardware.

The total transceiver current consumption will be application-specific. However, to help estimate how much current actually may be required in full-speed applications, [Equation 24-1](#) can be used.

[Example 24-2](#) shows how this equation can be used for a theoretical application.

EQUATION 24-1: ESTIMATING USB TRANSCEIVER CURRENT CONSUMPTION

$$I_{XCVR} = \frac{(60 \text{ mA} \cdot V_{USB3V3} \cdot P_{ZERO} \cdot P_{IN} \cdot L_{CABLE})}{(3.3V \cdot 5m)} + I_{PULLUP}$$

| | | |
|-----------------------|---|--|
| Legend: | V _{USB} : | Voltage applied to the V _{USB3V3} pin in volts. (Should be 3.0V to 3.6V.) |
| P _{ZERO} : | Percentage (in decimal) of the IN traffic bits sent by the PIC® device that are a value of '0'. | |
| P _{IN} : | Percentage (in decimal) of total bus bandwidth that is used for IN traffic. | |
| L _{CABLE} : | Length (in meters) of the USB cable. The USB 2.0 specification requires that full-speed applications use cables no longer than 5m. | |
| I _{PULLUP} : | Current which the nominal, 1.5 kΩ pull-up resistor (when enabled) must supply to the USB cable. On the host or hub end of the USB cable, 15 kΩ nominal resistors (14.25 kΩ to 24.8 kΩ) are present which pull both the D+ and D- lines to ground. During bus Idle conditions (such as between packets or during USB Suspend mode), this results in up to 218 μA of quiescent current drawn at 3.3V. I _{PULLUP} is also dependant on bus traffic conditions and can be as high as 2.2 mA when the USB bandwidth is fully utilized (either IN or OUT traffic) for data that drives the lines to the "K" state most of the time. | |

EXAMPLE 24-2: CALCULATING USB TRANSCEIVER CURRENT[†]

For this example, the following assumptions are made about the application:

- 3.3V will be applied to V_{USB3V3} and V_D, with the core voltage regulator enabled.
- This is a full-speed application that uses one interrupt IN endpoint that can send one packet of 64 bytes every 1 ms, with no restrictions on the values of the bytes being sent. The application may or may not have additional traffic on OUT endpoints.
- A regular USB "B" or "mini-B" connector will be used on the application circuit board.

In this case, P_{ZERO} = 100% = 1, because there should be no restriction on the value of the data moving through the IN endpoint. All 64 kBps of data could potentially be bytes of value, 00h. Since '0' bits cause toggling of the output state of the transceiver, they cause the USB transceiver to consume extra current charging/discharging the cable. In this case, 100% of the data bits sent can be of value '0'. This should be considered the "max" value, as normal data will consist of a fair mix of ones and zeros.

This application uses 64 kBps for IN traffic out of the total bus bandwidth of 1.5 MBps (12 Mbps), therefore:

$$P_{IN} = \frac{64 \text{ kBps}}{1.5 \text{ MBps}} = 4.3\% = 0.043$$

Since a regular "B" or "mini-B" connector is used in this application, the end user may plug in any type of cable up to the maximum allowed 5 m length. Therefore, we use the worst-case length:

$$L_{CABLE} = 5 \text{ meters}$$

Assume I_{PULLUP} = 2.2 mA. The actual value of I_{PULLUP} will likely be closer to 218 μA, but allow for the worst-case. USB bandwidth is shared between all the devices which are plugged into the root port (via hubs). If the application is plugged into a USB 1.1 hub that has other devices plugged into it, your device may see host to device traffic on the bus, even if it is not addressed to your device. Since any traffic, regardless of source, can increase the I_{PULLUP} current above the base 218 μA, it is safest to allow for the worst-case of 2.2 mA.

Therefore:

$$I_{XCVR} = \frac{(60 \text{ mA} \cdot 3.3V \cdot 1 \cdot 0.043 \cdot 5m)}{(3.3V \cdot 5m)} + 2.2 \text{ mA} = 4.8 \text{ mA}$$

The calculated value should be considered an approximation and additional guardband or application-specific product testing is recommended. The transceiver current is "in addition to" the rest of the current consumed by the PIC18F2X/45K50 device that is needed to run the core, drive the other I/O lines, power the various modules, etc.

PIC18(L)F2X/45K50

24.7 Oscillator

The USB module has specific clock requirements. For full-speed operation, the clock source must be 48 MHz. Even so, the microcontroller core and other peripherals are not required to run at that clock speed. Available clocking options are described in detail in [Section 3.14 “Oscillator Settings for USB”](#).

24.8 Interrupt-On-Change for D+/D- pins

The PIC18(L)F2X/45K50 has interrupt-on-change functionality on both D+ and D- data pins. This feature allows the device to detect voltage level changes when first connected to a USB host/hub.

The USB host/hub has 15K pull-down resistors on the D+ and D- pins. When the PIC18(L)F2X/45K50 attaches to the bus the D+ and D- pins can detect voltage changes. External resistors are needed for each pin to maintain a high state on the pins when detached.

The USB module must be disabled (`USBEN = 0`) for the interrupt-on-change to function. Enabling the USB module (`USBEN = 1`) will automatically disable the interrupt-on-change for D+ and D- pins. Refer to [Section 11.3.2 “Interrupt-on-Change”](#) and [Section 11.4.2 “Interrupt-on-Change”](#) for more details.

24.9 USB Firmware and Drivers

Microchip provides a number of application-specific resources, such as USB firmware and driver support. Refer to www.microchip.com for the latest firmware and driver support.

TABLE 24-4: REGISTERS ASSOCIATED WITH USB MODULE OPERATION⁽¹⁾

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|--------|----------|-----------|---------|--------|----------|-----------|-----------|---------|------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMROIF | INT0IF | IOCIF | 114 |
| IPR3 | — | — | — | — | CTMUIP | USBIP | TMR3GIP | TMR1GIP | 125 |
| PIR3 | — | — | — | — | CTMUIF | USBIF | TMR3GIF | TMR1GIF | 119 |
| PIE3 | — | — | — | — | CTMUIE | USBIE | TMR3GIE | TMR1GIE | 122 |
| UCON | — | PPBRST | SE0 | PKTDIS | USBEN | RESUME | SUSPND | — | 338 |
| UCFG | UTEYE | UOEMON | — | UPUEN | UTRDIS | FSEN | PPB<1:0>1 | — | 340 |
| USTAT | — | ENDP<3:0> | | | | DIR | PPBI | — | 342 |
| UADDR | — | ADDR<6:0> | | | | | | | 344 |
| UFRML | FRM<7:0> | | | | | | | | 337 |
| UFRMH | — | — | — | — | — | FRM<10:8> | | | 337 |
| UIR | — | SOFIF | STALLIF | IDLEIF | TRNIF | ACTVIF | UERRIF | URSTIF | 352 |
| UIE | — | SOFIE | STALLIE | IDLEIE | TRNIE | ACTVIE | UERRIE | URSTIE | 354 |
| UEIR | BTSEF | — | — | BTOEF | DFN8EF | CRC16EF | CRC5EF | PIDEF | 355 |
| UEIE | BTSEE | — | — | BTOEE | DFN8EE | CRC16EE | CRC5EE | PIDEE | 356 |
| UEP0 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP1 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP2 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP3 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP4 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP5 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP6 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP7 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP8 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP9 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP10 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP11 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP12 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP13 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP14 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |
| UEP15 | — | — | — | EPHSHK | EPCONDIS | EPOUTEN | EPINEN | EPSTALL | 343 |

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

Note 1: This table includes only those hardware mapped SFRs located in Bank 15 of the data memory space. The Buffer Descriptor registers, which are mapped into Bank 4 and are not true SFRs, are listed separately in [Table 24-3](#).

PIC18(L)F2X/45K50

24.10 Overview of USB

This section presents some of the basic USB concepts and useful information necessary to design a USB device. Although much information is provided in this section, there is a plethora of information provided within the USB specifications and class specifications. Thus, the reader is encouraged to refer to the USB specifications for more information (www.usb.org). If you are very familiar with the details of USB, then this section serves as a basic, high-level refresher of USB.

24.10.1 LAYERED FRAMEWORK

USB device functionality is structured into a layered framework graphically shown in [Figure 24-12](#). Each level is associated with a functional level within the device. The highest layer, other than the device, is the configuration. A device may have multiple configurations. For example, a particular device may have multiple power requirements based on Self-Power Only or Bus Power Only modes.

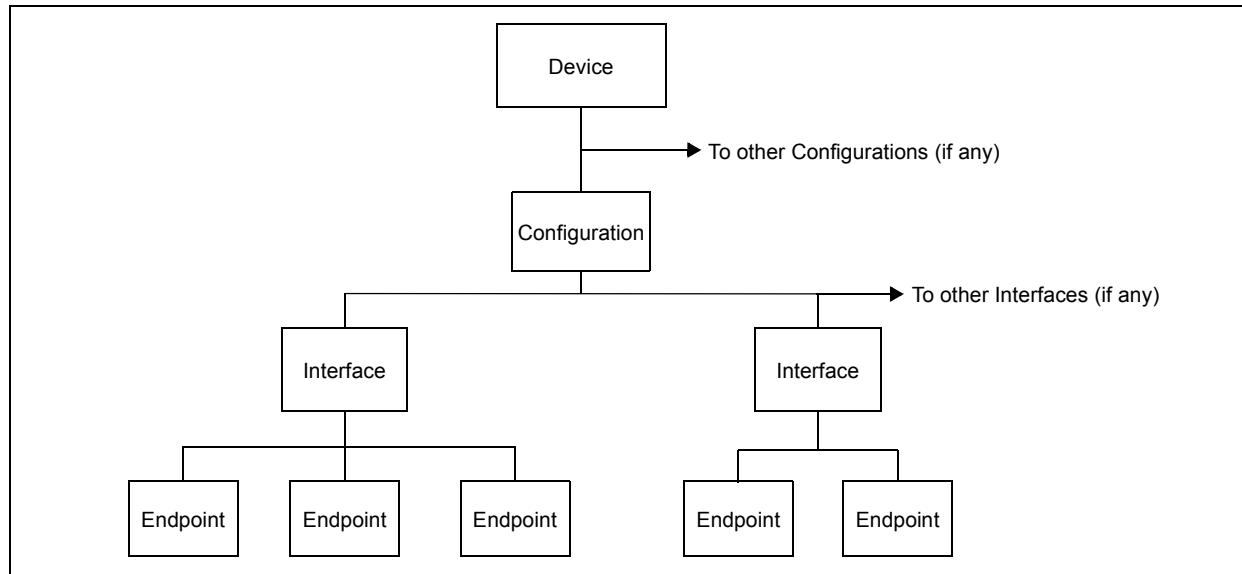
For each configuration, there may be multiple interfaces. Each interface could support a particular mode of that configuration.

Below the interface is the endpoint(s). Data is directly moved at this level. There can be as many as 16 bidirectional endpoints. Endpoint 0 is always a control endpoint and by default, when the device is on the bus, Endpoint 0 must be available to configure the device.

24.10.2 FRAMES

Information communicated on the bus is grouped into 1 ms time slots, referred to as frames. Each frame can contain many transactions to various devices and endpoints. [Figure 24-8](#) shows an example of a transaction within a frame.

FIGURE 24-12: USB LAYERS



24.10.3 TRANSFERS

There are four transfer types defined in the USB specification.

- **Isochronous:** This type provides a transfer method for large amounts of data (up to 1023 bytes) with timely delivery ensured; however, the data integrity is not ensured. This is good for streaming applications where small data loss is not critical, such as audio.
- **Bulk:** This type of transfer method allows for large amounts of data to be transferred with ensured data integrity; however, the delivery timeliness is not ensured.
- **Interrupt:** This type of transfer provides for ensured timely delivery for small blocks of data, plus data integrity is ensured.
- **Control:** This type provides for device setup control.

While full-speed devices support all transfer types, low-speed devices are limited to interrupt and control transfers only.

24.10.4 POWER

Power is available from the Universal Serial Bus. The USB specification defines the bus power requirements. Devices may either be self-powered or bus powered. Self-powered devices draw power from an external source, while bus powered devices use power supplied from the bus.

The USB specification limits the power taken from the bus. Each device is ensured 100 mA at approximately 5V (one unit load). Additional power may be requested, up to a maximum of 500 mA. Note that power above one unit load is a request and the host or hub is not obligated to provide the extra current. Thus, a device capable of consuming more than one unit load must be able to maintain a low-power configuration of a one unit load or less, if necessary.

The USB specification also defines a Suspend mode. In this situation, current must be limited to 500 μ A, averaged over 1 second. A device must enter a Suspend state after 3 ms of inactivity (i.e., no SOF tokens for 3 ms). A device entering Suspend mode must drop current consumption within 10 ms after Suspend. Likewise, when signaling a wake-up, the device must signal a wake-up within 10 ms of drawing current above the Suspend limit.

24.10.5 ENUMERATION

When the device is initially attached to the bus, the host enters an enumeration process in an attempt to identify the device. Essentially, the host interrogates the device, gathering information such as power consumption, data rates and sizes, protocol and other descriptive information; descriptors contain this information. A typical enumeration process would be as follows:

1. USB Reset: Reset the device. Thus, the device is not configured and does not have an address (address 0).
2. Get Device Descriptor: The host requests a small portion of the device descriptor.
3. USB Reset: Reset the device again.
4. Set Address: The host assigns an address to the device.
5. Get Device Descriptor: The host retrieves the device descriptor, gathering info such as manufacturer, type of device, maximum control packet size.
6. Get configuration descriptors.
7. Get any other descriptors.
8. Set a configuration.

The exact enumeration process depends on the host.

24.10.6 DESCRIPTORS

There are eight different standard descriptor types of which five are most important for this device.

24.10.6.1 Device Descriptor

The device descriptor provides general information, such as manufacturer, product number, serial number, the class of the device and the number of configurations. There is only one device descriptor.

24.10.6.2 Configuration Descriptor

The configuration descriptor provides information on the power requirements of the device and how many different interfaces are supported when in this configuration. There may be more than one configuration for a device (i.e., low-power and high-power configurations).

24.10.6.3 Interface Descriptor

The interface descriptor details the number of endpoints used in this interface, as well as the class of the interface. There may be more than one interface for a configuration.

24.10.6.4 Endpoint Descriptor

The endpoint descriptor identifies the transfer type ([Section 24.10.3 “Transfers”](#)) and direction, as well as some other specifics for the endpoint. There may be many endpoints in a device and endpoints may be shared in different configurations.

24.10.6.5 String Descriptor

Many of the previous descriptors reference one or more string descriptors. String descriptors provide human readable information about the layer ([Section 24.10.1 “Layered Framework”](#)) they describe. Often these strings show up in the host to help the user identify the device. String descriptors are generally optional to save memory and are encoded in a unicode format.

24.10.7 BUS SPEED

Each USB device must indicate its bus presence and speed to the host. This is accomplished through a 1.5 k Ω resistor which is connected to the bus at the time of the attachment event.

Depending on the speed of the device, the resistor either pulls up the D+ or D- line to 3.3V. For a low-speed device, the pull-up resistor is connected to the D- line. For a full-speed device, the pull-up resistor is connected to the D+ line.

24.10.8 CLASS SPECIFICATIONS AND DRIVERS

USB specifications include class specifications which operating system vendors optionally support. Examples of classes include Audio, Mass Storage, Communications and Human Interface (HID). In most cases, a driver is required at the host side to ‘talk’ to the USB device. In custom applications, a driver may need to be developed. Fortunately, drivers are available for most common host systems for the most common classes of devices. Thus, these drivers can be reused.

PIC18(L)F2X/45K50

25.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

The PIC18(L)F2X/45K50 devices have a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that sets both a device voltage trip point and the direction of change from that point. If the device experiences an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution branches to the interrupt vector address and the software responds to the interrupt.

25.1 Register – HLVD Control

REGISTER 25-1: HLVDCON: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER

| R/W-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-1 | | | |
|---------|-------|-------|--------|------------|-------|-------|-------|-------|--|--|
| VDIRMAG | BGVST | IRVST | HLVDEN | HLVDL<3:0> | | | | | | |
| bit 7 | | | | | | | | bit 0 | | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **VDIRMAG:** Voltage Direction Magnitude Select bit
1 = Event occurs when voltage equals or exceeds trip point (HLVDL<3:0>)
0 = Event occurs when voltage equals or falls below trip point (HLVDL<3:0>)
- bit 6 **BGVST:** Band Gap Reference Voltages Stable Status Flag bit
1 = Internal band gap voltage references are stable
0 = Internal band gap voltage reference is not stable
- bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit
1 = Indicates that the voltage detect logic will generate the interrupt flag at the specified voltage range
0 = Indicates that the voltage detect logic will not generate the interrupt flag at the specified voltage range and the HLVD interrupt should not be enabled
- bit 4 **HLVDEN:** High/Low-Voltage Detect Power Enable bit
1 = HLVD enabled
0 = HLVD disabled
- bit 3-0 **HLVDL<3:0>:** Voltage Detection Level bits⁽¹⁾
1111 = External analog input is used (input comes from the HLVDIN pin)
1110 = Maximum setting
.
.
0000 = Minimum setting

Note 1: See [Table 29-15](#) for specifications.

The High/Low-Voltage Detect Control register ([Register 25-1](#)) completely controls the operation of the HLVD module. This allows the circuitry to be "turned off" by the user under software control, which minimizes the current consumption for the device.

The module's block diagram is shown in [Figure 25-1](#).

The module is enabled by setting the HLVDEN bit (HLVDCON<4>). Each time the HLVD module is enabled, the circuitry requires some time to stabilize. The IRVST bit (HLVDCON<5>) is a read-only bit used to indicate when the circuit is stable. The module can only generate an interrupt after the circuit is stable and IRVST is set.

The VDIRMAG bit (HLVDCON<7>) determines the overall operation of the module. When VDIRMAG is cleared, the module monitors for drops in VDD below a predetermined set point. When the bit is set, the module monitors for rises in VDD above the set point.

25.2 Operation

When the HLVD module is enabled, a comparator uses an internally generated reference voltage as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a

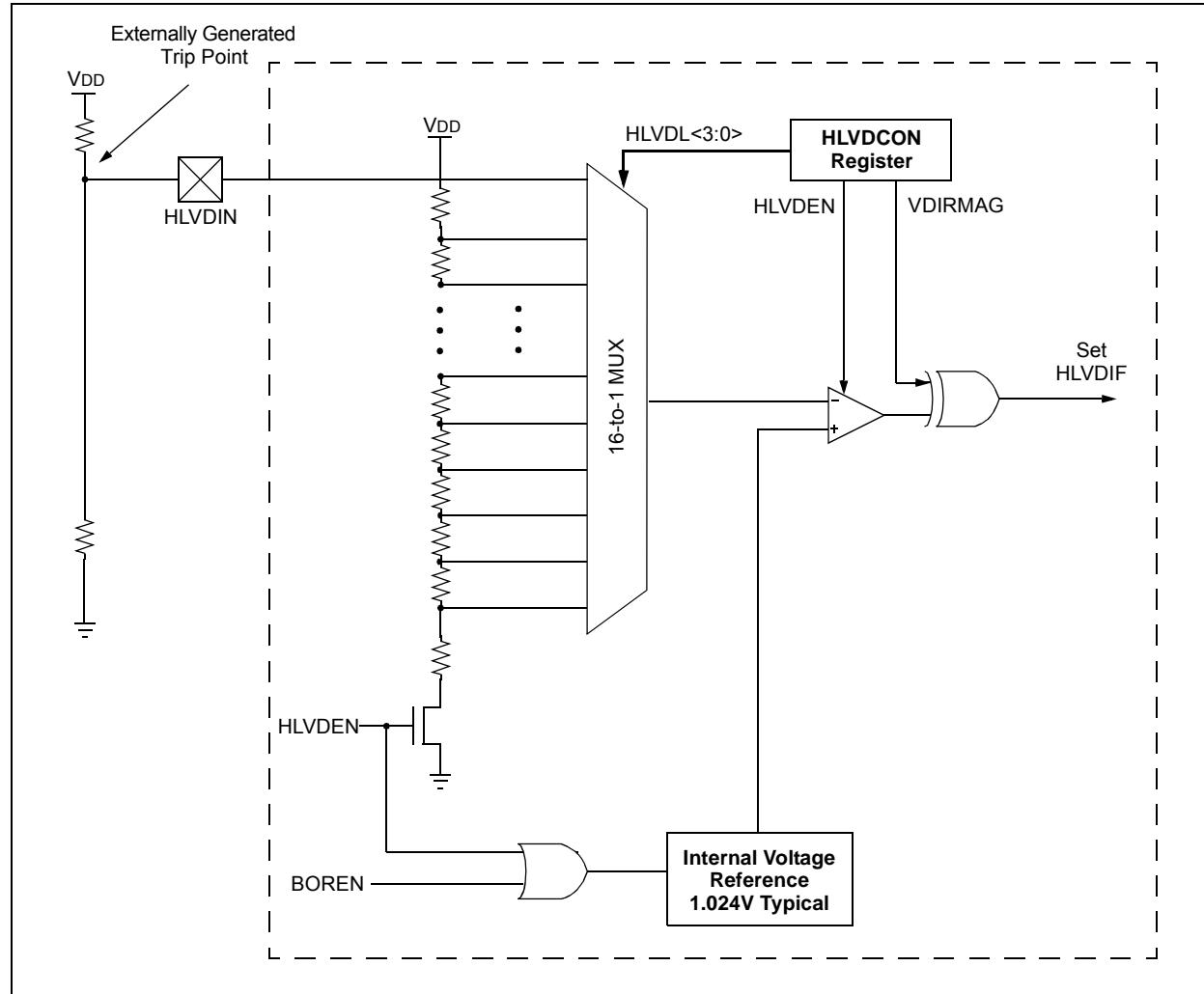
trip point voltage. The “trip point” voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module.

When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable to any of 16 values. The trip point is selected by programming the HLVDL<3:0> bits (HLVDCON<3:0>).

The HLVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits, HLVDL<3:0>, are set to ‘1111’. In this state, the comparator input is multiplexed from the external input pin, HLVDIN. This gives users the flexibility of configuring the High/Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

FIGURE 25-1: HLVD MODULE BLOCK DIAGRAM (WITH EXTERNAL INPUT)



25.3 HLVD Setup

To set up the HLVD module:

1. Select the desired HLVD trip point by writing the value to the HLVDL<3:0> bits.
2. Set the VDIRMAG bit to detect high voltage (VDIRMAG = 1) or low voltage (VDIRMAG = 0).
3. Enable the HLVD module by setting the HLVDEN bit.
4. Clear the HLVD interrupt flag (PIR2<2>), which may have been set from a previous interrupt.
5. If interrupts are desired, enable the HLVD interrupt by setting the HLVDIE and GIE/GIEH bits (PIE2<2> and INTCON<7>, respectively).

An interrupt will not be generated until the IRVST bit is set.

Note: Before changing any module settings (VDIRMAG, HLVDL<3:0>), first disable the module (HLVDEN = 0), make the changes and re-enable the module. This prevents the generation of false HLVD events.

25.4 Current Consumption

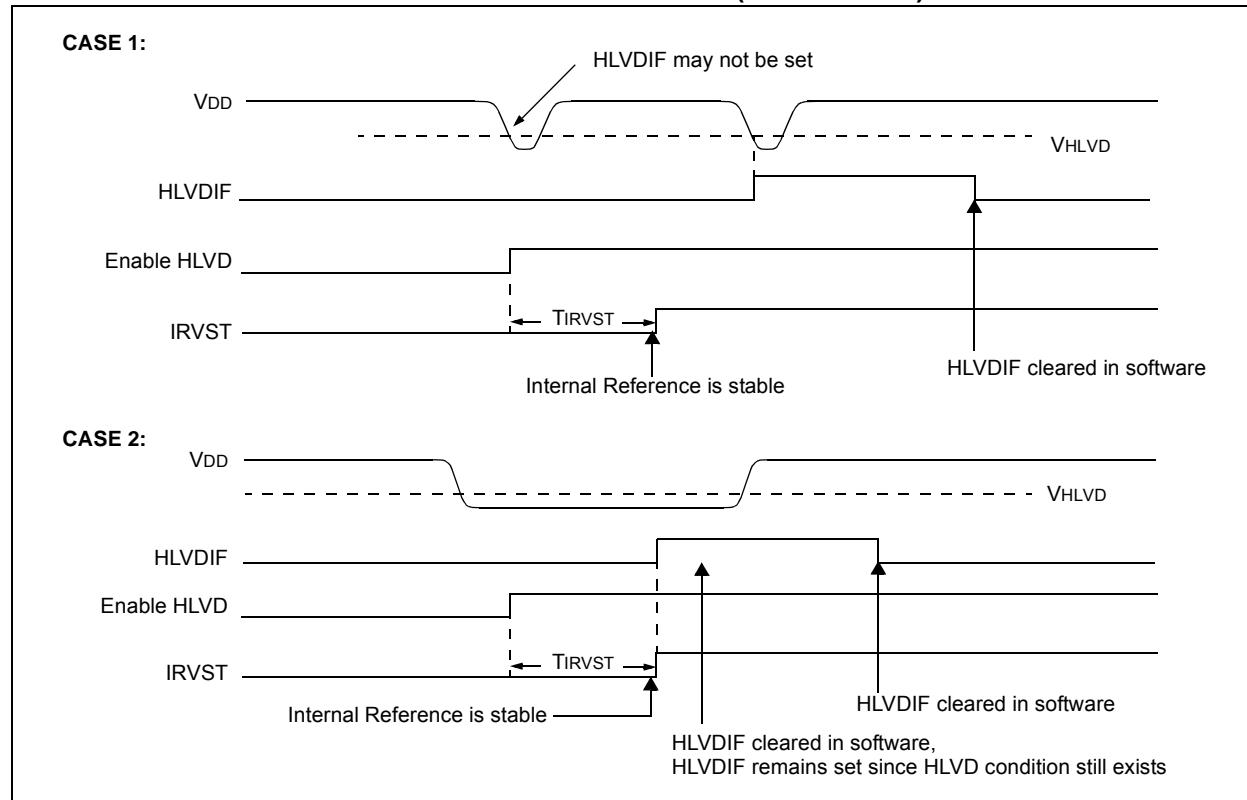
When the module is enabled, the HLVD comparator and voltage divider are enabled and consume static current. The total current consumption, when enabled, is specified in [Section 29.0 “Electrical Specifications”](#). Depending on the application, the HLVD module does not need to operate constantly. To reduce current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After such a check, the module could be disabled.

25.5 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in [Section 29.0 “Electrical Specifications”](#), may be used by other internal circuitry, such as the programmable Brown-out Reset. If the HLVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time, TIRVST, is an interval that is independent of device clock speed.

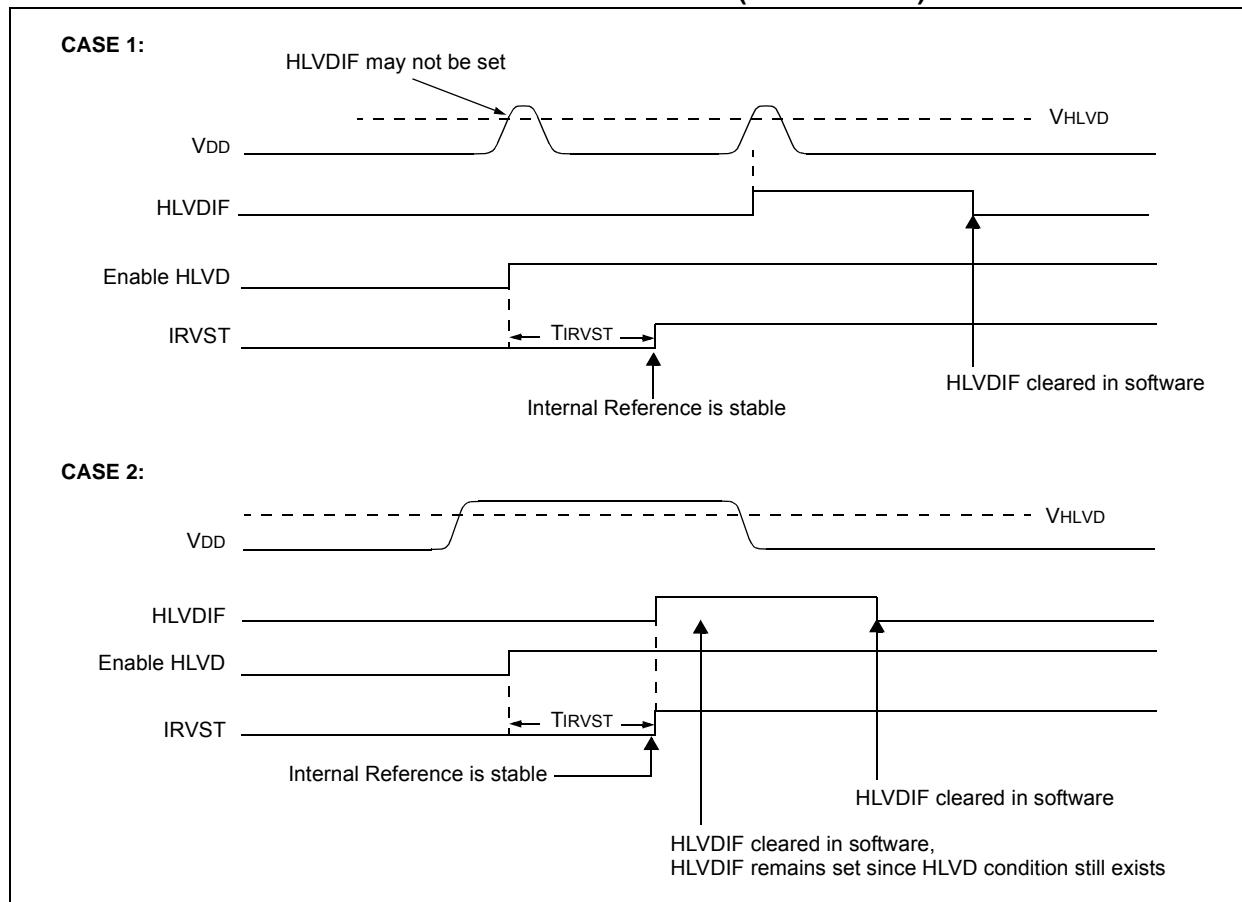
The HLVD interrupt flag is not enabled until TIRVST has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval (see [Figure 25-2](#) or [Figure 25-3](#)).

FIGURE 25-2: LOW-VOLTAGE DETECT OPERATION (VDIRMG = 0)



PIC18(L)F2X/45K50

FIGURE 25-3: HIGH-VOLTAGE DETECT OPERATION (VDIRMAG = 1)

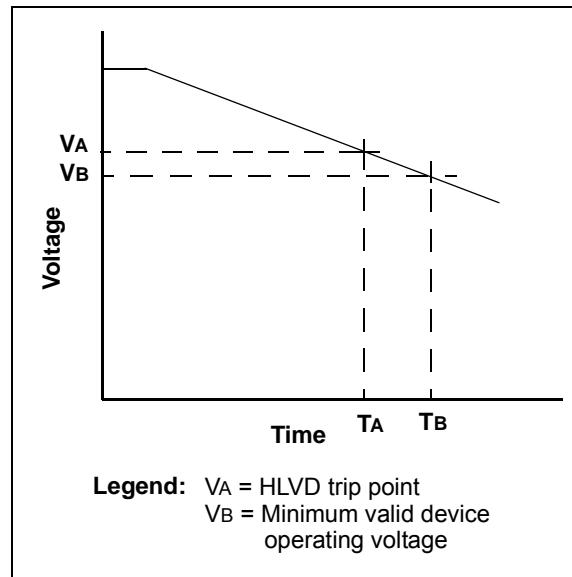


25.6 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. For example, the HLVD module could be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a high-voltage detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 25-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage V_A , the HLVD logic generates an interrupt at time, T_A . The interrupt could cause the execution of an ISR, which would allow the application to perform “house-keeping tasks” and a controlled shutdown before the device voltage exits the valid operating range at T_B . This would give the application a time window, represented by the difference between T_A and T_B , to safely exit.

FIGURE 25-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION



25.7 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the HLVDIF bit will be set and the device will wake-up from Sleep. Device execution will continue from the interrupt vector address if interrupts have been globally enabled.

25.8 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

TABLE 25-1: REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|---------|----------|-----------|--------|--------|------------|--------|--------|--------|------------------|
| HLVDCON | VDIRMAG | BGVST | IRVST | HLVDEN | HLVDL<3:0> | | | | 364 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | IOCIE | TMR0IF | INT0IF | IOCIF | 114 |
| IPR2 | OSCFIP | C1IP | C2IP | EEIP | BCLIP | HLVDIP | TMR3IP | CCP2IP | 124 |
| PIE2 | OSCFIE | C1IE | C2IE | EEIE | BCLIE | HLVDIE | TMR3IE | CCP2IE | 121 |
| PIR2 | OSCFIF | C1IF | C2IF | EEIF | BCLIF | HLVDIF | TMR3IF | CCP2IF | 118 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 149 |

Legend: — = Unimplemented locations, read as '0'. Shaded bits are unused by the HLVD module.

26.0 SPECIAL FEATURES OF THE CPU

PIC18(L)F2X/45K50 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 3.0 “Oscillator Module \(With Fail-Safe Clock Monitor\)”](#).

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)F2X/45K50 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.

26.1 Configuration Bits

The Configuration bits can be programmed (read as ‘0’) or left unprogrammed (read as ‘1’) to select various device configurations. These bits are mapped starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh), which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The WR bit in the EECON1 register starts a self-timed write to the Configuration register. In normal operation mode, a TBLWT instruction with the TBLPTR pointing to the Configuration register sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a TBLWT instruction can write a ‘1’ or a ‘0’ into the cell. For additional details on Flash programming, refer to [Section 7.6 “Writing to Flash Program Memory”](#).

TABLE 26-1: CONFIGURATION BITS AND DEVICE IDs

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Default/ Unprogrammed Value |
|---------|-----------------------|-----------|-------|----------------------|-------------|----------------------|----------------------|--------|-----------|-----------------------------------|
| 300000h | CONFIG1L | — | — | LS48MHZ | CPUDIV<1:0> | — | CFGPLLLEN | PLLSEL | 0000 0000 | |
| 300001h | CONFIG1H | IESO | FCMEN | PCLKEN | — | FOSC<3:0> | | | | 0010 0101 |
| 300002h | CONFIG2L | — | LPBOR | — | BORV<1:0> | — | BOREN<1:0> | PWRTEN | 0101 1111 | |
| 300003h | CONFIG2H | — | — | — | WDTPS<3:0> | — | WDTEN<1:0> | — | 0011 1111 | |
| 300004h | CONFIG3L | — | — | — | — | — | — | — | 0000 0000 | |
| 300005h | CONFIG3H | MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX | 1101 0011 |
| 300006h | CONFIG4L | DEBUG | XINST | ICPRT ⁽⁵⁾ | — | — | LVP ⁽¹⁾ | — | STRVEN | 1010 0101 |
| 300007h | CONFIG4H | — | — | — | — | — | — | — | — | 1111 1111 |
| 300008h | CONFIG5L | — | — | — | — | CP3 ⁽²⁾ | CP2 ⁽²⁾ | CP1 | CP0 | 0000 1111 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — | 1100 0000 |
| 30000Ah | CONFIG6L | — | — | — | — | WRT3 ⁽²⁾ | WRT2 ⁽²⁾ | WRT1 | WRT0 | 0000 1111 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC ⁽³⁾ | — | — | — | — | — | 1110 0000 |
| 30000Ch | CONFIG7L | — | — | — | — | EBTR3 ⁽²⁾ | EBTR2 ⁽²⁾ | EBTR1 | EBTR0 | 0000 1111 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — | 0100 0000 |
| 3FFFFEh | DEVID1 ⁽⁴⁾ | DEV<2:0> | | | REV<4:0> | | | | | qqqqq qqqqq |
| 3FFFFFh | DEVID2 ⁽⁴⁾ | DEV<10:3> | | | | | | | | 0101 1100 |

Legend: — = unimplemented, q = value depends on condition. Shaded bits are unimplemented, read as '0'.

Note 1: Can only be changed when in high voltage programming mode.

2: Available on PIC18(L)F45K50 and PIC18(L)F25K50 devices only.

3: In user mode, this bit is read-only and cannot be self-programmed.

4: See [Register 26-13](#) and [Register 26-14](#) for DEVID values. DEVID registers are read-only and cannot be programmed by the user.

5: Available only on 44-pin TQFP package devices. Program this bit clear on all other devices.

PIC18(L)F2X/45K50

26.2 Register Definitions: Configuration Word

REGISTER 26-1: CONFIG1L: CONFIGURATION REGISTER 1 LOW

| U-0 | U-0 | R/P-0 | R/P-0 | R/P-0 | U-0 | R/P-0 | R/P-0 |
|-------|-----|---------|-------------|-------|-----|----------|--------|
| — | — | LS48MHZ | CPUDIV<1:0> | — | — | CFGPLLEN | PLLSEL |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

x = Bit is unknown

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

bit 5 **LS48MHZ:** USB Low-Speed Clock Selection bit

Selects the clock source for Low-speed USB operation

1 = System clock is expected at 48 MHz, FS/LS USB clock divide-by is set to 8

0 = System clock is expected at 24 MHz, FS/LS USB clock divide-by is set to 4

bit 4-3 **CPUDIV<1:0>:** CPU System Clock Selection bits

11 = CPU system clock divided by 6

10 = CPU system clock divided by 3

01 = CPU system clock divided by 2

00 = No CPU system clock divide

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

bit 1 **CFGPLLEN:** PLL Enable bit⁽¹⁾

1 = Oscillator multiplied by 3 or 4, depending on the PLLSEL bit

0 = Oscillator used directly

bit 0 **PLLSEL:** PLL Multiplier Selection bit

1 = Output frequency is 3x the input frequency

0 = Output frequency is 4x the input frequency

Note 1: See [Table 3-1](#) for conditions under which the CFGPLLEN fuse is available.

REGISTER 26-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH

| R/P-0 | R/P-0 | R/P-1 | U-0 | R/P-0 | R/P-1 | R/P-0 | R/P-1 |
|-------|-------|--------|-----|-----------|-------|-------|-------|
| IESO | FCMEN | PCLKEN | — | FOSC<3:0> | | | |
| 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⁽¹⁾: Internal/External Oscillator Switchover bit 1 = Oscillator Switchover mode enabled 0 = Oscillator Switchover mode disabled |
| bit 6 | FCMEN⁽¹⁾: 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 is always enabled 0 = Primary Clock can be disabled by software |
| bit 4 | Unimplemented: Read as '0' |
| bit 3-0 | FOSC<3:0>: Oscillator Selection bits 1111 = Reserved 1110 = Reserved 1101 = EC oscillator (low power, <4 MHz) 1100 = EC oscillator, CLKO function on OSC2 (low power, <4 MHz) 1011 = EC oscillator (medium power, 4 MHz - 16 MHz) 1010 = EC oscillator, CLKO function on OSC2 (medium power, 4 MHz - 16 MHz) 1001 = Internal oscillator block, CLKO function on OSC2 1000 = Internal oscillator block 0111 = External RC oscillator 0110 = External RC oscillator, CLKO function on OSC2 0101 = EC oscillator (high power, 16 MHz - 48 MHz) 0100 = EC oscillator, CLKO function on OSC2 (high power, 16 MHz - 48 MHz) 0011= HS oscillator (medium power, 4 MHz - 16 MHz) 0010= HS oscillator (high power, 16 MHz - 25 MHz) 0001= XT oscillator 0000= LP oscillator |

Note 1: When FOSC<3:0> is configured for HS, XT, or LP oscillator and FCMEN bit is set, then the IESO bit should also be set to prevent a false failed clock indication and to enable automatic clock switch over from the internal oscillator block to the external oscillator when the OST times out.

PIC18(L)F2X/45K50

REGISTER 26-3: CONFIG2L: CONFIGURATION REGISTER 2 LOW

| U-0 | R/P-1 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|-------|-------|-----|--------------------------|---------------------------|----------------------|-------|-------|
| — | LPBOR | — | BORV<1:0> ⁽¹⁾ | BOREN<1:0> ⁽²⁾ | PWRTE ⁽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 **Unimplemented:** Read as '0'

bit 6 **LPBOR:** Low-Power Brown-out Reset Enable bits

1 = Low-Power Brown-out Reset disabled

0 = Low-Power Brown-out Reset enabled

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

bit 4-3 **BORV<1:0>:** Brown-out Reset Voltage bits⁽¹⁾

11 = VBOR set to 1.9V nominal

10 = VBOR set to 2.2V nominal

01 = VBOR set to 2.5V nominal

00 = VBOR 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 **PWRTE:** Power-up Timer Enable bit⁽²⁾

1 = PWRT disabled

0 = PWRT enabled

Note 1: See [Table 29-1](#) for specifications.

2: The Power-up Timer is decoupled from Brown-out Reset, allowing these features to be independently controlled.

REGISTER 26-4: CONFIG2H: CONFIGURATION REGISTER 2 HIGH

| U-0 | U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|-------|-----|-------|------------|-------|------------|-------|
| — | — | | WDTPS<3:0> | | WDTEN<1:0> | |
| bit 7 | | | | | | bit 0 |

Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

x = Bit is unknown

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

bit 5-2 **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 1-0 **WDTEN<1:0>:** Watchdog Timer Enable bits

11 = WDT enabled in hardware; SWDTEN bit disabled

10 = WDT controlled by the SWDTEN bit

01 = WDT enabled when device is active, disabled when device is in Sleep; SWDTEN bit disabled

00 = WDT disabled in hardware; SWDTEN bit disabled

PIC18(L)F2X/45K50

REGISTER 26-5: CONFIG3H: CONFIGURATION REGISTER 3 HIGH

| R/P-1 | R/P-1 | U-0 | R/P-1 | U-0 | U-0 | R/P-1 | R/P-1 |
|-------|-------|-----|-------|-----|-----|--------|--------|
| MCLRE | SDOMX | — | T3CMX | — | — | PBADEN | CCP2MX |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit P = Programmable bit U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed x = Bit is unknown

- bit 7 **MCLRE:** MCLR Pin Enable bit
1 = MCLR pin enabled; RE3 input pin disabled
0 = RE3 input pin enabled; MCLR disabled
- bit 6 **SDOMX:** SDO Output MUX bit
1 = SDO is on RB3
0 = SDO is on RC7
- bit 5 **Unimplemented:** Read as '0'
- bit 4 **T3CMX:** Timer3 Clock Input MUX bit
1 = T3CKI is on RC0
0 = T3CKI is on RB5
- bit 3-2 **Unimplemented:** Read as '0'
- bit 1 **PBADEN:** PORTB A/D Enable bit
1 = ANSELB<5:0> resets to 1, PORTB<5:0> pins are configured as analog inputs on Reset
0 = ANSELB<5:0> resets to 0, PORTB<4:0> pins are configured as digital I/O on Reset
- bit 0 **CCP2MX:** CCP2 MUX bit
1 = CCP2 input/output is multiplexed with RC1
0 = CCP2 input/output is multiplexed with RB3

REGISTER 26-6: CONFIG4L: CONFIGURATION REGISTER 4 LOW

| R/P-1 | R/P-0 | R/P-1 | U-0 | U-0 | R/P-1 | U-0 | R/P-1 |
|----------------------|-------|----------------------|-----|-----|--------------------|-----|--------|
| DEBUG ⁽²⁾ | XINST | ICPRT ⁽³⁾ | — | — | LVP ⁽¹⁾ | — | 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 **DEBUG:** Background Debugger Enable bit⁽²⁾
 1 = Background debugger disabled, RB6 and RB7 configured as general purpose I/O pins
 0 = Background debugger enabled, RB6 and RB7 are dedicated to In-Circuit Debug
- bit 6 **XINST:** Extended Instruction Set Enable bit
 1 = Instruction set extension and Indexed Addressing mode enabled
 0 = Instruction set extension and Indexed Addressing mode disabled (Legacy mode)
- bit 5 **ICPRT:** Dedicated In-Circuit (ICD) Port Enable bit⁽³⁾
 1 = ICPORt enabled (ICD function on dedicated ICD pins)
 0 = ICPORt disabled (ICD function on default ICD pins, RB6/7)
- bit 4-3 **Unimplemented:** Read as '0'
- 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: Can only be changed by a programmer in high-voltage programming mode.
 - 2: The DEBUG bit is managed automatically by device development tools including debuggers and programmers. For normal device operations, this bit should be maintained as a '1'.
 - 3: Available only on 44-pin TQFP package devices. Program this bit clear on all other devices.

PIC18(L)F2X/45K50

REGISTER 26-7: CONFIG5L: CONFIGURATION REGISTER 5 LOW

| U-0 | U-0 | U-0 | U-0 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|-------|-------|-----|-----|--------------------|--------------------|-------|-------|
| — | — | — | — | CP3 ⁽¹⁾ | CP2 ⁽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 7-4 **Unimplemented:** Read as '0'
bit 3 **CP3:** Code Protection bit⁽¹⁾
 1 = Block 3 not code-protected
 0 = Block 3 code-protected
bit 2 **CP2:** Code Protection bit⁽¹⁾
 1 = Block 2 not code-protected
 0 = Block 2 code-protected
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

Note 1: Available on PIC18(L)F45K50 and PIC18(L)F25K50 devices.

REGISTER 26-8: CONFIG5H: CONFIGURATION REGISTER 5 HIGH

| R/C-1 | R/C-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|-------|-------|-----|-----|-----|-----|-----|-----|
| CPD | CPB | — | — | — | — | — | — |
| bit 7 | bit 0 | | | | | | |

Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

- bit 7 **CPD:** Data EEPROM Code Protection bit
 1 = Data EEPROM not code-protected
 0 = Data EEPROM code-protected
bit 6 **CPB:** Boot Block Code Protection bit
 1 = Boot Block not code-protected
 0 = Boot Block code-protected
bit 5-0 **Unimplemented:** Read as '0'

REGISTER 26-9: CONFIG6L: CONFIGURATION REGISTER 6 LOW

| U-0 | U-0 | U-0 | U-0 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|-------|-----|-----|-----|---------------------|---------------------|-------|-------|
| — | — | — | — | WRT3 ⁽¹⁾ | WRT2 ⁽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 7-4 | Unimplemented: Read as '0' |
| bit 3 | WRT3: Write Protection bit ⁽¹⁾ 1 = Block 3 not write-protected 0 = Block 3 write-protected |
| bit 2 | WRT2: Write Protection bit ⁽¹⁾ 1 = Block 2 not write-protected 0 = Block 2 write-protected |
| bit 1 | WRT1: Write Protection bit 1 = Block 1 not write-protected 0 = Block 1 write-protected |
| bit 0 | WRTO: Write Protection bit 1 = Block 0 not write-protected 0 = Block 0 write-protected |

Note 1: Available on PIC18(L)F45K50 and PIC18(L)F25K50 devices.

PIC18(L)F2X/45K50

REGISTER 26-10: CONFIG6H: CONFIGURATION REGISTER 6 HIGH

| R/C-1 | R/C-1 | R/C-1 | U-0 | U-0 | U-0 | U-0 | U-0 |
|-------|-------|-------------------|-----|-----|-----|-----|-----|
| WRTD | WRTB | WR ⁽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 **WR⁽¹⁾:** 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 ICSP™ mode.

REGISTER 26-11: CONFIG7L: CONFIGURATION REGISTER 7 LOW

| U-0 | U-0 | U-0 | U-0 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|-------|-------|-----|-----|----------------------|----------------------|-------|-------|
| — | — | — | — | EBTR3 ⁽¹⁾ | EBTR2 ⁽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 7-4 **Unimplemented:** Read as '0'

bit 3 **EBTR3:** Table Read Protection bit⁽¹⁾

1 = Block 3 not protected from table reads executed in other blocks

0 = Block 3 protected from table reads executed in other blocks

bit 2 **EBTR2:** Table Read Protection bit⁽¹⁾

1 = Block 2 not protected from table reads executed in other blocks

0 = Block 2 protected from table reads executed in other blocks

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

Note 1: Available on PIC18(L)F45K50 and PIC18(L)F25K50 devices.

REGISTER 26-12: CONFIG7H: CONFIGURATION REGISTER 7 HIGH

| | | | | | | | |
|-------|-------|-----|-----|-----|-----|-----|-------|
| U-0 | R/C-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| — | EBTRB | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

U = Unimplemented bit, read as '0'

-n = Value when device is unprogrammed

C = Clearable only bit

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

bit 6 **EBTRB:** Boot Block Table Read Protection bit

1 = Boot Block not protected from table reads executed in other blocks

0 = Boot Block protected from table reads executed in other blocks

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

REGISTER 26-13: DEVID1: DEVICE ID REGISTER 1

| 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

These bits, together with DEV<10:3> in DEVID2, determine the device ID.

See [Table 26-2](#) for complete Device ID list.

bit 4-0 **REV<4:0>:** Revision ID bits

These bits indicate the device revision.

REGISTER 26-14: DEVID2: DEVICE ID REGISTER 2

| 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, together with DEV<2:0> in DEVID1, determine the device ID.

See [Table 26-2](#) for complete Device ID list.

PIC18(L)F2X/45K50

TABLE 26-2: DEVICE ID TABLE FOR THE PIC18(L)F2X/45K50 FAMILY

| DEV<10:3> | DEV<2:0> | Part Number |
|-----------|----------|--------------|
| 0101 1100 | 000 | PIC18F45K50 |
| | 001 | PIC18F25K50 |
| | 011 | PIC18F24K50 |
| | 100 | PIC18LF45K50 |
| | 101 | PIC18LF25K50 |
| | 111 | PIC18LF24K50 |

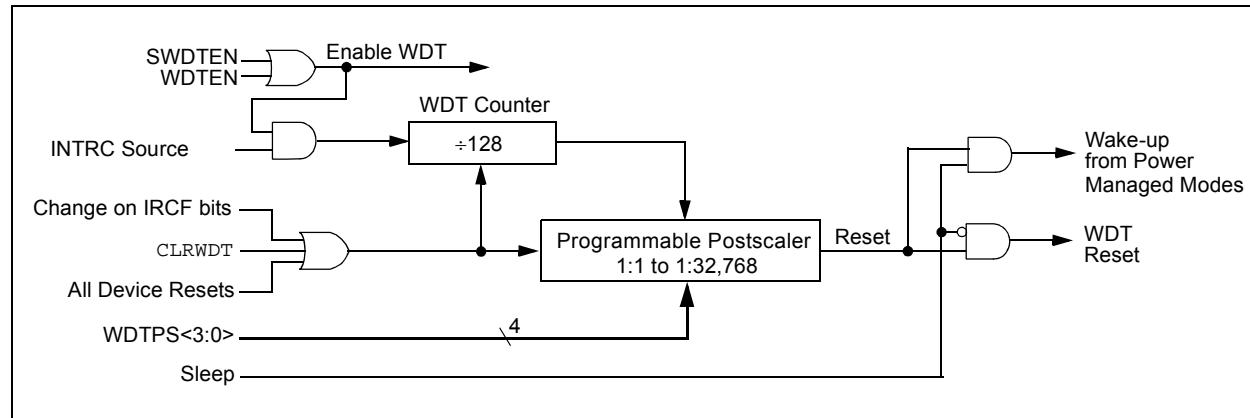
26.3 Watchdog Timer (WDT)

For PIC18(L)F2X/45K50 devices, the WDT is driven by the INTRC source. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by bits in Configuration Register 2H. Available periods range from 4 ms to 131.072 seconds (2.18 minutes). The WDT and postscaler are cleared when any of the following events occur: a SLEEP or CLRWDAT instruction is executed, the IRCF bits of the OSCCON register are changed or a clock failure has occurred.

- Note 1:** The CLRWDAT and SLEEP instructions clear the WDT and postscaler counts when executed.
- 2:** Changing the setting of the IRCF bits of the OSCCON register clears the WDT and postscaler counts.
- 3:** When a CLRWDAT instruction is executed, the postscaler count will be cleared.

FIGURE 26-1: WDT BLOCK DIAGRAM



PIC18(L)F2X/45K50

26.3.1 CONTROL REGISTER

Register 26-15 shows the WDTCON register. This is a readable and writable register which contains a control bit that allows software to override the WDT enable Configuration bit, but only if the Configuration bit has disabled the WDT.

26.4 Register Definitions: WDT Control

REGISTER 26-15: WDTCON: WATCHDOG TIMER CONTROL REGISTER

| U-0 | R/W-0 |
|-----|-----|-----|-----|-----|-----|-----|--------------------------------|
| — | — | — | — | — | — | — | SWDTEN ⁽¹⁾ bit 0 |

Legend:

R = Readable bit
-n = Value at POR

W = Writable bit
'1' = Bit is set

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

bit 7-1 **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 unless the Configuration bit, WDTEN<1:0>, is set to 10b (SWDTEN enabled).

TABLE 26-3: REGISTERS ASSOCIATED WITH WATCHDOG TIMER

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|--------|-------|--------|-------|-------|-------|-------|-------|--------|------------------|
| RCON | IPEN | SBOREN | — | RI | TO | PD | POR | BOR | 64 |
| WDTCON | — | — | — | — | — | — | — | SWDTEN | 384 |

Legend: — = unimplemented, read as '0'. Shaded bits are not used by the Watchdog Timer.

TABLE 26-4: CONFIGURATION REGISTERS ASSOCIATED WITH WATCHDOG TIMER

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|-------|------------|-------|-------|-------|-------|------------|------------------|
| CONFIG2H | — | — | WDTPS<3:0> | | | | | WDTEN<1:0> | 375 |

Legend: — = unimplemented, read as '0'. Shaded bits are not used by the Watchdog Timer.

26.5 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 three or five blocks, depending on the device. 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 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 26-2](#) shows the program memory organization for 16 and 32-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in [Table 26-5](#).

FIGURE 26-2: CODE-PROTECTED PROGRAM MEMORY FOR PIC18(L)F2X/45K50

| MEMORY SIZE/DEVICE | | Block Code Protection Controlled By: |
|--|-------------------------------|---|
| 16 Kbytes (PIC18(L)F24K50) | 32 Kbytes (PIC18(L)FX5K50) | |
| Boot Block (000h-7FFh) | Boot Block (000h-7FFh) | CPB, WRTB, EBTRB |
| Block 0 (800h-1FFFh) | Block 0 (800h-1FFFh) | CP0, WRT0, EBTR0 |
| Block 1 (2000h-3FFFh) | Block 1 (2000h-3FFFh) | CP1, WRT1, EBTR1 |
| Unimplemented Read '0's (4000h-1FFFFh) | Block 2 (4000h-5FFFh) | CP2, WRT2, EBTR2 |
| | Block 3 (6000h-7FFFh) | CP3, WRT3, EBTR3 |
| Unimplemented Read '0's (8000h-1FFFFh) | | (Unimplemented Memory Space) |

TABLE 26-5: CONFIGURATION REGISTERS ASSOCIATED WITH CODE PROTECTION

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-----------|----------|-------|-------|---------------------|----------------------|----------------------|-------|-------|
| 300008h | CONFIG5L | — | — | — | CP3 ⁽¹⁾ | CP2 ⁽¹⁾ | CP1 | CP0 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — |
| 30000Ah | CONFIG6L | — | — | — | WRT3 ⁽¹⁾ | WRT2 ⁽¹⁾ | WRT1 | WRT0 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC ⁽²⁾ | — | — | — | — |
| 30000Ch | CONFIG7L | — | — | — | EBTR3 ⁽¹⁾ | EBTR2 ⁽¹⁾ | EBTR1 | EBTR0 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — |

Legend: Shaded bits are unimplemented.

Note 1: Available on PIC18(L)F45K50 and PIC18(L)F25K50 devices only.

2: In User mode, this bit is read-only and cannot be self-programmed.

26.5.1 PROGRAM MEMORY CODE PROTECTION

The program memory may be read to or written from any location using the table read and table write instructions. The device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In normal execution mode, the 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 26-3 through 26-5 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 26-3: TABLE WRITE (WRTn) DISALLOWED

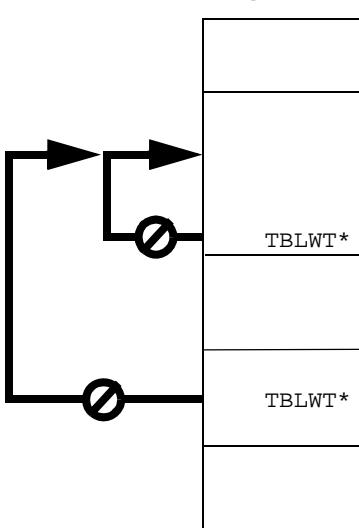
| Register Values | Program Memory | Configuration Bit Settings |
|--|--|--|
| TBLPTR = 0008FFh |  | 000000h 0007FFh 000800h WRTB, EBTRB = 11 |
| PC = 001FFEh | 001FFFFh 002000h 003FFFFh 004000h 005FFFFh 006000h 007FFFFh | WRT0, EBTR0 = 01 WRT1, EBTR1 = 11 WRT2, EBTR2 = 11 WRT3, EBTR3 = 11 |
| PC = 005FFEh | | |
| Results: All table writes disabled to Blockn whenever WRTn = 0. | | |

FIGURE 26-4: EXTERNAL BLOCK TABLE READ (EBTRn) DISALLOWED

| Register Values | Program Memory | Configuration Bit Settings |
|------------------|-------------------------------|----------------------------|
| TBLPTR = 0008FFh | 000000h 0007FFh 000800h | WRTB, EBTRB = 11 |
| PC = 003FFEh | 001FFFh 002000h TBLRD* | WRT0, EBTR0 = 10 |
| | 003FFFh 004000h | WRT1, EBTR1 = 11 |
| | 005FFFh 006000h | WRT2, EBTR2 = 11 |
| | 007FFFh | WRT3, EBTR3 = 11 |

Results: All table reads from external blocks to Blockn are disabled whenever EBTRn = 0.
TABLAT register returns a value of '0'.

FIGURE 26-5: INTERNAL BLOCK TABLE READ (EBTRn) ALLOWED

| Register Values | Program Memory | Configuration Bit Settings |
|------------------|-------------------------------|----------------------------|
| TBLPTR = 0008FFh | 000000h 0007FFh 000800h | WRTB, EBTRB = 11 |
| PC = 001FFEh | 001FFFh 002000h TBLRD* | WRT0, EBTR0 = 10 |
| | 003FFFh 004000h | WRT1, EBTR1 = 11 |
| | 005FFFh 006000h | WRT2, EBTR2 = 11 |
| | 007FFFh | WRT3, EBTR3 = 11 |

Results: Table reads permitted within Blockn, even when EBTRBn = 0.
TABLAT register returns the value of the data at the location TBLPTR.

26.5.2 DATA EEPROM CODE PROTECTION

The entire data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of data EEPROM. WRTD inhibits internal and external writes to data EEPROM. The CPU can always read data EEPROM under normal operation, regardless of the protection bit settings.

26.5.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In normal execution mode, the WRTC bit is readable only. WRTC can only be written via ICSP or an external programmer.

26.6 ID Locations

Eight memory locations (200000h-200007h) are designated as ID locations, where the user can store checksum or other code identification numbers. These locations are both readable and writable during normal execution through the TBLRD and TBLWT instructions or during program/verify. The ID locations can be read when the device is code-protected.

26.7 In-Circuit Serial Programming

PIC18(L)F2X/45K50 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.

26.8 In-Circuit Debugger

When the DEBUG Configuration bit is programmed to a '0', the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some resources are not available for general use. [Table 26-6](#) shows which resources are required by the background debugger.

TABLE 26-6: DEBUGGER RESOURCES

| | |
|-----------|----------|
| I/O pins: | RB6, RB7 |
|-----------|----------|

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to the following pins:

- MCLR/VPP/RE3
- VDD
- Vss
- RB7
- RB6

This will interface to the In-Circuit Debugger module available from Microchip or one of the third party development tool companies.

26.9 Special ICPORT Features (44-Pin TQFP Package Only)

Under specific circumstances, the No Connect (NC) pins of devices in 44-pin TQFP packages can provide additional functionality. These features are controlled by device Configuration bits and are available only in this package type and pin count.

26.9.1 DEDICATED ICD/ICSP PORT

The 44-pin TQFP devices can use NC pins to provide an alternate port for In-Circuit Debugging (ICD) and In-Circuit Serial Programming (ICSP). These pins are collectively known as the dedicated ICSP/ICD port, since they are not shared with any other function of the device.

When implemented, the dedicated port activates three NC pins to provide an alternate device Reset, data and clock ports. None of these ports overlap with standard I/O pins, making the I/O pins available to the user's application. The port functions the same way as the legacy ICSP/ICD/MCLR pins on RB6/RB7/MCLR and they have the same electrical specifications as their respective pins.

[Table 26-7](#) identifies the functionally equivalent pins for ICSP and ICD purposes. The dedicated ICSP/ICD port is enabled by setting the ICPRT Configuration bit. This bit is set by default on 44-pin TQFP devices and can only be set and cleared when using the MCLR/RB6/RB7 ICSP interface.

When ICPRT is set, several things must be taken into consideration. First and foremost, the ICRST pin functions as an additional MCLR pin and must be pulled high to keep the part out of Reset. Second, while the MCLR/RB6/RB7 pins can still be used to program the part, the dedicated ICPORT pins must be used to debug the part. Finally, the MCLRE bit still works as normal, meaning that if ICPRT = 1 and MCLRE = 0, MCLR will function as a general purpose RE3 input, with the ICRST providing the normal MCLR Reset functions.

TABLE 26-7: EQUIVALENT PINS FOR LEGACY AND DEDICATED ICD/ICSP™ PORTS

| Pin Name | | Pin Type | Pin Function |
|---------------|----------------|----------|-------------------------------------|
| Legacy Port | Dedicated Port | | |
| MCLR/VPP/RE3 | NC/ICRST/ICVPP | P | Device Reset and Programming Enable |
| RB6/IOCB6/PGC | NC/ICCK/ICPGC | I | Serial Clock |
| RB7/IOCB7/PGD | NC/ICDT/ICPGD | I/O | Serial Data |

Legend: I = Input, O = Output, P = Power

26.10 Single-Supply ICSP Programming

The LVP Configuration bit enables Single-Supply ICSP Programming (formerly known as Low-Voltage ICSP Programming or LVP). When Single-Supply Programming is enabled, the microcontroller can be programmed without requiring high voltage being applied to the MCLR/VPP/RE3 pin. See “*PIC18(L)F2X/45K50 Flash Memory Programming Specification*” (DS41630) for more details about low voltage programming.

- Note 1:** High-voltage programming is always available, regardless of the state of the LVP bit, by applying VIHH to the MCLR pin.
- 2:** By default, Single-Supply ICSP™ is enabled in unprogrammed devices (as supplied from Microchip) and erased devices.
- 3:** While in Low-Voltage ICSP™ mode, MCLR is always enabled, regardless of the MCLRE bit, and the RE3 pin can no longer be used as a general purpose input.

The LVP bit may be set or cleared only when using standard high-voltage programming (VIHH applied to the MCLR/VPP/RE3 pin). Once LVP has been disabled, only the standard high-voltage programming is available and must be used to program the device.

Memory that is not code-protected can be erased using either a block erase, or erased row by row, then written at any specified VDD. If code-protected memory is to be erased, a block erase is required.

27.0 INSTRUCTION SET SUMMARY

PIC18(L)F2X/45K50 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.

27.1 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 27-2](#) lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. [Table 27-1](#) shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The destination of the result (specified by 'd')
3. The accessed memory (specified by 'a')

The file register designator 'f' specifies which file register is to be used by the instruction. The destination designator 'd' specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the WREG register. If 'd' is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The bit in the file register (specified by 'b')
3. The accessed memory (specified by 'a')

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by 'k')
- The desired FSR register to load the literal value into (specified by 'f')
- No operand required (specified by '—')

The **control** instructions may use some of the following operands:

- A program memory address (specified by 'n')
- The mode of the CALL or RETURN instructions (specified by 's')
- The mode of the table read and table write instructions (specified by 'm')
- No operand required (specified by '—')

All instructions are a single word, except for four double-word instructions. These instructions were made double-word to contain the required information in 32 bits. In the second word, the 4 MSbs are '1's. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μ s. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 μ s. Two-word branch instructions (if true) would take 3 μ s.

[Figure 27-1](#) 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 27-2](#), lists the standard instructions recognized by the Microchip Assembler (MPASM™).

[Section 27.1.1 “Standard Instruction Set”](#) provides a description of each instruction.

TABLE 27-1: OPCODE FIELD DESCRIPTIONS

| Field | Description |
|-----------------|---|
| a | RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register |
| bbb | Bit address within an 8-bit file register (0 to 7). |
| BSR | Bank Select Register. Used to select the current RAM bank. |
| C, DC, Z, OV, N | ALU Status bits: C arry, D igit C arry, Z ero, O verflow, N egative. |
| d | Destination select bit d = 0: store result in WREG d = 1: store result in file register f |
| dest | Destination: either the WREG register or the specified register file location. |
| f | 8-bit Register file address (00h to FFh) or 2-bit FSR designator (0h to 3h). |
| f _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 | The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions: * No change to register (such as TBLPTR with table reads and writes) *+ Post-Increment register (such as TBLPTR with table reads and writes) *- Post-Decrement register (such as TBLPTR with table reads and writes) +* Pre-Increment register (such as TBLPTR with table reads and writes) |
| n | The relative address (2's complement number) for relative branch instructions or the direct address for CALL/BRANCH and RETURN instructions. |
| PC | Program Counter. |
| PCL | Program Counter Low Byte. |
| PCH | Program Counter High Byte. |
| PCLATH | Program Counter High Byte Latch. |
| PCLATU | Program Counter Upper Byte Latch. |
| PD | Power-down bit. |
| PRODH | Product of Multiply High Byte. |
| PRODL | Product of Multiply Low Byte. |
| s | Fast Call/Return mode select bit s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode) |
| TBLPTR | 21-bit Table Pointer (points to a Program Memory location). |
| TABLAT | 8-bit Table Latch. |
| TO | Time-out bit. |
| TOS | Top-of-Stack. |
| u | Unused or unchanged. |
| WDT | Watchdog Timer. |
| WREG | Working register (accumulator). |
| x | Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools. |
| z _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). |

PIC18(L)F2X/45K50

FIGURE 27-1: GENERAL FORMAT FOR INSTRUCTIONS

| Byte-oriented file register operations | Example Instruction | | | | | | | | | | | | | | | | | |
|---|------------------------|------------------|------------|---|--------|-------------------|------------------|--------|------------|------------|------------|----|-------------------|------------------------|-------------------|--|----------------------|-------------|
| <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px;">15</td><td style="width: 10px;">10</td><td style="width: 9px;">9</td><td style="width: 8px;">8</td><td style="width: 7px;">7</td><td style="width: 0px;">0</td></tr> <tr> <td>OPCODE</td><td>d</td><td>a</td><td colspan="3">f (FILE #)</td></tr> </table> <p> d = 0 for result destination to be WREG register d = 1 for result destination to be file register (f) a = 0 to force Access Bank a = 1 for BSR to select bank f = 8-bit file register address </p> | 15 | 10 | 9 | 8 | 7 | 0 | OPCODE | d | a | f (FILE #) | | | ADDWF MYREG, W, B | | | | | |
| 15 | 10 | 9 | 8 | 7 | 0 | | | | | | | | | | | | | |
| OPCODE | d | a | f (FILE #) | | | | | | | | | | | | | | | |
| Byte to Byte move operations (2-word) | | | | | | | | | | | | | | | | | | |
| <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px;">15</td><td style="width: 12px;">12</td><td style="width: 11px;">11</td><td style="width: 0px;">0</td></tr> <tr> <td>OPCODE</td><td colspan="3">f (Source FILE #)</td></tr> <tr> <td style="width: 15px;">15</td><td style="width: 12px;">12</td><td style="width: 11px;">11</td><td style="width: 0px;">0</td></tr> <tr> <td>1111</td><td colspan="3">f (Destination FILE #)</td></tr> </table> <p>f = 12-bit file register address</p> | 15 | 12 | 11 | 0 | OPCODE | f (Source FILE #) | | | 15 | 12 | 11 | 0 | 1111 | f (Destination FILE #) | | | MOVFF MYREG1, MYREG2 | |
| 15 | 12 | 11 | 0 | | | | | | | | | | | | | | | |
| OPCODE | f (Source FILE #) | | | | | | | | | | | | | | | | | |
| 15 | 12 | 11 | 0 | | | | | | | | | | | | | | | |
| 1111 | f (Destination FILE #) | | | | | | | | | | | | | | | | | |
| Bit-oriented file register operations | | | | | | | | | | | | | | | | | | |
| <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px;">15</td><td style="width: 12px;">12</td><td style="width: 11px;">11</td><td style="width: 9px;">9</td><td style="width: 8px;">8</td><td style="width: 7px;">7</td><td style="width: 0px;">0</td></tr> <tr> <td>OPCODE</td><td>b (BIT #)</td><td>a</td><td colspan="3">f (FILE #)</td></tr> </table> <p> b = 3-bit position of bit in file register (f) a = 0 to force Access Bank a = 1 for BSR to select bank f = 8-bit file register address </p> | 15 | 12 | 11 | 9 | 8 | 7 | 0 | OPCODE | b (BIT #) | a | f (FILE #) | | | BSF MYREG, bit, B | | | | |
| 15 | 12 | 11 | 9 | 8 | 7 | 0 | | | | | | | | | | | | |
| OPCODE | b (BIT #) | a | f (FILE #) | | | | | | | | | | | | | | | |
| Literal operations | | | | | | | | | | | | | | | | | | |
| <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px;">15</td><td style="width: 8px;">8</td><td style="width: 7px;">7</td><td style="width: 0px;">0</td></tr> <tr> <td>OPCODE</td><td colspan="3">k (literal)</td></tr> </table> <p>k = 8-bit immediate value</p> | 15 | 8 | 7 | 0 | OPCODE | k (literal) | | | MOVLW 7Fh | | | | | | | | | |
| 15 | 8 | 7 | 0 | | | | | | | | | | | | | | | |
| OPCODE | k (literal) | | | | | | | | | | | | | | | | | |
| Control operations | | | | | | | | | | | | | | | | | | |
| CALL, GOTO and Branch operations | | | | | | | | | | | | | | | | | | |
| <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px;">15</td><td style="width: 8px;">8</td><td style="width: 7px;">7</td><td style="width: 0px;">0</td></tr> <tr> <td>OPCODE</td><td colspan="3">n<7:0> (literal)</td></tr> <tr> <td style="width: 15px;">15</td><td style="width: 12px;">12</td><td style="width: 11px;">11</td><td style="width: 0px;">0</td></tr> <tr> <td>1111</td><td colspan="3">n<19:8> (literal)</td></tr> </table> <p>n = 20-bit immediate value</p> | 15 | 8 | 7 | 0 | OPCODE | n<7:0> (literal) | | | 15 | 12 | 11 | 0 | 1111 | n<19:8> (literal) | | | GOTO Label | |
| 15 | 8 | 7 | 0 | | | | | | | | | | | | | | | |
| OPCODE | n<7:0> (literal) | | | | | | | | | | | | | | | | | |
| 15 | 12 | 11 | 0 | | | | | | | | | | | | | | | |
| 1111 | n<19:8> (literal) | | | | | | | | | | | | | | | | | |
| <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px;">15</td><td style="width: 8px;">8</td><td style="width: 7px;">7</td><td style="width: 0px;">0</td></tr> <tr> <td>OPCODE</td><td>S</td><td colspan="3">n<7:0> (literal)</td></tr> <tr> <td style="width: 15px;">15</td><td style="width: 12px;">12</td><td style="width: 11px;">11</td><td style="width: 0px;">0</td></tr> <tr> <td>1111</td><td colspan="3">n<19:8> (literal)</td></tr> </table> <p>S = Fast bit</p> | 15 | 8 | 7 | 0 | OPCODE | S | n<7:0> (literal) | | | 15 | 12 | 11 | 0 | 1111 | n<19:8> (literal) | | | CALL MYFUNC |
| 15 | 8 | 7 | 0 | | | | | | | | | | | | | | | |
| OPCODE | S | n<7:0> (literal) | | | | | | | | | | | | | | | | |
| 15 | 12 | 11 | 0 | | | | | | | | | | | | | | | |
| 1111 | n<19:8> (literal) | | | | | | | | | | | | | | | | | |
| <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px;">15</td><td style="width: 11px;">11</td><td style="width: 10px;">10</td><td style="width: 0px;">0</td></tr> <tr> <td>OPCODE</td><td colspan="3">n<10:0> (literal)</td></tr> </table> | 15 | 11 | 10 | 0 | OPCODE | n<10:0> (literal) | | | BRA MYFUNC | | | | | | | | | |
| 15 | 11 | 10 | 0 | | | | | | | | | | | | | | | |
| OPCODE | n<10:0> (literal) | | | | | | | | | | | | | | | | | |
| <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td style="width: 15px;">15</td><td style="width: 8px;">8</td><td style="width: 7px;">7</td><td style="width: 0px;">0</td></tr> <tr> <td>OPCODE</td><td colspan="3">n<7:0> (literal)</td></tr> </table> | 15 | 8 | 7 | 0 | OPCODE | n<7:0> (literal) | | | BC MYFUNC | | | | | | | | | |
| 15 | 8 | 7 | 0 | | | | | | | | | | | | | | | |
| OPCODE | n<7:0> (literal) | | | | | | | | | | | | | | | | | |

TABLE 27-2: PIC18 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 f _d (destination) 2nd word | 2 | 1100 | ffff | ffff | ffff | None | | |
| | | | 1111 | ffff | ffff | ffff | | | |
| MOVWF f, a | Move WREG to f | 1 | 0110 | 111a | ffff | ffff | None | | |
| MULWF f, a | Multiply WREG with f | 1 | 0000 | 001a | ffff | ffff | None | 1, 2 | |
| NEGF f, a | Negate f | 1 | 0110 | 110a | ffff | ffff | C, DC, Z, OV, N | | |
| RLCF f, d, a | Rotate Left f through Carry | 1 | 0011 | 01da | ffff | ffff | C, Z, N | 1, 2 | |
| RLNCF f, d, a | Rotate Left f (No Carry) | 1 | 0100 | 01da | ffff | ffff | Z, N | | |
| RRCF f, d, a | Rotate Right f through Carry | 1 | 0011 | 00da | ffff | ffff | C, Z, N | | |
| RRNCF f, d, a | Rotate Right f (No Carry) | 1 | 0100 | 00da | ffff | ffff | Z, N | | |
| SETf f, a | Set f | 1 | 0110 | 100a | ffff | ffff | None | 1, 2 | |
| SUBFWB f, d, a | Subtract f from WREG with borrow | 1 | 0101 | 01da | ffff | ffff | C, DC, Z, OV, N | | |
| SUBWF f, d, a | Subtract WREG from f | 1 | 0101 | 11da | ffff | ffff | C, DC, Z, OV, N | 1, 2 | |
| SUBWFB f, d, a | Subtract WREG from f with borrow | 1 | 0101 | 10da | ffff | ffff | C, DC, Z, OV, N | | |
| SWAPf f, d, a | Swap nibbles in f | 1 | 0011 | 10da | ffff | ffff | None | 4 | |
| TSTFSZ f, a | Test f, skip if 0 | 1 (2 or 3) | 0110 | 011a | ffff | ffff | None | 1, 2 | |
| XORWF f, d, a | Exclusive OR WREG with f | 1 | 0001 | 10da | ffff | ffff | Z, N | | |

- Note 1:** When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

PIC18(L)F2X/45K50

TABLE 27-2: PIC18 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 2nd word | 2 | 1110 | 110s | kkkk | kkkk | None | |
| CLRWD | Clear Watchdog Timer | 1 | 0000 | 0000 | 0000 | 0100 | TO, PD | |
| DAW — | Decimal Adjust WREG | 1 | 0000 | 0000 | 0000 | 0111 | C | |
| GOTO k | Go to address 1st word 2nd word | 2 | 1110 | 1111 | kkkk | kkkk | None | |
| NOP — | No Operation | 1 | 0000 | 0000 | 0000 | 0000 | None | |
| NOP — | No Operation | 1 | 1111 | xxxx | xxxx | xxxx | None | 4 |
| POP — | Pop top of return stack (TOS) | 1 | 0000 | 0000 | 0000 | 0110 | None | |
| PUSH — | Push top of return stack (TOS) | 1 | 0000 | 0000 | 0000 | 0101 | None | |
| RCALL n | Relative Call | 2 | 1101 | 1nnn | nnnn | nnnn | None | |
| RESET | Software device Reset | 1 | 0000 | 0000 | 1111 | 1111 | All | |
| RETFIE s | Return from interrupt enable | 2 | 0000 | 0000 | 0001 | 000s | GIE/GIEH, PEIE/GIEL | |
| RETLW k | Return with literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| RETURN s | Return from Subroutine | 2 | 0000 | 0000 | 0001 | 001s | None | |
| SLEEP — | Go into Standby mode | 1 | 0000 | 0000 | 0000 | 0011 | TO, PD | |

- Note 1:** When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.
- 4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

TABLE 27-2: PIC18 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 to FSR(f) 1st word | 2 | 1110 | 1110 | 00ff | kkkk | None | |
| MOVLB | k | Move literal to BSR<3:0> | 1 | 0000 | 0001 | 0000 | kkkk | None | |
| MOVLW | k | Move literal to WREG | 1 | 0000 | 1110 | kkkk | kkkk | None | |
| MULLW | k | Multiply literal with WREG | 1 | 0000 | 1101 | kkkk | kkkk | None | |
| RETLW | k | Return with literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| SUBLW | k | Subtract WREG from literal | 1 | 0000 | 1000 | kkkk | kkkk | C, DC, Z, OV, N | |
| XORLW | k | Exclusive OR literal with WREG | 1 | 0000 | 1010 | kkkk | kkkk | Z, N | |
| 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.

PIC18(L)F2X/45K50

27.1.1 STANDARD INSTRUCTION SET

| ADDLW | ADD literal to W | ADDWF | ADD W to f |
|--|---|--|--|
| Syntax: | ADDLW k | Syntax: | ADDWF f {,d {,a}} |
| Operands: | 0 ≤ k ≤ 255 | Operands: | 0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] |
| Operation: | (W) + k → W | Operation: | (W) + (f) → dest |
| Status Affected: | N, OV, C, DC, Z | Status Affected: | N, OV, C, DC, Z |
| Encoding: | 0000 1111 kkkk kkkk | Encoding: | 0010 01da ffff ffff |
| Description: | The contents of W are added to the 8-bit literal 'k' and the result is placed in W. | Description: | Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details. |
| Words: | 1 | Words: | 1 |
| Cycles: | 1 | Cycles: | 1 |
| Q Cycle Activity: | | Q Cycle Activity: | |
| Q1 Q2 Q3 Q4 | | Q1 Q2 Q3 Q4 | |
| Decode | Read literal 'k' | Process Data | Write to W |

Example: ADDLW 15h

Before Instruction

W = 10h

After Instruction

W = 25h

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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] | | | | | | | | |
| Operation: | (W) + (f) + (C) → dest | | | | | | | | |
| Status Affected: | N,OV, C, DC, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0010</td> <td>00da</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0010 | 00da | ffff | ffff | | | | |
| 0010 | 00da | ffff | ffff | | | | | | |
| Description: | Add W, the CARRY flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

Example: 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 ≤ k ≤ 255 | | | | | | | | |
| Operation: | (W) .AND. k → W | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>1011</td> <td>kkkk</td> <td>kkkk</td> </tr> </table> | 0000 | 1011 | kkkk | kkkk | | | | |
| 0000 | 1011 | kkkk | kkkk | | | | | | |
| Description: | The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'k'</td> <td>Process Data</td> <td>Write to W</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | Write to W |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read literal 'k' | Process Data | Write to W | | | | | | |

Example: ANDLW 05Fh

Before Instruction

W = A3h

After Instruction

W = 03h

PIC18(L)F2X/45K50

| ANDWF | AND W with f | | | | | | | | |
|-------------------|--|--------------|----------------------|------|------|--------|-------------------|--------------|----------------------|
| Syntax: | ANDWF f {,d {,a}} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $(W) .AND. (f) \rightarrow \text{dest}$ | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0001</td> <td>01da</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0001 | 01da | ffff | ffff | | | | |
| 0001 | 01da | ffff | ffff | | | | | | |
| Description: | <p>The contents of W are AND'ed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

Example: ANDWF REG, 0, 0

Before Instruction

W = 17h
REG = C2h

After Instruction

W = 02h
REG = C2h

| BC | Branch if Carry | | | | | | | | | | | | |
|-------------------|--|--------------|--------------|------|------|--------|------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Syntax: | BC n | | | | | | | | | | | | |
| Operands: | $-128 \leq n \leq 127$ | | | | | | | | | | | | |
| Operation: | if CARRY bit is '1' $(PC) + 2 + 2n \rightarrow PC$ | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1110</td> <td>0010</td> <td>nnnn</td> <td>nnnn</td> </tr> </table> | 1110 | 0010 | nnnn | nnnn | | | | | | | | |
| 1110 | 0010 | nnnn | nnnn | | | | | | | | | | |
| Description: | <p>If the CARRY bit is '1', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC + 2 + 2n$. This instruction is then a two-cycle instruction.</p> | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>Write to PC</td> </tr> <tr> <td>No operation</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | Write to PC | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | Write to PC | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |
| If No Jump: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>No operation</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | No operation | | | | | | | | | | |

Example: HERE 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: | <code>BCF f, b {,a}</code> | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $0 \leq b \leq 7$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $0 \rightarrow f$ | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1001</td><td>bbba</td><td>ffff</td><td>ffff</td></tr> </table> | 1001 | bbba | ffff | ffff | | | | |
| 1001 | bbba | ffff | ffff | | | | | | |
| Description: | <p>Bit 'b' in register 'f' is cleared.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | |
| | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write register 'f'</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write register 'f' |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write register 'f' | | | | | | |

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
`FLAG_REG = C7h`

After Instruction
`FLAG_REG = 47h`

| BN | Branch if Negative | | | | | | | | | | | | |
|-------------------|--|--------------|--------------|------|------|--------|------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Syntax: | <code>BN n</code> | | | | | | | | | | | | |
| Operands: | $-128 \leq n \leq 127$ | | | | | | | | | | | | |
| Operation: | if NEGATIVE bit is '1' $(PC) + 2 + 2n \rightarrow PC$ | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0110</td><td>nnnn</td><td>nnnn</td></tr> </table> | 1110 | 0110 | nnnn | nnnn | | | | | | | | |
| 1110 | 0110 | nnnn | nnnn | | | | | | | | | | |
| Description: | <p>If the NEGATIVE bit is '1', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC + 2 + 2n$. This instruction is then a two-cycle instruction.</p> | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | |
| If Jump: | | | | | | | | | | | | | |
| | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | Write to PC | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | Write to PC | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |
| If No Jump: | | | | | | | | | | | | | |
| | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | No operation | | | | | | | | | | |

Example: `HERE BN Jump`

Before Instruction
`PC = address (HERE)`

After Instruction
`If NEGATIVE = 1;
 PC = address (Jump)
 If NEGATIVE = 0;
 PC = address (HERE + 2)`

PIC18(L)F2X/45K50

| BNC | Branch if Not Carry | | | | | | | | | | | | | | | |
|-------------------|--|--------------|--------------|--|------|------|------|------|--------|------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Syntax: | BNC n | | | | | | | | | | | | | | | |
| Operands: | $-128 \leq n \leq 127$ | | | | | | | | | | | | | | | |
| Operation: | if CARRY bit is '0' (PC) + 2 + 2n → PC | | | | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0011</td><td>nnnn</td><td>nnnn</td></tr> </table> | | | | 1110 | 0011 | nnnn | nnnn | | | | | | | | |
| 1110 | 0011 | nnnn | nnnn | | | | | | | | | | | | | |
| Description: | <p>If the CARRY bit is '0', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.</p> | | | | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | | | | |
| If Jump: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | Write to PC | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | Write to PC | | | | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | | | | |
| If No Jump: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | No operation | | | | | | | | | | | | | |

Example: HERE BNC Jump

Before Instruction
 PC = address (HERE)

After Instruction
 If CARRY = 0;
 PC = address (Jump)
 If CARRY = 1;
 PC = address (HERE + 2)

| BNN | Branch if Not Negative | | | | | | | | | | | | | | | |
|-------------------|--|--------------|--------------|--|------|------|------|------|--------|------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Syntax: | BNN n | | | | | | | | | | | | | | | |
| Operands: | $-128 \leq n \leq 127$ | | | | | | | | | | | | | | | |
| Operation: | if NEGATIVE bit is '0' (PC) + 2 + 2n → PC | | | | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0111</td><td>nnnn</td><td>nnnn</td></tr> </table> | | | | 1110 | 0111 | nnnn | nnnn | | | | | | | | |
| 1110 | 0111 | nnnn | nnnn | | | | | | | | | | | | | |
| Description: | <p>If the NEGATIVE bit is '0', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.</p> | | | | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | | | | |
| If Jump: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | Write to PC | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | Write to PC | | | | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | | | | |
| If No Jump: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | No operation | | | | | | | | | | | | | |

Example: HERE 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 two-cycle instruction. | | | | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | | | | |
| If Jump: | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | Write to PC | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | Write to PC | | | | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | | | | |
| If No Jump: | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> <tr> <td></td><td></td><td></td><td></td></tr> </tbody> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | No operation | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |

Example: HERE 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 two-cycle instruction. | | | | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | | | | |
| If Jump: | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | Write to PC | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | Write to PC | | | | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | | | | |
| If No Jump: | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> <tr> <td></td><td></td><td></td><td></td></tr> </tbody> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | No operation | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | |

Example: HERE BNZ Jump

Before Instruction
 PC = address (HERE)

After Instruction
 If ZERO = 0;
 PC = address (Jump)
 If ZERO = 1;
 PC = address (HERE + 2)

PIC18(L)F2X/45K50

BRA

Unconditional Branch

| | | | | | |
|-------------------|--|--------------|--------------|------|------|
| Syntax: | BRA n | | | | |
| Operands: | $-1024 \leq n \leq 1023$ | | | | |
| Operation: | $(PC) + 2 + 2n \rightarrow PC$ | | | | |
| Status Affected: | None | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1101</td> <td>0nnn</td> <td>nnnn</td> <td>nnnn</td> </tr> </table> | 1101 | 0nnn | nnnn | nnnn |
| 1101 | 0nnn | nnnn | nnnn | | |
| Description: | Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC + 2 + 2n$. This instruction is a two-cycle instruction. | | | | |
| Words: | 1 | | | | |
| Cycles: | 2 | | | | |
| Q Cycle Activity: | | | | | |
| Q1 | Q2 | Q3 | Q4 | | |
| Decode | Read literal 'n' | Process Data | Write to PC | | |
| No operation | No operation | No operation | No operation | | |

Example: HERE BRA Jump

Before Instruction
 PC = address (HERE)
 After Instruction
 PC = address (Jump)

BSF

Bit Set f

| | | | | | |
|-------------------|---|--------------|--------------------|------|------|
| Syntax: | BSF f, b {,a} | | | | |
| Operands: | $0 \leq f \leq 255$ $0 \leq b \leq 7$ $a \in [0,1]$ | | | | |
| Operation: | $1 \rightarrow f$ | | | | |
| Status Affected: | None | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1000</td> <td>bbba</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 1000 | bbba | ffff | ffff |
| 1000 | bbba | ffff | ffff | | |
| Description: | <p>Bit 'b' in register 'f' is set. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | |
| 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 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1] | | | |
| Operation: | | skip if (f) = 0 | | | |
| Status Affected: | | None | | | |
| Encoding: | | 1011 | bbba | ffff | ffff |
| Description: | | <p>If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh).</p> <p>See Section 27.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</p> | | | |
| 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 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 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1] | | | |
| Operation: | | skip if (f) = 1 | | | |
| Status Affected: | | None | | | |
| Encoding: | | 1010 | bbba | ffff | ffff |
| Description: | | <p>If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh).</p> <p>See Section 27.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</p> | | | |

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 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)

PIC18(L)F2X/45K50

| BTG Bit Toggle f | | | | | |
|--|--|--------------|--------------------|------|------|
| Syntax: | BTG f, b {,a} | | | | |
| Operands: | $0 \leq f \leq 255$ $0 \leq b < 7$ $a \in [0,1]$ | | | | |
| Operation: | $(f) \rightarrow f$ | | | | |
| Status Affected: | None | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0111</td> <td>bbba</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0111 | bbba | ffff | ffff |
| 0111 | bbba | ffff | ffff | | |
| Description: | <p>Bit 'b' in data memory location 'f' is inverted.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | |
| Words: | 1 | | | | |
| Cycles: | 1 | | | | |
| Q Cycle Activity: | | | | | |
| Q1 Q2 Q3 Q4 | | | | | |
| Decode | Read register 'f' | Process Data | Write register 'f' | | |

Example: BTG PORTC, 4, 0

Before Instruction:

PORTC = 0111 0101 [75h]

After Instruction:

PORTC = 0110 0101 [65h]

| BOV Branch if Overflow | | | | | |
|--|--|--------------|--------------|------|------|
| Syntax: | BOV n | | | | |
| Operands: | $-128 \leq n \leq 127$ | | | | |
| Operation: | if OVERFLOW bit is '1' $(PC) + 2 + 2n \rightarrow PC$ | | | | |
| Status Affected: | None | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1110</td> <td>0100</td> <td>nnnn</td> <td>nnnn</td> </tr> </table> | 1110 | 0100 | nnnn | nnnn |
| 1110 | 0100 | nnnn | nnnn | | |
| Description: | <p>If the OVERFLOW bit is '1', then the program will branch.</p> <p>The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC + 2 + 2n$. This instruction is then a two-cycle instruction.</p> | | | | |
| Words: | 1 | | | | |
| Cycles: | 1(2) | | | | |
| Q Cycle Activity: | | | | | |
| 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: | $-128 \leq n \leq 127$ | | | | | | | | | | | | |
| Operation: | if ZERO bit is '1' $(PC) + 2 + 2n \rightarrow PC$ | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>0000</td><td>nnnn</td><td>nnnn</td></tr> </table> | 1110 | 0000 | nnnn | nnnn | | | | | | | | |
| 1110 | 0000 | nnnn | nnnn | | | | | | | | | | |
| Description: | If the ZERO bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be $PC + 2 + 2n$. This instruction is then a two-cycle instruction. | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | |
| Q Cycle Activity: | If Jump: | | | | | | | | | | | | |
| | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | Write to PC | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | Write to PC | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |
| If No Jump: | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'n'</td><td>Process Data</td><td>No operation</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | No operation | | | | | | | | | | |

Example: HERE BZ Jump

Before Instruction
 PC = address (HERE)

After Instruction
 If ZERO PC = 1;
 If ZERO PC = address (Jump)
 PC = 0;
 PC = address (HERE + 2)

| CALL | Subroutine Call | | | | | | | | | | | | |
|-------------------|--|------------------|-------------------------------------|----------|---------|--------|------------------------|------------------|-------------------------------------|--------------|--------------|--------------|--------------|
| Syntax: | CALL k {s} | | | | | | | | | | | | |
| Operands: | $0 \leq k \leq 1048575$ $s \in [0,1]$ | | | | | | | | | | | | |
| Operation: | $(PC) + 4 \rightarrow TOS$, $k \rightarrow PC<20:1>$, if $s = 1$ $(W) \rightarrow WS$, $(Status) \rightarrow STATUS$, $(BSR) \rightarrow BSRS$ | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>1110</td><td>110s</td><td>k_7k_6</td><td>kkk_0</td></tr> <tr><td>1111</td><td>$k_{19}k_8$</td><td>kkkk</td><td>kkkk</td></tr> </table> | 1110 | 110s | k_7k_6 | kkk_0 | 1111 | $k_{19}k_8$ | kkkk | kkkk | | | | |
| 1110 | 110s | k_7k_6 | kkk_0 | | | | | | | | | | |
| 1111 | $k_{19}k_8$ | kkkk | kkkk | | | | | | | | | | |
| Description: | Subroutine call of entire 2-Mbyte memory range. First, return address ($PC + 4$) is pushed onto the return stack. If ' s ' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUS and BSRS. If ' s ' = 0, no update occurs. Then, the 20-bit value ' k ' is loaded into $PC<20:1>$. CALL is a two-cycle instruction. | | | | | | | | | | | | |
| Words: | 2 | | | | | | | | | | | | |
| Cycles: | 2 | | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'<7:0>,</td><td>PUSH PC to stack</td><td>Read literal 'k'<19:8>, Write to PC</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k'<7:0>, | PUSH PC to stack | Read literal 'k'<19:8>, Write to PC | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read literal 'k'<7:0>, | PUSH PC to stack | Read literal 'k'<19:8>, Write to PC | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |

Example: HERE CALL THERE, 1

Before Instruction
 PC = address (HERE)

After Instruction
 PC = address (THERE)
 TOS = address (HERE + 4)
 WS = W
 BSRS = BSR
 STATUS = Status

PIC18(L)F2X/45K50

| CLRF | Clear f | | | | | | | | |
|-------------------|--|--------------|--------------------|------|------|--------|-------------------|--------------|--------------------|
| Syntax: | CLRF f {,a} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $000h \rightarrow f$ $1 \rightarrow Z$ | | | | | | | | |
| Status Affected: | Z | | | | | | | | |
| Encoding: | <table border="1"><tr><td>0110</td><td>101a</td><td>ffff</td><td>ffff</td></tr></table> | 0110 | 101a | ffff | ffff | | | | |
| 0110 | 101a | ffff | ffff | | | | | | |
| Description: | Clears the contents of the specified register. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write register 'f'</td></tr></table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write register 'f' |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write register 'f' | | | | | | |

Example: CLRF FLAG_REG, 1

Before Instruction
FLAG_REG = 5Ah
After Instruction
FLAG_REG = 00h

| CLRWDT | Clear Watchdog Timer | | | | | | | | |
|-------------------|--|--------------|--------------|------|------|--------|--------------|--------------|--------------|
| Syntax: | CLRWDT | | | | | | | | |
| Operands: | None | | | | | | | | |
| Operation: | $000h \rightarrow WDT$, $000h \rightarrow WDT$ postscaler, $1 \rightarrow \overline{TO}$, $1 \rightarrow \overline{PD}$ | | | | | | | | |
| Status Affected: | $\overline{TO}, \overline{PD}$ | | | | | | | | |
| Encoding: | <table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0100</td></tr></table> | 0000 | 0000 | 0000 | 0100 | | | | |
| 0000 | 0000 | 0000 | 0100 | | | | | | |
| Description: | CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, \overline{TO} and \overline{PD} , are set. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1"><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>No operation</td><td>Process Data</td><td>No operation</td></tr></table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | Process Data | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | No operation | Process Data | No operation | | | | | | |

Example: CLRWDT

| | | |
|--------------------|---|-----|
| Before Instruction | | |
| WDT Counter | = | ? |
| After Instruction | | |
| WDT Counter | = | 00h |
| WDT Postscaler | = | 0 |
| \overline{TO} | = | 1 |
| PD | = | 1 |

| COMF | Complement f | | | | | | | | |
|-------------------|---|--------------|----------------------|------|------|--------|-------------------|--------------|----------------------|
| Syntax: | COMF f {,d {,a}} | | | | | | | | |
| Operands: | 0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] | | | | | | | | |
| Operation: | (f) → dest | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0001</td><td>11da</td><td>ffff</td><td>ffff</td></tr> </table> | 0001 | 11da | ffff | ffff | | | | |
| 0001 | 11da | ffff | ffff | | | | | | |
| Description: | <p>The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

Example: 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 ≤ f ≤ 255 a ∈ [0,1] | | | | |
| Operation: | (f) – (W), skip if (f) = (W) (unsigned comparison) | | | | |
| Status Affected: | None | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0110</td><td>001a</td><td>ffff</td><td>ffff</td></tr> </table> | 0110 | 001a | ffff | ffff |
| 0110 | 001a | ffff | ffff | | |
| Description: | <p>Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If f = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | |

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 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)

PIC18(L)F2X/45K50

| CPFSGT | Compare f with W, skip if f > W | | | | | | | | | | | | |
|--|--|--------------|--------------|------|------|--------------|-------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Syntax: | CPFSGT f {,a} | | | | | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $a \in [0,1]$ | | | | | | | | | | | | |
| Operation: | $(f) - (W)$, skip if $(f) > (W)$ (unsigned comparison) | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1"><tr><td>0110</td><td>010a</td><td>ffff</td><td>ffff</td></tr></table> | 0110 | 010a | ffff | ffff | | | | | | | | |
| 0110 | 010a | ffff | ffff | | | | | | | | | | |
| Description: | <p>Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction.</p> <p>If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | |
| Note: Three cycles if skip and followed by a 2-word instruction. | | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | |
| <table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>No operation</td></tr> </table> | | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read register 'f' | Process Data | No operation | | | | | | | | | | |
| If skip: | | | | | | | | | | | | | |
| <table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | | Q1 | Q2 | Q3 | Q4 | No operation | No operation | No operation | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |
| If skip and followed by 2-word instruction: | | | | | | | | | | | | | |
| <table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | | Q1 | Q2 | Q3 | Q4 | No operation | No operation | No operation | No operation | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |
| Example: | HERE CPFSGT REG, 0 NGREATER : GREATER : | | | | | | | | | | | | |
| Before Instruction | | | | | | | | | | | | | |
| PC = Address (HERE) W = ? | | | | | | | | | | | | | |
| After Instruction | | | | | | | | | | | | | |
| If REG > W; PC = Address (GREATER) | | | | | | | | | | | | | |
| If REG \leq W; PC = Address (NGREATER) | | | | | | | | | | | | | |

| CPFSLT | Compare f with W, skip if f < W | | | | | | | | | | | | |
|--|--|--------------|--------------|------|------|--------------|-------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Syntax: | CPFSLT f {,a} | | | | | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $a \in [0,1]$ | | | | | | | | | | | | |
| Operation: | $(f) - (W)$, skip if $(f) < (W)$ (unsigned comparison) | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1"><tr><td>0110</td><td>000a</td><td>ffff</td><td>ffff</td></tr></table> | 0110 | 000a | ffff | ffff | | | | | | | | |
| 0110 | 000a | ffff | ffff | | | | | | | | | | |
| Description: | <p>Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.</p> <p>If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | |
| Note: Three cycles if skip and followed by a 2-word instruction. | | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | |
| <table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>No operation</td></tr> </table> | | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read register 'f' | Process Data | No operation | | | | | | | | | | |
| If skip: | | | | | | | | | | | | | |
| <table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | | Q1 | Q2 | Q3 | Q4 | No operation | No operation | No operation | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |
| If skip and followed by 2-word instruction: | | | | | | | | | | | | | |
| <table border="1"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | | Q1 | Q2 | Q3 | Q4 | No operation | No operation | No operation | No operation | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |

Example: HERE CPFSLT REG, 1
NLESS :
LESS :

Before Instruction

PC = Address (HERE)
W = ?

After Instruction

| | | |
|------------------|--------|-----------------|
| If REG < W; | < | W; |
| PC | = | Address (LESS) |
| If REG \geq W; | \geq | W; |
| PC | = | Address (NLESS) |

| DAW | Decimal Adjust W Register | | | | | | | | |
|-------------------|--|--------------|---------|------|------|--------|-----------------|--------------|---------|
| Syntax: | DAW | | | | | | | | |
| Operands: | None | | | | | | | | |
| Operation: | <p>If $[W<3:0> > 9]$ or $[DC = 1]$ then $(W<3:0>) + 6 \rightarrow W<3:0>;$ else $(W<3:0>) \rightarrow W<3:0>;$</p> <p>If $[W<7:4> + DC > 9]$ or $[C = 1]$ then $(W<7:4>) + 6 + DC \rightarrow W<7:4>;$ $C = 1;$ else $(W<7:4>) + DC \rightarrow W<7:4>;$</p> | | | | | | | | |
| Status Affected: | C | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>0111</td> </tr> </table> | 0000 | 0000 | 0000 | 0111 | | | | |
| 0000 | 0000 | 0000 | 0111 | | | | | | |
| Description: | DAW adjusts the eight-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register W</td> <td>Process Data</td> <td>Write W</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register W | Process Data | Write W |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register W | Process Data | Write W | | | | | | |

Example1:

DAW

Before Instruction

| | | |
|----|---|-----|
| W | = | A5h |
| C | = | 0 |
| DC | = | 0 |

After Instruction

| | | |
|----|---|-----|
| W | = | 05h |
| C | = | 1 |
| DC | = | 0 |

Example 2:

Before Instruction

| | | |
|----|---|-----|
| W | = | CEh |
| C | = | 0 |
| DC | = | 0 |

After Instruction

| | | |
|----|---|-----|
| W | = | 34h |
| C | = | 1 |
| DC | = | 0 |

| DECF | Decrement f | | | | | | | | |
|-------------------|---|--------------|----------------------|------|------|--------|-------------------|--------------|----------------------|
| Syntax: | DECF f {,d {,a}} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $(f) - 1 \rightarrow \text{dest}$ | | | | | | | | |
| Status Affected: | C, DC, N, OV, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table;"> <tr> <td>0000</td> <td>01da</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0000 | 01da | ffff | ffff | | | | |
| 0000 | 01da | ffff | ffff | | | | | | |
| Description: | <p>Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

Example: DECF CNT, 1, 0

Before Instruction

| | | |
|-----|---|-----|
| CNT | = | 01h |
| Z | = | 0 |

After Instruction

| | | |
|-----|---|-----|
| CNT | = | 00h |
| Z | = | 1 |

PIC18(L)F2X/45K50

| DECFSZ | Decrement f, skip if 0 | DCFSNZ | Decrement f, skip if not 0 | | | | | | | | | | | | |
|--|---|------------------|---|----|----|----|----|--------------|-------------------|--------------|----------------------|--------------|--------------|--------------|--------------|
| Syntax: | DECFSZ f {,d {,a}} | Syntax: | DCFSNZ f {,d {,a}} | | | | | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | | | | | |
| Operation: | $(f) - 1 \rightarrow \text{dest}$, skip if result = 0 | Operation: | $(f) - 1 \rightarrow \text{dest}$, skip if result $\neq 0$ | | | | | | | | | | | | |
| Status Affected: | None | Status Affected: | None | | | | | | | | | | | | |
| Encoding: | 0010 11da ffff ffff | Encoding: | 0100 11da ffff ffff | | | | | | | | | | | | |
| Description: | <p>The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | Description: | <p>The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | | | | | |
| Words: | 1 | Words: | 1 | | | | | | | | | | | | |
| Cycles: | 1(2) | Cycles: | 1(2) | | | | | | | | | | | | |
| Note: Three cycles if skip and followed by a 2-word instruction. | | | | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | | | |
| <table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </tbody> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | | | | | | | |
| If skip: | | | | | | | | | | | | | | | |
| <table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table> | | | | Q1 | Q2 | Q3 | Q4 | No operation | No operation | No operation | No operation | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | | | |
| If skip and followed by 2-word instruction: | | | | | | | | | | | | | | | |
| <table border="1"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> <tr> <td>No operation</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </tbody> </table> | | | | Q1 | Q2 | Q3 | Q4 | No operation | No operation | No operation | No operation | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | | | |
| <u>Example:</u> | HERE DECFSZ CNT, 1, 1 GOTO LOOP CONTINUE | <u>Example:</u> | HERE DCFSNZ TEMP, 1, 0 ZERO : NZERO : | | | | | | | | | | | | |
| Before Instruction | | | | | | | | | | | | | | | |
| PC = Address (HERE) | | | | | | | | | | | | | | | |
| After Instruction | | | | | | | | | | | | | | | |
| CNT = CNT - 1 | | | | | | | | | | | | | | | |
| If CNT = 0; | | | | | | | | | | | | | | | |
| PC = Address (CONTINUE) | | | | | | | | | | | | | | | |
| If CNT \neq 0; | | | | | | | | | | | | | | | |
| PC = Address (HERE + 2) | | | | | | | | | | | | | | | |
| Before Instruction | | | | | | | | | | | | | | | |
| TEMP = ? | | | | | | | | | | | | | | | |
| After Instruction | | | | | | | | | | | | | | | |
| TEMP = TEMP - 1, | | | | | | | | | | | | | | | |
| If TEMP = 0; | | | | | | | | | | | | | | | |
| PC = Address (ZERO) | | | | | | | | | | | | | | | |
| If TEMP \neq 0; | | | | | | | | | | | | | | | |
| PC = Address (NZERO) | | | | | | | | | | | | | | | |

| GOTO | Unconditional Branch | | | |
|-------------------|--|--------------|--|----|
| Syntax: | GOTO k | | | |
| Operands: | $0 \leq k \leq 1048575$ | | | |
| Operation: | $k \rightarrow PC<20:1>$ | | | |
| Status Affected: | None | | | |
| Encoding: | 1st word (k<7:0>) 1110 1111 $k_7k_6k_5$ 2nd word(k<19:8>) 1111 $k_{19}k_{18}k_{17}$ $k_{16}k_{15}k_{14}$ $k_{13}k_{12}k_8$ | | | |
| Description: | GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle instruction. | | | |
| Words: | 2 | | | |
| Cycles: | 2 | | | |
| Q Cycle Activity: | | | | |
| | Q1 | Q2 | Q3 | Q4 |
| Decode | Read literal 'k'<7:0>, No operation | No operation | Read literal 'k'<19:8>, Write to PC | |
| | No operation | No operation | No operation | |

Example: GOTO THERE

After Instruction
 PC = Address (THERE)

| INCF | Increment f | | | |
|------------------|--|------|------|------|
| Syntax: | INCF f {,d {,a}} | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | |
| Operation: | $(f + 1) \rightarrow \text{dest}$ | | | |
| Status Affected: | C, DC, N, OV, Z | | | |
| Encoding: | 0010 | 10da | ffff | ffff |
| Description: | The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "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 |

PIC18(L)F2X/45K50

| INCF SZ | Increment f, skip if 0 | | | | INFS NZ | Increment f, skip if not 0 | | | |
|--|--|-------------------|--------------|----------------------|---|--|-------------------|--------------|----------------------|
| Syntax: | INCF SZ f {,d {,a}} | | | | Syntax: | INFS NZ f {,d {,a}} | | | |
| Operands: | 0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] | | | | Operands: | 0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] | | | |
| Operation: | (f) + 1 → dest, skip if result = 0 | | | | Operation: | (f) + 1 → dest, skip if result ≠ 0 | | | |
| Status Affected: | None | | | | Status Affected: | None | | | |
| Encoding: | 0011 11da ffff ffff | | | | Encoding: | 0100 10da ffff ffff | | | |
| Description: | The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details. | | | | Description: | The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details. | | | |
| Words: | 1 | | | | Words: | 1 | | | |
| Cycles: | 1(2) | | | | Cycles: | 1(2) | | | |
| | Note: 3 cycles if skip and followed by a 2-word instruction. | | | | Note: 3 cycles if skip and followed by a 2-word instruction. | | | | |
| Q Cycle Activity: | | | | | | | | | |
| | Q1 | Q2 | Q3 | Q4 | | Q1 | Q2 | Q3 | Q4 |
| | Decode | Read register 'f' | Process Data | Write to destination | | Decode | Read register 'f' | Process Data | Write to destination |
| If skip: | | | | | If skip: | | | | |
| | Q1 | Q2 | Q3 | Q4 | | Q1 | Q2 | Q3 | Q4 |
| | No operation | No operation | No operation | No operation | | No operation | No operation | No operation | No operation |
| If skip and followed by 2-word instruction: | | | | | If skip and followed by 2-word instruction: | | | | |
| | Q1 | Q2 | Q3 | Q4 | | Q1 | Q2 | Q3 | Q4 |
| | No operation | No operation | No operation | No operation | | No operation | No operation | No operation | No operation |
| | No operation | No operation | No operation | No operation | | No operation | No operation | No operation | No operation |
| <u>Example:</u> | HERE INCF SZ CNT, 1, 0 NZERO : ZERO : | | | | <u>Example:</u> | HERE INFS NZ REG, 1, 0 ZERO : NZERO : | | | |
| Before Instruction | | | | | | | | | |
| PC = Address (HERE) | | | | | | | | | |
| After Instruction | | | | | | | | | |
| CNT = CNT + 1 If CNT = 0; PC = Address (ZERO) If CNT ≠ 0; PC = Address (NZERO) | | | | | | | | | |
| PC = Address (NZERO) | | | | | | | | | |
| 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: | $0 \leq k \leq 255$ | | | | | | | | |
| Operation: | (W).OR. k → W | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>1001</td> <td>kkkk</td> <td>kkkk</td> </tr> </table> | 0000 | 1001 | kkkk | kkkk | | | | |
| 0000 | 1001 | kkkk | kkkk | | | | | | |
| Description: | The contents of W are ORed with the eight-bit literal 'k'. The result is placed in W. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read literal 'k'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write to W</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | Write to W |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read literal 'k' | Process Data | Write to W | | | | | | |

Example: IORLW 35h

Before Instruction

W = 9Ah

After Instruction

W = BFh

| IORWF | Inclusive OR W with f | | | | | | | | |
|-------------------|--|-----------------|-------------------------|------|------|--------|----------------------|-----------------|-------------------------|
| Syntax: | IORWF f {,d {,a}} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | |
| Operation: | (W).OR. (f) → dest | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0001</td> <td>00da</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0001 | 00da | ffff | ffff | | | | |
| 0001 | 00da | ffff | ffff | | | | | | |
| Description: | <p>Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read register 'f'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">Write to destination</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

Example: IORWF RESULT, 0, 1

Before Instruction

RESULT = 13h

W = 91h

After Instruction

RESULT = 13h

W = 93h

PIC18(L)F2X/45K50

| LFSR | | Load FSR | | |
|-------------------|---|----------------------|------------------------------------|--------------------------------|
| Syntax: | LFSR f, k | | | |
| Operands: | 0 ≤ f ≤ 2 0 ≤ k ≤ 4095 | | | |
| Operation: | k → FSRf | | | |
| Status Affected: | None | | | |
| Encoding: | 1110 1111 | 1110 0000 | 00ff k ₇ 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 | |
| | 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 |
| | Q4 | | | |

Example: LFSR 2, 3ABh

After Instruction

| | | |
|-------|---|-----|
| FSR2H | = | 03h |
| FSR2L | = | ABh |

| MOVE | | Move f | |
|------------------|--|--------|--|
| Syntax: | MOVEF f {,d {,a}} | | |
| Operands: | 0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] | | |
| Operation: | f → dest | | |
| Status Affected: | N, Z | | |
| Encoding: | 0101 00da ffff ffff | | |
| Description: | The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. Location 'f' can be anywhere in the 256-byte bank. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 “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: MOVEF 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 ≤ f _s ≤ 4095 0 ≤ f _d ≤ 4095 | | | | | | | | | | | | |
| Operation: | (f _s) → f _d | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | | | | | | | | | | | | | |
| 1st word (source) | 1100 ffff ffff ffff f _s | | | | | | | | | | | | |
| 2nd word (destin.) | 1111 ffff ffff ffff f _d | | | | | | | | | | | | |
| Description: | <p>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.</p> <p>Either source or destination can be W (a useful special situation).</p> <p>MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).</p> <p>The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.</p> | | | | | | | | | | | | |
| Words: | 2 | | | | | | | | | | | | |
| Cycles: | 2 (3) | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | |
| | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f' (src)</td> <td>Process Data</td> <td>No operation</td> </tr> <tr> <td>Decode</td> <td>No operation No dummy read</td> <td>No operation</td> <td>Write register 'f' (dest)</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' (src) | Process Data | No operation | Decode | No operation No dummy read | No operation | Write register 'f' (dest) |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read register 'f' (src) | Process Data | No operation | | | | | | | | | | |
| Decode | No operation No dummy read | No operation | Write register 'f' (dest) | | | | | | | | | | |

Example: MOVFF REG1, REG2

Before Instruction

REG1 = 33h
REG2 = 11h

After Instruction

REG1 = 33h
REG2 = 33h

| MOVLB | Move literal to low nibble in BSR | | | | | | | | |
|-------------------|---|--------------|--------------------------|------|------|--------|------------------|--------------|--------------------------|
| Syntax: | MOVLB k | | | | | | | | |
| Operands: | 0 ≤ k ≤ 255 | | | | | | | | |
| Operation: | k → BSR | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | | | | | | | | | |
| | <table border="1"> <tr> <td>0000</td> <td>0001</td> <td>kkkk</td> <td>kkkk</td> </tr> </table> | 0000 | 0001 | kkkk | kkkk | | | | |
| 0000 | 0001 | kkkk | kkkk | | | | | | |
| Description: | <p>The eight-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '0', regardless of the value of k_{7:k₄}.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | |
| | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read literal 'k'</td> <td>Process Data</td> <td>Write literal 'k' to BSR</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | Write literal 'k' to BSR |
| 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

PIC18(L)F2X/45K50

MOVlw Move literal to W

| | |
|------------------|---|
| Syntax: | MOVlw k |
| Operands: | $0 \leq k \leq 255$ |
| Operation: | $k \rightarrow W$ |
| Status Affected: | None |
| Encoding: | 0000 1110 kkkk kkkk |
| Description: | The eight-bit literal 'k' is loaded into W. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|------------------|--------------|------------|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: MOVlw 5Ah

After Instruction

W = 5Ah

MOVwf Move W to f

| | |
|------------------|--------------------------------------|
| Syntax: | MOVwf f {,a} |
| Operands: | $0 \leq f \leq 255$ $a \in [0,1]$ |
| Operation: | $(W) \rightarrow f$ |
| Status Affected: | None |
| Encoding: | 0110 111a ffff ffff |

Description:

Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 27.2.3 "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: | $0 \leq k \leq 255$ | | | | | | | | |
| Operation: | $(W) \times k \rightarrow PRODH:PRODL$ | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>1101</td> <td>kkkk</td> <td>kkkk</td> </tr> </table> | 0000 | 1101 | kkkk | kkkk | | | | |
| 0000 | 1101 | kkkk | kkkk | | | | | | |
| Description: | <p>An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.</p> <p>None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | |
| | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write registers PRODH: PRODL</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | Write registers PRODH: PRODL |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read literal 'k' | Process Data | Write registers PRODH: PRODL | | | | | | |

Example: MULLW 0C4h

Before Instruction

| | | |
|-------|---|-----|
| W | = | E2h |
| PRODH | = | ? |
| PRODL | = | ? |

After Instruction

| | | |
|-------|---|-----|
| W | = | E2h |
| PRODH | = | ADh |
| PRODL | = | 08h |

| MULWF | Multiply W with f | | | | | | | | |
|-------------------|--|--------------|------------------------------|------|------|--------|-------------------|--------------|------------------------------|
| Syntax: | MULWF f {,a} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $(W) \times (f) \rightarrow PRODH:PRODL$ | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>001a</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0000 | 001a | ffff | ffff | | | | |
| 0000 | 001a | ffff | ffff | | | | | | |
| Description: | <p>An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged.</p> <p>None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | |
| | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write registers PRODH: PRODL</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write registers PRODH: PRODL |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write registers PRODH: PRODL | | | | | | |

Example: MULWF REG, 1

Before Instruction

| | | |
|-------|---|-----|
| W | = | C4h |
| REG | = | B5h |
| PRODH | = | ? |
| PRODL | = | ? |

After Instruction

| | | |
|-------|---|-----|
| W | = | C4h |
| REG | = | B5h |
| PRODH | = | 8Ah |
| PRODL | = | 94h |

PIC18(L)F2X/45K50

| NEGF | Negate f | | | | | | | | |
|-------------------|---|--------------|--------------|------|------|--------|--------------|--------------|--------------|
| Syntax: | NEGF f {,a} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $a \in [0, 1]$ | | | | | | | | |
| Operation: | $(\bar{f}) + 1 \rightarrow f$ | | | | | | | | |
| Status Affected: | N, OV, C, DC, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0110</td><td>110a</td><td>ffff</td><td>ffff</td></tr> </table> | 0110 | 110a | ffff | ffff | | | | |
| 0110 | 110a | ffff | ffff | | | | | | |
| Description: | <p>Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | No operation | No operation | No operation | | | | | | |

Example: NEGF REG, 1

Before Instruction

REG = 0011 1010 [3Ah]

After Instruction

REG = 1100 0110 [C6h]

| NOP | No Operation | | | | | | | | |
|-------------------|--|--------------|--------------|------|------|--------|--------------|--------------|--------------|
| Syntax: | NOP | | | | | | | | |
| Operands: | None | | | | | | | | |
| Operation: | No operation | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0000</td></tr> <tr><td>1111</td><td>xxxx</td><td>xxxx</td><td>xxxx</td></tr> </table> | 0000 | 0000 | 0000 | 0000 | 1111 | xxxx | xxxx | xxxx |
| 0000 | 0000 | 0000 | 0000 | | | | | | |
| 1111 | xxxx | xxxx | xxxx | | | | | | |
| Description: | No operation. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>No operation</td><td>No operation</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | No operation | No operation | No operation | | | | | | |

Example:

None.

| POP | Pop Top of Return Stack | | | | | | | | |
|-------------------|--|---------------|--------------|------|------|--------|--------------|---------------|--------------|
| Syntax: | POP | | | | | | | | |
| Operands: | None | | | | | | | | |
| Operation: | (TOS) → bit bucket | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0110</td></tr> </table> | 0000 | 0000 | 0000 | 0110 | | | | |
| 0000 | 0000 | 0000 | 0110 | | | | | | |
| Description: | 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: | <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">POP TOS value</td> <td style="text-align: center;">No operation</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | POP TOS value | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | No operation | POP TOS value | No operation | | | | | | |

| | | |
|--------------------------------|-------------|-----|
| <u>Example:</u> | POP GOTO | NEW |
| | | |
| 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: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0000</td><td>0101</td></tr> </table> | 0000 | 0000 | 0000 | 0101 | | | | |
| 0000 | 0000 | 0000 | 0101 | | | | | | |
| Description: | The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">PUSH PC + 2 onto return stack</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | PUSH PC + 2 onto return stack | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | PUSH PC + 2 onto return stack | No operation | No operation | | | | | | |

| | |
|------------------------------|------|
| <u>Example:</u> | PUSH |
| | |
| Before Instruction | |
| TOS = 345Ah | |
| PC = 0124h | |
| | |
| After Instruction | |
| PC = 0126h | |
| TOS = 0126h | |
| Stack (1 level down) = 345Ah | |

PIC18(L)F2X/45K50

| RCALL | Relative Call | | | | | | | | | | | | |
|-------------------|---|-----------------|-----------------|------|------|--------|----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Syntax: | RCALL n | | | | | | | | | | | | |
| Operands: | -1024 ≤ n ≤ 1023 | | | | | | | | | | | | |
| Operation: | (PC) + 2 → TOS, (PC) + 2 + 2n → PC | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1101</td> <td>1nnn</td> <td>nnnn</td> <td>nnnn</td> </tr> </table> | 1101 | 1nnn | nnnn | nnnn | | | | | | | | |
| 1101 | 1nnn | nnnn | nnnn | | | | | | | | | | |
| Description: | Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction. | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 2 | | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Start Reset</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Start Reset | No operation | No operation | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Start Reset | No operation | No operation | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |

Example: HERE RCALL Jump

Before Instruction

PC = Address (HERE)

After Instruction

PC = Address (Jump)

TOS = Address (HERE + 2)

| RESET | Reset | | | | | | | | |
|-------------------|--|-----------------|-----------------|------|------|--------|----------------|-----------------|-----------------|
| Syntax: | RESET | | | | | | | | |
| Operands: | None | | | | | | | | |
| Operation: | Reset all registers and flags that are affected by a MCLR Reset. | | | | | | | | |
| Status Affected: | All | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>1111</td> <td>1111</td> </tr> </table> | 0000 | 0000 | 1111 | 1111 | | | | |
| 0000 | 0000 | 1111 | 1111 | | | | | | |
| Description: | This instruction provides a way to execute a MCLR Reset by software. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Start Reset</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Start Reset | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Start Reset | No operation | No operation | | | | | | |

Example: RESET

After Instruction

Registers = Reset Value

Flags* = Reset Value

| 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, (STATUS) → 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, STATUS and BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers occurs. | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 2 | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">POP PC from stack Set GIEH or GIEL</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | No operation | POP PC from stack Set GIEH or GIEL | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | No operation | No operation | POP PC from stack Set GIEH or GIEL | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |

Example: RETFIE 1

After Interrupt

| | | |
|---------------------|---|--------|
| PC | = | TOS |
| W | = | WS |
| BSR | = | BSRS |
| Status | = | STATUS |
| GIE/GIEH, PEIE/GIEL | = | 1 |

| RETLW | Return literal to W | | | | | | | | | | | | |
|-------------------|---|--------------|-------------------------------|----|----|--------|------------------|--------------|-------------------------------|--------------|--------------|--------------|--------------|
| Syntax: | RETLW k | | | | | | | | | | | | |
| Operands: | 0 ≤ 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 eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged. | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 2 | | | | | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | | | | | |
| | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">Read literal 'k'</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">POP PC from stack, Write to W</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | POP PC from stack, Write to W | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read literal 'k' | Process Data | POP PC from stack, Write to W | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |

Example:

```
CALL TABLE ; W contains table
; offset value
; W now has
; table value
```

:

```
TABLE
ADDWF PCL ; W = offset
RETLW k0 ; Begin table
RETLW k1 ;
:
:
RETLW kn ; End of table
```

Before Instruction

W = 07h

After Instruction

W = value of kn

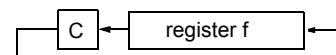
PIC18(L)F2X/45K50

| RETURN | Return from Subroutine | | | | | | | | | | | | |
|-------------------|---|--------------|-------------------|------|------|--------|--------------|--------------|-------------------|--------------|--------------|--------------|--------------|
| Syntax: | RETURN {s} | | | | | | | | | | | | |
| Operands: | $s \in [0,1]$ | | | | | | | | | | | | |
| Operation: | (TOS) \rightarrow PC, if $s = 1$ (WS) \rightarrow W, (STATUS) \rightarrow Status, (BSRS) \rightarrow BSR, PCLATU, PCLATH are unchanged | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0001</td><td>001s</td></tr> </table> | 0000 | 0000 | 0001 | 001s | | | | | | | | |
| 0000 | 0000 | 0001 | 001s | | | | | | | | | | |
| Description: | Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If ' $s=1$ ', the contents of the shadow registers, WS, STATUS and BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If ' $s=0$ ', no update of these registers occurs. | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 2 | | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th> <th style="text-align: center;">Q2</th> <th style="text-align: center;">Q3</th> <th style="text-align: center;">Q4</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">Process Data</td> <td style="text-align: center;">POP PC from stack</td> </tr> <tr> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> <td style="text-align: center;">No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | Process Data | POP PC from stack | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | No operation | Process Data | POP PC from stack | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |

Example: RETURN

After Instruction:
PC = TOS

| RLCF | Rotate Left f through Carry | | | | |
|------------------|---|------|------|------|------|
| Syntax: | RLCF f {,d {,a}} | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | |
| Operation: | $(f <n>) \rightarrow \text{dest} <n+1>$, $(f <7>) \rightarrow C$, $(C) \rightarrow \text{dest} <0>$ | | | | |
| Status Affected: | C, N, Z | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0011</td><td>01da</td><td>ffff</td><td>ffff</td></tr> </table> | 0011 | 01da | ffff | ffff |
| 0011 | 01da | ffff | ffff | | |
| Description: | The contents of register 'f' are rotated one bit to the left through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "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: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | | | | |
| Operation: | $(f < n >) \rightarrow \text{dest} < n + 1 >$, $(f < 7 >) \rightarrow \text{dest} < 0 >$ | | | | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0100</td><td>01da</td><td>ffff</td><td>ffff</td></tr> </table> | | | | 0100 | 01da | ffff | ffff | | | | |
| 0100 | 01da | ffff | ffff | | | | | | | | | |
| Description: | <p>The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p>  | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | |
| Cycles: | 1 | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th> </tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td> </tr> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | | | | |

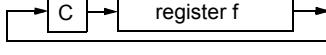
Example: RLNCF REG, 1, 0

Before Instruction

REG = 1010 1011

After Instruction

REG = 0101 0111

| RRCF | Rotate Right f through Carry | | | | | | | | | | | |
|-------------------|---|--------------|----------------------|--|------|------|------|------|--------|-------------------|--------------|----------------------|
| Syntax: | RRCF f {,d {,a}} | | | | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | | | | |
| Operation: | $(f < n >) \rightarrow \text{dest} < n - 1 >$, $(f < 0 >) \rightarrow C$, $(C) \rightarrow \text{dest} < 7 >$ | | | | | | | | | | | |
| Status Affected: | C, N, Z | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0011</td><td>00da</td><td>ffff</td><td>ffff</td></tr> </table> | | | | 0011 | 00da | ffff | ffff | | | | |
| 0011 | 00da | ffff | ffff | | | | | | | | | |
| Description: | <p>The contents of register 'f' are rotated one bit to the right through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p>  | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | |
| Cycles: | 1 | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th> </tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td> </tr> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | | | | |

Example: RRCF REG, 0, 0

Before Instruction

REG = 1110 0110

C = 0

After Instruction

REG = 1110 0110

W = 0111 0011

C = 0

PIC18(L)F2X/45K50

| RRNCF | Rotate Right f (No Carry) | SETF | Set f |
|-------------------|---|----------------------|--|
| Syntax: | RRNCF f {,d {,a}} | Syntax: | SETF f {,a} |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | Operands: | $0 \leq f \leq 255$ $a \in [0,1]$ |
| Operation: | $(f < n >) \rightarrow \text{dest} < n - 1 >$, $(f < 0 >) \rightarrow \text{dest} < 7 >$ | Operation: | $\text{FFh} \rightarrow f$ |
| Status Affected: | N, Z | Status Affected: | None |
| Encoding: | 0100 00da ffff ffff | Encoding: | 0110 100a ffff ffff |
| Description: | The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. | Description: | The contents of the specified register are set to FFh. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. |
| Words: | 1 | Words: | 1 |
| Cycles: | 1 | Cycles: | 1 |
| Q Cycle Activity: | | Q Cycle Activity: | |
| | Q1 Q2 Q3 Q4 | | Q1 Q2 Q3 Q4 |
| | Decode Read register 'f' | Read register 'f' | Process Data |
| | Process Data | Write to destination | Write register 'f' |

Example 1: RRNCF REG, 1, 0

Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

Example: SETF REG, 1

| | |
|--------------------|-------|
| Before Instruction | |
| REG | = 5Ah |
| After Instruction | |
| REG | = FFh |

| SLEEP | Enter Sleep mode | | | | | | | | |
|-------------------|---|--------------|-------------|------|------|--------|--------------|--------------|-------------|
| Syntax: | SLEEP | | | | | | | | |
| Operands: | None | | | | | | | | |
| Operation: | 00h → WDT, 0 → WDT postscaler, 1 → $\overline{\text{TO}}$, 0 → PD | | | | | | | | |
| Status Affected: | $\overline{\text{TO}}$, PD | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0000</td><td>0000</td><td>0000</td><td>0011</td></tr></table> | 0000 | 0000 | 0000 | 0011 | | | | |
| 0000 | 0000 | 0000 | 0011 | | | | | | |
| Description: | The Power-down Status bit (PD) is cleared. The Time-out Status bit ($\overline{\text{TO}}$) is set. The Watchdog Timer and its postscaler are cleared. The processor is put into Sleep mode with the oscillator stopped. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>No operation</td><td>Process Data</td><td>Go to Sleep</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | Process Data | Go to Sleep |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | No operation | Process Data | Go to Sleep | | | | | | |

Example: SLEEP

Before Instruction

$\overline{\text{TO}} = ?$
 $\overline{\text{PD}} = ?$

After Instruction

$\overline{\text{TO}} = 1 \dagger$
 $\overline{\text{PD}} = 0$

† If WDT causes wake-up, this bit is cleared.

| SUBFWB | Subtract f from W with borrow | | | | |
|------------------|---|------|------|------|------|
| Syntax: | SUBFWB f {,d {,a}} | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0, 1]$ $a \in [0, 1]$ | | | | |
| Operation: | $(W) - (f) - (\overline{C}) \rightarrow \text{dest}$ | | | | |
| Status Affected: | N, OV, C, DC, Z | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"><tr><td>0101</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table> | 0101 | 01da | ffff | ffff |
| 0101 | 01da | ffff | ffff | | |
| Description: | <p>Subtract register 'f' and CARRY flag (borrow) from W (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | |

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: SUBFWB REG, 1, 0

Before Instruction

REG = 3
W = 2
C = 1

After Instruction

REG = FF
W = 2
C = 0
Z = 0
N = 1 ; result is negative

Example 2: SUBFWB REG, 0, 0

Before Instruction

REG = 2
W = 5
C = 1

After Instruction

REG = 2
W = 3
C = 1
Z = 0
N = 0 ; result is positive

Example 3: SUBFWB REG, 1, 0

Before Instruction

REG = 1
W = 2
C = 0

After Instruction

REG = 0
W = 2
C = 1
Z = 1 ; result is zero
N = 0

PIC18(L)F2X/45K50

| SUBLW | Subtract W from literal | | | | | | | | | | | |
|-------------------|--|--------------|------------|------|----|----|----|----|--------|------------------|--------------|------------|
| Syntax: | SUBLW k | | | | | | | | | | | |
| Operands: | $0 \leq k \leq 255$ | | | | | | | | | | | |
| Operation: | $k - (W) \rightarrow W$ | | | | | | | | | | | |
| Status Affected: | N, OV, C, DC, Z | | | | | | | | | | | |
| Encoding: | 0000 | 1000 | kkkk | kkkk | | | | | | | | |
| Description | W is subtracted from the eight-bit literal 'k'. The result is placed in W. | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | |
| Cycles: | 1 | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to W</td></tr> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | Write to W |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | |
| Decode | Read literal 'k' | Process Data | Write to W | | | | | | | | | |

Example 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 03h

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 | | | | | | | | | | | |
|-------------------|--|--------------|----------------------|------|----|----|----|----|--------|-------------------|--------------|----------------------|
| Syntax: | SUBWF f {,d {,a}} | | | | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | | | | |
| Operation: | $(f) - (W) \rightarrow \text{dest}$ | | | | | | | | | | | |
| Status Affected: | N, OV, C, DC, Z | | | | | | | | | | | |
| Encoding: | 0101 | 11da | ffff | ffff | | | | | | | | |
| Description: | Subtract W from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | |
| Cycles: | 1 | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; text-align: center;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | | | | |

Example 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

| SUBWFB | Subtract W from f with Borrow | | | | | | | | |
|-------------------|--|--------------|----------------------|------|------|--------|-------------------|--------------|----------------------|
| Syntax: | SUBWFB f {,d {,a}} | | | | | | | | |
| Operands: | 0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] | | | | | | | | |
| Operation: | (f) – (W) – (C) → dest | | | | | | | | |
| Status Affected: | N, OV, C, DC, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0101</td><td>10da</td><td>ffff</td><td>ffff</td></tr> </table> | 0101 | 10da | ffff | ffff | | | | |
| 0101 | 10da | ffff | ffff | | | | | | |
| Description: | <p>Subtract W and the CARRY flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

Example 1: SUBWFB REG, 1, 0

Before Instruction

| | | | |
|-----|---|-----|-------------|
| REG | = | 19h | (0001 1001) |
| W | = | 0Dh | (0000 1101) |
| C | = | 1 | |

After Instruction

| | | | |
|-----|---|-----|----------------------|
| REG | = | 0Ch | (0000 1100) |
| 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 1110) |
| C | = | 1 | |

After Instruction

| | | | |
|-----|---|-----|-----------------------------|
| REG | = | F5h | (1111 0101) ; [2's comp] |
| W | = | 0Eh | (0000 1110) |
| C | = | 0 | |
| Z | = | 0 | |
| N | = | 1 | ; result is negative |

| SWAPF | Swap f | | | | | | | | |
|-------------------|--|--------------|----------------------|------|------|--------|-------------------|--------------|----------------------|
| Syntax: | SWAPF f {,d {,a}} | | | | | | | | |
| Operands: | 0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] | | | | | | | | |
| Operation: | (f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0> | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0011</td><td>10da</td><td>ffff</td><td>ffff</td></tr> </table> | 0011 | 10da | ffff | ffff | | | | |
| 0011 | 10da | ffff | ffff | | | | | | |
| Description: | <p>The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

Example: SWAPF REG, 1, 0

Before Instruction

| | | |
|-----|---|-----|
| REG | = | 53h |
|-----|---|-----|

After Instruction

| | | |
|-----|---|-----|
| REG | = | 35h |
|-----|---|-----|

PIC18(L)F2X/45K50

| TBLRD | Table Read | | | | |
|------------------|---|------|---|------|---|
| Syntax: | TBLRD (*; *+; *-; +*) | | | | |
| Operands: | None | | | | |
| Operation: | <p>if TBLRD *, (Prog Mem (TBLPTR)) → TABLAT; TBLPTR – No Change;</p> <p>if TBLRD *+, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) + 1 → TBLPTR;</p> <p>if TBLRD *-, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) – 1 → TBLPTR;</p> <p>if TBLRD +*, (TBLPTR) + 1 → TBLPTR; (Prog Mem (TBLPTR)) → TABLAT;</p> | | | | |
| Status Affected: | None | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>10nn nn=0 * =1 *+ =2 *- =3 +*</td> </tr> </table> | 0000 | 0000 | 0000 | 10nn nn=0 * =1 *+ =2 *- =3 +* |
| 0000 | 0000 | 0000 | 10nn nn=0 * =1 *+ =2 *- =3 +* | | |
| Description: | <p>This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.</p> <p>The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.</p> <ul style="list-style-type: none"> TBLPTR[0] = 0: Least Significant Byte of Program Memory Word TBLPTR[0] = 1: Most Significant Byte of Program Memory Word <p>The TBLRD instruction can modify the value of TBLPTR as follows:</p> <ul style="list-style-type: none"> • 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: | <p>if TBLWT*, (TABLAT) → Holding Register; TBLPTR – No Change;</p> <p>if TBLWT*+, (TABLAT) → Holding Register; (TBLPTR) + 1 → TBLPTR;</p> <p>if TBLWT*-, (TABLAT) → Holding Register; (TBLPTR) – 1 → TBLPTR;</p> <p>if TBLWT+*, (TBLPTR) + 1 → TBLPTR; (TABLAT) → Holding Register;</p> | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>11nn</td> </tr> <tr> <td></td> <td></td> <td></td> <td>nn=0 * =1 *+ =2 *- =3 +*</td> </tr> </table> | 0000 | 0000 | 0000 | 11nn | | | | nn=0 * =1 *+ =2 *- =3 +* |
| 0000 | 0000 | 0000 | 11nn | | | | | | |
| | | | nn=0 * =1 *+ =2 *- =3 +* | | | | | | |
| Description: | <p>This instruction uses the three 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 7.0 “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.</p> <p>TBLPTR[0] = 0: Least Significant Byte of Program Memory Word TBLPTR[0] = 1: Most Significant Byte of Program Memory Word</p> <p>The TBLWT instruction can modify the value of TBLPTR as follows:</p> <ul style="list-style-type: none"> • 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 |

PIC18(L)F2X/45K50

| TSTFSZ | Test f, skip if 0 | | | | |
|------------------|---|------|------|------|------|
| Syntax: | TSTFSZ f {,a} | | | | |
| Operands: | $0 \leq f \leq 255$ $a \in [0,1]$ | | | | |
| Operation: | skip if $f = 0$ | | | | |
| Status Affected: | None | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0110</td><td>011a</td><td>ffff</td><td>ffff</td></tr> </table> | 0110 | 011a | ffff | ffff |
| 0110 | 011a | ffff | ffff | | |
| Description: | If ' f ' = 0, the next instruction fetched during the current instruction execution is discarded and a NOP is executed, making this a two-cycle instruction. If ' a ' is '0', the Access Bank is selected. If ' a ' is '1', the BSR is used to select the GPR bank. If ' a ' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. | | | | |
| 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 TSTFSZ CNT, 1
NZERO :
ZERO :

Before Instruction

PC = Address (HERE)

After Instruction

If CNT = 00h,
PC = Address (ZERO)

If CNT ≠ 00h,
PC = Address (NZERO)

| XORLW | Exclusive OR literal with W | | | | | | | | |
|-------------------|---|--------------|------------|------|------|--------|------------------|--------------|------------|
| Syntax: | XORLW k | | | | | | | | |
| Operands: | $0 \leq k \leq 255$ | | | | | | | | |
| Operation: | (W) .XOR. K → W | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>1010</td><td>kkkk</td><td>kkkk</td></tr> </table> | 0000 | 1010 | kkkk | kkkk | | | | |
| 0000 | 1010 | kkkk | kkkk | | | | | | |
| Description: | The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th> </tr> <tr> <td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to W</td></tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | Write to W |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read literal 'k' | Process Data | Write to W | | | | | | |

Example: XORLW 0AFh

Before Instruction

W = B5h

After Instruction

W = 1Ah

| XORWF | Exclusive OR W with f | | | | | | | | |
|-------------------|--|--------------|----------------------|------|------|--------|-------------------|--------------|----------------------|
| Syntax: | XORWF f {,d {,a}} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $(W) .XOR. (f) \rightarrow \text{dest}$ | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0001</td> <td>10da</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0001 | 10da | ffff | ffff | | | | |
| 0001 | 10da | ffff | ffff | | | | | | |
| Description: | <p>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'.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

Example: XORWF REG, 1, 0

Before Instruction

REG = AFh
W = B5h

After Instruction

REG = 1Ah
W = B5h

27.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F2X/45K50 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 27-3](#). Detailed descriptions are provided in [Section 27.2.2 “Extended Instruction Set”](#). The opcode field descriptions in [Table 27-1](#) 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.

27.2.1 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 27.2.3.1 “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 27-3: 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 f _d (destination) 2nd word | 2 | 1110 | 1011 | 0zzz | zzzz | None |
| MOVSS z _s , z _d | Move z _s (source) to 1st word z _d (destination) 2nd word | 2 | 1111 | ffff | ffff | ffff | None |
| PUSHL k | Store literal at FSR2, decrement FSR2 | 1 | 1110 | 1011 | 1zzz | zzzz | None |
| SUBFSR f, k | Subtract literal from FSR | 1 | 1110 | 1001 | ffkk | kkkk | None |
| SUBULNK k | Subtract literal from FSR2 and return | 2 | 1110 | 1001 | 11kk | kkkk | None |

27.2.2 EXTENDED INSTRUCTION SET

| ADDFSR | Add Literal to FSR | ADDULNK | Add Literal to FSR2 and Return | | | | | | | | | | | | | | | | | | | | |
|-------------------|--|------------------|---|------|------|-----------|---|--------------|--------------|-------------------|---|----|----|----|----|--------|------------------|--------------|--------------|--------------|--------------|--------------|--------------|
| Syntax: | ADDFSR f, k | Syntax: | ADDULNK k | | | | | | | | | | | | | | | | | | | | |
| Operands: | $0 \leq k \leq 63$ $f \in [0, 1, 2]$ | Operands: | $0 \leq k \leq 63$ | | | | | | | | | | | | | | | | | | | | |
| Operation: | $\text{FSR}(f) + k \rightarrow \text{FSR}(f)$ | Operation: | $\text{FSR2} + k \rightarrow \text{FSR2}$, $(\text{TOS}) \rightarrow \text{PC}$ | | | | | | | | | | | | | | | | | | | | |
| Status Affected: | None | Status Affected: | None | | | | | | | | | | | | | | | | | | | | |
| Encoding: | <table border="1"><tr><td>1110</td><td>1000</td><td>ffkk</td><td>kkkk</td></tr></table> | 1110 | 1000 | ffkk | kkkk | Encoding: | <table border="1"><tr><td>1110</td><td>1000</td><td>11kk</td><td>kkkk</td></tr></table> | 1110 | 1000 | 11kk | kkkk | | | | | | | | | | | | |
| 1110 | 1000 | ffkk | kkkk | | | | | | | | | | | | | | | | | | | | |
| 1110 | 1000 | 11kk | kkkk | | | | | | | | | | | | | | | | | | | | |
| Description: | The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'. | Description: | The 6-bit literal 'k' is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS. | | | | | | | | | | | | | | | | | | | | |
| Words: | 1 | Words: | 1 | | | | | | | | | | | | | | | | | | | | |
| Cycles: | 1 | Cycles: | 2 | | | | | | | | | | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to FSR</td></tr></table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | Write to FSR | Q Cycle Activity: | <table border="1"><tr><td>Q1</td><td>Q2</td><td>Q3</td><td>Q4</td></tr><tr><td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to FSR</td></tr><tr><td>No Operation</td><td>No Operation</td><td>No Operation</td><td>No Operation</td></tr></table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | Write to FSR | No Operation | No Operation | No Operation | No Operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | | | | | | | | |
| Decode | Read literal 'k' | Process Data | Write to FSR | | | | | | | | | | | | | | | | | | | | |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | | | | | | | | |
| Decode | Read literal 'k' | Process Data | Write to FSR | | | | | | | | | | | | | | | | | | | | |
| No Operation | No Operation | No Operation | No Operation | | | | | | | | | | | | | | | | | | | | |

Example: ADDFSR 2, 23h

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

Words: 1
Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------------|------------------|--------------|--------------|
| Decode | Read literal 'k' | Process Data | Write to FSR |
| No Operation | No Operation | No Operation | No Operation |

Example: ADDULNK 23h

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 0422h
PC = (TOS)

Note: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).

PIC18(L)F2X/45K50

| CALLW | Subroutine Call Using WREG | | | | | | | | | | | | |
|-------------------|---|---------------------|-----------------|------|------|--------|--------------|---------------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| Syntax: | CALLW | | | | | | | | | | | | |
| Operands: | None | | | | | | | | | | | | |
| Operation: | (PC + 2) → TOS, (W) → PCL, (PCLATH) → PCH, (PCLATU) → PCU | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr><td>0000</td><td>0000</td><td>0001</td><td>0100</td></tr> </table> | 0000 | 0000 | 0001 | 0100 | | | | | | | | |
| 0000 | 0000 | 0001 | 0100 | | | | | | | | | | |
| Description | First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched into PCH and PCU, respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched. Unlike CALL, there is no option to update W, Status or BSR. | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 2 | | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">Q1</th><th style="text-align: center;">Q2</th><th style="text-align: center;">Q3</th><th style="text-align: center;">Q4</th></tr> </thead> <tbody> <tr> <td style="text-align: center;">Decode</td><td style="text-align: center;">Read WREG</td><td style="text-align: center;">PUSH PC to stack</td><td style="text-align: center;">No operation</td></tr> <tr> <td style="text-align: center;">No operation</td><td style="text-align: center;">No operation</td><td style="text-align: center;">No operation</td><td style="text-align: center;">No operation</td></tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read WREG | PUSH PC to stack | No operation | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read WREG | PUSH PC to stack | No operation | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |

Example: HERE CALLW

Before Instruction

```

PC      =    address (HERE)
PCLATH =    10h
PCLATU =    00h
W       =    06h

```

After Instruction

```

PC      =    001006h
TOS     =    address (HERE + 2)
PCLATH =    10h
PCLATU =    00h
W       =    06h

```

| MOVSF | Move Indexed to f |
|--------------|--------------------------|
|--------------|--------------------------|

| | |
|--------------------|---|
| Syntax: | MOVSF [z _s], f _d |
| Operands: | 0 ≤ z _s ≤ 127 0 ≤ f _d ≤ 4095 |
| Operation: | ((FSR2) + z _s) → f _d |
| Status Affected: | None |
| Encoding: | |
| 1st word (source) | 1110 |
| 2nd word (destin.) | 1111 ffff |

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 (00h to FFFh). The MOVSF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register. If the resultant source address points to an indirect addressing register, the value returned will be 00h.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------------------------|--------------------------|---------------------------------|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | No operation No dummy read | No operation | Write register 'f' (dest) |

Example: MOVSF [05h], REG2

Before Instruction

```

FSR2      =    80h
Contents of 85h =    33h
REG2      =    11h

```

After Instruction

```

FSR2      =    80h
Contents of 85h =    33h
REG2      =    33h

```

| MOVSS | Move Indexed to Indexed | | | | | | | | |
|-------------------|--|--------------|----------------------|----|----|--------|----------|--------------|----------------------|
| Syntax: | MOVSS [z _s], [z _d] | | | | | | | | |
| Operands: | 0 ≤ z _s ≤ 127 0 ≤ z _d ≤ 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 | <p>The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets 'z_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).</p> <p>The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.</p> <p>If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP.</p> | | | | | | | | |
| Words: | 2 | | | | | | | | |
| Cycles: | 2 | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | |
| | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read 'k'</td> <td>Process data</td> <td>Write to destination</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read 'k' | Process data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read 'k' | Process data | Write to destination | | | | | | |

Example: MOVSS [05h], [06h]

| | |
|--------------------|-------|
| Before Instruction | |
| FSR2 | = 80h |
| Contents of 85h | = 33h |
| Contents of 86h | = 11h |
| After Instruction | |
| FSR2 | = 80h |
| Contents of 85h | = 33h |
| Contents of 86h | = 33h |

| PUSHL | Store Literal at FSR2, Decrement FSR2 | | | | | | | | |
|-------------------|---|--------------|----------------------|------|------|--------|----------|--------------|----------------------|
| Syntax: | PUSHL k | | | | | | | | |
| Operands: | 0 ≤ k ≤ 255 | | | | | | | | |
| Operation: | k → (FSR2), FSR2 – 1 → FSR2 | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | | | | | | | | | |
| | <table border="1"> <tr> <td>1111</td> <td>1010</td> <td>kkkk</td> <td>kkkk</td> </tr> </table> | 1111 | 1010 | kkkk | kkkk | | | | |
| 1111 | 1010 | kkkk | kkkk | | | | | | |
| Description: | <p>The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation. This instruction allows users to push values onto a software stack.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | | | | | | | | | |
| | <table border="1"> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> <tr> <td>Decode</td> <td>Read 'k'</td> <td>Process data</td> <td>Write to destination</td> </tr> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read 'k' | Process data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read 'k' | Process data | Write to destination | | | | | | |

Example: PUSHL 08h

| | |
|--------------------|---------|
| Before Instruction | |
| FSR2H:FSR2L | = 01ECh |
| Memory (01ECh) | = 00h |
| After Instruction | |
| FSR2H:FSR2L | = 01EBh |
| Memory (01ECh) | = 08h |

PIC18(L)F2X/45K50

| SUBFSR | Subtract Literal from FSR | | | | | | | | | | | |
|-------------------|---|--------------|----------------------|------|----|----|----|----|--------|-------------------|--------------|----------------------|
| Syntax: | SUBFSR f, k | | | | | | | | | | | |
| Operands: | $0 \leq k \leq 63$ $f \in [0, 1, 2]$ | | | | | | | | | | | |
| Operation: | $\text{FSR}(f) - k \rightarrow \text{FSR}f$ | | | | | | | | | | | |
| 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: | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> </tbody> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | | | | |

Example: SUBFSR 2, 23h

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 03DCh

| SUBULNK | Subtract Literal from FSR2 and Return | | | | | | | | | | | | | | | |
|-------------------|--|--------------|----------------------|------|----|----|----|----|--------|-------------------|--------------|----------------------|--------------|--------------|--------------|--------------|
| Syntax: | SUBULNK k | | | | | | | | | | | | | | | |
| Operands: | $0 \leq k \leq 63$ | | | | | | | | | | | | | | | |
| Operation: | $\text{FSR2} - k \rightarrow \text{FSR2}$ (TOS) $\rightarrow \text{PC}$ | | | | | | | | | | | | | | | |
| 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: | <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr> </thead> <tbody> <tr> <td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr> <tr> <td>No Operation</td><td>No Operation</td><td>No Operation</td><td>No Operation</td></tr> </tbody> </table> | | | | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination | No Operation | No Operation | No Operation | No Operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | | | | | | | | |
| No Operation | No Operation | No Operation | No Operation | | | | | | | | | | | | | |

Example: SUBULNK 23h

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 03DCh
PC = (TOS)

27.2.3 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 6.7.1 "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 27.2.3.1 "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.

27.2.3.1 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.

27.2.4 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)F2X/45K50, 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.

PIC18(L)F2X/45K50

| ADDWF | ADD W to Indexed (Indexed Literal Offset mode) | | | |
|-------------------|---|----------|--------------|----------------------|
| Syntax: | ADDWF | [k] {,d} | | |
| Operands: | 0 ≤ k ≤ 95 d ∈ [0,1] | | | |
| Operation: | (W) + ((FSR2) + k) → dest | | | |
| Status Affected: | N, OV, C, DC, Z | | | |
| Encoding: | 0010 | 01d0 | kkkk | kkkk |
| Description: | The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. | | | |
| 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 ≤ f ≤ 95 0 ≤ b ≤ 7 | | | |
| Operation: | 1 → ((FSR2) + k) | | | |
| 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 ≤ k ≤ 95 | | | |
| Operation: | FFh → ((FSR2) + k) | | | |
| Status Affected: | None | | | |
| Encoding: | 0110 | 1000 | kkkk | kkkk |
| Description: | The contents of the register indicated by FSR2, offset by 'k', are set to FFh. | | | |
| Words: | 1 | | | |
| Cycles: | 1 | | | |
| Q Cycle Activity: | 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 |

27.2.5 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)F2X/45K50 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.

28.0 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

28.1 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

28.2 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

28.3 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

28.4 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

28.5 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

28.6 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.

28.7 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 upgradeable 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.

28.8 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.

28.9 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).

28.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.

28.11 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, KEELQ® 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.

28.12 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®

PIC18(L)F2X/45K50

29.0 ELECTRICAL SPECIFICATIONS

29.1 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 | |
| PIC18LF2X/45K50..... | -0.3V to +4.5V |
| PIC18F2X/45K50..... | -0.3V to +6.5V |
| on VUSB3V3 pin ⁽³⁾ | -0.3V to +4.0V |
| on D+ and D- pins | |
| 0Ω source impedance ⁽⁵⁾ | -0.5V to (VUSB3V3 + 0.5V) |
| source impedance ≥28Ω, VUSB3V3 ≥3.0V | -1.0V to + 4.6V) |
| on MCLR/nICRST pin ⁽²⁾ | 0V to +11.0V |
| on all other pins..... | -0.3V to (VDD + 0.3V) |
| Total power dissipation ⁽¹⁾ | 1.0W |
| Maximum current | |
| out of Vss pin | |
| -40°C to +125°C | 125 mA |
| into VDD pin | |
| -40°C to +125°C | 85 mA |
| Input clamp current, I _{IK} (V _I < 0 or V _I > V _{DD}) ⁽⁴⁾ | ±20 mA |
| Output clamp current, I _{OK} (V _O < 0 or V _O > V _{DD}) ⁽⁴⁾ | ±20 mA |
| Maximum output current | |
| sunk by any I/O pin | 25 mA |
| sourced by any I/O pin | 25 mA |
| Maximum current | |
| sunk by all ports (-40°C to +125°C) | 110 mA |
| sourced by all ports (-40°C to +125°C) | 70 mA |

Note 1: Power dissipation is calculated as follows:

$$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$$

- 2:** Voltage spikes below V_{SS} at the MCLR/VPP/RE3 pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a “low” level to the MCLR/VPP/RE3 pin, rather than pulling this pin directly to V_{SS}.
- 3:** V_{USB3V3} must always be ≤V_{DD} + 0.3V. V_{USB3V3} must also be maintained ≥V_{DD} – 0.3V on PIC18LF2X/45K50 devices.
- 4:** Stress rating only. For proper functional operation, I/O pins should be maintained within the -0.3V to (V_{DD} + 0.3V) range, which will not result in injected current. See TB3013 technical brief (DS93013) for details.
- 5:** The original [Universal Serial Bus Specification Revision 2.0](#) indicated that USB devices should withstand 24-hour short circuits of D+ or D- to V_{BUS} voltages. This requirement was later removed in an Engineering Change Notice (ECN) supplement to the USB specifications, which supersedes the original specifications. PIC18F2X/45K50 family devices will typically be able to survive this short-circuit test, but it is recommended to adhere to the absolute maximum specified here to avoid damaging the device.

† NOTICE: Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

29.2 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

PIC18LF2X/45K50

| | |
|---|-------|
| V _{DDMIN} (Fosc \leq 4 MHz, Industrial Temperature)..... | +1.8V |
| V _{DDMIN} (Fosc \leq 48 MHz) | +2.7V |
| V _{DDMAX} | +3.6V |

PIC18F2X/45K50

| | |
|--|-------|
| V _{DDMIN} (Fosc \leq 20 MHz, Industrial Temperature)..... | +2.3V |
| V _{DDMIN} (Fosc \leq 16 MHz, Extended Temperature) | +2.3V |
| V _{DDMIN} (Fosc \leq 48 MHz) | +2.7V |
| V _{DDMAX} | +5.5V |

T_A — Operating Ambient Temperature Range

Industrial Temperature

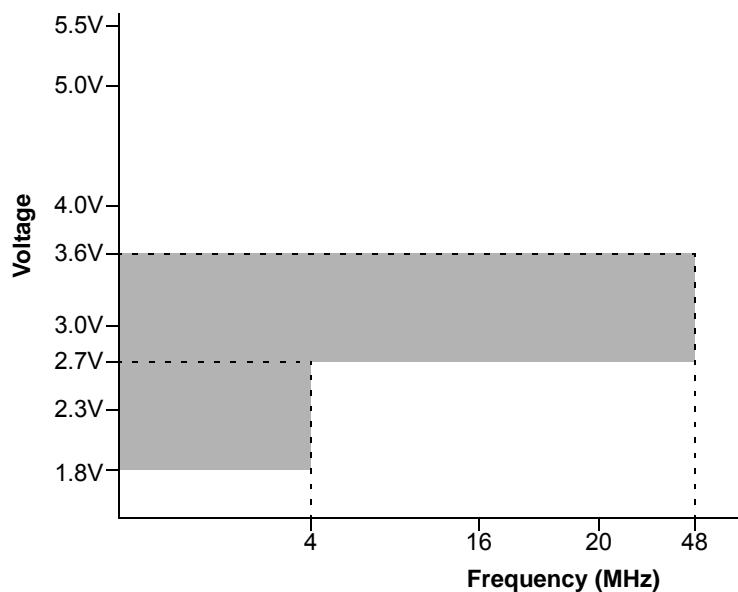
| | |
|--------------------------|-------|
| T _{A_MIN} | -40°C |
| T _{A_MAX} | +85°C |

Extended Temperature

| | |
|--------------------------|--------|
| T _{A_MIN} | -40°C |
| T _{A_MAX} | +125°C |

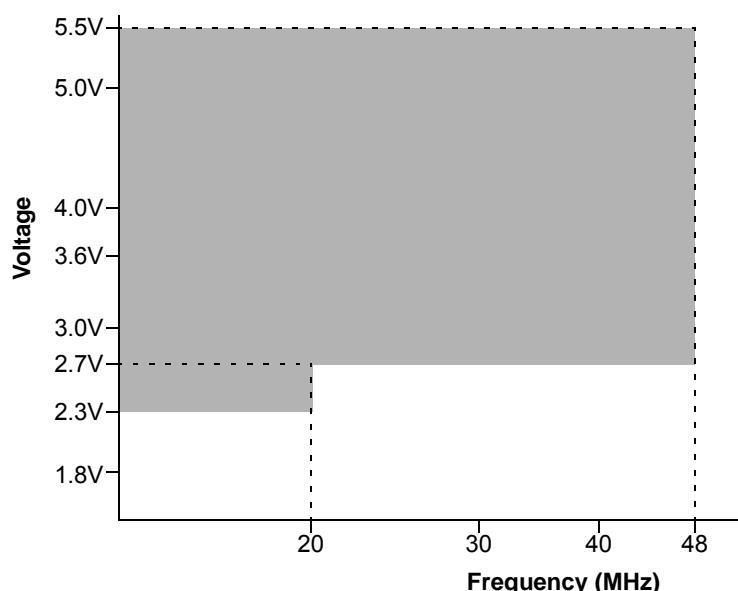
PIC18(L)F2X/45K50

FIGURE 29-1: PIC18LF2X/45K50 FAMILY VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL AND EXTENDED TEMPERATURE)



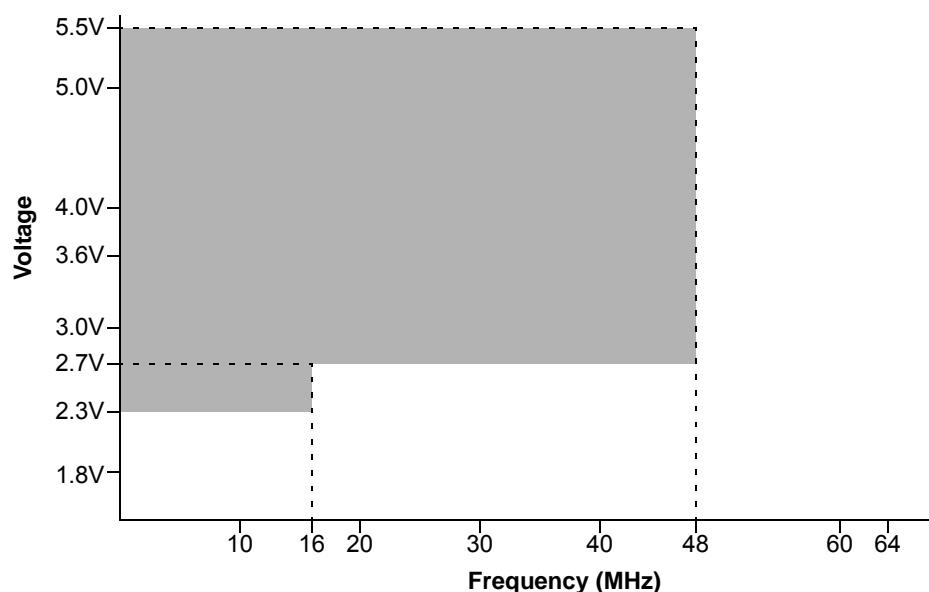
Note 1: Maximum Frequency 4 MHz, 1.8V to 2.7V, -40°C to +85°C
2: Maximum Frequency 48 MHz, 2.7V to 3.6V, -40°C to +85°C

FIGURE 29-2: PIC18F2X/45K50 FAMILY VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL TEMPERATURE)



Note 1: Maximum Frequency 20 MHz, 2.3V to 2.7V, -40°C to +85°C
2: Maximum Frequency 48 MHz, 2.7V to 5.5V, -40°C to +85°C

FIGURE 29-3: PIC18F2X/45K50 FAMILY VOLTAGE-FREQUENCY GRAPH (EXTENDED TEMPERATURE)



Note 1: Maximum Frequency 16 MHz, 2.3V to 2.7V, +85°C to +125°C

2: Maximum Frequency 48 MHz, 2.7V to 5.5V, +85°C to +125°C

PIC18(L)F2X/45K50

29.3 DC Characteristics

TABLE 29-1: SUPPLY VOLTAGE, PIC18(L)F2X/45K50

| PIC18(L)F2X/45K50 | | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|-------------------|---------|---|---|------|------|-------|---|--------------------|
| Param. No. | Symbol | Characteristic | Min. | Typ. | Max. | Units | Conditions | |
| D001 | VDD | Supply Voltage | PIC18LF2X/45K50 | 1.8 | — | 3.6 | V | Regulator disabled |
| | | | PIC18F2X/45K50 | 2.3 | — | 5.5 | V | Regulator enabled |
| D001B | VUSB3v3 | USB Supply Voltage | 3.0 | 3.3 | 3.6 | V | USB module enabled | |
| D001C D001D | | VUSB3v3 Capacitor Charging (PIC18F2X/45K50) | | — | 200 | — | mA | Note 4, 5 |
| | | Charging current | — | 0.0 | — | mA | Note 4 | |
| | | Source/sink capability when charging is complete | — | — | — | mA | Note 4 | |
| D002 | VDR | RAM Data Retention Voltage⁽¹⁾ | 1.5 | — | — | V | | |
| D003 | VPOR | VDD Start Voltage to ensure internal Power-on Reset signal | — | — | 0.7 | V | See section on Power-on Reset for details | |
| D004 | SVDD | VDD Rise Rate to ensure internal Power-on Reset signal | 0.05 | — | — | V/ms | See section on Power-on Reset for details | |
| D005 | VBOR | Brown-out Reset Voltage | | | | | | |
| | | BORV<1:0> = 11 ⁽²⁾ | 1.75 | 1.9 | 2.05 | V | | |
| | | BORV<1:0> = 10 | 2.05 | 2.2 | 2.35 | V | | |
| | | BORV<1:0> = 01 | 2.35 | 2.5 | 2.65 | V | | |
| | | BORV<1:0> = 00 ⁽³⁾ | 2.65 | 2.85 | 3.05 | V | | |
| D006 | VLPBOR | Low-Power Brown-out Reset (LPBOR) Voltage | 1.8V | — | 2.1 | V | | |

Note 1: This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

- 2:** On LF devices with BOR enabled, operation is supported until a BOR occurs. This is valid although VDD may be below the minimum rated supply voltage.
- 3:** With BOR enabled, full-speed operation (Fosc = 48 MHz) is supported until a BOR occurs. This is valid although VDD may be below the minimum voltage for this frequency.
- 4:** This is the inrush current associated with initial charging of the VUSB3v3 capacitor during a fast VDD ramp. The microcontroller can still start-up from VDD power sources that are limited to significantly less than this value.
- 5:** The VUSB3v3 regulator is only designed to supply the current requirements of the microcontroller and USB transceiver. It is not intended to supply external loads.

TABLE 29-2: POWER-DOWN CURRENT, PIC18(L)F2X/45K50

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | | |
|---|--|---|---------------|---------------|----------------|-------|------------|---|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | | |
| Param. No. | Device Characteristics | Typ. +25°C | Typ. +60°C | Max. +85°C | Max. +125°C | Units | Conditions | |
| | | VDD | | | | | Notes | |
| Power-down Base Current (IPD)⁽¹⁾ | | | | | | | | |
| D006 | Sleep mode | 0.01 | 0.04 | 2 | 10 | µA | 1.8V | WDT, BOR, FVR and SOSC disabled, all Peripherals inactive |
| | | 0.01 | 0.06 | 2 | 40 | µA | 3.0V | |
| | | 12 | 13 | 25 | 35 | µA | 2.3V | |
| | | 13 | 14 | 30 | 40 | µA | 3.0V | |
| | | 13 | 14 | 35 | 50 | µA | 5.0V | |
| Power-down Module Differential Current (delta IPD) | | | | | | | | |
| D007 | Watchdog Timer | 0.3 | 0.3 | 2.5 | 2.5 | µA | 1.8V | |
| | | 0.5 | 0.5 | 2.5 | 5 | µA | 3.0V | |
| | | 0.35 | 0.35 | 5.0 | 5.0 | µA | 2.3V | |
| | | 0.5 | 0.5 | 5.0 | 5.0 | µA | 3.0V | |
| | | 0.5 | 0.5 | 5.0 | 5.0 | µA | 5.0V | |
| D008 | Brown-out Reset ⁽²⁾ | 8 | 8.5 | 15 | 16 | µA | 2.0V | |
| | | 9 | 9.5 | 15 | 16 | µA | 3.0V | |
| | | 3.4 | 3.4 | 15 | 16 | µA | 2.3V | |
| | | 3.8 | 3.8 | 15 | 16 | µA | 3.0V | |
| | | 5.2 | 5.2 | 15 | 16 | µA | 5.0V | |
| D010 | High/Low Voltage Detect ⁽²⁾ | 6.5 | 6.7 | 15 | 15 | µA | 2.0V | |
| | | 7 | 7.5 | 15 | 15 | µA | 3.0V | |
| | | 2.1 | 2.1 | 15 | 15 | µA | 2.3V | |
| | | 2.4 | 2.4 | 15 | 15 | µA | 3.0V | |
| | | 3.2 | 3.2 | 15 | 15 | µA | 5.0V | |
| D011 | Secondary Oscillator | 0.5 | 1 | 3 | 10 | µA | 1.8V | 32 kHz on SOSC |
| | | 0.6 | 1.1 | 4 | 10 | µA | 3.0V | |
| | | 0.5 | 1 | 3 | 10 | µA | 2.3V | |
| | | 0.6 | 1.1 | 4 | 10 | µA | 3.0V | |
| | | 0.6 | 1.1 | 5 | 10 | µA | 5.0V | |

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or Vss and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

- 2:** On LF devices, the BOR, HLVD and FVR enable internal band gap reference. With more than one of these modules enabled, the current consumption will be less than the sum of the specifications. On F devices, the internal band gap reference is always enabled and its current consumption is included in the Power-down Base Current (IPD).
- 3:** A/D converter differential currents apply only in Run mode. In Sleep or Idle mode both the ADC and the FRC turn off as soon as conversion (if any) is complete.

PIC18(L)F2X/45K50

TABLE 29-2: POWER-DOWN CURRENT, PIC18(L)F2X/45K50 (CONTINUED)

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | | |
|-----------------|------------------------------|---|---------------|---------------|----------------|-------|------------|------------------------|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | | |
| Param. No. | Device Characteristics | Typ. +25°C | Typ. +60°C | Max. +85°C | Max. +125°C | Units | Conditions | |
| | | V _{DD} | | | | | Notes | |
| D015 | Comparators | 7 | 7 | 18 | 18 | µA | 1.8V | LP mode |
| | | 7 | 7 | 18 | 18 | µA | 3.0V | |
| | | 7 | 7 | 18 | 18 | µA | 2.3V | |
| | | 7 | 7 | 18 | 18 | µA | 3.0V | |
| | | 8 | 8 | 20 | 20 | µA | 5.0V | |
| D16 | Comparators | 38 | 38 | 95 | 95 | µA | 1.8V | HP mode |
| | | 40 | 40 | 105 | 105 | µA | 3.0V | |
| | | 39 | 39 | 95 | 95 | µA | 2.3V | |
| | | 40 | 40 | 105 | 105 | µA | 3.0V | |
| | | 40 | 40 | 105 | 105 | µA | 5.0V | |
| D017 | DAC | 12 | 12 | 22 | 25 | µA | 1.8V | |
| | | 20 | 20 | 35 | 35 | µA | 3.0V | |
| | | 15 | 15 | 30 | 30 | µA | 2.3V | |
| | | 20 | 20 | 35 | 35 | µA | 3.0V | |
| | | 32 | 32 | 60 | 60 | µA | 5.0V | |
| D018 | FVR ⁽²⁾ | 15 | 16 | 25 | 25 | µA | 1.8V | |
| | | 15 | 16 | 25 | 25 | µA | 3.0V | |
| | | 28 | 28 | 45 | 45 | µA | 2.3V | |
| | | 31 | 31 | 55 | 55 | µA | 3.0V | |
| | | 66 | 66 | 100 | 100 | µA | 5.0V | |
| D013 | A/D Converter ⁽³⁾ | 185 | 185 | 370 | 370 | µA | 1.8V | A/D on, not converting |
| | | 210 | 210 | 400 | 400 | µA | 3.0V | |
| | | 200 | 200 | 380 | 380 | µA | 2.3V | |
| | | 210 | 210 | 400 | 400 | µA | 3.0V | |
| | | 250 | 250 | 450 | 450 | µA | 5.0V | |

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to V_{DD} or V_{SS} and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

- 2:** On LF devices, the BOR, HLVD and FVR enable internal band gap reference. With more than one of these modules enabled, the current consumption will be less than the sum of the specifications. On F devices, the internal band gap reference is always enabled and its current consumption is included in the Power-down Base Current (IPD).
- 3:** A/D converter differential currents apply only in Run mode. In Sleep or Idle mode both the ADC and the FRC turn off as soon as conversion (if any) is complete.

TABLE 29-3: RC RUN SUPPLY CURRENT, PIC18(L)F2X/45K50

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | |
|-----------------|---|---|------|-------|------------|---|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | |
| D020 | Supply Current (IDD) ^{(1),(2)} | 3.6 | 23 | µA | -40°C | VDD = 1.8V Fosc = 31 kHz (RC_RUN mode, INTRC source) |
| | | 3.9 | 25 | µA | +25°C | |
| | | 3.9 | — | µA | +60°C | |
| | | 3.9 | 28 | µA | +85°C | |
| | | 4.0 | 30 | µA | +125°C | |
| D021 | | 8.1 | 26 | µA | -40°C | VDD = 3.0V |
| | | 8.4 | 30 | µA | +25°C | |
| | | 8.6 | — | µA | +60°C | |
| | | 8.7 | 35 | µA | +85°C | |
| | | 10.7 | 40 | µA | +125°C | |
| D022 | | 16 | 35 | µA | -40°C | VDD = 2.3V Fosc = 31 kHz (RC_RUN mode, INTRC source) |
| | | 17 | 35 | µA | +25°C | |
| | | 18 | 35 | µA | +85°C | |
| | | 19 | 50 | µA | +125°C | |
| D023 | | 18 | 50 | µA | -40°C | VDD = 3.0V |
| | | 20 | 50 | µA | +25°C | |
| | | 21 | 50 | µA | +85°C | |
| | | 22 | 60 | µA | +125°C | |
| D024 | | 19 | 55 | µA | -40°C | VDD = 5.0V |
| | | 21 | 55 | µA | +25°C | |
| | | 22 | 55 | µA | +85°C | |
| | | 23 | 70 | µA | +125°C | |

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0 and PMD1 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;

MCLR = VDD;

OSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).

PIC18(L)F2X/45K50

TABLE 29-3: RC RUN SUPPLY CURRENT, PIC18(L)F2X/45K50 (CONTINUED)

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|-----------------|------------------------|---|------|-------|-----------------|------------|---|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D030 | | 0.35 | 0.50 | mA | -40°C to +125°C | VDD = 1.8V | Fosc = 1 MHz (RC_RUN mode, HFINTOSC source) |
| D031 | | 0.45 | 0.65 | mA | -40°C to +125°C | VDD = 3.0V | |
| D032 | | 0.40 | 0.60 | mA | -40°C to +125°C | VDD = 2.3V | Fosc = 1 MHz (RC_RUN mode, HFINTOSC source) |
| D033 | | 0.50 | 0.65 | mA | -40°C to +125°C | VDD = 3.0V | |
| D034 | | 0.55 | 0.75 | mA | -40°C to +125°C | VDD = 5.0V | Fosc = 16 MHz (RC_RUN mode, HFINTOSC source) |
| D035 | | 1.3 | 2.0 | mA | -40°C to +125°C | VDD = 1.8V | |
| D036 | | 2.2 | 3.0 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 16 MHz (RC_RUN mode, HFINTOSC source) |
| D037 | | 1.7 | 2.0 | mA | -40°C to +125°C | VDD = 2.3V | |
| D038 | | 2.2 | 3.0 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 16 MHz (RC_RUN mode, HFINTOSC source) |
| D039 | | 2.5 | 3.5 | mA | -40°C to +125°C | VDD = 5.0V | |
| D041 | | 6.2 | 8.5 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 48 MHz (RC_RUN mode, HFINTOSC + PLL source) |
| D043 | | 6.2 | 8.5 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 48 MHz (RC_RUN mode, HFINTOSC + PLL source) |
| D044 | | 6.8 | 9.5 | mA | -40°C to +125°C | VDD = 5.0V | |

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0 and PMD1 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;

MCLR = VDD;

OSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).

TABLE 29-4: RC IDLE SUPPLY CURRENT, PIC18(L)F2X/45K50

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | |
|-----------------|---|---|------|-------|------------|--|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | |
| D045 | Supply Current (IDD) ^{(1),(2)} | 0.5 | 18 | µA | -40°C | VDD = 1.8V Fosc = 31 kHz (RC_IDLE mode, INTRC source) |
| | | 0.6 | 18 | µA | +25°C | |
| | | 0.7 | — | µA | +60°C | |
| | | 0.75 | 20 | µA | +85°C | |
| | | 2.3 | 22 | µA | +125°C | |
| D046 | | 1.1 | 20 | µA | -40°C | VDD = 3.0V |
| | | 1.2 | 20 | µA | +25°C | |
| | | 1.3 | — | µA | +60°C | |
| | | 1.4 | 22 | µA | +85°C | |
| | | 3.2 | 25 | µA | +125°C | |
| D047 | | 17 | 30 | µA | -40°C | VDD = 2.3V Fosc = 31 kHz (RC_IDLE mode, INTRC source) |
| | | 13 | 30 | µA | +25°C | |
| | | 14 | 30 | µA | +85°C | |
| | | 15 | 45 | µA | +125°C | |
| D048 | | 19 | 35 | µA | -40°C | VDD = 3.0V |
| | | 15 | 35 | µA | +25°C | |
| | | 16 | 35 | µA | +85°C | |
| | | 17 | 50 | µA | +125°C | |
| D049 | | 21 | 40 | µA | -40°C | VDD = 5.0V |
| | | 15 | 40 | µA | +25°C | |
| | | 16 | 40 | µA | +85°C | |
| | | 18 | 60 | µA | +125°C | |

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0 and PMD1 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;

MCLR = VDD;

OSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).

PIC18(L)F2X/45K50

TABLE 29-4: RC IDLE SUPPLY CURRENT, PIC18(L)F2X/45K50 (CONTINUED)

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|-----------------|------------------------|---|------|-------|-----------------|------------|---|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D055 | | 0.25 | 0.40 | mA | -40°C to +125°C | VDD = 1.8V | Fosc = 1 MHz (RC_IDLE mode, HFINTOSC source) |
| D056 | | 0.35 | 0.50 | mA | -40°C to +125°C | VDD = 3.0V | |
| D057 | | 0.30 | 0.45 | mA | -40°C to +125°C | VDD = 2.3V | Fosc = 1 MHz (RC_IDLE mode, HFINTOSC source) |
| D058 | | 0.40 | 0.50 | mA | -40°C to +125°C | VDD = 3.0V | |
| D059 | | 0.45 | 0.60 | mA | -40°C to +125°C | VDD = 5.0V | Fosc = 16 MHz (RC_IDLE mode, HFINTOSC source) |
| D060 | | 0.50 | 0.7 | mA | -40°C to +125°C | VDD = 1.8V | |
| D061 | | 0.80 | 1.1 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 16 MHz (RC_IDLE mode, HFINTOSC source) |
| D062 | | 0.65 | 1.0 | mA | -40°C to +125°C | VDD = 2.3V | |
| D063 | | 0.80 | 1.1 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 48 MHz (RC_IDLE mode, HFINTOSC + PLL source) |
| D064 | | 0.95 | 1.2 | mA | -40°C to +125°C | VDD = 5.0V | |
| D066 | | 2.5 | 3.5 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 48 MHz (RC_IDLE mode, HFINTOSC + PLL source) |
| D068 | | 2.5 | 3.5 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 48 MHz (RC_IDLE mode, HFINTOSC + PLL source) |
| D069 | | 3.0 | 4.5 | mA | -40°C to +125°C | VDD = 5.0V | |

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0 and PMD1 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;

MCLR = VDD;

OSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).

TABLE 29-5: PRIMARY RUN SUPPLY CURRENT, PIC18(L)F2X/45K50

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|-----------------|---|---|------|-------|-----------------|------------|---|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D070 | Supply Current (IDD) ^{(1),(2)} | 0.11 | 0.20 | mA | -40°C to +125°C | VDD = 1.8V | Fosc = 1 MHz (PRI_RUN mode, ECM source) |
| D071 | | 0.17 | 0.25 | mA | -40°C to +125°C | VDD = 3.0V | |
| D072 | | 0.15 | 0.25 | mA | -40°C to +125°C | VDD = 2.3V | Fosc = 1 MHz (PRI_RUN mode, ECM source) |
| D073 | | 0.20 | 0.30 | mA | -40°C to +125°C | VDD = 3.0V | |
| D074 | | 0.25 | 0.35 | mA | -40°C to +125°C | VDD = 5.0V | |
| D075 | | 1.45 | 2.0 | mA | -40°C to +125°C | VDD = 1.8V | Fosc = 20 MHz (PRI_RUN mode, ECH source) |
| D076 | | 2.60 | 3.5 | mA | -40°C to +125°C | VDD = 3.0V | |
| D077 | | 1.95 | 2.5 | mA | -40°C to +125°C | VDD = 2.3V | Fosc = 20 MHz (PRI_RUN mode, ECH source) |
| D078 | | 2.65 | 3.5 | mA | -40°C to +125°C | VDD = 3.0V | |
| D079 | | 2.95 | 4.5 | mA | -40°C to +125°C | VDD = 5.0V | |
| D080 | | 7.5 | 10 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 48 MHz (PRI_RUN, ECH oscillator) |
| D081 | | 7.5 | 10 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 48 MHz (PRI_RUN mode, ECH source) |
| D082 | | 8.5 | 11.5 | mA | -40°C to +125°C | VDD = 5.0V | |
| D083 | | 1.0 | 1.5 | mA | -40°C to +125°C | VDD = 1.8V | Fosc = 4 MHz 16 MHz Internal (PRI_RUN mode, ECM + PLL source) |
| D084 | | 1.8 | 3.0 | mA | -40°C to +125°C | VDD = 3.0V | |
| D085 | | 1.4 | 2.0 | mA | -40°C to +125°C | VDD = 2.3V | Fosc = 4 MHz 16 MHz Internal (PRI_RUN mode, ECM + PLL source) |
| D086 | | 1.85 | 2.5 | mA | -40°C to +125°C | VDD = 3.0V | |
| D087 | | 2.1 | 3.0 | mA | -40°C to +125°C | VDD = 5.0V | |
| D088 | | 6.35 | 9.0 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 16 MHz 48 MHz Internal (PRI_RUN mode, ECH + PLL source) |
| D089 | | 6.35 | 9.0 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 16 MHz 48 MHz Internal (PRI_RUN mode, ECH + PLL source) |
| D090 | | 7.0 | 10 | mA | -40°C to +125°C | VDD = 5.0V | |

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0 and PMD1 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;

MCLR = VDD;

OSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).

PIC18(L)F2X/45K50

TABLE 29-6: PRIMARY IDLE SUPPLY CURRENT, PIC18(L)F2X/45K50

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|-----------------|--|---|-------|-------|-----------------|------------|---|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D100 | Supply Current (I_{DD}) ^{(1),(2)} | 0.030 | 0.050 | mA | -40°C to +125°C | VDD = 1.8V | Fosc = 1 MHz (PRI_IDLE mode, ECM source) |
| D101 | | 0.045 | 0.065 | mA | -40°C to +125°C | VDD = 3.0V | |
| D102 | | 0.06 | 0.12 | mA | -40°C to +125°C | VDD = 2.3V | |
| D103 | | 0.08 | 0.15 | mA | -40°C to +125°C | VDD = 3.0V | |
| D104 | | 0.13 | 0.20 | mA | -40°C to +125°C | VDD = 5.0V | Fosc = 1 MHz (PRI_IDLE mode, ECM source) |
| D105 | | 0.45 | 0.8 | mA | -40°C to +125°C | VDD = 1.8V | |
| D106 | | 0.70 | 1.0 | mA | -40°C to +125°C | VDD = 3.0V | |
| D107 | | 0.55 | 0.8 | mA | -40°C to +125°C | VDD = 2.3V | |
| D108 | | 0.75 | 1.0 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 20 MHz (PRI_IDLE mode, ECH source) |
| D109 | | 0.90 | 1.2 | mA | -40°C to +125°C | VDD = 5.0V | |
| D110 | | 2.25 | 3.0 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 48 MHz (PRI_IDLE mode, ECH source) |
| D111 | | 2.25 | 3.0 | mA | -40°C to +125°C | VDD = 3.0V | Fosc = 48 MHz (PRI_IDLE mode, ECH source) |
| D112 | | 2.60 | 3.5 | mA | -40°C to +125°C | VDD = 5.0V | |
| D113 | | 0.35 | 0.6 | mA | -40°C to +125°C | VDD = 1.8V | Fosc = 4 MHz 16 MHz Internal (PRI_IDLE mode, ECM + PLL source) |
| D114 | | 0.55 | 0.8 | mA | -40°C to +125°C | VDD = 3.0V | |
| D115 | | 0.45 | 0.6 | mA | -40°C to +125°C | VDD = 2.3V | |
| D116 | | 0.60 | 0.9 | mA | -40°C to +125°C | VDD = 3.0V | |
| D117 | | 0.70 | 1.0 | mA | -40°C to +125°C | VDD = 5.0V | Fosc = 4 MHz 16 MHz Internal (PRI_IDLE mode, ECM + PLL source) |
| D118 | | 2.2 | 3.0 | mA | -40°C to +125°C | VDD = 3.0V | |
| D119 | | 2.2 | 3.0 | mA | -40°C to +125°C | VDD = 3.0V | |
| D120 | | 2.5 | 3.5 | mA | -40°C to +125°C | VDD = 5.0V | |

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0 and PMD1 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;

MCLR = VDD;

OSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).

TABLE 29-7: SECONDARY OSCILLATOR SUPPLY CURRENT, PIC18(L)F2X/45K50

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|-----------------|---|---|------|-------|------------|------------|--|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | | |
| D130 | Supply Current (IDD) ^{(1),(2)} | 3.5 | 23 | µA | -40°C | VDD = 1.8V | Fosc = 32 kHz (SEC_RUN mode, SOSC source) |
| | | 3.7 | 25 | µA | +25°C | | |
| | | 3.8 | — | µA | +60°C | | |
| | | 4.0 | 28 | µA | +85°C | | |
| | | 5.1 | 30 | µA | +125°C | | |
| D131 | | 6.2 | 26 | µA | -40°C | VDD = 3.0V | |
| | | 6.4 | 30 | µA | +25°C | | |
| | | 6.5 | — | µA | +60°C | | |
| | | 6.8 | 35 | µA | +85°C | | |
| | | 7.8 | 40 | µA | +125°C | | |
| D132 | | 15 | 35 | µA | -40°C | VDD = 2.3V | Fosc = 32 kHz (SEC_RUN mode, SOSC source) |
| | | 16 | 35 | µA | +25°C | | |
| | | 17 | 35 | µA | +85°C | | |
| | | 19 | 50 | µA | +125°C | | |
| D133 | | 18 | 50 | µA | -40°C | VDD = 3.0V | |
| | | 19 | 50 | µA | +25°C | | |
| | | 21 | 50 | µA | +85°C | | |
| | | 22 | 60 | µA | +125°C | | |
| D134 | | 19 | 55 | µA | -40°C | VDD = 5.0V | |
| | | 20 | 55 | µA | +25°C | | |
| | | 22 | 55 | µA | +85°C | | |
| | | 23 | 70 | µA | +125°C | | |

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0 and PMD1 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;

MCLR = VDD;

SOSCI / SOSCO = complementary external square wave, from rail-to-rail.

PIC18(L)F2X/45K50

TABLE 29-7: SECONDARY OSCILLATOR SUPPLY CURRENT, PIC18(L)F2X/45K50 (CONTINUED)

| PIC18LF2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | |
|-----------------|------------------------|---|------|-------|------------|--|
| PIC18F2X/45K50 | | Standard Operating Conditions (unless otherwise stated) | | | | |
| Param. No. | Device Characteristics | Typ. | Max. | Units | Conditions | |
| D135 | | 0.9 | 18 | µA | -40°C | VDD = 1.8V Fosc = 32 kHz (SEC_IDLE mode, SOSC source) |
| | | 1.0 | 18 | µA | +25°C | |
| | | 1.1 | — | µA | +60°C | |
| | | 1.3 | 20 | µA | +85°C | |
| | | 2.3 | 22 | µA | +125°C | |
| D136 | | 1.3 | 20 | µA | -40°C | VDD = 3.0V |
| | | 1.4 | 20 | µA | +25°C | |
| | | 1.5 | — | µA | +60°C | |
| | | 1.8 | 22 | µA | +85°C | |
| | | 2.9 | 25 | µA | +125°C | |
| D137 | | 12 | 30 | µA | -40°C | VDD = 2.3V |
| | | 13 | 30 | µA | +25°C | |
| | | 14 | 30 | µA | +85°C | |
| | | 16 | 45 | µA | +125°C | |
| D138 | | 13 | 35 | µA | -40°C | VDD = 3.0V |
| | | 14 | 35 | µA | +25°C | |
| | | 16 | 35 | µA | +85°C | |
| | | 18 | 50 | µA | +125°C | |
| D139 | | 14 | 40 | µA | -40°C | VDD = 5.0V |
| | | 15 | 40 | µA | +25°C | |
| | | 16 | 40 | µA | +85°C | |
| | | 18 | 60 | µA | +125°C | |

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0 and PMD1 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;

MCLR = VDD;

SOSCI / SOSCO = complementary external square wave, from rail-to-rail.

TABLE 29-8: INPUT/OUTPUT CHARACTERISTICS, PIC18(L)F2X/45K50

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|--------------|---|---|-------------------|----------|-------|--|--|--|--|--|
| Param. No. | Symbol | Characteristic | Min. | Typ. [†] | Max. | Units | Conditions | | | | |
| D140 D140A D141 D142 D142A | VIL | Input Low Voltage | | | | | | | | | |
| | | I/O PORT: | | | | | | | | | |
| | | with TTL buffer | — | — | 0.8 | V | $4.5V \leq VDD \leq 5.5V$ | | | | |
| | | | — | — | 0.15 VDD | V | $1.8V \leq VDD \leq 4.5V$ | | | | |
| | | with Schmitt Trigger buffer | — | — | 0.2 VDD | V | $2.0V \leq VDD \leq 5.5V$ | | | | |
| | | with I ² C™ levels | — | — | 0.3 VDD | V | | | | | |
| | | with SMBus levels | — | — | 0.8 | V | $2.7V \leq VDD \leq 5.5V$ | | | | |
| | | MCLR, OSC1 (RC mode) ⁽¹⁾ | — | — | 0.2 VDD | V | | | | | |
| | | OSC1 (HS mode) | — | — | 0.3 VDD | V | | | | | |
| D147 D147A D148 D149 D150A D150B | VIH | Input High Voltage | | | | | | | | | |
| | | I/O ports: | | | | | | | | | |
| | | with TTL buffer | 2.0 | — | — | V | $4.5V \leq VDD \leq 5.5V$ | | | | |
| | | | 0.25 VDD + 0.8 | — | — | V | $1.8V \leq VDD \leq 4.5V$ | | | | |
| | | with Schmitt Trigger buffer | 0.8 VDD | — | — | V | $2.0V \leq VDD \leq 5.5V$ | | | | |
| | | with I ² C levels | 0.7 VDD | — | — | V | | | | | |
| | | with SMBus levels | 2.1 | — | — | V | $2.7V \leq VDD \leq 5.5V$ | | | | |
| | | MCLR | 0.8 VDD | — | — | V | | | | | |
| | | OSC1 (HS mode) | 0.7 VDD | — | — | V | | | | | |
| D155 | IIL | Input Leakage I/O and MCLR^{(2),(3)} | | | | | | | | | |
| | | I/O ports and MCLR | — | 0.1 | 50 | nA | $VSS \leq VPIN \leq VDD$, Pin at high-impedance | | | | |
| | | | — | 0.7 | 100 | nA | $\leq +25^{\circ}C^{(4)}$ | | | | |
| D158 | IPU IPURB | Weak Pull-up Current⁽⁴⁾ | — | 4 | 200 | nA | $+60^{\circ}C$ | | | | |
| | | PORTB weak pull-up current | 25 | 85 | 200 | μA | $+85^{\circ}C$ | | | | |
| | | | 25 | 130 | 300 | μA | $VDD = 3.3V, VPIN = VSS$ | | | | |

Note 1: In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC® device be driven with an external clock while in RC mode.

- 2:** The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.
- 3:** Negative current is defined as current sourced by the pin.
- 4:** Parameter is characterized but not tested.

PIC18(L)F2X/45K50

TABLE 29-8: INPUT/OUTPUT CHARACTERISTICS, PIC18(L)F2X/45K50 (CONTINUED)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) | | | | |
|--------------------|---------------------|---|---|-------------------|------|-------|--|
| Param. No. | Symbol | Characteristic | Min. | Typ. [†] | Max. | Units | Conditions |
| D159 | V _{OL} | Output Low Voltage I/O ports | — | — | 0.6 | V | I _{OL} = 8 mA, V _{DD} = 5V I _{OL} = 6 mA, V _{DD} = 3.3V I _{OL} = 1.8 mA, V _{DD} = 1.8V |
| D160 | I _{PUMCLR} | MCLR and ICRST weak pull-up current | 25 | 85 | 200 | μA | V _{DD} = 3.3V, V _{PIN} = V _{SS} |
| | | | 25 | 130 | 300 | μA | V _{DD} = 5.0V, V _{PIN} = V _{SS} |
| D161 | V _{OH} | Output High Voltage⁽³⁾ I/O ports | V _{DD} - 0.7 | — | — | V | I _{OH} = 3.5 mA, V _{DD} = 5V I _{OH} = 3 mA, V _{DD} = 3.3V I _{OH} = 1 mA, V _{DD} = 1.8V |

- Note**
- 1: In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC® device be driven with an external clock while in RC mode.
 - 2: The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.
 - 3: Negative current is defined as current sourced by the pin.
 - 4: Parameter is characterized but not tested.

TABLE 29-9: MEMORY PROGRAMMING REQUIREMENTS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|-------|---|--------|-------------------|--------|-------|---|
| Param. No. | Sym. | Characteristic | Min. | Typ. [†] | Max. | Units | Conditions |
| D170 | VPP | Internal Program Memory Programming Specifications⁽¹⁾ | 8 | — | 9 | V | (Note 3), (Note 4) |
| D171 | IDDP | Voltage on MCLR/VPP pin Supply Current during Programming | — | — | 10 | mA | |
| Data EEPROM Memory | | | | | | | |
| D172 | ED | Byte Endurance | 100K | — | — | E/W | -40°C to +85°C |
| D173 | VDRW | VDD for Read/Write | VDDMIN | — | VDDMAX | V | Using EECON to read/write |
| D175 | TDEW | Erase/Write Cycle Time | — | 3 | 4 | ms | |
| D176 | TRETD | Characteristic Retention | — | 40 | — | Year | Provided no other specifications are violated |
| D177 | TREF | Number of Total Erase/Write Cycles before Refresh ⁽²⁾ | 1M | 10M | — | E/W | -40°C to +85°C |
| Program Flash Memory | | | | | | | |
| D178 | EP | Cell Endurance | 10K | — | — | E/W | -40°C to +85°C (Note 5) |
| D179 | VPR | VDD for Read | VDDMIN | — | VDDMAX | V | |
| D181 | VIW | VDD for Row Erase or Write | 2.2 | — | VDDMAX | V | PIC18LF2X/45K50 |
| D182 | VIW | | VDDMIN | — | VDDMAX | V | PIC18F2X/45K50 |
| D183 | TIW | Self-timed Write Cycle Time | — | 2 | — | ms | |
| D184 | 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 8.8 “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.

5: Self-write and Block Erase.

PIC18(L)F2X/45K50

TABLE 29-10: USB MODULE SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|-------------------|--|------|------|------|-------|--|
| Param. No. | Sym. | Characteristic | Min. | Typ. | Max. | Units | Conditions |
| D313 | V _{USB} | USB Voltage | 3.0 | — | 3.6 | V | Voltage on V _{USB3V3} pin must be in this range for proper USB operation |
| D314 | I _{IL} | Input Leakage on pin | — | — | ± 1 | µA | V _{SS} ≤ V _{PIN} ≤ V _{DD} pin at high-impedance |
| D315 | V _{LUSB} | Input Low Voltage for USB Buffer | — | — | 0.8 | V | For V _{USB3V3} range |
| D316 | V _{HUSB} | Input High Voltage for USB Buffer | 2.0 | — | — | V | For V _{USB3V3} range |
| D318 | V _{DIFS} | Differential Input Sensitivity | — | — | 0.2 | V | The difference between D+ and D- must exceed this value while V _{CM} is met |
| D319 | V _{CM} | Differential Common Mode Range | 0.8 | — | 2.5 | V | |
| D320 | Z _{OUT} | Driver Output Impedance ⁽¹⁾ | 28 | — | 44 | Ω | |
| D321 | V _{OL} | Voltage Output Low | 0.0 | — | 0.3 | V | 1.5 kΩ load connected to 3.6V |
| D322 | V _{OH} | Voltage Output High | 2.8 | — | 3.6 | V | 1.5 kΩ load connected to ground |
| D323 | C _{USB} | V _{USB} Capacitor Value | 0.33 | 0.47 | 8 | µF | |

Note 1: The D+ and D- signal lines have been built-in impedance matching resistors. No external resistors, capacitors or magnetic components are necessary on the D+/D- signal paths between the PIC18(L)F2X/45K50 family device and USB cable.

29.4 Analog Characteristics

TABLE 29-11: COMPARATOR SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|--------|--|------|------|------|-------|---------------------------------|
| Param. No. | Sym. | Characteristics | Min. | Typ. | Max. | Units | Comments |
| CM01 | VIOFF | Input Offset Voltage | — | 3 | 30 | mV | High-Power mode VREF = VDD/2 |
| | | | — | 4 | 40 | mV | Low-Power mode VREF = VDD/2 |
| CM02 | VICM | Input Common-mode Voltage | Vss | — | VDD | V | |
| CM04* | TRESP | Response Time ⁽¹⁾ | — | 200 | 400 | ns | High-Power mode |
| | | | — | 600 | 3500 | 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 29-12: DIGITAL-TO-ANALOG CONVERTER (DAC) SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) 2.0V < VDD < 5.5V, -40°C < TA < +125°C | | | | | | | |
|---|-------|-------------------------------------|-----------|--------|-----------|-------|--------------|
| Param. No. | Sym. | Characteristics | Min. | Typ. | Max. | Units | Comments |
| CV01* | CLSB | Step Size ⁽²⁾ | — | VDD/32 | — | V | |
| CV02* | CACC | Absolute Accuracy | — | — | ±1/2 | LSb | ΔVSRC ≥ 2.0V |
| CV03* | CR | Unit Resistor Value (R) | — | 5k | — | Ω | |
| CV04* | CST | Settling Time ⁽¹⁾ | — | — | 10 | μs | |
| CV05* | VSRC+ | DAC Positive Reference | VSRC- + 2 | — | VDD | V | |
| CV06* | VSRC- | DAC Negative Reference | Vss | — | VSRC+ – 2 | V | |
| CV07* | ΔVSRC | DAC Reference Range (VSRC+ - VSRC-) | 2 | — | VDD | V | |

* These parameters are characterized but not tested.

Note 1: Settling time measured while CVRR = 1 and CVR3:CVR0 transitions from '0000' to '1111'.

2: See [Section 23.0 “Digital-to-Analog Converter \(DAC\) Module”](#) for more information.

PIC18(L)F2X/45K50

TABLE 29-13: FIXED VOLTAGE REFERENCE (FVR) SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---------|-------------------------------------|-------|-------|-------|---------|--|
| Param. No. | Sym. | Characteristics | Min. | Typ. | Max. | Units | Comments |
| VR01 | VROUT | VR voltage output to ADC | 0.973 | 1.024 | 1.085 | V | 1x output, VDD \geq 2.5V |
| | | | 1.946 | 2.048 | 2.171 | V | 2x output, VDD \geq 2.5V |
| | | | 3.891 | 4.096 | 4.342 | V | 4x output, VDD \geq 4.75V (PIC18F2X/45K50) |
| VR02 | VROUT | VR voltage output all other modules | 0.942 | 1.024 | 1.096 | V | 1x output, VDD \geq 2.5V |
| | | | 1.884 | 2.048 | 2.191 | V | 2x output, VDD \geq 2.5V |
| | | | 3.768 | 4.096 | 4.383 | V | 4x output, VDD \geq 4.75V (PIC18F2X/45K50) |
| VR04* | TSTABLE | Settling Time | — | 25 | 100 | μ s | 0 to 85°C |

* These parameters are characterized but not tested.

TABLE 29-14: CHARGE TIME MEASUREMENT UNIT (CTMU) SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|-------|---------------------------------|------|---------------------|------|---------|-----------------------------------|
| Param. No. | Sym. | Characteristics | Min. | Typ. ⁽¹⁾ | Max. | Units | Comments |
| CT01 | IOUT1 | CTMU Current Source, Base Range | — | 0.55 | — | μ A | IRNG<1:0> = 01 |
| CT02 | IOUT2 | CTMU Current Source, 10X Range | — | 5.5 | — | μ A | IRNG<1:0> = 10 |
| CT03 | IOUT3 | CTMU Current Source, 100X Range | — | 55 | — | μ A | IRNG<1:0> = 11 VDD \geq 3.0V |

Note 1: Nominal value at center point of current trim range (CTMUICON<7:2> = 000000).

FIGURE 29-4: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

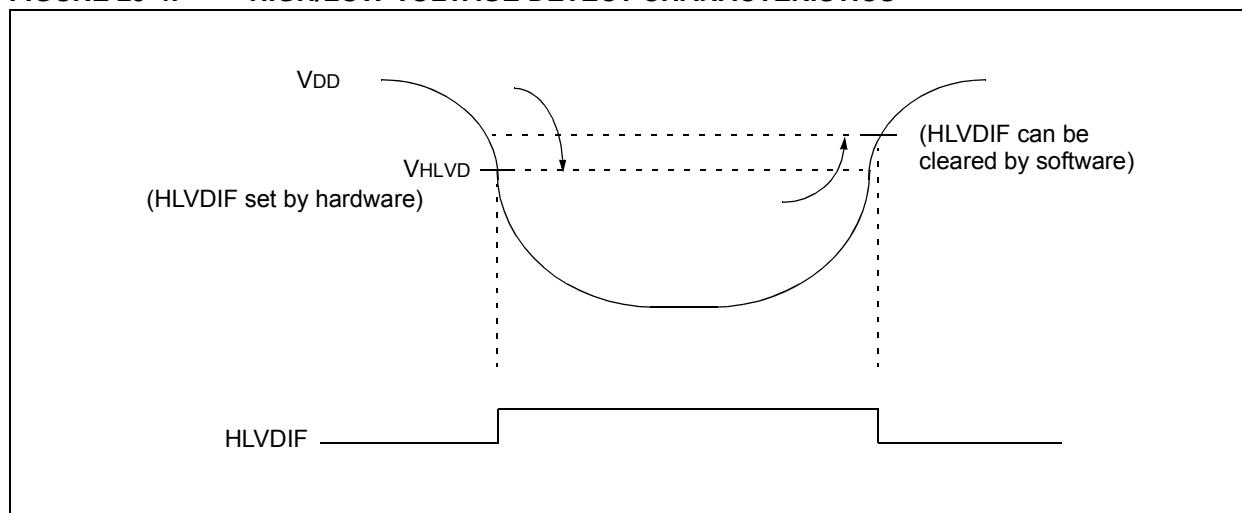


TABLE 29-15: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|--------|---|------------|---------------|-------|------|-------|------------|
| Param. No. | Symbol | Characteristic | HLVDL<3:0> | Min. | Typ.† | Max. | Units | Conditions |
| D420 | — | HLVD Voltage on VDD Transition High-to-Low | 0000 | 1.69 | 1.84 | 1.99 | V | |
| | | | 0001 | 1.92 | 2.07 | 2.22 | V | |
| | | | 0010 | 2.08 | 2.28 | 2.48 | V | |
| | | | 0011 | 2.24 | 2.44 | 2.64 | V | |
| | | | 0100 | 2.34 | 2.54 | 2.74 | V | |
| | | | 0101 | 2.54 | 2.74 | 2.94 | V | |
| | | | 0110 | 2.62 | 2.87 | 3.12 | V | |
| | | | 0111 | 2.76 | 3.01 | 3.26 | V | |
| | | | 1000 | 3.00 | 3.30 | 3.60 | V | |
| | | | 1001 | 3.18 | 3.48 | 3.78 | V | |
| | | | 1010 | 3.44 | 3.69 | 3.94 | V | |
| | | | 1011 | 3.66 | 3.91 | 4.16 | V | |
| | | | 1100 | 3.90 | 4.15 | 4.40 | V | |
| | | | 1101 | 4.11 | 4.41 | 4.71 | V | |
| | | | 1110 | 4.39 | 4.74 | 5.09 | V | |
| | | | 1111 | V(HLVDIN pin) | | | v | |

† Production tested at TAMB = +25°C. Specifications over temperature limits ensured by characterization.

PIC18(L)F2X/45K50

29.5 AC (Timing) Characteristics

29.5.1 TIMING PARAMETER SYMOLOGY

The timing parameter symbols have been created using one of the following formats:

- | | |
|-------------|--|
| 1. TppS2ppS | 3. TCC:ST (I^2C ™ specifications only) |
| 2. TppS | 4. Ts (I^2C specifications only) |

| | |
|------------------|-------------|
| T | |
| F Frequency | T Time |

Lowercase letters (pp) and their meanings:

| pp | | | |
|----|------------------------|-----|---|
| cc | CCP1 | osc | OSC1 |
| ck | CLKO | rd | $\overline{\text{RD}}$ |
| cs | $\overline{\text{CS}}$ | rw | $\overline{\text{RD}} \text{ or } \overline{\text{WR}}$ |
| di | SDI | sc | SCK |
| do | SDO | ss | $\overline{\text{SS}}$ |
| dt | Data in | t0 | T0CKI |
| io | I/O port | t1 | T13CKI |
| mc | MCLR | wr | $\overline{\text{WR}}$ |

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 |
| I^2C only | | High | High |
| AA | output access | Low | Low |
| BUF | Bus free | | |

Tcc:ST (I^2C specifications only)

| CC | | | |
|-----|-----------------|-----|----------------|
| HD | Hold | SU | Setup |
| ST | | STO | Stop condition |
| DAT | DATA input hold | | |
| STA | Start condition | | |

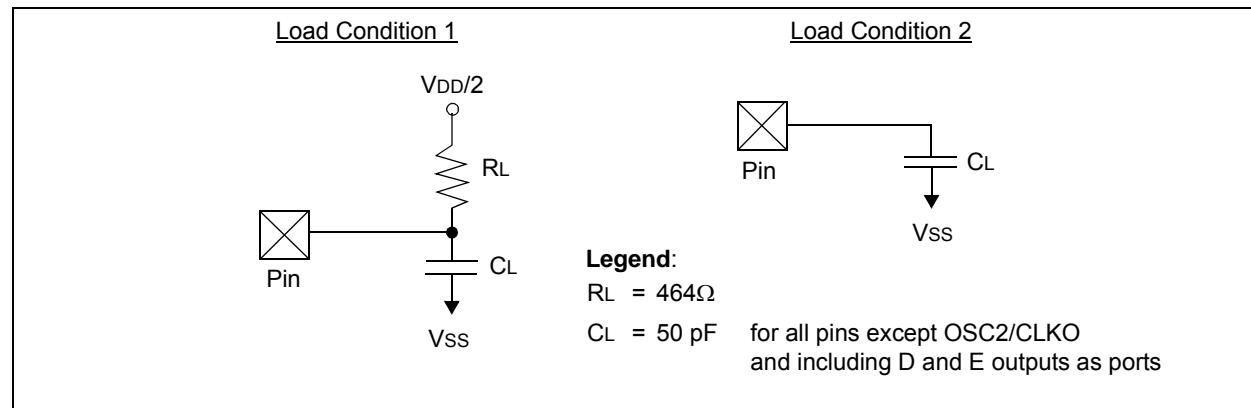
29.5.2 TIMING CONDITIONS

The temperature and voltages specified in [Table 29-16](#) apply to all timing specifications unless otherwise noted. [Figure 29-5](#) specifies the Load conditions for the timing specifications.

TABLE 29-16: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC

| AC CHARACTERISTICS | Standard Operating Conditions (unless otherwise stated) Operating voltage VDD range as described in Table 29-1 and Table 29-9 . |
|--------------------|---|
|--------------------|---|

FIGURE 29-5: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS



PIC18(L)F2X/45K50

29.5.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 29-6: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)

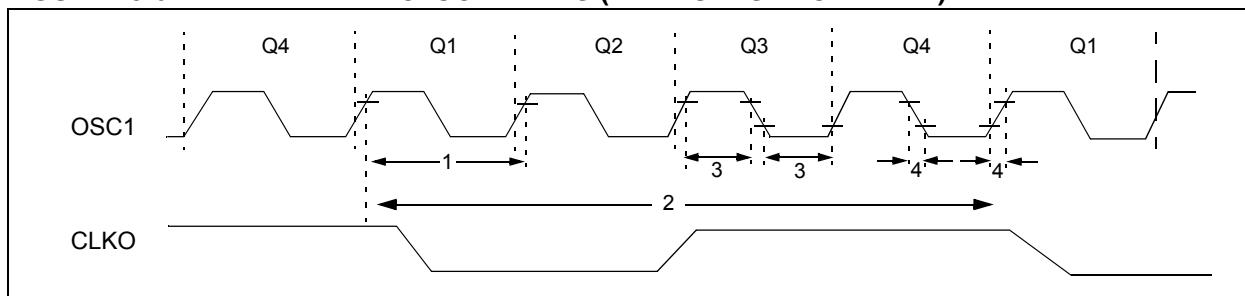


TABLE 29-17: EXTERNAL CLOCK TIMING REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|------------|------------|--|---------------------------------------|-------------------------------------|--|--|
| 1A | Fosc | External CLKIN Frequency ⁽¹⁾ | DC DC DC | 4 16 48 | MHz MHz MHz | EC, ECIO Oscillator mode (low power) EC, ECIO Oscillator mode (medium power) EC, ECIO Oscillator mode (high power) |
| | | Oscillator Frequency ⁽¹⁾ | DC 5 0.1 4 4 4 | 4 200 4 4 16 20 | MHz kHz MHz MHz MHz MHz | RC Oscillator mode LP Oscillator mode XT Oscillator mode HS Oscillator mode, VDD < 2.7V HS Oscillator mode, VDD ≥ 2.7V, Medium-Power mode (HSMP) HS Oscillator mode, VDD ≥ 2.7V, High-Power mode (HSHP) |
| 1 | Tosc | External CLKIN Period ⁽¹⁾ | 0.25 62.5 20.8 | — — — | μs ns ns | EC, ECIO Oscillator mode (low power) EC, ECIO Oscillator mode (medium power) EC, ECIO Oscillator mode (high power) |
| | | Oscillator Period ⁽¹⁾ | 250 5 0.25 250 62.5 50 | — 200 10 250 250 250 | ns μs μs ns ns ns | RC Oscillator mode LP Oscillator mode XT Oscillator mode HS Oscillator mode, VDD < 2.7V HS Oscillator mode, VDD ≥ 2.7V, Medium-Power mode (HSMP) HS Oscillator mode, VDD ≥ 2.7V, High-Power mode (HSHP) |
| 2 | Tcy | Instruction Cycle Time ⁽¹⁾ | 83.3 | — | ns | Tcy = 4/Fosc |
| 3 | TosL, TosH | External Clock in (OSC1) High or Low Time | 2.5 30 10 | — — — | μs ns ns | LP Oscillator mode XT Oscillator mode HS Oscillator mode |
| 4 | TosR, TosF | External Clock in (OSC1) Rise or Fall Time | — — — | 50 20 7.5 | ns ns ns | LP Oscillator mode XT Oscillator mode 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 29-18: PLL CLOCK TIMING SPECIFICATIONS

| Param. No. | Sym. | Characteristic | Min. | Max. | Units | Conditions |
|------------|-----------------|----------------------------------|------|------|-------|-------------------------------|
| F10 | FOSC | 4xPLL Oscillator Frequency Range | 4 | 5 | MHz | VDD < 2.7V, -40°C to +85°C |
| | | | 4 | 12 | MHz | 2.7V ≤ VDD, -40°C to +85°C |
| F10B | FOSC | 3xPLL Oscillator Frequency Range | 4 | 4 | MHz | 2.7V ≤ VDD, -40°C to +85°C |
| F11 | FSYS | On-Chip VCO System Frequency | 16 | 20 | MHz | VDD < 2.7V, -40°C to +85°C |
| | | | 16 | 48 | MHz | 2.7V ≤ VDD, -40°C to +85°C |
| F12 | t _{rc} | PLL Start-up Time (Lock Time) | — | 2 | ms | |

TABLE 29-19: INTERNAL OSCILLATORS ACCURACY (PIC18(L)F2X/45K50)

| Standard Operating Conditions (unless otherwise stated) | | | | | | |
|---|--|-------|---------|-------|-----------------|--|
| Param. No. | Characteristic | Min. | Typ. | Max. | Units | Conditions |
| OA1 | HF-INTOSC Accuracy⁽¹⁾ | | | | | |
| | -2 | ±1 | +2 | % | +0°C to +70°C | |
| | -3 | — | +2 | % | +70°C to +85°C | |
| | -5 | — | +5 | % | -40°C to +125°C | |
| OA1B | HF-INTOSC Accuracy with Active Clock Tuning (ACT) | | | | | |
| | | -0.20 | ±0.05 | +0.20 | % | -40°C to +85°C ⁽²⁾ , Active Clock Tune is enabled and locked. |
| OA1C | OSCTUNE Step Size | — | 0.1 | — | % | |
| OA2 | INTRC Accuracy @ Freq = 31.25 kHz | | | | | |
| | 26.5625 | — | 35.9375 | kHz | -40°C to +85°C | |
| | 25 | — | 37.2 | kHz | +85°C to +125°C | |

Note 1: Frequency calibrated at 25°C. OSCTUNE register can be used to compensate for temperature drift.

2: Accuracy measured with respect to reference source.

PIC18(L)F2X/45K50

FIGURE 29-7: CLKO AND I/O TIMING

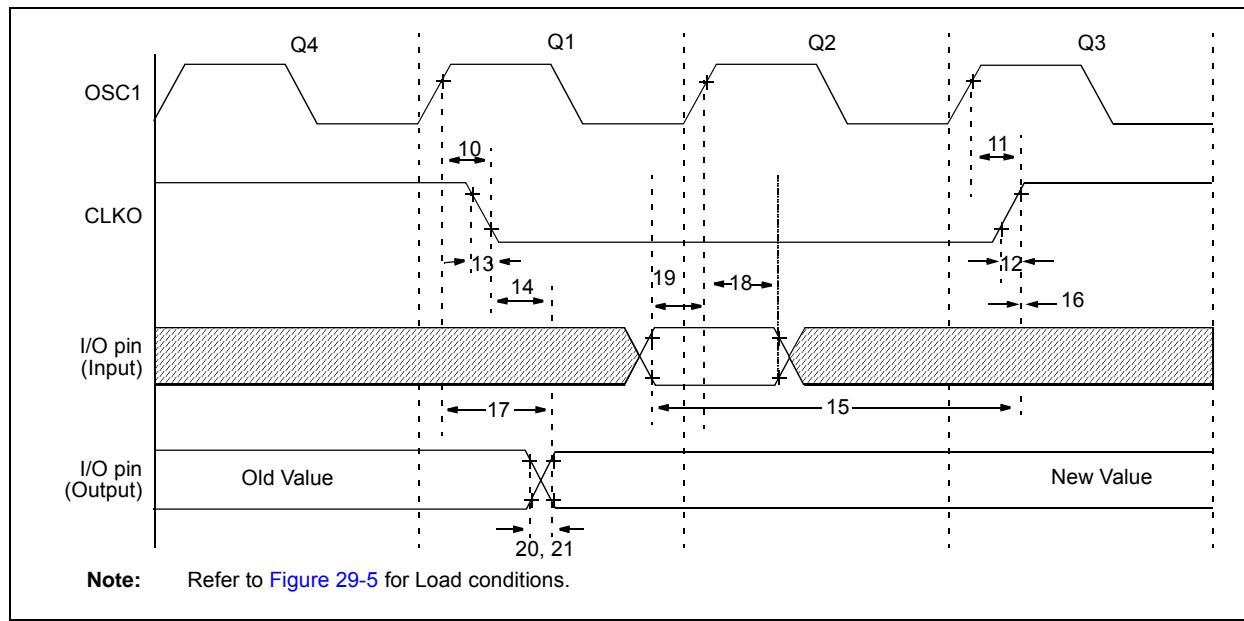


TABLE 29-20: CLKO AND I/O TIMING REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min. | Typ. | Max. | Units | Conditions |
|------------|----------|---|---------------|----------|--------------|----------|---------------------------------|
| 10 | TosH2ckL | OSC1 \uparrow to CLKO \downarrow | — | 75 | 200 | ns | Note 1 |
| 11 | TosH2ckH | OSC1 \uparrow to CLKO \uparrow | — | 75 | 200 | ns | Note 1 |
| 12 | TckR | CLKO Rise Time | — | 35 | 100 | ns | Note 1 |
| 13 | TckF | CLKO Fall Time | — | 35 | 100 | ns | Note 1 |
| 14 | TckL2ioV | CLKO \downarrow to Port Out Valid | — | — | 0.5 TCY + 20 | ns | Note 1 |
| 15 | TioV2ckH | Port In Valid before CLKO \uparrow | 0.25 TCY + 25 | — | — | ns | Note 1 |
| 16 | TckH2ioI | Port In Hold after CLKO \uparrow | 0 | — | — | ns | Note 1 |
| 17 | TosH2ioV | OSC1 \uparrow (Q1 cycle) to Port Out Valid | — | 50 | 150 | ns | |
| 18 | TosH2ioI | OSC1 \uparrow (Q2 cycle) to Port Input Invalid (I/O in hold time) | 100 | — | — | ns | |
| 19 | TioV2osH | Port Input Valid to OSC1 \uparrow (I/O in setup time) | 0 | — | — | ns | |
| 20 | TioR | Port Output Rise Time | — | 40 15 | 72 32 | ns ns | VDD = 1.8V VDD = 3.3V – 5.0V |
| 21 | TioF | Port Output Fall Time | — | 28 15 | 55 30 | ns ns | VDD = 1.8V VDD = 3.3V – 5.0V |
| 22† | TINP | INTx pin High or Low Time | 20 | — | — | ns | |
| 23† | TRBP | RB<7:4> Change KBIx High or Low Time | TCY | — | — | ns | |

† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in RC mode, where CLKO output is 4 x Tosc.

FIGURE 29-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

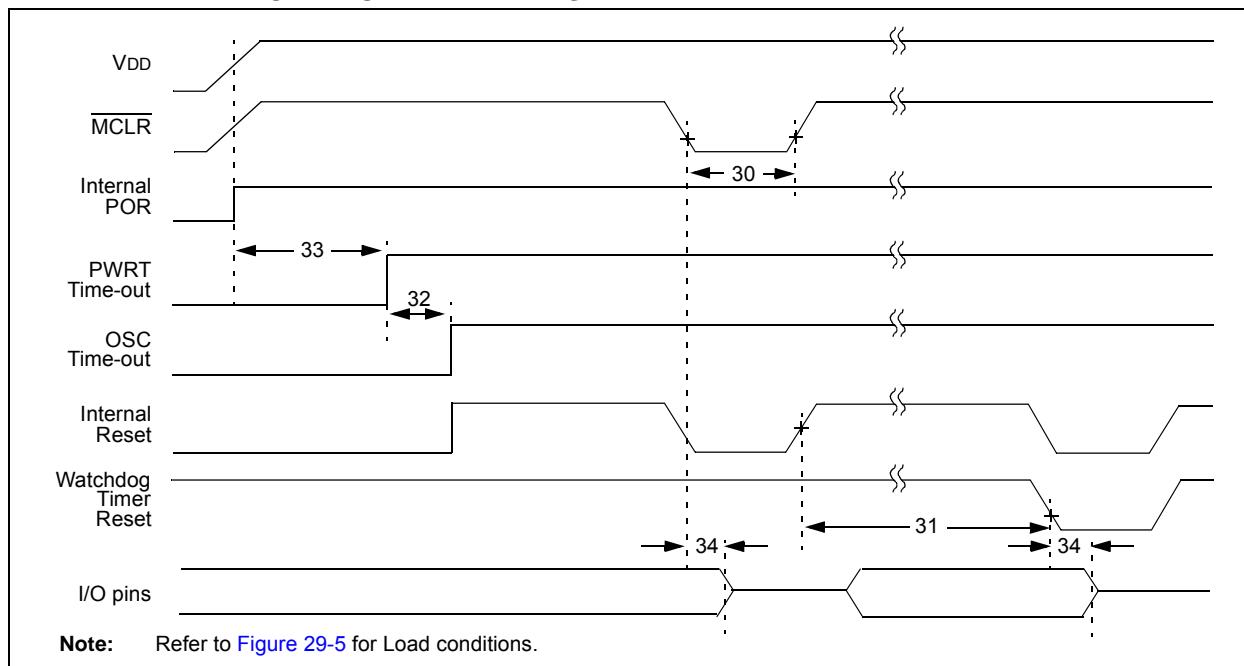
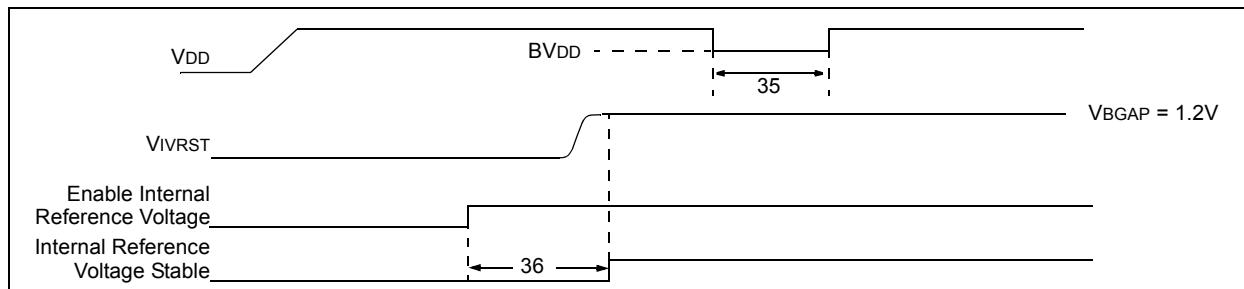


FIGURE 29-9: BROWN-OUT RESET TIMING



PIC18(L)F2X/45K50

TABLE 29-21: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min. | Typ. | Max. | Units | Conditions |
|------------|--------|--|--------------------|------|-----------|-------|-----------------------|
| 30 | TMCL | MCLR Pulse Width (low) | 2 | — | — | μs | |
| 31 | TWDT | Watchdog Timer Time-out Period (no postscaler) | 3.5 | 4.1 | 4.7 | ms | 1:1 prescaler |
| 32 | TOST | Oscillation Start-up Timer Period | 1024 Tosc | — | 1024 Tosc | — | Tosc = OSC1 period |
| 33 | TPWRT | Power-up Timer Period | 54.8 | 64.4 | 74.1 | ms | |
| 34 | TIOZ | I/O High-Impedance from MCLR Low or Watchdog Timer Reset | — | 2 | — | μs | |
| 35 | TBOR | Brown-out Reset Pulse Width | 200 ⁽¹⁾ | — | — | μs | VDD ≤ BVDD (see D005) |
| 36 | TIVRST | Internal Reference Voltage Stable | — | 25 | 35 | μs | |
| 37 | THLVD | High/Low-Voltage Detect Pulse Width | 200 ⁽¹⁾ | — | — | μs | VDD ≤ VHLVD |
| 38 | TCS0 | CPU Start-up Time | 5 | — | 10 | μs | |
| 39 | TIOB0 | Time for HF-INTOSC to Stabilize | — | 0.25 | 1 | ms | |

Note 1: Minimum pulse width that will consistently trigger a Reset or interrupt. Shorter pulses may intermittently trigger a response.

FIGURE 29-10: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

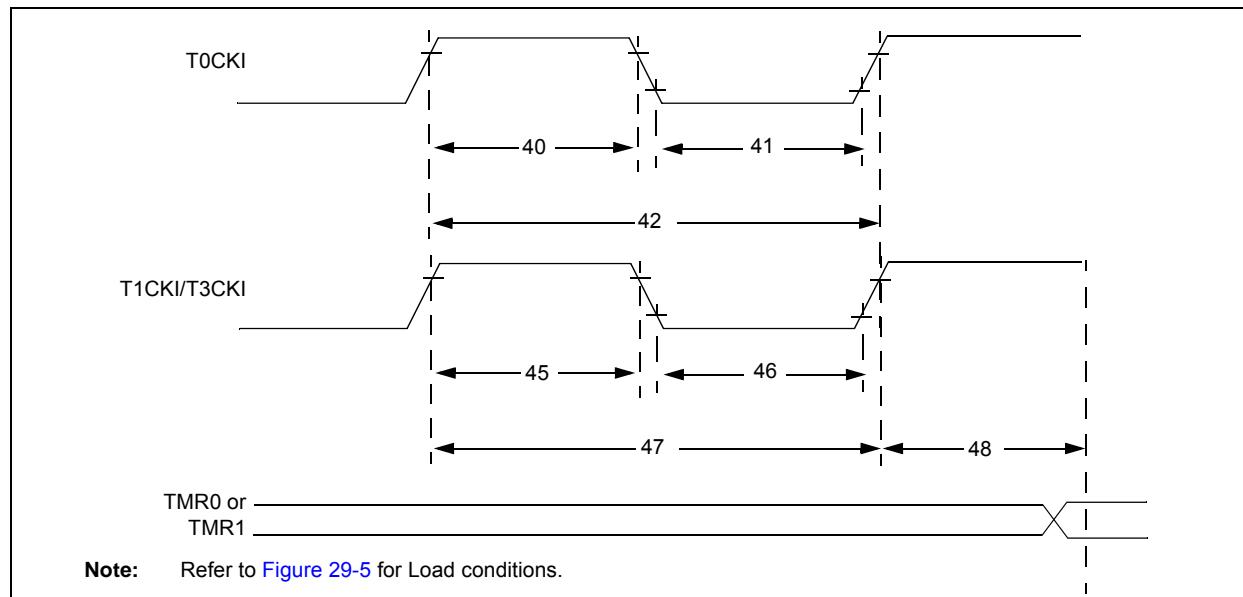
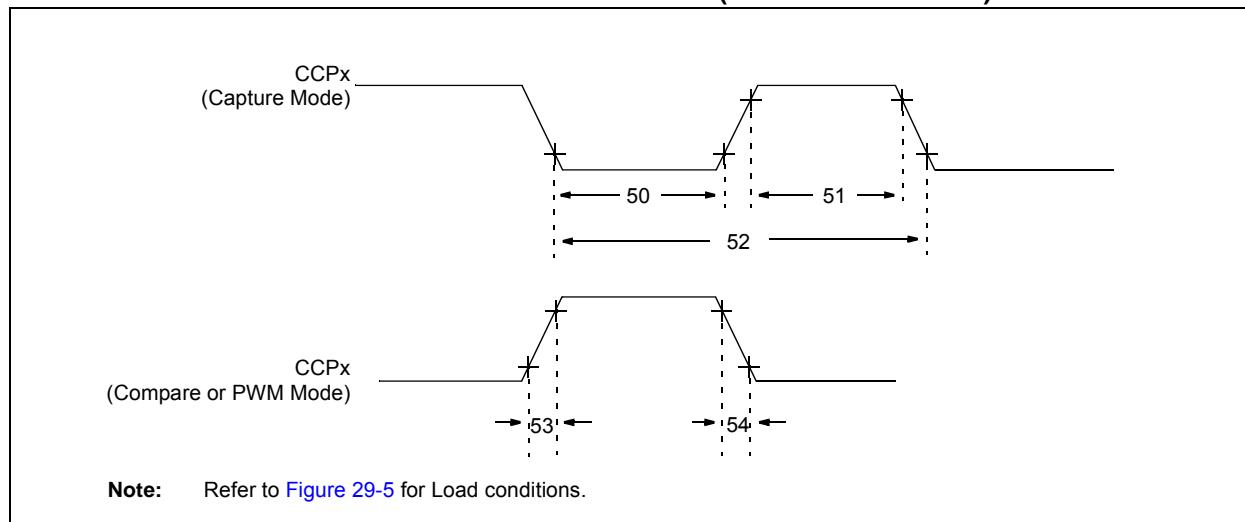


TABLE 29-22: TIMER0 AND TIMER1/3 EXTERNAL CLOCK REQUIREMENTS

| Param. No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|------------|-----------|---|--------------------------------|---|--------|-------|--|
| 40 | TT0H | T0CKI High Pulse Width | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | 10 | — | ns | |
| 41 | TT0L | T0CKI Low Pulse Width | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | 10 | — | ns | |
| 42 | TT0P | T0CKI Period | No prescaler | TCY + 10 | — | ns | N = prescale value (1, 2, 4,..., 256) |
| | | | With prescaler | Greater of: 20 ns or (TCY + 40)/N | — | ns | |
| 45 | TT1H | TxCKI High Time | Synchronous, no prescaler | 0.5 TCY + 20 | — | ns | |
| | | | Synchronous, with prescaler | 10 | — | ns | |
| | | | Asynchronous | 30 | — | ns | |
| 46 | TT1L | TxCKI Low Time | Synchronous, no prescaler | 0.5 TCY + 5 | — | ns | |
| | | | Synchronous, with prescaler | 10 | — | ns | |
| | | | Asynchronous | 30 | — | ns | |
| 47 | TT1P | TxCKI Input Period | Synchronous | Greater of: 20 ns or (TCY + 40)/N | — | ns | N = prescale value (1, 2, 4, 8) |
| | | | Asynchronous | 60 | — | ns | |
| | FT1 | TxCKI Clock Input Frequency Range | | DC | 50 | kHz | |
| 48 | TCKE2TMRL | Delay from External TxCKI Clock Edge to Timer Increment | | 2 Tosc | 7 Tosc | — | |

FIGURE 29-11: CAPTURE/COMPARE/PWM TIMINGS (ALL CCP MODULES)



PIC18(L)F2X/45K50

TABLE 29-23: CAPTURE/COMPARE/PWM REQUIREMENTS (ALL CCP MODULES)

| Param. No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|------------|--------|-----------------------|----------------|--|------|-------|---------------------------------|
| 50 | TccL | CCPx Input Low Time | No prescaler | 0.5 T _{CY} + 20 | — | ns | |
| | | | With prescaler | 10 | — | ns | |
| 51 | TccH | CCPx Input High Time | No prescaler | 0.5 T _{CY} + 20 | — | ns | |
| | | | With prescaler | 10 | — | ns | |
| 52 | TccP | CCPx Input Period | | $\frac{3 \text{ T}_{\text{CY}} + 40}{N}$ | — | ns | N = prescale value (1, 4 or 16) |
| 53 | TccR | CCPx Output Fall Time | | — | 25 | ns | |
| 54 | TccF | CCPx Output Fall Time | | — | 25 | ns | |

FIGURE 29-12: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)

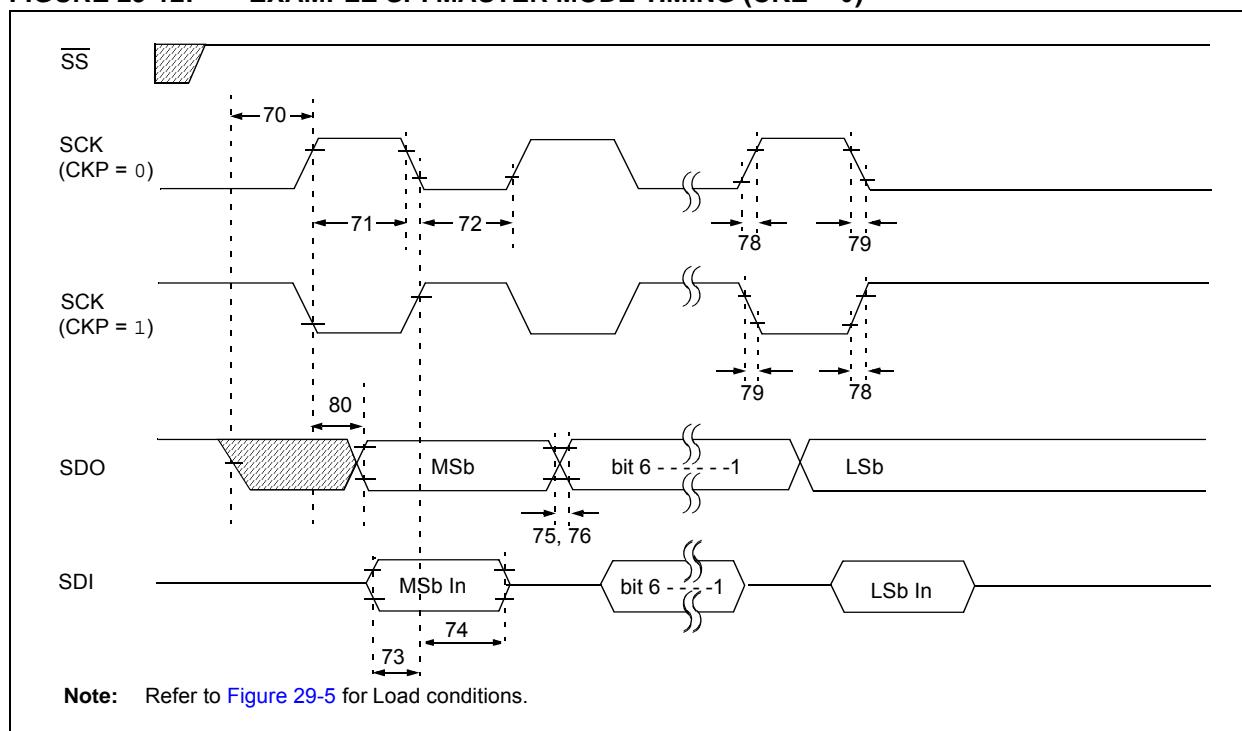
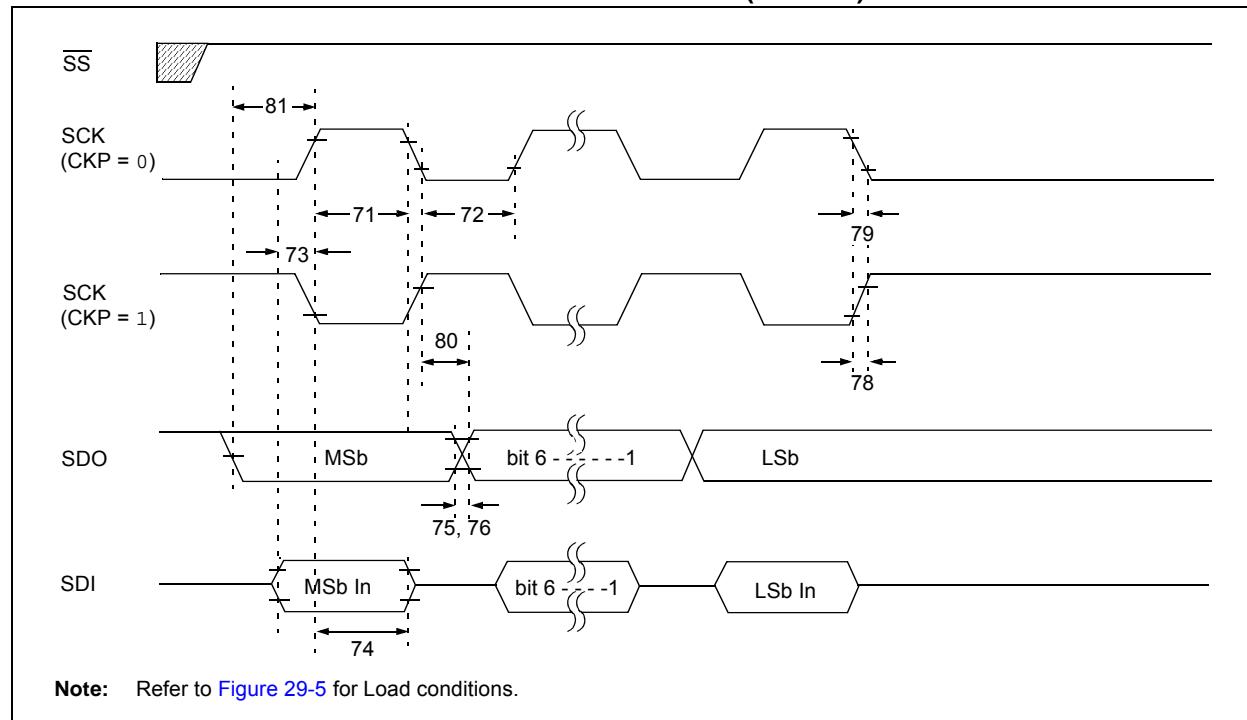


TABLE 29-24: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0 OR 1)

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|------------|--------------------|--|------|------|-------|---------------|
| 73 | TdIV2sch, TdIV2scl | Setup Time of SDI Data Input to SCK Edge | 25 | — | ns | |
| 74 | Tsch2dil, Tscl2dil | Hold Time of SDI Data Input to SCK Edge | 25 | — | ns | |
| 75 | TdoR | SDO Data Output Rise Time | — | 30 | ns | Note 1 |
| 76 | TdoF | SDO Data Output Fall Time | — | 20 | ns | Note 1 |
| 78 | Tscr | SCK Output Rise Time (Master mode) | — | 30 | ns | Note 1 |
| 79 | Tscf | SCK Output Fall Time (Master mode) | — | 20 | ns | Note 1 |
| 80 | Tsch2dov, Tscl2dov | SDO Data Output Valid after SCK Edge | — | 20 | ns | |
| 81 | TdoV2sch, TdoV2scl | SDO Data Output Setup to SCK Edge | Tcy | — | ns | |

Note 1: When the slew rate control limiting I/O port feature is disabled.

FIGURE 29-13: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)



PIC18(L)F2X/45K50

FIGURE 29-14: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)

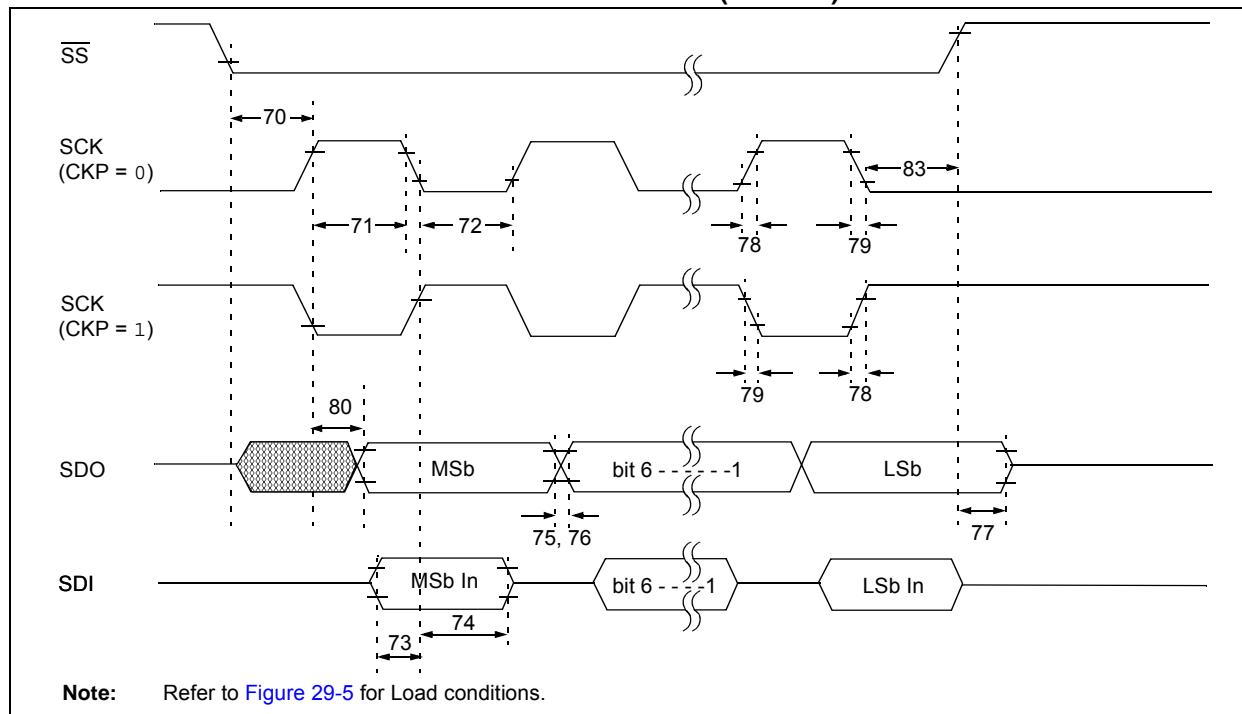


TABLE 29-25: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0 OR 1)

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|------------|-----------------------|--|--------------|------|-------|---------------|
| 70 | TssL2sch, TssL2scL | SS _l ↓ to SCK ↓ or SCK ↑ Input | TCY | — | ns | |
| 71 | Tsch | SCK Input High Time | Continuous | 25 | — | ns |
| 72 | TscL | SCK Input Low Time | Continuous | 30 | — | ns |
| 73 | TdIV2sch, TdIV2scL | Setup Time of SDI Data Input to SCK Edge | 25 | — | ns | |
| 74 | Tsch2dil, Tscl2dil | Hold Time of SDI Data Input to SCK Edge | 25 | — | ns | |
| 75 | Tdor | SDO Data Output Rise Time | — | 30 | ns | Note 1 |
| 76 | Tdof | SDO Data Output Fall Time | — | 20 | ns | Note 1 |
| 77 | TssH2doz | SS _l ↑ to SDO Output High-Impedance | 10 | 50 | ns | |
| 80 | Tsch2dov, Tscl2dov | SDO Data Output Valid after SCK Edge | — | 60 | ns | Note 1 |
| 82 | TssL2dov | SDO Data Output Valid after SS _l ↓ Edge | — | 60 | ns | Note 1 |
| 83 | Tsch2ssH, Tscl2ssH | SS _l ↑ after SCK edge | 1.5 TCY + 40 | — | ns | |

Note 1: When the slew rate control limiting I/O port feature is disabled.

FIGURE 29-15: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)

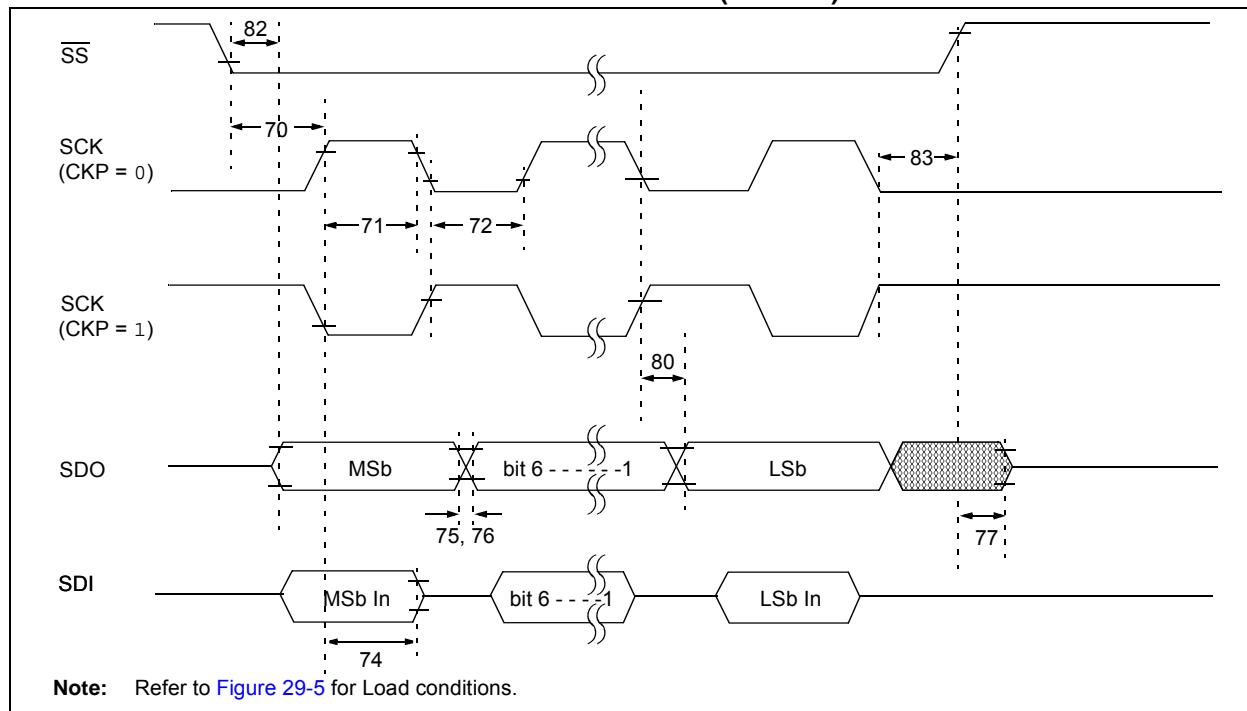
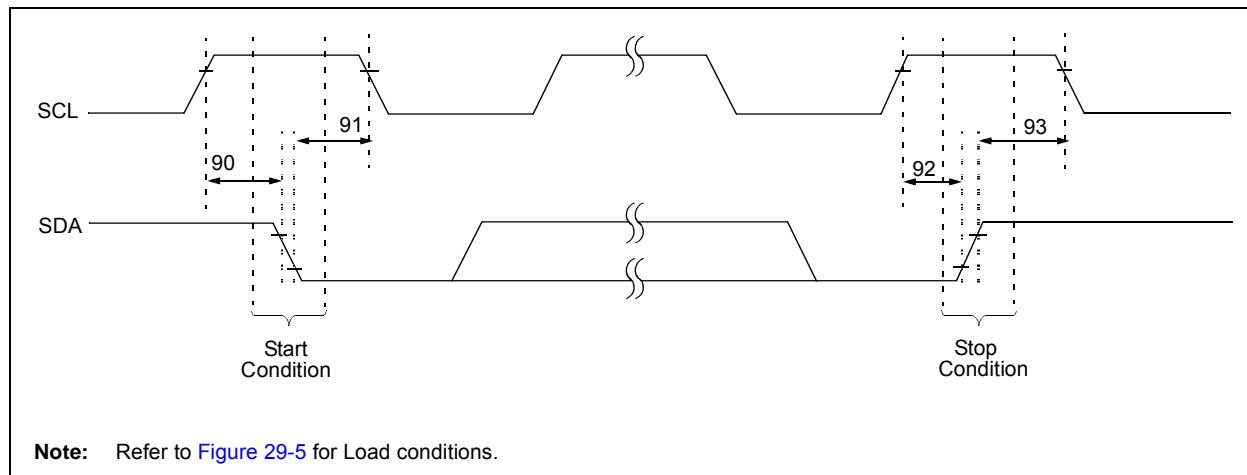


FIGURE 29-16: I²CTM BUS START/STOP BITS TIMING



PIC18(L)F2X/45K50

TABLE 29-26: I²C™ BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

| Param. No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|------------|---------|----------------------------|--------------|------|------|-------|---|
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 4700 | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 600 | — | | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 4000 | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 600 | — | | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 4700 | — | ns | |
| | | | 400 kHz mode | 600 | — | | |
| 93 | THD:STO | Stop Condition Hold Time | 100 kHz mode | 4000 | — | ns | |
| | | | 400 kHz mode | 600 | — | | |

FIGURE 29-17: I²C™ BUS DATA TIMING

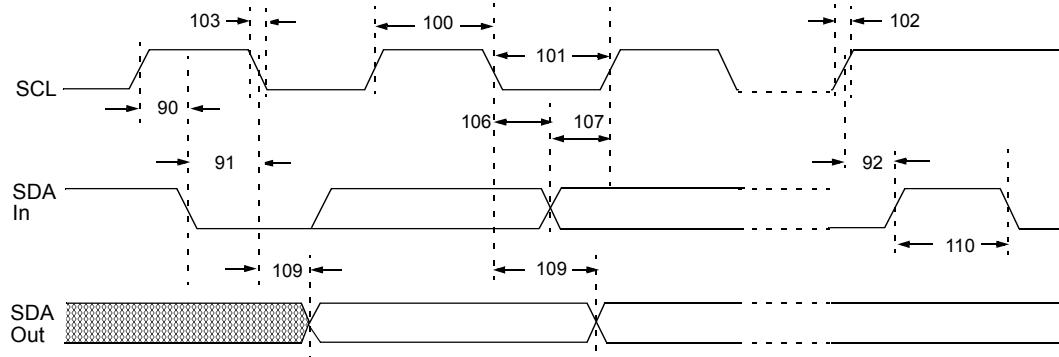


TABLE 29-27: I²C™ BUS DATA REQUIREMENTS (SLAVE MODE)

| Param. No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|------------|---------|----------------------------|--------------|-------------|------|-------|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 4.0 | — | μs | Must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 0.6 | — | μs | Must operate at a minimum of 10 MHz |
| | | | SSP Module | 1.5 TCY | — | | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 4.7 | — | μs | Must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 1.3 | — | μs | Must operate at a minimum of 10 MHz |
| | | | SSP Module | 1.5 TCY | — | | |
| 102 | TR | SDA and SCL Rise Time | 100 kHz mode | — | 1000 | ns | |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | CB is specified to be from 10 to 400 pF |
| 103 | TF | SDA and SCL Fall Time | 100 kHz mode | — | 300 | ns | |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | CB is specified to be from 10 to 400 pF |
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 4.7 | — | μs | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 0.6 | — | μs | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 4.0 | — | μs | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 0.6 | — | μs | |
| 106 | THD:DAT | Data Input Hold Time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | μs | |
| 107 | TSU:DAT | Data Input Setup Time | 100 kHz mode | 250 | — | ns | Note 2 |
| | | | 400 kHz mode | 100 | — | ns | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 4.7 | — | μs | |
| | | | 400 kHz mode | 0.6 | — | μs | |
| 109 | TAA | Output Valid from Clock | 100 kHz mode | — | 3500 | ns | Note 1 |
| | | | 400 kHz mode | — | — | ns | |
| 110 | TBUF | Bus Free Time | 100 kHz mode | 4.7 | — | μs | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | μs | |
| D102 | CB | Bus Capacitive Loading | | — | 400 | pF | |

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

2: A fast mode I²C™ bus device can be used in a standard mode 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.

PIC18(L)F2X/45K50

FIGURE 29-18: MASTER SSP I²C™ BUS START/STOP BITS TIMING WAVEFORMS

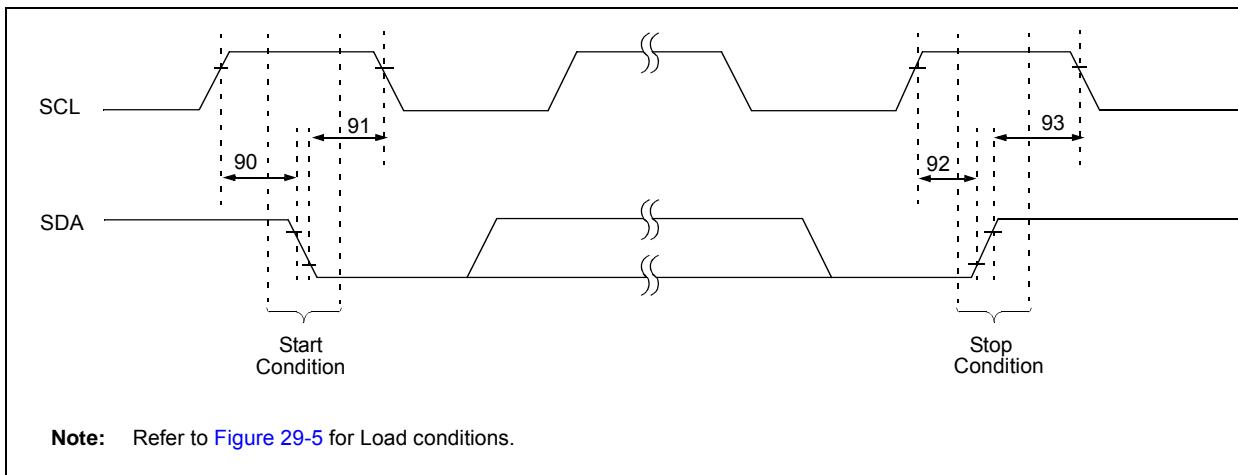


TABLE 29-28: MASTER SSP I²C™ BUS START/STOP BITS REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|------------|---------|---------------------------|------------------|------------------|-------|---|
| 90 | TSU:STA | Start Condition | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ns Only relevant for Repeated Start condition |
| | | Setup Time | 400 kHz mode | 2(Tosc)(BRG + 1) | — | |
| | | 1 MHz mode ⁽¹⁾ | 2(Tosc)(BRG + 1) | — | — | |
| 91 | THD:STA | Start Condition | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ns After this period, the first clock pulse is generated |
| | | Hold Time | 400 kHz mode | 2(Tosc)(BRG + 1) | — | |
| | | 1 MHz mode ⁽¹⁾ | 2(Tosc)(BRG + 1) | — | — | |
| 92 | TSU:STO | Stop Condition | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ns |
| | | Setup Time | 400 kHz mode | 2(Tosc)(BRG + 1) | — | |
| | | 1 MHz mode ⁽¹⁾ | 2(Tosc)(BRG + 1) | — | — | |
| 93 | THD:STO | Stop Condition | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ns |
| | | Hold Time | 400 kHz mode | 2(Tosc)(BRG + 1) | — | |
| | | 1 MHz mode ⁽¹⁾ | 2(Tosc)(BRG + 1) | — | — | |

Note 1: Maximum pin capacitance = 10 pF for all I²C™ pins.

FIGURE 29-19: MASTER SSP I²C™ BUS DATA TIMING

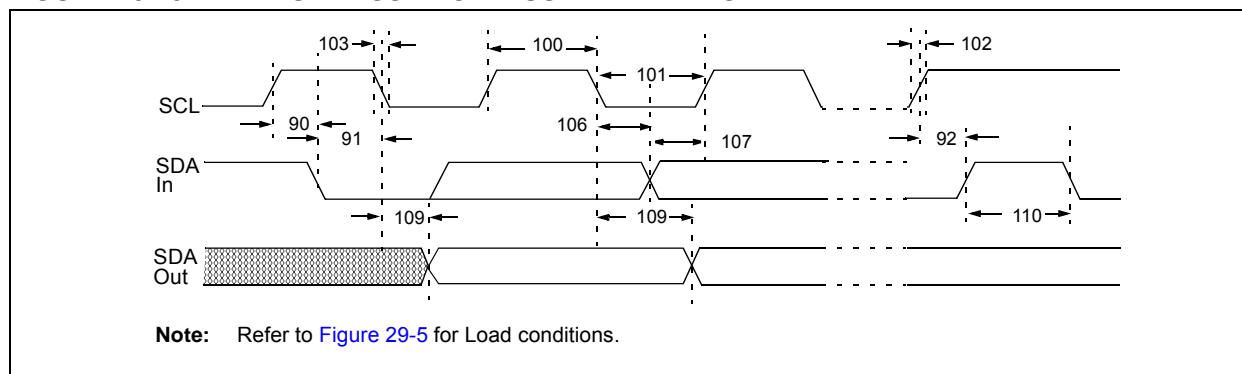


TABLE 29-29: MASTER SSP I²C™ BUS DATA REQUIREMENTS

| Param. No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|------------|---------|----------------------------|---------------------------|------------------|------|-------|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode ⁽¹⁾ | 2(Tosc)(BRG + 1) | — | ms | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode ⁽¹⁾ | 2(Tosc)(BRG + 1) | — | ms | |
| 102 | TR | SDA and SCL Rise Time | 100 kHz mode | — | 1000 | ns | CB is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | |
| | | | 1 MHz mode ⁽¹⁾ | — | 300 | ns | |
| 103 | TF | SDA and SCL Fall Time | 100 kHz mode | — | 300 | ns | CB is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | |
| | | | 1 MHz mode ⁽¹⁾ | — | 100 | ns | |
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode ⁽¹⁾ | 2(Tosc)(BRG + 1) | — | ms | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode ⁽¹⁾ | 2(Tosc)(BRG + 1) | — | ms | |
| 106 | THD:DAT | Data Input Hold Time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | ms | |
| 107 | TSU:DAT | Data Input Setup Time | 100 kHz mode | 250 | — | ns | Note 2 |
| | | | 400 kHz mode | 100 | — | ns | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(Tosc)(BRG + 1) | — | ms | |
| | | | 1 MHz mode ⁽¹⁾ | 2(Tosc)(BRG + 1) | — | ms | |
| 109 | TAA | Output Valid from Clock | 100 kHz mode | — | 3500 | ns | |
| | | | 400 kHz mode | — | 1000 | ns | |
| | | | 1 MHz mode ⁽¹⁾ | — | — | ns | |
| 110 | TBUF | Bus Free Time | 100 kHz mode | 4.7 | — | ms | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | ms | |
| D102 | CB | Bus Capacitive Loading | | — | 400 | pF | |

Note 1: Maximum pin capacitance = 10 pF for all I²C™ pins.

2: A fast mode I²C bus device can be used in a standard mode I²C bus system, but parameter 107 \geq 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line, parameter 102 + parameter 107 = 1000 + 250 = 1250 ns (for 100 kHz mode), before the SCL line is released.

PIC18(L)F2X/45K50

FIGURE 29-20: EUSART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

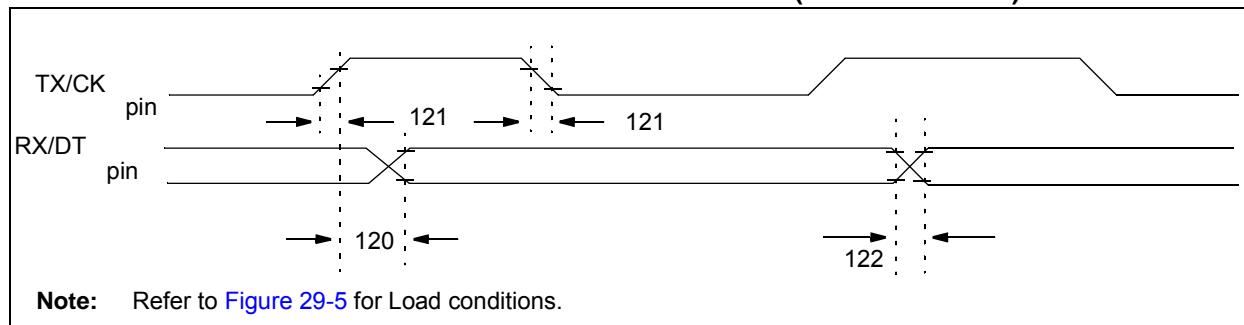


TABLE 29-30: EUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|------------|----------|---|------|------|-------|---------------|
| 120 | TckH2DTV | <u>SYNC XMIT (MASTER & SLAVE)</u> Clock High to Data Out Valid | — | 40 | ns | |
| 121 | TCKRF | Clock Out Rise Time and Fall Time (Master mode) | — | 20 | ns | Note 1 |
| 122 | TDTRF | Data Out Rise Time and Fall Time | — | 20 | ns | Note 1 |

Note 1: When the slew rate control limiting I/O port feature is disabled.

FIGURE 29-21: EUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

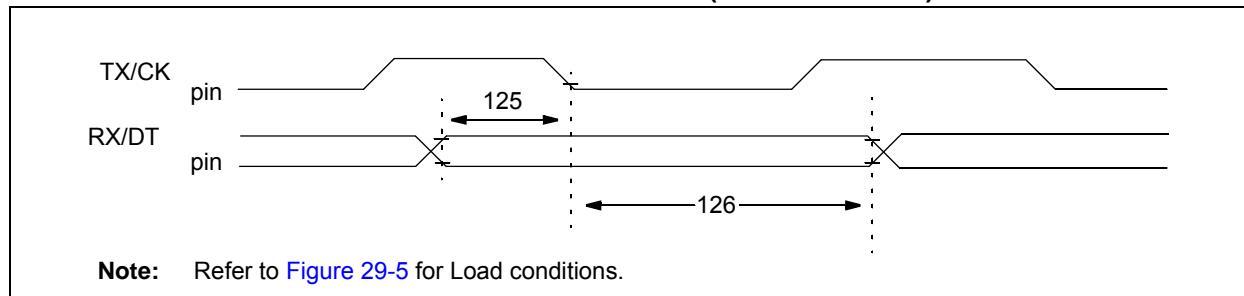


TABLE 29-31: EUSART SYNCHRONOUS RECEIVE REQUIREMENTS

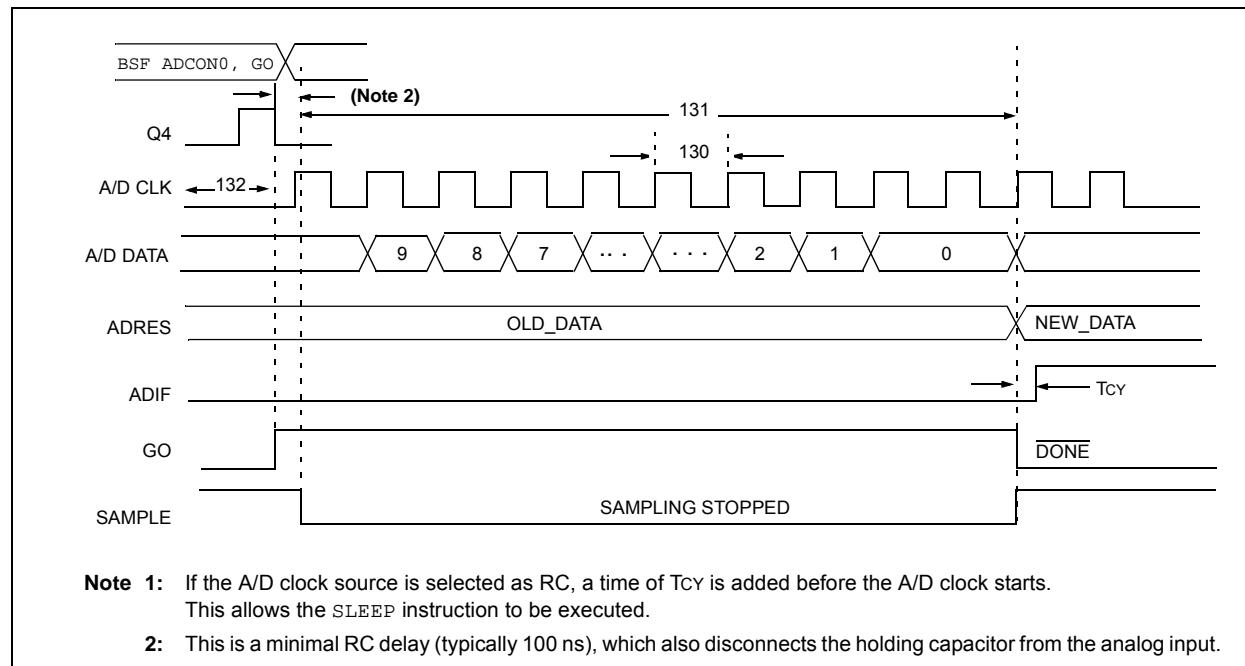
| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|------------|----------|--|------|------|-------|------------|
| 125 | TDTV2CKL | <u>SYNC RCV (MASTER & SLAVE)</u> Data Setup before CK ↓ (DT setup time) | 10 | — | ns | |
| 126 | TckL2DTL | Data Hold after CK ↓ (DT hold time) | 15 | — | ns | |

TABLE 29-32: A/D CONVERTER CHARACTERISTICS (PIC18(L)F2X/45K50)⁽¹⁾

| PIC18(L)F2X/45K50 | | | Standard Operating Conditions (unless otherwise stated) Operating temperature: Tested at +25°C | | | | |
|-------------------|--------------------|---|---|-----------|-----------------------|-------|-------------------------|
| Param. No. | Symbol | Characteristic | Min. | Typ. | Max. | Units | Conditions |
| A01 | NR | Resolution | — | — | 10 | bits | $\Delta V_{REF} = 3.0V$ |
| A03 | EIL | Integral Linearity Error | — | ± 0.5 | ± 1 | LSb | $\Delta V_{REF} = 3.0V$ |
| A04 | EDL | Differential Linearity Error | — | ± 0.5 | ± 1 | LSb | $\Delta V_{REF} = 3.0V$ |
| A06 | E _{OFF} | Offset Error | — | ± 0.7 | ± 2 | LSb | $\Delta V_{REF} = 3.0V$ |
| A07 | E _{GN} | Gain Error | — | ± 0.7 | ± 2 | LSb | $\Delta V_{REF} = 3.0V$ |
| A08 | E _{TOTAL} | Total Error | — | ± 0.8 | ± 3 | LSb | $\Delta V_{REF} = 3.0V$ |
| A20 | ΔV_{REF} | Reference Voltage Range ($V_{REFH} - V_{REFL}$) | 2 | — | V _{DD} | V | |
| A21 | V _{REFH} | Reference Voltage High | V _{DD} /2 | — | V _{DD} + 0.3 | V | |
| A22 | V _{REFL} | Reference Voltage Low | V _{SS} - 0.3V | — | V _{DD} /2 | V | |
| A25 | V _{AIN} | Analog Input Voltage | V _{REFL} | — | V _{REFH} | V | |
| A30 | Z _{AIN} | Recommended Impedance of Analog Voltage Source | — | — | 10 | kΩ | |

Note 1: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

FIGURE 29-22: A/D CONVERSION TIMING



PIC18(L)F2X/45K50

TABLE 29-33: A/D CONVERSION REQUIREMENTS (PIC18(L)F2X/45K50)

| Standard Operating Conditions (unless otherwise stated) Operating temperature: Tested at +25°C | | | | | | | |
|---|--------|--|------|------|----------|-------|--------------------|
| Param. No. | Symbol | Characteristic | Min. | Typ. | Max. | Units | Conditions |
| 130 | TAD | A/D Clock Period | 1 | — | 25 | μs | -40°C to +85°C |
| 131 | TCNV | Conversion Time (not including acquisition time) ⁽¹⁾ | 12 | — | 12 | TAD | |
| 132 | TACQ | Acquisition Time ⁽²⁾ | 1.4 | — | — | μs | VDD = 3V, Rs = 50Ω |
| 135 | TSWC | Switching Time from Convert → Sample | — | — | (Note 3) | — | |
| 136 | TDIS | Discharge Time | 2 | — | 2 | TAD | |

Note 1: ADRES register may be read on the following TCY cycle.

2: The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (VDD to Vss or Vss to VDD). The source impedance (Rs) on the input channels is 50 Ω.

3: On the following cycle of the device clock.

30.0 DC AND AC CHARACTERISTICS GRAPHS AND CHARTS

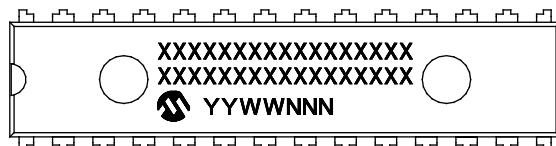
Graphs and charts are not available at this time.

PIC18(L)F2X/45K50

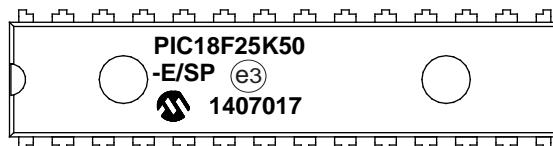
31.0 PACKAGING INFORMATION

31.1 Package Marking Information

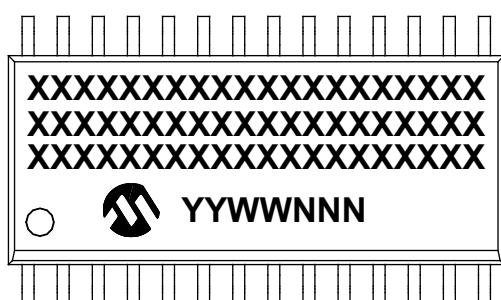
28-Lead SPDIP (.300")



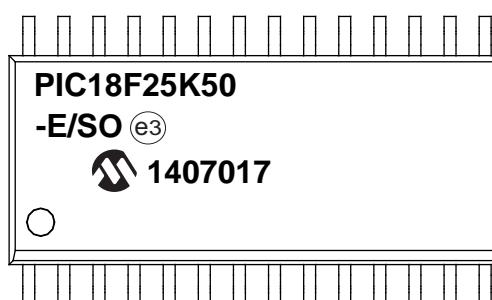
Example



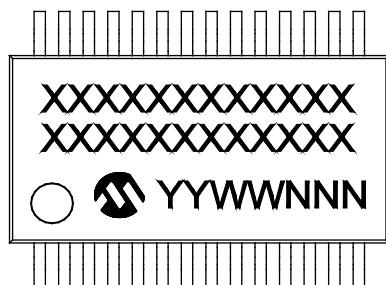
28-Lead SOIC (7.50 mm)



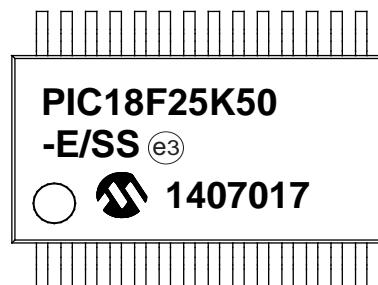
Example



28-Lead SSOP (5.30 mm)



Example



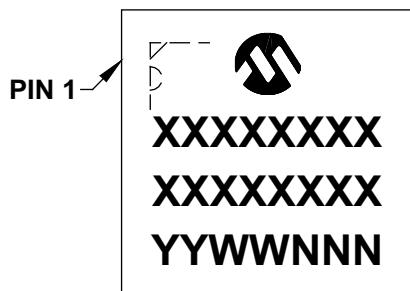
Legend:

| | |
|--------|--|
| XX...X | Customer-specific information or Microchip part number |
| 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. |

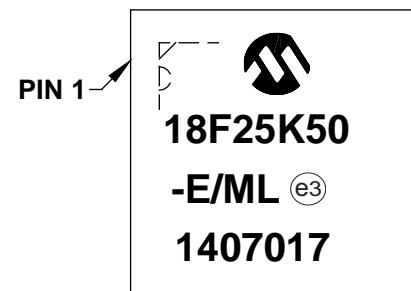
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

Package Marking Information (Continued)

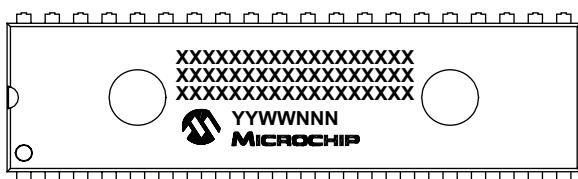
28-Lead QFN (6x6 mm)



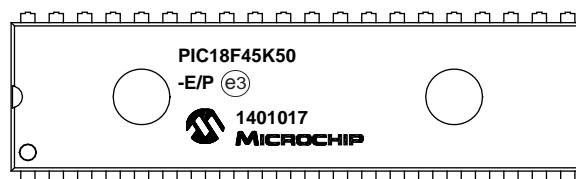
Example



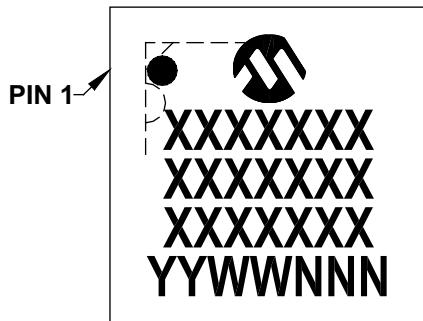
40-Lead PDIP (600 mil)



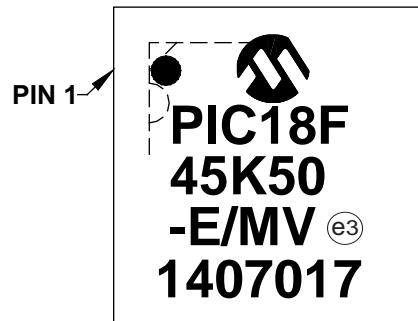
Example



40-Lead UQFN (5x5x0.5 mm)



Example



Legend:

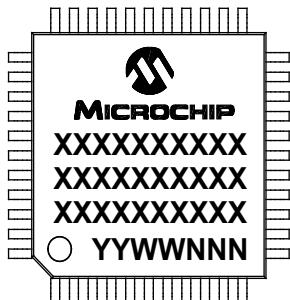
- XX...X Customer-specific information or Microchip part number
- Y Year code (last digit of calendar year)
- YY Year code (last 2 digits of calendar year)
- WW Week code (week of January 1 is week '01')
- NNN Alphanumeric traceability code
- (e3) Pb-free JEDEC® designator for Matte Tin (Sn)
- * This 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.

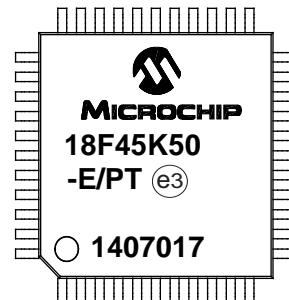
PIC18(L)F2X/45K50

Package Marking Information (Continued)

44-Lead TQFP (10x10x1 mm)



Example



| | |
|----------------|--|
| Legend: | XX...X Customer-specific information or Microchip part number |
| 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. |

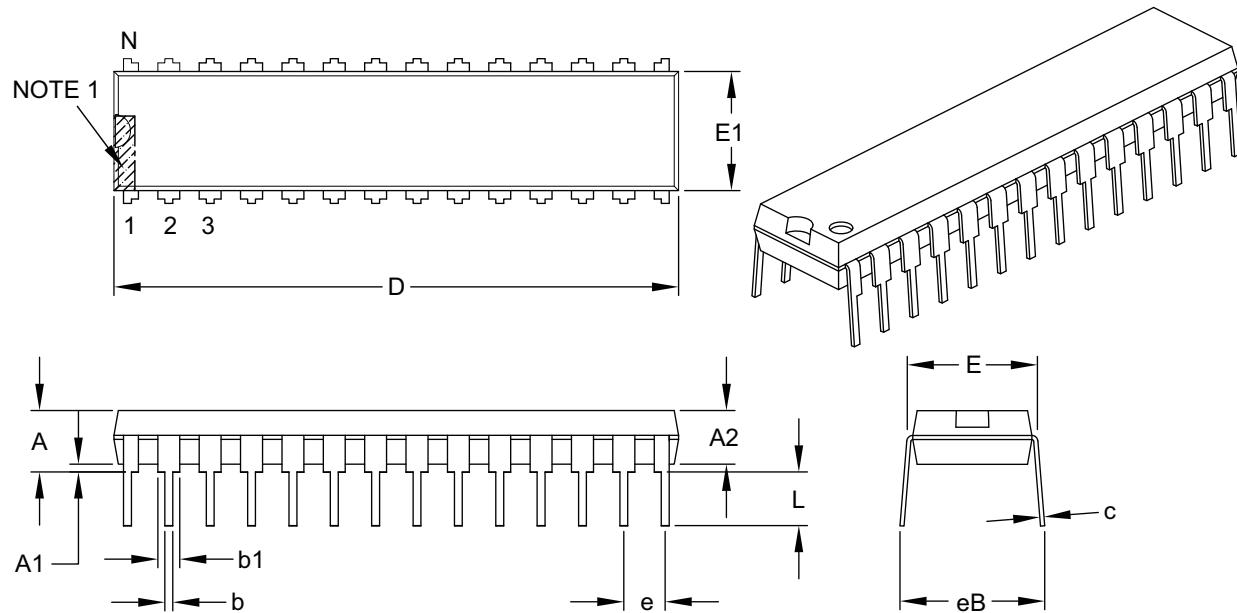
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

31.2 Package Details

The following sections give the technical details of the packages.

28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| | | Units | INCHES | | |
|----------------------------|----|-------|----------|-------|-----|
| Dimension Limits | | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | | |
| Pitch | e | | .100 BSC | | |
| Top to Seating Plane | A | — | — | .200 | |
| Molded Package Thickness | A2 | .120 | .135 | .150 | |
| Base to Seating Plane | A1 | .015 | — | — | |
| Shoulder to Shoulder Width | E | .290 | .310 | .335 | |
| Molded Package Width | E1 | .240 | .285 | .295 | |
| Overall Length | D | 1.345 | 1.365 | 1.400 | |
| Tip to Seating Plane | L | .110 | .130 | .150 | |
| Lead Thickness | c | .008 | .010 | .015 | |
| Upper Lead Width | b1 | .040 | .050 | .070 | |
| Lower Lead Width | b | .014 | .018 | .022 | |
| Overall Row Spacing § | eB | — | — | .430 | |

Notes:

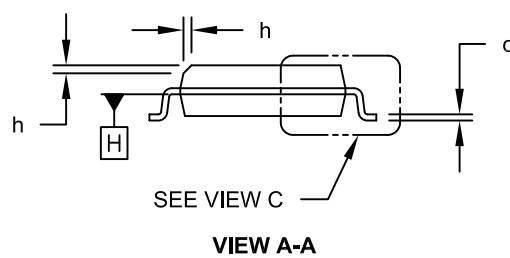
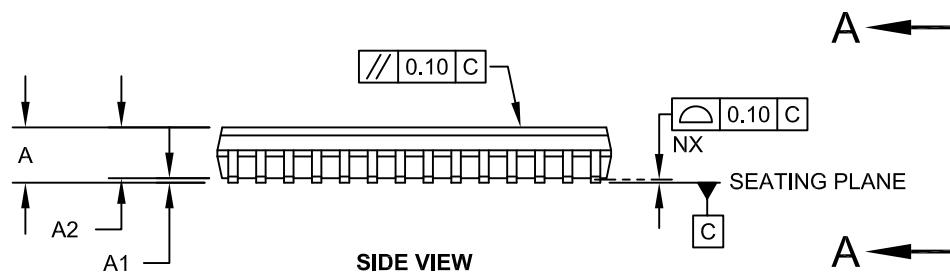
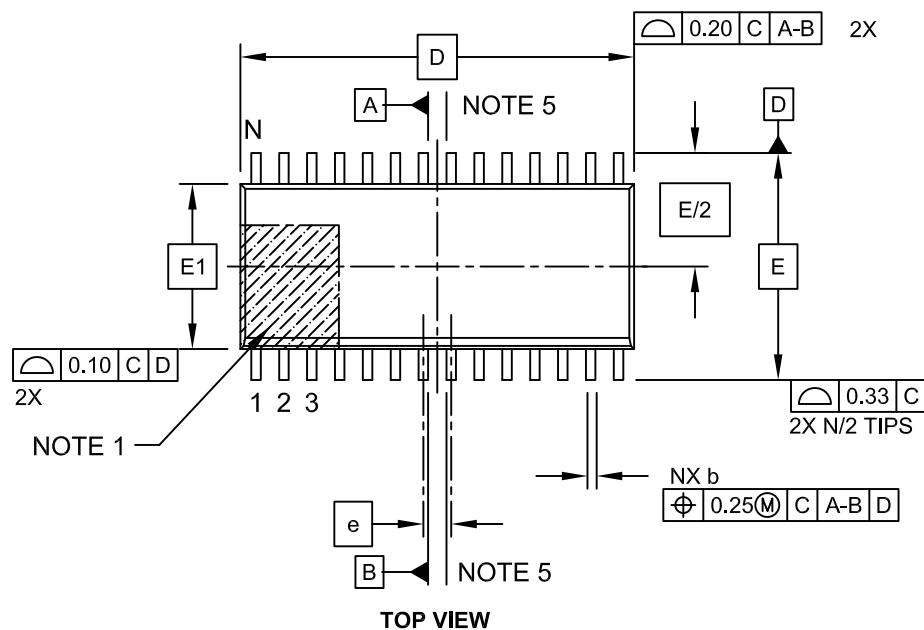
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

PIC18(L)F2X/45K50

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

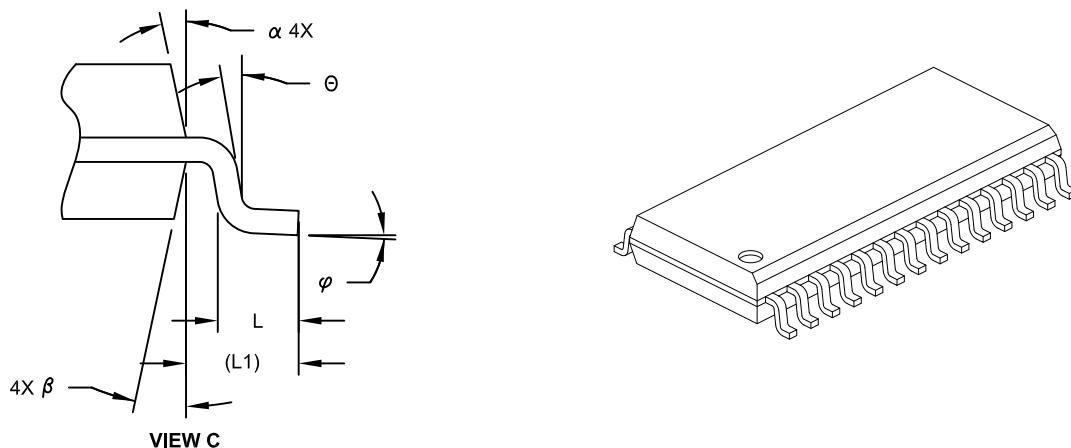
Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



Microchip Technology Drawing C04-052C Sheet 1 of 2

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| | | Units | MILLIMETERS | | |
|--------------------------|----|-------|-------------|-----|------|
| Dimension Limits | | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | | |
| Pitch | e | | 1.27 | BSC | |
| Overall Height | A | | - | - | 2.65 |
| Molded Package Thickness | A2 | | 2.05 | - | - |
| Standoff | § | A1 | 0.10 | - | 0.30 |
| Overall Width | E | | 10.30 | BSC | |
| Molded Package Width | E1 | | 7.50 | BSC | |
| Overall Length | D | | 17.90 | BSC | |
| Chamfer (Optional) | h | | 0.25 | - | 0.75 |
| Foot Length | L | | 0.40 | - | 1.27 |
| Footprint | L1 | | 1.40 REF | | |
| Lead Angle | θ | | 0° | - | - |
| Foot Angle | φ | | 0° | - | 8° |
| Lead Thickness | c | | 0.18 | - | 0.33 |
| Lead Width | b | | 0.31 | - | 0.51 |
| Mold Draft Angle Top | α | | 5° | - | 15° |
| Mold Draft Angle Bottom | β | | 5° | - | 15° |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.

2. § Significant Characteristic

3. 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.

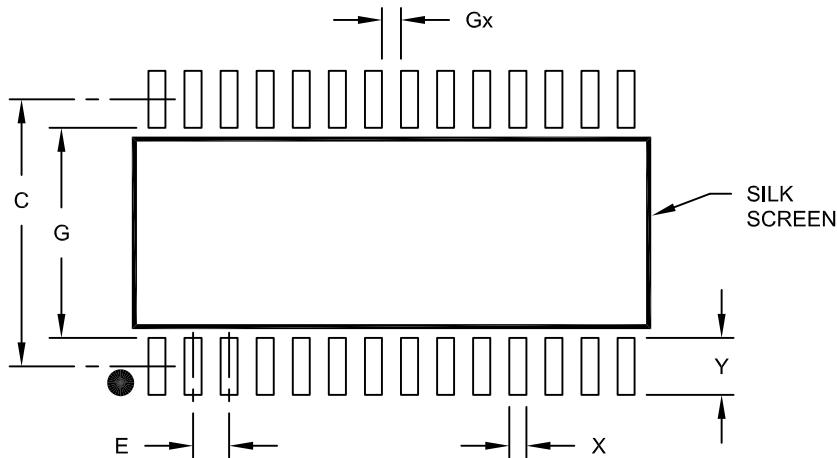
REF: Reference Dimension, usually without tolerance, for information purposes only.

5. Datums A & B to be determined at Datum H.

PIC18(L)F2X/45K50

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



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 (X28) | X | | | 0.60 |
| Contact Pad Length (X28) | Y | | | 2.00 |
| Distance Between Pads | Gx | 0.67 | | |
| Distance Between Pads | G | 7.40 | | |

Notes:

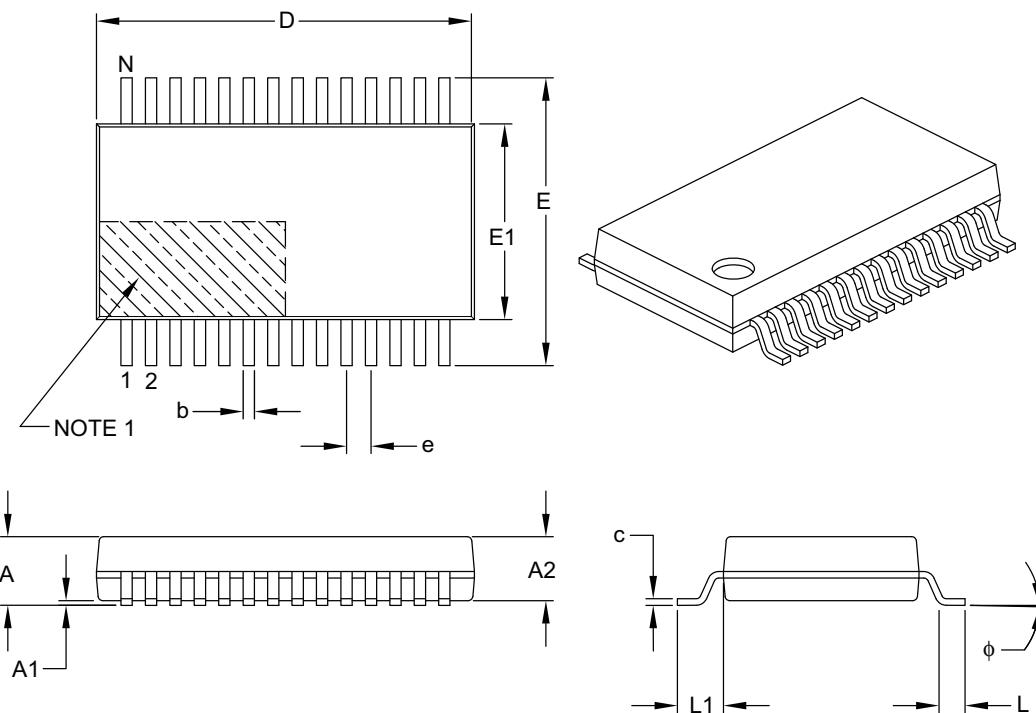
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2052A

28-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 | | 28 | | |
| 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 | 9.90 | 10.20 | 10.50 | |
| 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.

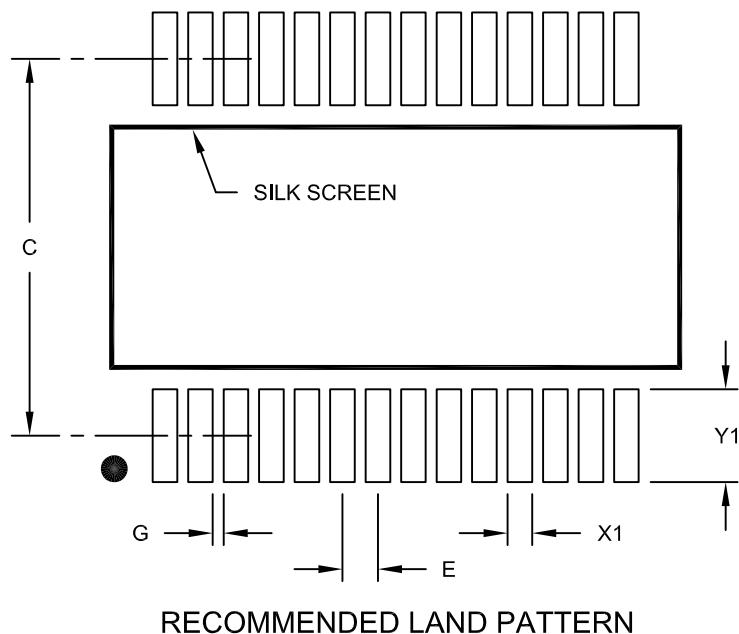
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

REF: Reference Dimension, usually without tolerance, for information purposes only.

PIC18(L)F2X/45K50

28-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 |
| Contact Pitch | | E | 0.65 BSC | | |
| Contact Pad Spacing | C | | | 7.20 | |
| Contact Pad Width (X28) | X1 | | | | 0.45 |
| Contact Pad Length (X28) | Y1 | | | | 1.75 |
| Distance Between Pads | G | 0.20 | | | |

Notes:

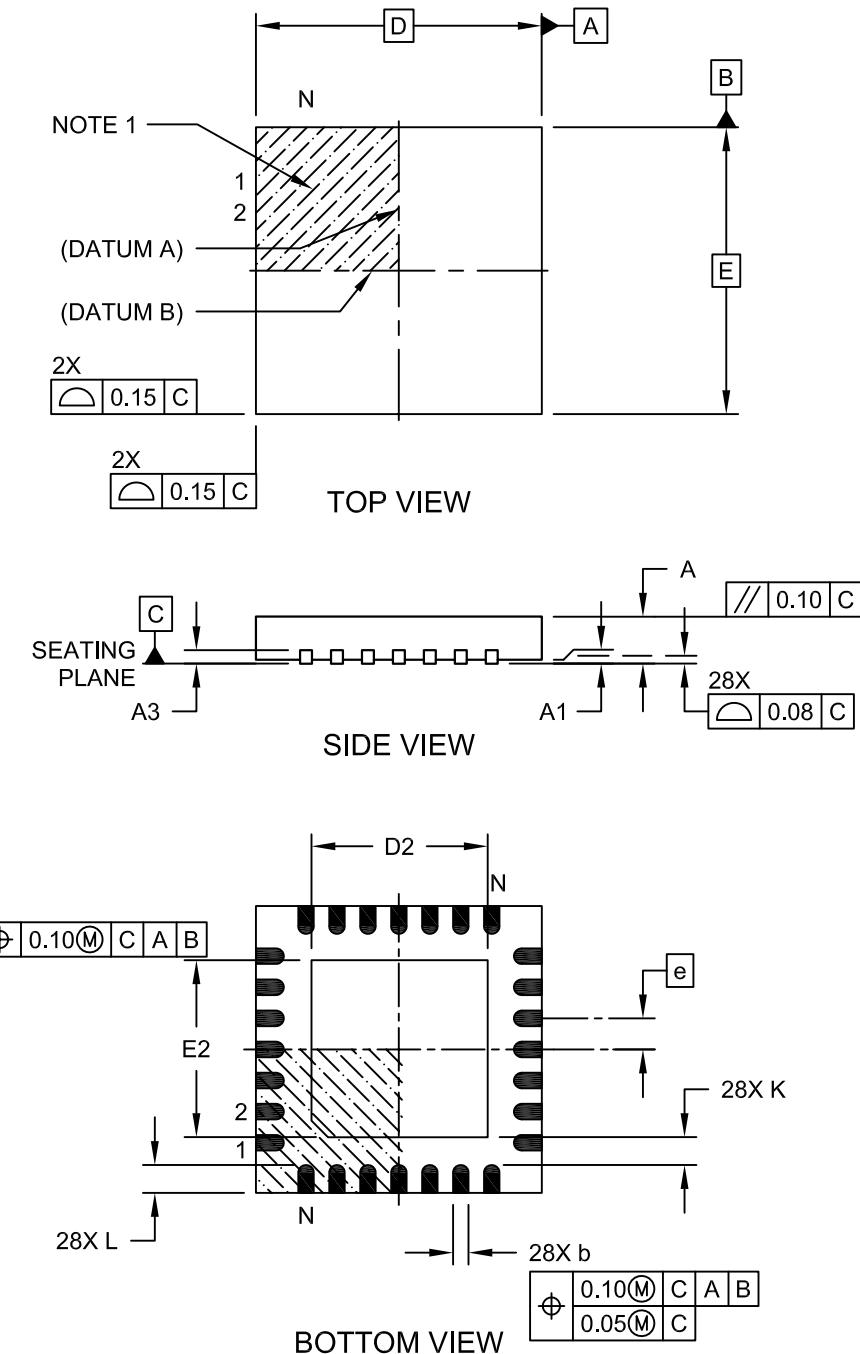
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

**28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN]
With 0.55 mm Terminal Length**

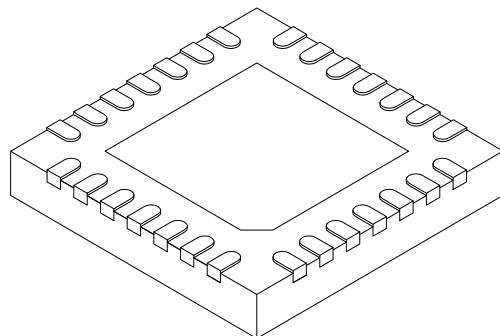
Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



PIC18(L)F2X/45K50

28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN] With 0.55 mm Terminal Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| Dimension | Units | MILLIMETERS | | |
|-------------------------|-------|-------------|----------|------|
| | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Terminal Thickness | A3 | | 0.20 REF | |
| Overall Width | E | | 6.00 BSC | |
| Exposed Pad Width | E2 | 3.65 | 3.70 | 4.20 |
| Overall Length | D | | 6.00 BSC | |
| Exposed Pad Length | D2 | 3.65 | 3.70 | 4.20 |
| Terminal Width | b | 0.23 | 0.30 | 0.35 |
| Terminal Length | L | 0.50 | 0.55 | 0.70 |
| Terminal-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.

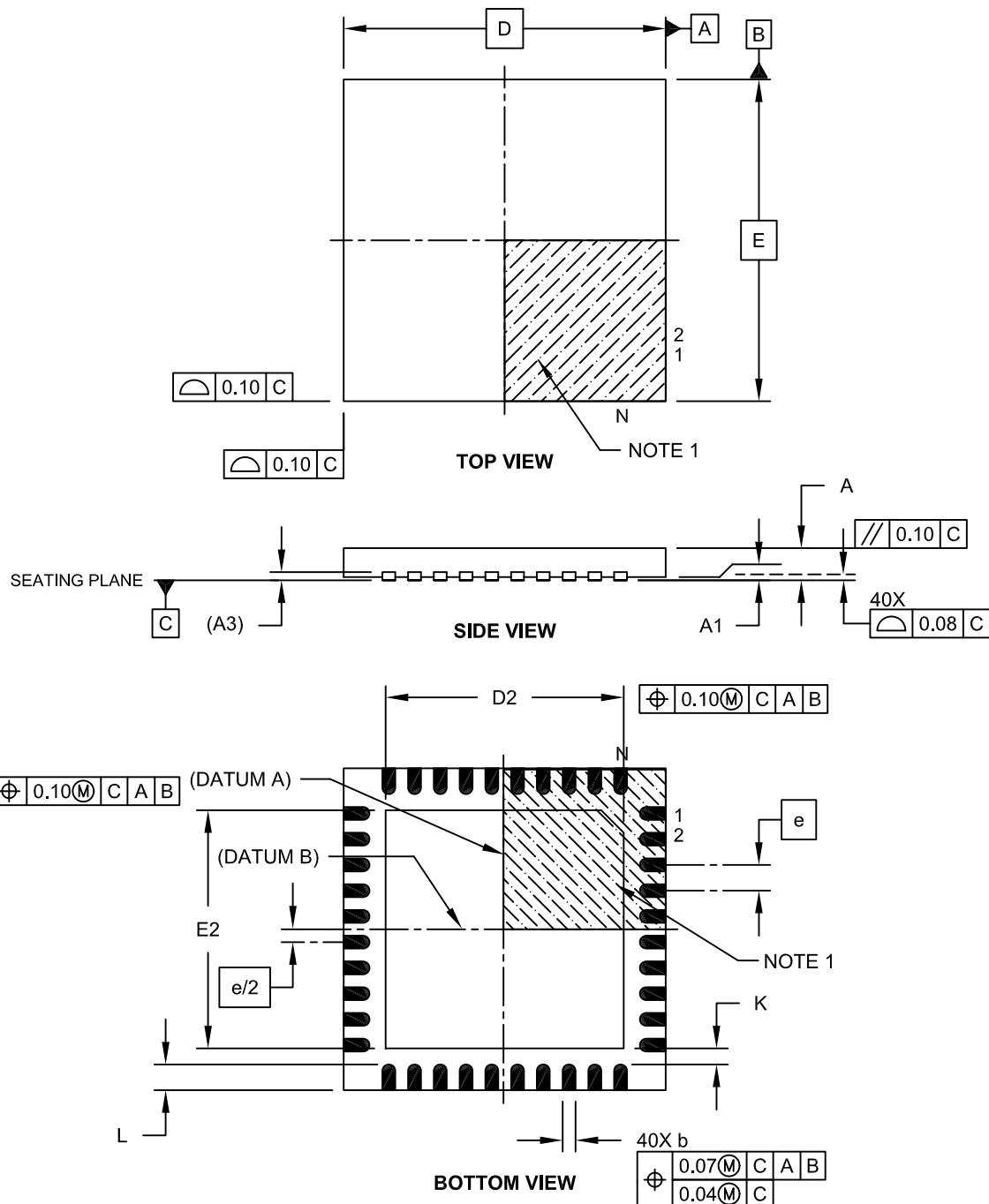
Microchip Technology Drawing C04-105C Sheet 2 of 2

40-Lead Plastic Dual In-Line (P) – 600 mil Body [PDIP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>

40-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) – 5x5x0.5 mm Body [UQFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>

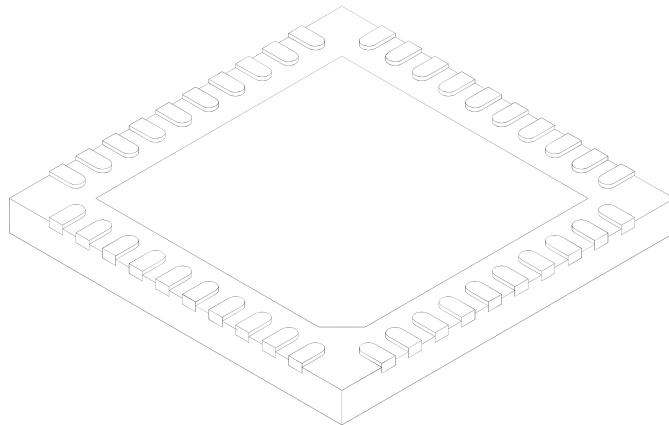


Microchip Technology Drawing C04-156A Sheet 1 of 2

PIC18(L)F2X/45K50

40-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) – 5x5x0.5 mm Body [UQFN]

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 | | 40 |
| Pitch | | e | | 0.40 BSC |
| Overall Height | | A | | 0.45 0.50 0.55 |
| Standoff | | A1 | | 0.00 0.02 0.05 |
| Contact Thickness | | A3 | | 0.127 REF |
| Overall Width | | E | | 5.00 BSC |
| Exposed Pad Width | | E2 | | 3.60 3.70 3.80 |
| Overall Length | | D | | 5.00 BSC |
| Exposed Pad Length | | D2 | | 3.60 3.70 3.80 |
| Contact Width | | b | | 0.15 0.20 0.25 |
| Contact Length | | L | | 0.30 0.40 0.50 |
| Contact-to-Exposed Pad | | K | | - - |

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.

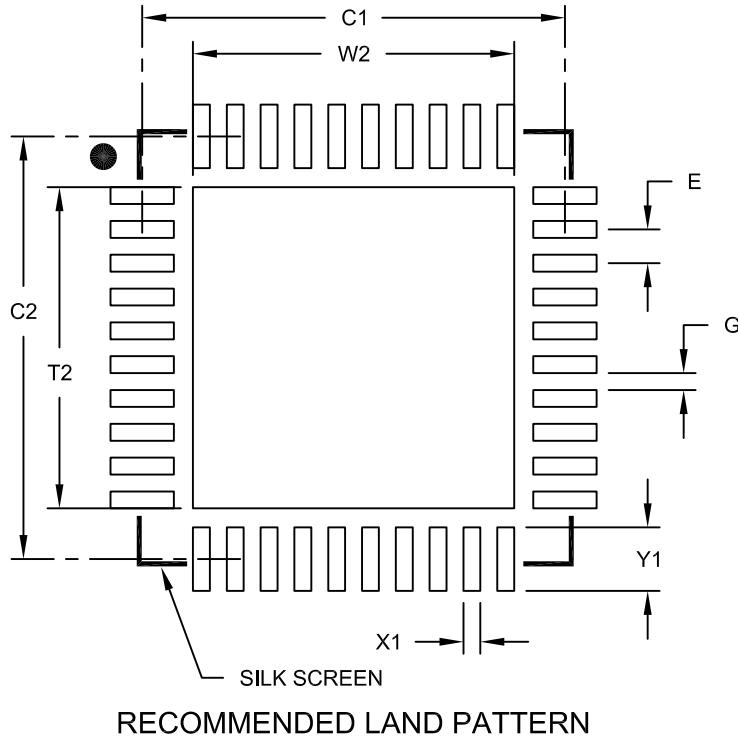
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-156A Sheet 2 of 2

40-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) - 5x5 mm Body [UQFN]

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 |
| Contact Pitch | E | | 0.40 | BSC |
| Optional Center Pad Width | W2 | | | 3.80 |
| Optional Center Pad Length | T2 | | | 3.80 |
| Contact Pad Spacing | C1 | | 5.00 | |
| Contact Pad Spacing | C2 | | 5.00 | |
| Contact Pad Width (X40) | X1 | | | 0.20 |
| Contact Pad Length (X40) | Y1 | | | 0.75 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

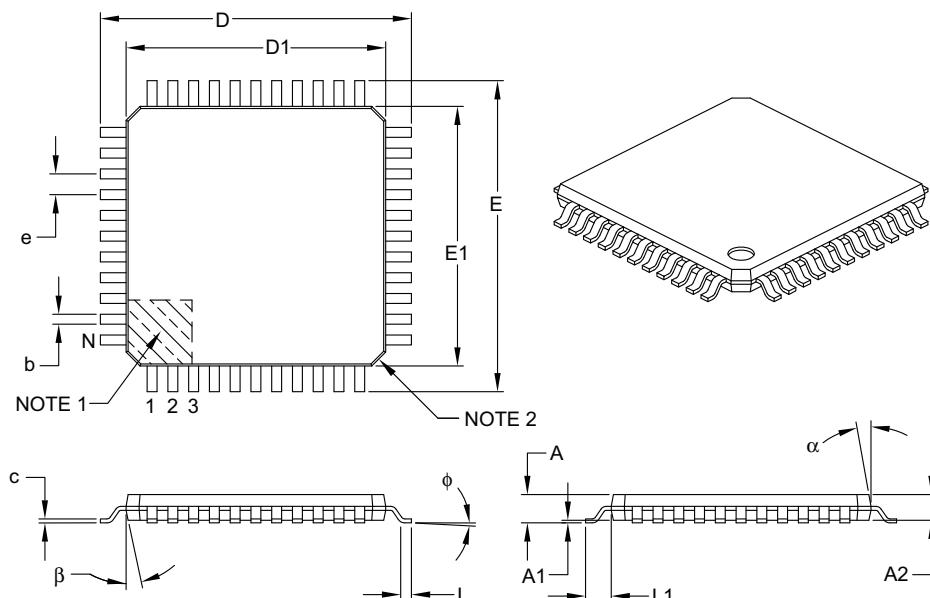
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2156B

PIC18(L)F2X/45K50

44-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| | | Units | MILLIMETERS | | |
|--------------------------|--|-------|-------------|------|------|
| Dimension Limits | | | MIN | NOM | MAX |
| Number of Leads | | N | 44 | | |
| Lead Pitch | | e | 0.80 BSC | | |
| Overall Height | | A | – | – | 1.20 |
| Molded Package Thickness | | A2 | 0.95 | 1.00 | 1.05 |
| Standoff | | A1 | 0.05 | – | 0.15 |
| Foot Length | | L | 0.45 | 0.60 | 0.75 |
| Footprint | | L1 | 1.00 REF | | |
| Foot Angle | | ϕ | 0° | 3.5° | 7° |
| Overall Width | | E | 12.00 BSC | | |
| Overall Length | | D | 12.00 BSC | | |
| Molded Package Width | | E1 | 10.00 BSC | | |
| Molded Package Length | | D1 | 10.00 BSC | | |
| Lead Thickness | | c | 0.09 | – | 0.20 |
| Lead Width | | b | 0.30 | 0.37 | 0.45 |
| Mold Draft Angle Top | | α | 11° | 12° | 13° |
| Mold Draft Angle Bottom | | β | 11° | 12° | 13° |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

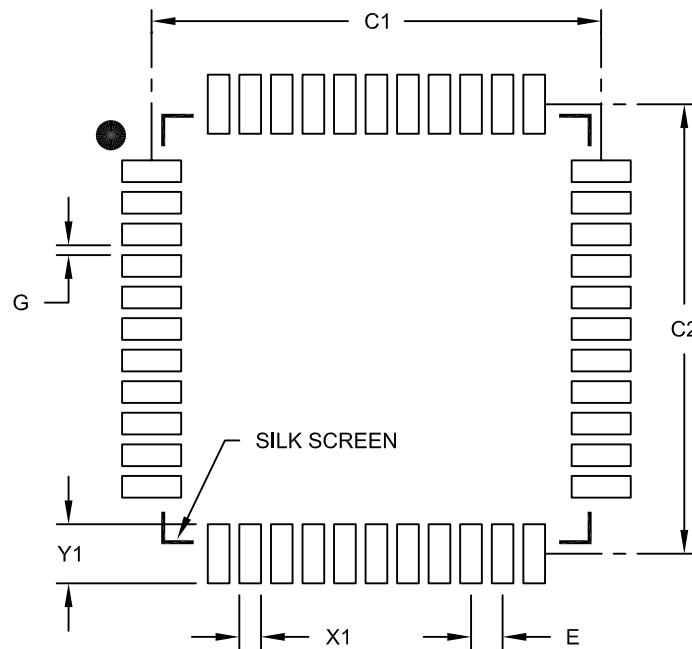
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-076B

44-Lead Plastic Thin Quad Flatpack (PT) 10X10X1 mm Body, 2.00 mm Footprint [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



RECOMMENDED LAND PATTERN

| | | Units | MILLIMETERS | | |
|--------------------------|----|-------|-------------|------|------|
| Dimension Limits | | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.80 | BSC | |
| Contact Pad Spacing | C1 | | 11.40 | | |
| Contact Pad Spacing | C2 | | 11.40 | | |
| Contact Pad Width (X44) | X1 | | | 0.55 | |
| Contact Pad Length (X44) | Y1 | | | | 1.50 |
| Distance Between Pads | G | 0.25 | | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2076B

PIC18(L)F2X/45K50

NOTES:

APPENDIX A: REVISION HISTORY

Revision A (08/2012)

Initial release.

Revision B (07/2014)

Updated Figures 2, 4 and 3-1; Updated Section 1.2 (Other Special Features), Section 2.4 (Voltage Regulator Pins (VUSB3v3)) and Section 26.9.1 (Dedicated ICD/ICSP Port); Added note to Section 24.4.1.1 (Buffer Ownership), Updated Tables 3-6 and 3-7; Updated Chapter 29.0 (Electrical Specifications), Chapter 31 (Packaging Information) and the Product Identification System page; Other minor corrections.

PIC18(L)F2X/45K50

APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in [Table B-1](#).

TABLE B-1: DEVICE DIFFERENCES

| Features | PIC18F24K50 | PIC18LF24K50 | PIC18F25K50 | PIC18LF25K50 | PIC18F45K50 | PIC18LF45K50 |
|---------------------------------|--|--|--|--|---|---|
| Program Memory (Bytes) | 16384 | 16384 | 32768 | 32768 | 32768 | 32768 |
| VDD Range | 2.3V to 5.5V | 1.8V to 3.6V | 2.3V to 5.5V | 1.8V to 3.6V | 2.3V to 5.5V | 1.8V to 3.6V |
| I/O Ports | Ports A, B, C, (E) | Ports A, B, C, D, E | Ports A, B, C, D, E |
| 10-Bit Analog-to-Digital Module | 14 input channels | 14 input channels | 14 input channels | 14 input channels | 25 input channels | 25 input channels |
| Packages | 28-pin SPDIP 28-pin SOIC 28-pin SSOP 28-pin QFN | 40-pin PDIP 40-pin UQFN 44-pin TQFP | 40-pin PDIP 40-pin UQFN 44-pin TQFP |

THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com. 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 web site at: <http://microchip.com/support>

PIC18(L)F2X/45K50

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| <u>PART NO.</u> | <u>[X]⁽²⁾</u> | - | X | /XX | XXX | |
|------------------------------|--|-------------------|---------|---------|-----|--|
| Device | Tape and Reel Option | Temperature Range | Package | Pattern | | |
| Device: | PIC18F45K50, PIC18LF45K50 PIC18F25K50, PIC18LF25K50 PIC18F24K50, PIC18LF24K50 | | | | | |
| Tape and Reel Option: | Blank = standard packaging (tube or tray) T = Tape and Reel ^{(1), (2)} | | | | | |
| Temperature Range: | I = -40°C to +85°C (Industrial) E = -40°C to +125°C (Extended) | | | | | |
| Package: | ML = QFN MV = UQFN P = PDIP PT = TQFP (Thin Quad Flatpack) SO = SOIC SP = Skinny Plastic DIP SS = SSOP | | | | | |
| Pattern: | QTP, SQTP, Code or Special Requirements (blank otherwise) | | | | | |

Examples:

- a) PIC18F45K50-E/P 301 = Extended temp., PDIP package, QTP pattern #301.
- b) PIC18LF25K50-E/SO = Extended temp., SOIC package.
- c) PIC18F45K50-E/P = Extended temp., PDIP package.
- d) PIC18F24K50T-E/ML = Tape and reel, Extended temp., QFN package.

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.

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, PIC³² 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, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, RightTouch logo, REAL ICE, SQL, 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.

© 2012-2014, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-63276-360-0

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.



MICROCHIP

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
<http://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 - 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

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

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: 886-7-213-7830

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 - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Pforzheim
Tel: 49-7231-424750

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