# PIC16(L)F1782/3 Data Sheet

## 28-Pin 8-Bit Advanced Analog Flash Microcontrollers

**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.

**Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC32 logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

Analog-for-the-Digital Age, Application Maestro, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

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

*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.*

**QUALITY MANAGEMENT SYSTEM**
**CERTIFIED BY DNV**
**═ ISO/TS 16949 ═**

# PIC16(L)F1782/3

## 28-Pin 8-Bit Advanced Analog Flash Microcontroller

### High-Performance RISC CPU:

- Only 49 Instructions
- Operating Speed:
  - DC – 32 MHz clock input
  - DC – 125 ns instruction cycle
- Interrupt Capability with Automatic Context Saving
- 16-Level Deep Hardware Stack with optional Overflow/Underflow Reset
- Direct, Indirect and Relative Addressing modes:
  - Two full 16-bit File Select Registers (FSRs)
  - FSRs can read program and data memory

### Memory Features:

- Up to 4 KW Flash Program Memory:
  - Self-programmable under software control
  - Programmable code protection
  - Programmable write protection
- 256 Bytes of Data EEPROM
- Up to 512 Bytes of RAM

### High Performance PWM Controller:

- Two Programmable Switch Mode Controller (PSMC) modules:
  - Digital and/or analog feedback control of PWM frequency and pulse begin/end times
  - 16-bit Period, Duty Cycle and Phase
  - 16 ns clock resolution
  - Supports Single PWM, Complementary, Push-Pull and 3-phase modes of operation
  - Dead-band control with 8-bit counter
  - Auto-shutdown and restart
  - Leading and falling edge blanking
  - Burst mode

### Extreme Low-Power Management PIC16LF1782/3 with XLP:

- Sleep mode: 50 nA @ 1.8V, typical
- Watchdog Timer: 500 nA @ 1.8V, typical
- Secondary Oscillator: 500 nA @ 32 kHz
- Operating Current:
  - 8 μA @ 32 kHz, 1.8V, typical
  - 32 μA/MHz @ 1.8V, typical

### Analog Peripheral Features:

- Analog-to-Digital Converter (ADC):
  - Fully differential 12-bit converter
  - 100 ksps conversion rate
  - 11 single-ended channels
  - 5 differential channels
  - Positive and negative reference selection
- 8-bit Digital-to-Analog Converter (DAC):
  - Output available externally
  - Positive and negative reference selection
  - Internal connections to comparators, op amps, Fixed Voltage Reference (FVR) and ADC
- Three High-Speed Comparators:
  - 50 ns response time @ $V_{DD}$ = 5V
  - Rail-to-rail inputs
  - Software selectable hysteresis
  - Internal connection to op amps, FVR and DAC
- Two Operational Amplifiers:
  - Rail-to-rail inputs/outputs
  - High/Low selectable Gain Bandwidth Product
  - Internal connection to DAC and FVR
- Fixed Voltage Reference (FVR):
  - 1.024V, 2.048V and 4.096V output levels
  - Internal connection to ADC, comparators and DAC

### I/O Features:

- Up to 24 I/O Pins and 1 Input-only Pin:
  - High current sink/source for LED drivers
  - Individually programmable interrupt-on-change pins
  - Individually programmable weak pull-ups
  - Individual input level selection
  - Individually programmable slew rate control
  - Individually programmable open drain outputs

# PIC16(L)F1782/3

## Digital Peripheral Features:

- Timer0: 8-Bit Timer/Counter with 8-Bit Programmable Prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Dedicated low-power 32 kHz oscillator driver
- Timer2: 8-Bit Timer/Counter with 8-Bit Period Register, Prescaler and Postscaler
- Two Capture/Compare/PWM modules (CCP):
  - 16-bit capture, maximum resolution 12.5 ns
  - 16-bit compare, max resolution 31.25 ns
  - 10-bit PWM, max frequency 32 kHz
- Master Synchronous Serial Port (SSP) with SPI and I²C™ with:
  - 7-bit address masking
  - SMBus/PMBus™ compatibility
- Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART):
  - RS-232, RS-485 and LIN compatible
  - Auto-baud detect
  - Auto-wake-up on start

## Oscillator Features:

- Operate up to 32 MHz from Precision Internal Oscillator:
  - Factory calibrated to ±1%, typical
  - Software selectable frequency range from 32 MHz to 31 kHz
- 31 kHz Low-Power Internal Oscillator
- 32.768 kHz Timer1 Oscillator:
  - Available as system clock
  - Low-power RTC
- External Oscillator Block with:
  - 4 crystal/resonator modes up to 32 MHz using 4x PLL
  - 3 external clock modes up to 32 MHz
- 4x Phase-Locked Loop (PLL)
- Fail-Safe Clock Monitor:
  - Detect and recover from external oscillator failure
- Two-Speed Start-up:
  - Minimize latency between code execution and external oscillator start-up

## General Microcontroller Features:

- Power-Saving Sleep mode
- Power-on Reset (POR)
- Power-up Timer (PWRT)
- Oscillator Start-up Timer (OST)
- Brown-out Reset (BOR) with Selectable Trip Point
- Extended Watchdog Timer (WDT)
- In-Circuit Serial Programming™ (ICSP™)
- In-Circuit Debug (ICD)
- Enhanced Low-Voltage Programming (LVP)
- Operating Voltage Range:
  - 1.8V to 3.6V (PIC16LF1782/3)
  - 2.3V to 5.5V (PIC16F1782/3)

## PIC16(L)F178X Family Types

| Device | Data Sheet Index | Program Memory Flash (words) | Data EEPROM (bytes) | Data SRAM (bytes) | I/O's[2] | 12-bit ADC (ch) | Comparators | Operational Amplifiers | 8-bit DAC | Timers (8/16-bit) | Programmable Switch Mode Controllers (PSMC) | CCP | EUSART | MSSP (I²C™/SPI) | Debug[1] | XLP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PIC16(L)F1782 | (1) | 2048 | 256 | 256 | 25 | 11 | 3 | 2 | 1 | 2/1 | 2 | 2 | 1 | 1 | I | Y |
| PIC16(L)F1783 | (1) | 4096 | 256 | 512 | 25 | 11 | 3 | 2 | 1 | 2/1 | 2 | 2 | 1 | 1 | I | Y |
| PIC16(L)F1784 | (2) | 4096 | 256 | 512 | 36 | 14 | 4 | 3 | 1 | 2/1 | 3 | 3 | 1 | 1 | I | Y |
| PIC16(L)F1786 | (2) | 8192 | 256 | 1024 | 25 | 11 | 4 | 2 | 1 | 2/1 | 3 | 3 | 1 | 1 | I | Y |
| PIC16(L)F1787 | (2) | 8192 | 256 | 1024 | 36 | 14 | 4 | 3 | 1 | 2/1 | 3 | 3 | 1 | 1 | I | Y |

**Note 1:** I - Debugging, Integrated on Chip; H - Debugging, available using Debug Header.
    **2:** One pin is input-only.

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

    **1:** DS41579    PIC16(L)F1782/3 Data Sheet, 28-Pin Flash, 8-bit Advanced Analog MCUs.
    **2:** Future Release    PIC16(L)F1784/6/7 Data Sheet, 28/40/44-Pin Flash, 8-bit Advanced Analog MCUs.

**FIGURE 1:**     **28-PIN DIAGRAM FOR PIC16(L)F1782/3**

**SPDIP, SOIC, SSOP**

| | | |
|---|---|---|
| $V_{PP}$/$\overline{MCLR}$/RE3 → | 1 | 28 ↔ RB7/ICSPDAT |
| RA0 ↔ | 2 | 27 ↔ RB6/ICSPCLK |
| RA1 ↔ | 3 | 26 ↔ RB5 |
| RA2 ↔ | 4 | 25 ↔ RB4 |
| RA3 ↔ | 5 | 24 ↔ RB3 |
| RA4 ↔ | 6 | 23 ↔ RB2 |
| RA5 ↔ | 7 | 22 ↔ RB1 |
| $V_{SS}$ → | 8 | 21 ↔ RB0 |
| RA7 ↔ | 9 | 20 ← $V_{DD}$ |
| RA6 ↔ | 10 | 19 ← $V_{SS}$ |
| RC0 ↔ | 11 | 18 ↔ RC7 |
| RC1 ↔ | 12 | 17 ↔ RC6 |
| RC2 ↔ | 13 | 16 ↔ RC5 |
| RC3 ↔ | 14 | 15 ↔ RC4 |
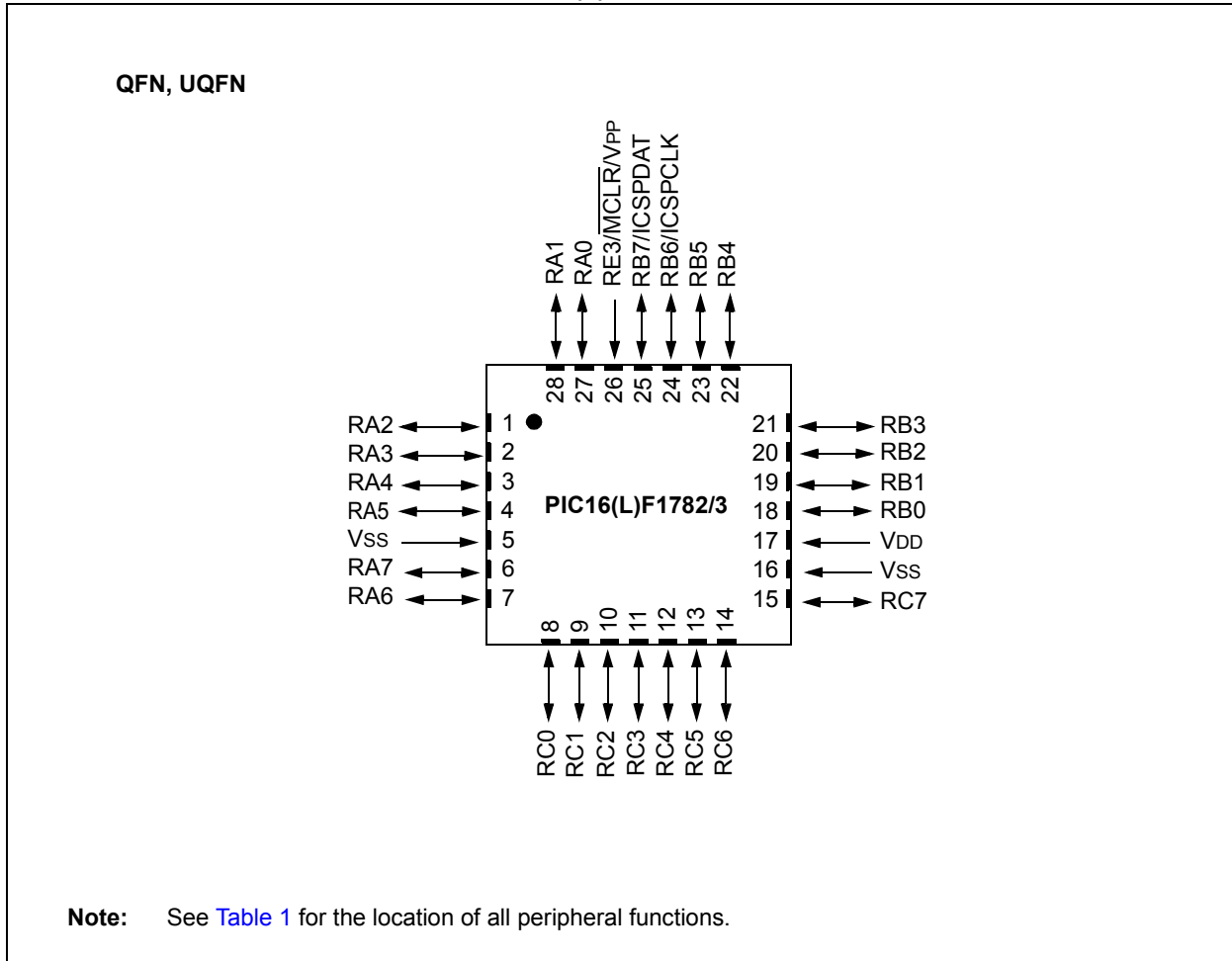
PIC16(L)F1782/3

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

# PIC16(L)F1782/3

**QFN, UQFN**



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

**Preliminary**
© 2011-2012 Microchip Technology Inc.

# PIC16(L)F1782/3

## TABLE 1: 28-PIN ALLOCATION TABLE (PIC16(L)F1782/3)

| I/O | 28-Pin SPDIP, SOIC, SSOP | 28-Pin QFN, UQFN | ADC | ADC Reference | Comparator | Operation Amplifiers | 8-bit DAC | Timers | PSMC | CCP | EUSART | MSSP | Interrupt | Pull-up | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 2 | 27 | AN0 | — | C1IN0-<br>C2IN0-<br>C3IN0- | — | — | — | — | — | — | — | IOC | Y | — |
| RA1 | 3 | 28 | AN1 | — | C1IN1-<br>C2IN1-<br>C3IN1- | OPA1OUT | — | — | — | — | — | — | IOC | Y | — |
| RA2 | 4 | 1 | AN2 | VREF- | C1IN0+<br>C2IN0+<br>C3IN0+ | — | DACOUT1<br>DACVREF- | — | — | — | — | — | IOC | Y | — |
| RA3 | 5 | 2 | AN3 | VREF1+ | C1IN1+ | — | DACVREF+ | — | — | — | — | — | IOC | Y | — |
| RA4 | 6 | 3 | — | — | C1OUT | OPA1IN+ | — | T0CKI | — | — | — | — | IOC | Y | — |
| RA5 | 7 | 4 | AN4 | — | C2OUT[1] | OPA1IN- | — | — | — | — | — | $\overline{SS}$ | IOC | Y | — |
| RA6 | 10 | 7 | — | — | C2OUT[2] | — | — | — | — | — | — | — | IOC | Y | OSC2/<br>CLKOUT |
| RA7 | 9 | 6 | — | VREF2+ | — | — | — | — | PSMC1CLK<br>PSMC2CLK | — | — | — | IOC | Y | OSC1/<br>CLKIN |
| RB0 | 21 | 18 | AN12 | — | C2IN1+ | — | — | — | PSMC1IN<br>PSMC2IN | CCP1[2] | — | — | INT/<br>IOC | Y | — |
| RB1 | 22 | 19 | AN10 | — | C1IN3-<br>C2IN3-<br>C3IN3- | OPA2OUT | — | — | — | — | — | — | IOC | Y | — |
| RB2 | 23 | 20 | AN8 | — | — | OPA2IN- | — | — | — | — | — | — | IOC | Y | CLKR |
| RB3 | 24 | 21 | AN9 | — | C1IN2-<br>C2IN2-<br>C3IN2- | OPA2IN+ | — | — | — | CCP2[2] | — | — | IOC | Y | — |
| RB4 | 25 | 22 | AN11 | — | C3IN1+ | — | — | — | — | — | — | — | IOC | Y | — |
| RB5 | 26 | 23 | AN13 | — | C3OUT | — | — | T1G | — | — | — | SDO[2] | IOC | Y | — |
| RB6 | 27 | 24 | — | — | — | — | — | — | — | — | TX[2]<br>CK[2] | SDI[2]<br>SDA[2] | IOC | Y | ICSPCLK |
| RB7 | 28 | 25 | — | — | — | — | DACOUT2 | — | — | — | RX[2]<br>DT[2] | SCK[2]<br>SCL[2] | IOC | Y | ICSPDAT |
| RC0 | 11 | 8 | — | — | — | — | — | T1OSO<br>T1CKI | PSMC1A | — | — | — | IOC | Y | — |
| RC1 | 12 | 9 | — | — | — | — | — | T1OSI | PSMC1B | CCP2[1] | — | — | IOC | Y | — |
| RC2 | 13 | 10 | — | — | — | — | — | — | PSMC1C | CCP1[1] | — | — | IOC | Y | — |
| RC3 | 14 | 11 | — | — | — | — | — | — | PSMC1D | — | — | SCK[1]<br>SCL[1] | IOC | Y | — |
| RC4 | 15 | 12 | — | — | — | — | — | — | PSMC1E | — | — | SDI[1]<br>SDA[1] | IOC | Y | — |
| RC5 | 16 | 13 | — | — | — | — | — | — | PSMC1F | — | — | SDO[1] | IOC | Y | — |
| RC6 | 17 | 14 | — | — | — | — | — | — | PSMC2A | — | TX[1]<br>CK[1] | — | IOC | Y | — |
| RC7 | 18 | 15 | — | — | — | — | — | — | PSMC2B | — | RX[1]<br>DT[1] | — | IOC | Y | — |
| RE3 | 1 | 26 | — | — | — | — | — | — | — | — | — | — | IOC | Y | $\overline{MCLR}$/<br>VPP |
| VDD | 20 | 17 | — | — | — | — | — | — | — | — | — | — | — | — | VDD |
| VSS | 8, 19 | 5, 16 | — | — | — | — | — | — | — | — | — | — | — | — | VSS |

Note 1: Default pin assignment.
2: Alternate pin assignment that can be selected via software.

# PIC16(L)F1782/3

**Table of Contents**

## TO OUR VALUED CUSTOMERS

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

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

### Most Current Data Sheet

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

> http://www.microchip.com

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

### Errata

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

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

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

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

### Customer Notification System

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

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

# 1.0 DEVICE OVERVIEW

The PIC16(L)F1782/3 are described within this data sheet. They are available in 28-pin packages. Figure 1-1 shows a block diagram of the PIC16(L)F1782/3 devices. Table 1-2 shows the pinout descriptions.

Reference Table 1-1 for peripherals available per device.

**TABLE 1-1: DEVICE PERIPHERAL SUMMARY**

| Peripheral | PIC16(L)F1782 | PIC16(L)F1783 |
|---|:---:|:---:|
| Analog-to-Digital Converter (ADC) | ● | ● |
| Digital-to-Analog Converter (DAC) | ● | ● |
| Fixed Voltage Reference (FVR) | ● | ● |
| Reference Clock Module | ● | ● |
| Temperature Indicator | ● | ● |
| Capture/Compare/PWM (CCP/ECCP) Modules | | |
| CCP1 | ● | ● |
| CCP2 | ● | ● |
| Comparators | | |
| C1 | ● | ● |
| C2 | ● | ● |
| C3 | ● | ● |
| Enhanced Universal Synchronous/Asynchronous Receiver/Transmitter (EUSART) | | |
| EUSART | ● | ● |
| Master Synchronous Serial Ports | | |
| MSSP | ● | ● |
| Op Amp | | |
| Op Amp 1 | ● | ● |
| Op Amp 2 | ● | ● |
| Programmable Switch Mode Controller (PSMC) | | |
| PSMC1 | ● | ● |
| PSMC2 | ● | ● |
| Timers | | |
| Timer0 | ● | ● |
| Timer1 | ● | ● |
| Timer2 | ● | ● |

# PIC16(L)F1782/3

**FIGURE 1-1:** **PIC16(L)F1782/3 BLOCK DIAGRAM**



**Note 1:** See applicable chapters for more information on peripherals.

**Preliminary** © 2011-2012 Microchip Technology Inc.
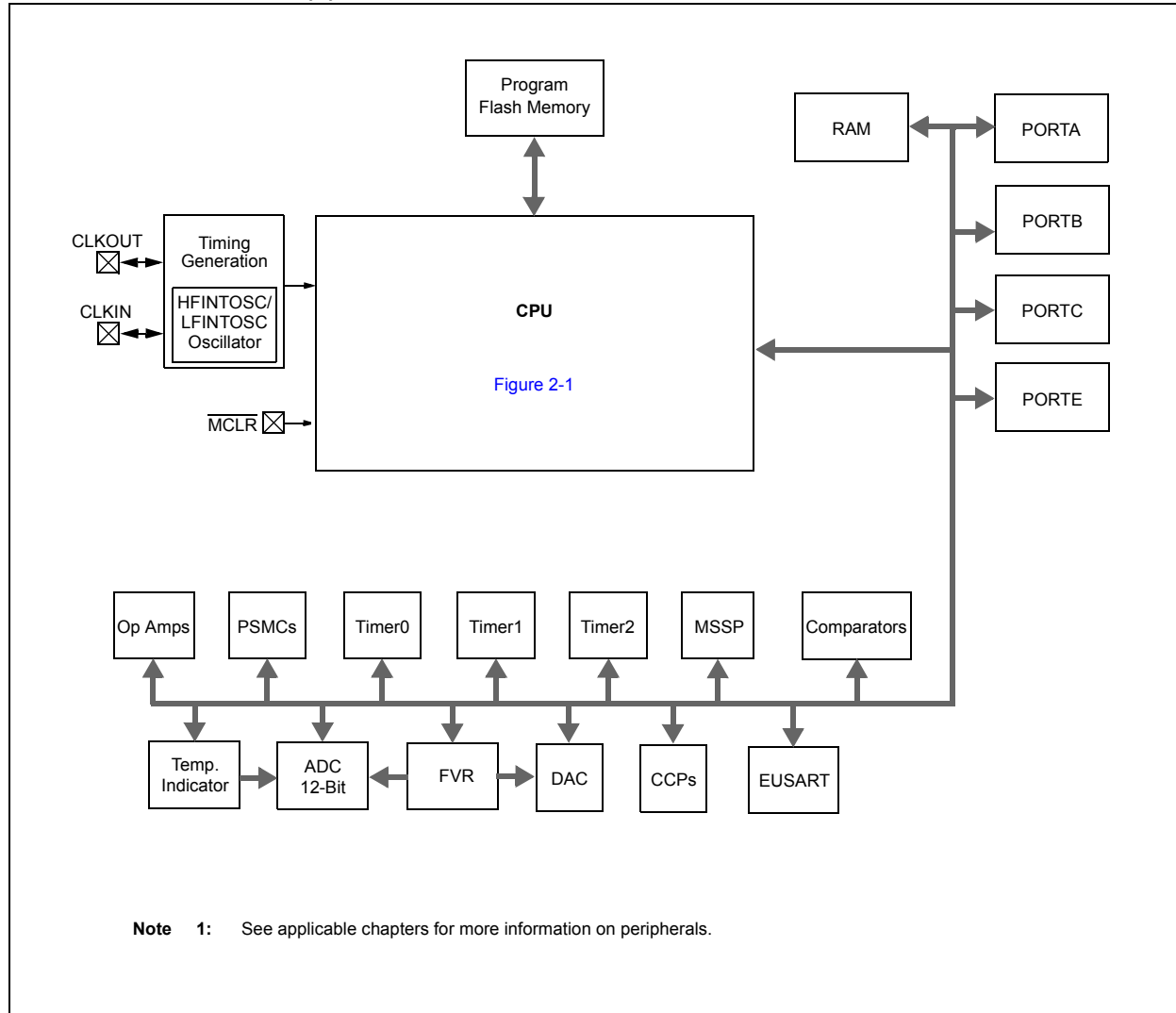
### TABLE 1-2: PIC16(L)F1782/3 PINOUT DESCRIPTION

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RA0/AN0/C1IN0-/C2IN0-/C3IN0- | RA0 | TTL/ST | CMOS | General purpose I/O. |
| | AN0 | AN | — | A/D Channel 0 input. |
| | C1IN0- | AN | — | Comparator C1 negative input. |
| | C2IN0- | AN | — | Comparator C2 negative input. |
| | C3IN0- | AN | — | Comparator C3 negative input. |
| RA1/AN1/C1IN1-/C2IN1-/C3IN1-/OPA1OUT | RA1 | TTL/ST | CMOS | General purpose I/O. |
| | AN1 | AN | — | A/D Channel 1 input. |
| | C1IN1- | AN | — | Comparator C1 negative input. |
| | C2IN1- | AN | — | Comparator C2 negative input. |
| | C3IN1- | AN | — | Comparator C3 negative input. |
| | OPA1OUT | — | AN | Operational Amplifier 1 output. |
| RA2/AN2/C1IN0+/C2IN0+/C3IN0+/DACOUT1/V$_{REF}$-/DACV$_{REF}$- | RA2 | TTL/ST | CMOS | General purpose I/O. |
| | AN2 | AN | — | A/D Channel 2 input. |
| | C1IN0+ | AN | — | Comparator C1 positive input. |
| | C2IN0+ | AN | — | Comparator C2 positive input. |
| | C3IN0+ | AN | — | Comparator C3 positive input. |
| | DACOUT | — | AN | Digital-to-Analog Converter output. |
| | V$_{REF}$- | AN | — | A/D Negative Voltage Reference input. |
| | DACV$_{REF}$- | AN | — | Digital-to-Analog Converter negative reference. |
| RA3/AN3/V$_{REF}$+[1]/C1IN1+/DACV$_{REF}$+ | RA3 | TTL/ST | CMOS | General purpose I/O. |
| | AN3 | AN | — | A/D Channel 3 input. |
| | V$_{REF}$+ | AN | — | A/D Voltage Reference input. |
| | C1IN1+ | AN | — | Comparator C1 positive input. |
| | DACV$_{REF}$+ | AN | — | Digital-to-Analog Converter positive reference. |
| RA4/C1OUT/OPA1IN+/T0CKI | RA4 | TTL/ST | CMOS | General purpose I/O. |
| | C1OUT | — | CMOS | Comparator C1 output. |
| | OPA1IN+ | AN | — | Operational Amplifier 1 non-inverting input. |
| | T0CKI | ST | — | Timer0 clock input. |
| RA5/AN4/C2OUT[1]/OP1INA-/$\overline{SS}$ | RA5 | TTL/ST | CMOS | General purpose I/O. |
| | AN4 | AN | — | A/D Channel 4 input. |
| | C2OUT | — | CMOS | Comparator C2 output. |
| | OPA1IN- | AN | — | Operational Amplifier 1 inverting input. |
| | $\overline{SS}$ | ST | — | Slave Select input. |
| RA6/C2OUT/OSC2/CLKOUT | RA6 | TTL/ST | CMOS | General purpose I/O. |
| | C2OUT | — | CMOS | Comparator C2 output. |
| | OSC2 | — | XTAL | Crystal/Resonator (LP, XT, HS modes). |
| | CLKOUT | — | CMOS | F$_{OSC}$/4 output. |

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

**Note 1:** Pin functions can be assigned to one of two locations via software. See Register 13-1.

**2:** All pins have Interrupt-on-Change functionality.

**TABLE 1-2:** **PIC16(L)F1782/3 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RA7/V<sub>REF</sub>+(1)/PSMC1CLK/ PSMC2CLK/OSC1/CLKIN | RA7 | TTL/ST | CMOS | General purpose I/O. |
| | VREF+ | AN | — | A/D Voltage Reference input. |
| | PSMC1CLK | ST | — | PSMC1 clock input. |
| | PSMC2CLK | ST | — | PSMC2 clock input. |
| | OSC1 | — | XTAL | Crystal/Resonator (LP, XT, HS modes). |
| | CLKIN | st | — | External clock input (EC mode). |
| RB0/AN12/C2IN1+/PSMC1IN/ PSMC2IN/CCP1(1)/INT | RB0 | TTL/ST | CMOS | General purpose I/O. |
| | AN12 | AN | — | A/D Channel 12 input. |
| | C2IN1+ | AN | — | Comparator C2 positive input. |
| | PSMC1IN | ST | — | PSMC1 Event Trigger input. |
| | PSMC2IN | ST | — | PSMC2 Event Trigger input. |
| | CCP1 | ST | CMOS | Capture/Compare/PWM1. |
| | INT | ST | — | External interrupt. |
| RB1/AN10/C1IN3-/C2IN3-/ C3IN3-/OPA2OUT | RB1 | TTL/ST | CMOS | General purpose I/O. |
| | AN10 | AN | — | A/D Channel 10 input. |
| | C1IN3- | AN | — | Comparator C1 negative input. |
| | C2IN3- | AN | — | Comparator C2 negative input. |
| | C3IN3- | AN | — | Comparator C3 negative input. |
| | OPA2OUT | — | AN | Operational Amplifier 2 output. |
| RB2/AN8/OPA2IN-/CLKR | RB2 | TTL/ST | CMOS | General purpose I/O. |
| | AN8 | AN | — | A/D Channel 8 input. |
| | OPA2IN- | AN | — | Operational Amplifier 2 inverting input. |
| | CLKR | — | CMOS | Clock output. |
| RB3/AN9/C1IN2-/C2IN2-/ C3IN2-/OPA2IN+/CCP2(1) | RB3 | TTL/ST | CMOS | General purpose I/O. |
| | AN9 | AN | — | A/D Channel 9 input. |
| | C1IN2- | AN | — | Comparator C1 negative input. |
| | C2IN2- | AN | — | Comparator C2 negative input. |
| | C3IN2- | AN | — | Comparator C3 negative input. |
| | OPA2IN+ | AN | — | Operational Amplifier 2 non-inverting input. |
| | CCP2 | ST | CMOS | Capture/Compare/PWM2. |
| RB4/AN11/C3IN1+ | RB4 | TTL/ST | CMOS | General purpose I/O. |
| | AN11 | AN | — | A/D Channel 11 input. |
| | C3IN1+ | AN | — | Comparator C3 positive input. |
| RB5/AN13/C3OUT/T1G/SDO(1) | RB5 | TTL/ST | CMOS | General purpose I/O. |
| | AN13 | AN | — | A/D Channel 13 input. |
| | C3OUT | — | CMOS | Comparator C3 output. |
| | T1G | ST | — | Timer1 gate input. |
| | SDO | — | CMOS | SPI data output. |

**Legend:** AN = Analog input or output  CMOS= CMOS compatible input or output  OD = Open Drain
TTL = TTL compatible input  ST = Schmitt Trigger input with CMOS levels  I$^2$C™ = Schmitt Trigger input with I$^2$C
HV = High Voltage  XTAL = Crystal  levels
**Note  1:** Pin functions can be assigned to one of two locations via software. See Register 13-1.
  **2:** All pins have Interrupt-on-Change functionality.

**TABLE 1-2: PIC16(L)F1782/3 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RB6/TX[1]/CK[1]/SDI[1]/SDA[1]/ ICSPCLK | RB6 | TTL/ST | CMOS | General purpose I/O. |
| | TX | — | CMOS | USART asynchronous transmit. |
| | CK | ST | CMOS | USART synchronous clock. |
| | SDI | ST | — | SPI data input. |
| | SDA | I²C | OD | I²C™ data input/output. |
| | ICSPCLK | ST | — | Serial Programming Clock. |
| RB7/DACOUT2/RX[1]/DT[1]/ SCK[1]/SCL[1]/ICSPDAT | RB7 | TTL/ST | CMOS | General purpose I/O. |
| | DACOUT2 | — | AN | Voltage Reference output. |
| | RX | ST | — | USART asynchronous input. |
| | DT | ST | CMOS | USART synchronous data. |
| | SCK | ST | CMOS | SPI clock. |
| | SCL | I²C | OD | I²C™ clock. |
| | ICSPDAT | ST | CMOS | ICSP™ Data I/O. |
| RC0/T1OSO/T1CKI/PSMC1A | RC0 | TTL/ST | CMOS | General purpose I/O. |
| | T1OSO | XTAL | XTAL | Timer1 oscillator connection. |
| | T1CKI | ST | — | Timer1 clock input. |
| | PSMC1A | — | CMOS | PSMC1 output A. |
| RC1/T1OSI/PSMC1B/CCP2[1] | RC1 | TTL/ST | CMOS | General purpose I/O. |
| | T1OSI | XTAL | XTAL | Timer1 oscillator connection. |
| | PSMC1B | — | CMOS | PSMC1 output B. |
| | CCP2 | ST | CMOS | Capture/Compare/PWM2. |
| RC2/PSMC1C/CCP1[1] | RC2 | TTL/ST | CMOS | General purpose I/O. |
| | PSMC1C | — | CMOS | PSMC1 output C. |
| | CCP1 | ST | CMOS | Capture/Compare/PWM1. |
| RC3/PSMC1D/SCK[1]/SCL[1] | RC3 | TTL/ST | CMOS | General purpose I/O. |
| | PSMC1D | — | CMOS | PSMC1 output D. |
| | SCK | ST | CMOS | SPI clock. |
| | SCL | I²C | OD | I²C™ clock. |
| RC4/PSMC1E/SDI[1]/SDA[1] | RC4 | TTL/ST | CMOS | General purpose I/O. |
| | PSMC1E | — | CMOS | PSMC1 output E. |
| | SDI | ST | — | SPI data input. |
| | SDA | I²C | OD | I²C™ data input/output. |
| RC5/PSMC1F/SDO[1] | RC5 | TTL/ST | CMOS | General purpose I/O. |
| | PSMC1F | — | CMOS | PSMC1 output F. |
| | SDO | — | CMOS | SPI data output. |
| RC6/PSMC2A/TX[1]/CK[1] | RC6 | TTL/ST | CMOS | General purpose I/O. |
| | PSMC2A | — | CMOS | PSMC2 output A. |
| | TX | — | CMOS | USART asynchronous transmit. |
| | CK | ST | CMOS | USART synchronous clock. |
| RC7/PSMC2B/RX[1]/DT[1] | RC7 | TTL/ST | CMOS | General purpose I/O. |
| | PSMC2B | — | CMOS | PSMC2 output B. |
| | RX | ST | — | USART asynchronous input. |
| | DT | ST | CMOS | USART synchronous data. |

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

**Note 1:** Pin functions can be assigned to one of two locations via software. See Register 13-1.
**2:** All pins have Interrupt-on-Change functionality.

**TABLE 1-2: PIC16(L)F1782/3 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RE3/MCLR/Vpp | RE3 | TTL/ST | — | General purpose input. |
| | MCLR | ST | — | Master Clear with internal pull-up. |
| | Vpp | HV | — | Programming voltage. |
| Vdd | Vdd | Power | — | Positive supply. |
| Vss | Vss | Power | — | Ground reference. |

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

**Note 1:** Pin functions can be assigned to one of two locations via software. See Register 13-1.
**2:** All pins have Interrupt-on-Change functionality.

## 2.0 ENHANCED MID-RANGE CPU

This family of devices contain an enhanced mid-range 8-bit CPU core. The CPU has 49 instructions. Interrupt capability includes automatic context saving. The hardware stack is 16 levels deep and has Overflow and Underflow Reset capability. Direct, Indirect, and Relative addressing modes are available. Two File Select Registers (FSRs) provide the ability to read program and data memory.

- Automatic Interrupt Context Saving
- 16-level Stack with Overflow and Underflow
- File Select Registers
- Instruction Set

### 2.1 Automatic Interrupt Context Saving

During interrupts, certain registers are automatically saved in shadow registers and restored when returning from the interrupt. This saves stack space and user code. See **8.5 "Automatic Context Saving"**, for more information.

### 2.2 16-level Stack with Overflow and Underflow

These devices have an external stack memory 15 bits wide and 16 words deep. A Stack Overflow or Underflow will set the appropriate bit (STKOVF or STKUNF) in the PCON register, and if enabled will cause a software Reset. See **Section 3.5 "Stack"** for more details.

### 2.3 File Select Registers

There are two 16-bit File Select Registers (FSR). FSRs can access all file registers and program memory, which allows one Data Pointer for all memory. When an FSR points to program memory, there is one additional instruction cycle in instructions using INDF to allow the data to be fetched. General purpose memory can now also be addressed linearly, providing the ability to access contiguous data larger than 80 bytes. There are also new instructions to support the FSRs. See **Section 3.6 "Indirect Addressing"** for more details.

### 2.4 Instruction Set

There are 49 instructions for the enhanced mid-range CPU to support the features of the CPU. See **Section 29.0 "Instruction Set Summary"** for more details.

---

# PIC16(L)F1782/3

**FIGURE 2-1:** **CORE BLOCK DIAGRAM**
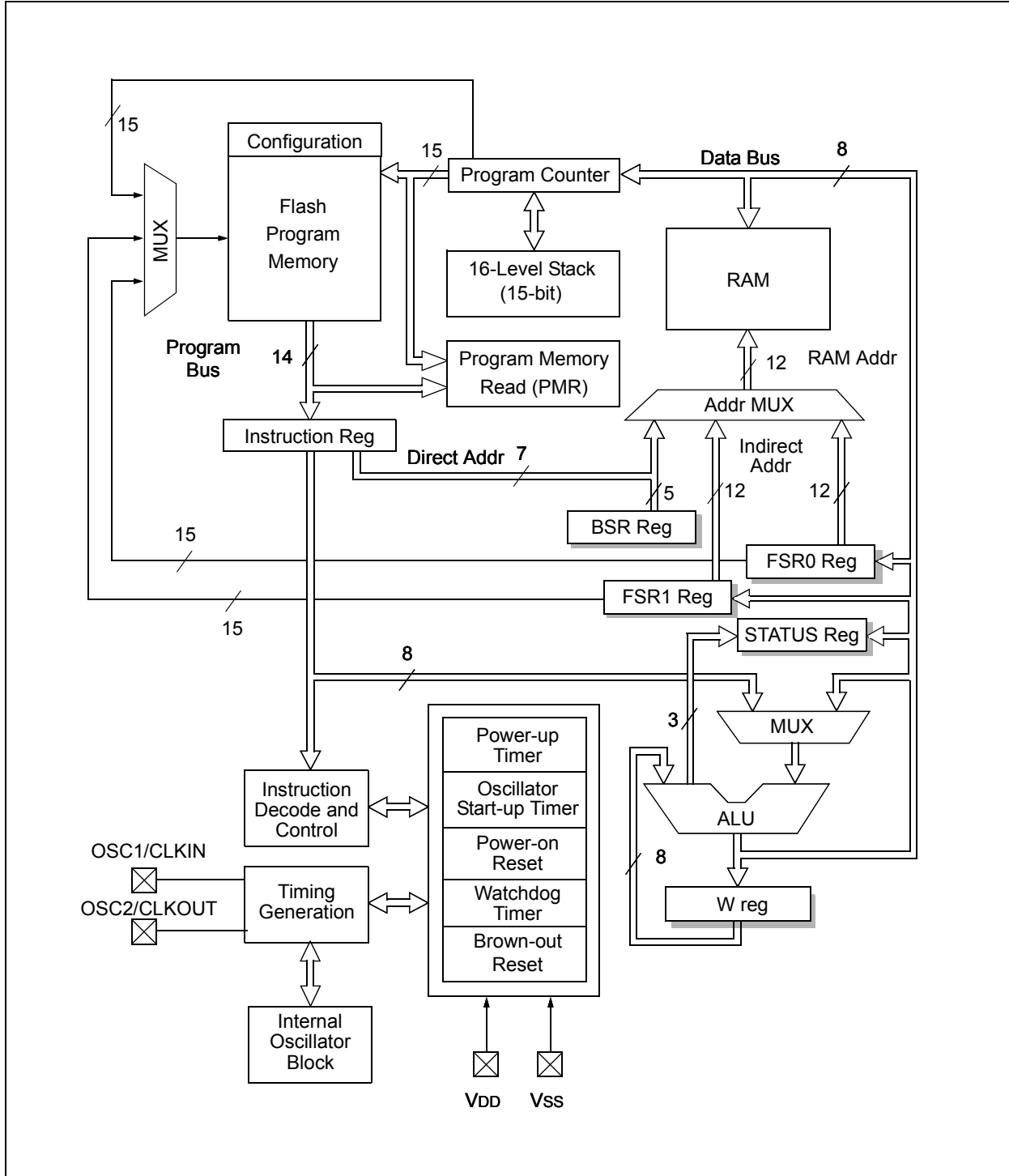
## 3.0    MEMORY ORGANIZATION

These devices contain the following types of memory:

- Program Memory
  - Configuration Words
  - Device ID
  - User ID
  - Flash Program Memory
- Data Memory
  - Core Registers
  - Special Function Registers
  - General Purpose RAM
  - Common RAM
- Data EEPROM memory[1]

> **Note 1:** The Data EEPROM Memory and the method to access Flash memory through the EECON registers is described in **Section 12.0 "Data EEPROM and Flash Program Memory Control"**.

The following features are associated with access and control of program memory and data memory:

- PCL and PCLATH
- Stack
- Indirect Addressing

### 3.1    Program Memory Organization

The enhanced mid-range core has a 15-bit program counter capable of addressing a 32K x 14 program memory space. Table 3-1 shows the memory sizes implemented for the PIC16(L)F1782/3 family. Accessing a location above these boundaries will cause a wrap-around within the implemented memory space. The Reset vector is at 0000h and the interrupt vector is at 0004h (see Figures 3-1, and 3-2).

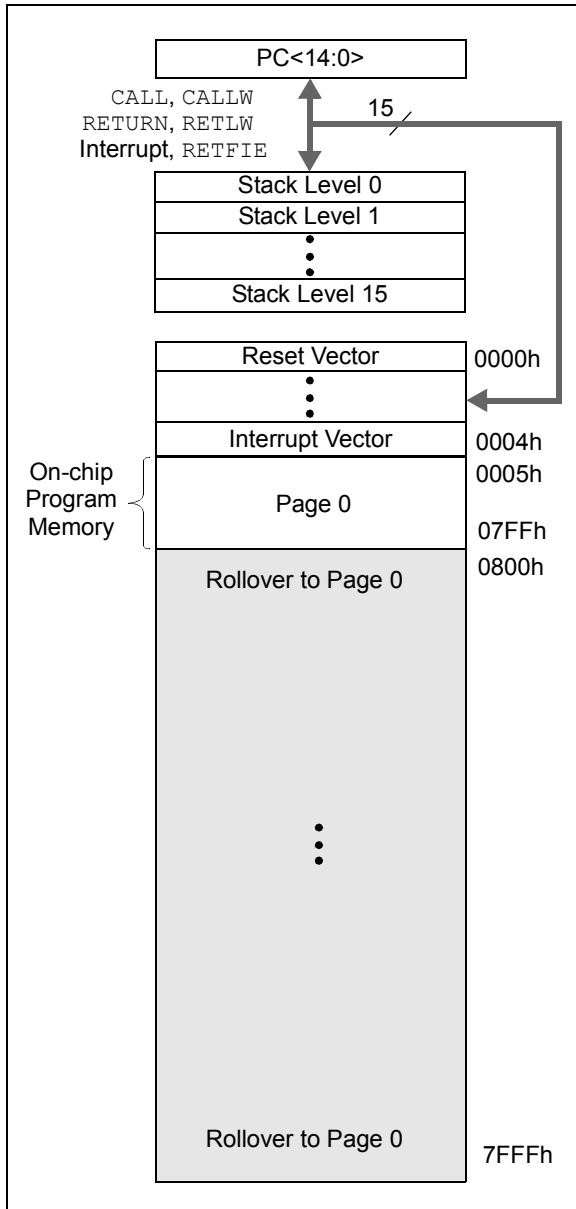**TABLE 3-1:    DEVICE SIZES AND ADDRESSES**

| Device | Program Memory Space (Words) | Last Program Memory Address |
|---|---|---|
| PIC16(L)F1782 | 2,048 | 07FFh |
| PIC16(L)F1783 | 4,096 | 0FFFh |

# PIC16(L)F1782/3

**FIGURE 3-1:** PROGRAM MEMORY MAP
AND STACK FOR
PIC16F1782

```
┌─────────────────────────────────┐
│          PC<14:0>               │
│  CALL, CALLW    15 /            │
│  RETURN, RETLW                  │
│  Interrupt, RETFIE              │
│  ┌───────────────────────────┐  │
│  │     Stack Level 0         │  │
│  │     Stack Level 1         │  │
│  │          ·                │  │
│  │     Stack Level 15        │  │
│  └───────────────────────────┘  │
│                                 │
│  ┌───────────────────────────┐  │
│  │     Reset Vector    0000h │  │
│  │          ·                │  │
│  │   Interrupt Vector  0004h │  │
│  │                     0005h │  │
│  │      Page 0               │  │
│  │                     07FFh │  │
│  │                     0800h │  │
│  │  Rollover to Page 0       │  │
│  │                           │  │
│  │          ·                │  │
│  │          ·                │  │
│  │          ·                │  │
│  │  Rollover to Page 0 7FFFh │  │
│  └───────────────────────────┘  │
│  On-chip Program Memory         │
└─────────────────────────────────┘
```

**FIGURE 3-2:** PROGRAM MEMORY MAP
AND STACK FOR
PIC16F1783

```
┌─────────────────────────────────┐
│          PC<14:0>               │
│  CALL, CALLW    15 /            │
│  RETURN, RETLW                  │
│  Interrupt, RETFIE              │
│  ┌───────────────────────────┐  │
│  │     Stack Level 0         │  │
│  │     Stack Level 1         │  │
│  │          ·                │  │
│  │     Stack Level 15        │  │
│  └───────────────────────────┘  │
│                                 │
│  ┌───────────────────────────┐  │
│  │     Reset Vector    0000h │  │
│  │          ·                │  │
│  │   Interrupt Vector  0004h │  │
│  │                     0005h │  │
│  │      Page 0               │  │
│  │                     07FFh │  │
│  │                     0800h │  │
│  │      Page 1               │  │
│  │                     0FFFh │  │
│  │                     1000h │  │
│  │  Rollover to Page 0       │  │
│  │          ·                │  │
│  │          ·                │  │
│  │          ·                │  │
│  │  Rollover to Page 1 7FFFh │  │
│  └───────────────────────────┘  │
│  On-chip Program Memory         │
└─────────────────────────────────┘
```

**Preliminary**                    © 2011-2012 Microchip Technology Inc.
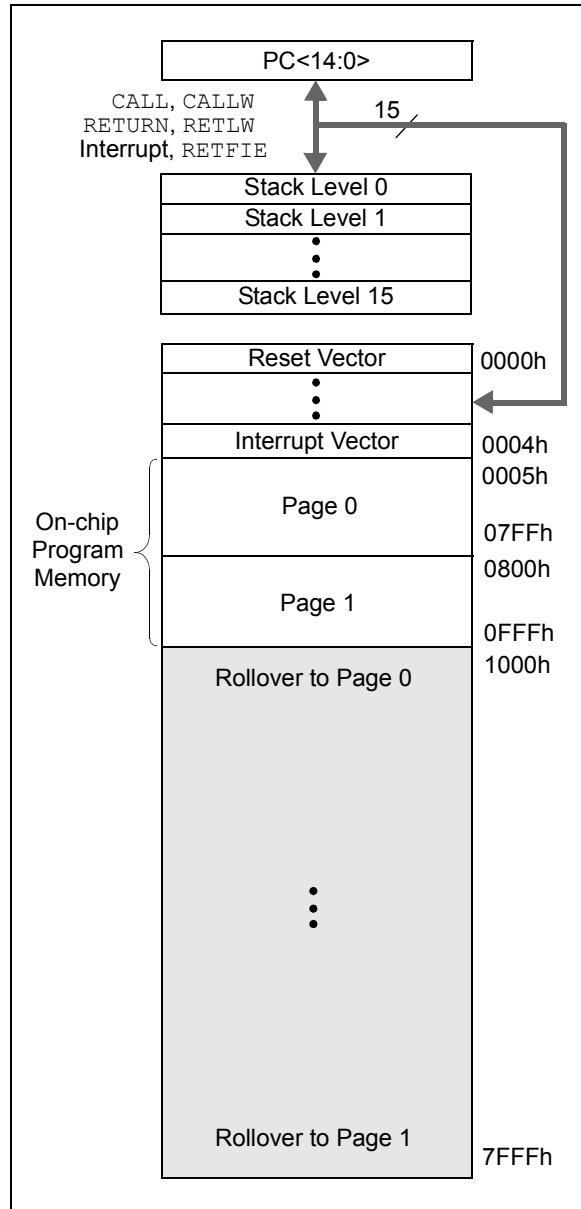
### 3.1.1 READING PROGRAM MEMORY AS DATA

There are two methods of accessing constants in program memory. The first method is to use tables of `RETLW` instructions. The second method is to set an FSR to point to the program memory.

#### 3.1.1.1 `RETLW` Instruction

The `RETLW` instruction can be used to provide access to tables of constants. The recommended way to create such a table is shown in Example 3-1.

**EXAMPLE 3-1: `RETLW` INSTRUCTION**

```
constants
    BRW                 ;Add Index in W to
                        ;program counter to
                        ;select data
    RETLW DATA0         ;Index0 data
    RETLW DATA1         ;Index1 data
    RETLW DATA2
    RETLW DATA3


my_function
    ;… LOTS OF CODE…
    MOVLW       DATA_INDEX
    call constants
    ;… THE CONSTANT IS IN W
```

The `BRW` instruction makes this type of table very simple to implement. If your code must remain portable with previous generations of microcontrollers, then the `BRW` instruction is not available so the older table read method must be used.

#### 3.1.1.2 Indirect Read with FSR

The program memory can be accessed as data by setting bit 7 of the FSRxH register and reading the matching INDFx register. The `MOVIW` instruction will place the lower 8 bits of the addressed word in the W register. Writes to the program memory cannot be performed via the INDF registers. Instructions that access the program memory via the FSR require one extra instruction cycle to complete. Example 3-2 demonstrates accessing the program memory via an FSR.

The HIGH directive will set bit<7> if a label points to a location in program memory.

**EXAMPLE 3-2: ACCESSING PROGRAM MEMORY VIA FSR**

```
constants
    retlw DATA0         ;Index0 data
    retlw DATA1         ;Index1 data
    retlw DATA2
    retlw DATA3
my_function
    ;… LOTS OF CODE…
    movlw  LOW constants
    movwf  FSR1L
    movlw  HIGH constants
    movwf  FSR1H
    moviw 0[INDF1]
;THE PROGRAM MEMORY IS IN W
```

## 3.2 Data Memory Organization

The data memory is partitioned in 32 memory banks with 128 bytes in a bank. Each bank consists of (Figure 3-3):

- 12 core registers
- 20 Special Function Registers (SFR)
- Up to 80 bytes of General Purpose RAM (GPR)
- 16 bytes of common RAM

The active bank is selected by writing the bank number into the Bank Select Register (BSR). Unimplemented memory will read as '0'. All data memory can be accessed either directly (via instructions that use the file registers) or indirectly via the two File Select Registers (FSR). See **Section 3.6 "Indirect Addressing"** for more information.

Data Memory uses a 12-bit address. The upper 7-bits of the address define the Bank address and the lower 5-bits select the registers/RAM in that bank.

### 3.2.1 CORE REGISTERS

The core registers contain the registers that directly affect the basic operation. The core registers occupy the first 12 addresses of every data memory bank (addresses x00h/x08h through x0Bh/x8Bh). These registers are listed below in Table 3-2. For detailed information, see Table 3-7.

**TABLE 3-2: CORE REGISTERS**

| Addresses | BANKx |
|---|---|
| x00h or x80h | INDF0 |
| x01h or x81h | INDF1 |
| x02h or x82h | PCL |
| x03h or x83h | STATUS |
| x04h or x84h | FSR0L |
| x05h or x85h | FSR0H |
| x06h or x86h | FSR1L |
| x07h or x87h | FSR1H |
| x08h or x88h | BSR |
| x09h or x89h | WREG |
| x0Ah or x8Ah | PCLATH |
| x0Bh or x8Bh | INTCON |

### 3.2.1.1 STATUS Register

The STATUS register, shown in Register 3-1, contains:

- the arithmetic status of the ALU
- the Reset status

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, CLRF STATUS will clear the upper three bits and set the Z bit. This leaves the STATUS register as '000u u1uu' (where u = unchanged).

It is recommended, therefore, that only BCF, BSF, SWAPF and MOVWF instructions are used to alter the STATUS register, because these instructions do not affect any Status bits. For other instructions not affecting any Status bits (Refer to **Section 29.0 "Instruction Set Summary"**).

| Note: | The C and DC bits operate as $\overline{Borrow}$ and $\overline{Digit\ Borrow}$ out bits, respectively, in subtraction. |
|---|---|

## 3.3 Register Definitions: Status

### REGISTER 3-1: STATUS: STATUS REGISTER

| U-0 | U-0 | U-0 | R-1/q | R-1/q | R/W-0/u | R/W-0/u | R/W-0/u |
|---|---|---|---|---|---|---|---|
| — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC[1] | C[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 | q = Value depends on condition |

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **$\overline{TO}$:** Time-out bit

1 = After power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$:** Power-down bit

1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z:** Zero bit

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

bit 1 **DC:** Digit Carry/$\overline{Digit\ Borrow}$ 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/$\overline{Borrow}$ bit[1] (ADDWF, ADDLW, SUBLW, SUBWF instructions)[1]

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

**Note 1:** For $\overline{Borrow}$, 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.

# PIC16(L)F1782/3

## 3.3.1 SPECIAL FUNCTION REGISTER

The Special Function Registers are registers used by the application to control the desired operation of peripheral functions in the device. The Special Function Registers occupy the 20 bytes after the core registers of every data memory bank (addresses x0Ch/x8Ch through x1Fh/x9Fh). The registers associated with the operation of the peripherals are described in the appropriate peripheral chapter of this data sheet.

## 3.3.2 GENERAL PURPOSE RAM

There are up to 80 bytes of GPR in each data memory bank. The Special Function Registers occupy the 20 bytes after the core registers of every data memory bank (addresses x0Ch/x8Ch through x1Fh/x9Fh).

### 3.3.2.1 Linear Access to GPR

The general purpose RAM can be accessed in a non-banked method via the FSRs. This can simplify access to large memory structures. See **Section 3.6.2 "Linear Data Memory"** for more information.

## 3.3.3 COMMON RAM

There are 16 bytes of common RAM accessible from all banks.

**FIGURE 3-3: BANKED MEMORY PARTITIONING**
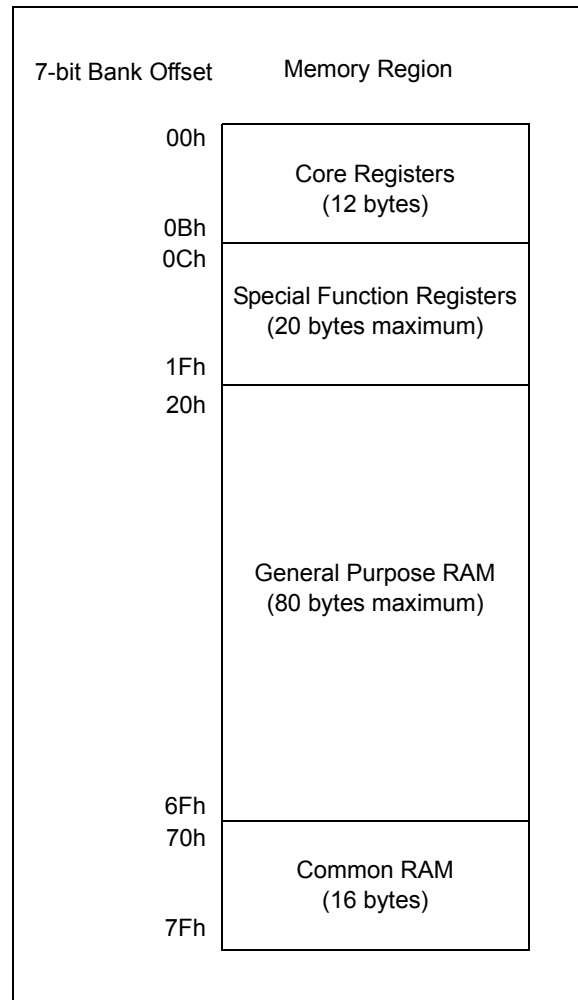


## 3.3.4 DEVICE MEMORY MAPS

The memory maps for the device family are as shown in Table 3-3.

**Preliminary**

© 2011-2012 Microchip Technology Inc.

**TABLE 3-3: PIC16(L)F1782/3 MEMORY MAP (BANKS 0-7)**

| Offset | BANK 0 | BANK 1 | BANK 2 | BANK 3 | BANK 4 | BANK 5 | BANK 6 | BANK 7 |
|---|---|---|---|---|---|---|---|---|
| Base | 000h | 080h | 100h | 180h | 200h | 280h | 300h | 380h |
| x00h–x0Bh | Core Registers (Table 3-2) | Core Registers (Table 3-2) | Core Registers (Table 3-2) | Core Registers (Table 3-2) | Core Registers (Table 3-2) | Core Registers (Table 3-2) | Core Registers (Table 3-2) | Core Registers (Table 3-2) |
| x0Ch | PORTA | TRISA | LATA | ANSELA | WPUA | ODCONA | SLRCONA | INLVLA |
| x0Dh | PORTB | TRISB | LATB | ANSELB | WPUB | ODCONB | SLRCONB | INLVLB |
| x0Eh | PORTC | TRISC | LATC | — | WPUC | ODCONC | SLRCONC | INLVLC |
| x0Fh | — | — | — | — | — | — | — | — |
| x10h | PORTE | TRISE | — | — | WPUE | — | — | INLVLE |
| x11h | PIR1 | PIE1 | CM1CON0 | EEADRL | SSPBUF | CCPR1L | — | IOCAP |
| x12h | PIR2 | PIE2 | CM1CON1 | EEADRH | SSPADD | CCPR1H | — | IOCAN |
| x13h | — | — | CM2CON0 | EEDATL | SSPMSK | CCPR1CON | — | IOCAF |
| x14h | PIR4 | PIE4 | CM2CON1 | EEDATH | SSPSTAT | — | — | IOCBP |
| x15h | TMR0 | OPTION_REG | CMOUT | EECON1 | SSPCON | — | — | IOCBN |
| x16h | TMR1L | PCON | BORCON | EECON2 | SSPCON2 | — | — | IOCBF |
| x17h | TMR1H | WDTCON | FVRCON | VREGCON | SSPCON3 | — | — | IOCCP |
| x18h | T1CON | OSCTUNE | DACCON0 | — | — | CCPR2L | — | IOCCN |
| x19h | T1GCON | OSCCON | DACCON1 | RCREG | — | CCPR2H | — | IOCCF |
| x1Ah | TMR2 | OSCSTAT | — | TXREG | — | CCPR2CON | — | — |
| x1Bh | PR2 | ADRESL | — | SPBRG | — | — | — | — |
| x1Ch | T2CON | ADRESH | — | SPBRGH | — | — | — | — |
| x1Dh | — | ADCON0 | APFCON | RCSTA | — | — | — | IOCEP |
| x1Eh | — | ADCON1 | CM3CON0 | TXSTA | — | — | — | IOCEN |
| x1Fh | — | ADCON2 | CM3CON1 | BAUDCON | — | — | — | IOCEF |
| GPR | General Purpose Register 80 Bytes (020h–06Fh) | General Purpose Register 80 Bytes (0A0h–0EFh) | General Purpose Register 80 Bytes (120h/13Fh/140h–16Fh) | General Purpose Register 80 Bytes[1] (1A0h–1EFh) | General Purpose Register 80 Bytes[1] (220h–26Fh) | General Purpose Register 80 Bytes[1] (2A0h–2EFh) | General Purpose Register 32 Bytes[1] (320h–33Fh); Unimplemented Read as '0' (340h–36Fh) | Unimplemented Read as '0' (3A0h–3EFh) |
| x70h–x7Fh | Common RAM 70h–7Fh (070h–07Fh) | Accesses 70h–7Fh (0F0h–0FFh) | Accesses 70h–7Fh (170h–17Fh) | Accesses 70h–7Fh (1F0h–1FFh) | Accesses 70h–7Fh (270h–27Fh) | Accesses 70h–7Fh (2F0h–2FFh) | Accesses 70h–7Fh (370h–37Fh) | Accesses 70h–7Fh (3F0h–3FFh) |

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

**PIC16(L)F1782/3**

**PIC16(L)F1782/3**

**TABLE 3-4: PIC16(L)F1782/3 MEMORY MAP (BANKS 8-31)**

| | BANK 8 | | BANK 9 | | BANK 10 | | BANK 11 | | BANK 12 | | BANK 13 | | BANK 14 | | BANK 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 400h | Core Registers (Table 3-2) | 480h | Core Registers (Table 3-2) | 500h | Core Registers (Table 3-2) | 580h | Core Registers (Table 3-2) | 600h | Core Registers (Table 3-2) | 680h | Core Registers (Table 3-2) | 700h | Core Registers (Table 3-2) | 780h | Core Registers (Table 3-2) |
| 40Bh | | 48Bh | | 50Bh | | 58Bh | | 60Bh | | 68Bh | | 70Bh | | 78Bh | |
| 40Ch | | 48Ch | | 50Ch | Unimplemented Read as '0' | 58Ch | | 60Ch | | 68Ch | | 70Ch | | 78Ch | |
| | | | | 510h | | | | | | | | | | | |
| | | | | 511h | OPA1CON | | | | | | | | | | |
| | | | | 512h | — | | | | | | | | | | |
| | Unimplemented Read as '0' | | Unimplemented Read as '0' | 513h | OPA2CON | | Unimplemented Read as '0' | | Unimplemented Read as '0' | | Unimplemented Read as '0' | | Unimplemented Read as '0' | | Unimplemented Read as '0' |
| | | | | 514h | Unimplemented Read as '0' | | | | | | | | | | |
| | | | | 519h | | | | | | | | | | | |
| | | | | 51Ah | CLKRCON | | | | | | | | | | |
| | | | | 51Bh | Unimplemented Read as '0' | | | | | | | | | | |
| 46Fh | | 4EFh | | 56Fh | | 5EFh | | 66Fh | | 6EFh | | 76Fh | | 7EFh | |
| 470h | Common RAM (Accesses 70h – 7Fh) | 4F0h | Common RAM (Accesses 70h – 7Fh) | 570h | Common RAM (Accesses 70h – 7Fh) | 5F0h | Common RAM (Accesses 70h – 7Fh) | 670h | Common RAM (Accesses 70h – 7Fh) | 6F0h | Common RAM (Accesses 70h – 7Fh) | 770h | Common RAM (Accesses 70h – 7Fh) | 7F0h | Common RAM (Accesses 70h – 7Fh) |
| 47Fh | | 4FFh | | 57Fh | | 5FFh | | 67Fh | | 6FFh | | 77Fh | | 7FFh | |

| | BANK 16 | | BANK 17 | | BANK 18 | | BANK 19 | | BANK 20 | | BANK 21 | | BANK 22 | | BANK 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 800h | Core Registers (Table 3-2) | 880h | Core Registers (Table 3-2) | 900h | Core Registers (Table 3-2) | 980h | Core Registers (Table 3-2) | A00h | Core Registers (Table 3-2) | A80h | Core Registers (Table 3-2) | B00h | Core Registers (Table 3-2) | B80h | Core Registers (Table 3-2) |
| 80Bh | | 88Bh | | 90Bh | | 98Bh | | A0Bh | | A8Bh | | B0Bh | | B8Bh | |
| 80Ch | See Table 3-5 | 88Ch | Unimplemented Read as '0' | 90Ch | Unimplemented Read as '0' | 98Ch | Unimplemented Read as '0' | A0Ch | Unimplemented Read as '0' | A8Ch | Unimplemented Read as '0' | B0Ch | Unimplemented Read as '0' | B8Ch | Unimplemented Read as '0' |
| 86Fh | | 8EFh | | 96Fh | | 9EFh | | A6Fh | | AEFh | | B6Fh | | BEFh | |
| 870h | Common RAM (Accesses 70h – 7Fh) | 8F0h | Common RAM (Accesses 70h – 7Fh) | 970h | Common RAM (Accesses 70h – 7Fh) | 9F0h | Common RAM (Accesses 70h – 7Fh) | A70h | Common RAM (Accesses 70h – 7Fh) | AF0h | Common RAM (Accesses 70h – 7Fh) | B70h | Common RAM (Accesses 70h – 7Fh) | BF0h | Common RAM (Accesses 70h – 7Fh) |
| 87Fh | | 8FFh | | 97Fh | | 9FFh | | A7Fh | | AFFh | | B7Fh | | BFFh | |

| | BANK 24 | | BANK 25 | | BANK 26 | | BANK 27 | | BANK 28 | | BANK 29 | | BANK 30 | | BANK 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C00h | Core Registers (Table 3-2) | C80h | Core Registers (Table 3-2) | D00h | Core Registers (Table 3-2) | D80h | Core Registers (Table 3-2) | E00h | Core Registers (Table 3-2) | E80h | Core Registers (Table 3-2) | F00h | Core Registers (Table 3-2) | F80h | Core Registers (Table 3-2) |
| C0Bh | | C8Bh | | D0Bh | | D8Bh | | E0Bh | | E8Bh | | F0Bh | | F8Bh | |
| C0Ch | Unimplemented Read as '0' | C8Ch | Unimplemented Read as '0' | D0Ch | Unimplemented Read as '0' | D8Ch | Unimplemented Read as '0' | E0Ch | Unimplemented Read as '0' | E8Ch | Unimplemented Read as '0' | F0Ch | Unimplemented Read as '0' | F8Ch | See Table 3-6 |
| C6Fh | | CEFh | | D6Fh | | DEFh | | E6Fh | | EEFh | | F6Fh | | FEFh | |
| C70h | Common RAM (Accesses 70h – 7Fh) | CF0h | Common RAM (Accesses 70h – 7Fh) | D70h | Common RAM (Accesses 70h – 7Fh) | DF0h | Common RAM (Accesses 70h – 7Fh) | E70h | Common RAM (Accesses 70h – 7Fh) | EF0h | Common RAM (Accesses 70h – 7Fh) | F70h | Common RAM (Accesses 70h – 7Fh) | FF0h | Common RAM (Accesses 70h – 7Fh) |
| C7Fh | | CFFh | | D7Fh | | D7Fh | | E7Fh | | EFFh | | F7Fh | | FFFh | |

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

**TABLE 3-5:** **PIC16(L)F1782/3 MEMORY MAP (BANK 16 DETAILS)**

| | BANK 16 | | | BANK 16 |
|---|---|---|---|---|
| 811h | PSMC1CON | | 831h | PSMC2CON |
| 812h | PSMC1MDL | | 832h | PSMC2MDL |
| 813h | PSMC1SYNC | | 833h | PSMC2SYNC |
| 814h | PSMC1CLK | | 834h | PSMC2CLK |
| 815h | PSMC1OEN | | 835h | PSMC2OEN |
| 816h | PSMC1POL | | 836h | PSMC2POL |
| 817h | PSMC1BLNK | | 837h | PSMC2BLNK |
| 818h | PSMC1REBS | | 838h | PSMC2REBS |
| 819h | PSMC1FEBS | | 839h | PSMC2FEBS |
| 81Ah | PSMC1PHS | | 83Ah | PSMC2PHS |
| 81Bh | PSMC1DCS | | 83Bh | PSMC2DCS |
| 81Ch | PSMC1PRS | | 83Ch | PSMC2PRS |
| 81Dh | PSMC1ASDC | | 83Dh | PSMC2ASDC |
| 81Eh | PSMC1ASDD | | 83Eh | PSMC2ASDD |
| 81Fh | PSMC1ASDS | | 83Fh | PSMC2ASDS |
| 820h | PSMC1INT | | 840h | PSMC2INT |
| 821h | PSMC1PHL | | 841h | PSMC2PHL |
| 822h | PSMC1PHH | | 842h | PSMC2PHH |
| 823h | PSMC1DCL | | 843h | PSMC2DCL |
| 824h | PSMC1DCH | | 844h | PSMC2DCH |
| 825h | PSMC1PRL | | 845h | PSMC2PRL |
| 826h | PSMC1PRH | | 846h | PSMC2PRH |
| 827h | PSMC1TMRL | | 847h | PSMC2TMRL |
| 828h | PSMC1TMRH | | 848h | PSMC2TMRH |
| 829h | PSMC1DBR | | 849h | PSMC2DBR |
| 82Ah | PSMC1DBF | | 84Ah | PSMC2DBF |
| 82Bh | PSMC1BLKR | | 84Bh | PSMC2BLKR |
| 82Ch | PSMC1BLKF | | 84Ch | PSMC2BLKF |
| 82Dh | PSMC1FFA | | 84Dh | PSMC1FFA |
| 82Eh | PSMC1STR0 | | 84Eh | PSMC2STR0 |
| 82Fh | PSMC1STR1 | | 84Fh | PSMC2STR1 |
| 830h | — | | 840h | Unimplemented Read as '0' |
| | | | 86Fh | |

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

**TABLE 3-6:** **PIC16(L)F1782/3 MEMORY MAP (BANK 31 DETAILS)**

| | BANK 31 |
|---|---|
| F8Ch — FE3h | Unimplemented Read as '0' |
| FE4h | STATUS_SHAD |
| FE5h | WREG_SHAD |
| FE6h | BSR_SHAD |
| FE7h | PCLATH_SHAD |
| FE8h | FSR0L_SHAD |
| FE9h | FSR0H_SHAD |
| FEAh | FSR1L_SHAD |
| FEBh | FSR1H_SHAD |
| FECh | — |
| FEDh | STKPTR |
| FEEh | TOSL |
| FEFh | TOSH |

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

# PIC16(L)F1782/3

### 3.3.5 CORE FUNCTION REGISTERS SUMMARY

The Core Function registers listed in Table 3-7 can be addressed from any Bank.

### TABLE 3-7: CORE FUNCTION REGISTERS SUMMARY

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------|---------------------------|
| **Bank 0-31** | | | | | | | | | | | |
| x00h or x80h | INDF0 | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| x01h or x81h | INDF1 | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| x02h or x82h | PCL | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| x03h or x83h | STATUS | — | — | — | TO | PD | Z | DC | C | ---1 1000 | ---q quuu |
| x04h or x84h | FSR0L | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| x05h or x85h | FSR0H | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| x06h or x86h | FSR1L | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| x07h or x87h | FSR1H | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| x08h or x88h | BSR | — | — | — | BSR4 | BSR3 | BSR2 | BSR1 | BSR0 | ---0 0000 | ---0 0000 |
| x09h or x89h | WREG | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| x0Ah or x8Ah | PCLATH | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| x0Bh or x8Bh | INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

## TABLE 3-8: SPECIAL FUNCTION REGISTER SUMMARY

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 0** | | | | | | | | | | | |
| 00Ch | PORTA | PORTA Data Latch when written: PORTA pins when read | | | | | | | | xxxx xxxx | uuuu uuuu |
| 00Dh | PORTB | PORTB Data Latch when written: PORTB pins when read | | | | | | | | xxxx xxxx | uuuu uuuu |
| 00Eh | PORTC | PORTC Data Latch when written: PORTC pins when read | | | | | | | | xxxx xxxx | uuuu uuuu |
| 00Fh | — | Unimplemented | | | | | | | | — | — |
| 010h | PORTE | — | — | — | — | RE3 | — | — | — | ---- x--- | ---- u--- |
| 011h | PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| 012h | PIR2 | OSFIF | C2IF | C1IF | EEIF | BCL1IF | — | C3IF | CCP2IF | 0000 0-00 | 0000 0-00 |
| 013h | — | Unimplemented | | | | | | | | — | — |
| 014h | PIR4 | — | — | PSMC2TIF | PSMC1TIF | — | — | PSMC2SIF | PSMC1SIF | --00 --00 | --00 --00 |
| 015h | TMR0 | Timer0 Module Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 016h | TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 017h | TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 018h | T1CON | TMR1CS1 | TMR1CS0 | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | — | TMR1ON | 0000 00-0 | uuuu uu-u |
| 019h | T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/$\overline{\text{DONE}}$ | T1GVAL | T1GSS1 | T1GSS0 | 0000 0x00 | uuuu uxuu |
| 016h | TMR2 | Holding Register for the Least Significant Byte of the 16-bit TMR2 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 017h | PR2 | Holding Register for the Most Significant Byte of the 16-bit TMR2 Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 018h | T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | -000 0000 | -000 0000 |
| 01Dh to 01Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 1** | | | | | | | | | | | |
| 08Ch | TRISA | PORTA Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| 08Dh | TRISB | PORTB Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| 08Eh | TRISC | PORTC Data Direction Register | | | | | | | | 1111 1111 | 1111 1111 |
| 08Fh | — | Unimplemented | | | | | | | | — | — |
| 090h | TRISE | — | — | — | — | —(2) | — | — | — | ---- 1--- | ---- 1--- |
| 091h | PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| 092h | PIE2 | OSFIE | C2IE | C1IE | EEIE | BCL1IE | — | C3IE | CCP2IE | 0000 0-00 | 0000 0-00 |
| 093h | — | Unimplemented | | | | | | | | — | — |
| 094h | PIE4 | — | — | PSMC2TIE | PSMC1TIE | — | — | PSMC2SIE | PSMC2SIE | --00 --00 | --00 --00 |
| 095h | OPTION_REG | $\overline{\text{WPUEN}}$ | INTEDG | TMR0CS | TMR0SE | PSA | PS2 | PS1 | PS0 | 1111 1111 | 1111 1111 |
| 096h | PCON | STKOVF | STKUNF | — | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 00-1 11qq | qq-q qquu |
| 097h | WDTCON | — | — | WDTPS4 | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | SWDTEN | --01 0110 | --01 0110 |
| 098h | OSCTUNE | — | — | TUN5 | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 | --00 0000 | --00 0000 |
| 099h | OSCCON | SPLLEN | IRCF3 | IRCF2 | IRCF1 | IRCF0 | — | SCS1 | SCS0 | 0011 1-00 | 0011 1-00 |
| 09Ah | OSCSTAT | T1OSCR | PLLR | OSTS | HFIOFR | HFIOFL | MFIOFR | LFIOFR | HFIOFS | 00q0 --00 | qqqq --0q |
| 09Bh | ADRESL | A/D Result Register Low | | | | | | | | xxxx xxxx | uuuu uuuu |
| 09Ch | ADRESH | A/D Result Register High | | | | | | | | xxxx xxxx | uuuu uuuu |
| 09Dh | ADCON0 | — | CHS4 | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON | -000 0000 | -000 0000 |
| 09Eh | ADCON1 | ADFM | ADCS2 | ADCS1 | ADCS0 | — | ADNREF | ADPREF1 | ADPREF0 | 0000 -000 | 0000 -000 |
| 09Fh | ADCON2 | TRIGSEL3 | TRIGSEL2 | TRIGSEL1 | TRIGSEL0 | CHSN3 | CHSN2 | CHSN1 | CHSN0 | 000- -000 | 000- -000 |

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

**Note** **1:** These registers can be addressed from any bank.
**2:** Unimplemented, read as '1'.

## TABLE 3-8: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|------|------|-------|-------|-------|-------|-------|-------|-------|-------|------|------|
| **Bank 2** | | | | | | | | | | | |
| 10Ch | LATA | PORTA Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| 10Dh | LATB | PORTB Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| 10Eh | LATC | PORTC Data Latch | | | | | | | | xxxx xxxx | uuuu uuuu |
| 10Fh | — | Unimplemented | | | | | | | | — | — |
| 110h | — | Unimplemented | | | | | | | | — | — |
| 111h | CM1CON0 | C1ON | C1OUT | C1OE | C1POL | C1ZLF | C1SP | C1HYS | C1SYNC | 0000 0100 | 0000 0100 |
| 112h | CM1CON1 | C1INTP | C1INTN | C1PCH<2:0> | | | C1NCH<2:0> | | | 0000 0000 | 0000 0000 |
| 113h | CM2CON0 | C2ON | C2OUT | C2OE | C2POL | C2ZLF | C2SP | C2HYS | C2SYNC | 0000 0100 | 0000 0100 |
| 114h | CM2CON1 | C2INTP | C2INTN | C2PCH<2:0> | | | C2NCH<2:0> | | | 0000 0000 | 0000 0000 |
| 115h | CMOUT | — | — | — | — | — | MC3OUT | MC2OUT | MC1OUT | ---- -000 | ---- -000 |
| 116h | BORCON | SBOREN | BORFS | — | — | — | — | — | BORRDY | 1x-- ---q | uu-- ---u |
| 117h | FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR1 | CDAFVR0 | ADFVR1 | ADFVR0 | 0q00 0000 | 0q00 0000 |
| 118h | DACCON0 | DACEN | --- | DACOE1 | DACOE2 | DACPSS<1:0> | | --- | DACNSS | 0-00 00-0 | 0-00 00-0 |
| 119h | DACCON1 | DACR<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 11Ah to 11Ch | — | Unimplemented | | | | | | | | — | — |
| 11Dh | APFCON | C2OUTSEL | CC1PSEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 0000 0000 | 0000 0000 |
| 11Eh | CM3CON0 | C3ON | C3OUT | C3OE | C3POL | C3ZLF | C3SP | C3HYS | C3SYNC | 0000 0100 | 0000 0100 |
| 11Fh | CM3CON1 | C3INTP | C3INTN | C3PCH<2:0> | | | C3NCH<2:0> | | | 0000 0000 | 0000 0000 |
| **Bank 3** | | | | | | | | | | | |
| 18Ch | ANSELA | ANSA7 | — | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 1-11 1111 | 1-11 1111 |
| 18Dh | ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | --11 1111 | --11 1111 |
| 18Eh to 190h | — | Unimplemented | | | | | | | | — | — |
| 191h | EEADRL | EEPROM / Program Memory Address Register Low Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 192h | EEADRH | —(2) | EEPROM / Program Memory Address Register High Byte | | | | | | | 1000 0000 | 1000 0000 |
| 193h | EEDATL | EEPROM / Program Memory Read Data Register Low Byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| 194h | EEDATH | — | — | EEPROM / Program Memory Read Data Register High Byte | | | | | | --xx xxxx | --uu uuuu |
| 195h | EECON1 | EEPGD | CFGS | LWLO | FREE | WRERR | WREN | WR | RD | 0000 x000 | 0000 q000 |
| 196h | EECON2 | EEPROM / Program Memory Control Register 2 | | | | | | | | 0000 0000 | 0000 0000 |
| 197h | VREGCON | — | — | — | — | — | — | VREGPM | Reserved | ---- --01 | ---- --01 |
| 198h | — | Unimplemented | | | | | | | | — | — |
| 199h | RCREG | USART Receive Data Register | | | | | | | | 0000 0000 | 0000 0000 |
| 19Ah | TXREG | USART Transmit Data Register | | | | | | | | 0000 0000 | 0000 0000 |
| 19Bh | SPBRG | BRG<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 19Ch | SPBRGH | BRG<15:8> | | | | | | | | 0000 0000 | 0000 0000 |
| 19Dh | RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 0000 | 0000 0000 |
| 19Eh | TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 0000 0010 |
| 19Fh | BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 01-0 0-00 | 01-0 0-00 |

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

**Note** 1: These registers can be addressed from any bank.
2: Unimplemented, read as '1'.

**Preliminary**
© 2011-2012 Microchip Technology Inc.

## TABLE 3-8: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 4** | | | | | | | | | | | |
| 20Ch | WPUA | WPUA7 | WPUA6 | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 | 1111 1111 | 1111 1111 |
| 20Dh | WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 | 1111 1111 | 1111 1111 |
| 20Eh | WPUC | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 | 1111 1111 | 1111 1111 |
| 20Fh | — | Unimplemented | | | | | | | | — | — |
| 210h | WPUE | — | — | — | — | WPUE3 | — | — | — | ---- 1--- | ---- 1--- |
| 211h | SSPBUF | Synchronous Serial Port Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 212h | SSPADD | ADD<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 213h | SSPMSK | MSK<7:0> | | | | | | | | 1111 1111 | 1111 1111 |
| 214h | SSPSTAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 0000 0000 | 0000 0000 |
| 215h | SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 0000 0000 | 0000 0000 |
| 216h | SSPCON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | 0000 0000 |
| 217h | SSPCON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 0000 0000 | 0000 0000 |
| 218h — 21Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 5** | | | | | | | | | | | |
| 28Ch | ODCONA | Open Drain Control for PORTA | | | | | | | | 0000 0000 | 0000 0000 |
| 28Dh | ODCONB | Open Drain Control for PORTB | | | | | | | | 0000 0000 | 0000 0000 |
| 28Eh | ODCONC | Open Drain Control for PORTC | | | | | | | | 0000 0000 | 0000 0000 |
| 28Fh | — | Unimplemented | | | | | | | | — | — |
| 290h | — | Unimplemented | | | | | | | | — | — |
| 291h | CCPR1L | Capture/Compare/PWM Register 1 (LSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 292h | CCPR1H | Capture/Compare/PWM Register 1 (MSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 293h | CCP1CON | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | | 0000 0000 | 0000 0000 |
| 294h — 297h | — | Unimplemented | | | | | | | | — | — |
| 298h | CCPR2L | Capture/Compare/PWM Register 2 (LSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 299h | CCPR2H | Capture/Compare/PWM Register 2 (MSB) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 29Ah | CCP2CON | P2M<1:0> | | DC2B<1:0> | | CCP2M<3:0> | | | | 0000 0000 | 0000 0000 |
| 29Bh — 29Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 6** | | | | | | | | | | | |
| 30Ch | SLRCONA | Slew Rate Control for PORTA | | | | | | | | 0000 0000 | 0000 0000 |
| 30Dh | SLRCONB | Slew Rate Control for PORTB | | | | | | | | 0000 0000 | 0000 0000 |
| 30Eh | SLRCONC | Slew Rate Control for PORTC | | | | | | | | 0000 0000 | 0000 0000 |
| 30Fh — 31Fh | — | Unimplemented | | | | | | | | — | — |

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.
**Note 1:** These registers can be addressed from any bank.
**2:** Unimplemented, read as '1'.

**TABLE 3-8:** **SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)**

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 7** | | | | | | | | | | | |
| 38Ch | INLVLA | Input Type Control for PORTA | | | | | | | | 0000 0000 | 0000 0000 |
| 38Dh | INLVLB | Input Type Control for PORTB | | | | | | | | 0000 0000 | 0000 0000 |
| 38Eh | INLVLC | Input Type Control for PORTC | | | | | | | | 1111 1111 | 1111 1111 |
| 38Fh | — | Unimplemented | | | | | | | | — | — |
| 390h | INLVLE | — | — | — | — | INLVLE3 | — | — | — | ---- 1--- | ---- 1--- |
| 391h | IOCAP | IOCAP<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 392h | IOCAN | IOCAN<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 393h | IOCAF | IOCAF<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 394h | IOCBP | IOCBP<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 395h | IOCBN | IOCBN<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 396h | IOCBF | IOCBF<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 397h | IOCCP | IOCCP<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 398h | IOCCN | IOCCN<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 399h | IOCCF | IOCCF<7:0> | | | | | | | | 0000 0000 | 0000 0000 |
| 39Ah — 39Ch | — | Unimplemented | | | | | | | | — | — |
| 39Dh | IOCEP | — | — | — | — | IOCEP3 | — | — | — | ---- 0--- | ---- 0--- |
| 39Eh | IOCEN | — | — | — | — | IOCEN3 | — | — | — | ---- 0--- | ---- 0--- |
| 39Fh | IOCEF | — | — | — | — | IOCEF3 | — | — | — | ---- 0--- | ---- 0--- |
| **Bank 8-9** | | | | | | | | | | | |
| 40Ch or 41Fh and 48Ch or 49Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 10** | | | | | | | | | | | |
| 50Ch — 510h | — | Unimplemented | | | | | | | | — | — |
| 511h | OPA1CON | OPA1EN | OPA1SP | — | — | — | — | OPA1PCH<1:0> | | 00-- --00 | 00-- --00 |
| 512h | — | Unimplemented | | | | | | | | — | — |
| 513h | OPA2CON | OPA2EN | OPA2SP | — | — | — | — | OPA2PCH<1:0> | | 00-- --00 | 00-- --00 |
| 514h — 519h | — | Unimplemented | | | | | | | | — | — |
| 51Ah | CLKRCON | CLKREN | CLKROE | CLKRSLR | CLKRDC<1:0> | | CLKRDIV<2:0> | | | 0011 0000 | 0011 0000 |
| 51Bh — 51Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 11-15** | | | | | | | | | | | |
| x0Ch or x8Ch to x6Fh or xEFh | — | Unimplemented | | | | | | | | — | — |

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

**Note 1:** These registers can be addressed from any bank.
**2:** Unimplemented, read as '1'.

**Preliminary**

© 2011-2012 Microchip Technology Inc.

**TABLE 3-8:** **SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)**

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 16** | | | | | | | | | | | |
| 80Ch — 810h | — | Unimplemented | | | | | | | | — | — |
| 811h | PSMC1CON | PSMC1EN | PSMC1LD | PSMC1DBFE | PSMC1DBRE | P1MODE<3:0> | | | | 0000 0000 | 0000 0000 |
| 812h | PSMC1MDL | P1MDLEN | P1MDLPOL | P1MDLBIT | — | P1MSRC<3:0> | | | | 000- 0000 | 000- 0000 |
| 813h | PSMC1SYNC | — | — | — | — | — | — | P1SYNC<1:0> | | ---- --00 | ---- --00 |
| 814h | PSMC1CLK | — | — | P1CPRE<1:0> | | — | — | P1CSRC<1:0> | | --00 --00 | --00 --00 |
| 815h | PSMC1OEN | — | — | P1OEF | P1OEE | P1OED | P1OEC | P1OEB | P1OEA | --00 0000 | --00 0000 |
| 816h | PSMC1POL | — | P1INPOL | P1POLF | P1POLE | P1POLD | P1POLC | P1POLB | P1POLA | -000 0000 | -000 0000 |
| 817h | PSMC1BLNK | — | — | P1FEBM<1:0> | | — | — | P1REBM<1:0> | | --00 --00 | --00 --00 |
| 818h | PSMCIREBS | P1REBIN | — | — | — | P1REBSC3 | P1REBSC2 | P1REBSC1 | — | 0--- 000- | 0000 000- |
| 819h | PSMCIFEBS | P1FEBIN | — | — | — | P1FEBSC3 | P1FEBSC2 | P1FEBSC1 | — | 0--- 000- | 0000 000- |
| 81Ah | PSMC1PHS | P1PHSIN | — | — | — | P1PHSC3 | P1PHSC2 | P1PHSC1 | P1PHST | 0--- 0000 | 0--- 0000 |
| 81Bh | PSMC1DCS | P1DCSIN | — | — | — | P1DCSC3 | P1DCSC2 | P1DCSC1 | P1DCST | 0--- 0000 | 0--- 0000 |
| 81Ch | PSMC1PRS | P1PRSIN | — | — | — | P1PRSC3 | P1PRSC2 | P1PRSC1 | P1PRST | 0--- 0000 | 0--- 0000 |
| 81Dh | PSMC1ASDC | P1ASE | P1ASDEN | P1ARSEN | — | — | — | — | P1ASDOV | 000- ---0 | 000- ---0 |
| 81Eh | PSMC1ASDL | — | — | P1ASDLF | P1ASDLE | P1ASDLD | P1ASDLC | P1ASDLB | P1ASDLA | --00 0000 | --00 0000 |
| 81Fh | PSMC1ASDS | P1ASDSIN | — | — | — | P1ASDSC3 | P1ASDSC2 | P1ASDSC1 | — | 0--- 000- | 0--- 000- |
| 820h | PSMC1INT | P1TOVIE | P1TPHIE | P1TDCIE | P1TPRIE | P1TOVIF | P1TPHIF | P1TDCIF | P1TPRIF | 0000 0000 | 0000 0000 |
| 821h | PSMC1PHL | Phase Low Count | | | | | | | | 0000 0000 | 0000 0000 |
| 822h | PSMC1PHH | Phase High Count | | | | | | | | 0000 0000 | 0000 0000 |
| 823h | PSMC1DCL | Duty Cycle Low Count | | | | | | | | 0000 0000 | 0000 0000 |
| 824h | PSMC1DCH | Duty Cycle High Count | | | | | | | | 0000 0000 | 0000 0000 |
| 825h | PSMC1PRL | Period Low Count | | | | | | | | 0000 0000 | 0000 0000 |
| 826h | PSMC1PRH | Period High Count | | | | | | | | 0000 0000 | 0000 0000 |
| 827h | PSMC1TMRL | Time base Low Counter | | | | | | | | 0000 0001 | 0000 0001 |
| 828h | PSMC1TMRH | Time base High Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 829h | PSMC1DBR | rising Edge Dead-band Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 82Ah | PSMC1DBF | Falling Edge Dead-band Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 82Bh | PSMC1BLKR | rising Edge Blanking Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 82Ch | PSMC1BLKF | Falling Edge Blanking Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 82Dh | PSMC1FFA | — | — | — | — | Fractional Frequency Adjust Register | | | | ---- 0000 | ---- 0000 |
| 82Eh | PSMC1STR0 | — | — | P1STRF | P1STRE | P1STRD | P1STRC | P1STRB | P1STRA | --00 0001 | --00 0001 |
| 82Fh | PSMC1STR1 | P1SYNC | — | — | — | — | — | P1LSMEN | P1HSMEN | 0--- --00 | 0--- --00 |
| 830h | — | Unimplemented | | | | | | | | — | — |

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.
**Note 1:** These registers can be addressed from any bank.
**2:** Unimplemented, read as '1'.

# PIC16(L)F1782/3

### TABLE 3-8: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------|---------------------------|
| **Bank 16 (Continued)** | | | | | | | | | | | |
| 831h | PSMC2CON | PSMC2EN | PSMC2LD | PSMC2DBFE | PSMC2DBRE | P2MODE<3:0> | | | | 0000 0000 | 0000 0000 |
| 832h | PSMC2MDL | P2MDLEN | P2MDLPOL | P2MDLBIT | — | P2MSRC<3:0> | | | | 000- 0000 | 000- 0000 |
| 833h | PSMC2SYNC | — | — | — | — | — | — | P2SYNC<1:0> | | ---- --00 | ---- --00 |
| 834h | PSMC2CLK | — | — | P2CPRE<1:0> | | — | — | P2CSRC<1:0> | | --00 --00 | --00 --00 |
| 835h | PSMC2OEN | — | — | — | — | — | — | P2OEB | P2OEA | ---- --00 | ---- --00 |
| 836h | PSMC2POL | — | P2INPOL | — | — | — | — | P2POLB | P2POLA | -0-- --00 | -0-- --00 |
| 837h | PSMC2BLNK | — | — | P2FEBM<1:0> | | — | — | P2REBM<1:0> | | --00 --00 | --00 --00 |
| 838h | PSMC2REBS | P2REBIN | — | — | — | P2REBSC3 | P2REBSC2 | P2REBSC1 | — | 0--- 000- | 0000 000- |
| 839h | PSMC2FEBS | P2FEBIN | — | — | — | P2FEBSC3 | P2FEBSC2 | P2FEBSC1 | — | 0--- 000- | 0000 000- |
| 83Ah | PSMC2PHS | P2PHSIN | — | — | — | P2PHSC3 | P2PHSC2 | P2PHSC1 | P2PHST | 0--- 0000 | 0--- 0000 |
| 83Bh | PSMC2DCS | P2DCSIN | — | — | — | P2DCSC3 | P2DCSC2 | P2DCSC1 | P2DCST | 0--- 0000 | 0--- 0000 |
| 83Ch | PSMC2PRS | P2PRSIN | — | — | — | P2PRSC3 | P2PRSC2 | P2PRSC1 | P2PRST | 0--- 0000 | 0--- 0000 |
| 83Dh | PSMC2ASDC | P2ASE | P2ASDEN | P2ARSEN | — | — | — | — | P2ASDOV | 000- ---0 | 000- ---0 |
| 83Eh | PSMC2ASDL | — | — | P2ASDLF | P2ASDLE | P2ASDLD | P2ASDLC | P2ASDLB | P2ASDLA | --00 0000 | --00 0000 |
| 83Fh | PSMC2ASDS | P2ASDSIN | — | — | — | P2ASDSC3 | P2ASDSC2 | P2ASDSC1 | — | 0--- 000- | 0--- 000- |
| 840h | PSMC2INT | P2TOVIE | P2TPHIE | P2TDCIE | P2TPRIE | P2TOVIF | P2TPHIF | P2TDCIF | P2TPRIF | 0000 0000 | 0000 0000 |
| 841h | PSMC2PHL | Phase Low Count | | | | | | | | 0000 0000 | 0000 0000 |
| 842h | PSMC2PHH | Phase High Count | | | | | | | | 0000 0000 | 0000 0000 |
| 843h | PSMC2DCL | Duty Cycle Low Count | | | | | | | | 0000 0000 | 0000 0000 |
| 844h | PSMC2DCH | Duty Cycle High Count | | | | | | | | 0000 0000 | 0000 0000 |
| 845h | PSMC2PRL | Period Low Count | | | | | | | | 0000 0000 | 0000 0000 |
| 846h | PSMC2PRH | Period High Count | | | | | | | | 0000 0000 | 0000 0000 |
| 847h | PSMC2TMRL | Time base Low Counter | | | | | | | | 0000 0001 | 0000 0001 |
| 848h | PSMC2TMRH | Time base High Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 849h | PSMC2DBR | rising Edge Dead-band Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 84Ah | PSMC2DBF | Falling Edge Dead-band Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 84Bh | PSMC2BLKR | rising Edge Blanking Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 84Ch | PSMC2BLKF | Falling Edge Blanking Counter | | | | | | | | 0000 0000 | 0000 0000 |
| 84Dh | PSMC2FFA | — | — | — | — | Fractional Frequency Adjust Register | | | | ---- 0000 | ---- 0000 |
| 84Eh | PSMC2STR0 | — | — | — | — | — | — | P2STRB | P2STRA | ---- --01 | ---- --01 |
| 84Fh | PSMC2STR1 | P2SYNC | — | — | — | — | — | P2LSMEN | P2HSMEN | 0--- --00 | 0--- --00 |
| 850h — 86Fh | — | Unimplemented | | | | | | | | — | — |
| **Bank 17-30** | | | | | | | | | | | |
| x0Ch or x8Ch to x1Fh or x9Fh | — | Unimplemented | | | | | | | | — | — |

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

**Note 1:** These registers can be addressed from any bank.

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

**Preliminary**

© 2011-2012 Microchip Technology Inc.

## TABLE 3-8: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

| Addr | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Value on all other Resets |
|------|------|-------|-------|-------|-------|-------|-------|-------|-------|-------------------|---------------------------|
| **Bank 31** | | | | | | | | | | | |
| F8Ch to FE3h | — | Unimplemented | | | | | | | | — | — |
| FE4h | STATUS_SHAD | — | — | — | — | — | Z | DC | C | ---- -xxx | ---- -uuu |
| FE5h | WREG_SHAD | Working Register Shadow | | | | | | | | xxxx xxxx | uuuu uuuu |
| FE6h | BSR_SHAD | — | — | — | Bank Select Register Shadow | | | | | ---x xxxx | ---u uuuu |
| FE7h | PCLATH_SHAD | — | Program Counter Latch High Register Shadow | | | | | | | -xxx xxxx | uuuu uuuu |
| FE8h | FSR0L_SHAD | Indirect Data Memory Address 0 Low Pointer Shadow | | | | | | | | xxxx xxxx | uuuu uuuu |
| FE9h | FSR0H_SHAD | Indirect Data Memory Address 0 High Pointer Shadow | | | | | | | | xxxx xxxx | uuuu uuuu |
| FEAh | FSR1L_SHAD | Indirect Data Memory Address 1 Low Pointer Shadow | | | | | | | | xxxx xxxx | uuuu uuuu |
| FEBh | FSR1H_SHAD | Indirect Data Memory Address 1 High Pointer Shadow | | | | | | | | xxxx xxxx | uuuu uuuu |
| FECh | — | Unimplemented | | | | | | | | — | — |
| FEDh | STKPTR | — | — | — | Current Stack Pointer | | | | | ---1 1111 | ---1 1111 |
| FEEh | TOSL | Top of Stack Low byte | | | | | | | | xxxx xxxx | uuuu uuuu |
| FEFh | TOSH | — | Top of Stack High byte | | | | | | | -xxx xxxx | -uuu uuuu |

**Legend:** x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

**Note 1:** These registers can be addressed from any bank.
**2:** Unimplemented, read as '1'.

## 3.4 PCL and PCLATH

The Program Counter (PC) is 15 bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC<14:8>) is not directly readable or writable and comes from PCLATH. On any Reset, the PC is cleared. Figure 3-4 shows the five situations for the loading of the PC.

**FIGURE 3-4: LOADING OF PC IN DIFFERENT SITUATIONS**
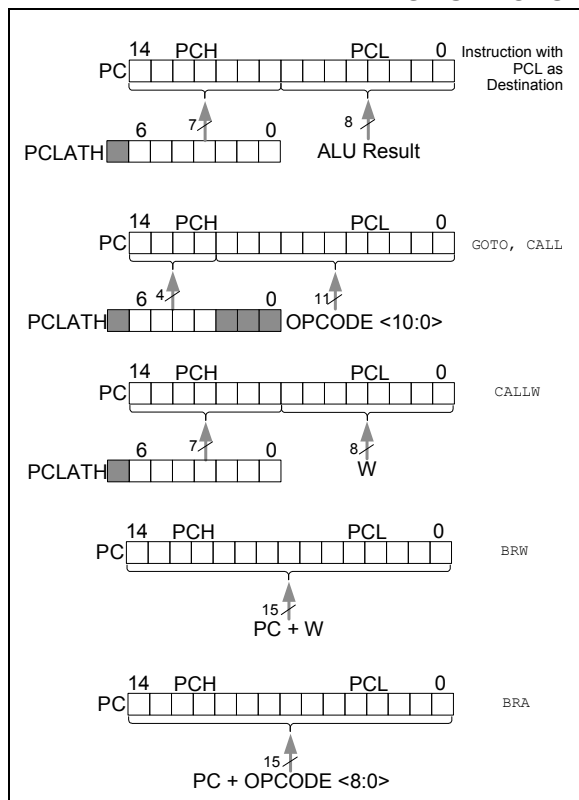


### 3.4.1 MODIFYING PCL

Executing any instruction with the PCL register as the destination simultaneously causes the Program Counter PC<14:8> bits (PCH) to be replaced by the contents of the PCLATH register. This allows the entire contents of the program counter to be changed by writing the desired upper 7 bits to the PCLATH register. When the lower 8 bits are written to the PCL register, all 15 bits of the program counter will change to the values contained in the PCLATH register and those being written to the PCL register.

### 3.4.2 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL). When performing a table read using a computed GOTO method, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block). Refer to Application Note AN556, *"Implementing a Table Read"* (DS00556).

### 3.4.3 COMPUTED FUNCTION CALLS

A computed function CALL allows programs to maintain tables of functions and provide another way to execute state machines or look-up tables. When performing a table read using a computed function CALL, care should be exercised if the table location crosses a PCL memory boundary (each 256-byte block).

If using the CALL instruction, the PCH<2:0> and PCL registers are loaded with the operand of the CALL instruction. PCH<6:3> is loaded with PCLATH<6:3>.

The CALLW instruction enables computed calls by combining PCLATH and W to form the destination address. A computed CALLW is accomplished by loading the W register with the desired address and executing CALLW. The PCL register is loaded with the value of W and PCH is loaded with PCLATH.

### 3.4.4 BRANCHING

The branching instructions add an offset to the PC. This allows relocatable code and code that crosses page boundaries. There are two forms of branching, BRW and BRA. The PC will have incremented to fetch the next instruction in both cases. When using either branching instruction, a PCL memory boundary may be crossed.

If using BRW, load the W register with the desired unsigned address and execute BRW. The entire PC will be loaded with the address PC + 1 + W.

If using BRA, the entire PC will be loaded with PC + 1 +, the signed value of the operand of the BRA instruction.

## 3.5    Stack

All devices have a 16-level x 15-bit wide hardware stack (refer to Figures 3-3 and 3-3). The stack space is not part of either program or data space. The PC is PUSHed onto the stack when `CALL` or `CALLW` instructions are executed or an interrupt causes a branch. The stack is POPed in the event of a `RETURN`, `RETLW` or a `RETFIE` instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer if the STVREN bit is programmed to '0' (Configuration Words). This means that after the stack has been PUSHed sixteen times, the seventeenth PUSH overwrites the value that was stored from the first PUSH. The eighteenth PUSH overwrites the second PUSH (and so on). The STKOVF and STKUNF flag bits will be set on an Overflow/Underflow, regardless of whether the Reset is enabled.

> **Note:** There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the `CALL`, `CALLW`, `RETURN`, `RETLW` and `RETFIE` instructions or the vectoring to an interrupt address.

### 3.5.1    ACCESSING THE STACK

The stack is available through the TOSH, TOSL and STKPTR registers. STKPTR is the current value of the Stack Pointer. TOSH:TOSL register pair points to the TOP of the stack. Both registers are read/writable. TOS is split into TOSH and TOSL due to the 15-bit size of the PC. To access the stack, adjust the value of STKPTR, which will position TOSH:TOSL, then read/write to TOSH:TOSL. STKPTR is 5 bits to allow detection of overflow and underflow.

> **Note:** Care should be taken when modifying the STKPTR while interrupts are enabled.

During normal program operation, `CALL`, `CALLW` and interrupts will increment STKPTR while `RETLW`, `RETURN`, and `RETFIE` will decrement STKPTR. At any time, STKPTR can be inspected to see how much stack is left. The STKPTR always points at the currently used place on the stack. Therefore, a `CALL` or `CALLW` will increment the STKPTR and then write the PC, and a return will unload the PC and then decrement the STKPTR.

Reference Figure 3-5 through Figure 3-8 for examples of accessing the stack.

**FIGURE 3-5:        ACCESSING THE STACK EXAMPLE 1**
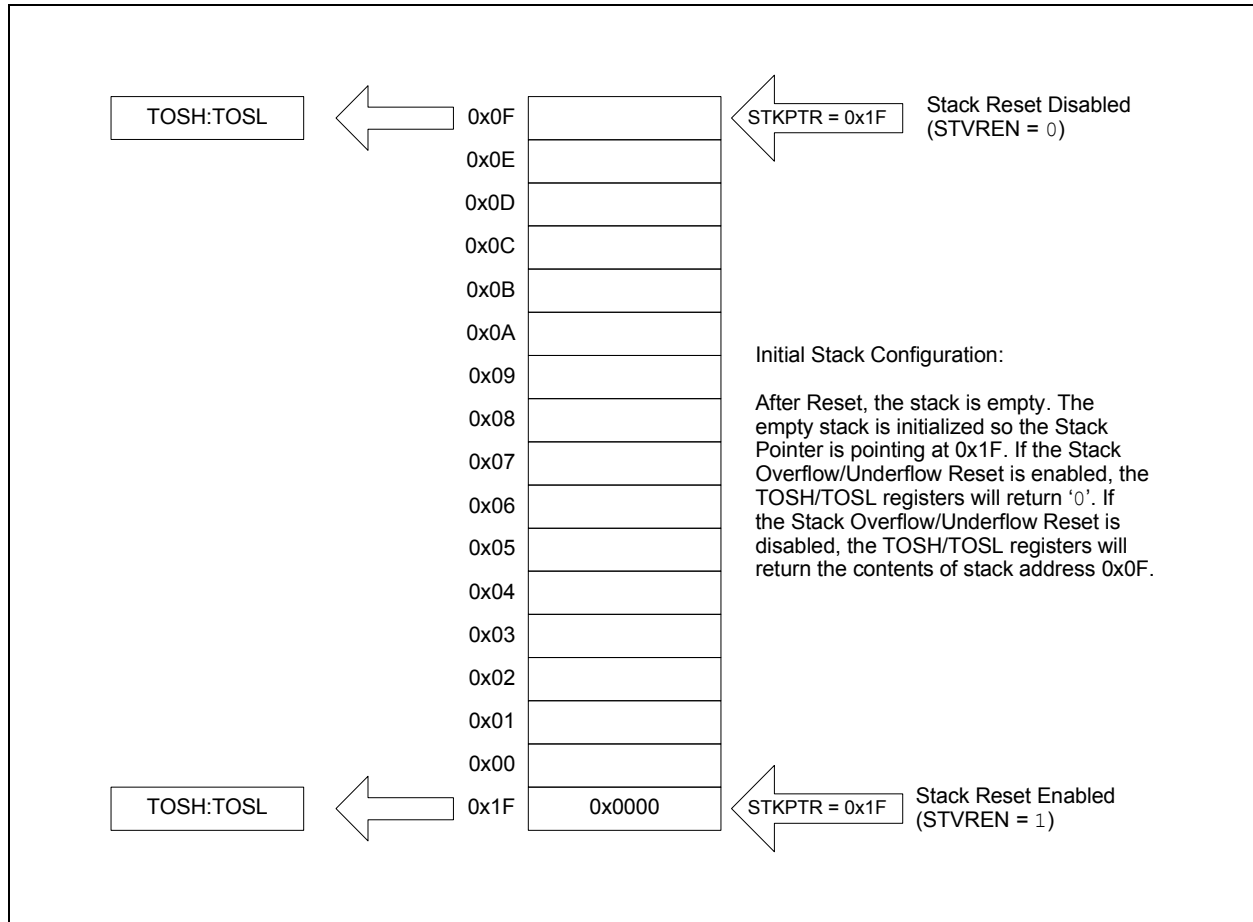
**FIGURE 3-6:** **ACCESSING THE STACK EXAMPLE 2**

| | |
|---|---|
| 0x0F | |
| 0x0E | |
| 0x0D | |
| 0x0C | |
| 0x0B | |
| 0x0A | |
| 0x09 | |
| 0x08 | |
| 0x07 | |
| 0x06 | |
| 0x05 | |
| 0x04 | |
| 0x03 | |
| 0x02 | |
| 0x01 | |

TOSH:TOSL ⟸ 0x00 | Return Address | ⟸ STKPTR = 0x00

This figure shows the stack configuration after the first CALL or a single interrupt. If a RETURN instruction is executed, the return address will be placed in the Program Counter and the Stack Pointer decremented to the empty state (0x1F).
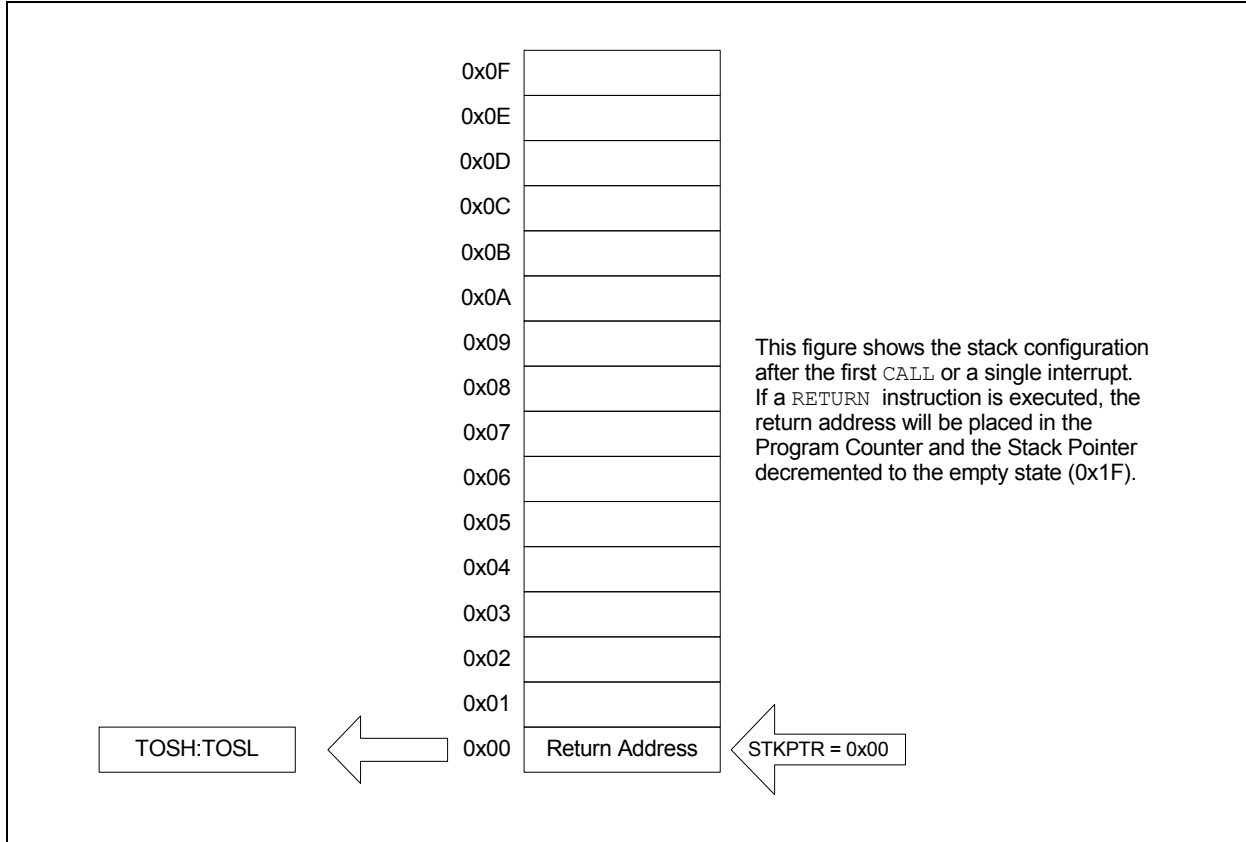
**FIGURE 3-7:** **ACCESSING THE STACK EXAMPLE 3**

| | |
|---|---|
| 0x0F | |
| 0x0E | |
| 0x0D | |
| 0x0C | |
| 0x0B | |
| 0x0A | |
| 0x09 | |
| 0x08 | |
| 0x07 | |

TOSH:TOSL ⟸ 0x06 | Return Address | ⟸ STKPTR = 0x06

| | |
|---|---|
| 0x05 | Return Address |
| 0x04 | Return Address |
| 0x03 | Return Address |
| 0x02 | Return Address |
| 0x01 | Return Address |
| 0x00 | Return Address |

After seven CALLs or six CALLs and an interrupt, the stack looks like the figure on the left. A series of RETURN instructions will repeatedly place the return addresses into the Program Counter and pop the stack.

**FIGURE 3-8: ACCESSING THE STACK EXAMPLE 4**

| | |
|---|---|
| 0x0F | Return Address |
| 0x0E | Return Address |
| 0x0D | Return Address |
| 0x0C | Return Address |
| 0x0B | Return Address |
| 0x0A | Return Address |
| 0x09 | Return Address |
| 0x08 | Return Address |
| 0x07 | Return Address |
| 0x06 | Return Address |
| 0x05 | Return Address |
| 0x04 | Return Address |
| 0x03 | Return Address |
| 0x02 | Return Address |
| 0x01 | Return Address |

TOSH:TOSL ⟸ 0x00 Return Address ⟸ STKPTR = 0x10

When the stack is full, the next CALL or an interrupt will set the Stack Pointer to 0x10. This is identical to address 0x00 so the stack will wrap and overwrite the return address at 0x00. If the Stack Overflow/Underflow Reset is enabled, a Reset will occur and location 0x00 will not be overwritten.

### 3.5.2 OVERFLOW/UNDERFLOW RESET

If the STVREN bit in Configuration Words is programmed to '1', the device will be reset if the stack is PUSHed beyond the sixteenth level or POPed beyond the first level, setting the appropriate bits (STKOVF or STKUNF, respectively) in the PCON register.

## 3.6 Indirect Addressing

The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the File Select Registers (FSR). If the FSRn address specifies one of the two INDFn registers, the read will return '0' and the write will not occur (though Status bits may be affected). The FSRn register value is created by the pair FSRnH and FSRnL.

The FSR registers form a 16-bit address that allows an addressing space with 65536 locations. These locations are divided into three memory regions:

• Traditional Data Memory
• Linear Data Memory
• Program Flash Memory

# PIC16(L)F1782/3

**FIGURE 3-9:** INDIRECT ADDRESSING

| | | |
|---|---|---|
| 0x0000 | 0x0000 | |
| | **Traditional Data Memory** | |
| 0x0FFF | 0x0FFF | |
| 0x1000 | | |
| 0x1FFF | Reserved | |
| 0x2000 | | |
| | **Linear Data Memory** | |
| 0x29AF | | |
| 0x29B0 | Reserved | |
| 0x7FFF | | |
| 0x8000 | 0x0000 | |
| | **Program Flash Memory** | |
| 0xFFFF | 0x7FFF | |

FSR Address Range

**Note:** Not all memory regions are completely implemented. Consult device memory tables for memory limits.

**Preliminary**
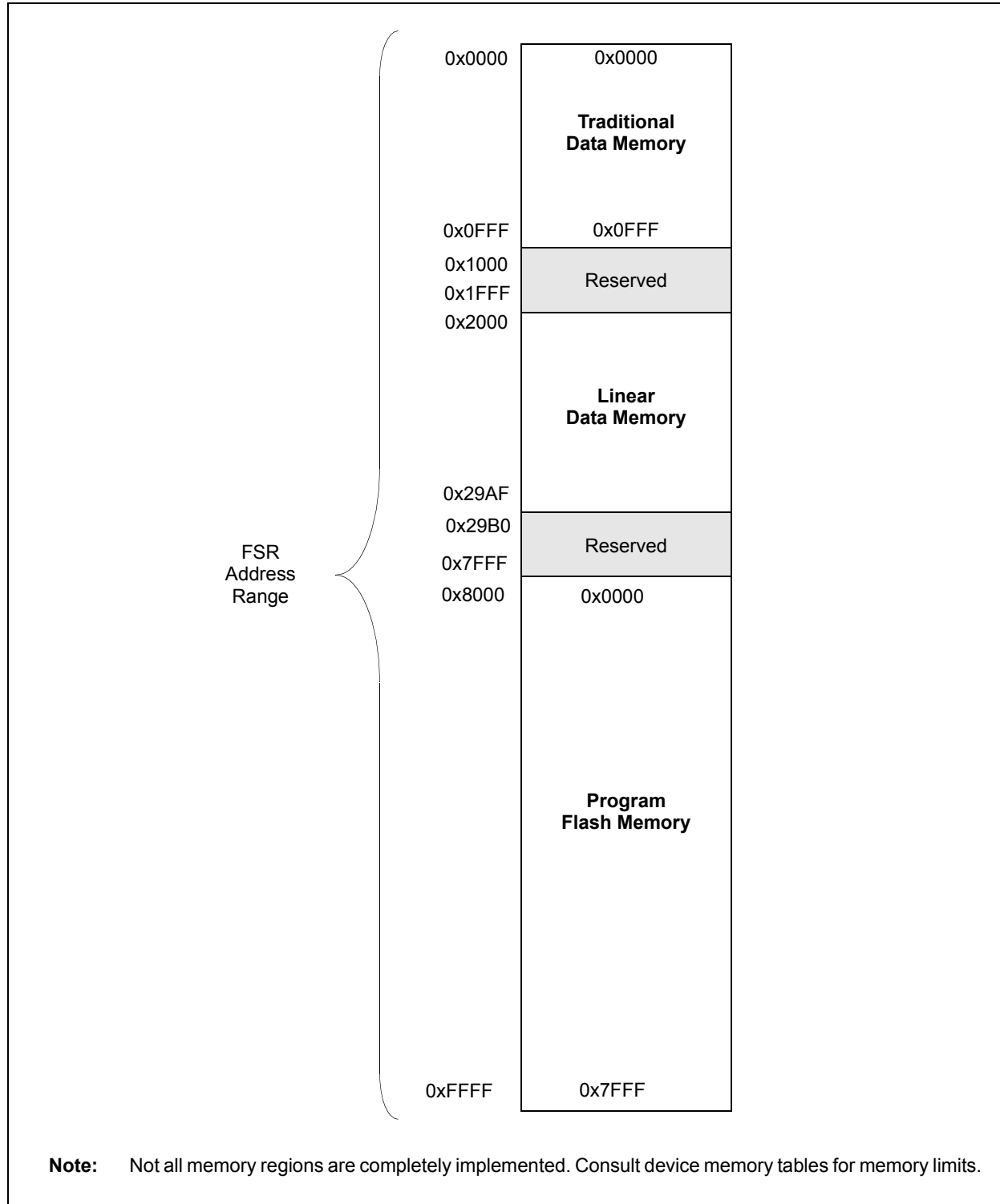
### 3.6.1 TRADITIONAL DATA MEMORY

The traditional data memory is a region from FSR address 0x000 to FSR address 0xFFF. The addresses correspond to the absolute addresses of all SFR, GPR and common registers.

**FIGURE 3-10:** **TRADITIONAL DATA MEMORY MAP**
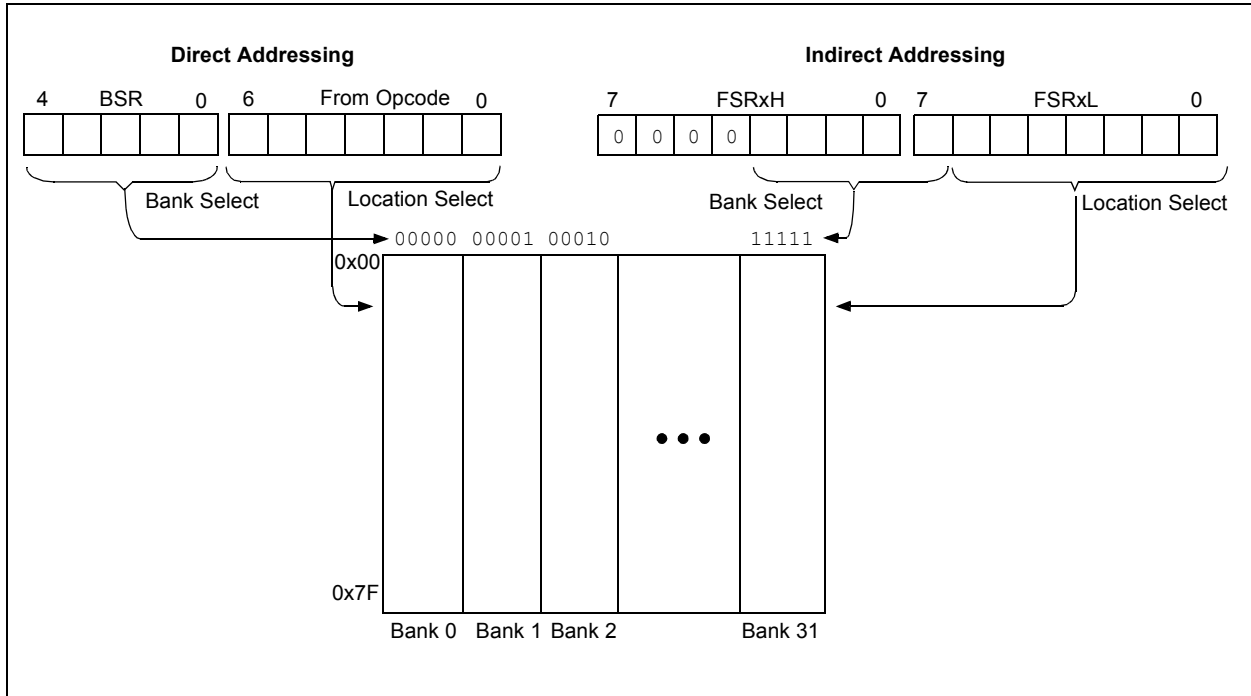
# PIC16(L)F1782/3

## 3.6.2 LINEAR DATA MEMORY

The linear data memory is the region from FSR address 0x2000 to FSR address 0x29AF. This region is a virtual region that points back to the 80-byte blocks of GPR memory in all the banks.

Unimplemented memory reads as 0x00. Use of the linear data memory region allows buffers to be larger than 80 bytes because incrementing the FSR beyond one bank will go directly to the GPR memory of the next bank.

The 16 bytes of common memory are not included in the linear data memory region.

**FIGURE 3-11:** **LINEAR DATA MEMORY MAP**
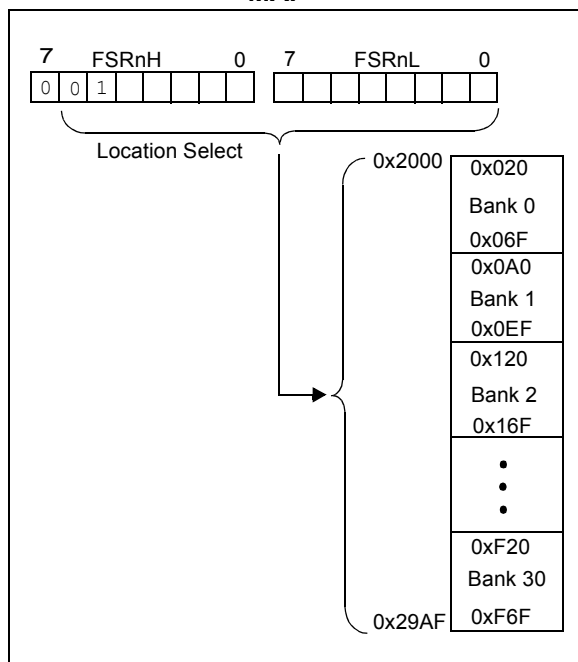


## 3.6.3 PROGRAM FLASH MEMORY

To make constant data access easier, the entire program Flash memory is mapped to the upper half of the FSR address space. When the MSB of FSRnH is set, the lower 15 bits are the address in program memory which will be accessed through INDF. Only the lower 8 bits of each memory location is accessible via INDF. Writing to the program Flash memory cannot be accomplished via the FSR/INDF interface. All instructions that access program Flash memory via the FSR/INDF interface will require one additional instruction cycle to complete.

**FIGURE 3-12:** **PROGRAM FLASH MEMORY MAP**



**Preliminary** © 2011-2012 Microchip Technology Inc.
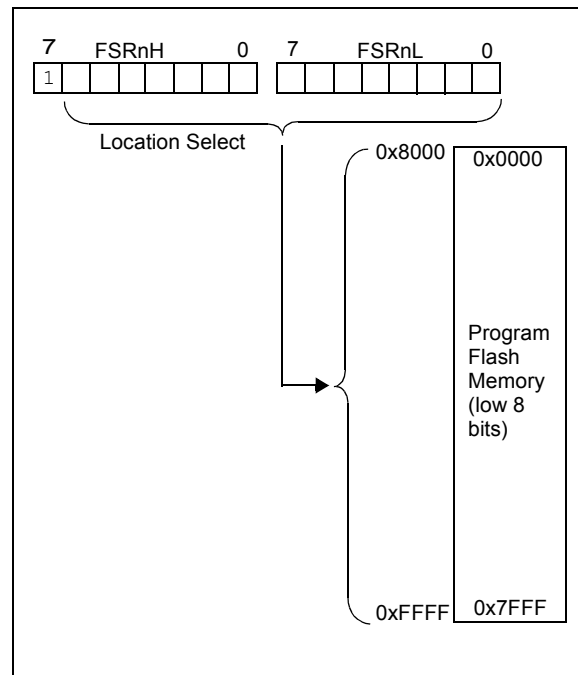
## 4.0    DEVICE CONFIGURATION

Device Configuration consists of Configuration Words, Code Protection and Device ID.

## 4.1    Configuration Words

There are several Configuration Word bits that allow different oscillator and memory protection options. These are implemented as Configuration Word 1 at 8007h and Configuration Word 2 at 8008h.

| Note: | The $\overline{\text{DEBUG}}$ bit in Configuration Words is managed automatically by device development tools including debuggers and programmers. For normal device operation, this bit should be maintained as a '1'. |
|---|---|

## 4.2 Register Definitions: Configuration Words

**REGISTER 4-1:** **CONFIG1: CONFIGURATION WORD 1**

| R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|
| FCMEN | IESO | CLKOUTEN | BOREN<1:0> | | CPD |
| bit 13 | | | | | bit 8 |

| R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| CP | MCLRE | PWRTE | WDTE<1:0> | | FOSC<2:0> | | |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---|---|---|
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '1' |
| '0' = Bit is cleared | '1' = Bit is set | -n = Value when blank or after Bulk Erase |

bit 13     **FCMEN:** Fail-Safe Clock Monitor Enable bit
  1 = Fail-Safe Clock Monitor and internal/external switchover are both enabled.
  0 = Fail-Safe Clock Monitor is disabled

bit 12     **IESO:** Internal External Switchover bit
  1 = Internal/External Switchover mode is enabled
  0 = Internal/External Switchover mode is disabled

bit 11     **CLKOUTEN:** Clock Out Enable bit
  If FOSC configuration bits are set to LP, XT, HS modes:
    This bit is ignored, CLKOUT function is disabled. Oscillator function on the CLKOUT pin.
  All other FOSC modes:
    1 = CLKOUT function is disabled. I/O function on the CLKOUT pin.
    0 = CLKOUT function is enabled on the CLKOUT pin

bit 10-9    **BOREN<1:0>:** Brown-out Reset Enable bits
  11 = BOR enabled
  10 = BOR enabled during operation and disabled in Sleep
  01 = BOR controlled by SBOREN bit of the BORCON register
  00 = BOR disabled

bit 8      **CPD:** Data Code Protection bit[1]
  1 = Data memory code protection is disabled
  0 = Data memory code protection is enabled

bit 7      **CP:** Code Protection bit
  1 = Program memory code protection is disabled
  0 = Program memory code protection is enabled

bit 6      **MCLRE:** MCLR/VPP Pin Function Select bit
  If LVP bit = 1:
    This bit is ignored.
  If LVP bit = 0:
    1 = MCLR/VPP pin function is MCLR; Weak pull-up enabled.
    0 = MCLR/VPP pin function is digital input; MCLR internally disabled; Weak pull-up under control of
       WPUE3 bit.

bit 5      **PWRTE:** Power-up Timer Enable bit
  1 = PWRT disabled
  0 = PWRT enabled

bit 4-3     **WDTE<1:0>:** Watchdog Timer Enable bit
  11 = WDT enabled
  10 = WDT enabled while running and disabled in Sleep
  01 = WDT controlled by the SWDTEN bit in the WDTCON register
  00 = WDT disabled

**REGISTER 4-1:    CONFIG1: CONFIGURATION WORD 1 (CONTINUED)**

bit 2-0          **FOSC<2:0>:** Oscillator Selection bits

111 = ECH: External Clock, High-Power mode (4-20 MHz): device clock supplied to CLKIN pin
110 = ECM: External Clock, Medium-Power mode (0.5-4 MHz): device clock supplied to CLKIN pin
101 = ECL: External Clock, Low-Power mode (0-0.5 MHz): device clock supplied to CLKIN pin
100 = INTOSC oscillator: I/O function on CLKIN pin
011 = EXTRC oscillator: External RC circuit connected to CLKIN pin
010 = HS oscillator: High-speed crystal/resonator connected between OSC1 and OSC2 pins
001 = XT oscillator: Crystal/resonator connected between OSC1 and OSC2 pins
000 =LP oscillator: Low-power crystal connected between OSC1 and OSC2 pins

**Note  1:**    The entire data EEPROM will be erased when the code protection is turned off during an erase.Once the Data Code Protection bit is enabled, ($\overline{CPD}$ = 0), the Bulk Erase Program Memory Command (through ICSP) can disable the Data Code Protection ($\overline{CPD}$ =1). When a Bulk Erase Program Memory Command is executed, the entire Program Flash Memory, Data EEPROM and configuration memory will be erased.

**REGISTER 4-2:** **CONFIG2: CONFIGURATION WORD 2**

| R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|-------|-------|-------|-------|-------|-------|
| LVP | DEBUG | LPBOR | BORV | STVREN | PLLEN |

bit 13                                                                    bit 8

| U-1 | U-1 | R/P-1 | U-1 | U-1 | U-1 | R/P-1 | R/P-1 |
|-----|-----|-------|-----|-----|-----|-------|-------|
| — | — | VCAPEN | — | — | — | WRT<1:0> | |

bit 7                                                                    bit 0

| Legend: | | |
|---------|---|---|
| R = Readable bit | P = Programmable bit | U = Unimplemented bit, read as '1' |
| '0' = Bit is cleared | '1' = Bit is set | -n = Value when blank or after Bulk Erase |

bit 13     **LVP:** Low-Voltage Programming Enable bit[1]
           `1` = Low-voltage programming enabled
           `0` = High-voltage on $\overline{MCLR}$ must be used for programming

bit 12     **DEBUG:** In-Circuit Debugger Mode bit[3]
           `1` = In-Circuit Debugger disabled, ICSPCLK and ICSPDAT are general purpose I/O pins
           `0` = In-Circuit Debugger enabled, ICSPCLK and ICSPDAT are dedicated to the debugger

bit 11     **LPBOR:** Low-Power BOR Enable bit
           `1` = Low-Power Brown-out Reset is disabled
           `0` = Low-Power Brown-out Reset is enabled

bit 10     **BORV:** Brown-out Reset Voltage Selection bit[4]
           `1` = Brown-out Reset voltage (Vbor), low trip point selected.
           `0` = Brown-out Reset voltage (Vbor), high trip point selected.

bit 9      **STVREN:** Stack Overflow/Underflow Reset Enable bit
           `1` = Stack Overflow or Underflow will cause a Reset
           `0` = Stack Overflow or Underflow will not cause a Reset

bit 8      **PLLEN:** PLL Enable bit
           `1` = 4xPLL enabled
           `0` = 4xPLL disabled

bit 7-6    **Unimplemented:** Read as '`1`'

bit 5      **VCAPEN:** Voltage Regulator Capacitor Enable bit[2]
           `1` = VCAP functionality is disabled on RA6
           `0` = VCAP functionality is enabled on RA6

bit 4-2    **Unimplemented:** Read as '`1`'

bit 1-0    **WRT<1:0>:** Flash Memory Self-Write Protection bits
           4 kW Flash memory (PIC16(L)F1782 only):
               `11` =   Write protection off
               `10` =   000h to 1FFh write-protected, 200h to 7FFh may be modified by EECON control
               `01` =   000h to 3FFh write-protected, 400h to 7FFh may be modified by EECON control
               `00` =   000h to 7FFh write-protected, no addresses may be modified by EECON control
           8 kW Flash memory (PIC16(L)F1783 only):
               `11` =   Write protection off
               `10` =   000h to 1FFh write-protected, 200h to FFFh may be modified by EECON control
               `01` =   000h to 7FFh write-protected, 800h to FFFh may be modified by EECON control
               `00` =   000h to FFFh write-protected, no addresses may be modified by EECON control

**Note 1:** The LVP bit cannot be programmed to '`0`' when Programming mode is entered via LVP.
     **2:** Not implemented on PIC16LF1782/3.
     **3:** The DEBUG bit in Configuration Words is managed automatically by device development tools including debuggers and programmers. For normal device operation, this bit should be maintained as a '`1`'.
     **4:** See Vbor parameter for specific trip point voltages.

## 4.3 Code Protection

Code protection allows the device to be protected from unauthorized access. Program memory protection and data EEPROM protection are controlled independently. Internal access to the program memory and data EEPROM are unaffected by any code protection setting.

### 4.3.1 PROGRAM MEMORY PROTECTION

The entire program memory space is protected from external reads and writes by the $\overline{CP}$ bit in Configuration Words. When $\overline{CP}$ = 0, external reads and writes of program memory are inhibited and a read will return all '0's. The CPU can continue to read program memory, regardless of the protection bit settings. Writing the program memory is dependent upon the write protection setting. See **Section 4.4 "Write Protection"** for more information.

### 4.3.2 DATA EEPROM PROTECTION

The entire data EEPROM is protected from external reads and writes by the $\overline{CPD}$ bit. When $\overline{CPD}$ = 0, external reads and writes of data EEPROM are inhibited. The CPU can continue to read and write data EEPROM regardless of the protection bit settings.

## 4.4 Write Protection

Write protection allows the device to be protected from unintended self-writes. Applications, such as boot loader software, can be protected while allowing other regions of the program memory to be modified.

The WRT<1:0> bits in Configuration Words define the size of the program memory block that is protected.

## 4.5 User ID

Four memory locations (8000h-8003h) are designated as ID locations where the user can store checksum or other code identification numbers. These locations are readable and writable during normal execution. See **Section 12.5 "User ID, Device ID and Configuration Word Access"** for more information on accessing these memory locations. For more information on checksum calculation, see the *"PIC16(L)F178X Memory Programming Specification"* (DS41457).

# PIC16(L)F1782/3

## 4.6 Device ID and Revision ID

The memory location 8006h is where the Device ID and Revision ID are stored. The upper nine bits hold the Device ID. The lower five bits hold the Revision ID. See **Section 12.5 "User ID, Device ID and Configuration Word Access"** for more information on accessing these memory locations.

Development tools, such as device programmers and debuggers, may be used to read the Device ID and Revision ID.

## 4.7 Register Definitions: Device ID

**REGISTER 4-3: DEVICEID: DEVICE ID REGISTER**

| R | R | R | R | R | R |
|---|---|---|---|---|---|
| DEV<8:3> |
| bit 13 | | | | | bit 8 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| DEV<2:0> | | | REV<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | | |
| '1' = Bit is set | '0' = Bit is cleared | -n = Value when blank or after Bulk Erase |

bit 13-5     **DEV<8:0>:** Device ID bits

| Device | DEVICEID<13:0> Values | |
|---|---|---|
| | **DEV<8:0>** | **REV<4:0>** |
| PIC16F1782 | 10 1010 000 | x xxxx |
| PIC16LF1782 | 10 1010 101 | x xxxx |
| PIC16F1783 | 10 1010 001 | x xxxx |
| PIC16LF1783 | 10 1010 110 | x xxxx |

bit 4-0     **REV<4:0>:** Revision ID bits

       These bits are used to identify the revision (see Table under DEV<8:0> above).

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 5.0 RESETS

There are multiple ways to reset this device:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Low Power Brown-out Reset (LPBOR)
- $\overline{\text{MCLR}}$ Reset
- WDT Reset
- RESET instruction
- Stack Overflow
- Stack Underflow
- Programming mode exit

To allow VDD to stabilize, an optional power-up timer can be enabled to extend the Reset time after a BOR or POR event.

A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 5-1.

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



**Note 1:** See Table 5-1 for BOR active conditions.
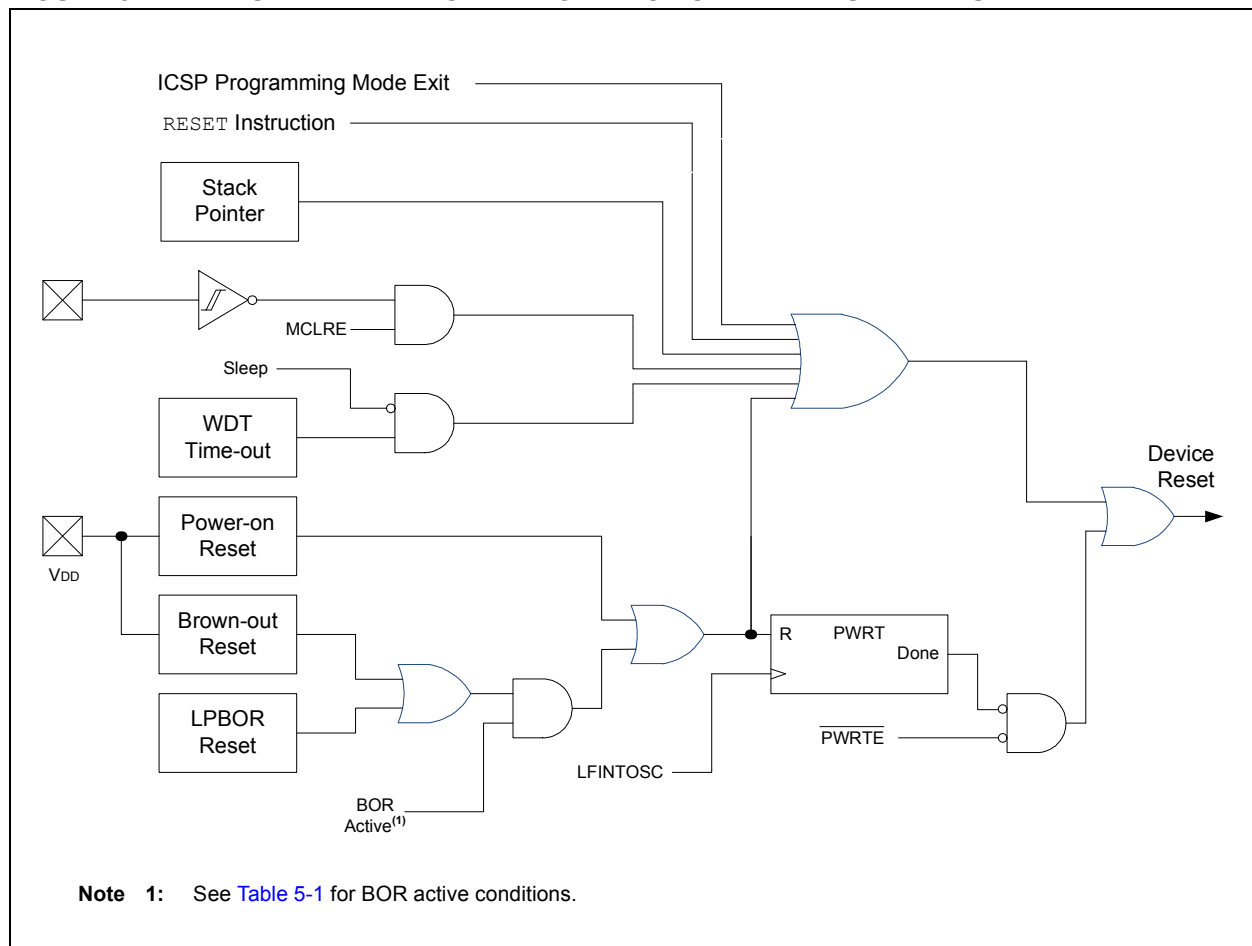
## 5.1 Power-On Reset (POR)

The POR circuit holds the device in Reset until V$_{DD}$ has reached an acceptable level for minimum operation. Slow rising V$_{DD}$, fast operating speeds or analog performance may require greater than minimum V$_{DD}$. The PWRT, BOR or $\overline{MCLR}$ features can be used to extend the start-up period until all device operation conditions have been met.

### 5.1.1 POWER-UP TIMER (PWRT)

The Power-up Timer provides a nominal 64 ms time-out on POR or Brown-out Reset.

The device is held in Reset as long as PWRT is active. The PWRT delay allows additional time for the V$_{DD}$ to rise to an acceptable level. The Power-up Timer is enabled by clearing the $\overline{PWRTE}$ bit in Configuration Words.

The Power-up Timer starts after the release of the POR and BOR.

For additional information, refer to Application Note AN607, *"Power-up Trouble Shooting"* (DS00607).

## 5.2 Brown-Out Reset (BOR)

The BOR circuit holds the device in Reset when V$_{DD}$ reaches a selectable minimum level. Between the POR and BOR, complete voltage range coverage for execution protection can be implemented.

The Brown-out Reset module has four operating modes controlled by the BOREN<1:0> bits in Configuration Words. The four operating modes are:

- BOR is always on
- BOR is off when in Sleep
- BOR is controlled by software
- BOR is always off

Refer to Table 5-1 for more information.

The Brown-out Reset voltage level is selectable by configuring the BORV bit in Configuration Words.

A V$_{DD}$ noise rejection filter prevents the BOR from triggering on small events. If V$_{DD}$ falls below V$_{BOR}$ for a duration greater than parameter T$_{BORDC}$, the device will reset. See Figure 5-2 for more information.

**TABLE 5-1: BOR OPERATING MODES**

| BOREN<1:0> | SBOREN | Device Mode | BOR Mode | Instruction Execution upon: Release of POR or Wake-up from Sleep |
|---|---|---|---|---|
| 11 | X | X | Active | Waits for BOR ready[1] (BORRDY = 1) |
| 10 | X | Awake | Active | Waits for BOR ready (BORRDY = 1) |
| | | Sleep | Disabled | |
| 01 | 1 | X | Active | Waits for BOR ready[1] (BORRDY = 1) |
| | 0 | X | Disabled | Begins immediately (BORRDY = x) |
| 00 | X | X | Disabled | |

**Note 1:** In these specific cases, "Release of POR" and "Wake-up from Sleep", there is no delay in start-up. The BOR ready flag, (BORRDY = 1), will be set before the CPU is ready to execute instructions because the BOR circuit is forced on by the BOREN<1:0> bits.

### 5.2.1 BOR IS ALWAYS ON

When the BOREN bits of Configuration Words are programmed to '11', the BOR is always on. The device start-up will be delayed until the BOR is ready and V$_{DD}$ is higher than the BOR threshold.

BOR protection is active during Sleep. The BOR does not delay wake-up from Sleep.

### 5.2.2 BOR IS OFF IN SLEEP

When the BOREN bits of Configuration Words are programmed to '10', the BOR is on, except in Sleep. The device start-up will be delayed until the BOR is ready and V$_{DD}$ is higher than the BOR threshold.

BOR protection is not active during Sleep. The device wake-up will be delayed until the BOR is ready.

### 5.2.3 BOR CONTROLLED BY SOFTWARE

When the BOREN bits of Configuration Words are programmed to '01', the BOR is controlled by the SBOREN bit of the BORCON register. The device start-up is not delayed by the BOR ready condition or the V$_{DD}$ level.

BOR protection begins as soon as the BOR circuit is ready. The status of the BOR circuit is reflected in the BORRDY bit of the BORCON register.

BOR protection is unchanged by Sleep.

**Preliminary**

**FIGURE 5-2:** **BROWN-OUT SITUATIONS**



**Note 1:** T$_{PWRT}$ delay only if $\overline{PWRTE}$ bit is programmed to '0'.
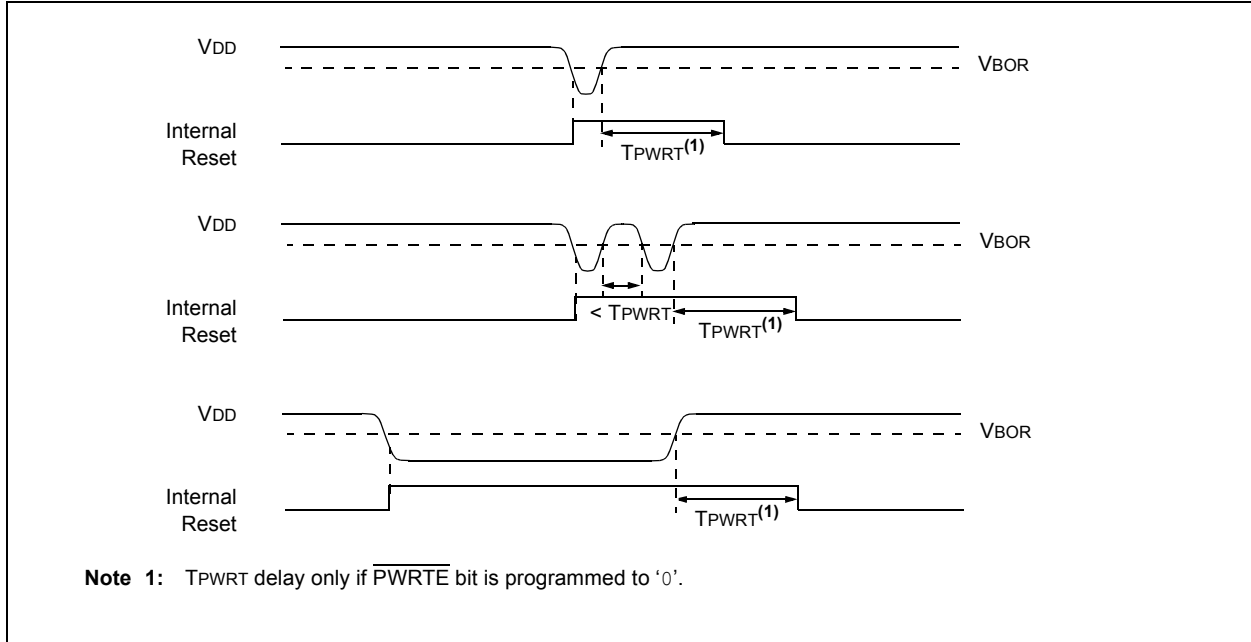
## 5.3 Register Definitions: BOR Control

**REGISTER 5-1:** **BORCON: BROWN-OUT RESET CONTROL REGISTER**

| R/W-1/u | R/W-0/u | U-0 | U-0 | U-0 | U-0 | U-0 | R-q/u |
|---------|---------|-----|-----|-----|-----|-----|-------|
| SBOREN | BORFS | — | — | — | — | — | BORRDY |
| 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 | q = Value depends on condition |

bit 7      **SBOREN:** Software Brown-out Reset Enable bit
<u>If BOREN <1:0> in Configuration Words $\neq$ 01</u>:
SBOREN is read/write, but has no effect on the BOR.
<u>If BOREN <1:0> in Configuration Words = 01</u>:
1 = BOR Enabled
0 = BOR Disabled

bit 6      **BORFS:** Brown-out Reset Fast Start bit[1]
<u>If BOREN<1:0> = 11 (Always on) or BOREN<1:0> = 00 (Always off)</u>
BORFS is Read/Write, but has no effect.
<u>If BOREN <1:0> = 10 (Disabled in Sleep) or BOREN<1:0> = 01 (Under software control):</u>
1 = Band gap is forced on always (covers sleep/wake-up/operating cases)
0 = Band gap operates normally, and may turn off

bit 5-1      **Unimplemented:** Read as '0'

bit 0      **BORRDY:** Brown-out Reset Circuit Ready Status bit
1 = The Brown-out Reset circuit is active
0 = The Brown-out Reset circuit is inactive

**Note 1:** BOREN<1:0> bits are located in Configuration Words.

## 5.4 Low-Power Brown-Out Reset (LPBOR)

The Low-Power Brown-Out Reset (LPBOR) is an essential part of the Reset subsystem. Refer to Figure 5-1 to see how the BOR interacts with other modules.

The LPBOR is used to monitor the external VDD pin. When too low of a voltage is detected, the device is held in Reset. When this occurs, a register bit ($\overline{BOR}$) is changed to indicate that a BOR Reset has occurred. The same bit is set for both the BOR and the LPBOR. Refer to Register 5-2.

### 5.4.1 ENABLING LPBOR

The LPBOR is controlled by the $\overline{LPBOR}$ bit of Configuration Words. When the device is erased, the LPBOR module defaults to disabled.

#### 5.4.1.1 LPBOR Module Output

The output of the LPBOR module is a signal indicating whether or not a Reset is to be asserted. This signal is OR'd together with the Reset signal of the BOR module to provide the generic $\overline{BOR}$ signal, which goes to the PCON register and to the power control block.

## 5.5 $\overline{MCLR}$

The $\overline{MCLR}$ is an optional external input that can reset the device. The $\overline{MCLR}$ function is controlled by the MCLRE bit of Configuration Words and the LVP bit of Configuration Words (Table 5-2).

**TABLE 5-2: $\overline{MCLR}$ CONFIGURATION**

| MCLRE | LVP | $\overline{MCLR}$ |
|-------|-----|------|
| 0 | 0 | Disabled |
| 1 | 0 | Enabled |
| x | 1 | Enabled |

### 5.5.1 $\overline{MCLR}$ ENABLED

When $\overline{MCLR}$ is enabled and the pin is held low, the device is held in Reset. The $\overline{MCLR}$ pin is connected to VDD through an internal weak pull-up.

The device has a noise filter in the $\overline{MCLR}$ Reset path. The filter will detect and ignore small pulses.

> **Note:** A Reset does not drive the $\overline{MCLR}$ pin low.

### 5.5.2 $\overline{MCLR}$ DISABLED

When $\overline{MCLR}$ is disabled, the pin functions as a general purpose input and the internal weak pull-up is under software control. See **Section 13.9 "PORTE Registers"** for more information.

## 5.6 Watchdog Timer (WDT) Reset

The Watchdog Timer generates a Reset if the firmware does not issue a `CLRWDT` instruction within the time-out period. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register are changed to indicate the WDT Reset. See **Section 11.0 "Watchdog Timer (WDT)"** for more information.

## 5.7 `RESET` Instruction

A `RESET` instruction will cause a device Reset. The $\overline{RI}$ bit in the PCON register will be set to '0'. See Table 5-4 for default conditions after a `RESET` instruction has occurred.

## 5.8 Stack Overflow/Underflow Reset

The device can reset when the Stack Overflows or Underflows. The STKOVF or STKUNF bits of the PCON register indicate the Reset condition. These Resets are enabled by setting the STVREN bit in Configuration Words. See **Section 5.8 "Stack Overflow/Underflow Reset"** for more information.

## 5.9 Programming Mode Exit

Upon exit of Programming mode, the device will behave as if a POR had just occurred.

## 5.10 Power-up Timer

The Power-up Timer optionally delays device execution after a BOR or POR event. This timer is typically used to allow VDD to stabilize before allowing the device to start running.

The Power-up Timer is controlled by the $\overline{PWRTE}$ bit of Configuration Words.

## 5.11 Start-up Sequence

Upon the release of a POR or BOR, the following must occur before the device will begin executing:

1. Power-up Timer runs to completion (if enabled).
2. Oscillator start-up timer runs to completion (if required for oscillator source).
3. $\overline{MCLR}$ must be released (if enabled).

The total time-out will vary based on oscillator configuration and Power-up Timer configuration. See **Section 6.0 "Oscillator Module (with Fail-Safe Clock Monitor)"** for more information.

The Power-up Timer and oscillator start-up timer run independently of $\overline{MCLR}$ Reset. If $\overline{MCLR}$ is kept low long enough, the Power-up Timer and oscillator start-up timer will expire. Upon bringing $\overline{MCLR}$ high, the device will begin execution immediately (see Figure 5-3). This is useful for testing purposes or to synchronize more than one device operating in parallel.

**Preliminary**  © 2011-2012 Microchip Technology Inc.

**FIGURE 5-3:** **RESET START-UP SEQUENCE**
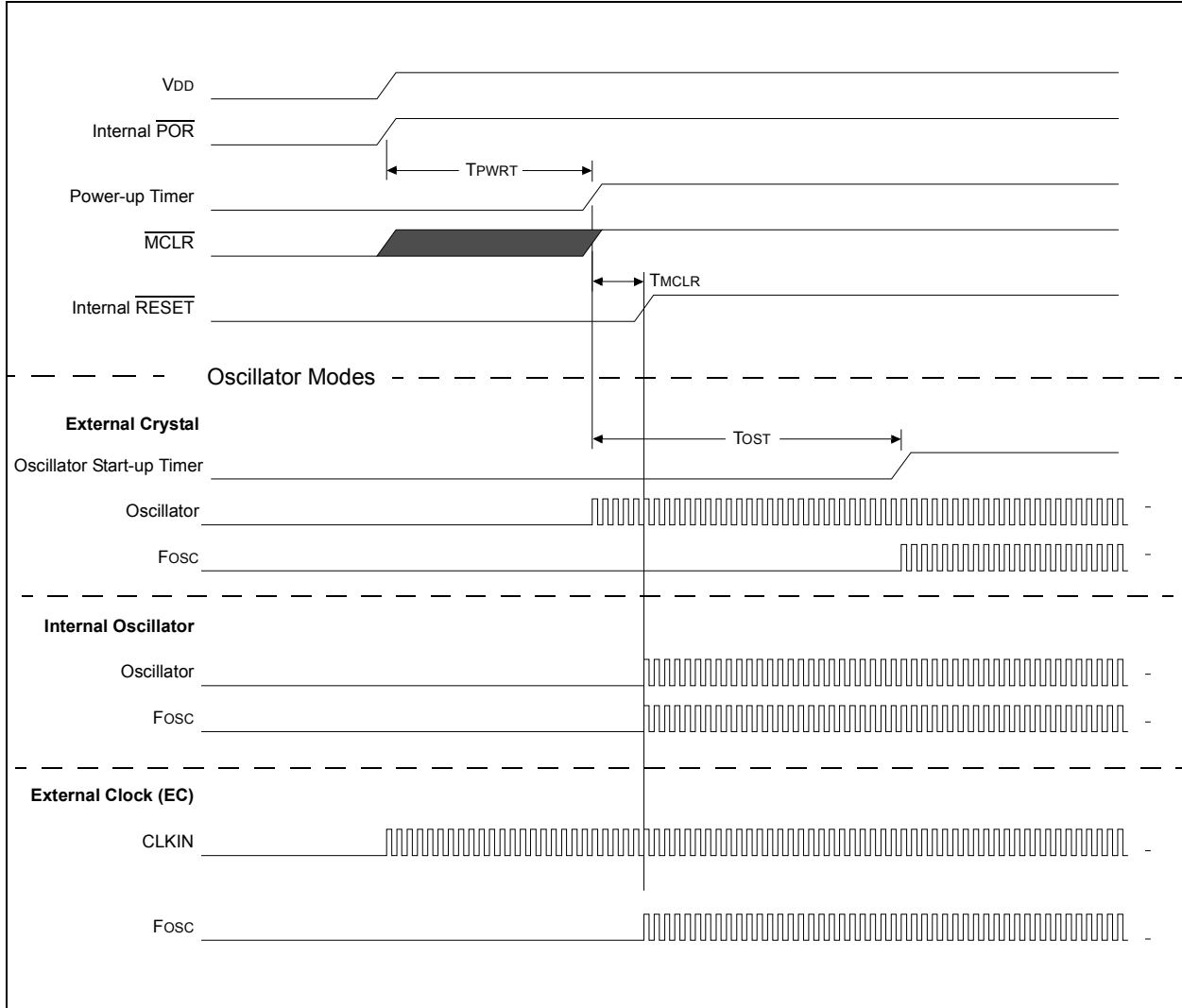
## 5.12 Determining the Cause of a Reset

Upon any Reset, multiple bits in the STATUS and PCON register are updated to indicate the cause of the Reset. Table 5-3 and Table 5-4 show the Reset conditions of these registers.

**TABLE 5-3:** **RESET STATUS BITS AND THEIR SIGNIFICANCE**

| STKOVF | STKUNF | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | POR | BOR | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | Condition |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | x | 1 | 1 | Power-on Reset |
| 0 | 0 | 1 | 1 | 1 | 0 | x | 0 | x | Illegal, $\overline{\text{TO}}$ is set on $\overline{\text{POR}}$ |
| 0 | 0 | 1 | 1 | 1 | 0 | x | x | 0 | Illegal, $\overline{\text{PD}}$ is set on $\overline{\text{POR}}$ |
| 0 | 0 | u | 1 | 1 | u | 0 | 1 | 1 | Brown-out Reset |
| u | u | 0 | u | u | u | u | 0 | u | WDT Reset |
| u | u | u | u | u | u | u | 0 | 0 | WDT Wake-up from Sleep |
| u | u | u | u | u | u | u | 1 | 0 | Interrupt Wake-up from Sleep |
| u | u | u | 0 | u | u | u | u | u | $\overline{\text{MCLR}}$ Reset during normal operation |
| u | u | u | 0 | u | u | u | 1 | 0 | $\overline{\text{MCLR}}$ Reset during Sleep |
| u | u | u | u | 0 | u | u | u | u | RESET Instruction Executed |
| 1 | u | u | u | u | u | u | u | u | Stack Overflow Reset (STVREN = 1) |
| u | 1 | u | u | u | u | u | u | u | Stack Underflow Reset (STVREN = 1) |

**TABLE 5-4:** **RESET CONDITION FOR SPECIAL REGISTERS**

| Condition | Program Counter | STATUS Register | PCON Register |
|---|---|---|---|
| Power-on Reset | 0000h | ---1 1000 | 00-1 110x |
| $\overline{\text{MCLR}}$ Reset during normal operation | 0000h | ---u uuuu | uu-u 0uuu |
| $\overline{\text{MCLR}}$ Reset during Sleep | 0000h | ---1 0uuu | uu-u 0uuu |
| WDT Reset | 0000h | ---0 uuuu | uu-0 uuuu |
| WDT Wake-up from Sleep | PC + 1 | ---0 0uuu | uu-u uuuu |
| Brown-out Reset | 0000h | ---1 1uuu | 00-1 11u0 |
| Interrupt Wake-up from Sleep | PC + 1[1] | ---1 0uuu | uu-u uuuu |
| RESET Instruction Executed | 0000h | ---u uuuu | uu-u u0uu |
| Stack Overflow Reset (STVREN = 1) | 0000h | ---u uuuu | 1u-u uuuu |
| Stack Underflow Reset (STVREN = 1) | 0000h | ---u uuuu | u1-u uuuu |

**Legend:** u = unchanged, x = unknown, – = unimplemented bit, reads as '0'.

**Note 1:** When the wake-up is due to an interrupt and Global Enable bit (GIE) is set, the return address is pushed on the stack and PC is loaded with the interrupt vector (0004h) after execution of PC + 1.

**Preliminary**  © 2011-2012 Microchip Technology Inc.

## 5.13 Power Control (PCON) Register

The Power Control (PCON) register contains flag bits to differentiate between a:

- Power-on Reset ($\overline{POR}$)
- Brown-out Reset ($\overline{BOR}$)
- Reset Instruction Reset ($\overline{RI}$)
- $\overline{MCLR}$ Reset ($\overline{RMCLR}$)
- Watchdog Timer Reset ($\overline{RWDT}$)
- Stack Underflow Reset (STKUNF)
- Stack Overflow Reset (STKOVF)

The PCON register bits are shown in Register 5-2.

## 5.14 Register Definitions: Power Control

### REGISTER 5-2: PCON: POWER CONTROL REGISTER

| R/W/HS-0/q | R/W/HS-0/q | U-0 | R/W/HC-1/q | R/W/HC-1/q | R/W/HC-1/q | R/W/HC-q/u | R/W/HC-q/u |
|---|---|---|---|---|---|---|---|
| STKOVF | STKUNF | — | $\overline{RWDT}$ | $\overline{RMCLR}$ | $\overline{RI}$ | $\overline{POR}$ | $\overline{BOR}$ |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---|---|---|
| HC = Bit is cleared by hardware | | HS = Bit is set by hardware |
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -m/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7 **STKOVF:** Stack Overflow Flag bit

`1` = A Stack Overflow occurred
`0` = A Stack Overflow has not occurred or cleared by firmware

bit 6 **STKUNF:** Stack Underflow Flag bit

`1` = A Stack Underflow occurred
`0` = A Stack Underflow has not occurred or cleared by firmware

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

bit 4 **$\overline{RWDT}$:** Watchdog Timer Reset Flag bit

`1` = A Watchdog Timer Reset has not occurred or set to '`1`' by firmware
`0` = A Watchdog Timer Reset has occurred (cleared by hardware)

bit 3 **$\overline{RMCLR}$:** $\overline{MCLR}$ Reset Flag bit

`1` = A $\overline{MCLR}$ Reset has not occurred or set to '`1`' by firmware
`0` = A $\overline{MCLR}$ Reset has occurred (cleared by hardware)

bit 2 **$\overline{RI}$:** `RESET` Instruction Flag bit

`1` = A `RESET` instruction has not been executed or set to '`1`' by firmware
`0` = A `RESET` instruction has been executed (cleared by hardware)

bit 1 **$\overline{POR}$:** Power-on Reset Status bit

`1` = No Power-on Reset occurred
`0` = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit

`1` = No Brown-out Reset occurred
`0` = A Brown-out Reset occurred (must be set in software after a Power-on Reset or Brown-out Reset occurs)

# PIC16(L)F1782/3

**TABLE 5-5: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| BORCON | SBOREN | BORFS | — | — | — | — | — | BORRDY | 51 |
| PCON | STKOVF | STKUNF | — | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 55 |
| STATUS | — | — | — | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | Z | DC | C | 23 |
| WDTCON | — | — | WDTPS<4:0> | | | | | SWDTEN | 101 |

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

## 6.0 OSCILLATOR MODULE (WITH FAIL-SAFE CLOCK MONITOR)

### 6.1 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 6-1 illustrates a block diagram of the oscillator module.

Clock sources can be supplied from external oscillators, quartz crystal resonators, ceramic resonators and Resistor-Capacitor (RC) circuits. In addition, the system clock source can be supplied from one of two internal oscillators and PLL circuits, 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 oscillator module can be configured in one of eight clock modes.

1. ECL – External Clock Low-Power mode (0 MHz to 0.5 MHz)
2. ECM – External Clock Medium-Power mode (0.5 MHz to 4 MHz)
3. ECH – External Clock High-Power mode (4 MHz to 32 MHz)
4. LP – 32 kHz Low-Power Crystal mode.
5. XT – Medium Gain Crystal or Ceramic Resonator Oscillator mode (up to 4 MHz)
6. HS – High Gain Crystal or Ceramic Resonator mode (4 MHz to 20 MHz)
7. RC – External Resistor-Capacitor (RC).
8. INTOSC – Internal oscillator (31 kHz to 32 MHz).

Clock Source modes are selected by the FOSC<2:0> bits in the Configuration Words. The FOSC bits determine the type of oscillator that will be used when the device is first powered.

The EC clock mode relies on an external logic level signal as the device clock source. The LP, XT, and HS clock modes require an external crystal or resonator to be connected to the device. Each mode is optimized for a different frequency range. The RC clock mode requires an external resistor and capacitor to set the oscillator frequency.

The INTOSC internal oscillator block produces low, medium, and high-frequency clock sources, designated LFINTOSC, MFINTOSC and HFINTOSC. (see Internal Oscillator Block, Figure 6-1). A wide selection of device clock frequencies may be derived from these three clock sources.

# PIC16(L)F1782/3

**FIGURE 6-1:** **SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM**



| SCS | FOSC<2:0> | PLLEN or SPLLEN | PRIMUX | PSMCMUX | PLLMUX |
|-----|-----------|-----------------|--------|---------|--------|
| =00 | =100 | 0 | 1 | 1 | 10 |
| | | 1 | 1 | 1 | 01 |
| | ≠100 | 0 | 0 | 1 | 10 |
| | | 1 **(1)** | 0 | 0 | 00 |
| ≠00 | XXX | X | X | 1 | XX |

**Note 1:** This selection should not be made when the PSMC is using the 64 MHz clock option.
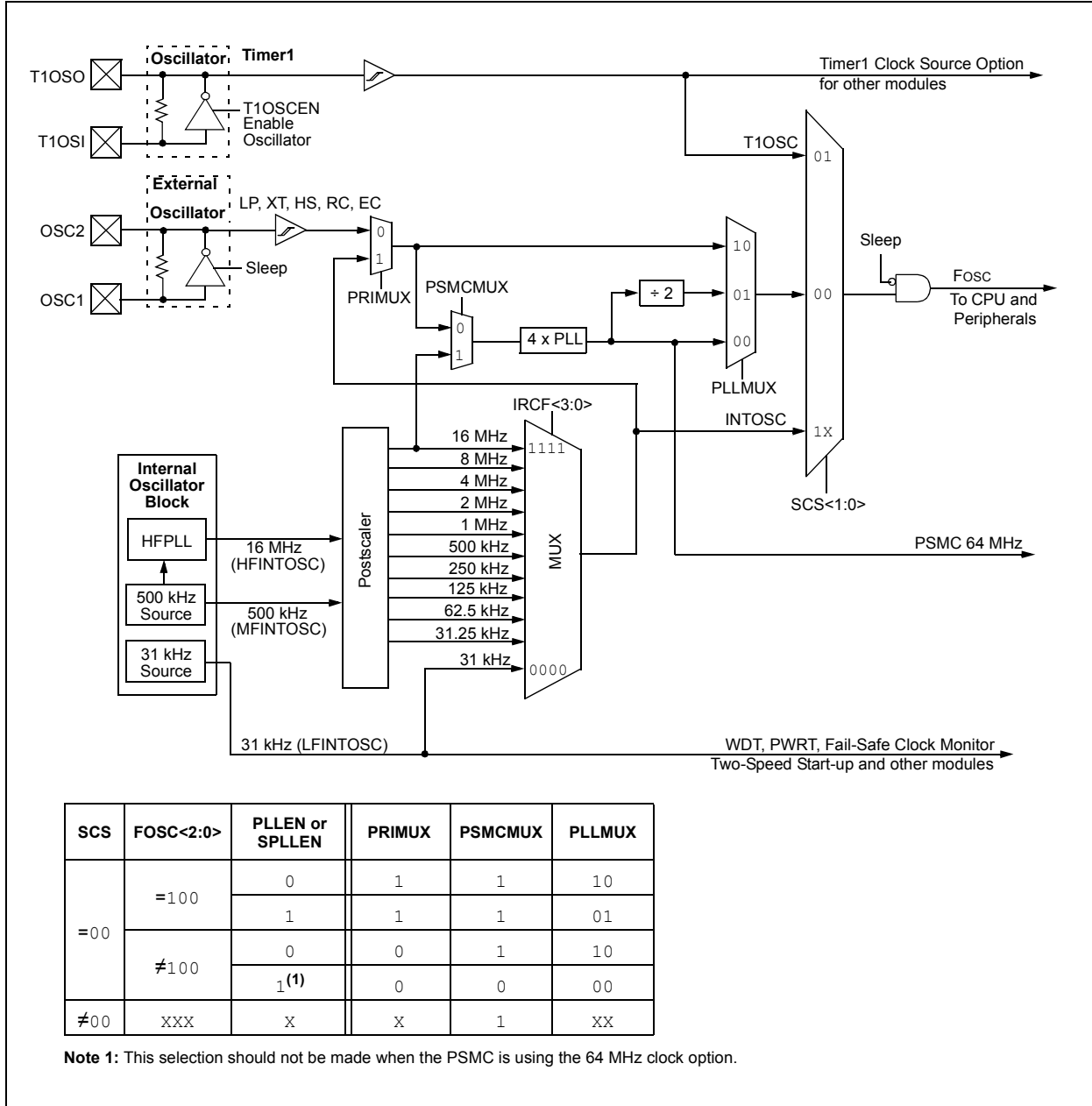
## 6.2 Clock Source Types

Clock sources can be classified as external or internal.

External clock sources rely on external circuitry for the clock source to function. Examples are: oscillator 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 within the oscillator module. The internal oscillator block has two internal oscillators and a dedicated Phase-Lock Loop (HFPLL) that are used to generate three internal system clock sources: the 16 MHz High-Frequency Internal Oscillator (HFINTOSC), 500 kHz (MFINTOSC) and the 31 kHz Low-Frequency Internal Oscillator (LFINTOSC).

The system clock can be selected between external or internal clock sources via the System Clock Select (SCS) bits in the OSCCON register. See **Section 6.3 "Clock Switching"** for additional information.

### 6.2.1 EXTERNAL CLOCK SOURCES

An external clock source can be used as the device system clock by performing one of the following actions:

• Program the FOSC<2:0> bits in the Configuration Words to select an external clock source that will be used as the default system clock upon a device Reset.
• Write the SCS<1:0> bits in the OSCCON register to switch the system clock source to:
  - Timer1 oscillator during run-time, or
  - An external clock source determined by the value of the FOSC bits.

See **Section 6.3 "Clock Switching"** for more information.

#### 6.2.1.1 EC Mode

The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the OSC1 input. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. Figure 6-2 shows the pin connections for EC mode.

EC mode has 3 power modes to select from through Configuration Words:

• High power, 4-32 MHz (FOSC = `111`)
• Medium power, 0.5-4 MHz (FOSC = `110`)
• Low power, 0-0.5 MHz (FOSC = `101`)

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.

**FIGURE 6-2: EXTERNAL CLOCK (EC) MODE OPERATION**



Clock from Ext. System → OSC1/CLKIN

**PIC® MCU**

$F_{OSC}/4$ or I/O[1] ← OSC2/CLKOUT

**Note 1:** Output depends upon $\overline{CLKOUTEN}$ bit of the Configuration Words.
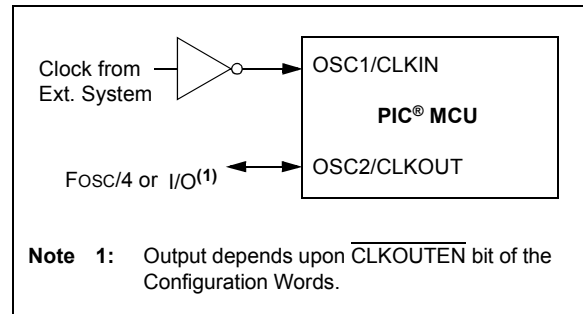
#### 6.2.1.2 LP, XT, HS Modes

The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to OSC1 and OSC2 (Figure 6-3). The three modes select 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 designed to drive only 32.768 kHz tuning-fork type crystals (watch crystals).

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

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

Figure 6-3 and Figure 6-4 show typical circuits for quartz crystal and ceramic resonators, respectively.

# PIC16(L)F1782/3

FIGURE 6-3:      QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)



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

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

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

**2:** Always verify oscillator performance over the $V_{DD}$ and temperature range that is expected for the application.

**3:** For oscillator design assistance, reference the following Microchip Applications Notes:

- AN826, "*Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices*" (DS00826)
- AN849, "*Basic PIC® Oscillator Design*" (DS00849)
- AN943, "*Practical PIC® Oscillator Analysis and Design*" (DS00943)
- AN949, "*Making Your Oscillator Work*" (DS00949)

FIGURE 6-4:      CERAMIC RESONATOR OPERATION (XT OR HS MODE)



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

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

**3:** An additional parallel feedback resistor ($R_P$) may be required for proper ceramic resonator operation.
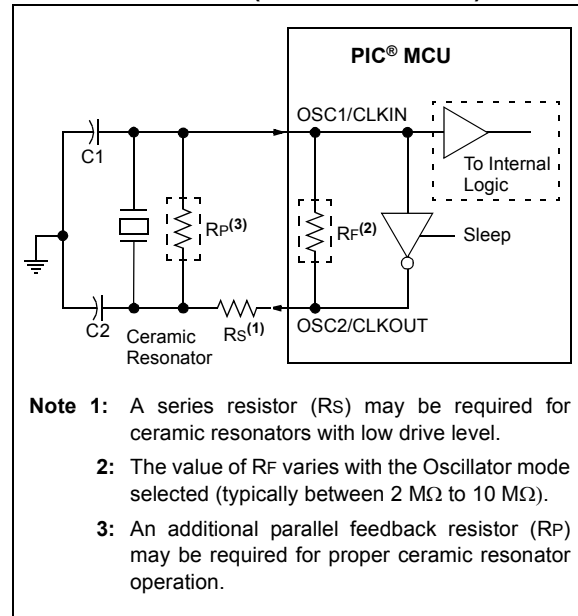
### 6.2.1.3 Oscillator Start-up Timer (OST)

If 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.

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 6.4 "Two-Speed Clock Start-up Mode"**).

**Preliminary**

### 6.2.1.4    4x PLL

The oscillator module contains a 4x PLL that can be used with both external and internal clock sources to provide a system clock source. The input frequency for the 4x PLL must fall within specifications. See the PLL Clock Timing Specifications in **Section 30.0 "Electrical Specifications"**.

The 4x PLL may be enabled for use by one of two methods:

1.  Program the PLLEN bit in Configuration Words to a '1'.
2.  Write the SPLLEN bit in the OSCCON register to a '1'. If the PLLEN bit in Configuration Words is programmed to a '1', then the value of SPLLEN is ignored.

### 6.2.1.5    TIMER1 Oscillator

The Timer1 Oscillator is a separate crystal oscillator that is associated with the Timer1 peripheral. It is optimized for timekeeping operations with a 32.768 kHz crystal connected between the T1OSO and T1OSI device pins.

The Timer1 Oscillator can be used as an alternate system clock source and can be selected during run-time using clock switching. Refer to **Section 6.3 "Clock Switching"** for more information.

| 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 $V_{DD}$ and temperature range that is expected for the application. |
| 3: | For oscillator design assistance, reference the following Microchip Applications Notes: |

- AN826, "*Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices*" (DS00826)
- AN849, "*Basic PIC® Oscillator Design*" (DS00849)
- AN943, "*Practical PIC® Oscillator Analysis and Design*" (DS00943)
- AN949, "*Making Your Oscillator Work*" (DS00949)
- TB097, "*Interfacing a Micro Crystal MS1V-T1K 32.768 kHz Tuning Fork Crystal to a PIC16F690/SS*" (DS91097)
- AN1288, "*Design Practices for Low-Power External Oscillators*" (DS01288)

**FIGURE 6-5:    QUARTZ CRYSTAL OPERATION (TIMER1 OSCILLATOR)**
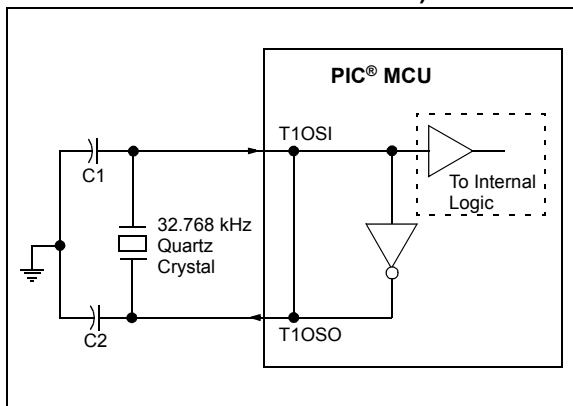
#### 6.2.1.6 External RC Mode

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.

The RC circuit connects to OSC1. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. The function of the OSC2/CLKOUT pin is determined by the CLKOUTEN bit in Configuration Words.

Figure 6-6 shows the external RC mode connections.

**FIGURE 6-6: EXTERNAL RC MODES**



Recommended values: 10 kΩ ≤ REXT ≤ 100 kΩ, <3V
3 kΩ ≤ REXT ≤ 100 kΩ, 3-5V
CEXT > 20 pF, 2-5V

**Note 1:** Output depends upon CLKOUTEN bit of the Configuration Words.
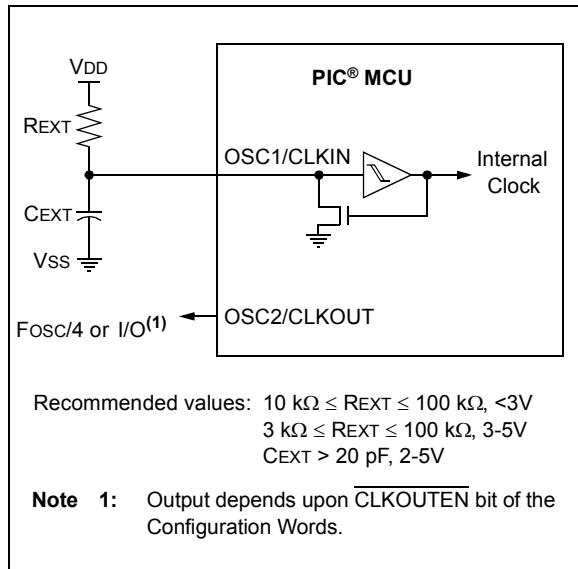
The RC oscillator frequency is a function of the supply voltage, the resistor (REXT) and capacitor (CEXT) values and the operating temperature. Other factors affecting the oscillator frequency are:

- 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.

#### 6.2.2 INTERNAL CLOCK SOURCES

The device may be configured to use the internal oscillator block as the system clock by performing one of the following actions:

- Program the FOSC<2:0> bits in Configuration Words to select the INTOSC clock source, which will be used as the default system clock upon a device Reset.
- Write the SCS<1:0> bits in the OSCCON register to switch the system clock source to the internal oscillator during run-time. See **Section 6.3 "Clock Switching"** for more information.

In **INTOSC** mode, OSC1/CLKIN is available for general purpose I/O. OSC2/CLKOUT is available for general purpose I/O or CLKOUT.

The function of the OSC2/CLKOUT pin is determined by the CLKOUTEN bit in Configuration Words.

The internal oscillator block has two independent oscillators and a dedicated Phase-Lock Loop, HFPLL that can produce one of three internal system clock sources.

1. The **HFINTOSC** (High-Frequency Internal Oscillator) is factory calibrated and operates at 16 MHz. The HFINTOSC source is generated from the 500 kHz MFINTOSC source and the dedicated Phase-Lock Loop, HFPLL. The frequency of the HFINTOSC can be user-adjusted via software using the OSCTUNE register (Register 6-3).

2. The **MFINTOSC** (Medium-Frequency Internal Oscillator) is factory calibrated and operates at 500 kHz. The frequency of the MFINTOSC can be user-adjusted via software using the OSCTUNE register (Register 6-3).

3. The **LFINTOSC** (Low-Frequency Internal Oscillator) is uncalibrated and operates at 31 kHz.

### 6.2.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a factory calibrated 16 MHz internal clock source. The frequency of the HFINTOSC can be altered via software using the OSCTUNE register (Register 6-3).

The output of the HFINTOSC connects to a postscaler and multiplexer (see Figure 6-1). One of multiple frequencies derived from the HFINTOSC can be selected via software using the IRCF<3:0> bits of the OSCCON register. See **Section 6.2.2.7 "Internal Oscillator Clock Switch Timing"** for more information.

The HFINTOSC is enabled by:

- Configure the IRCF<3:0> bits of the OSCCON register for the desired HF frequency, and
- FOSC<2:0> = `100`, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '`1x`'.

A fast startup oscillator allows internal circuits to power up and stabilize before switching to HFINTOSC.

The High Frequency Internal Oscillator Ready bit (HFIOFR) of the OSCSTAT register indicates when the HFINTOSC is running.

The High Frequency Internal Oscillator Status Locked bit (HFIOFL) of the OSCSTAT register indicates when the HFINTOSC is running within 2% of its final value.

The High Frequency Internal Oscillator Stable bit (HFIOFS) of the OSCSTAT register indicates when the HFINTOSC is running within 0.5% of its final value.

### 6.2.2.2 MFINTOSC

The Medium-Frequency Internal Oscillator (MFINTOSC) is a factory calibrated 500 kHz internal clock source. The frequency of the MFINTOSC can be altered via software using the OSCTUNE register (Register 6-3).

The output of the MFINTOSC connects to a postscaler and multiplexer (see Figure 6-1). One of nine frequencies derived from the MFINTOSC can be selected via software using the IRCF<3:0> bits of the OSCCON register. See **Section 6.2.2.7 "Internal Oscillator Clock Switch Timing"** for more information.

The MFINTOSC is enabled by:

- Configure the IRCF<3:0> bits of the OSCCON register for the desired HF frequency, and
- FOSC<2:0> = `100`, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '`1x`'

The Medium Frequency Internal Oscillator Ready bit (MFIOFR) of the OSCSTAT register indicates when the MFINTOSC is running.

### 6.2.2.3 Internal Oscillator Frequency Adjustment

The 500 kHz internal oscillator is factory calibrated. This internal oscillator can be adjusted in software by writing to the OSCTUNE register (Register 6-3). Since the HFINTOSC and MFINTOSC clock sources are derived from the 500 kHz internal oscillator a change in the OSCTUNE register value will apply to both.

The default value of the OSCTUNE register is '`0`'. The value is a 6-bit two's complement number. A value of 1Fh will provide an adjustment to the maximum frequency. A value of 20h will provide an adjustment to the minimum frequency.

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

OSCTUNE does not affect the LFINTOSC frequency. Operation of features that depend on the LFINTOSC clock source frequency, such as the Power-up Timer (PWRT), Watchdog Timer (WDT), Fail-Safe Clock Monitor (FSCM) and peripherals, are *not* affected by the change in frequency.

### 6.2.2.4 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is an uncalibrated 31 kHz internal clock source.

The output of the LFINTOSC connects to a multiplexer (see Figure 6-1). Select 31 kHz, via software, using the IRCF<3:0> bits of the OSCCON register. See **Section 6.2.2.7 "Internal Oscillator Clock Switch Timing"** for more information. The LFINTOSC is also the frequency for the Power-up Timer (PWRT), Watchdog Timer (WDT) and Fail-Safe Clock Monitor (FSCM).

The LFINTOSC is enabled by selecting 31 kHz (IRCF<3:0> bits of the OSCCON register = `000`) as the system clock source (SCS bits of the OSCCON register = `1x`), or when any of the following are enabled:

- Configure the IRCF<3:0> bits of the OSCCON register for the desired LF frequency, and
- FOSC<2:0> = `100`, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '`1x`'

Peripherals that use the LFINTOSC are:

- Power-up Timer (PWRT)
- Watchdog Timer (WDT)
- Fail-Safe Clock Monitor (FSCM)

The Low-Frequency Internal Oscillator Ready bit (LFIOFR) of the OSCSTAT register indicates when the LFINTOSC is running.

# PIC16(L)F1782/3

### 6.2.2.5 Internal Oscillator Frequency Selection

The system clock speed can be selected via software using the Internal Oscillator Frequency Select bits IRCF<3:0> of the OSCCON register.

The output of the 16 MHz HFINTOSC, 500 kHz MFINTOSC, and 31 kHz LFINTOSC connects to a postscaler and multiplexer (see Figure 6-1). The Internal Oscillator Frequency Select bits IRCF<3:0> of the OSCCON register select the frequency output of the internal oscillators. One of the following frequencies can be selected via software:

- 32 MHz (requires 4x PLL)
- 16 MHz
- 8 MHz
- 4 MHz
- 2 MHz
- 1 MHz
- 500 kHz (Default after Reset)
- 250 kHz
- 125 kHz
- 62.5 kHz
- 31.25 kHz
- 31 kHz (LFINTOSC)

> **Note:** Following any Reset, the IRCF<3:0> bits of the OSCCON register are set to '0111' and the frequency selection is set to 500 kHz. The user can modify the IRCF bits to select a different frequency.

The IRCF<3:0> bits of the OSCCON register allow duplicate selections for some frequencies. These duplicate choices can offer system design trade-offs. Lower power consumption can be obtained when changing oscillator sources for a given frequency. Faster transition times can be obtained between frequency changes that use the same oscillator source.

### 6.2.2.6 32 MHz Internal Oscillator Frequency Selection

The Internal Oscillator Block can be used with the 4x PLL associated with the External Oscillator Block to produce a 32 MHz internal system clock source. The following settings are required to use the 32 MHz internal clock source:

- The FOSC bits in Configuration Words must be set to use the INTOSC source as the device system clock (FOSC<2:0> = 100).
- The SCS bits in the OSCCON register must be cleared to use the clock determined by FOSC<2:0> in Configuration Words (SCS<1:0> = 00).
- The IRCF bits in the OSCCON register must be set to the 8 MHz or 16 MHz HFINTOSC set to use (IRCF<3:0> = 111x).
- The SPLLEN bit in the OSCCON register must be set to enable the 4x PLL, or the PLLEN bit of the Configuration Words must be programmed to a '1'.

> **Note:** When using the PLLEN bit of the Configuration Words, the 4x PLL cannot be disabled by software and the SPLLEN option will not be available.

The 4x PLL is not available for use with the internal oscillator when the SCS bits of the OSCCON register are set to '1x'. The SCS bits must be set to '00' to use the 4x PLL with the internal oscillator.

### 6.2.2.7 Internal Oscillator Clock Switch Timing

When switching between the HFINTOSC, MFINTOSC and the LFINTOSC, the new oscillator may already be shut down to save power (see Figure 6-7). If this is the case, there is a delay after the IRCF<3:0> bits of the OSCCON register are modified before the frequency selection takes place. The OSCSTAT register will reflect the current active status of the HFINTOSC, MFINTOSC and LFINTOSC oscillators. The sequence of a frequency selection is as follows:

1. IRCF<3:0> bits of the OSCCON register are modified.
2. If the new clock is shut down, a clock start-up delay is started.
3. Clock switch circuitry waits for a falling edge of the current clock.
4. The current clock is held low and the clock switch circuitry waits for a rising edge in the new clock.
5. The new clock is now active.
6. The OSCSTAT register is updated as required.
7. Clock switch is complete.

See Figure 6-7 for more details.

If the internal oscillator speed is switched between two clocks of the same source, there is no start-up delay before the new frequency is selected. Clock switching time delays are shown in Table 6-1.

Start-up delay specifications are located in the oscillator tables of **Section 30.0 "Electrical Specifications"**.

**FIGURE 6-7: INTERNAL OSCILLATOR SWITCH TIMING**



**Preliminary** © 2011-2012 Microchip Technology Inc.
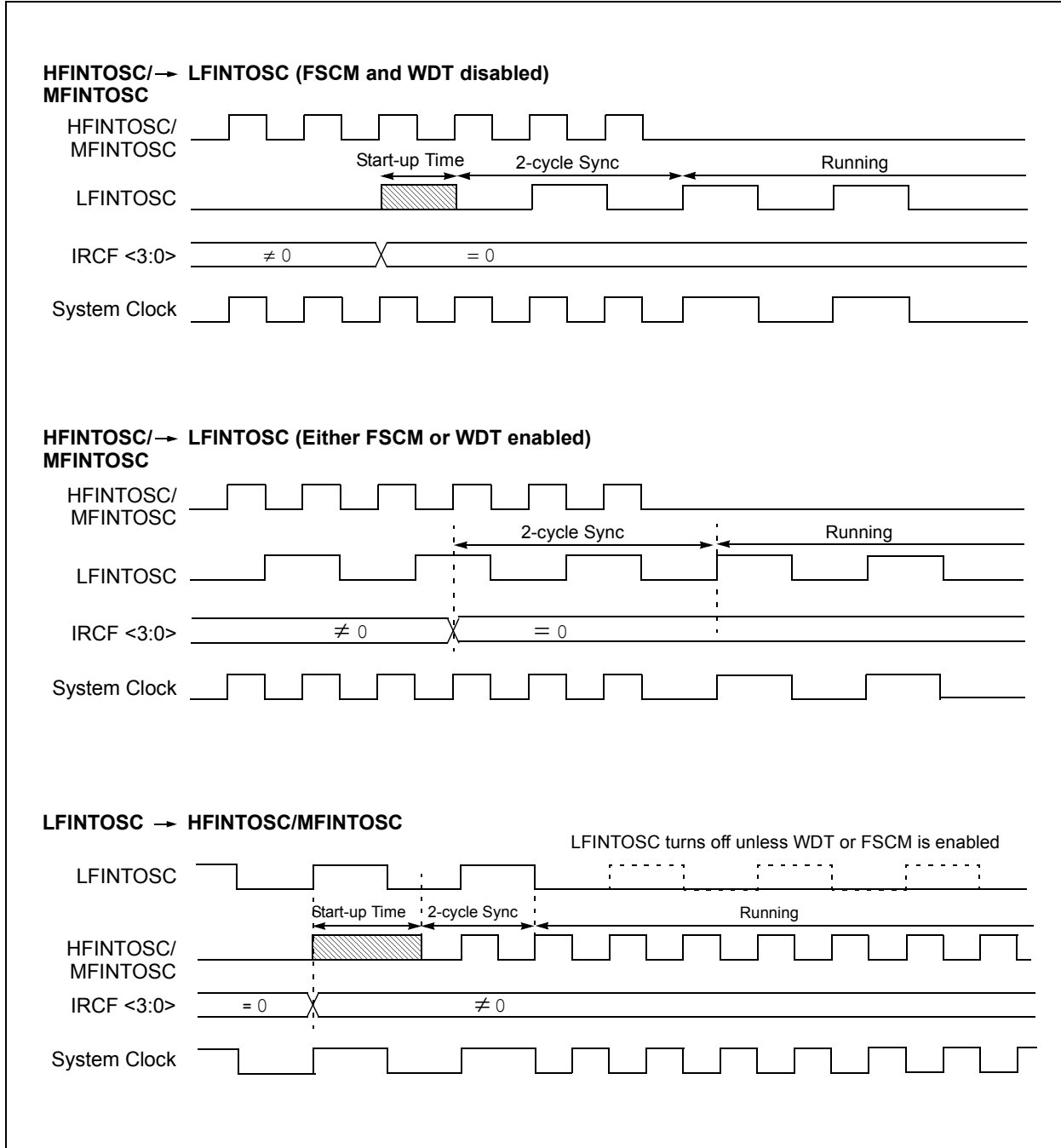
## 6.3    Clock Switching

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

- Default system oscillator determined by FOSC bits in Configuration Words
- Timer1 32 kHz crystal oscillator
- Internal Oscillator Block (INTOSC)

### 6.3.1    SYSTEM CLOCK SELECT (SCS) BITS

The System Clock Select (SCS) bits of the OSCCON register selects the system clock source that is used for the CPU and peripherals.

- When the SCS bits of the OSCCON register = `00`, the system clock source is determined by value of the FOSC<2:0> bits in the Configuration Words.
- When the SCS bits of the OSCCON register = `01`, the system clock source is the Timer1 oscillator.
- When the SCS bits of the OSCCON register = `1x`, the system clock source is chosen by the internal oscillator frequency selected by the IRCF<3:0> bits of the OSCCON register. After a Reset, the SCS 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 bits of the OSCCON register. The user can monitor the OSTS bit of the OSCSTAT register to determine the current system clock source.

When switching between clock sources, a delay is required to allow the new clock to stabilize. These oscillator delays are shown in Table 6-1.

### 6.3.2    OSCILLATOR START-UP TIME-OUT STATUS (OSTS) BIT

The Oscillator Start-up Time-out Status (OSTS) bit of the OSCSTAT register indicates whether the system clock is running from the external clock source, as defined by the FOSC<2:0> bits in the Configuration Words, or from the internal clock source. In particular, OSTS indicates that the Oscillator Start-up Timer (OST) has timed out for LP, XT or HS modes. The OST does not reflect the status of the Timer1 oscillator.

### 6.3.3    TIMER1 OSCILLATOR

The Timer1 oscillator is a separate crystal oscillator associated with the Timer1 peripheral. It is optimized for timekeeping operations with a 32.768 kHz crystal connected between the T1OSO and T1OSI device pins.

The Timer1 oscillator is enabled using the T1OSCEN control bit in the T1CON register. See **Section 22.0 "Timer1 Module with Gate Control"** for more information about the Timer1 peripheral.

### 6.3.4    TIMER1 OSCILLATOR READY (T1OSCR) BIT

The user must ensure that the Timer1 oscillator is ready to be used before it is selected as a system clock source. The Timer1 Oscillator Ready (T1OSCR) bit of the OSCSTAT register indicates whether the Timer1 oscillator is ready to be used. After the T1OSCR bit is set, the SCS bits can be configured to select the Timer1 oscillator.

## 6.4 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 INTOSC internal oscillator block as the clock source and go back to Sleep without waiting for the external oscillator to become stable.

Two-Speed Start-up provides benefits when the oscillator module is configured for LP, XT or HS modes. The Oscillator Start-up Timer (OST) is enabled for these modes and must count 1024 oscillations before the oscillator can be used as the system clock source.

If the oscillator module is configured for any mode other than LP, XT or HS mode, then Two-Speed Start-up is disabled. This is because the external clock oscillator does not require any stabilization time after POR or an exit from Sleep.

If the OST count reaches 1024 before the device enters Sleep mode, the OSTS bit of the OSCSTAT register is set and program execution switches to the external oscillator. However, the system may never operate from the external oscillator if the time spent awake is very short.

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

### 6.4.1 TWO-SPEED START-UP MODE CONFIGURATION

Two-Speed Start-up mode is configured by the following settings:

- IESO (of the Configuration Words) = `1`; Internal/External Switchover bit (Two-Speed Start-up mode enabled).
- SCS (of the OSCCON register) = `00`.
- FOSC<2:0> bits in the Configuration Words configured for LP, XT or HS mode.

Two-Speed Start-up mode is entered after:

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

### TABLE 6-1: OSCILLATOR SWITCHING DELAYS

| Switch From | Switch To | Frequency | Oscillator Delay |
|---|---|---|---|
| Sleep/POR | LFINTOSC[1] MFINTOSC[1] HFINTOSC[1] | 31 kHz 31.25 kHz-500 kHz 31.25 kHz-16 MHz | Oscillator Warm-up Delay ($T_{WARM}$) |
| Sleep/POR | EC, RC[1] | DC – 32 MHz | 2 cycles |
| LFINTOSC | EC, RC[1] | DC – 32 MHz | 1 cycle of each |
| Sleep/POR | Timer1 Oscillator LP, XT, HS[1] | 32 kHz-20 MHz | 1024 Clock Cycles (OST) |
| Any clock source | MFINTOSC[1] HFINTOSC[1] | 31.25 kHz-500 kHz 31.25 kHz-16 MHz | 2 $\mu$s (approx.) |
| Any clock source | LFINTOSC[1] | 31 kHz | 1 cycle of each |
| Any clock source | Timer1 Oscillator | 32 kHz | 1024 Clock Cycles (OST) |
| PLL inactive | PLL active | 16-32 MHz | 2 ms (approx.) |

**Note 1:** PLL inactive.

### 6.4.2 TWO-SPEED START-UP SEQUENCE

1. Wake-up from Power-on Reset or Sleep.
2. Instructions begin execution by the internal oscillator at the frequency set in the IRCF<3:0> bits of the OSCCON register.
3. OST enabled to count 1024 clock cycles.
4. OST timed out, wait for falling edge of the internal oscillator.
5. OSTS is set.
6. System clock held low until the next falling edge of new clock (LP, XT or HS mode).
7. System clock is switched to external clock source.

### 6.4.3 CHECKING TWO-SPEED CLOCK STATUS

Checking the state of the OSTS bit of the OSCSTAT register will confirm if the microcontroller is running from the external clock source, as defined by the FOSC<2:0> bits in the Configuration Words, or the internal oscillator.

**FIGURE 6-8: TWO-SPEED START-UP**
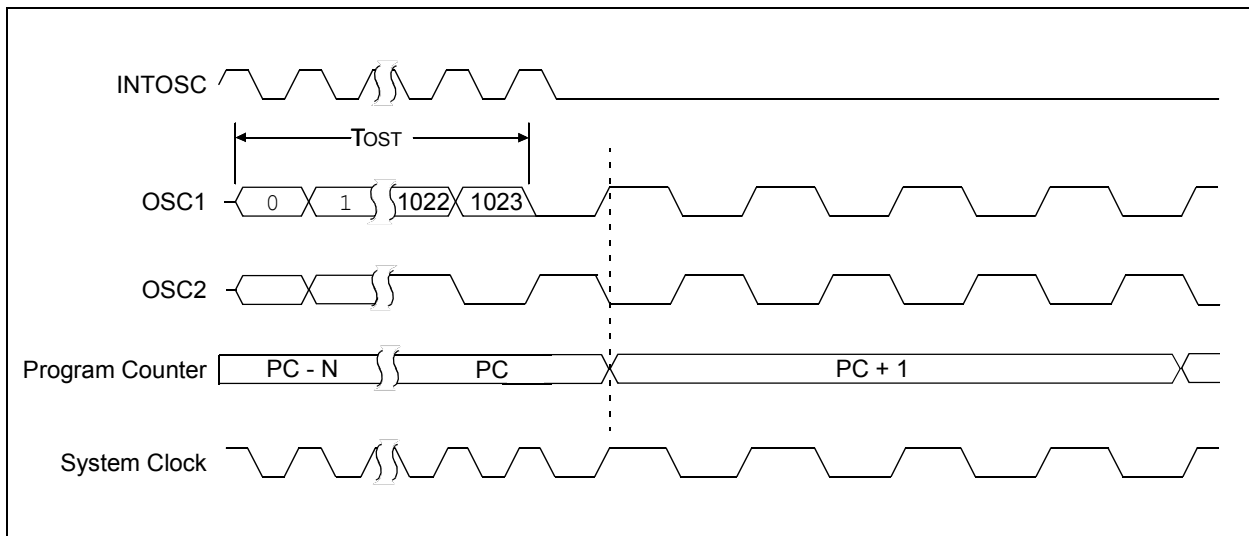
# PIC16(L)F1782/3

## 6.5 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 Configuration Words. The FSCM is applicable to all external Oscillator modes (LP, XT, HS, EC, Timer1 Oscillator and RC).

**FIGURE 6-9: FSCM BLOCK DIAGRAM**
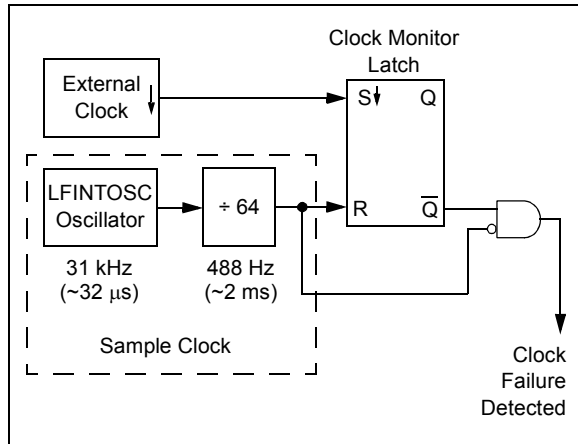


### 6.5.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64. See Figure 6-9. 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 external clock goes low.

### 6.5.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM switches the device clock to an internal clock source and sets the bit flag OSFIF of the PIR2 register. Setting this flag will generate an interrupt if the OSFIE 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.

The internal clock source chosen by the FSCM is determined by the IRCF<3:0> bits of the OSCCON register. This allows the internal oscillator to be configured before a failure occurs.

### 6.5.3 FAIL-SAFE CONDITION CLEARING

The Fail-Safe condition is cleared after a Reset, executing a SLEEP instruction or changing the SCS bits of the OSCCON register. When the SCS bits are changed, the OST is restarted. 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 after successfully switching to the external clock source. The OSFIF bit should be cleared prior to switching to the external clock source. If the Fail-Safe condition still exists, the OSFIF flage will again become set by hardware.

### 6.5.4 RESET OR WAKE-UP FROM SLEEP

The FSCM is designed to detect an oscillator failure after the Oscillator Start-up Timer (OST) has expired. The OST is used after waking up from Sleep and after any type of Reset. The OST is not used with the EC or RC Clock modes so that the FSCM will be active as soon as the Reset or wake-up has completed. When the FSCM is enabled, the Two-Speed Start-up is also enabled. Therefore, the device will always be executing code while the OST is operating.

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

**FIGURE 6-10:** **FSCM TIMING DIAGRAM**



Note: The system clock is normally at a much higher frequency than the sample clock. The relative frequencies in this example have been chosen for clarity.
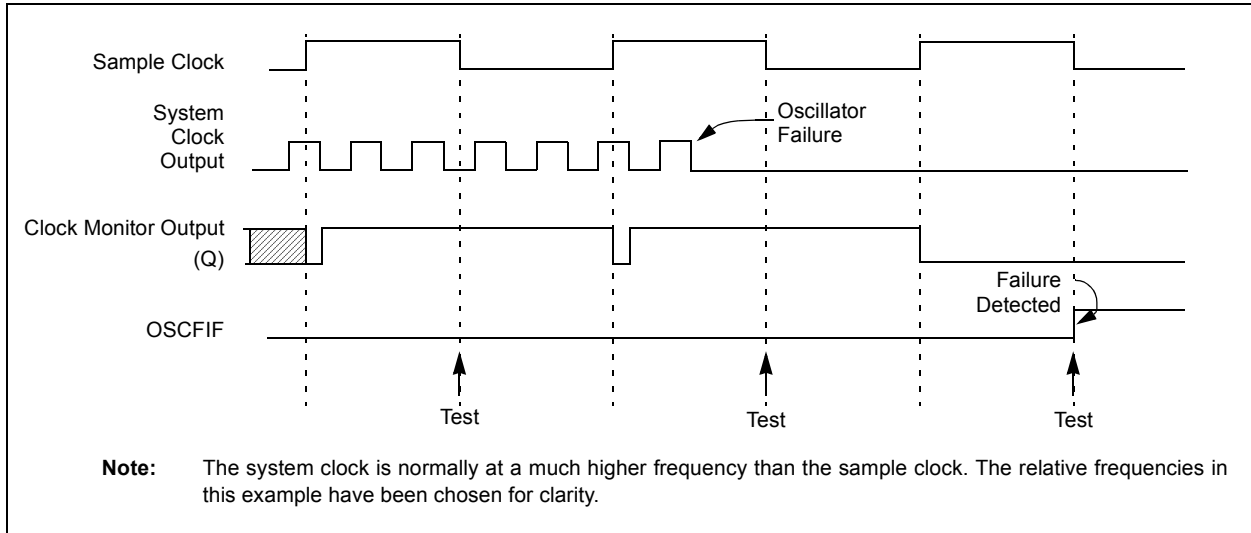
# PIC16(L)F1782/3

## 6.6    Register Definitions: Oscillator Control

### REGISTER 6-1:    OSCCON: OSCILLATOR CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | U-0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|-----|---------|---------|
| SPLLEN | | IRCF<3:0> | | | — | | SCS<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          **SPLLEN:** Software PLL Enable bit
               <u>If PLLEN in Configuration Words = 1:</u>
               SPLLEN bit is ignored. 4x PLL is always enabled (subject to oscillator requirements)

               <u>If PLLEN in Configuration Words = 0:</u>
               1 =  4x PLL Is enabled
               0 =  4x PLL is disabled

bit 6-3        **IRCF<3:0>:** Internal Oscillator Frequency Select bits
               1111 =   16 MHz HF or 32 MHz HF[2]
               1110 =   8 MHz or 32 MHz HF[2]
               1101 =   4 MHz HF
               1100 =   2 MHz HF
               1011 =   1 MHz HF
               1010 =   500 kHz HF[1]
               1001 =   250 kHz HF[1]
               1000 =   125 kHz HF[1]
               0111 =   500 kHz MF (default upon Reset)
               0110 =   250 kHz MF
               0101 =   125 kHz MF
               0100 =   62.5 kHz MF
               0011 =   31.25 kHz HF[1]
               0010 =   31.25 kHz MF
               000x =   31 kHz LF

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

bit 1-0        **SCS<1:0>:** System Clock Select bits
               1x = Internal oscillator block
               01 = Timer1 oscillator
               00 = Clock determined by FOSC<2:0> in Configuration Words.

Note  1:    Duplicate frequency derived from HFINTOSC.
      2:    32 MHz when SPLLEN bit is set. Refer to **Section 6.2.2.6 "32 MHz Internal Oscillator Frequency Selection"**.

### REGISTER 6-2: OSCSTAT: OSCILLATOR STATUS REGISTER

| R-1/q | R-0/q | R-q/q | R-0/q | R-0/q | R-q/q | R-0/0 | R-0/q |
|-------|-------|-------|-------|-------|-------|-------|-------|
| T1OSCR | PLLR | OSTS | HFIOFR | HFIOFL | MFIOFR | LFIOFR | HFIOFS |
| 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 | q = Conditional |

bit 7 **T1OSCR:** Timer1 Oscillator Ready bit

<u>If T1OSCEN = 1</u>:
1 = Timer1 oscillator is ready
0 = Timer1 oscillator is not ready
<u>If T1OSCEN = 0</u>:
1 = Timer1 clock source is always ready

bit 6 **PLLR** 4x PLL Ready bit
1 = 4x PLL is ready
0 = 4x PLL is not ready

bit 5 **OSTS:** Oscillator Start-up Time-out Status bit
1 = Running from the clock defined by the FOSC<2:0> bits of the Configuration Words
0 = Running from an internal oscillator (FOSC<2:0> = 100)

bit 4 **HFIOFR:** High-Frequency Internal Oscillator Ready bit
1 = HFINTOSC is ready
0 = HFINTOSC is not ready

bit 3 **HFIOFL:** High-Frequency Internal Oscillator Locked bit
1 = HFINTOSC is at least 2% accurate
0 = HFINTOSC is not 2% accurate

bit 2 **MFIOFR:** Medium-Frequency Internal Oscillator Ready bit
1 = MFINTOSC is ready
0 = MFINTOSC is not ready

bit 1 **LFIOFR:** Low-Frequency Internal Oscillator Ready bit
1 = LFINTOSC is ready
0 = LFINTOSC is not ready

bit 0 **HFIOFS:** High-Frequency Internal Oscillator Stable bit
1 = HFINTOSC is at least 0.5% accurate
0 = HFINTOSC is not 0.5% accurate

**REGISTER 6-3: OSCTUNE: OSCILLATOR TUNING REGISTER**

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-----|-----|---------|---------|---------|---------|---------|---------|
| — | — | TUN<5:0> | | | | | |
| 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 7-6  **Unimplemented:** Read as '0'

bit 5-0  **TUN<5:0>:** Frequency Tuning bits
100000 = Minimum frequency
•
•
•
111111 =
000000 = Oscillator module is running at the factory-calibrated frequency.
000001 =
•
•
•
011110 =
011111 = Maximum frequency

**TABLE 6-2: SUMMARY OF 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 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| OSCCON | SPLLEN | IRCF<3:0> | | | | — | SCS<1:0> | | 72 |
| OSCSTAT | T1OSCR | PLLR | OSTS | HFIOFR | HFIOFL | MFIOFR | LFIOFR | HFIOFS | 73 |
| OSCTUNE | — | — | TUN<5:0> | | | | | | 74 |
| PIR2 | OSFIF | C2IF | C1IF | EEIF | BCL1IF | — | C3IF | CCP2IF | 89 |
| PIE2 | OSFIE | C2IE | C1IE | EEIE | BCL1IE | — | C3IE | CCP2IE | 86 |
| T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | T1OSCEN | T1SYNC | — | TMR1ON | 193 |

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

**TABLE 6-3: SUMMARY OF CONFIGURATION WORD WITH CLOCK SOURCES**

| Name | Bits | Bit -/7 | Bit -/6 | Bit 13/5 | Bit 12/4 | Bit 11/3 | Bit 10/2 | Bit 9/1 | Bit 8/0 | Register on Page |
|------|------|---------|---------|----------|----------|----------|----------|---------|---------|------------------|
| CONFIG1 | 13:8 | — | — | FCMEN | IESO | CLKOUTEN | BOREN<1:0> | | CPD | 44 |
| | 7:0 | CP | MCLRE | PWRTE | WDTE<1:0> | | FOSC<2:0> | | | |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by clock sources.
**Note  1:** PIC16F1782/3 only.

# PIC16(L)F1782/3

## 7.0 REFERENCE CLOCK MODULE

The reference clock module provides the ability to send a divided clock to the clock output pin of the device (CLKR). This module is available in all oscillator configurations and allows the user to select a greater range of clock submultiples to drive external devices in the application. The reference clock module includes the following features:

• System clock is the source
• Available in all oscillator configurations
• Programmable clock divider
• Output enable to a port pin
• Selectable duty cycle
• Slew rate control

The reference clock module is controlled by the CLKRCON register (Register 7-1) and is enabled when setting the CLKREN bit. To output the divided clock signal to the CLKR port pin, the CLKROE bit must be set. The CLKRDIV<2:0> bits enable the selection of 8 different clock divider options. The CLKRDC<1:0> bits can be used to modify the duty cycle of the output clock**(1)**. The CLKRSLR bit controls slew rate limiting.

> **Note 1:** If the base clock rate is selected without a divider, the output clock will always have a duty cycle equal to that of the source clock, unless a 0% duty cycle is selected. If the clock divider is set to base clock/2, then 25% and 75% duty cycle accuracy will be dependent upon the source clock.

### 7.1 Slew Rate

The slew rate limitation on the output port pin can be disabled. The slew rate limitation is removed by clearing the CLKRSLR bit in the CLKRCON register.

### 7.2 Effects of a Reset

Upon any device Reset, the reference clock module is disabled. The user's firmware is responsible for initializing the module before enabling the output. The registers are reset to their default values.

### 7.3 Conflicts with the CLKR Pin

There are two cases when the reference clock output signal cannot be output to the CLKR pin, if:

• LP, XT or HS Oscillator mode is selected.
• CLKOUT function is enabled.

#### 7.3.1 OSCILLATOR MODES

If LP, XT or HS oscillator modes are selected, the OSC2/CLKR pin must be used as an oscillator input pin and the CLKR output cannot be enabled. See **Section 6.2 "Clock Source Types"** for more information on different oscillator modes.

#### 7.3.2 CLKOUT FUNCTION

The CLKOUT function has a higher priority than the reference clock module. Therefore, if the CLKOUT function is enabled by the CLKOUTEN bit in Configuration Words, FOSC/4 will always be output on the port pin. Reference **Section 4.0 "Device Configuration"** for more information.

### 7.4 Operation During Sleep

As the reference clock module relies on the system clock as its source, and the system clock is disabled in Sleep, the module does not function in Sleep, even if an external clock source or the Timer1 clock source is configured as the system clock. The module outputs will remain in their current state until the device exits Sleep.

# PIC16(L)F1782/3

## 7.5 Register Definition: Reference Clock Control

**REGISTER 7-1:** **CLKRCON: REFERENCE CLOCK CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-1/1 | R/W-1/1 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| CLKREN | CLKROE | CLKRSLR | CLKRDC<1:0> | | CLKRDIV<2:0> | | |
| 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　**CLKREN:** Reference Clock Module Enable bit

　　1 = Reference clock module is enabled
　　0 = Reference clock module is disabled

bit 6　**CLKROE:** Reference Clock Output Enable bit

　　1 = Reference clock output is enabled on CLKR pin
　　0 = Reference clock output disabled on CLKR pin

bit 5　**CLKRSLR:** Reference Clock Slew Rate Control Limiting Enable bit

　　1 = Slew rate limiting is enabled
　　0 = Slew rate limiting is disabled

bit 4-3　**CLKRDC<1:0>:** Reference Clock Duty Cycle bits

　　11 = Clock outputs duty cycle of 75%
　　10 = Clock outputs duty cycle of 50%
　　01 = Clock outputs duty cycle of 25%
　　00 = Clock outputs duty cycle of 0%

bit 2-0　**CLKRDIV<2:0>** Reference Clock Divider bits

　　111 = Base clock value divided by 128
　　110 = Base clock value divided by 64
　　101 = Base clock value divided by 32
　　100 = Base clock value divided by 16
　　011 = Base clock value divided by 8
　　010 = Base clock value divided by 4
　　001 = Base clock value divided by 2[1]
　　000 = Base clock value[2]

**Note 1:** In this mode, the 25% and 75% duty cycle accuracy will be dependent on the source clock duty cycle.

**2:** In this mode, the duty cycle will always be equal to the source clock duty cycle, unless a duty cycle of 0% is selected.

**Preliminary**

**TABLE 7-1:** **SUMMARY OF REGISTERS ASSOCIATED WITH REFERENCE CLOCK SOURCES**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| CLKRCON | CLKREN | CLKROE | CLKRSLR | CLKRDC<1:0> | | CLKRDIV<2:0> | | | 76 |

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used by reference clock sources.

**TABLE 7-2:** **SUMMARY OF CONFIGURATION WORD WITH REFERENCE CLOCK SOURCES**

| Name | Bits | Bit -/7 | Bit -/6 | Bit 13/5 | Bit 12/4 | Bit 11/3 | Bit 10/2 | Bit 9/1 | Bit 8/0 | Register on Page |
|------|------|---------|---------|----------|----------|----------|----------|---------|---------|------------------|
| CONFIG1 | 13:8 | — | — | FCMEN | IESO | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | $\overline{\text{CPD}}$ | 44 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE1<:0> | | FOSC<2:0> | | | |

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used by reference clock sources.

**NOTES:**

.

**Preliminary**

© 2011-2012 Microchip Technology Inc.

# 8.0 INTERRUPTS

The interrupt feature allows certain events to preempt normal program flow. Firmware is used to determine the source of the interrupt and act accordingly. Some interrupts can be configured to wake the MCU from Sleep mode.

This chapter contains the following information for Interrupts:

• Operation
• Interrupt Latency
• Interrupts During Sleep
• INT Pin
• Automatic Context Saving

Many peripherals produce interrupts. Refer to the corresponding chapters for details.

A block diagram of the interrupt logic is shown in Figure 8-1.

**FIGURE 8-1:        INTERRUPT LOGIC**
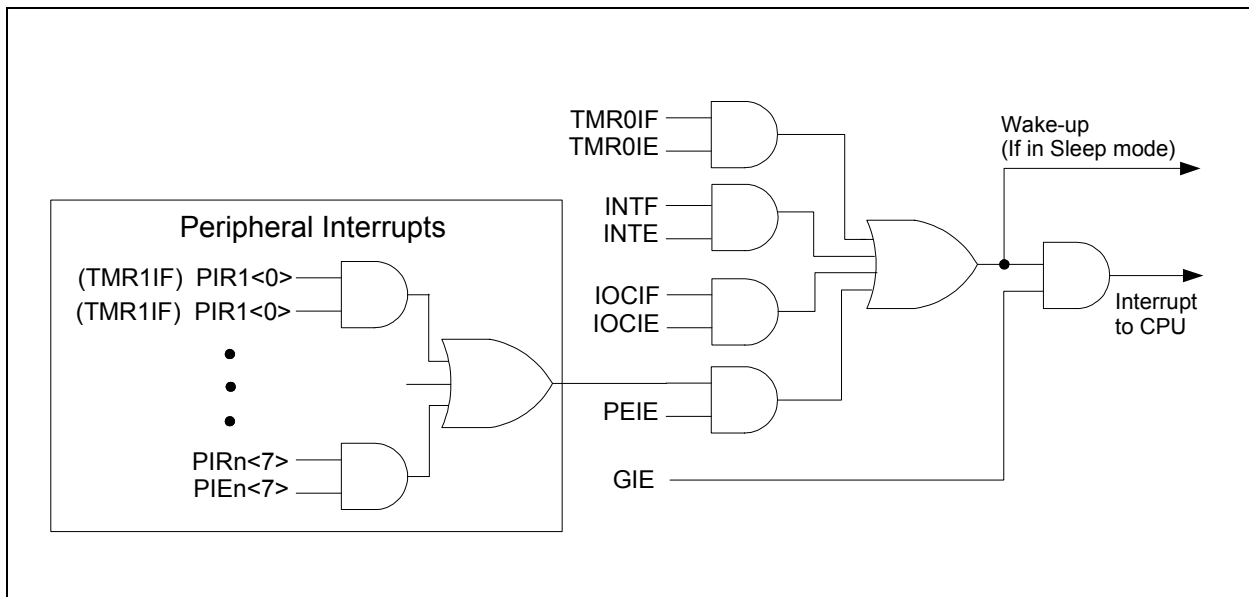
## 8.1 Operation

Interrupts are disabled upon any device Reset. They are enabled by setting the following bits:

- GIE bit of the INTCON register
- Interrupt Enable bit(s) for the specific interrupt event(s)
- PEIE bit of the INTCON register (if the Interrupt Enable bit of the interrupt event is contained in the PIE1 or PIE2 registers)

The INTCON, PIR1 and PIR2 registers record individual interrupts via interrupt flag bits. Interrupt flag bits will be set, regardless of the status of the GIE, PEIE and individual interrupt enable bits.

The following events happen when an interrupt event occurs while the GIE bit is set:

- Current prefetched instruction is flushed
- GIE bit is cleared
- Current Program Counter (PC) is pushed onto the stack
- Critical registers are automatically saved to the shadow registers (See **Section 8.5 "Automatic Context Saving".**)
- PC is loaded with the interrupt vector 0004h

The firmware within the Interrupt Service Routine (ISR) should determine the source of the interrupt by polling the interrupt flag bits. The interrupt flag bits must be cleared before exiting the ISR to avoid repeated interrupts. Because the GIE bit is cleared, any interrupt that occurs while executing the ISR will be recorded through its interrupt flag, but will not cause the processor to redirect to the interrupt vector.

The RETFIE instruction exits the ISR by popping the previous address from the stack, restoring the saved context from the shadow registers and setting the GIE bit.

For additional information on a specific interrupt's operation, refer to its peripheral chapter.

| | |
|---|---|
| **Note 1:** | Individual interrupt flag bits are set, regardless of the state of any other enable bits. |
| **2:** | All interrupts will be ignored while the GIE bit is cleared. Any interrupt occurring while the GIE bit is clear will be serviced when the GIE bit is set again. |

## 8.2 Interrupt Latency

Interrupt latency is defined as the time from when the interrupt event occurs to the time code execution at the interrupt vector begins. The latency for synchronous interrupts is 3 or 4 instruction cycles. For asynchronous interrupts, the latency is 3 to 5 instruction cycles, depending on when the interrupt occurs. See Figure 8-2 and Figure 8.3 for more details.

**FIGURE 8-2:** **INTERRUPT LATENCY**
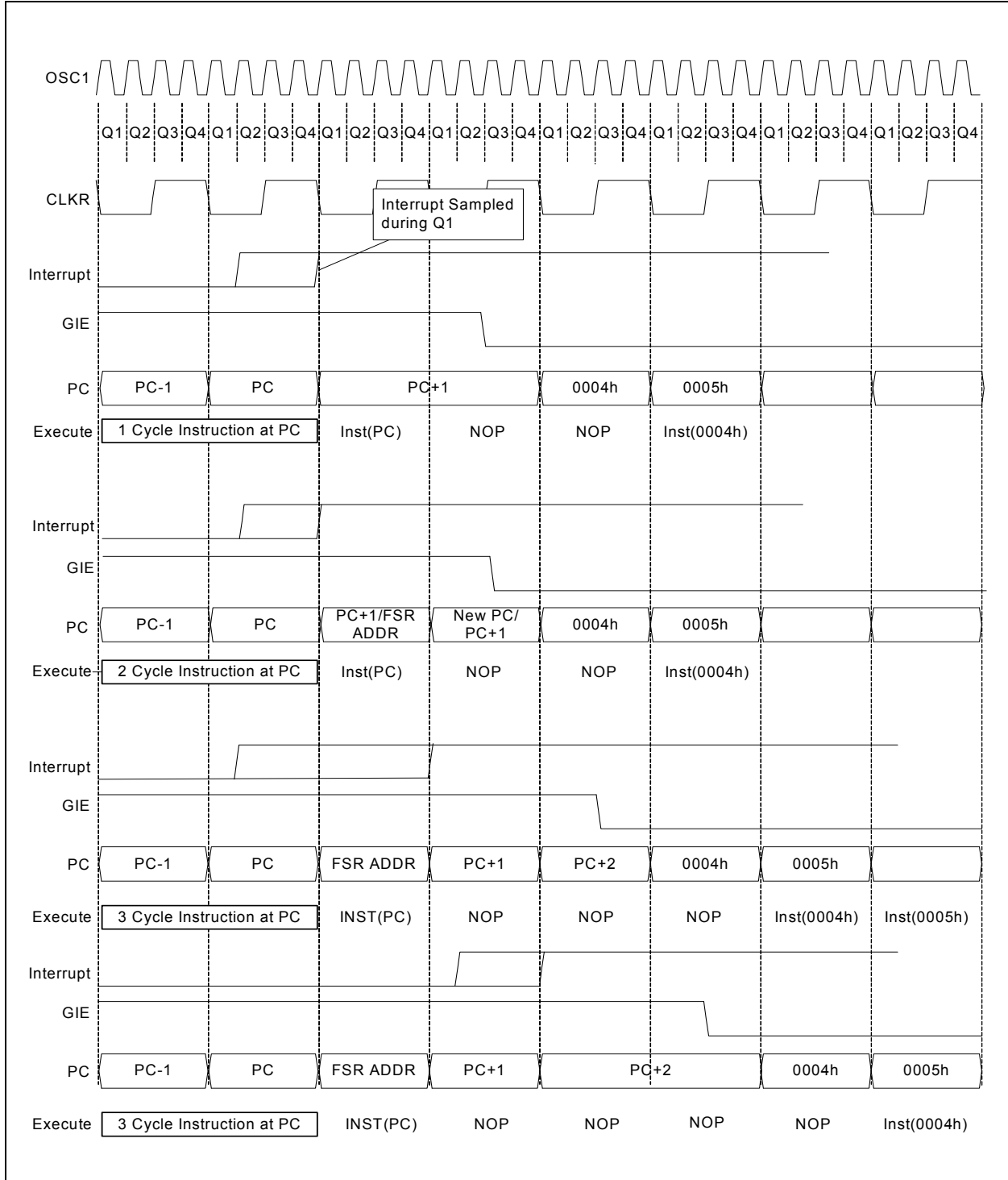
# PIC16(L)F1782/3

**FIGURE 8-3:** INT PIN INTERRUPT TIMING



**Note 1:** INTF flag is sampled here (every Q1).

   **2:** Asynchronous interrupt latency = 3-5 TCY. Synchronous latency = 3-4 TCY, where TCY = instruction cycle time. Latency is the same whether Inst (PC) is a single cycle or a 2-cycle instruction.

   **3:** CLKOUT not available in all oscillator modes.

   **4:** For minimum width of INT pulse, refer to AC specifications in **Section 30.0 "Electrical Specifications"**.

   **5:** INTF is enabled to be set any time during the Q4-Q1 cycles.

**Preliminary**

## 8.3 Interrupts During Sleep

Some interrupts can be used to wake from Sleep. To wake from Sleep, the peripheral must be able to operate without the system clock. The interrupt source must have the appropriate Interrupt Enable bit(s) set prior to entering Sleep.

On waking from Sleep, if the GIE bit is also set, the processor will branch to the interrupt vector. Otherwise, the processor will continue executing instructions after the SLEEP instruction. The instruction directly after the SLEEP instruction will always be executed before branching to the ISR. Refer to **Section 9.0 "Power-Down Mode (Sleep)"** for more details.

## 8.4 INT Pin

The INT pin can be used to generate an asynchronous edge-triggered interrupt. This interrupt is enabled by setting the INTE bit of the INTCON register. The INTEDG bit of the OPTION_REG register determines on which edge the interrupt will occur. When the INTEDG bit is set, the rising edge will cause the interrupt. When the INTEDG bit is clear, the falling edge will cause the interrupt. The INTF bit of the INTCON register will be set when a valid edge appears on the INT pin. If the GIE and INTE bits are also set, the processor will redirect program execution to the interrupt vector.

## 8.5 Automatic Context Saving

Upon entering an interrupt, the return PC address is saved on the stack. Additionally, the following registers are automatically saved in the Shadow registers:

- W register
- STATUS register (except for $\overline{TO}$ and $\overline{PD}$)
- BSR register
- FSR registers
- PCLATH register

Upon exiting the Interrupt Service Routine, these registers are automatically restored. Any modifications to these registers during the ISR will be lost. If modifications to any of these registers are desired, the corresponding Shadow register should be modified and the value will be restored when exiting the ISR. The Shadow registers are available in Bank 31 and are readable and writable. Depending on the user's application, other registers may also need to be saved.

# PIC16(L)F1782/3

## 8.6 Register Definitions: Interrupt Control

**REGISTER 8-1: INTCON: INTERRUPT CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-0/0 |
|---------|---------|---------|---------|---------|---------|---------|-------|
| GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF[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 | |

bit 7      **GIE:** Global Interrupt Enable bit
         1 = Enables all active interrupts
         0 = Disables all interrupts

bit 6      **PEIE:** Peripheral Interrupt Enable bit
         1 = Enables all active peripheral interrupts
         0 = Disables all peripheral interrupts

bit 5      **TMR0IE:** Timer0 Overflow Interrupt Enable bit
         1 = Enables the Timer0 interrupt
         0 = Disables the Timer0 interrupt

bit 4      **INTE:** INT External Interrupt Enable bit
         1 = Enables the INT external interrupt
         0 = Disables the INT external interrupt

bit 3      **IOCIE:** Interrupt-on-Change Enable bit
         1 = Enables the interrupt-on-change
         0 = Disables the interrupt-on-change

bit 2      **TMR0IF:** Timer0 Overflow Interrupt Flag bit
         1 = TMR0 register has overflowed
         0 = TMR0 register did not overflow

bit 1      **INTF:** INT External Interrupt Flag bit
         1 = The INT external interrupt occurred
         0 = The INT external interrupt did not occur

bit 0      **IOCIF:** Interrupt-on-Change Interrupt Flag bit[1]
         1 = When at least one of the interrupt-on-change pins changed state
         0 = None of the interrupt-on-change pins have changed state

**Note 1:** The IOCIF Flag bit is read-only and cleared when all the Interrupt-on-Change flags in the IOCBF register have been cleared by software.

> **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, GIE, of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

**REGISTER 8-2:** **PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TMR1GIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE |
| 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          **TMR1GIE:** Timer1 Gate Interrupt Enable bit

1 = Enables the Timer1 gate acquisition interrupt
0 = Disables the Timer1 gate acquisition interrupt

bit 6          **ADIE:** A/D Converter (ADC) Interrupt Enable bit

1 = Enables the ADC interrupt
0 = Disables the ADC interrupt

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

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

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

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

bit 3          **SSPIE:** Synchronous Serial Port (MSSP) 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 Timer2 to PR2 match interrupt
0 = Disables the Timer2 to PR2 match interrupt

bit 0          **TMR1IE:** Timer1 Overflow Interrupt Enable bit

1 = Enables the Timer1 overflow interrupt
0 = Disables the Timer1 overflow interrupt

| **Note:** | Bit PEIE of the INTCON register must be set to enable any peripheral interrupt. |
|-----------|---------------------------------------------------------------------------------|

# PIC16(L)F1782/3

**REGISTER 8-3:** **PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|-----|---------|---------|
| OSFIE | C2IE | C1IE | EEIE | BCLIE | — | C3IE | CCP2IE |
| 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        **OSFIE:** Oscillator Fail Interrupt Enable bit
             1 = Enables the Oscillator Fail interrupt
             0 = Disables the Oscillator Fail interrupt

bit 6        **C2IE:** Comparator C2 Interrupt Enable bit
             1 = Enables the Comparator C2 interrupt
             0 = Disables the Comparator C2 interrupt

bit 5        **C1IE:** Comparator C1 Interrupt Enable bit
             1 = Enables the Comparator C1 interrupt
             0 = Disables the Comparator C1 interrupt

bit 4        **EEIE:** EEPROM Write Completion Interrupt Enable bit
             1 = Enables the EEPROM Write Completion interrupt
             0 = Disables the EEPROM Write Completion interrupt

bit 3        **BCLIE:** MSSP Bus Collision Interrupt Enable bit
             1 = Enables the MSSP Bus Collision Interrupt
             0 = Disables the MSSP Bus Collision Interrupt

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

bit 1        **C3IE:** Comparator C3 Interrupt Enable bit
             1 = Enables the Comparator C3 Interrupt
             0 = Disables the Comparator C3 Interrupt

bit 0        **CCP2IE:** CCP2 Interrupt Enable bit
             1 = Enables the CCP2 interrupt
             0 = Disables the CCP2 interrupt

| **Note:** | Bit PEIE of the INTCON register must be set to enable any peripheral interrupt. |
|-----------|-------------------------------------------------------------------------------|

## REGISTER 8-4: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|------|------|----------|----------|------|------|----------|----------|
| — | — | PSMC2TIE | PSMC1TIE | — | — | PSMC2SIE | PSMC1SIE |
| 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      **Unimplemented:** Read as '0'

bit 5      **PSMC2TIE:** PSMC2 Time Base Interrupt Enable bit

       1 = Enables PSMC2 time base generated interrupts
       0 = Disables PSMC2 time base generated interrupts

bit 4      **PSMC1TIE:** PSMC1 Time Base Interrupt Enable bit

       1 = Enables PSMC1 time base generated interrupts
       0 = Disables PSMC1 time base generated interrupts

bit 3-2      **Unimplemented:** Read as '0'

bit 1      **PSMC2SIE:** PSMC2 Auto-Shutdown Interrupt Enable bit

       1 = Enables PSMC2 auto-shutdown interrupts
       0 = Disables PSMC2 auto-shutdown interrupts

bit 0      **PSMC1SIE:** PSMC1 Auto-Shutdown Interrupt Enable bit

       1 = Enables PSMC1 auto-shutdown interrupts
       0 = Disables PSMC1 auto-shutdown interrupts

> **Note:** Bit PEIE of the INTCON register must be set to enable any peripheral interrupt.

# PIC16(L)F1782/3

**REGISTER 8-5:    PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1**

| R/W-0/0 | R/W-0/0 | R-0/0 | R-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|-------|-------|---------|---------|---------|---------|
| TMR1GIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF |
| 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    **TMR1GIF:** Timer1 Gate Interrupt Flag bit

1 = Interrupt is pending
0 = Interrupt is not pending

bit 6    **ADIF:** A/D Converter Interrupt Flag bit

1 = Interrupt is pending
0 = Interrupt is not pending

bit 5    **RCIF:** USART Receive Interrupt Flag bit

1 = Interrupt is pending
0 = Interrupt is not pending

bit 4    **TXIF:** USART Transmit Interrupt Flag bit

1 = Interrupt is pending
0 = Interrupt is not pending

bit 3    **SSPIF:** Synchronous Serial Port (MSSP) Interrupt Flag bit

1 = Interrupt is pending
0 = Interrupt is not pending

bit 2    **CCP1IF:** CCP1 Interrupt Flag bit

1 = Interrupt is pending
0 = Interrupt is not pending

bit 1    **TMR2IF:** Timer2 to PR2 Interrupt Flag bit

1 = Interrupt is pending
0 = Interrupt is not pending

bit 0    **TMR1IF:** Timer1 Overflow Interrupt Flag bit

1 = Interrupt is pending
0 = Interrupt is not pending

| **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, GIE, of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. |
|---|---|

**Preliminary**    © 2011-2012 Microchip Technology Inc.

# PIC16(L)F1782/3

**REGISTER 8-6:** **PIR2: PERIPHERAL INTERRUPT REQUEST REGISTER 2**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|-----|---------|---------|
| OSFIF | C2IF | C1IF | EEIF | BCLIF | — | C3IF | CCP2IF |
| 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     **OSFIF:** Oscillator Fail Interrupt Flag bit
- 1 = Interrupt is pending
- 0 = Interrupt is not pending

bit 6     **C2IF:** Comparator C2 Interrupt Flag bit
- 1 = Interrupt is pending
- 0 = Interrupt is not pending

bit 5     **C1IF:** Comparator C1 Interrupt Flag bit
- 1 = Interrupt is pending
- 0 = Interrupt is not pending

bit 4     **EEIF:** EEPROM Write Completion Interrupt Flag bit
- 1 = Interrupt is pending
- 0 = Interrupt is not pending

bit 3     **BCLIF:** MSSP Bus Collision Interrupt Flag bit
- 1 = Interrupt is pending
- 0 = Interrupt is not pending

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

bit 1     **C3IF:** Comparator C3 Interrupt Flag bit
- 1 = Interrupt is pending
- 0 = Interrupt is not pending

bit 0     **CCP2IF:** CCP2 Interrupt Flag bit
- 1 = Interrupt is pending
- 0 = Interrupt is not pending

| 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, GIE, of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. |
|---|---|

# PIC16(L)F1782/3

**REGISTER 8-7:** **PIR4: PERIPHERAL INTERRUPT REQUEST REGISTER42**

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | PSMC2TIF | PSMC1TIF | — | — | PSMC2SIF | PSMC1SIF |
| 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      **Unimplemented:** Read as '0'

bit 5      **PSMC2TIF:** PSMC2 Time Base Interrupt Flag bit

         1 = Interrupt is pending
         0 = Interrupt is not pending

bit 4      **PSMC1TIF:** PSMC1 Time Base Interrupt Flag bit

         1 = Interrupt is pending
         0 = Interrupt is not pending

bit 3-2      **Unimplemented:** Read as '0'

bit 1      **PSMC2SIF:** PSMC2 Auto-shutdown Flag bit

         1 = Interrupt is pending
         0 = Interrupt is not pending

bit 0      **PSMC1SIF:** PSMC1 Auto-shutdown Flag bit

         1 = Interrupt is pending
         0 = Interrupt is not pending

| 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, GIE, of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. |
|---|---|

**TABLE 8-1:** **SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| OPTION_REG | $\overline{\text{WPUEN}}$ | INTEDG | TMR0CS | TMR0SE | PSA | PS<2:0> | | | 183 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IFE | TMR1IE | 85 |
| PIE2 | OSFIE | C2IE | C1IE | EEIE | BCL1IE | — | C3IE | CCP2IE | 86 |
| PIE4 | — | — | PSMC2TIE | PSMC1TIE | — | — | PSMC2SIE | PSMC2SIE | 87 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| PIR2 | OSFIF | C2IF | C1IF | EEIF | BCL1IF | — | C3IF | CCP2IF | 89 |
| PIR4 | — | — | PSMC2TIF | PSMC1TIF | — | — | PSMC2SIF | PSMC1SIF | 90 |

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

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 9.0 POWER-DOWN MODE (SLEEP)

The Power-down mode is entered by executing a `SLEEP` instruction.

Upon entering Sleep mode, the following conditions exist:

1. WDT will be cleared but keeps running, if enabled for operation during Sleep.
2. $\overline{PD}$ bit of the STATUS register is cleared.
3. $\overline{TO}$ bit of the STATUS register is set.
4. CPU clock is disabled.
5. 31 kHz LFINTOSC is unaffected and peripherals that operate from it may continue operation in Sleep.
6. ADC is unaffected, if the dedicated FRC clock is selected.
7. I/O ports maintain the status they had before `SLEEP` was executed (driving high, low or high-impedance).
8. Resets other than WDT are not affected by Sleep mode.

Refer to individual chapters for more details on peripheral operation during Sleep.

To minimize current consumption, the following conditions should be considered:

- I/O pins should not be floating
- External circuitry sinking current from I/O pins
- Internal circuitry sourcing current from I/O pins
- Current draw from pins with internal weak pull-ups
- Modules using 31 kHz LFINTOSC
- Modules using Timer1 oscillator

I/O pins that are high-impedance inputs should be pulled to V$_{DD}$ or V$_{SS}$ externally to avoid switching currents caused by floating inputs.

Examples of internal circuitry that might be sourcing current include modules such as the DAC and FVR modules. See **Section 19.0 "Digital-to-Analog Converter (DAC) Module"** and **Section 15.0 "Fixed Voltage Reference (FVR)"** for more information on these modules.

## 9.1 Wake-up from Sleep

The device can wake-up from Sleep through one of the following events:

1. External Reset input on $\overline{MCLR}$ pin, if enabled
2. BOR Reset, if enabled
3. POR Reset
4. Watchdog Timer, if enabled
5. Any external interrupt
6. Interrupts by peripherals capable of running during Sleep (see individual peripheral for more information)

The first three events will cause a device Reset. The last three events are considered a continuation of program execution. To determine whether a device Reset or wake-up event occurred, refer to **Section 5.12 "Determining the Cause of a Reset"**.

When the `SLEEP` instruction is being executed, the next instruction (PC + 1) is prefetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be enabled. Wake-up will occur regardless of the state of the GIE bit. If the GIE bit is disabled, the device continues execution at the instruction after the `SLEEP` instruction. If the GIE bit is enabled, the device executes the instruction after the `SLEEP` instruction, the device will then call the Interrupt Service Routine. In cases where the execution of the instruction following `SLEEP` is not desirable, the user should have a `NOP` after the `SLEEP` instruction.

The WDT is cleared when the device wakes up from Sleep, regardless of the source of wake-up.

### 9.1.1 WAKE-UP USING INTERRUPTS

When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs **before** the execution of a `SLEEP` instruction
  - `SLEEP` instruction will execute as a `NOP`.
  - WDT and WDT prescaler will not be cleared
  - $\overline{TO}$ bit of the STATUS register will not be set
  - $\overline{PD}$ bit of the STATUS register will not be cleared.

- If the interrupt occurs **during or after** the execution of a `SLEEP` instruction
  - `SLEEP` instruction will be completely executed
  - Device will immediately wake-up from Sleep
  - WDT and WDT prescaler will be cleared
  - $\overline{TO}$ bit of the STATUS register will be set
  - $\overline{PD}$ bit of the STATUS register will be cleared.

Even if the flag bits were checked before executing a `SLEEP` instruction, it may be possible for flag bits to become set before the `SLEEP` instruction completes. To determine whether a `SLEEP` instruction executed, test the $\overline{PD}$ bit. If the $\overline{PD}$ bit is set, the `SLEEP` instruction was executed as a `NOP`.

**FIGURE 9-1:      WAKE-UP FROM SLEEP THROUGH INTERRUPT**



**Note    1:**   External clock. High, Medium, Low mode assumed.
**        2:**   CLKOUT is shown here for timing reference.
**        3:**   T1OSC; See **Section 25.0 "Electrical Specifications"**.
**        4:**   GIE = `1` assumed. In this case after wake-up, the processor calls the ISR at 0004h. If GIE = `0`, execution will continue in-line.
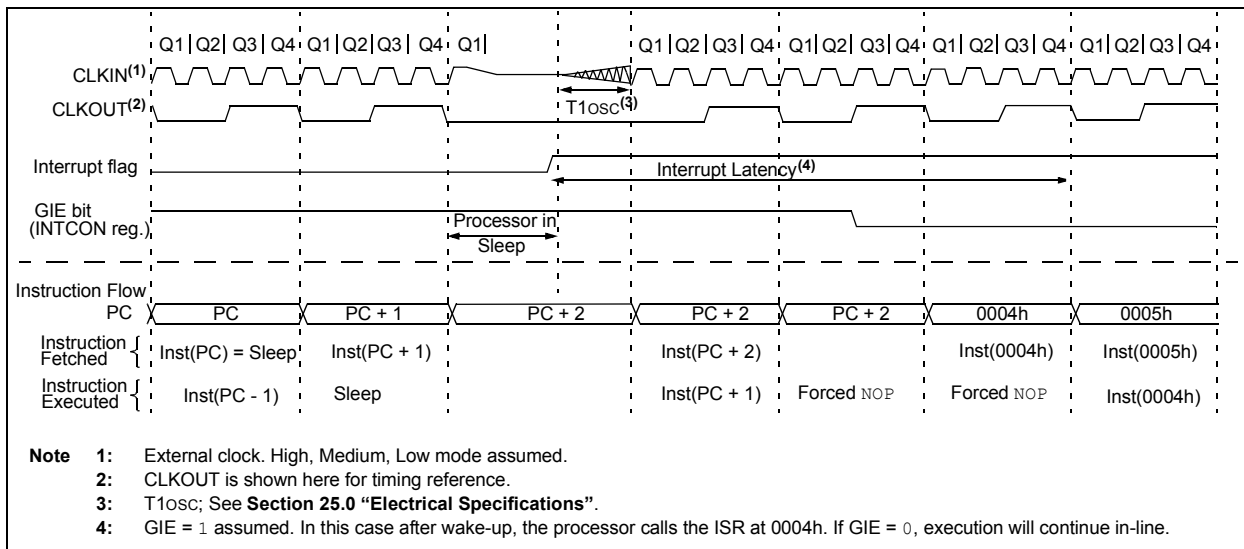
## 9.2 Low-Power Sleep Mode

The PIC16(L)F1783 device contains an internal Low Dropout (LDO) voltage regulator, which allows the device I/O pins to operate at voltages up to 5.5V while the internal device logic operates at a lower voltage. The LDO and its associated reference circuitry must remain active when the device is in Sleep mode. The PIC16(L)F1783 allows the user to optimize the operating current in Sleep, depending on the application requirements.

A Low-Power Sleep mode can be selected by setting the VREGPM bit of the VREGCON register. With this bit set, the LDO and reference circuitry are placed in a low-power state when the device is in Sleep.

### 9.2.1 SLEEP CURRENT VS. WAKE-UP TIME

In the default operating mode, the LDO and reference circuitry remain in the normal configuration while in Sleep. The device is able to exit Sleep mode quickly since all circuits remain active. In Low-Power Sleep mode, when waking up from Sleep, an extra delay time is required for these circuits to return to the normal configuration and stabilize.

The Low-Power Sleep mode is beneficial for applications that stay in Sleep mode for long periods of time. The normal mode is beneficial for applications that need to wake from Sleep quickly and frequently.

### 9.2.2 PERIPHERAL USAGE IN SLEEP

Some peripherals that can operate in Sleep mode will not operate properly with the Low-Power Sleep mode selected. The LDO will remain in the normal power mode when those peripherals are enabled. The Low-Power Sleep mode is intended for use with these peripherals:

- Brown-Out Reset (BOR)
- Watchdog Timer (WDT)
- External interrupt pin/Interrupt-on-change pins
- Timer1 (with external clock source)

| | |
|---|---|
| **Note:** | The PIC16LF1782 does not have a configurable Low-Power Sleep mode. PIC16LF1782 is an unregulated device and is always in the lowest power state when in Sleep, with no wake-up time penalty. This device has a lower maximum $V_{DD}$ and I/O voltage than the PIC16(L)F1783. See **Section 29.0 "Electrical Specifications"** for more information. |

## 9.3 Register Definitions: Voltage Regulator Control

**REGISTER 9-1: VREGCON: VOLTAGE REGULATOR CONTROL REGISTER[1]**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-1/1 |
|-----|-----|-----|-----|-----|-----|---------|---------|
| — | — | — | — | — | — | VREGPM | Reserved |
| 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 7-2      **Unimplemented:** Read as '0'

bit 1      **VREGPM:** Voltage Regulator Power Mode Selection bit

         1 = Low-Power Sleep mode enabled in Sleep[2]
             Draws lowest current in Sleep, slower wake-up
         0 = Normal Power mode enabled in Sleep[2]
             Draws higher current in Sleep, faster wake-up

bit 0      **Reserved:** Read as '1'. Maintain this bit set.

**Note 1:** PIC16F1782/3 only.

     **2:** See **Section 30.0 "Electrical Specifications"**.

**TABLE 9-1: SUMMARY OF REGISTERS ASSOCIATED WITH POWER-DOWN MODE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | RAIF | 84 |
| IOCBF | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 | 142 |
| IOCBN | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 | 141 |
| IOCBP | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 | 141 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IFE | TMR1IE | 85 |
| PIE2 | OSFIE | C2IE | C1IE | EEIE | BCL1IE | — | C3IE | CCP2IE | 86 |
| PIE4 | — | — | PSMC2TIE | PSMC1TIE | — | — | PSMC2SIE | PSMC2SIE | 87 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| PIR2 | OSFIF | C2IF | C1IF | EEIF | BCL1IF | — | C3IF | CCP2IF | 89 |
| PIR4 | — | — | PSMC2TIF | PSMC1TIF | — | — | PSMC2SIF | PSMC1SIF | 90 |
| STATUS | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | 23 |
| VREGCON | — | — | — | — | — | — | VREGPM | Reserved | 96 |
| WDTCON | — | — | WDTPS<4:0> | | | | | SWDTEN | 101 |

**Legend:**    — = unimplemented location, read as '0'. Shaded cells are not used in Power-Down mode.

## 10.0 LOW DROPOUT (LDO) VOLTAGE REGULATOR

The PIC16F1782/3 has an internal Low Dropout Regulator (LDO) which provides operation above 3.6V. The LDO regulates a voltage for the internal device logic while permitting the VDD and I/O pins to operate at a higher voltage. There is no user enable/disable control available for the LDO, it is always active. The PIC16LF1782/3 operates at a maximum VDD of 3.6V and does not incorporate an LDO.

A device I/O pin may be configured as the LDO voltage output, identified as the VCAP pin. Although not required, an external low-ESR capacitor may be connected to the VCAP pin for additional regulator stability.

The $\overline{\text{VCAPEN}}$ bit of Configuration Words determines if which pin is assigned as the VCAP pin. Refer to Table 10-1.

On power-up, the external capacitor will load the LDO voltage regulator. To prevent erroneous operation, the device is held in Reset while a constant current source charges the external capacitor. After the cap is fully charged, the device is released from Reset. For more information on the constant current rate, refer to the LDO Regulator Characteristics Table in **Section 30.0 "Electrical Specifications"**.

### TABLE 10-1: $\overline{\text{VCAPEN}}$ SELECT BIT

| $\overline{\text{VCAPEN}}$ | Pin |
|---|---|
| 0 | RA6 |
| 1 | No VCAP |

### TABLE 10-2: SUMMARY OF CONFIGURATION WORD WITH LDO

| Name | Bits | Bit -/7 | Bit -/6 | Bit 13/5 | Bit 12/4 | Bit 11/3 | Bit 10/2 | Bit 9/1 | Bit 8/0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|---|
| CONFIG2 | 13:8 | — | — | LVP | $\overline{\text{DEBUG}}$ | $\overline{\text{LPBOR}}$ | BORV | STVREN | PLLEN | 46 |
| | 7:0 | — | — | $\overline{\text{VCAPEN}}$[1] | — | — | — | WRT<1:0> | | |

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used by LDO.
**Note 1:** Not implemented on PIC16LF1782/3.

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 11.0   WATCHDOG TIMER (WDT)

The Watchdog Timer is a system timer that generates a Reset if the firmware does not issue a `CLRWDT` instruction within the time-out period. The Watchdog Timer is typically used to recover the system from unexpected events.

The WDT has the following features:

- Independent clock source
- Multiple operating modes
  - WDT is always on
  - WDT is off when in Sleep
  - WDT is controlled by software
  - WDT is always off
- Configurable time-out period is from 1 ms to 256 seconds (nominal)
- Multiple Reset conditions
- Operation during Sleep

**FIGURE 11-1:       WATCHDOG TIMER BLOCK DIAGRAM**
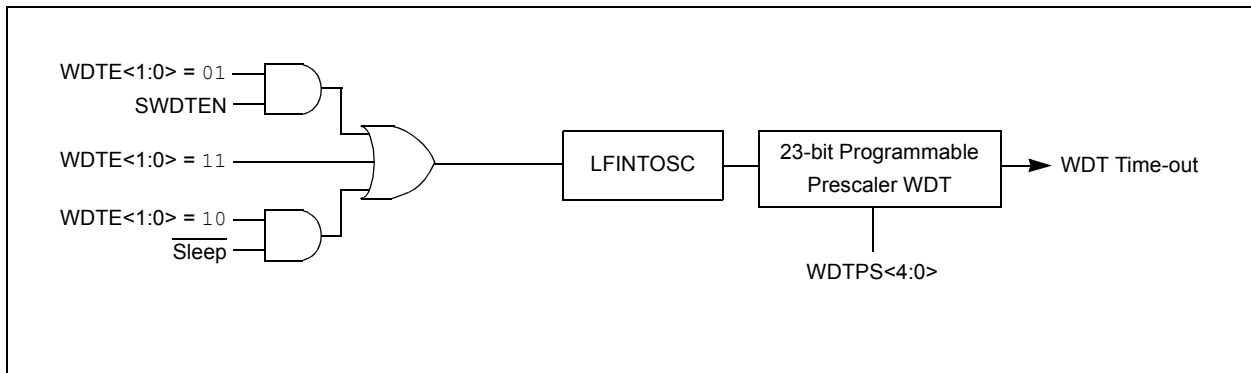
## 11.1 Independent Clock Source

The WDT derives its time base from the 31 kHz LFINTOSC internal oscillator. Time intervals in this chapter are based on a nominal interval of 1ms. See **Section 30.0 "Electrical Specifications"** for the LFINTOSC tolerances.

## 11.2 WDT Operating Modes

The Watchdog Timer module has four operating modes controlled by the WDTE<1:0> bits in Configuration Words. See Table 11-1.

### 11.2.1 WDT IS ALWAYS ON

When the WDTE bits of Configuration Words are set to '11', the WDT is always on.

WDT protection is active during Sleep.

### 11.2.2 WDT IS OFF IN SLEEP

When the WDTE bits of Configuration Words are set to '10', the WDT is on, except in Sleep.

WDT protection is not active during Sleep.

### 11.2.3 WDT CONTROLLED BY SOFTWARE

When the WDTE bits of Configuration Words are set to '01', the WDT is controlled by the SWDTEN bit of the WDTCON register.

WDT protection is unchanged by Sleep. See Table 11-1 for more details.

## 11.3 Time-Out Period

The WDTPS bits of the WDTCON register set the time-out period from 1 ms to 256 seconds (nominal). After a Reset, the default time-out period is 2 seconds.

## 11.4 Clearing the WDT

The WDT is cleared when any of the following conditions occur:

- Any Reset
- `CLRWDT` instruction is executed
- Device enters Sleep
- Device wakes up from Sleep
- Oscillator fail
- WDT is disabled
- Oscillator Start-up TImer (OST) is running

See Table 11-2 for more information.

## 11.5 Operation During Sleep

When the device enters Sleep, the WDT is cleared. If the WDT is enabled during Sleep, the WDT resumes counting.

When the device exits Sleep, the WDT is cleared again. The WDT remains clear until the OST, if enabled, completes. See **Section 6.0 "Oscillator Module (with Fail-Safe Clock Monitor)"** for more information on the OST.

When a WDT time-out occurs while the device is in Sleep, no Reset is generated. Instead, the device wakes up and resumes operation. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register are changed to indicate the event. See **Section 3.0 "Memory Organization"** and Status Register (Register 3-1) for more information.

### TABLE 11-1: WDT OPERATING MODES

| WDTE<1:0> | SWDTEN | Device Mode | WDT Mode |
|---|---|---|---|
| 11 | X | X | Active |
| 10 | X | Awake | Active |
|  |  | Sleep | Disabled |
| 01 | 1 | X | Active |
|  | 0 |  | Disabled |
| 00 | X | X | Disabled |

### TABLE 11-2: WDT CLEARING CONDITIONS

| Conditions | WDT |
|---|---|
| WDTE<1:0> = 00 | Cleared |
| WDTE<1:0> = 01 and SWDTEN = 0 |  |
| WDTE<1:0> = 10 and enter Sleep |  |
| `CLRWDT` Command |  |
| Oscillator Fail Detected |  |
| Exit Sleep + System Clock = T1OSC, EXTRC, INTOSC, EXTCLK |  |
| Exit Sleep + System Clock = XT, HS, LP | Cleared until the end of OST |
| Change INTOSC divider (IRCF bits) | Unaffected |

 © 2011-2012 Microchip Technology Inc.

## 11.6    Register Definitions: Watchdog Control

**REGISTER 11-1:    WDTCON: WATCHDOG TIMER CONTROL REGISTER**

| U-0 | U-0 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 | R/W-1/1 | R/W-0/0 |
|:---:|:---:|:---:|:---:|:---:|:---:|:---:|:---:|
| — | — | WDTPS<4:0> | | | | | SWDTEN |
| 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 | -m/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

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

bit 5-1  **WDTPS<4:0>:** Watchdog Timer Period Select bits[1]

Bit Value  =  Prescale Rate

11111 = Reserved. Results in minimum interval (1:32)

•
•
•

10011 = Reserved. Results in minimum interval (1:32)

10010 = 1:8388608 ($2^{23}$) (Interval 256s nominal)
10001 = 1:4194304 ($2^{22}$) (Interval 128s nominal)
10000 = 1:2097152 ($2^{21}$) (Interval 64s nominal)
01111 = 1:1048576 ($2^{20}$) (Interval 32s nominal)
01110 = 1:524288 ($2^{19}$) (Interval 16s nominal)
01101 = 1:262144 ($2^{18}$) (Interval 8s nominal)
01100 = 1:131072 ($2^{17}$) (Interval 4s nominal)
01011 = 1:65536  (Interval 2s nominal) (Reset value)
01010 = 1:32768 (Interval 1s nominal)
01001 = 1:16384 (Interval 512 ms nominal)
01000 = 1:8192 (Interval 256 ms nominal)
00111 = 1:4096 (Interval 128 ms nominal)
00110 = 1:2048 (Interval 64 ms nominal)
00101 = 1:1024 (Interval 32 ms nominal)
00100 = 1:512 (Interval 16 ms nominal)
00011 = 1:256 (Interval 8 ms nominal)
00010 = 1:128 (Interval 4 ms nominal)
00001 = 1:64 (Interval 2 ms nominal)
00000 = 1:32 (Interval 1 ms nominal)

bit 0  **SWDTEN:** Software Enable/Disable for Watchdog Timer bit
If WDTE<1:0> = 1x:
This bit is ignored.
If WDTE<1:0> = 01:
1 =  WDT is turned on
0 =  WDT is turned off
If WDTE<1:0> = 00:
This bit is ignored.

**Note 1:**    Times are approximate. WDT time is based on 31 kHz LFINTOSC.

# PIC16(L)F1782/3

TABLE 11-3: SUMMARY OF 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 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| OSCCON | SPLLEN | IRCF<3:0> | | | | — | SCS<1:0> | | 72 |
| STATUS | — | — | — | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | Z | DC | C | 23 |
| WDTCON | — | — | WDTPS<4:0> | | | | | SWDTEN | 101 |

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

TABLE 11-4: SUMMARY OF CONFIGURATION WORD WITH WATCHDOG TIMER

| Name | Bits | Bit -/7 | Bit -/6 | Bit 13/5 | Bit 12/4 | Bit 11/3 | Bit 10/2 | Bit 9/1 | Bit 8/0 | Register on Page |
|------|------|---------|---------|----------|----------|----------|----------|---------|---------|------------------|
| CONFIG1 | 13:8 | — | — | FCMEN | IESO | CLKOUTEN | BOREN<1:0> | | $\overline{\text{CPD}}$ | 44 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | FOSC<2:0> | | | |

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

## 12.0 DATA EEPROM AND FLASH PROGRAM MEMORY CONTROL

The data EEPROM and Flash program memory are readable and writable during normal operation (full V_DD range). These memories are not directly mapped in the register file space. Instead, they are indirectly addressed through the Special Function Registers (SFRs). There are six SFRs used to access these memories:

• EECON1
• EECON2
• EEDATL
• EEDATH
• EEADRL
• EEADRH

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

When accessing the program memory block, the EEDATH:EEDATL register pair forms a 2-byte word that holds the 14-bit data for read/write, and the EEADRL and EEADRH registers form a 2-byte word that holds the 15-bit address of the program memory location being read.

The EEPROM data memory allows byte read and write. An EEPROM byte write automatically erases the location and writes the new data (erase before write).

The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump rated to operate over the voltage range of the device for byte or word operations.

Depending on the setting of the Flash Program Memory Self Write Enable bits WRT<1:0> of the Configuration Words, the device may or may not be able to write certain blocks of the program memory. However, reads from the program memory are always allowed.

When the device is code-protected, the device programmer can no longer access data or program memory. When code-protected, the CPU may continue to read and write the data EEPROM memory and Flash program memory.

## 12.1 EEADRL and EEADRH Registers

The EEADRH:EEADRL register pair can address up to a maximum of 256 bytes of data EEPROM or up to a maximum of 32K words of program memory.

When selecting a program address value, the MSB of the address is written to the EEADRH register and the LSB is written to the EEADRL register. When selecting a EEPROM address value, only the LSB of the address is written to the EEADRL register.

### 12.1.1 EECON1 AND EECON2 REGISTERS

EECON1 is the control register for EE memory accesses.

Control bit EEPGD determines if the access will be a program or data memory access. When clear, any subsequent operations will operate on the EEPROM memory. When set, any subsequent operations will operate on the program memory. On Reset, EEPROM is selected by default.

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

The WREN bit, when set, will allow a write operation to occur. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a Reset during normal operation. In these situations, following Reset, the user can check the WRERR bit and execute the appropriate error handling routine.

Interrupt flag bit EEIF of the PIR2 register is set when write is complete. It must be cleared in the software.

Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the data EEPROM write sequence. To enable writes, a specific pattern must be written to EECON2.

## 12.2 Using the Data EEPROM

The data EEPROM is a high-endurance, byte address-able 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 **Section 30.0 "Electrical Specifications"**. If this is the case, then a refresh of the array must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in Flash program memory.

### 12.2.1 READING THE DATA EEPROM MEMORY

To read a data memory location, the user must write the address to the EEADRL register, clear the EEPGD and CFGS control bits of the EECON1 register, and then set control bit RD. The data is available at the very next cycle, in the EEDATL register; therefore, it can be read in the next instruction. EEDATL will hold this value until another read or until it is written to by the user (during a write operation).

**EXAMPLE 12-1: DATA EEPROM READ**

```
BANKSEL EEADRL        ;
MOVLW   DATA_EE_ADDR ;
MOVWF   EEADRL       ;Data Memory
                     ;Address to read
BCF     EECON1, CFGS ;Deselect Config space
BCF     EECON1, EEPGD;Point to DATA memory
BSF     EECON1, RD   ;EE Read
MOVF    EEDATL, W    ;W = EEDATL
```

| Note: | Data EEPROM can be read regardless of the setting of the $\overline{CPD}$ bit. |
|---|---|

### 12.2.2 WRITING TO THE DATA EEPROM MEMORY

To write an EEPROM data location, the user must first write the address to the EEADRL register and the data to the EEDATL register. Then the user must follow a specific sequence to initiate the write for each byte.

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

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

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. The WR bit will be inhibited from being set unless the WREN bit is set.

At the completion of the write cycle, the WR bit is cleared in hardware and the EE Write Complete Interrupt Flag bit (EEIF) is set. The user can either enable this interrupt or poll this bit. EEIF must be cleared by software.

### 12.2.3 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 built-in. On power-up, WREN is cleared. Also, the Power-up Timer (64 ms duration) prevents EEPROM write.

The write initiate sequence and the WREN bit together help prevent an accidental write during:

• Brown-out
• Power Glitch
• Software Malfunction

### 12.2.4 DATA EEPROM OPERATION DURING CODE-PROTECT

Data memory can be code-protected by programming the $\overline{CPD}$ bit in the Configuration Words to '0'.

When the data memory is code-protected, only the CPU is able to read and write data to the data EEPROM. It is recommended to code-protect the program memory when code-protecting data memory. This prevents anyone from replacing your program with a program that will access the contents of the data EEPROM.

**EXAMPLE 12-2:    DATA EEPROM WRITE**

```
        BANKSEL  EEADRL          ;
        MOVLW    DATA_EE_ADDR     ;
        MOVWF    EEADRL           ;Data Memory Address to write
        MOVLW    DATA_EE_DATA     ;
        MOVWF    EEDATL           ;Data Memory Value to write
        BCF      EECON1, CFGS     ;Deselect Configuration space
        BCF      EECON1, EEPGD    ;Point to DATA memory
        BSF      EECON1, WREN     ;Enable writes

        BCF      INTCON, GIE      ;Disable INTs.
        MOVLW    55h              ;
        MOVWF    EECON2           ;Write 55h
        MOVLW    0AAh             ;
        MOVWF    EECON2           ;Write AAh
        BSF      EECON1, WR       ;Set WR bit to begin write
        BSF      INTCON, GIE      ;Enable Interrupts
        BCF      EECON1, WREN     ;Disable writes
        BTFSC    EECON1, WR       ;Wait for write to complete
        GOTO     $-2              ;Done
```

(Required Sequence brackets the block from `MOVLW 55h` through `BSF EECON1, WR`)

**FIGURE 12-1:        FLASH PROGRAM MEMORY READ CYCLE EXECUTION**
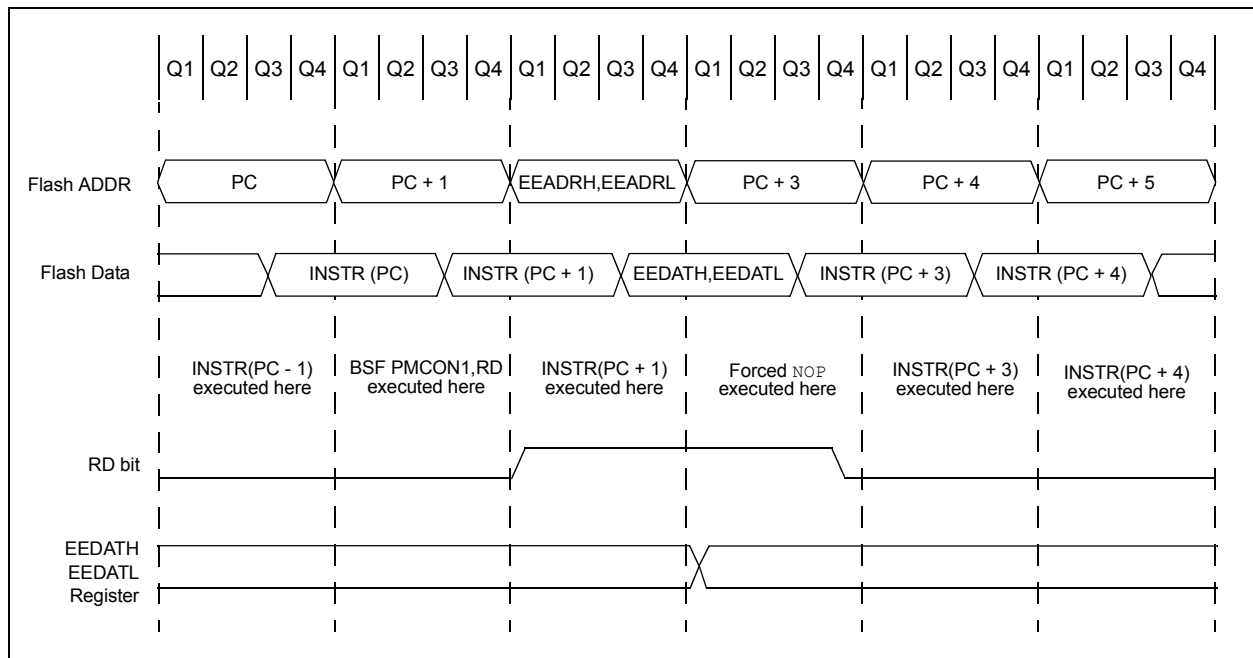
# PIC16(L)F1782/3

## 12.3 Flash Program Memory Overview

It is important to understand the Flash program memory structure for erase and programming operations. Flash Program memory is arranged in rows. A row consists of a fixed number of 14-bit program memory words. A row is the minimum block size that can be erased by user software.

Flash program memory may only be written or erased if the destination address is in a segment of memory that is not write-protected, as defined in bits WRT<1:0> of Configuration Words.

After a row has been erased, the user can reprogram all or a portion of this row. Data to be written into the program memory row is written to 14-bit wide data write latches. These write latches are not directly accessible to the user, but may be loaded via sequential writes to the EEDATH:EEDATL register pair.

> **Note:** If the user wants to modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase.

The number of data write latches may not be equivalent to the number of row locations. During programming, user software may need to fill the set of write latches and initiate a programming operation multiple times in order to fully reprogram an erased row. For example, a device with a row size of 32 words and eight write latches will need to load the write latches with data and initiate a programming operation four times.

The size of a program memory row and the number of program memory write latches may vary by device. See Table 12-1 for details.

**TABLE 12-1: FLASH MEMORY ORGANIZATION BY DEVICE**

| Device | Erase Block (Row) Size/Boundary | Number of Write Latches/Boundary |
|---|---|---|
| PIC16F1782 PIC16LF1782 PIC16F1783 PIC16LF1783 | 32 words, EEADRL<4:0> = 00000 | 32 words, EEADRL<4:0> = 00000 |

### 12.3.1 READING THE FLASH PROGRAM MEMORY

To read a program memory location, the user must:

1. Write the Least and Most Significant address bits to the EEADRH:EEADRL register pair.
2. Clear the CFGS bit of the EECON1 register.
3. Set the EEPGD control bit of the EECON1 register.
4. Then, set control bit RD of the EECON1 register.

Once the read control bit is set, the program memory Flash controller will use the second instruction cycle to read the data. This causes the second instruction immediately following the "`BSF EECON1,RD`" instruction to be ignored. The data is available in the very next cycle, in the EEDATH:EEDATL register pair; therefore, it can be read as two bytes in the following instructions.

EEDATH:EEDATL register pair will hold this value until another read or until it is written to by the user.

> **Note 1:** The two instructions following a program memory read are required to be NOPs. This prevents the user from executing a two-cycle instruction on the next instruction after the RD bit is set.
>
> **2:** Flash program memory can be read regardless of the setting of the $\overline{CP}$ bit.

**EXAMPLE 12-3:    FLASH PROGRAM MEMORY READ**

```
* This code block will read 1 word of program
* memory at the memory address:
    PROG_ADDR_HI : PROG_ADDR_LO
*   data will be returned in the variables;
*   PROG_DATA_HI, PROG_DATA_LO

    BANKSEL    EEADRL            ; Select Bank for EEPROM registers
    MOVLW      PROG_ADDR_LO      ;
    MOVWF      EEADRL            ; Store LSB of address
    MOVLW      PROG_ADDR_HI      ;
    MOVWL      EEADRH            ; Store MSB of address

    BCF        EECON1,CFGS       ; Do not select Configuration Space
    BSF        EECON1,EEPGD      ; Select Program Memory
    BCF        INTCON,GIE        ; Disable interrupts
    BSF        EECON1,RD         ; Initiate read
    NOP                          ; Executed (Figure 12-1)
    NOP                          ; Ignored (Figure 12-1)
    BSF        INTCON,GIE        ; Restore interrupts

    MOVF       EEDATL,W          ; Get LSB of word
    MOVWF      PROG_DATA_LO      ; Store in user location
    MOVF       EEDATH,W          ; Get MSB of word
    MOVWF      PROG_DATA_HI      ; Store in user location
```

## 12.3.2 ERASING FLASH PROGRAM MEMORY

While executing code, program memory can only be erased by rows. To erase a row:

1. Load the EEADRH:EEADRL register pair with the address of new row to be erased.
2. Clear the CFGS bit of the EECON1 register.
3. Set the EEPGD, FREE, and WREN bits of the EECON1 register.
4. Write 55h, then AAh, to EECON2 (Flash programming unlock sequence).
5. Set control bit WR of the EECON1 register to begin the erase operation.
6. Poll the FREE bit in the EECON1 register to determine when the row erase has completed.

See Example 12-4.

After the "BSF EECON1,WR" instruction, the processor requires two cycles to set up the erase operation. The user must place two NOP instructions after the WR bit is set. The processor will halt internal operations for the typical 2 ms erase time. This is not Sleep mode as the clocks and peripherals will continue to run. After the erase cycle, the processor will resume operation with the third instruction after the EECON1 write instruction.

## 12.3.3 WRITING TO FLASH PROGRAM MEMORY

Program memory is programmed using the following steps:

1. Load the starting address of the word(s) to be programmed.
2. Load the write latches with data.
3. Initiate a programming operation.
4. Repeat steps 1 through 3 until all data is written.

Before writing to program memory, the word(s) to be written must be erased or previously unwritten. Program memory can only be erased one row at a time. No automatic erase occurs upon the initiation of the write.

Program memory can be written one or more words at a time. The maximum number of words written at one time is equal to the number of write latches. See Figure 12-2 (block writes to program memory with 32 write latches) for more details. The write latches are aligned to the address boundary defined by EEADRL as shown in Table 12-1. Write operations do not cross these boundaries. At the completion of a program memory write operation, the write latches are reset to contain 0x3FFF.

The following steps should be completed to load the write latches and program a block of program memory. These steps are divided into two parts. First, all write latches are loaded with data except for the last program memory location. Then, the last write latch is loaded and the programming sequence is initiated. A special unlock sequence is required to load a write latch with data or initiate a Flash programming operation. This unlock sequence should not be interrupted.

1. Set the EEPGD and WREN bits of the EECON1 register.
2. Clear the CFGS bit of the EECON1 register.
3. Set the LWLO bit of the EECON1 register. When the LWLO bit of the EECON1 register is '1', the write sequence will only load the write latches and will not initiate the write to Flash program memory.
4. Load the EEADRH:EEADRL register pair with the address of the location to be written.
5. Load the EEDATH:EEDATL register pair with the program memory data to be written.
6. Write 55h, then AAh, to EECON2, then set the WR bit of the EECON1 register (Flash programming unlock sequence). The write latch is now loaded.
7. Increment the EEADRH:EEADRL register pair to point to the next location.
8. Repeat steps 5 through 7 until all but the last write latch has been loaded.
9. Clear the LWLO bit of the EECON1 register. When the LWLO bit of the EECON1 register is '0', the write sequence will initiate the write to Flash program memory.
10. Load the EEDATH:EEDATL register pair with the program memory data to be written.
11. Write 55h, then AAh, to EECON2, then set the WR bit of the EECON1 register (Flash programming unlock sequence). The entire latch block is now written to Flash program memory.

It is not necessary to load the entire write latch block with user program data. However, the entire write latch block will be written to program memory.

An example of the complete write sequence for eight words is shown in Example 12-5. The initial address is loaded into the EEADRH:EEADRL register pair; the eight words of data are loaded using indirect addressing.

After the "BSF EECON1,WR" instruction, the processor requires two cycles to set up the write operation. The user must place two NOP instructions after the WR bit is set. The processor will halt internal operations for the typical 2 ms, only during the cycle in which the write takes place (i.e., the last word of the block write). This is not Sleep mode as the clocks and peripherals will continue to run. The processor does not stall when LWLO = 1, loading the write latches. After the write cycle, the processor will resume operation with the third instruction after the EECON1 WRITE instruction.

**FIGURE 12-2:** **BLOCK WRITES TO FLASH PROGRAM MEMORY WITH 32 WRITE LATCHES**
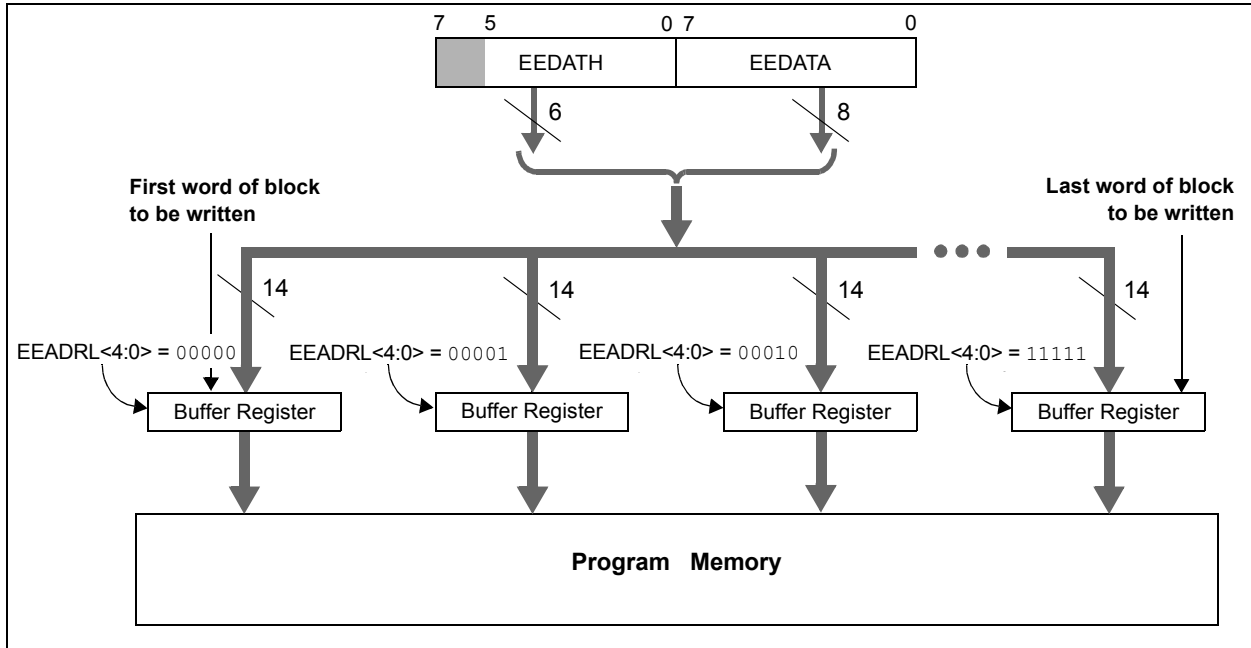


**EXAMPLE 12-4:** **ERASING ONE ROW OF PROGRAM MEMORY**

```
; This row erase routine assumes the following:
; 1. A valid address within the erase block is loaded in ADDRH:ADDRL
; 2. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F (common RAM)


        BCF         INTCON,GIE      ; Disable ints so required sequences will execute properly
        BANKSEL     EEADRL
        MOVF        ADDRL,W         ; Load lower 8 bits of erase address boundary
        MOVWF       EEADRL
        MOVF        ADDRH,W         ; Load upper 6 bits of erase address boundary
        MOVWF       EEADRH
        BSF         EECON1,EEPGD    ; Point to program memory
        BCF         EECON1,CFGS     ; Not configuration space
        BSF         EECON1,FREE     ; Specify an erase operation
        BSF         EECON1,WREN     ; Enable writes

        MOVLW       55h             ; Start of required sequence to initiate erase
        MOVWF       EECON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       EECON2          ; Write AAh
        BSF         EECON1,WR       ; Set WR bit to begin erase
        NOP                         ; Any instructions here are ignored as processor
                                    ; halts to begin erase sequence
        NOP                         ; Processor will stop here and wait for erase complete.

                                    ; after erase processor continues with 3rd instruction

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

*Required Sequence* (bracket label next to MOVLW 55h through NOP lines)

---

**EXAMPLE 12-5: WRITING TO FLASH PROGRAM MEMORY**

```
; This write routine assumes the following:
; 1. The 16 bytes of data are loaded, starting at the address in DATA_ADDR
; 2. Each word of data to be written is made up of two adjacent bytes in DATA_ADDR,
;    stored in little endian format
; 3. A valid starting address (the least significant bits = 000) is loaded in ADDRH:ADDRL
; 4. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F (common RAM)
;
        BCF        INTCON,GIE     ; Disable ints so required sequences will execute properly
        BANKSEL    EEADRH         ; Bank 3
        MOVF       ADDRH,W        ; Load initial address
        MOVWF      EEADRH         ;
        MOVF       ADDRL,W        ;
        MOVWF      EEADRL         ;
        MOVLW      LOW DATA_ADDR  ; Load initial data address
        MOVWF      FSR0L          ;
        MOVLW      HIGH DATA_ADDR ; Load initial data address
        MOVWF      FSR0H          ;
        BSF        EECON1,EEPGD   ; Point to program memory
        BCF        EECON1,CFGS    ; Not configuration space
        BSF        EECON1,WREN    ; Enable writes
        BSF        EECON1,LWLO    ; Only Load Write Latches
LOOP
        MOVIW      FSR0++         ; Load first data byte into lower
        MOVWF      EEDATL         ;
        MOVIW      FSR0++         ; Load second data byte into upper
        MOVWF      EEDATH         ;

        MOVF       EEADRL,W       ; Check if lower bits of address are '000'
        XORLW      0x07           ; Check if we're on the last of 8 addresses
        ANDLW      0x07           ;
        BTFSC      STATUS,Z       ; Exit if last of eight words,
        GOTO       START_WRITE    ;

┌─      MOVLW      55h            ; Start of required write sequence:
│       MOVWF      EECON2         ; Write 55h
│       MOVLW      0AAh           ;
│ Required
│ Sequence
│       MOVWF      EECON2         ; Write AAh
│       BSF        EECON1,WR      ; Set WR bit to begin write
│       NOP                       ; Any instructions here are ignored as processor
│                                 ; halts to begin write sequence
│       NOP                       ; Processor will stop here and wait for write to complete.
└─
                                  ; After write processor continues with 3rd instruction.

        INCF       EEADRL,F       ; Still loading latches Increment address
        GOTO       LOOP           ; Write next latches

START_WRITE
        BCF        EECON1,LWLO    ; No more loading latches - Actually start Flash program
                                  ; memory write

┌─      MOVLW      55h            ; Start of required write sequence:
│       MOVWF      EECON2         ; Write 55h
│       MOVLW      0AAh           ;
│ Required
│ Sequence
│       MOVWF      EECON2         ; Write AAh
│       BSF        EECON1,WR      ; Set WR bit to begin write
│       NOP                       ; Any instructions here are ignored as processor
│                                 ; halts to begin write sequence
│       NOP                       ; Processor will stop here and wait for write complete.
└─
                                  ; after write processor continues with 3rd instruction
        BCF        EECON1,WREN    ; Disable writes
        BSF        INTCON,GIE     ; Enable interrupts
```

## 12.4 Modifying Flash Program Memory

When modifying existing data in a program memory row, and data within that row must be preserved, it must first be read and saved in a RAM image. Program memory is modified using the following steps:

1. Load the starting address of the row to be modified.
2. Read the existing data from the row into a RAM image.
3. Modify the RAM image to contain the new data to be written into program memory.
4. Load the starting address of the row to be rewritten.
5. Erase the program memory row.
6. Load the write latches with data from the RAM image.
7. Initiate a programming operation.
8. Repeat steps 6 and 7 as many times as required to reprogram the erased row.

## 12.5 User ID, Device ID and Configuration Word Access

Instead of accessing program memory or EEPROM data memory, the User ID's, Device ID/Revision ID and Configuration Words can be accessed when CFGS = 1 in the EECON1 register. This is the region that would be pointed to by PC<15> = 1, but not all addresses are accessible. Different access may exist for reads and writes. Refer to Table 12-2.

When read access is initiated on an address outside the parameters listed in Table 12-2, the EEDATH:EEDATL register pair is cleared.

**TABLE 12-2: USER ID, DEVICE ID AND CONFIGURATION WORD ACCESS (CFGS = 1)**

| Address | Function | Read Access | Write Access |
|---------|----------|-------------|--------------|
| 8000h-8003h | User IDs | Yes | Yes |
| 8006h | Device ID/Revision ID | Yes | No |
| 8007h-8008h | Configuration Words 1 and 2 | Yes | No |

**EXAMPLE 12-3: CONFIGURATION WORD AND DEVICE ID ACCESS**

```
* This code block will read 1 word of program memory at the memory address:
*   PROG_ADDR_LO (must be 00h-08h) data will be returned in the variables;
*   PROG_DATA_HI, PROG_DATA_LO

    BANKSEL  EEADRL           ; Select correct Bank
    MOVLW    PROG_ADDR_LO     ;
    MOVWF    EEADRL           ; Store LSB of address
    CLRF     EEADRH           ; Clear MSB of address

    BSF      EECON1,CFGS      ; Select Configuration Space
    BCF      INTCON,GIE       ; Disable interrupts
    BSF      EECON1,RD        ; Initiate read
    NOP                       ; Executed (See Figure 12-1)
    NOP                       ; Ignored (See Figure 12-1)
    BSF      INTCON,GIE       ; Restore interrupts

    MOVF     EEDATL,W         ; Get LSB of word
    MOVWF    PROG_DATA_LO     ; Store in user location
    MOVF     EEDATH,W         ; Get MSB of word
    MOVWF    PROG_DATA_HI     ; Store in user location
```

## 12.6 Write Verify

Depending on the application, good programming practice may dictate that the value written to the data EEPROM or program memory should be verified (see Example 12-6) to the desired value to be written. Example 12-6 shows how to verify a write to EEPROM.

**EXAMPLE 12-6: EEPROM WRITE VERIFY**

```
BANKSEL EEDATL     ;
MOVF    EEDATL, W  ;EEDATL not changed
                   ;from previous write
BSF     EECON1, RD ;YES, Read the
                   ;value written
XORWF   EEDATL, W  ;
BTFSS   STATUS, Z  ;Is data the same
GOTO    WRITE_ERR  ;No, handle error
:                  ;Yes, continue
```

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 12.7 Register Definitions: EEPROM and Flash Control

### REGISTER 12-1: EEDATL: EEPROM DATA LOW BYTE REGISTER

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | EEDAT<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 | |

bit 7-0    **EEDAT<7:0>**: Read/write value for EEPROM data byte or Least Significant bits of program memory

### REGISTER 12-2: EEDATH: EEPROM DATA HIGH BYTE REGISTER

| U-0 | U-0 | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|-----|-----|---------|---------|---------|---------|---------|---------|
| — | — | | | EEDAT<13:8> | | | |
| 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    **Unimplemented:** Read as '0'

bit 5-0    **EEDAT<13:8>**: Read/write value for Most Significant bits of program memory

### REGISTER 12-3: EEADRL: EEPROM ADDRESS REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | EEADR<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 | |

bit 7-0    **EEADR<7:0>**: Specifies the Least Significant bits for program memory address or EEPROM address

### REGISTER 12-4: EEADRH: EEPROM ADDRESS HIGH BYTE REGISTER

| U-1 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-----|---------|---------|---------|---------|---------|---------|---------|
| —[1] | | | | EEADR<14:8> | | | |
| 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 '1'

bit 6-0    **EEADR<14:8>**: Specifies the Most Significant bits for program memory address or EEPROM address

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

**REGISTER 12-5: EECON1: EEPROM CONTROL 1 REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W/HC-0/0 | R/W-x/q | R/W-0/0 | R/S/HC-0/0 | R/S/HC-0/0 |
|---------|---------|---------|------------|---------|---------|------------|------------|
| EEPGD | CFGS | LWLO | FREE | WRERR | WREN | WR | RD |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---------|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| S = Bit can only be set | 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     **EEPGD:** Flash Program/Data EEPROM Memory Select bit

        1 = Accesses program space Flash memory
        0 = Accesses data EEPROM memory

bit 6     **CFGS:** Flash Program/Data EEPROM or Configuration Select bit

        1 = Accesses Configuration, User ID and Device ID registers
        0 = Accesses Flash Program or data EEPROM memory

bit 5     **LWLO:** Load Write Latches Only bit

        If CFGS = 1 (Configuration space) **OR** CFGS = 0 and EEPGD = 1 (program Flash):

           1 = The next WR command does not initiate a write; only the program memory latches are updated.
           0 = The next WR command writes a value from EEDATH:EEDATL into program memory latches and initiates a write of all the data stored in the program memory latches.

        If CFGS = 0 and EEPGD = 0: (Accessing data EEPROM)
        LWLO is ignored. The next WR command initiates a write to the data EEPROM.

bit 4     **FREE:** Program Flash Erase Enable bit

        If CFGS = 1 (Configuration space) **OR** CFGS = 0 and EEPGD = 1 (program Flash):

           1 = Performs an erase operation on the next WR command (cleared by hardware after completion of erase).
           0 = Performs a write operation on the next WR command.

        If EEPGD = 0 and CFGS = 0: (Accessing data EEPROM)
        FREE is ignored. The next WR command will initiate both a erase cycle and a write cycle.

bit 3     **WRERR:** EEPROM Error Flag bit

        1 = Condition indicates an improper program or erase sequence attempt or termination (bit is set automatically on any set attempt (write '1') of the WR bit).
        0 = The program or erase operation completed normally.

bit 2     **WREN:** Program/Erase Enable bit

        1 = Allows program/erase cycles
        0 = Inhibits programming/erasing of program Flash and data EEPROM

bit 1     **WR:** Write Control bit

        1 = Initiates a program Flash or data EEPROM program/erase operation.
           The operation is self-timed and the bit is cleared by hardware once operation is complete.
           The WR bit can only be set (not cleared) in software.
        0 = Program/erase operation to the Flash or data EEPROM is complete and inactive.

bit 0     **RD:** Read Control bit

        1 = Initiates an program Flash or data EEPROM read. Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software.
        0 = Does not initiate a program Flash or data EEPROM data read.

**REGISTER 12-6:    EECON2: EEPROM CONTROL 2 REGISTER**

| W-0/0 | W-0/0 | W-0/0 | W-0/0 | W-0/0 | W-0/0 | W-0/0 | W-0/0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| | | | EEPROM Control Register 2 | | | | |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---------|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| S = Bit can only be set | 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 7-0        **Data EEPROM Unlock Pattern bits**

To unlock writes, a 55h must be written first, followed by an AAh, before setting the WR bit of the EECON1 register. The value written to this register is used to unlock the writes. There are specific timing requirements on these writes. Refer to **Section 12.2.2 "Writing to the Data EEPROM Memory"** for more information.

**TABLE 12-3:    SUMMARY OF REGISTERS ASSOCIATED WITH DATA EEPROM**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| EECON1 | EEPGD | CFGS | LWLO | FREE | WRERR | WREN | WR | RD | 114 |
| EECON2 | | | EEPROM Control Register 2 (not a physical register) | | | | | | 115* |
| EEADRL | | | | EEADRL<7:0> | | | | | 113 |
| EEADRH | —(1) | | | | EEADRH<6:0> | | | | 113 |
| EEDATL | | | | EEDATL<7:0> | | | | | 113 |
| EEDATH | — | — | | | EEDATH<5:0> | | | | 113 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE2 | OSFIE | C2IE | C1IE | EEIE | BCLIE | — | C3IE | CCP2IE | 86 |
| PIR2 | OSFIF | C2IF | C1IF | EEIF | BCLIF | — | C3IF | CCP2IF | 89 |

**Legend:**    — = unimplemented location, read as '0'. Shaded cells are not used by data EEPROM module.
    *    Page provides register information.
    **2:**    Unimplemented, read as '1'.

**Preliminary**

## 13.0 I/O PORTS

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

• TRISx registers (data direction)
• PORTx registers (reads the levels on the pins of the device)
• LATx registers (output latch)

Some ports may have one or more of the following additional registers. These registers are:

• ANSELx (analog select)
• WPUx (weak pull-up)

In general, when a peripheral is enabled on a port pin, that pin cannot be used as a general purpose output. However, the pin can still be read.

**TABLE 13-1: PORT AVAILABILITY PER DEVICE**

| Device | PORTA | PORTB | PORTC | PORTE |
|---|---|---|---|---|
| PIC16(L)F1782 | ● | ● | ● | ● |
| PIC16(L)F1783 | ● | ● | ● | ● |

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

A write operation to the LATx register has the same effect as a write to the corresponding PORTx register. A read of the LATx register reads of the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

Ports that support analog inputs have an associated ANSELx register. When an ANSEL bit is set, the digital input buffer associated with that bit is disabled. Disabling the input buffer prevents analog signal levels on the pin between a logic high and low from causing excessive current in the logic input circuitry. A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 13-1.

**FIGURE 13-1: GENERIC I/O PORT OPERATION**
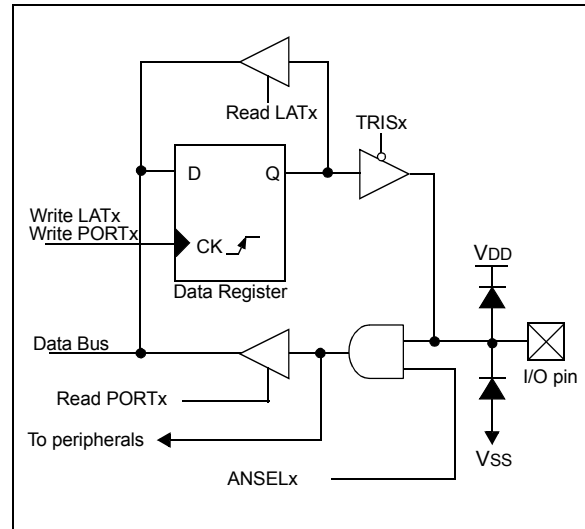


**EXAMPLE 13-1: INITIALIZING PORTA**

```
; This code example illustrates
; initializing the PORTA register. The
; other ports are initialized in the same
; manner.

BANKSEL PORTA        ;
CLRF    PORTA        ;Init PORTA
BANKSEL LATA         ;Data Latch
CLRF    LATA         ;
BANKSEL ANSELA       ;
CLRF    ANSELA       ;digital I/O
BANKSEL TRISA        ;
MOVLW   B'00111000'  ;Set RA<5:3> as inputs
MOVWF   TRISA        ;and set RA<2:0> as
                     ;outputs
```

## 13.1 Alternate Pin Function

The Alternate Pin Function Control (APFCON) register is used to steer specific peripheral input and output functions between different pins. The APFCON register is shown in Register 13-1. For this device family, the following functions can be moved between different pins.

- C2OUT output
- CCP1 output
- SDO output
- SCL/SCK output
- SDA/SDI output
- TX/RX output
- CCP2 output

These bits have no effect on the values of any TRIS register. PORT and TRIS overrides will be routed to the correct pin. The unselected pin will be unaffected.

**Preliminary**

## 13.2 Register Definitions: Alternate Pin Function Control

**REGISTER 13-1: APFCON: ALTERNATE PIN FUNCTION CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| C2OUTSEL | CCP1SEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL |
| 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      **C2OUTSEL**: C2OUT pin selection bit
           1 = C2OUT is on pin RA6
           0 = C2OUT is on pin RA5

bit 6      **CCP1SEL**: CCP1 Input/Output Pin Selection bit
           1 = CCP1 is on pin RB0
           0 = CCP1 is on pin RC2

bit 5      **SDOSEL**: MSSP SDO Pin Selection bit
           1 = SDO is on pin RB5
           0 = SDO is on pin RC5

bit 4      **SCKSEL**: MSSP Serial Clock (SCL/SCK) Pin Selection bit
           1 = SCL/SCK is on pin RB7
           0 = SCL/SCK is on pin RC3

bit 3      **SDISEL**: MSSP Serial Data (SDA/SDI) Output Pin Selection bit
           1 = SDA/SDI is on pin RB6
           0 = SDA/SDI is on pin RC4

bit 2      **TXSEL**: TX Pin Selection bit
           1 = TX is on pin RB6
           0 = TX is on pin RC6

bit 1      **RXSEL**: RX Pin Selection bit
           1 = RX is on pin RB7
           0 = RX is on pin RC7

bit 0      **CCP2SEL**: CCP2 Input/Output Pin Selection bit
           1 = CCP2 is on pin RB3
           0 = CCP2 is on pin RC1

## 13.3 PORTA Registers

### 13.3.1 DATA REGISTER

PORTA is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA (Register 13-3). 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., enables output driver and puts the contents of the output latch on the selected pin). The exception is RA3, which is input only and its TRIS bit will always read as '1'. Example 13-1 shows how to initialize PORTA.

Reading the PORTA register (Register 13-2) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATA).

### 13.3.2 DIRECTION CONTROL

The TRISA register (Register 13-3) controls the PORTA pin output drivers, 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. I/O pins configured as analog inputs always read '0'.

### 13.3.3 OPEN DRAIN CONTROL

The ODCONA register (Register 13-7) controls the open-drain feature of the port. Open drain operation is independently selected for each pin. When an ODCONA bit is set, the corresponding port output becomes an open drain driver capable of sinking current only. When an ODCONA bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

### 13.3.4 SLEW RATE CONTROL

The SLRCONA register (Register 13-8) controls the slew rate option for each port pin. Slew rate control is independently selectable for each port pin. When an SLRCONA bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONA bit is cleared, The corresponding port pin drive slews at the maximum rate possible.

### 13.3.5 INPUT THRESHOLD CONTROL

The INLVLA register (Register 13-9) controls the input voltage threshold for each of the available PORTA input pins. A selection between the Schmitt Trigger CMOS or the TTL Compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTA register and also the level at which an interrupt-on-change occurs, if that feature is enabled. See **Section 30.1 "DC Characteristics: PIC16(L)F1782/3-I/E (Industrial, Extended)"** for more information on threshold levels.

> **Note:** Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin.

### 13.3.6 ANALOG CONTROL

The ANSELA register (Register 13-5) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELA bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELA bits has no effect on digital output functions. A pin with TRIS clear and ANSEL set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

> **Note:** The ANSELA bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to '0' by user software.

### 13.3.7 PORTA FUNCTIONS AND OUTPUT PRIORITIES

Each PORTA pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 13-2.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the highest priority.

Analog input functions, such as ADC, and comparator 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 in the priority list.

**TABLE 13-2:    PORTA OUTPUT PRIORITY**

| Pin Name | Function Priority[1] |
|----------|----------------------|
| RA0 | RA0 |
| RA1 | OPA1OUT<br>RA1 |
| RA2 | DACOUT1<br>RA2 |
| RA3 | RA3 |
| RA4 | C1OUT<br>RA4 |
| RA5 | C2OUT<br>RA5 |
| RA6 | CLKOUT<br>C2OUT<br>RA6 |
| RA7 | RA7 |

**Note 1:** Priority listed from highest to lowest.

## 13.4 Register Definitions: PORTA

**REGISTER 13-2: PORTA: PORTA REGISTER**

| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
|---------|---------|---------|---------|-------|---------|---------|---------|
| RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 |
| 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 7-0      **RA<7:0>**: PORTA I/O Value bits[1]
                1 = Port pin is $\geq V_{IH}$
                0 = Port pin is $\leq V_{IL}$

**Note 1:** Writes to PORTA are actually written to corresponding LATA register. Reads from PORTA register is return of actual I/O pin values.

**REGISTER 13-3: TRISA: PORTA TRI-STATE REGISTER**

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|-------|---------|---------|---------|
| TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| 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      **TRISA<7:4>:** PORTA Tri-State Control bits
                1 = PORTA pin configured as an input (tri-stated)
                0 = PORTA pin configured as an output
bit 3           **TRISA3:** RA3 Port Tri-State Control bit
                This bit is always '1' as RA3 is an input only
bit 2-0      **TRISA<2:0>:** PORTA Tri-State Control bits
                1 = PORTA pin configured as an input (tri-stated)
                0 = PORTA pin configured as an output

**REGISTER 13-4: LATA: PORTA DATA LATCH REGISTER**

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 |
| 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      **LATA<7:0>**: PORTA Output Latch Value bits[1]

**Note 1:** Writes to PORTA are actually written to corresponding LATA register. Reads from PORTA register is return of actual I/O pin values.

**REGISTER 13-5:   ANSELA: PORTA ANALOG SELECT REGISTER**

| R/W-1/1 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|-----|---------|---------|---------|---------|---------|---------|
| ANSA7 | — | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 |
| 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 5        **ANSA7**: Analog Select between Analog or Digital Function on pins RA7, respectively
      0 =  Digital I/O. Pin is assigned to port or digital special function.
      1 =  Analog input. Pin is assigned as analog input[1]. Digital input buffer disabled.

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

bit 5-0      **ANSA<5:0>**: Analog Select between Analog or Digital Function on pins RA<5:0>, respectively
      0 =  Digital I/O. Pin is assigned to port or digital special function.
      1 =  Analog input. Pin is assigned as analog input[1]. Digital input buffer disabled.

**Note  1:**   When setting a pin to an analog input, the corresponding TRIS bit must be set to Input mode in order to allow external control of the voltage on the pin.

**REGISTER 13-6:   WPUA: WEAK PULL-UP PORTA REGISTER**

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| WPUA7 | WPUA6 | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 |
| 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 7-0      **WPUA<7:0>**: Weak Pull-up Register bits
      1 = Pull-up enabled
      0 = Pull-up disabled

**Note  1:**   Global $\overline{\text{WPUEN}}$ bit of the OPTION_REG register must be cleared for individual pull-ups to be enabled.
     **2:**   The weak pull-up device is automatically disabled if the pin is in configured as an output.

**REGISTER 13-7: ODCONA: PORTA OPEN DRAIN CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| ODA7 | ODA6 | ODA5 | ODA4 | ODA3 | ODA2 | ODA1 | ODA0 |
| 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 7-0 **ODA<7:0>:** PORTA Open Drain Enable bits
For RA<7:0> pins, respectively
1 = Port pin operates as open-drain drive (sink current only)
0 = Port pin operates as standard push-pull drive (source and sink current)

**REGISTER 13-8: SLRCONA: PORTA SLEW RATE CONTROL REGISTER**

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| SLRA7 | SLRA6 | SLRA5 | SLRA4 | SLRA3 | SLRA2 | SLRA1 | SLRA0 |
| 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 7-0 **SLRA<7:0>:** PORTA Slew Rate Enable bits
For RA<7:0> pins, respectively
1 = Port pin slew rate is limited
0 = Port pin slews at maximum rate

**REGISTER 13-9: INLVLA: PORTA INPUT LEVEL CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| INLVLA7 | INLVLA6 | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 |
| 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 7-0 **INLVLA<7:0>:** PORTA Input Level Select bits
For RA<7:0> pins, respectively
1 = ST input used for PORT reads and interrupt-on-change
0 = TTL input used for PORT reads and interrupt-on-change

**Preliminary** © 2011-2012 Microchip Technology Inc.

**TABLE 13-3: SUMMARY OF 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 | ANSA7 | — | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 123 |
| INLVLA | INLVLA7 | INLVLA6 | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 | 124 |
| LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | 122 |
| ODCONA | ODA7 | ODA6 | ODA5 | ODA4 | ODA3 | ODA2 | ODA1 | ODA0 | 124 |
| OPTION_REG | $\overline{\text{WPUEN}}$ | INTEDG | TMR0CS | TMR0SE | PSA | PS<2:0> | | | 183 |
| PORTA | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 122 |
| SLRCONA | SLRA7 | SLRA6 | SLRA5 | SLRA4 | SLRA3 | SLRA2 | SLRA1 | SLRA0 | 124 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 122 |
| WPUA | WPUA7 | WPUA6 | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 | 123 |

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

**TABLE 13-4: SUMMARY OF CONFIGURATION WORD WITH PORTA**

| Name | Bits | Bit -/7 | Bit -/6 | Bit 13/5 | Bit 12/4 | Bit 11/3 | Bit 10/2 | Bit 9/1 | Bit 8/0 | Register on Page |
|------|------|---------|---------|----------|----------|----------|----------|---------|---------|------------------|
| CONFIG1 | 13:8 | — | — | FCMEN | IESO | CLKOUTEN | BOREN<1:0> | | $\overline{\text{CPD}}$ | 44 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | FOSC<2:0> | | | |

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

## 13.5 PORTB Registers

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB (Register 13-11). Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin). Example 13-1 shows how to initialize an I/O port.

Reading the PORTB register (Register 13-10) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATB).

### 13.5.1 DIRECTION CONTROL

The TRISB register (Register 13-11) controls the PORTB pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISB register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

### 13.5.2 OPEN DRAIN CONTROL

The ODCONB register (Register 13-15) controls the open-drain feature of the port. Open drain operation is independently selected for each pin. When an ODCONB bit is set, the corresponding port output becomes an open drain driver capable of sinking current only. When an ODCONB bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

### 13.5.3 SLEW RATE CONTROL

The SLRCONB register (Register 13-16) controls the slew rate option for each port pin. Slew rate control is independently selectable for each port pin. When an SLRCONB bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONB bit is cleared, The corresponding port pin drive slews at the maximum rate possible.

### 13.5.4 INPUT THRESHOLD CONTROL

The INLVLB register (Register 13-17) controls the input voltage threshold for each of the available PORTB input pins. A selection between the Schmitt Trigger CMOS or the TTL Compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTB register and also the level at which an interrupt-on-change occurs, if that feature is enabled. See **Section 30.1 "DC Characteristics: PIC16(L)F1782/3-I/E (Industrial, Extended)"** for more information on threshold levels.

| Note: | Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin. |
|---|---|

### 13.5.5 ANALOG CONTROL

The ANSELB register (Register 13-13) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELB bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELB bits has no effect on digital output functions. A pin with TRIS clear and ANSELB set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

| Note: | The ANSELB bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to '0' by user software. |
|---|---|

## 13.5.6 PORTB FUNCTIONS AND OUTPUT PRIORITIES

Each PORTB pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 13-5.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the highest priority.

Analog input and some digital input functions are not included in the list below. These input functions can remain active when the pin is configured as an output. Certain digital input functions override other port functions and are included in the priority list.

**TABLE 13-5:    PORTB OUTPUT PRIORITY**

| Pin Name | Function Priority[1] |
|----------|-----------------------|
| RB0 | CCP1<br>RB0 |
| RB1 | OPA2OUT<br>RB1 |
| RB2 | CLKR<br>RB2 |
| RB3 | CCP2<br>RB3 |
| RB4 | RB4 |
| RB5 | SDO<br>C3OUT<br>RB5 |
| RB6 | ICSPCLK<br>SDA<br>TX/CK<br>RB6 |
| RB7 | ICSPDAT<br>DACOUT2<br>SCL/SCK<br>DT<br>RB7 |

**Note 1:** Priority listed from highest to lowest.

## 13.6    Register Definitions: PORTB

### REGISTER 13-10:    PORTB: PORTB REGISTER

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
| 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 7-0        **RB<7:0>**: PORTB General Purpose I/O Pin bits[1]
1 = Port pin is $\geq$ V$_{IH}$
0 = Port pin is $\leq$ V$_{IL}$

**Note  1:**    Writes to PORTB are actually written to corresponding LATB register. Reads from PORTB register is return of actual I/O pin values.

### REGISTER 13-11:    TRISB: PORTB TRI-STATE REGISTER

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| 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 7-0        **TRISB<7:0>:** PORTB Tri-State Control bits
1 = PORTB pin configured as an input (tri-stated)
0 = PORTB pin configured as an output

### REGISTER 13-12:    LATB: PORTB DATA LATCH REGISTER

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 |
| 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 7-0        **LATB<7:0>**: PORTB Output Latch Value bits[1]

**Note  1:**    Writes to PORTB are actually written to corresponding LATB register. Reads from PORTB register is return of actual I/O pin values.

**REGISTER 13-13: ANSELB: PORTB ANALOG SELECT REGISTER**

| U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|-----|-----|---------|---------|---------|---------|---------|---------|
| — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 |
| 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      **Unimplemented:** Read as '0'

bit 5-0      **ANSB<5:0>**: Analog Select between Analog or Digital Function on pins RB<5:0>, respectively
             0 =  Digital I/O. Pin is assigned to port or digital special function.
             1 =  Analog input. Pin is assigned as analog input[(1)]. Digital input buffer disabled.

**Note 1:** When setting a pin to an analog input, the corresponding TRIS bit must be set to Input mode in order to allow external control of the voltage on the pin.


**REGISTER 13-14: WPUB: WEAK PULL-UP PORTB REGISTER**

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| 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' |
| 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 7-0      **WPUB<7:0>**: Weak Pull-up Register bits
             1 =  Pull-up enabled
             0 =  Pull-up disabled

**Note 1:** Global $\overline{\text{WPUEN}}$ bit of the OPTION_REG register must be cleared for individual pull-ups to be enabled.

     **2:** The weak pull-up device is automatically disabled if the pin is in configured as an output.

# PIC16(L)F1782/3

### REGISTER 13-15: ODCONB: PORTB OPEN DRAIN CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| ODB7 | ODB6 | ODB5 | ODB4 | ODB3 | ODB2 | ODB1 | ODB0 |
| 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 7-0     **ODB<7:0>:** PORTB Open Drain Enable bits
            For RB<7:0> pins, respectively
            1 = Port pin operates as open-drain drive (sink current only)
            0 = Port pin operates as standard push-pull drive (source and sink current)

### REGISTER 13-16: SLRCONB: PORTB SLEW RATE CONTROL REGISTER

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| SLRB7 | SLRB6 | SLRB5 | SLRB4 | SLRB3 | SLRB2 | SLRB1 | SLRB0 |
| 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 7-0     **SLRB<7:0>:** PORTB Slew Rate Enable bits
            For RB<7:0> pins, respectively
            1 = Port pin slew rate is limited
            0 = Port pin slews at maximum rate

### REGISTER 13-17: INLVLB: PORTB INPUT LEVEL CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| INLVLB7 | INLVLB6 | INLVLB5 | INLVLB4 | INLVLB3 | INLVLB2 | INLVLB1 | INLVLB0 |
| 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 7-0     **INLVLB<7:0>:** PORTB Input Level Select bits
            For RB<7:0> pins, respectively
            1 = ST input used for PORT reads and interrupt-on-change
            0 = TTL input used for PORT reads and interrupt-on-change

**TABLE 13-6:    SUMMARY OF 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 | 129 |
| INLVLB | INLVLB7 | INLVLB6 | INLVLB5 | INLVLB4 | INLVLB3 | INLVLB2 | INLVLB1 | INLVLB0 | 130 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | 128 |
| ODCONB | ODB7 | ODB6 | ODB5 | ODB4 | ODB3 | ODB2 | ODB1 | ODB0 | 130 |
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | 128 |
| SLRCONB | SLRB7 | SLRB6 | SLRB5 | SLRB4 | SLRB3 | SLRB2 | SLRB1 | SLRB0 | 130 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 128 |
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 | 129 |

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

## 13.7 PORTC Registers

### 13.7.1 DATA REGISTER

PORTC is an 8-bit wide bidirectional port. The corresponding data direction register is TRISC (Register 13-19). Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin). Example 13-1 shows how to initialize an I/O port.

Reading the PORTC register (Register 13-18) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATC).

### 13.7.2 DIRECTION CONTROL

The TRISC register (Register 13-19) controls the PORTC pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISC register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

### 13.7.3 OPEN DRAIN CONTROL

The ODCONC register (Register 13-22) controls the open-drain feature of the port. Open drain operation is independently selected for each pin. When an ODCONC bit is set, the corresponding port output becomes an open drain driver capable of sinking current only. When an ODCONC bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

### 13.7.4 SLEW RATE CONTROL

The SLRCONC register (Register 13-23) controls the slew rate option for each port pin. Slew rate control is independently selectable for each port pin. When an SLRCONC bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONC bit is cleared, The corresponding port pin drive slews at the maximum rate possible.

### 13.7.5 INPUT THRESHOLD CONTROL

The INLVLC register (Register 13-24) controls the input voltage threshold for each of the available PORTC input pins. A selection between the Schmitt Trigger CMOS or the TTL Compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTC register and also the level at which an interrupt-on-change occurs, if that feature is enabled. See **Section 30.1 "DC Characteristics: PIC16(L)F1782/3-I/E (Industrial, Extended)"** for more information on threshold levels.

| Note: | Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin. |
|---|---|

### 13.7.6 PORTC FUNCTIONS AND OUTPUT PRIORITIES

Each PORTC pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 13-7.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the highest priority.

Analog input and some digital input functions are not included in the list below. These input functions can remain active when the pin is configured as an output. Certain digital input functions override other port functions and are included in the priority list.

### TABLE 13-7: PORTC OUTPUT PRIORITY

| Pin Name | Function Priority[1] |
|---|---|
| RC0 | T1OSO<br>PSMC1A<br>RC0 |
| RC1 | PSMC1B<br>CCP2<br>RC1 |
| RC2 | PSMC1C<br>CCP1<br>RC2 |
| RC3 | PSMC1D<br>SCL<br>SCK<br>RC3 |
| RC4 | PSMC1E<br>SDA<br>RC4 |
| RC5 | PSMC1F<br>SDO<br>RC5 |
| RC6 | PSMC2A<br>TX/CK<br>RC6 |
| RC7 | PSMC2B<br>DT<br>RC7 |

**Note 1:** Priority listed from highest to lowest.

## 13.8 Register Definitions: PORTC

**REGISTER 13-18: PORTC: PORTC REGISTER**

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

| Legend: | | |
|---------|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| 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 7-0      **RC<7:0>**: PORTC General Purpose I/O Pin bits[1]
            1 = Port pin is $\geq$ $V_{IH}$
            0 = Port pin is $\leq$ $V_{IL}$

**Note 1:** Writes to PORTC are actually written to corresponding LATC register. Reads from PORTC register is return of actual I/O pin values.

**REGISTER 13-19: TRISC: PORTC TRI-STATE REGISTER**

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

| Legend: | | |
|---------|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| 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 7-0      **TRISC<7:0>:** PORTC Tri-State Control bits
            1 = PORTC pin configured as an input (tri-stated)
            0 = PORTC pin configured as an output

**REGISTER 13-20: LATC: PORTC DATA LATCH REGISTER**

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

| Legend: | | |
|---------|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| 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 7-0      **LATC<7:0>**: PORTC Output Latch Value bits[1]

**Note 1:** Writes to PORTC are actually written to corresponding LATC register. Reads from PORTC register is return of actual I/O pin values.

### REGISTER 13-21: WPUC: WEAK PULL-UP PORTC REGISTER

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 |
| 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 7-0      **WPUC<7:0>**: Weak Pull-up Register bits
                     1 = Pull-up enabled
                     0 = Pull-up disabled

**Note 1:** Global $\overline{\text{WPUEN}}$ bit of the OPTION_REG register must be cleared for individual pull-ups to be enabled.
       **2:** The weak pull-up device is automatically disabled if the pin is in configured as an output.

### REGISTER 13-22: ODCONC: PORTC OPEN DRAIN CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| ODC7 | ODC6 | ODC5 | ODC4 | ODC3 | ODC2 | ODC1 | ODC0 |
| 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 7-0      **ODC<7:0>:** PORTC Open Drain Enable bits
                     For RC<7:0> pins, respectively
                     1 = Port pin operates as open-drain drive (sink current only)
                     0 = Port pin operates as standard push-pull drive (source and sink current)

### REGISTER 13-23: SLRCONC: PORTC SLEW RATE CONTROL REGISTER

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| SLRC7 | SLRC6 | SLRC5 | SLRC4 | SLRC3 | SLRC2 | SLRC1 | SLRC0 |
| 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 7-0      **SLRC<7:0>:** PORTC Slew Rate Enable bits
                     For RC<7:0> pins, respectively
                     1 = Port pin slew rate is limited
                     0 = Port pin slews at maximum rate

**REGISTER 13-24: INLVLC: PORTC INPUT LEVEL CONTROL REGISTER**

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 |
| 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 7-0 **INLVLC<7:0>:** PORTC Input Level Select bits
For RC<7:0> pins, respectively
1 = ST input used for PORT reads and interrupt-on-change
0 = TTL input used for PORT reads and interrupt-on-change

**TABLE 13-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | 133 |
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | 133 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |
| WPUC | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 | 134 |
| INLVLC | INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 | 135 |
| LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | 133 |
| ODCONC | ODC7 | ODC6 | ODC5 | ODC4 | ODC3 | ODC2 | ODC1 | ODC0 | 134 |
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | 133 |
| SLRCONC | SLRC7 | SLRC6 | SLRC5 | SLRC4 | SLRC3 | SLRC2 | SLRC1 | SLRC0 | 134 |

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

# PIC16(L)F1782/3

## 13.9 PORTE Registers

RE3 is input only, and also functions as $\overline{MCLR}$. The $\overline{MCLR}$ feature can be disabled via a configuration fuse. RE3 also supplies the programming voltage. The TRIS bit for RE3 (TRISE3) always reads '1'.

### 13.9.1 INPUT THRESHOLD CONTROL

The INLVLE register (Register 13-28) controls the input voltage threshold for each of the available PORTE input pins. A selection between the Schmitt Trigger CMOS or the TTL Compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTE register and also the level at which an interrupt-on-change occurs, if that feature is enabled. See **Section 30.1 "DC Characteristics: PIC16(L)F1782/3-I/E (Industrial, Extended)"** for more information on threshold levels.

| Note: | Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin. |
|---|---|

### REGISTER 13-25: PORTE: PORTE REGISTER

| U-0 | U-0 | U-0 | U-0 | R-x/u | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | RE3 | — | — | — |
| 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 **RE3**: PORTE Input Pin bit
1 = Port pin is > $V_{IH}$
0 = Port pin is < $V_{IL}$

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

### 13.9.2 PORTE FUNCTIONS AND OUTPUT PRIORITIES

No output priorities, RE3 is an input only pin.

**Preliminary**

© 2011-2012 Microchip Technology Inc.

## 13.10   Register Definitions: PORTE

### REGISTER 13-26:   TRISE: PORTE TRI-STATE REGISTER

| U-0 | U-0 | U-0 | U-0 | U-1[1] | U-0 | U-0 | U-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 7-4      **Unimplemented**: Read as '0'

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

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

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

### REGISTER 13-27:   WPUE: WEAK PULL-UP PORTE REGISTER

| U-0 | U-0 | U-0 | U-0 | R/W-1/1 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | WPUE3 | — | — | — |
| 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         **WPUE3:** Weak Pull-up Register bit
               1 = Pull-up enabled
               0 = Pull-up disabled

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

**Note   1:**   Global $\overline{WPUEN}$ bit of the OPTION_REG register must be cleared for individual pull-ups to be enabled.

**2:**   The weak pull-up device is automatically disabled if the pin is in configured as an output.

# PIC16(L)F1782/3

## REGISTER 13-28: INLVLE: PORTE INPUT LEVEL CONTROL REGISTER

| U-0 | U-0 | U-0 | U-0 | R/W-1/1 | U-0 | U-0 | U-0 |
|-----|-----|-----|-----|---------|-----|-----|-----|
| — | — | — | — | INLVLE3 | — | — | — |
| 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      **INLVLE3:** PORTE Input Level Select bit
           1 = ST input used for PORT reads and interrupt-on-change
           0 = TTL input used for PORT reads and interrupt-on-change

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

## TABLE 13-9: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

| 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/$\overline{\text{DONE}}$ | ADON | 155 |
| INLVLE | — | — | — | — | INLVLE3 | — | — | — | 138 |
| PORTE | — | — | — | — | RE3 | — | — | — | 136 |
| TRISE | — | — | — | — | —[1] | — | — | — | 137 |
| WPUE | — | — | — | — | WPUE3 | — | — | — | 137 |

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

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

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 14.0 INTERRUPT-ON-CHANGE

All pins on all ports can be configured to operate as Interrupt-On-Change (IOC) pins. An interrupt can be generated by detecting a signal that has either a rising edge or a falling edge. Any individual pin, or combination of pins, can be configured to generate an interrupt. The interrupt-on-change module has the following features:

- Interrupt-on-Change enable (Master Switch)
- Individual pin configuration
- Rising and falling edge detection
- Individual pin interrupt flags

Figure 14-1 is a block diagram of the IOC module.

### 14.1 Enabling the Module

To allow individual pins to generate an interrupt, the IOCIE bit of the INTCON register must be set. If the IOCIE bit is disabled, the edge detection on the pin will still occur, but an interrupt will not be generated.

### 14.2 Individual Pin Configuration

For each pin, a rising edge detector and a falling edge detector are present. To enable a pin to detect a rising edge, the associated bit of the IOCxP register is set. To enable a pin to detect a falling edge, the associated bit of the IOCxN register is set.

A pin can be configured to detect rising and falling edges simultaneously by setting the associated bits in both of the IOCxP and IOCxN registers.

### 14.3 Interrupt Flags

The bits located in the IOCxF registers are status flags that correspond to the Interrupt-on-change pins of each port. If an expected edge is detected on an appropriately enabled pin, then the status flag for that pin will be set, and an interrupt will be generated if the IOCIE bit is set. The IOCIF bit of the INTCON register reflects the status of all IOCxF bits.

### 14.4 Clearing Interrupt Flags

The individual status flags, (IOCxF register bits), can be cleared by resetting them to zero. If another edge is detected during this clearing operation, the associated status flag will be set at the end of the sequence, regardless of the value actually being written.

In order to ensure that no detected edge is lost while clearing flags, only AND operations masking out known changed bits should be performed. The following sequence is an example of what should be performed.

**EXAMPLE 14-1: CLEARING INTERRUPT FLAGS (PORTA EXAMPLE)**

```
    MOVLW  0xff
    XORWF  IOCAF, W
    ANDWF  IOCAF, F
```

### 14.5 Operation in Sleep

The interrupt-on-change interrupt sequence will wake the device from Sleep mode, if the IOCIE bit is set.

If an edge is detected while in Sleep mode, the affected IOCxF register will be updated prior to the first instruction executed out of Sleep.

**EXAMPLE 14-2: CLEARING INTERRUPT FLAGS (PORTA EXAMPLE)**

```
    MOVLW  0xff
    XORWF  IOCAF, W
    ANDWF  IOCAF, F
```

**FIGURE 14-1:** **INTERRUPT-ON-CHANGE BLOCK DIAGRAM**



**Preliminary** © 2011-2012 Microchip Technology Inc.
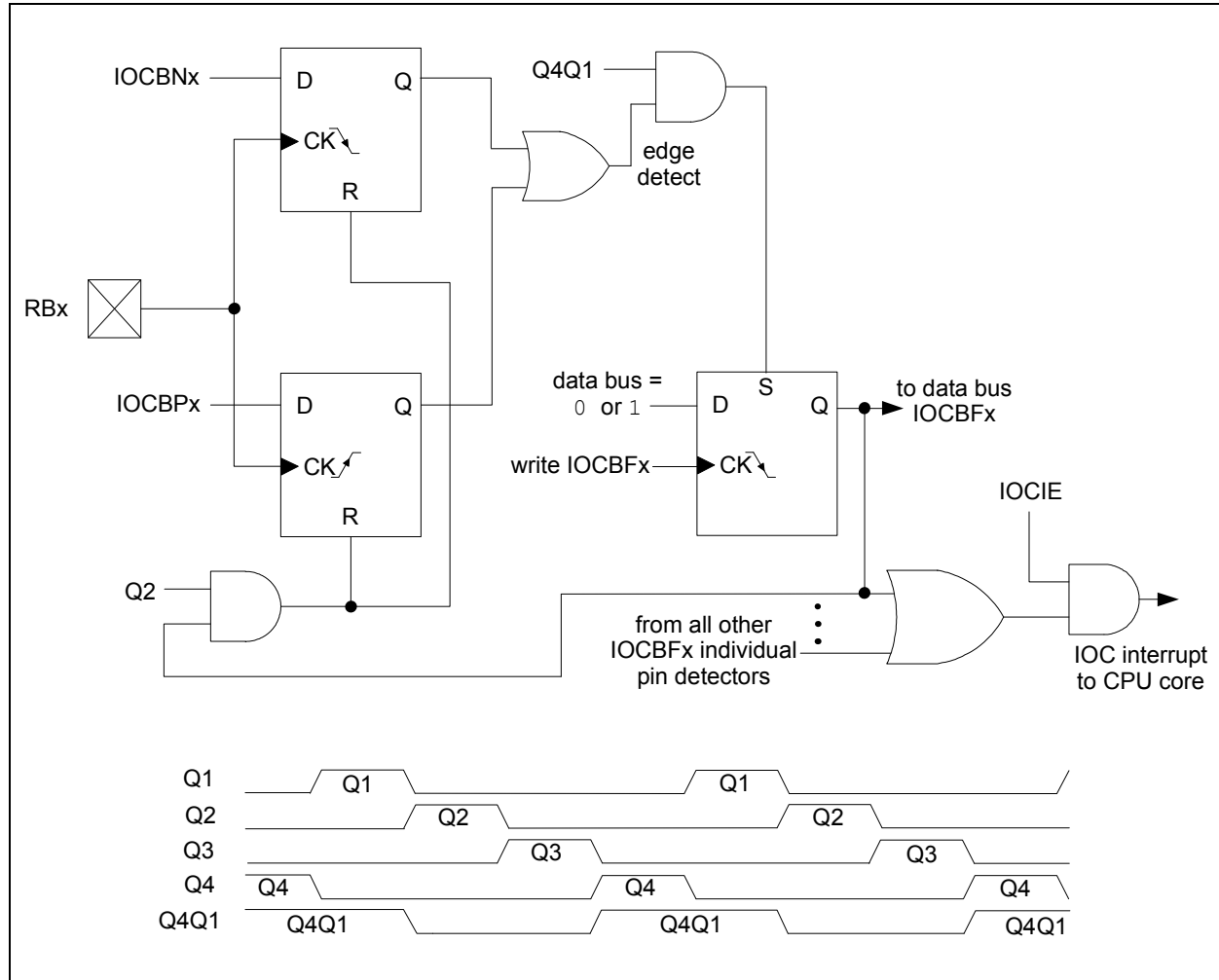
## 14.6    Interrupt-On-Change Registers

**REGISTER 14-1:    IOCxP: INTERRUPT-ON-CHANGE POSITIVE EDGE REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| IOCxP7 | IOCxP6 | IOCxP5 | IOCxP4 | IOCxP3 | IOCxP2 | IOCxP1 | IOCxP0 |
| 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 7-0    **IOC$x$P<7:0>:** Interrupt-on-Change Positive Edge Enable bits[1]

> 1 = Interrupt-on-Change enabled on the pin for a positive going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.
> 0 = Interrupt-on-Change disabled for the associated pin.

**Note 1:**    For IOCEP register, bit 3 (IOCEP3) is the only implemented bit in the register.

**REGISTER 14-2:    IOCxN: INTERRUPT-ON-CHANGE NEGATIVE EDGE REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| IOCxN7 | IOCxN6 | IOCxN5 | IOCxN4 | IOCxN3 | IOCxN2 | IOCxN1 | IOCxN0 |
| 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 7-0    **IOCxN<7:0>:** Interrupt-on-Change Negative Edge Enable bits[1]

> 1 = Interrupt-on-Change enabled on the pin for a negative going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.
> 0 = Interrupt-on-Change disabled for the associated pin.

**Note 1:**    For IOCEN register, bit 3 (IOCEN3) is the only implemented bit in the register.

# PIC16(L)F1782/3

## REGISTER 14-3: IOCxF: INTERRUPT-ON-CHANGE FLAG REGISTER

| R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
|---|---|---|---|---|---|---|---|
| IOCxF7 | IOCxF6 | IOCxF5 | IOCxF4 | IOCxF3 | IOCxF2 | IOCxF1 | IOCxF0 |
| 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 | HS - Bit is set in hardware |

bit 7-0    **IOCxF<7:0>:** Interrupt-on-Change Flag bits[1]

    1 = An enabled change was detected on the associated pin.
        Set when IOCxPx = 1 and a rising edge was detected RBx, or when IOCxNx = 1 and a falling edge
        was detected on RBx.
    0 = No change was detected, or the user cleared the detected change.

 **Note  1:**    For IOCEF register, bit 3 (IOCEF3) is the only implemented bit in the register.

## TABLE 14-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE

| 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 | 129 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| IOCAF | IOCAF7 | IOCAF6 | IOCAF5 | IOCAF4 | IOCAF3 | IOCAF2 | IOCAF1 | IOCAF0 | 142 |
| IOCAN | IOCAN7 | IOCAN6 | IOCAN5 | IOCAN4 | IOCAN3 | IOCAN2 | IOCAN1 | IOCAN0 | 141 |
| IOCAP | IOCAP7 | IOCAP6 | IOCAP5 | IOCAP4 | IOCAP3 | IOCAP2 | IOCAP1 | IOCAP0 | 141 |
| IOCBF | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 | 142 |
| IOCBN | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 | 141 |
| IOCBP | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 | 141 |
| IOCCF | IOCCF7 | IOCCF6 | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 | 142 |
| IOCCN | IOCCN7 | IOCCN6 | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 | 141 |
| IOCCP | IOCCP7 | IOCCP6 | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 | 141 |
| IOCEF | — | — | — | — | IOCEF3 | — | — | — | 142 |
| IOCEN | — | — | — | — | IOCEN3 | — | — | — | 141 |
| IOCEP | — | — | — | — | IOCEP3 | — | — | — | 141 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 128 |

 **Legend:**    — = unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.

## 15.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of V$_{DD}$, 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 FVRCON register.

## 15.1 Independent Gain Amplifiers

The output of the FVR supplied to the ADC, Comparators, and DAC is routed through two independent programmable gain amplifiers. Each amplifier can be programmed for a gain of 1x, 2x or 4x, to produce the three possible voltage levels.

The ADFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module. Reference **Section 19.0 "Digital-to-Analog Converter (DAC) Module"** for additional information.

The CDAFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the DAC and comparator module. Reference Section 17.0 **Section 19.0 "Digital-to-Analog Converter (DAC) Module"** and **Section 20.0 "Comparator Module"** for additional information.

## 15.2 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 FVRRDY bit of the FVRCON register will be set. See **Section 30.0 "Electrical Specifications"** for the minimum delay requirement.

# PIC16(L)F1782/3

**FIGURE 15-1:** **VOLTAGE REFERENCE BLOCK DIAGRAM**
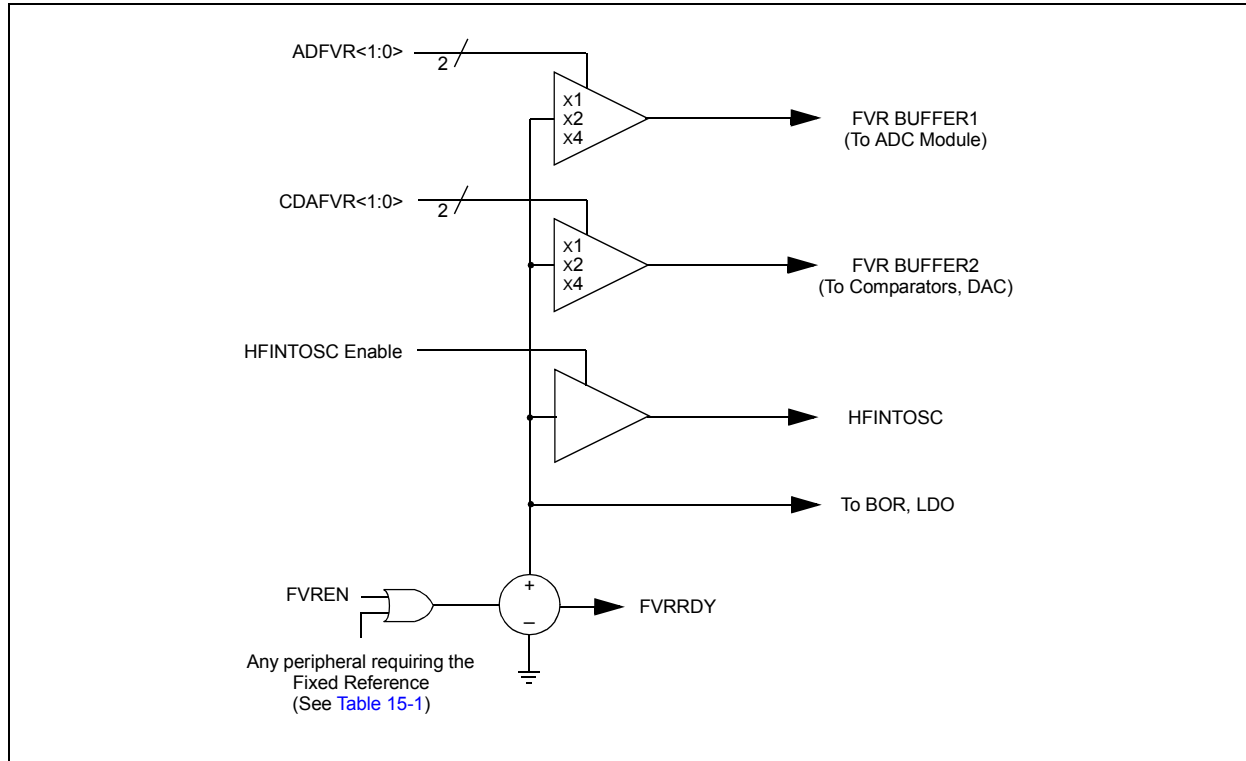


**TABLE 15-1:** **PERIPHERALS REQUIRING THE FIXED VOLTAGE REFERENCE (FVR)**

| Peripheral | Conditions | Description |
|---|---|---|
| HFINTOSC | FOSC<2:0> = 100 and IRCF<3:0> ≠ 000x | INTOSC is active and device is not in Sleep |
| BOR | BOREN<1:0> = 11 | BOR always enabled |
| | BOREN<1:0> = 10 and BORFS = 1 | BOR disabled in Sleep mode, BOR Fast Start enabled. |
| | BOREN<1:0> = 01 and BORFS = 1 | BOR under software control, BOR Fast Start enabled |
| LDO | All PIC16F1782/3 devices, when VREGPM = 10 and not in Sleep | The device runs off of the ULP regulator when in Sleep mode. |
| PSMC 64 MHz | PxSRC<1:0> | 64 MHz clock forces HFINTOSC on during Sleep. |

## 15.3    Register Definitions: FVR Control

REGISTER 15-1:    FVRCON: FIXED VOLTAGE REFERENCE CONTROL REGISTER

| R/W-0/0 | R-q/q | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|-------|---------|---------|---------|---------|---------|---------|
| FVREN | FVRRDY[1] | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<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 | q = Value depends on condition |

bit 7        **FVREN:** Fixed Voltage Reference Enable bit
             1 = Fixed Voltage Reference is enabled
             0 = Fixed Voltage Reference is disabled

bit 6        **FVRRDY:** Fixed Voltage Reference Ready Flag bit[1]
             1 = Fixed Voltage Reference output is ready for use
             0 = Fixed Voltage Reference output is not ready or not enabled

bit 5        **TSEN:** Temperature Indicator Enable bit[3]
             1 = Temperature Indicator is enabled
             0 = Temperature Indicator is disabled

bit 4        **TSRNG:** Temperature Indicator Range Selection bit[3]
             1 = $V_{OUT}$ = $V_{DD}$ - 4$V_T$ (High Range)
             0 = $V_{OUT}$ = $V_{DD}$ - 2$V_T$ (Low Range)

bit 3-2      CDAFVR<1:0>: Comparator and DAC Fixed Voltage Reference Selection bit
             11 = Comparator and DAC Fixed Voltage Reference Peripheral output is 4x (4.096V)[2]
             10 = Comparator and DAC Fixed Voltage Reference Peripheral output is 2x (2.048V)[2]
             01 = Comparator and DAC Fixed Voltage Reference Peripheral output is 1x (1.024V)
             00 = Comparator and DAC Fixed Voltage Reference Peripheral output is off.

bit 1-0      **ADFVR<1:0>:** ADC Fixed Voltage Reference Selection bit
             11 = ADC Fixed Voltage Reference Peripheral output is 4x (4.096V)[2]
             10 = ADC Fixed Voltage Reference Peripheral output is 2x (2.048V)[2]
             01 = ADC Fixed Voltage Reference Peripheral output is 1x (1.024V)
             00 = ADC Fixed Voltage Reference Peripheral output is off.

 **Note  1:**   FVRRDY is always '1' on PIC16F1782/3 only.
     **2:**   Fixed Voltage Reference output cannot exceed $V_{DD}$.
     **3:**   See **Section 16.0 "Temperature Indicator Module"** for additional information.

TABLE 15-2:    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 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | 145 |

**Legend:**    Shaded cells are not used with the Fixed Voltage Reference.

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 16.0 TEMPERATURE INDICATOR MODULE

This family of devices is equipped with a temperature circuit designed to measure the operating temperature of the silicon die. The circuit's range of operating temperature falls between -40°C and +85°C. The output is a voltage that is proportional to the device temperature. The output of the temperature indicator is internally connected to the device ADC.

The circuit may be used as a temperature threshold detector or a more accurate temperature indicator, depending on the level of calibration performed. A one-point calibration allows the circuit to indicate a temperature closely surrounding that point. A two-point calibration allows the circuit to sense the entire range of temperature more accurately. Reference Application Note AN1333, "*Use and Calibration of the Internal Temperature Indicator*" (DS01333) for more details regarding the calibration process.

### 16.1 Circuit Operation

Figure 16-1 shows a simplified block diagram of the temperature circuit. The proportional voltage output is achieved by measuring the forward voltage drop across multiple silicon junctions.

Equation 16-1 describes the output characteristics of the temperature indicator.

**EQUATION 16-1:** $V_{OUT}$ **RANGES**

High Range: $V_{OUT} = V_{DD} - 4V_T$

Low Range: $V_{OUT} = V_{DD} - 2V_T$

The temperature sense circuit is integrated with the Fixed Voltage Reference (FVR) module. See **Section 15.0 "Fixed Voltage Reference (FVR)"** for more information.

The circuit is enabled by setting the TSEN bit of the FVRCON register. When disabled, the circuit draws no current.

The circuit operates in either high or low range. The high range, selected by setting the TSRNG bit of the FVRCON register, provides a wider output voltage. This provides more resolution over the temperature range, but may be less consistent from part to part. This range requires a higher bias voltage to operate and thus, a higher $V_{DD}$ is needed.

The low range is selected by clearing the TSRNG bit of the FVRCON register. The low range generates a lower voltage drop and thus, a lower bias voltage is needed to operate the circuit. The low range is provided for low voltage operation.

**FIGURE 16-1:** TEMPERATURE CIRCUIT DIAGRAM
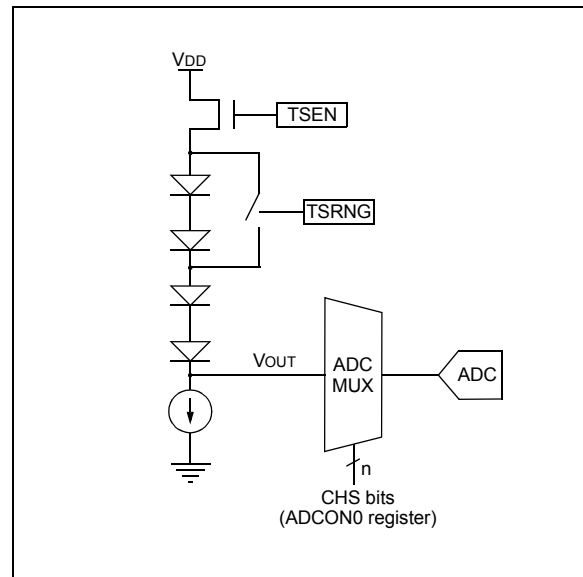


### 16.2 Minimum Operating $V_{DD}$ vs. Minimum Sensing Temperature

When the temperature circuit is operated in low range, the device may be operated at any operating voltage that is within specifications.

When the temperature circuit is operated in high range, the device operating voltage, $V_{DD}$, must be high enough to ensure that the temperature circuit is correctly biased.

Table 16-1 shows the recommended minimum $V_{DD}$ vs. range setting.

**TABLE 16-1:** RECOMMENDED $V_{DD}$ VS. RANGE

| Min. $V_{DD}$, TSRNG = 1 | Min. $V_{DD}$, TSRNG = 0 |
|---|---|
| 3.6V | 1.8V |

### 16.3 Temperature Output

The output of the circuit is measured using the internal Analog-to-Digital Converter. A channel is reserved for the temperature circuit output. Refer to **Section 17.0 "Analog-to-Digital Converter (ADC) Module"** for detailed information.

### 16.4 ADC Acquisition Time

To ensure accurate temperature measurements, the user must wait at least 200 µs after the ADC input multiplexer is connected to the temperature indicator output before the conversion is performed. In addition, the user must wait 200 µs between sequential conversions of the temperature indicator output.

**NOTES:**

**Preliminary**

© 2011-2012 Microchip Technology Inc.

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

The Analog-to-Digital Converter (ADC) allows conversion of a single-ended and differential analog input signals to a 12-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 12-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESH:ADRESL register pair). Figure 17-1 shows the block diagram of the ADC.

The ADC voltage reference is software selectable to be either internally generated or externally supplied.

The ADC can generate an interrupt upon completion of a conversion. This interrupt can be used to wake-up the device from Sleep.

**FIGURE 17-1:     ADC BLOCK DIAGRAM**



Note 1:   When ADON = 0, all multiplexer inputs are disconnected.

2:   See ADCON0 register (Register 17-1) and ADCON2 register (Register 17-3) for detailed analog channel selection per device.
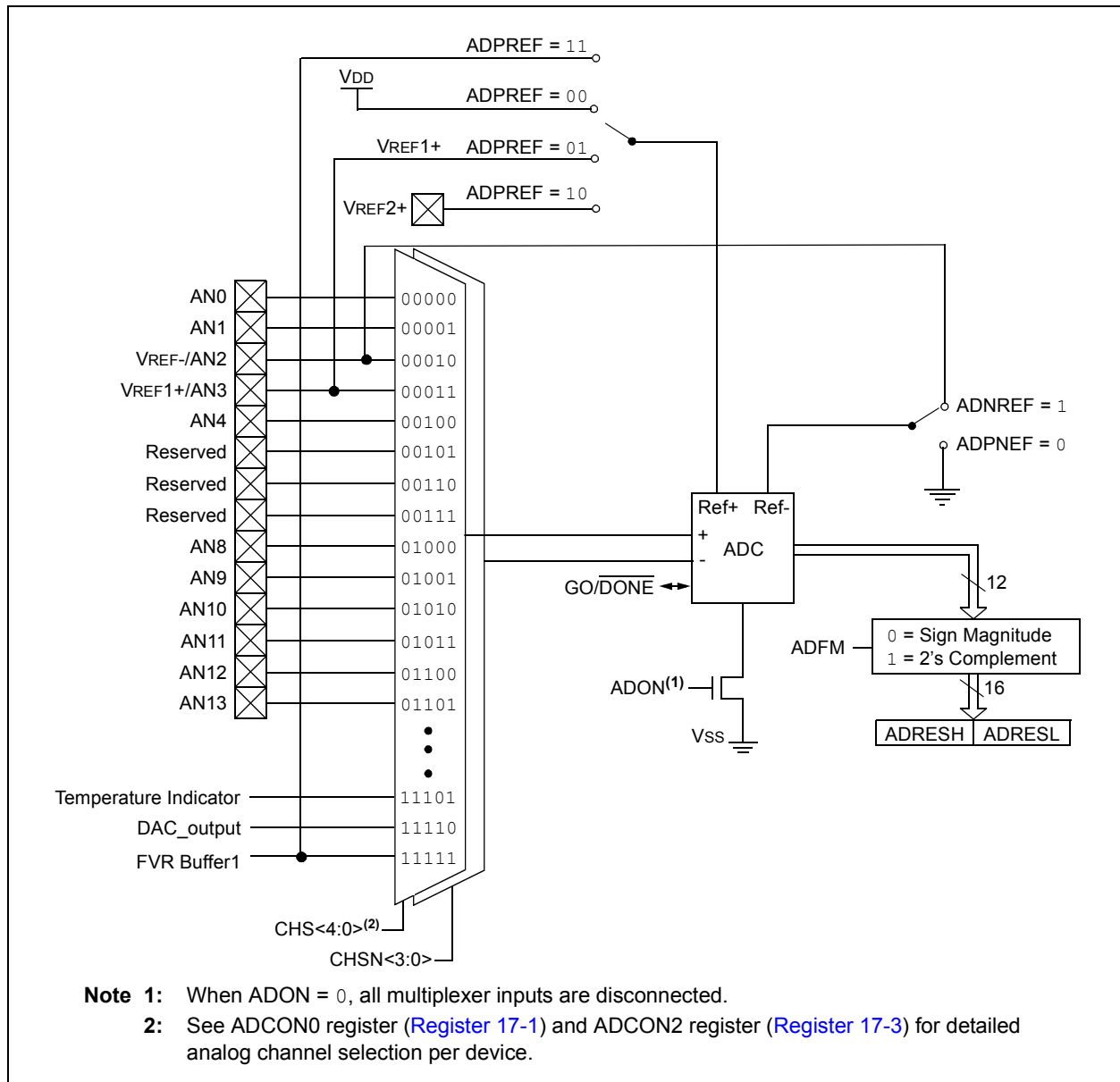
## 17.1 ADC Configuration

When configuring and using the ADC the following functions must be considered:

- Port configuration
- Channel selection
  - Single-ended
  - Differential
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Result formatting

### 17.1.1 PORT CONFIGURATION

The ADC can be used to convert both analog and digital signals. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits. Refer to **Section 13.0 "I/O Ports"** for more information.

> **Note:** Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

### 17.1.2 CHANNEL SELECTION

There are up to 14 channel selections available:

- AN<13:8, 4:0> pins
- Temperature Indicator
- DAC Output
- FVR (Fixed Voltage Reference) Output

Refer to **Section 15.0 "Fixed Voltage Reference (FVR)"** and **Section 16.0 "Temperature Indicator Module"** for more information on these channel selections.

When converting differential signals, the negative input for the channel is selected with the CHSN<3:0> bits of the ADCON2 register. Any positive input can be paired with any negative input to determine the differential channel.

The CHS<4:0> bits of the ADCON0 register determine which positive channel is selected.

When CHSN<3:0> = `1111` then the ADC is effectively a single ended ADC converter.

When changing channels, a delay is required before starting the next conversion.

### 17.1.3 ADC VOLTAGE REFERENCE

The ADPREF bits of the ADCON1 register provide control of the positive voltage reference. The positive voltage reference can be:

- $V_{REF}1+$
- $V_{DD}$
- $V_{REF}2+$
- FVR Buffer1

The ADNREF bits of the ADCON1 register provide control of the negative voltage reference. The negative voltage reference can be:

- $V_{REF}-$ pin
- $V_{SS}$

See **Section 15.0 "Fixed Voltage Reference (FVR)"** for more details on the Fixed Voltage Reference.

### 17.1.4 CONVERSION CLOCK

The source of the conversion clock is software selectable via the ADCS bits of the ADCON1 register. There are seven possible clock options:

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

The time to complete one bit conversion is defined as $T_{AD}$. One full 12-bit conversion requires 15 $T_{AD}$ periods as shown in Figure 17-2.

For correct conversion, the appropriate $T_{AD}$ specification must be met. Refer to the A/D conversion requirements in **Section 30.0 "Electrical Specifications"** for more information. Table 17-1 gives examples of appropriate ADC clock selections.

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

**TABLE 17-1: ADC CLOCK PERIOD (T$_{AD}$) VS. DEVICE OPERATING FREQUENCIES**

| ADC Clock Period (T$_{AD}$) | | Device Frequency (F$_{OSC}$) | | | | | |
|---|---|---|---|---|---|---|---|
| ADC Clock Source | ADCS<2:0> | 32 MHz | 20 MHz | 16 MHz | 8 MHz | 4 MHz | 1 MHz |
| F$_{OSC}$/2 | 000 | 62.5ns[2] | 100 ns[2] | 125 ns[2] | 250 ns[2] | 500 ns[2] | 2.0 μs |
| F$_{OSC}$/4 | 100 | 125 ns[2] | 200 ns[2] | 250 ns[2] | 500 ns[2] | 1.0 μs | 4.0 μs |
| F$_{OSC}$/8 | 001 | 0.5 μs[2] | 400 ns[2] | 0.5 μs[2] | 1.0 μs | 2.0 μs | 8.0 μs[3] |
| F$_{OSC}$/16 | 101 | 800 ns | 800 ns | 1.0 μs | 2.0 μs | 4.0 μs | 16.0 μs[3] |
| F$_{OSC}$/32 | 010 | 1.0 μs | 1.6 μs | 2.0 μs | 4.0 μs | 8.0 μs[3] | 32.0 μs[3] |
| F$_{OSC}$/64 | 110 | 2.0 μs | 3.2 μs | 4.0 μs | 8.0 μs[3] | 16.0 μs[3] | 64.0 μs[3] |
| F$_{RC}$ | x11 | 1.0-6.0 μs[1,4] | 1.0-6.0 μs[1,4] | 1.0-6.0 μs[1,4] | 1.0-6.0 μs[1,4] | 1.0-6.0 μs[1,4] | 1.0-6.0 μs[1,4] |

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

**Note 1:** The F$_{RC}$ source has a typical T$_{AD}$ time of 1.6 μs for V$_{DD}$.

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

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

**4:** The ADC clock period (T$_{AD}$) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock F$_{OSC}$. However, the F$_{RC}$ clock source must be used when conversions are to be performed with the device in Sleep mode.

**FIGURE 17-2: ANALOG-TO-DIGITAL CONVERSION T$_{AD}$ CYCLES**
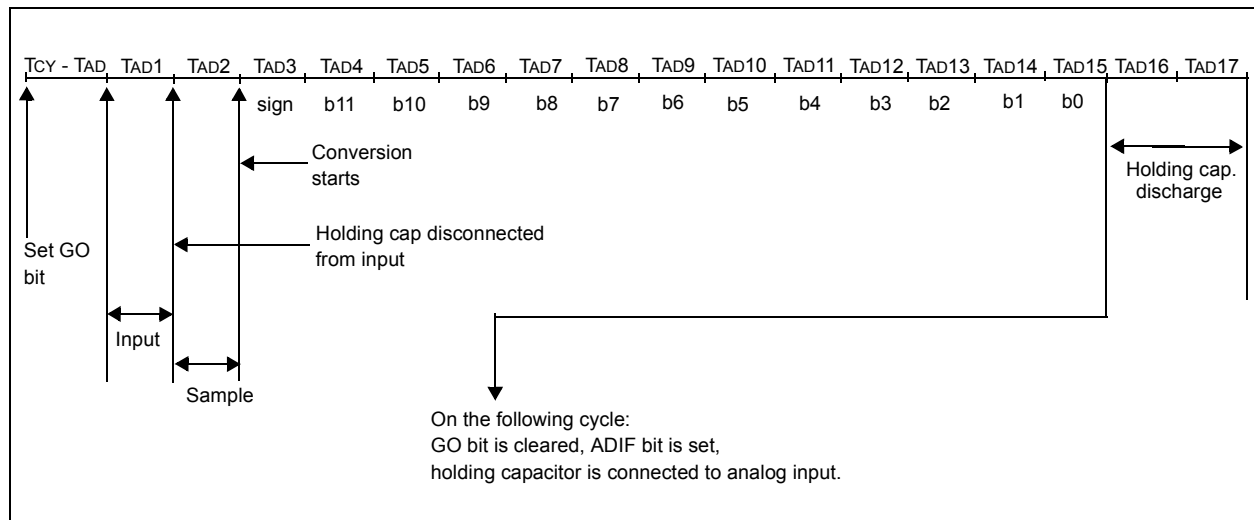
# PIC16(L)F1782/3

## 17.1.5    INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion. The ADC Interrupt Flag is the ADIF bit in the PIR1 register. The ADC Interrupt Enable is the ADIE bit in the PIE1 register. The ADIF bit must be cleared in software.

> **Note 1:** The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled.
>
> **2:** The ADC operates during Sleep only when the FRC oscillator is selected.

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 GIE and PEIE bits of the INTCON register must be disabled. If the GIE and PEIE bits of the INTCON register are enabled, execution will switch to the Interrupt Service Routine.

## 17.1.6    RESULT FORMATTING

The 12-bit A/D conversion result can be supplied in two formats, 2's complement or sign-magnitude. The ADFM bit of the ADCON1 register controls the output format.

Figure 17-3 shows the two output formats.

**FIGURE 17-3:    12-BIT A/D CONVERSION RESULT FORMAT**
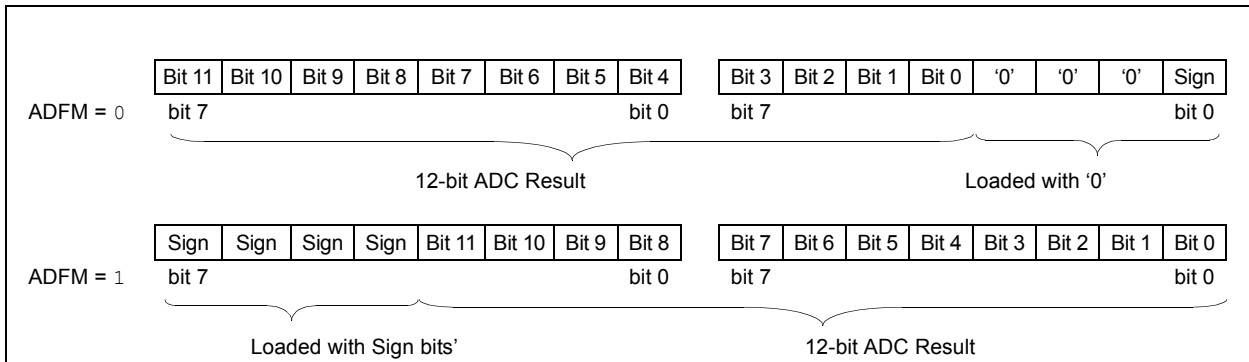


**TABLE 17-2:    ADC OUTPUT RESULTS FORMAT**

| Sign and Magnitude Result ADFM = 0 | | 2's Complement Result ADFM = 1 | |
|---|---|---|---|
| **ADRESH** | **ADRESL** | **ADRESH** | **ADRESL** |
| 1001 0011 | 0011 0000 | 0000 1001 | 0011 0011 |
| 1111 1111 | 1111 0000 | 0000 1111 | 1111 1111 |
| 1111 1111 | 1111 0001 | 1111 0000 | 0000 0001 |
| 0000 0000 | 0001 0001 | 1111 1111 | 1111 1111 |
| **Note:** The raw 13-bits from the ADC is presented in sign and magnitude format, so no data translation is required for sign and magnitude results. | | | |

## 17.2 ADC Operation

### 17.2.1 STARTING A CONVERSION

To enable the ADC module, the ADON bit of the ADCON0 register must be set to a '1'. Setting the GO/$\overline{\text{DONE}}$ bit of the ADCON0 register to a '1' will clear the ADRESH and ADRESL registers and start the Analog-to-Digital conversion.

> **Note:** The GO/$\overline{\text{DONE}}$ bit should not be set in the same instruction that turns on the ADC. Refer to **Section 17.2.6 "A/D Conversion Procedure"**.

### 17.2.2 COMPLETION OF A CONVERSION

When the conversion is complete, the ADC module will:

- Clear the GO/$\overline{\text{DONE}}$ bit
- Set the ADIF Interrupt Flag bit

### 17.2.3 TERMINATING A CONVERSION

If a conversion must be terminated before completion, the GO/$\overline{\text{DONE}}$ bit can be cleared in software. The ADRESH and ADRESL registers will contain the partially complete Analog-to-Digital conversion sample. Results shift into the ADRES registers from LSb to MSb as each bit is converted. Incomplete results remain where left by the shifting process. When the ADRESH bit is clear then the shifted result enters the result registers at the unsigned LS bit, which is ADRESL bit 4.

> **Note:** A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated.

### 17.2.4 ADC OPERATION DURING SLEEP

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the FRC option. When the FRC clock source is selected, the ADC waits one additional instruction before starting the conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled, the device will wake-up from Sleep when the conversion completes. If the ADC interrupt is disabled, the ADC module is turned off after the conversion completes, although the ADON bit remains set.

When the ADC clock source is something other than FRC, a SLEEP instruction causes the present conversion to be aborted and the ADC module is turned off, although the ADON bit remains set.

### 17.2.5 AUTO-CONVERSION TRIGGER

The Auto-conversion Trigger of the CCP module allows periodic ADC measurements without software intervention. When a rising edge of the selected source occurs, the GO/$\overline{\text{DONE}}$ bit is set by hardware and the Timer1 counter resets to zero.

The Auto-conversion Trigger source is selected with the TRIGSEL<3:0> bits of the ADCON2 register.

Using the Auto-conversion Trigger does not assure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met.

Auto-conversion sources are:

- CCP1
- CCP2
- PSMC1
- PSMC2

# PIC16(L)F1782/3

## 17.2.6 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 (Refer to the TRIS register)
   - Configure pin as analog (Refer to the ANSEL register)
2. Configure the ADC module:
   - Select ADC conversion clock
   - Configure voltage reference
   - Select ADC input channel
   - Turn on ADC module
3. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable peripheral interrupt
   - Enable global interrupt[1]
4. Wait the required acquisition time[2].
5. Start conversion by setting the GO/$\overline{DONE}$ bit.
6. Wait for ADC conversion to complete by one of the following:
   - Polling the GO/$\overline{DONE}$ 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:** Refer to **Section 17.4 "A/D Acquisition Requirements"**.

## EXAMPLE 17-1:   A/D CONVERSION

```
;This code block configures the ADC
;for polling, Vdd and Vss references, Frc
;clock and AN0 input.
;
;Conversion start & polling for completion
; are included.
;
BANKSEL    ADCON1       ;
MOVLW      B'11110000'  ;2's complement, Frc
                        ;clock
MOVWF      ADCON1       ;Vdd and Vss Vref
MOVLW      B'00001111'  ;set negative input
MOVWF      ADCON2       ;to negative
                        ;reference
BANKSEL    TRISA        ;
BSF        TRISA,0      ;Set RA0 to input
BANKSEL    ANSEL        ;
BSF        ANSEL,0      ;Set RA0 to analog
BANKSEL    ADCON0       ;
MOVLW      B'00000001'  ;Select channel AN0
MOVWF      ADCON0       ;Turn ADC On
CALL       SampleTime   ;Acquisiton delay
BSF        ADCON0,ADGO  ;Start conversion
BTFSC      ADCON0,ADGO  ;Is conversion done?
GOTO       $-1          ;No, test again
BANKSEL    ADRESH       ;
MOVF       ADRESH,W     ;Read upper 2 bits
MOVWF      RESULTHI     ;store in GPR space
```

**Preliminary**

## 17.3    Register Definitions: ADC Control

**REGISTER 17-1:    ADCON0: A/D CONTROL REGISTER 0**

| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-----|---------|---------|---------|---------|---------|---------|---------|
| — | | | CHS<4:0> | | | GO/$\overline{\text{DONE}}$ | ADON |
| 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 6-2        **CHS<4:0>:** Positive Differential Input Channel Select bits

11111 = FVR (Fixed Voltage Reference) Buffer 1 Output[2]
11110 = DAC_output[1]
11101 = Temperature Indicator[3]
11100 = Reserved. No channel connected.
     •
     •
     •
01110 = Reserved. No channel connected.
01101 = AN13
01100 = AN12
01011 = AN11
01010 = AN10
01001 = AN9
01000 = AN8
00111 = Reserved. No channel connected.
00110 = Reserved. No channel connected.
00101 = Reserved. No channel connected.
00100 = AN4
00011 = AN3
00010 = AN2
00001 = AN1
00000 = AN0

bit 1          **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.
    This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress

bit 0          **ADON:** ADC Enable bit
1 = ADC is enabled
0 = ADC is disabled and consumes no operating current

Note  1:    See **Section 19.0 "Digital-to-Analog Converter (DAC) Module"** for more information.
     2:    See **Section 15.0 "Fixed Voltage Reference (FVR)"** for more information.
     3:    See **Section 16.0 "Temperature Indicator Module"** for more information.

**REGISTER 17-2:     ADCON1: A/D CONTROL REGISTER 1**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|-----|---------|---------|---------|
| ADFM | ADCS<2:0> | | | — | ADNREF | ADPREF<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          **ADFM:** A/D Result Format Select bit (see Figure 17-3)
               1 =  2's complement format.
               0 =  Sign-magnitude result format.

bit 6-4        **ADCS<2:0>:** A/D Conversion Clock Select bits
               111 = $F_{RC}$ (clock supplied from a dedicated RC oscillator)
               110 = $F_{OSC}$/64
               101 = $F_{OSC}$/16
               100 = $F_{OSC}$/4
               011 = $F_{RC}$ (clock supplied from a dedicated RC oscillator)
               010 = $F_{OSC}$/32
               001 = $F_{OSC}$/8
               000 = $F_{OSC}$/2

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

bit 2          **ADNREF:** A/D Negative Voltage Reference Configuration bit
               1 =   $V_{REF}$- is connected to external $V_{REF}$- pin[1]
               0 =   $V_{REF}$- is connected to $V_{SS}$

bit 1-0        **ADPREF<1:0>:** A/D Positive Voltage Reference Configuration bits
               11 = $V_{REF}$+ is connected internally to FVR Buffer 1
               10 = $V_{REF}$+ is connected to $V_{REF}$2+ pin
               01 = $V_{REF}$+ is connected to $V_{REF}$1+ pin
               00 = $V_{REF}$+ is connected to $V_{DD}$

**Note  1:**    When selecting the FVR, $V_{REF}$1+ pin, or $V_{REF}$2+ pin as the source of the positive reference, be aware that
               a minimum voltage specification exists. See **Section 30.0 "Electrical Specifications"** for details.

**REGISTER 17-3: ADCON2: A/D CONTROL REGISTER 2**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TRIGSEL<3:0> | | | | CHSN<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 Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-4      **TRIGSEL<3:0>:** ADC Auto-conversion Trigger Source Selection bits

         1111 = Reserved. Auto-conversion Trigger disabled.
         1111 = Reserved. Auto-conversion Trigger disabled.
         1110 = Reserved. Auto-conversion Trigger disabled.
         1101 = Reserved. Auto-conversion Trigger disabled.
         1100 = Reserved. Auto-conversion Trigger disabled.
         1011 = Reserved. Auto-conversion Trigger disabled.
         1010 = Reserved. Auto-conversion Trigger disabled.
         1001 = PSMC2 Falling Match Event
         1000 = PSMC2 Rising Edge Event
         0111 = PSMC2 Period Edge Event
         0110 = PSMC1 Falling Edge Event
         0101 = PSMC1 Rising Edge Event
         0100 = PSMC1 Period Match Event
         0011 = Reserved. Auto-conversion Trigger disabled.
         0010 = CCP2, Auto-conversion Trigger
         0001 = CCP1, Auto-conversion Trigger
         0000 = Disabled

bit 3-0      **CHSN<3:0>:** Negative Differential Input Channel Select bits

         When ADON = 0, all multiplexer inputs are disconnected.
         1111 = ADC Negative reference - selected by ADNREF
         1110 = Reserved. No channel connected.
         1101 = AN13
         1100 = AN12
         1011 = AN11
         1010 = AN10
         1001 = AN9
         1000 = AN8
         0111 = Reserved. No channel connected.
         0110 = Reserved. No channel connected.
         0101 = Reserved. No channel connected.
         0100 = AN4
         0011 = AN3
         0010 = AN2
         0001 = AN1
         0000 = AN0

# PIC16(L)F1782/3

**REGISTER 17-4:** **ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 0**

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| AD<11:4> | | | | | | | |
| 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 7-0      **AD<11:4>**: ADC Result Register bits
Upper 8 bits of 12-bit conversion result

**REGISTER 17-5:** **ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 0**

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---------|---------|---------|---------|---------|---------|---------|---------|
| AD<3:0> | | | | — | — | — | ADSIGN |
| 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      **AD<3:0>**: ADC Result Register bits
Lower 4 bits of 12-bit conversion result

bit 3-1      **Extended LSb bits**: These are cleared to zero by DC conversion.

bit 0      **ADSIGN**: ADC Result Sign bit

    **Preliminary**    

## REGISTER 17-6: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 1

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---|---|---|---|---|---|---|---|
| ADSIGN | | | | AD<11:8> | | | |
| 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      **ADSIGN**: Extended AD Result Sign bit

bit 3-0      **AD<11:8>**: ADC Result Register bits
Most significant 4 bits of 12-bit conversion result

## REGISTER 17-7: ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 1

| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---|---|---|---|---|---|---|---|
| AD<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 | |

bit 7-0      **AD<7:0>**: ADC Result Register bits
Least significant 8 bits of 12-bit conversion result

# PIC16(L)F1782/3

## 17.4 A/D Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 17-4. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to Figure 17-4. **The maximum recommended impedance for analog sources is 10 k$\Omega$.** As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an A/D acquisition must be done before the conversion can be started. To calculate the minimum acquisition time, Equation 17-1 may be used. This equation assumes that 1/2 LSb error is used (4,096 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

### EQUATION 17-1: ACQUISITION TIME EXAMPLE

*Assumptions:* $\quad$ *Temperature = 50°C and external impedance of 10k$\Omega$ 5.0V VDD*

$$T_{ACQ} = \textit{Amplifier Settling Time} + \textit{Hold Capacitor Charging Time} + \textit{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$
$$= 2\mu s + T_C + [(\textit{Temperature} - 25°C)(0.05\mu s/°C)]$$

*The value for $T_C$ can be approximated with the following equations:*

$$V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) = V_{CHOLD} \qquad ;[1]\ V_{CHOLD}\ \textit{charged to within 1/2 lsb}$$

$$V_{APPLIED}\left(1 - e^{\frac{-Tc}{RC}}\right) = V_{CHOLD} \qquad ;[2]\ V_{CHOLD}\ \textit{charge response to } V_{APPLIED}$$

$$V_{APPLIED}\left(1 - e^{\frac{-Tc}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{(2^{n+1}) - 1}\right) \qquad ;\textit{combining [1] and [2]}$$

*Note: Where n = number of bits of the ADC.*

*Solving for $T_C$:*

$$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln(1/8191)$$
$$= -10pF(1k\Omega + 7k\Omega + 10k\Omega)\ ln(0.000122)$$
$$= 1.62\mu s$$

*Therefore:*
$$T_{ACQ} = 2\mu s + 1.62\mu s + [(50°C - 25°C)(0.05\mu s/°C)]$$
$$= 4.87\mu s$$

---

**Note 1:** The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

**2:** The charge holding capacitor (CHOLD) is not discharged after each conversion.

**3:** Maximum source impedance feeding the input pin should be considered so that the pin leakage does not cause a voltage divider, thereby limiting the absolute accuracy.

---

$\qquad$ **Preliminary** $\qquad$ © 2011-2012 Microchip Technology Inc.

**FIGURE 17-4:** **ANALOG INPUT MODEL**



Legend:
- CHOLD = Sample/Hold Capacitance
- CPIN = Input Capacitance
- I LEAKAGE = Leakage current at the pin due to various junctions
- RIC = Interconnect Resistance
- RSS = Resistance of Sampling Switch
- SS = Sampling Switch
- VT = Threshold Voltage

**Note 1:** Refer to **Section 30.0 "Electrical Specifications"**.
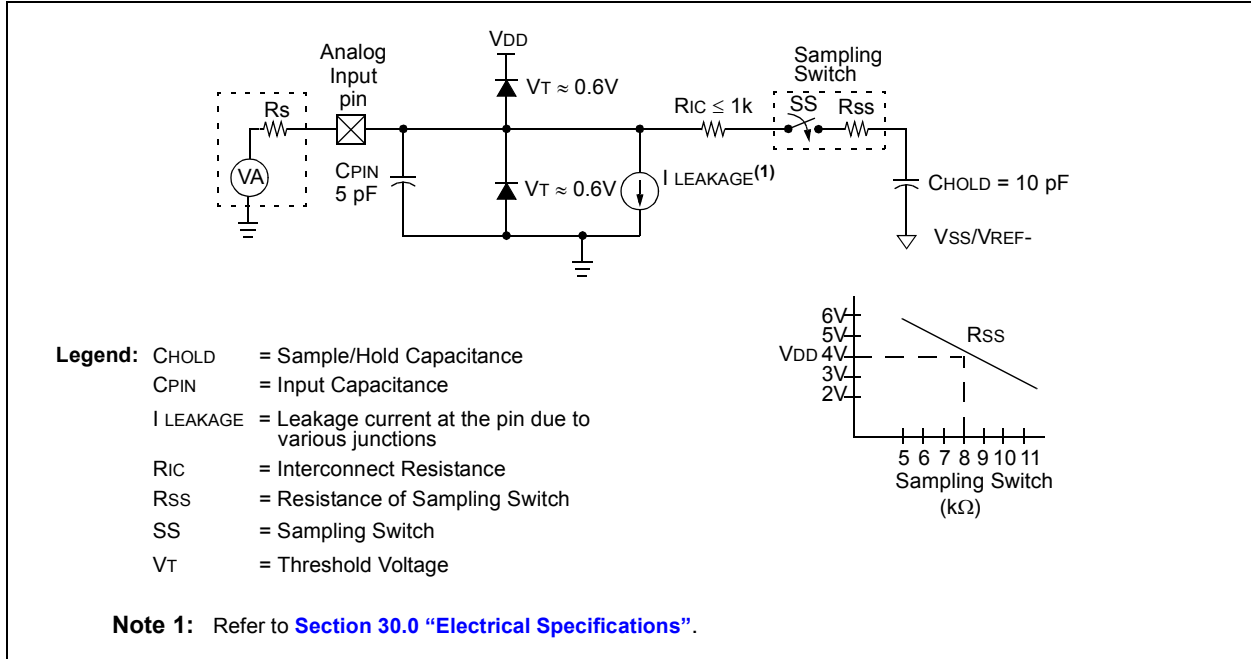
**FIGURE 17-5:** **ADC TRANSFER FUNCTION**
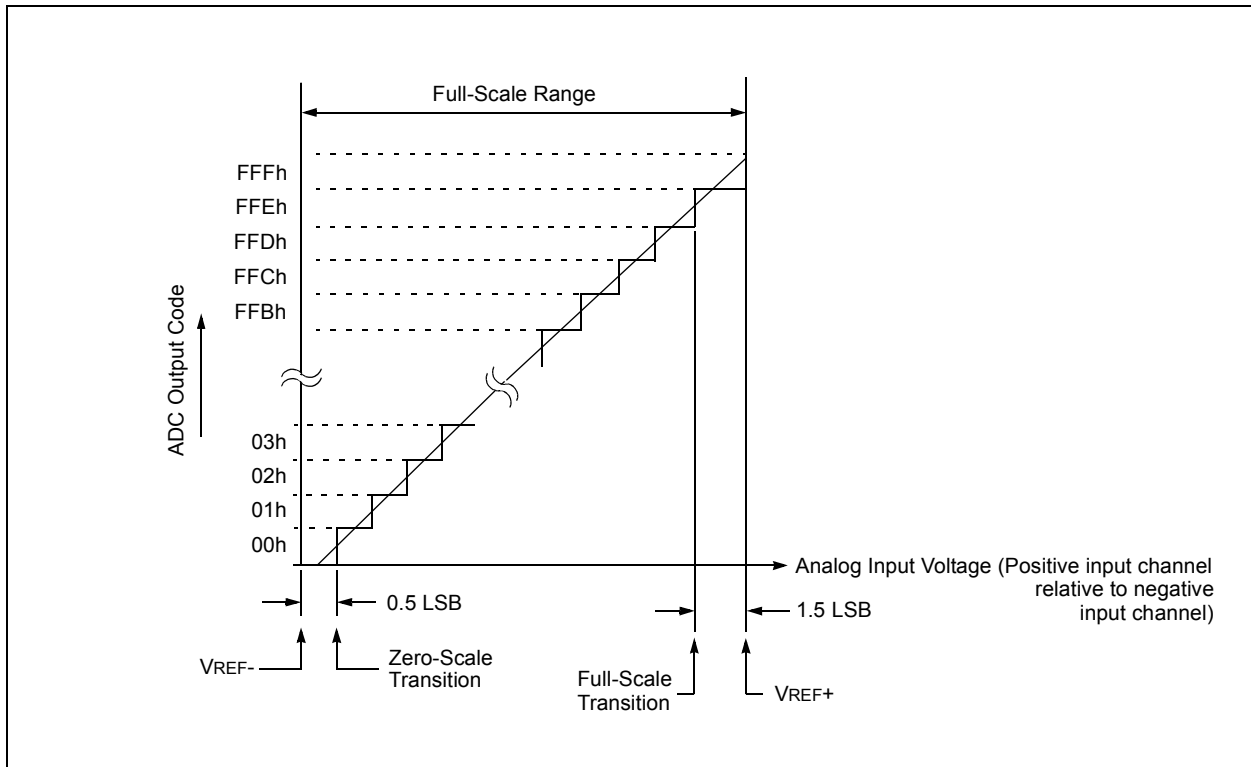
# PIC16(L)F1782/3

**TABLE 17-3:** **SUMMARY OF REGISTERS ASSOCIATED WITH ADC**

| 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/$\overline{\text{DONE}}$ | ADON | 155 |
| ADCON1 | ADFM | ADCS<2:0> | | | — | ADNREF | ADPREF<1:0> | | 156 |
| ADCON2 | TRIGSEL<3:0> | | | | CHSN<3:0> | | | | 157 |
| ADRESH | A/D Result Register High | | | | | | | | 158, 159 |
| ADRESL | A/D Result Register Low | | | | | | | | 158, 159 |
| ANSELA | ANSA7 | — | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 123 |
| ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 129 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 122 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 128 |
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | 145 |

**Legend:** x = unknown, u = unchanged, — = unimplemented read as '0', q = value depends on condition. Shaded cells are not used for ADC module.

**Preliminary** © 2011-2012 Microchip Technology Inc.

# 18.0 OPERATIONAL AMPLIFIER (OPA) MODULES

The Operational Amplifier (OPA) is a standard three-terminal device requiring external feedback to operate. The OPA module has the following features:

- External connections to I/O ports
- Selectable Gain Bandwidth Product
- Low leakage inputs
- Factory Calibrated Input Offset Voltage

**FIGURE 18-1:    OPAx MODULE BLOCK DIAGRAM**
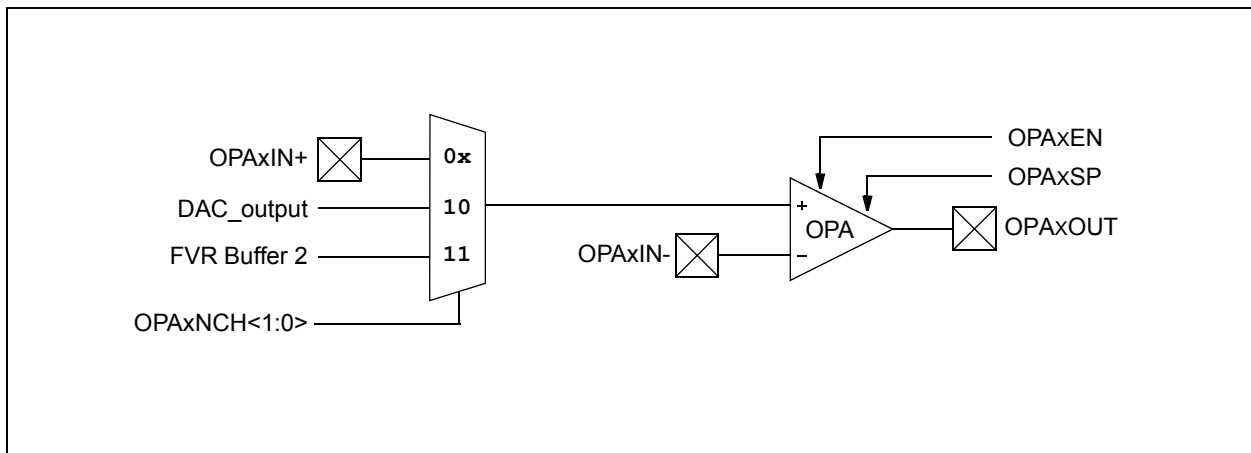
## 18.1 Effects of Reset

A device Reset forces all registers to their Reset state. This disables the OPA module.

## 18.2 OPA Module Performance

Common AC and DC performance specifications for the OPA module:

- Common Mode Voltage Range
- Leakage Current
- Input Offset Voltage
- Open Loop Gain
- Gain Bandwidth Product

**Common mode voltage range** is the specified voltage range for the OPA+ and OPA- inputs, for which the OPA module will perform to within its specifications. The OPA module is designed to operate with input voltages between 0 and VDD-1.4V. Behavior for Common mode voltages greater than VDD-1.4V, or below 0V, are not guaranteed.

**Leakage current** is a measure of the small source or sink currents on the OPA+ and OPA- inputs. To minimize the effect of leakage currents, the effective impedances connected to the OPA+ and OPA- inputs should be kept as small as possible and equal.

**Input offset voltage** is a measure of the voltage difference between the OPA+ and OPA- inputs in a closed loop circuit with the OPA in its linear region. The offset voltage will appear as a DC offset in the output equal to the input offset voltage, multiplied by the gain of the circuit. The input offset voltage is also affected by the Common mode voltage. The OPA is factory calibrated to minimize the input offset voltage of the module.

**Open loop gain** is the ratio of the output voltage to the differential input voltage, (OPA+) - (OPA-). The gain is greatest at DC and falls off with frequency.

**Gain Bandwidth Product** or GBWP is the frequency at which the open loop gain falls off to 0 dB. The lower GBWP is optimized for systems requiring low frequency response and low power consumption.

## 18.3 OPAxCON Control Register

The OPAxCON register, shown in Register 18-1, controls the OPA module.

The OPA module is enabled by setting the OPAxEN bit of the OPAxCON register. When enabled, the OPA forces the output driver of OPAxOUT pin into tri-state to prevent contention between the driver and the OPA output.

The OPAxSP bit of the OPAxCON register controls the power and gain bandwidth of the amplifier. Higher power and greater bandwidth operations are selected by setting the OPAxSP bit. The default is low power reduced bandwidth.

| Note: | When the OPA module is enabled, the OPAxOUT pin is driven by the op amp output, not by the PORT digital driver. Refer to the Electrical specifications for the op amp output drive capability. |
|-------|---|

## 18.4 Register Definitions: Op Amp Control

REGISTER 18-1: OPAxCON: OPERATIONAL AMPLIFIERS (OPAx) CONTROL REGISTERS

| R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---------|---------|-----|-----|-----|-----|---------|---------|
| OPAxEN | OPAxSP | — | — | — | — | OPAxCH<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 | q = Value depends on condition |

bit 7      **OPAxEN:** Op Amp Enable bit

1 = Op amp is enabled
0 = Op amp is disabled and consumes no active power

bit 6      **OPAxSP:** Op Amp Speed/Power Select bit

1 = Comparator operates in high GBWP mode
0 = Comparator operates in low GBWP mode

bit 5-2      **Unimplemented:** Read as '0'

bit 1-0      **OPAxCH<1:0>:** Non-inverting Channel Selection bits

11 = Non-inverting input connects to FVR Buffer 2 output
10 = Non-inverting input connects to DAC_output
0x = Non-inverting input connects to OPAxIN+ pin

TABLE 18-1: SUMMARY OF REGISTERS ASSOCIATED WITH OP AMPS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| ANSELA | ANSA7 | — | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 123 |
| ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 129 |
| DACCON0 | DACEN | — | DACOE1 | DACOE2 | DACPSS<1:0> | | — | DACNSS | 170 |
| DACCON1 | DACR<7:0> | | | | | | | | 170 |
| OPA1CON | OPA1EN | OPA1SP | — | — | — | — | OPA1PCH<1:0> | | 165 |
| OPA2CON | OPA2EN | OPA2SP | — | — | — | — | OPA2PCH<1:0> | | 165 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 122 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 128 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |

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

**NOTES:**

.

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 19.0 DIGITAL-TO-ANALOG CONVERTER (DAC) MODULE

The Digital-to-Analog Converter supplies a variable voltage reference, ratiometric with the input source, with 256 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
- DACOUT1 pin
- DACOUT2 pin

The Digital-to-Analog Converter (DAC) is enabled by setting the DACEN bit of the DACCON0 register.

### EQUATION 19-1: DAC OUTPUT VOLTAGE

$$IF\ DACEN = 1$$

$$V_{OUT} = \left( (V_{SOURCE+} - V_{SOURCE-}) \times \frac{DACR[7:0]}{2^8} \right) + V_{SOURCE-}$$

$$V_{SOURCE+} = V_{DD}, V_{REF}, \text{ or } FVR\ BUFFER\ 2$$

$$V_{SOURCE-} = V_{SS}$$

## 19.2 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 30.0 "Electrical Specifications"**.

## 19.3 DAC Voltage Reference Output

The DAC voltage can be output to the DACOUT1 and DACOUT2 pins by setting the respective DACOE1 and DACOE2 pins of the DACCON0 register. Selecting the DAC reference voltage for output on either DACOUTx pin automatically overrides the digital output buffer and digital input threshold detector functions of that pin. Reading the DACOUTx 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 either DACOUTx pin. Figure 19-2 shows an example buffering technique.

## 19.1 Output Voltage Selection

The DAC has 256 voltage level ranges. The 256 levels are set with the DACR<7:0> bits of the DACCON1 register.

The DAC output voltage is determined by Equation 19-1:

**FIGURE 19-1:** DIGITAL-TO-ANALOG CONVERTER BLOCK DIAGRAM
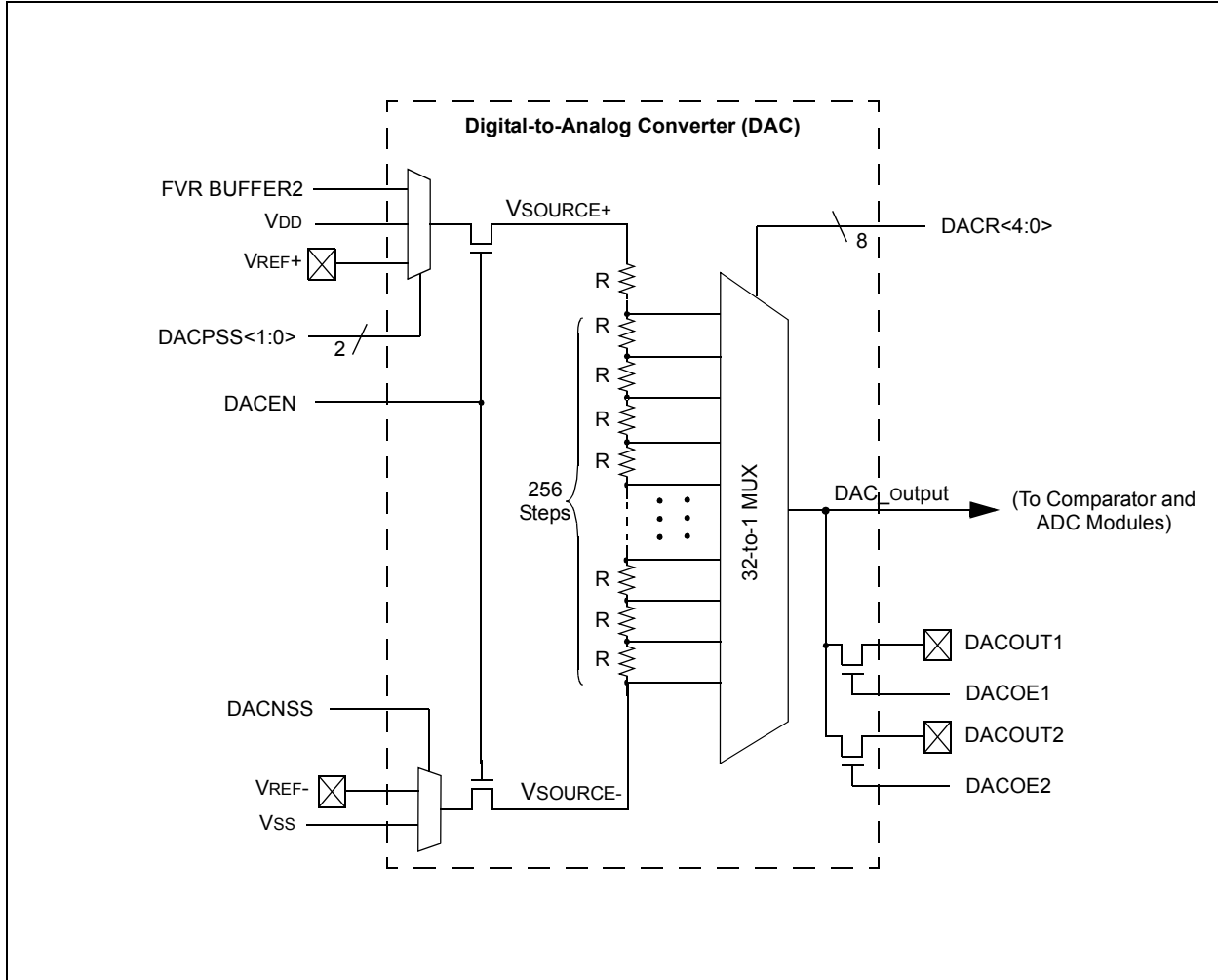


**FIGURE 19-2:** VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE
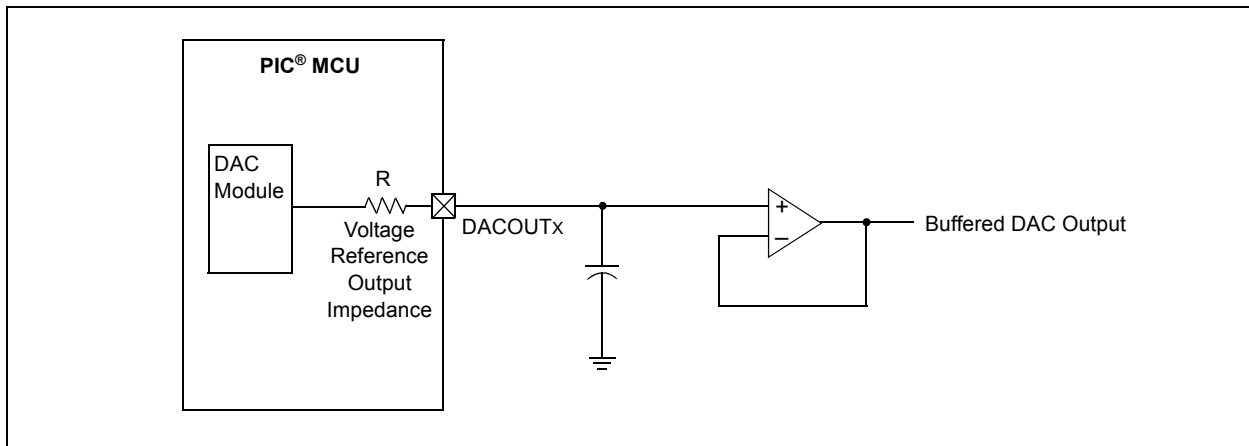
## 19.4 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the DACCON0 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

## 19.5 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.

# PIC16(L)F1782/3

## 19.6 Register Definitions: DAC Control

**REGISTER 19-1: DACCON0: VOLTAGE REFERENCE CONTROL REGISTER 0**

| R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 |
|---------|-----|---------|---------|---------|---------|-----|---------|
| DACEN | — | DACOE1 | DACOE2 | 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      **Unimplemented:** Read as '0'

bit 5      **DACOE1:** DAC Voltage Output 1 Enable bit
                   1 = DAC voltage level is also an output on the DACOUT1 pin
                   0 = DAC voltage level is disconnected from the DACOUT1 pin

bit 4      **DACOE2:** DAC Voltage Output 2 Enable bit
                   1 = DAC voltage level is also an output on the DACOUT2 pin
                   0 = DAC voltage level is disconnected from the DACOUT2 pin

bit 3-2      **DACPSS<1:0>:** DAC Positive Source Select bits
                   11 = Reserved, do not use
                   10 = FVR Buffer2 output
                   01 = V$_{REF}$+ pin
                   00 = V$_{DD}$

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

bit 0      **DACNSS:** DAC Negative Source Select bits
                   1 = V$_{REF}$- pin
                   0 = V$_{SS}$

**REGISTER 19-2: DACCON1: VOLTAGE REFERENCE CONTROL REGISTER 1**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| DACR<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 | |

bit 7-0      **DACR<7:0>:** DAC Voltage Output Select bits

**TABLE 19-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE DAC MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | **145** |
| DACCON0 | DACEN | — | DACOE1 | DACOE2 | DACPSS<1:0> | | — | DACNSS | **170** |
| DACCON1 | DACR<7:0> | | | | | | | | **170** |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used with the DAC module.

     **Preliminary**      © 2011-2012 Microchip Technology Inc.

## 20.0 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. Comparators are very useful mixed signal building blocks because they provide analog functionality independent of 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

### 20.1 Comparator Overview

A single comparator is shown in Figure 20-1 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.

The comparators available for this device are located in Table 20-1.

**TABLE 20-1: COMPARATOR AVAILABILITY PER DEVICE**

| Device | C1 | C2 | C3 |
|---|---|---|---|
| PIC16(L)F1782 | ● | ● | ● |
| PIC16(L)F1783 | ● | ● | ● |

**FIGURE 20-1: SINGLE COMPARATOR**



**Note:** The black areas of the output of the comparator represents the uncertainty due to input offsets and response time.

**FIGURE 20-2:** **COMPARATOR MODULE SIMPLIFIED BLOCK DIAGRAM**



**Note 1:** When CxON = 0, the comparator will produce a '0' at the output.
**2:** When CxON = 0, all multiplexer inputs are disconnected.

**Preliminary**
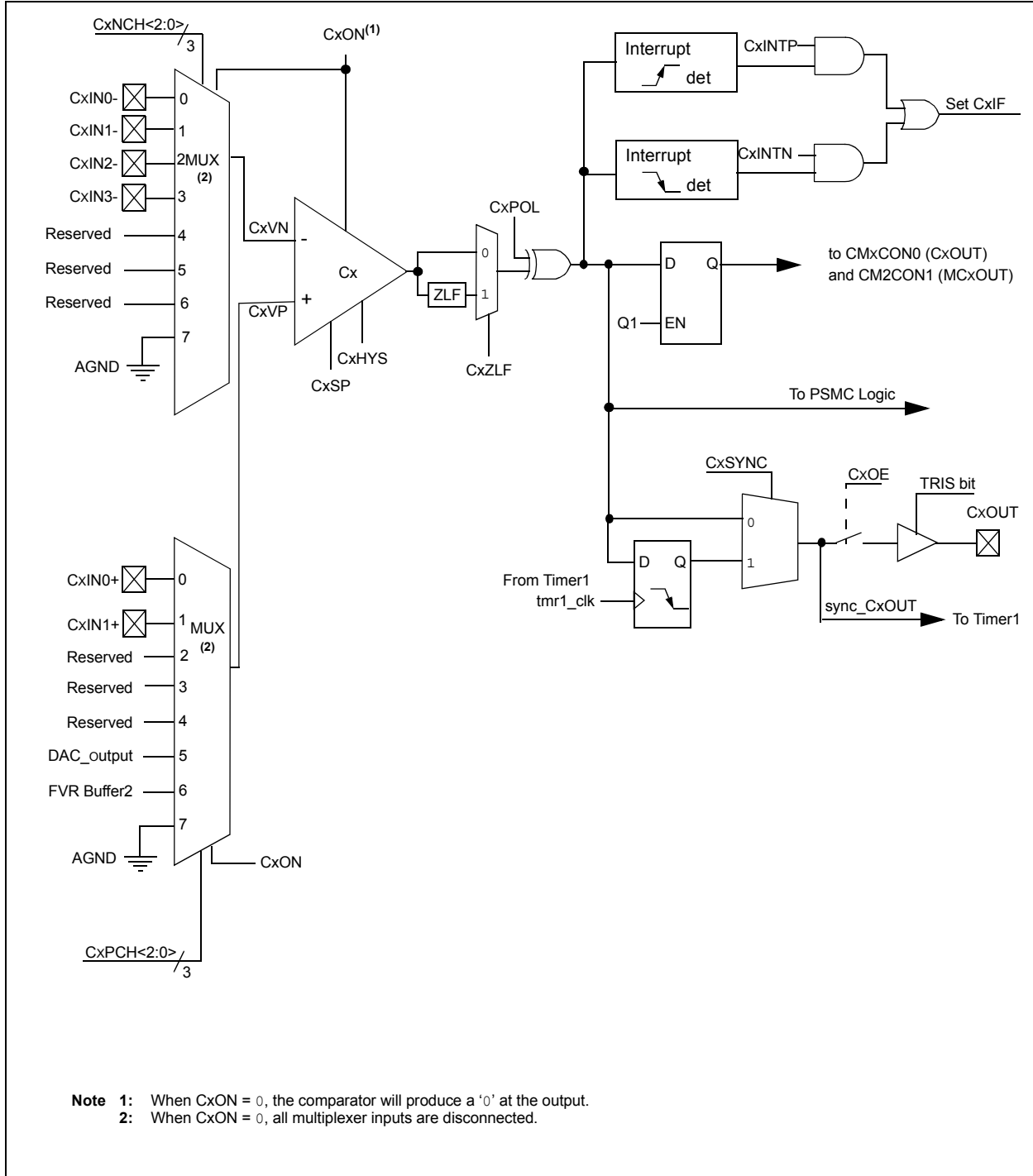
## 20.2 Comparator Control

Each comparator has 2 control registers: CMxCON0 and CMxCON1.

The CMxCON0 registers (see Register 20-1) contain Control and Status bits for the following:

- Enable
- Output selection
- Output polarity
- Speed/Power selection
- Hysteresis enable
- Output synchronization

The CMxCON1 registers (see Register 20-2) contain Control bits for the following:

- Interrupt enable
- Interrupt edge polarity
- Positive input channel selection
- Negative input channel selection

### 20.2.1 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.

### 20.2.2 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 CMOUT 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 of the CMxCON0 register overrides the PORT data latch. Setting the CxON bit of the CMxCON0 register 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.

### 20.2.3 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 20-2 shows the output state versus input conditions, including polarity control.

**TABLE 20-2: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS**

| Input Condition | CxPOL | CxOUT |
|---|---|---|
| $C_xV_N > C_xV_P$ | 0 | 0 |
| $C_xV_N < C_xV_P$ | 0 | 1 |
| $C_xV_N > C_xV_P$ | 1 | 1 |
| $C_xV_N < C_xV_P$ | 1 | 0 |

### 20.2.4 COMPARATOR SPEED/POWER 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'.

## 20.3 Comparator Hysteresis

A selectable amount of separation voltage can be added to the input pins of each comparator to provide a hysteresis function to the overall operation. Hysteresis is enabled by setting the CxHYS bit of the CMxCON0 register.                              .

See **Section 30.0 "Electrical Specifications"** for more information.

## 20.4 Timer1 Gate Operation

The output resulting from a comparator operation can be used as a source for gate control of Timer1. See **Section 22.6 "Timer1 Gate"** for more information. This feature is useful for timing the duration or interval of an analog event.

It is recommended that the comparator output be synchronized to Timer1. This ensures that Timer1 does not increment while a change in the comparator is occurring.

### 20.4.1 COMPARATOR OUTPUT SYNCHRONIZATION

The output from a comparator can be synchronized with Timer1 by setting the CxSYNC bit of the CMxCON0 register.

Once enabled, the comparator output is latched on the falling edge of the Timer1 source clock. If a prescaler is used with Timer1, the comparator output is latched after the prescaling function. To prevent a race condition, the comparator output is latched on the falling edge of the Timer1 clock source and Timer1 increments on the rising edge of its clock source. See the Comparator Block Diagram (Figure 20-2) and the Timer1 Block Diagram (Figure 22-1) for more information.

## 20.5 Comparator Interrupt

An interrupt can be generated upon a change in the output value of the comparator for each comparator, a rising edge detector and a falling edge detector are present.

When either edge detector is triggered and its associated enable bit is set (CxINTP and/or CxINTN bits of the CMxCON1 register), the Corresponding Interrupt Flag bit (CxIF bit of the PIR2 register) will be set.

To enable the interrupt, you must set the following bits:

- CxON, CxPOL and CxSP bits of the CMxCON0 register
- CxIE bit of the PIE2 register
- CxINTP bit of the CMxCON1 register (for a rising edge detection)
- CxINTN bit of the CMxCON1 register (for a falling edge detection)
- PEIE and GIE bits of the INTCON register

The associated interrupt flag bit, CxIF bit of the PIR2 register, must be cleared in software. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

> **Note:** Although a comparator is disabled, an interrupt can be generated by changing the output polarity with the CxPOL bit of the CMxCON0 register, or by switching the comparator on or off with the CxON bit of the CMxCON0 register.

## 20.6 Comparator Positive Input Selection

Configuring the CxPCH<2:0> bits of the CMxCON1 register directs an internal voltage reference or an analog pin to the non-inverting input of the comparator:

- CxIN+ analog pin
- DAC output
- FVR (Fixed Voltage Reference)
- Vss (Ground)

See **Section 15.0 "Fixed Voltage Reference (FVR)"** for more information on the Fixed Voltage Reference module.

See **Section 19.0 "Digital-to-Analog Converter (DAC) Module"** for more information on the DAC input signal.

Any time the comparator is disabled (CxON = 0), all comparator inputs are disabled.

## 20.7 Comparator Negative Input Selection

The CxNCH<2:0> bits of the CMxCON0 register direct one of eight analog pins to the comparator inverting input.

| Note: | To use CxINy+ and CxINy- pins as analog input, the appropriate bits must be set in the ANSEL register and the corresponding TRIS bits must also be set to disable the output drivers. |
|---|---|

## 20.8 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 30.0 "Electrical Specifications"** for more details.

## 20.9 Zero Latency Filter

In high-speed operation, and under proper circuit conditions, it is possible for the comparator output to oscillate. This oscillation can have adverse effects on the hardware and software relying on this signal. Therefore, a digital filter has been added to the comparator output to suppress the comparator output oscillation. Once the comparator output changes, the output is prevented from reversing the change for a nominal time of 20 ns. This allows the comparator output to stabilize without affecting other dependent devices. Refer to Figure 20-3.

**FIGURE 20-3: COMPARATOR ZERO LATENCY FILTER OPERATION**
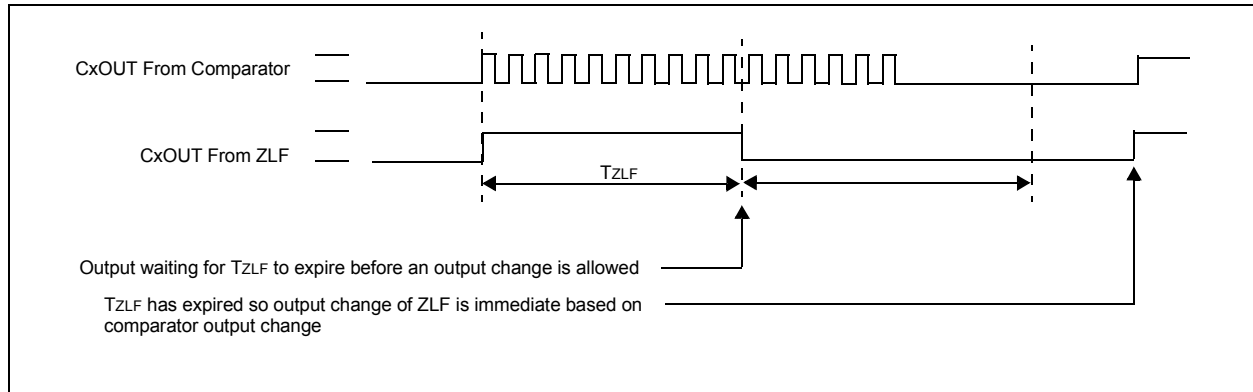
# PIC16(L)F1782/3

## 20.10 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 20-4. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur.

A maximum source impedance of 10 kΩ is recommended for the analog sources. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current to minimize inaccuracies introduced.

---

**Note 1:** When reading a PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert as an analog input, according to the input specification.

**2:** Analog levels on any pin defined as a digital input, may cause the input buffer to consume more current than is specified.

---

### 20.10.1 ALTERNATE PIN LOCATIONS

This module incorporates I/O pins that can be moved to other locations with the use of the alternate pin function register APFCON. To determine which pins can be moved and what their default locations are upon a reset, see **Section 13.1 "Alternate Pin Function"** for more information.

**FIGURE 20-4: ANALOG INPUT MODEL**



Legend:  
CPIN = Input Capacitance  
ILEAKAGE = Leakage Current at the pin due to various junctions  
RIC = Interconnect Resistance  
RS = Source Impedance  
VA = Analog Voltage  
VT = Threshold Voltage

Note 1: See **Section 30.0 "Electrical Specifications"**

**Preliminary**
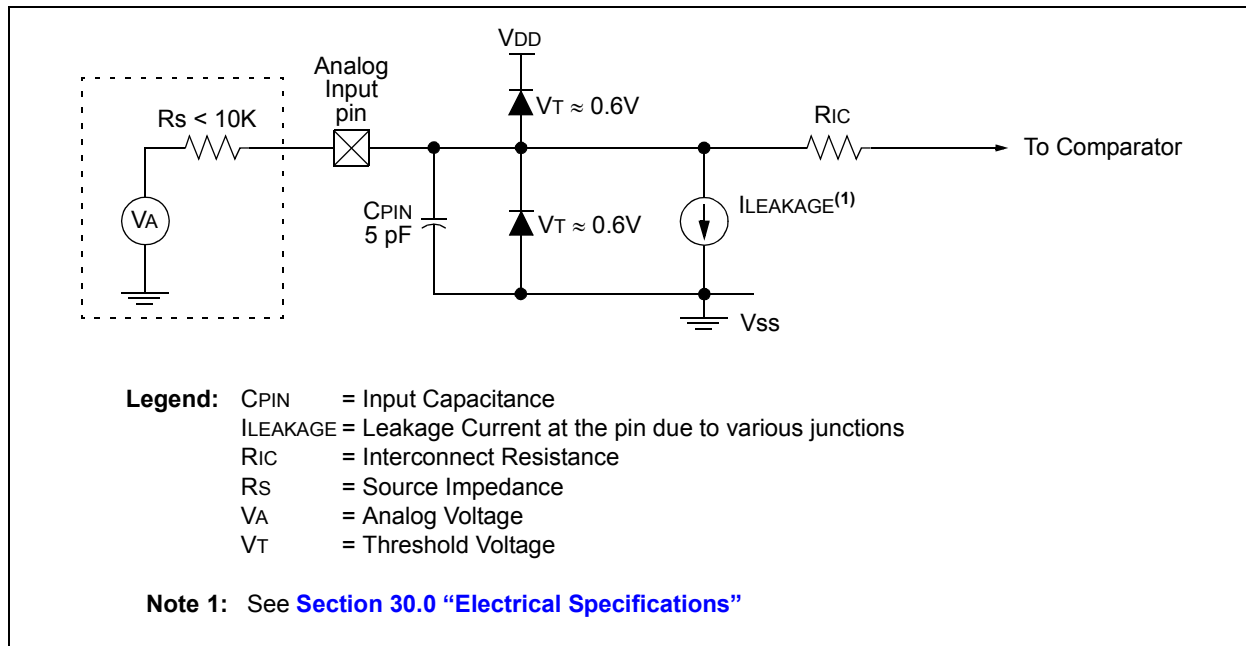
## 20.11 Register Definitions: Comparator Control

**REGISTER 20-1: CMxCON0: COMPARATOR Cx CONTROL REGISTER 0**

| R/W-0/0 | R-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-0/0 |
|---------|-------|---------|---------|---------|---------|---------|---------|
| CxON | CxOUT | CxOE | CxPOL | CxZLF | CxSP | CxHYS | CxSYNC |
| 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 **CxON:** Comparator Enable bit
1 = Comparator is enabled
0 = Comparator is disabled and consumes no active power

bit 6 **CxOUT:** Comparator Output bit
If CxPOL = 1 (inverted polarity):
1 = CxVP < CxVN
0 = CxVP > CxVN
If CxPOL = 0 (non-inverted polarity):
1 = CxVP > CxVN
0 = CxVP < CxVN

bit 5 **CxOE:** Comparator Output Enable bit
1 = CxOUT is present on the CxOUT pin. Requires that the associated TRIS bit be cleared to actually drive the pin. Not affected by CxON.
0 = CxOUT is internal only

bit 4 **CxPOL:** Comparator Output Polarity Select bit
1 = Comparator output is inverted
0 = Comparator output is not inverted

bit 3 **CxZLF:** Comparator Zero Latency Filter Enable bit
1 = Comparator output is filtered
0 = Comparator output is unfiltered

bit 2 **CxSP:** Comparator Speed/Power Select bit
1 = Comparator operates in normal power, higher speed mode
0 = Comparator operates in low-power, low-speed mode

bit 1 **CxHYS:** Comparator Hysteresis Enable bit
1 = Comparator hysteresis enabled
0 = Comparator hysteresis disabled

bit 0 **CxSYNC:** Comparator Output Synchronous Mode bit
1 = Comparator output to Timer1 and I/O pin is synchronous to changes on Timer1 clock source. Output updated on the falling edge of Timer1 clock source.
0 = Comparator output to Timer1 and I/O pin is asynchronous.

**REGISTER 20-2:** **CMxCON1: COMPARATOR Cx CONTROL REGISTER 1**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| CxINTP | CxINTN | CxPCH<2:0> | | | CxNCH<2:0> | | |
| 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      **CxINTP:** Comparator Interrupt on Positive Going Edge Enable bits
         1 = The CxIF interrupt flag will be set upon a positive going edge of the CxOUT bit
         0 = No interrupt flag will be set on a positive going edge of the CxOUT bit

bit 6      **CxINTN:** Comparator Interrupt on Negative Going Edge Enable bits
         1 = The CxIF interrupt flag will be set upon a negative going edge of the CxOUT bit
         0 = No interrupt flag will be set on a negative going edge of the CxOUT bit

bit 5-3      **CxPCH<2:0>:** Comparator Positive Input Channel Select bits
         111 = CxVP connects to AGND
         110 = CxVP connects to FVR Buffer 2
         101 = CxVP connects to $V_{DAC}$
         100 = Reserved, input floating
         011 = Reserved, input floating
         010 = Reserved, input floating
         001 = CxVP connects to CxIN1+ pin
         000 = CxVP connects to CxIN0+ pin

bit 2-0      **CxNCH<2:0>:** Comparator Negative Input Channel Select bits
         111 = CxVN connects to AGND
         110 = CxVN unconnected, input floating
         101 = Reserved, input floating
         100 = Reserved, input floating
         011 = CxVN connects to CxIN3- pin
         010 = CxVN connects to CxIN2- pin
         001 = CxVN connects to CxIN1- pin
         000 = CxVN connects to CxIN0- pin

**REGISTER 20-3:** **CMOUT: COMPARATOR OUTPUT REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | R-0/0 | R-0/0 | R-0/0 |
|-----|-----|-----|-----|-----|-------|-------|-------|
| — | — | — | — | — | MC3OUT | MC2OUT | MC1OUT |
| 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 7-3      **Unimplemented:** Read as '0'
bit 2      **MC3OUT:** Mirror Copy of C3OUT bit
bit 1      **MC2OUT:** Mirror Copy of C2OUT bit
bit 0      **MC1OUT:** Mirror Copy of C1OUT bit

**TABLE 20-3:** **SUMMARY OF 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 | ANSA7 | — | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 123 |
| ANSELB | — | — | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 129 |
| CM1CON0 | C1ON | C1OUT | C1OE | C1POL | C1ZLF | C1SP | C1HYS | C1SYNC | 177 |
| CM2CON0 | C2ON | C2OUT | C2OE | C2POL | C2ZLF | C2SP | C2HYS | C2SYNC | 177 |
| CM1CON1 | C1NTP | C1INTN | C1PCH<2:0> | | | C1NCH<2:0> | | | 178 |
| CM2CON1 | C2NTP | C2INTN | C2PCH<2:0> | | | C2NCH<2:0> | | | 178 |
| CM3CON0 | C3ON | C3OUT | C3OE | C3POL | C3ZLF | C3SP | C3HYS | C3SYNC | 177 |
| CM3CON1 | C3INTP | C3INTN | C3PCH<2:0> | | | C3NCH<2:0> | | | 178 |
| CMOUT | — | — | — | — | — | MC3OUT | MC2OUT | MC1OUT | 178 |
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | 145 |
| DACCON0 | DACEN | — | DACOE1 | DACOE2 | DACPSS<1:0> | | — | DACNSS | 170 |
| DACCON1 | DACR<7:0> | | | | | | | | 170 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE2 | OSFIE | C2IE | C1IE | EEIE | BCLIE | — | C3IE | CCP2IE | 86 |
| PIR2 | OSFIF | C2IF | C1IF | EEIF | BCLIF | — | C3IF | CCP2IF | 89 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 123 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 129 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are unused by the comparator module.

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 21.0    TIMER0 MODULE

The Timer0 module is an 8-bit timer/counter with the following features:

- 8-bit timer/counter register (TMR0)
- 8-bit prescaler (independent of Watchdog Timer)
- Programmable internal or external clock source
- Programmable external clock edge selection
- Interrupt on overflow
- TMR0 can be used to gate Timer1

Figure 21-1 is a block diagram of the Timer0 module.

## 21.1    Timer0 Operation

The Timer0 module can be used as either an 8-bit timer or an 8-bit counter.

### 21.1.1    8-BIT TIMER MODE

The Timer0 module will increment every instruction cycle, if used without a prescaler. 8-bit Timer mode is selected by clearing the TMR0CS bit of the OPTION_REG register.

When TMR0 is written, the increment is inhibited for two instruction cycles immediately following the write.

> **Note:**    The value written to the TMR0 register can be adjusted, in order to account for the two instruction cycle delay when TMR0 is written.

### 21.1.2    8-BIT COUNTER MODE

In 8-Bit Counter mode, the Timer0 module will increment on every rising or falling edge of the T0CKI pin.

8-Bit Counter mode using the T0CKI pin is selected by setting the TMR0CS bit in the OPTION_REG register to '1'.

The rising or falling transition of the incrementing edge for either input source is determined by the TMR0SE bit in the OPTION_REG register.

**FIGURE 21-1:    BLOCK DIAGRAM OF THE TIMER0**
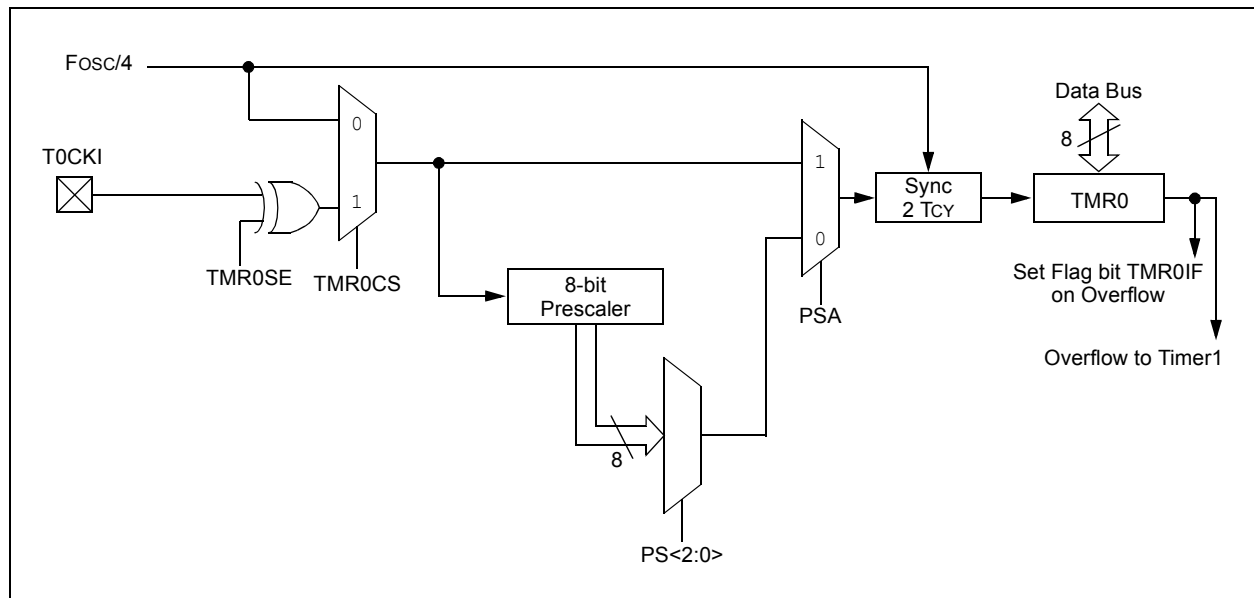
# PIC16(L)F1782/3

### 21.1.3 SOFTWARE PROGRAMMABLE PRESCALER

A software programmable prescaler is available for exclusive use with Timer0. The prescaler is enabled by clearing the PSA bit of the OPTION_REG register.

> **Note:** The Watchdog Timer (WDT) uses its own independent prescaler.

There are 8 prescaler options for the Timer0 module ranging from 1:2 to 1:256. The prescale values are selectable via the PS<2:0> bits of the OPTION_REG register. In order to have a 1:1 prescaler value for the Timer0 module, the prescaler must be disabled by setting the PSA bit of the OPTION_REG register.

The prescaler is not readable or writable. All instructions writing to the TMR0 register will clear the prescaler.

### 21.1.4 TIMER0 INTERRUPT

Timer0 will generate an interrupt when the TMR0 register overflows from FFh to 00h. The TMR0IF interrupt flag bit of the INTCON register is set every time the TMR0 register overflows, regardless of whether or not the Timer0 interrupt is enabled. The TMR0IF bit can only be cleared in software. The Timer0 interrupt enable is the TMR0IE bit of the INTCON register.

> **Note:** The Timer0 interrupt cannot wake the processor from Sleep since the timer is frozen during Sleep.

### 21.1.5 8-BIT COUNTER MODE SYNCHRONIZATION

When in 8-Bit Counter mode, the incrementing edge on the T0CKI pin must be synchronized to the instruction clock. Synchronization can be accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the instruction clock. The high and low periods of the external clocking source must meet the timing requirements as shown in **Section 30.0 "Electrical Specifications"**.

### 21.1.6 OPERATION DURING SLEEP

Timer0 cannot operate while the processor is in Sleep mode. The contents of the TMR0 register will remain unchanged while the processor is in Sleep mode.

## 21.2 Register Definitions: Option Register

### REGISTER 21-1: OPTION_REG: OPTION REGISTER

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| $\overline{\text{WPUEN}}$ | INTEDG | TMR0CS | TMR0SE | PSA | | PS<2:0> | |

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 **WPUEN:** Weak Pull-Up Enable bit

1 = All weak pull-ups are disabled (except $\overline{\text{MCLR}}$, if it is enabled)
0 = Weak pull-ups are enabled by individual WPUx latch values

bit 6 **INTEDG:** Interrupt Edge Select bit

1 = Interrupt on rising edge of INT pin
0 = Interrupt on falling edge of INT pin

bit 5 **TMR0CS:** Timer0 Clock Source Select bit

1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (F$_{OSC}$/4)

bit 4 **TMR0SE:** 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:** Prescaler Assignment bit

1 = Prescaler is not assigned to the Timer0 module
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS<2:0>:** Prescaler Rate Select bits

| Bit Value | Timer0 Rate |
|-----------|-------------|
| 000 | 1 : 2 |
| 001 | 1 : 4 |
| 010 | 1 : 8 |
| 011 | 1 : 16 |
| 100 | 1 : 32 |
| 101 | 1 : 64 |
| 110 | 1 : 128 |
| 111 | 1 : 256 |

### TABLE 21-1: SUMMARY OF 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 | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| OPTION_REG | $\overline{\text{WPUEN}}$ | INTEDG | TMR0CS | TMR0SE | PSA | PS<2:0> | | | 183 |
| TMR0 | Timer0 Module Register | | | | | | | | 181* |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 122 |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used by the Timer0 module.

\* Page provides register information.

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 22.0 TIMER1 MODULE WITH GATE CONTROL

The Timer1 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMR1H:TMR1L)
- Programmable internal or external clock source
- 2-bit prescaler
- Dedicated 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- Time base for the Capture/Compare function
- Auto-conversion Trigger (with CCP)
- Selectable Gate Source Polarity

- Gate Toggle mode
- Gate Single-pulse mode
- Gate Value Status
- Gate Event Interrupt

Figure 22-1 is a block diagram of the Timer1 module.

**FIGURE 22-1:** **TIMER1 BLOCK DIAGRAM**



Note 1: ST Buffer is high speed type when using T1CKI.
2: Timer1 register increments on rising edge.
3: Synchronize does not operate while in Sleep.
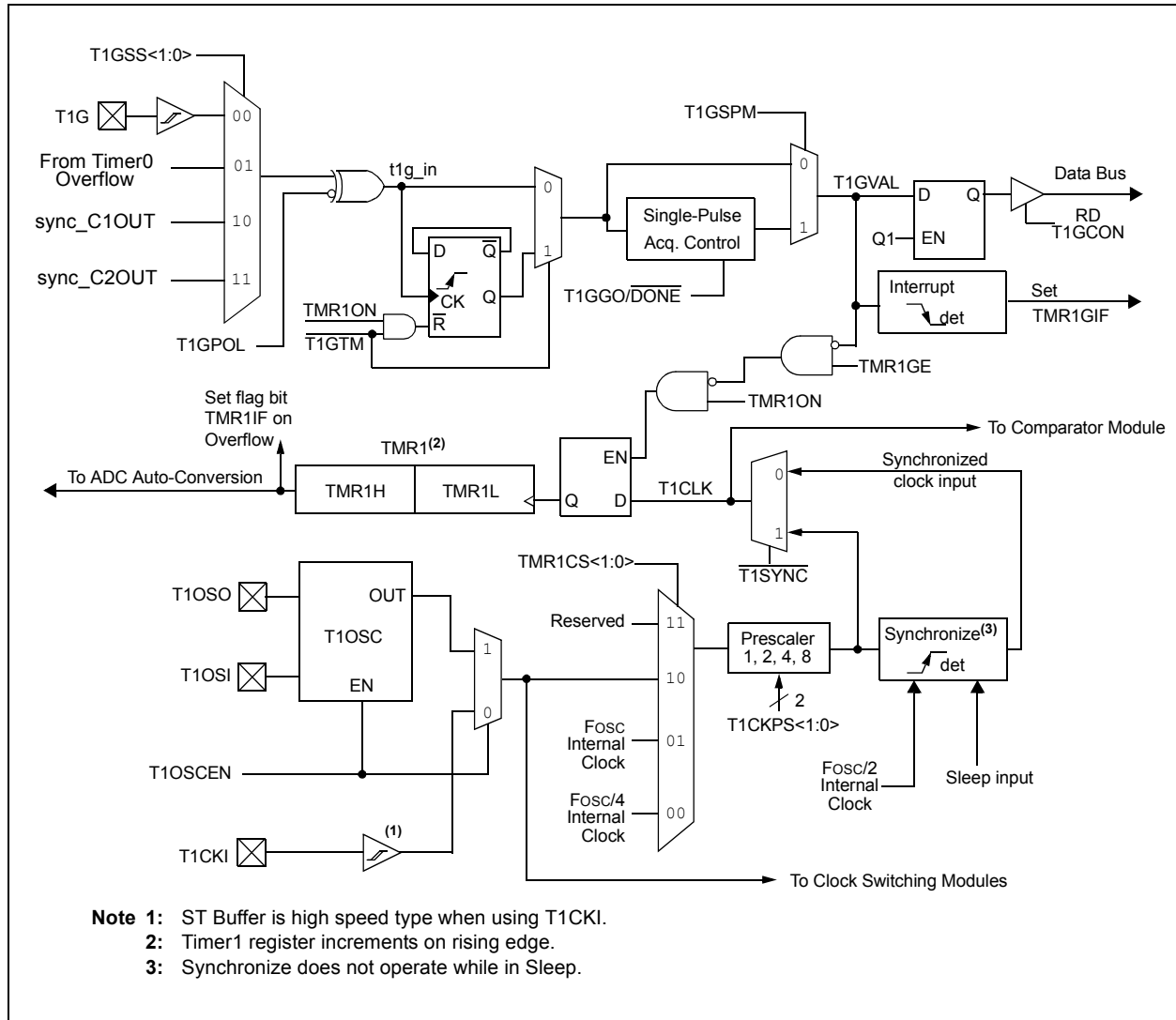
## 22.1 Timer1 Operation

The Timer1 module is a 16-bit incrementing counter which is accessed through the TMR1H:TMR1L register pair. Writes to TMR1H or TMR1L 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 is enabled by configuring the TMR1ON and TMR1GE bits in the T1CON and T1GCON registers, respectively. Table 22-1 displays the Timer1 enable selections.

TABLE 22-1: TIMER1 ENABLE SELECTIONS

| TMR1ON | TMR1GE | Timer1 Operation |
|--------|--------|------------------|
| 0 | 0 | Off |
| 0 | 1 | Off |
| 1 | 0 | Always On |
| 1 | 1 | Count Enabled |

## 22.2 Clock Source Selection

The TMR1CS<1:0> and T1OSCEN bits of the T1CON register are used to select the clock source for Timer1. Table 22-2 displays the clock source selections.

### 22.2.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected, the TMR1H:TMR1L register pair will increment on multiples of $F_{OSC}$ as determined by the Timer1 prescaler.

When the $F_{OSC}$ internal clock source is selected, the Timer1 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 value. To utilize the full resolution of Timer1, an asynchronous input signal must be used to gate the Timer1 clock input.

The following asynchronous sources may be used:

- Asynchronous event on the T1G pin to Timer1 gate
- C1 or C2 comparator input to Timer1 gate

### 22.2.2 EXTERNAL CLOCK SOURCE

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

When enabled to count, Timer1 is incremented on the rising edge of the external clock input T1CKI, which can be synchronized to the microcontroller system clock or 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 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 enabled after POR |
| | • Write to TMR1H or TMR1L |
| | • Timer1 is disabled |
| | • Timer1 is disabled (TMR1ON = 0) when T1CKI is high then Timer1 is enabled (TMR1ON=1) when T1CKI is low. |

TABLE 22-2: CLOCK SOURCE SELECTIONS

| TMR1CS<1:0> | T1OSCEN | Clock Source |
|-------------|---------|--------------|
| 11 | x | LFINTOSC |
| 10 | 0 | External Clocking on T1CKI Pin |
| 01 | x | System Clock ($F_{OSC}$) |
| 00 | x | Instruction Clock ($F_{OSC}$/4) |

## 22.3 Timer1 Prescaler

Timer1 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The T1CKPS bits of the T1CON register control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMR1H or TMR1L.

## 22.4 Timer1 Oscillator

A dedicated low-power 32.768 kHz oscillator circuit is built-in between pins T1OSI (input) and T1OSO (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 T1OSCEN bit of the T1CON register. The oscillator will continue to run during Sleep.

| Note: | The oscillator requires a start-up and stabilization time before use. Thus, T1OSCEN should be set and a suitable delay observed prior to using Timer1. A suitable delay similar to the OST delay can be implemented in software by clearing the TMR1IF bit then presetting the TMR1H:TMR1L register pair to FC00h. The TMR1IF flag will be set when 1024 clock cycles have elapsed, thereby indicating that the oscillator is running and reasonably stable. |
|---|---|

## 22.5 Timer1 Operation in Asynchronous Counter Mode

If the control bit $\overline{\text{T1SYNC}}$ of the T1CON register is set, the external clock input is not synchronized. The timer increments asynchronously to the internal phase clocks. If the 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 22.5.1 "Reading and Writing Timer1 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. |
|---|---|

### 22.5.1 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading TMR1H or TMR1L 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 TMR1H:TMR1L register pair.

## 22.6 Timer1 Gate

Timer1 can be configured to count freely or the count can be enabled and disabled using Timer1 gate circuitry. This is also referred to as Timer1 Gate Enable.

Timer1 gate can also be driven by multiple selectable sources.

### 22.6.1 TIMER1 GATE ENABLE

The Timer1 Gate Enable mode is enabled by setting the TMR1GE bit of the T1GCON register. The polarity of the Timer1 Gate Enable mode is configured using the T1GPOL bit of the T1GCON register.

When Timer1 Gate Enable mode is enabled, Timer1 will increment on the rising edge of the Timer1 clock source. When Timer1 Gate Enable mode is disabled, no incrementing will occur and Timer1 will hold the current count. See Figure 22-3 for timing details.

**TABLE 22-3: TIMER1 GATE ENABLE SELECTIONS**

| T1CLK | T1GPOL | T1G | Timer1 Operation |
|---|---|---|---|
| ↑ | 0 | 0 | Counts |
| ↑ | 0 | 1 | Holds Count |
| ↑ | 1 | 0 | Holds Count |
| ↑ | 1 | 1 | Counts |

# PIC16(L)F1782/3

## 22.6.2 TIMER1 GATE SOURCE SELECTION

Timer1 gate source selections are shown in Table 22-4. Source selection is controlled by the T1GSS bits of the T1GCON register. The polarity for each available source is also selectable. Polarity selection is controlled by the T1GPOL bit of the T1GCON register.

**TABLE 22-4: TIMER1 GATE SOURCES**

| T1GSS | Timer1 Gate Source |
|-------|---------------------|
| 00 | Timer1 Gate Pin |
| 01 | Overflow of Timer0 (TMR0 increments from FFh to 00h) |
| 10 | Comparator 1 Output sync_C1OUT (optionally Timer1 synchronized output) |
| 11 | Comparator 2 Output sync_C2OUT (optionally Timer1 synchronized output) |

### 22.6.2.1 T1G Pin Gate Operation

The T1G pin is one source for Timer1 gate control. It can be used to supply an external source to the Timer1 gate circuitry.

### 22.6.2.2 Timer0 Overflow Gate Operation

When Timer0 increments from FFh to 00h, a low-to-high pulse will automatically be generated and internally supplied to the Timer1 gate circuitry.

### 22.6.2.3 Comparator C1 Gate Operation

The output resulting from a Comparator 1 operation can be selected as a source for Timer1 gate control. The Comparator 1 output (sync_C1OUT) can be synchronized to the Timer1 clock or left asynchronous. For more information see **Section 20.4.1 "Comparator Output Synchronization"**.

### 22.6.2.4 Comparator C2 Gate Operation

The output resulting from a Comparator 2 operation can be selected as a source for Timer1 gate control. The Comparator 2 output (sync_C2OUT) can be synchronized to the Timer1 clock or left asynchronous. For more information see **Section 20.4.1 "Comparator Output Synchronization"**.

### 22.6.3 TIMER1 GATE TOGGLE MODE

When Timer1 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1 gate signal, as opposed to the duration of a single level pulse.

The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 22-4 for timing details.

Timer1 Gate Toggle mode is enabled by setting the T1GTM bit of the T1GCON register. When the T1GTM 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. |
|-------|---|

### 22.6.4 TIMER1 GATE SINGLE-PULSE MODE

When Timer1 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1 Gate Single-Pulse mode is first enabled by setting the T1GSPM bit in the T1GCON register. Next, the T1GGO/$\overline{DONE}$ bit in the T1GCON register must be set. The Timer1 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the T1GGO/$\overline{DONE}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the T1GGO/$\overline{DONE}$ bit is once again set in software. See Figure 22-5 for timing details.

If the Single-Pulse Gate mode is disabled by clearing the T1GSPM bit in the T1GCON register, the T1GGO/$\overline{DONE}$ bit should also be cleared.

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 gate source to be measured. See Figure 22-6 for timing details.

### 22.6.5 TIMER1 GATE VALUE STATUS

When Timer1 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 T1GVAL bit in the T1GCON register. The T1GVAL bit is valid even when the Timer1 gate is not enabled (TMR1GE bit is cleared).

### 22.6.6 TIMER1 GATE EVENT INTERRUPT

When Timer1 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of T1GVAL occurs, the TMR1GIF flag bit in the PIR1 register will be set. If the TMR1GIE bit in the PIE1 register is set, then an interrupt will be recognized.

The TMR1GIF flag bit operates even when the Timer1 gate is not enabled (TMR1GE bit is cleared).

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 22.7 Timer1 Interrupt

The Timer1 register pair (TMR1H:TMR1L) increments to FFFFh and rolls over to 0000h. When Timer1 rolls over, the Timer1 interrupt flag bit of the PIR1 register is set. To enable the interrupt on rollover, you must set these bits:

• TMR1ON bit of the T1CON register
• TMR1IE bit of the PIE1 register
• PEIE bit of the INTCON register
• GIE bit of the INTCON register

The interrupt is cleared by clearing the TMR1IF bit in the Interrupt Service Routine.

> **Note:** The TMR1H:TMR1L register pair and the TMR1IF bit should be cleared before enabling interrupts.

## 22.8 Timer1 Operation During Sleep

Timer1 can only operate during Sleep when setup 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:

• TMR1ON bit of the T1CON register must be set
• TMR1IE bit of the PIE1 register must be set
• PEIE bit of the INTCON register must be set
• $\overline{\text{T1SYNC}}$ bit of the T1CON register must be set
• TMR1CS bits of the T1CON register must be configured
• T1OSCEN bit of the T1CON register must be configured

The device will wake-up on an overflow and execute the next instructions. If the GIE bit of the INTCON register is set, the device will call the Interrupt Service Routine.

Timer1 oscillator will continue to operate in Sleep regardless of the $\overline{\text{T1SYNC}}$ bit setting.

## 22.9 CCP Capture/Compare Time Base

The CCP modules use the TMR1H:TMR1L register pair as the time base when operating in Capture or Compare mode.

In Capture mode, the value in the TMR1H:TMR1L register pair is copied into the CCPR1H:CCPR1L register pair on a configured event.

In Compare mode, an event is triggered when the value CCPR1H:CCPR1L register pair matches the value in the TMR1H:TMR1L register pair. This event can be a Auto-conversion Trigger.

For more information, see **Section 13.0 "I/O Ports"**.

## 22.10 CCP Auto-Conversion Trigger

When any of the CCP's are configured to trigger a auto-conversion, the trigger will clear the TMR1H:TMR1L register pair. This auto-conversion does not cause a Timer1 interrupt. The CCP module may still be configured to generate a CCP interrupt.

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

Timer1 should be synchronized and FOSC/4 should be selected as the clock source in order to utilize the Auto-conversion Trigger. Asynchronous operation of Timer1 can cause a Auto-conversion Trigger to be missed.

In the event that a write to TMR1H or TMR1L coincides with a Auto-conversion Trigger from the CCP, the write will take precedence.

For more information, see **Section 25.2.4 "Auto-Conversion Trigger"**.

**FIGURE 22-2:       TIMER1 INCREMENTING EDGE**



T1CKI = 1
when TMR1
Enabled

T1CKI = 0
when TMR1
Enabled

**Note 1:** Arrows indicate counter increments.

**2:** In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge of the clock.
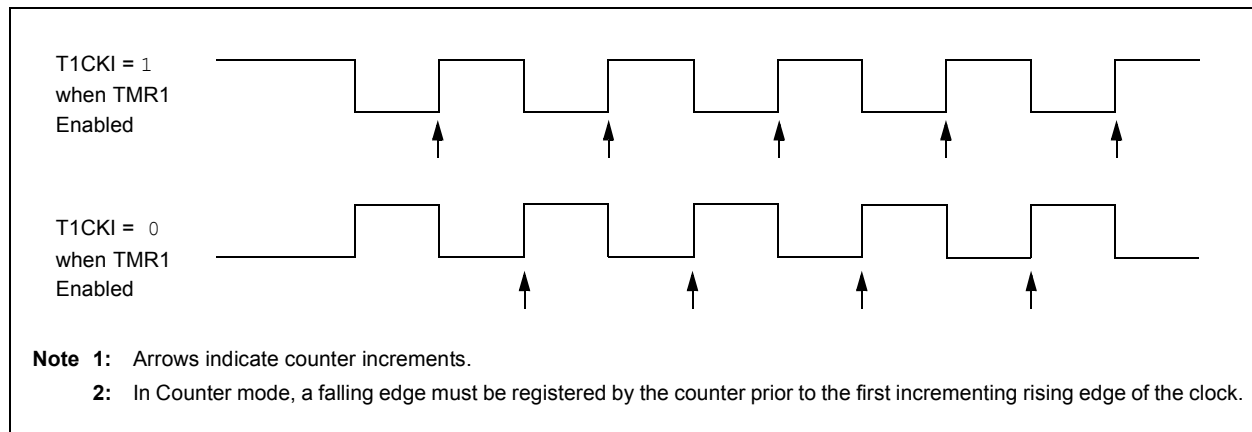
# PIC16(L)F1782/3

**FIGURE 22-3:** **TIMER1 GATE ENABLE MODE**
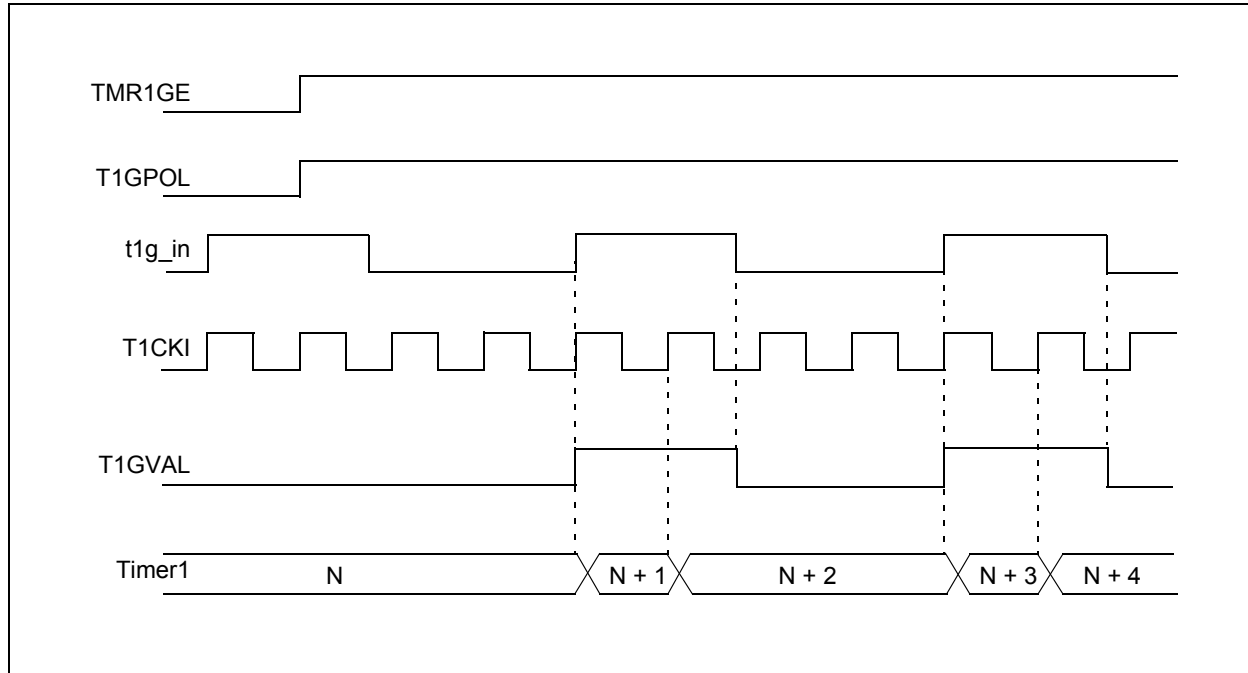


**FIGURE 22-4:** **TIMER1 GATE TOGGLE MODE**

**Preliminary**
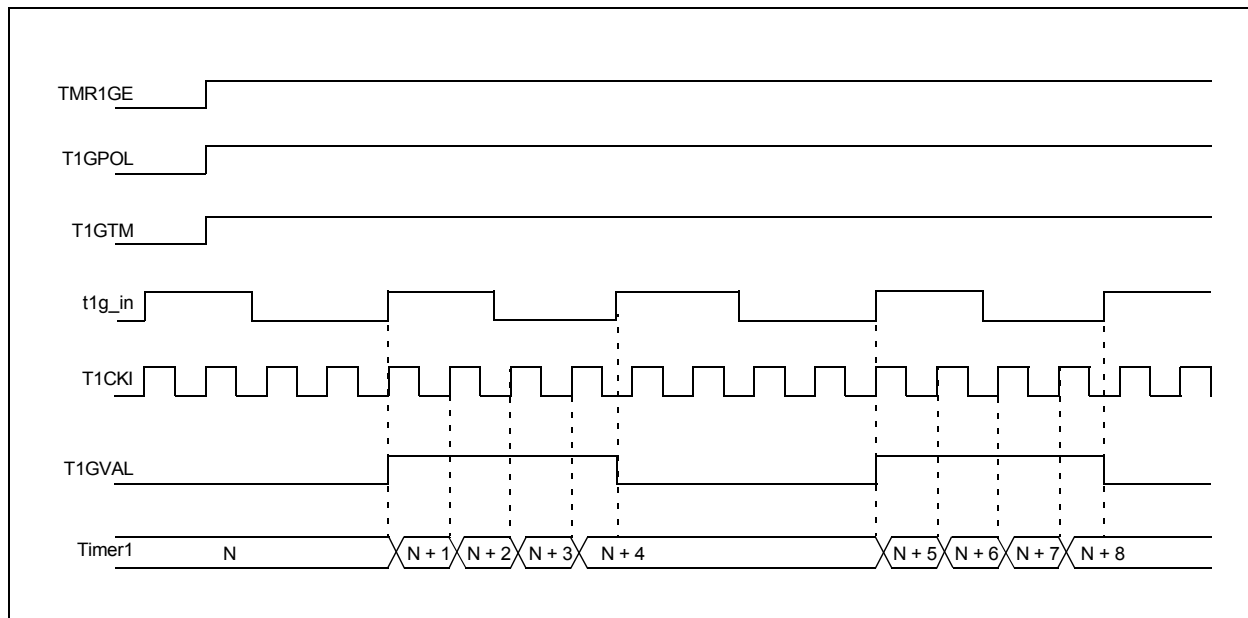
**FIGURE 22-5:** **TIMER1 GATE SINGLE-PULSE MODE**
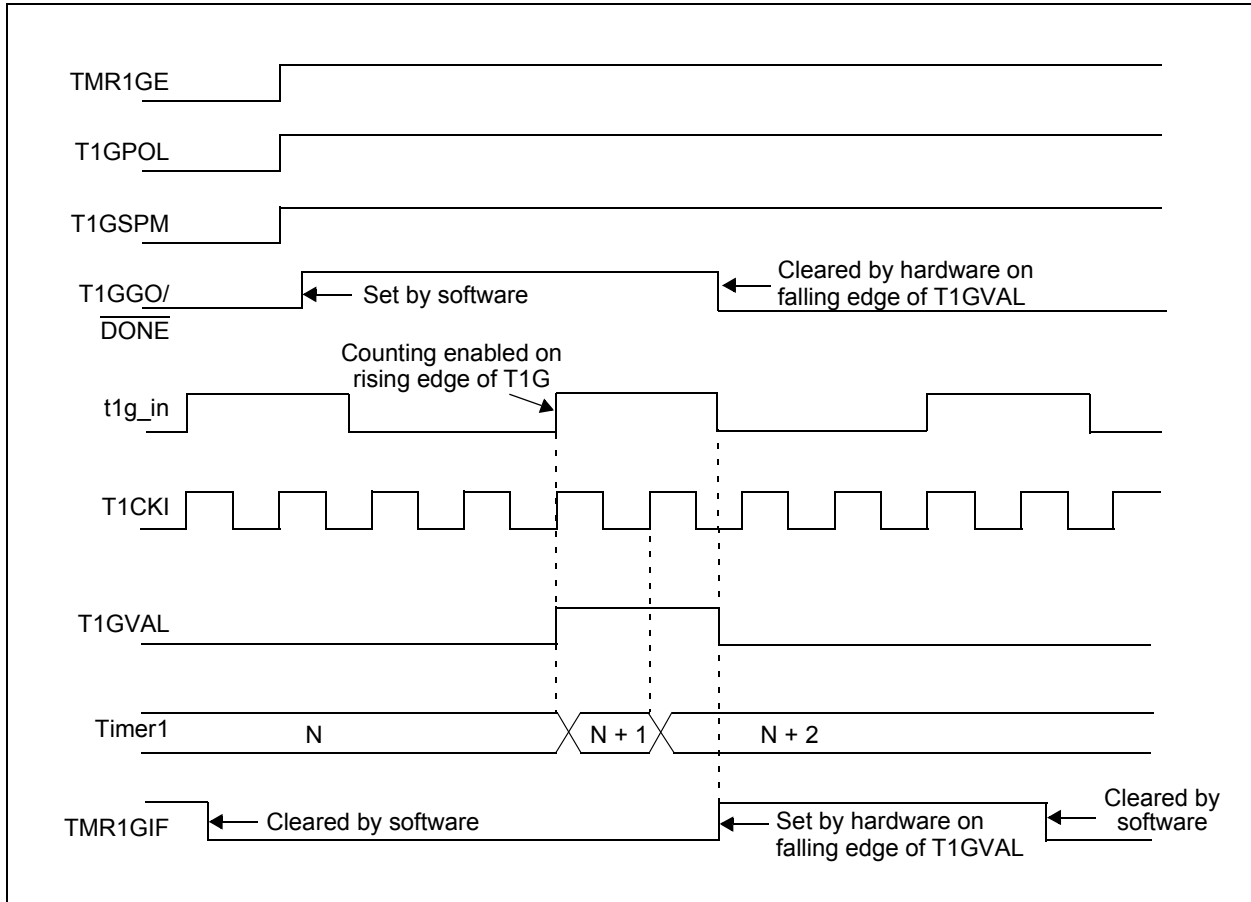
# PIC16(L)F1782/3

**FIGURE 22-6:** **TIMER1 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE**



**Preliminary** © 2011-2012 Microchip Technology Inc.
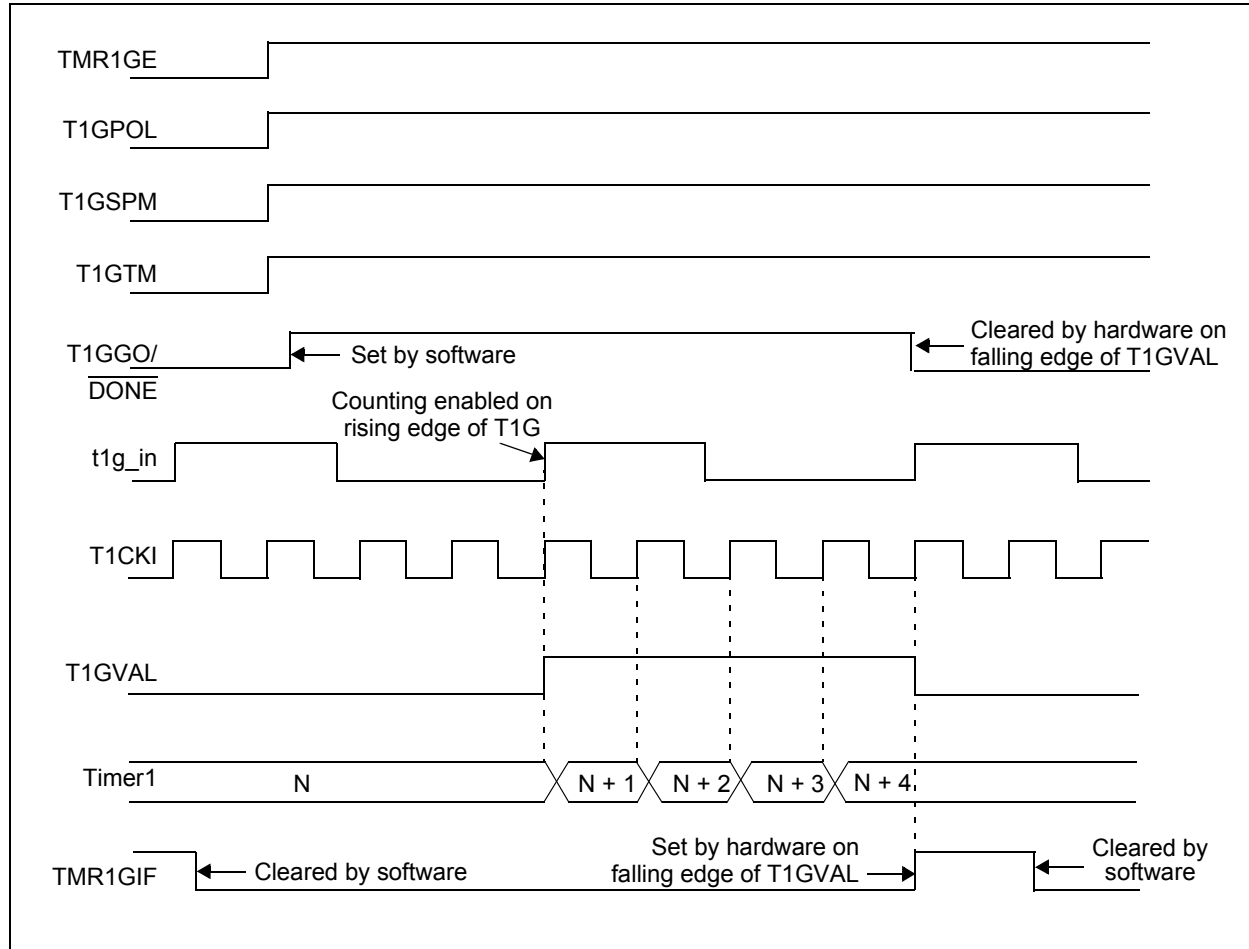
## 22.11 Register Definitions: Timer1 Control

T

**REGISTER 22-1: T1CON: TIMER1 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 | U-0 | R/W-0/u |
|---------|---------|---------|---------|---------|---------|-----|---------|
| TMR1CS<1:0> | | T1CKPS<1:0> | | T1OSCEN | T1SYNC | — | TMR1ON |
| 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 **TMR1CS<1:0>:** Timer1 Clock Source Select bits

11 = Reserved, do not use.
10 = Timer1 clock source is pin or oscillator:
<u>If T1OSCEN = 0</u>:
External clock from T1CKI pin (on the rising edge)
<u>If T1OSCEN = 1</u>:
Crystal oscillator on T1OSI/T1OSO pins
01 = Timer1 clock source is system clock (F$_{OSC}$)
00 = Timer1 clock source is instruction clock (F$_{OSC}$/4)

bit 5-4 **T1CKPS<1:0>:** Timer1 Input Clock Prescale Select bits

11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** LP Oscillator Enable Control bit

1 = Dedicated Timer1 oscillator circuit enabled
0 = Dedicated Timer1 oscillator circuit disabled

bit 2 **T1SYNC:** Timer1 Synchronization Control bit

1 = Do not synchronize asynchronous clock input
0 = Synchronize asynchronous clock input with system clock (F$_{OSC}$)

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

bit 0 **TMR1ON:** Timer1 On bit

1 = Enables Timer1
0 = Stops Timer1 and clears Timer1 gate flip-flop

# PIC16(L)F1782/3

## REGISTER 22-2: T1GCON: TIMER1 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 |
|---------|---------|---------|---------|------------|-------|---------|---------|
| TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/$\overline{DONE}$ | T1GVAL | T1GSS<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　　　　**TMR1GE:** Timer1 Gate Enable bit

If TMR1ON = 0:
This bit is ignored
If TMR1ON = 1:
1 = Timer1 counting is controlled by the Timer1 gate function
0 = Timer1 counts regardless of Timer1 gate function

bit 6　　　　**T1GPOL:** Timer1 Gate Polarity bit

1 = Timer1 gate is active-high (Timer1 counts when gate is high)
0 = Timer1 gate is active-low (Timer1 counts when gate is low)

bit 5　　　　**T1GTM:** Timer1 Gate Toggle Mode bit

1 = Timer1 Gate Toggle mode is enabled
0 = Timer1 Gate Toggle mode is disabled and toggle flip-flop is cleared
Timer1 gate flip-flop toggles on every rising edge.

bit 4　　　　**T1GSPM:** Timer1 Gate Single-Pulse Mode bit

1 = Timer1 Gate Single-Pulse mode is enabled and is controlling Timer1 gate
0 = Timer1 Gate Single-Pulse mode is disabled

bit 3　　　　**T1GGO/$\overline{DONE}$:** Timer1 Gate Single-Pulse Acquisition Status bit

1 = Timer1 gate single-pulse acquisition is ready, waiting for an edge
0 = Timer1 gate single-pulse acquisition has completed or has not been started

bit 2　　　　**T1GVAL:** Timer1 Gate Current State bit

Indicates the current state of the Timer1 gate that could be provided to TMR1H:TMR1L.
Unaffected by Timer1 Gate Enable (TMR1GE).

bit 1-0　　　**T1GSS<1:0>:** Timer1 Gate Source Select bits

11 = Comparator 2 optionally synchronized output (sync_C2OUT)
10 = Comparator 1 optionally synchronized output (sync_C1OUT)
01 = Timer0 overflow output
00 = Timer1 gate pin

**Preliminary**

**TABLE 22-5: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1**

| 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 | 129 |
| CCP1CON | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | | 264 |
| CCP2CON | P2M<1:0> | | DC2B<1:0> | | CCP2M<3:0> | | | | 264 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 88 |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | 185* |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | 185* |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 128 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |
| T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | T1OSCEN | $\overline{\text{T1SYNC}}$ | — | TMR1ON | 193 |
| T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/$\overline{\text{DONE}}$ | T1GVAL | T1GSS<1:0> | | 194 |

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

 * Page provides register information.

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 23.0 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, and 1:64)
- 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 23-1 for a block diagram of Timer2.

**FIGURE 23-1:** **TIMER2 BLOCK DIAGRAM**
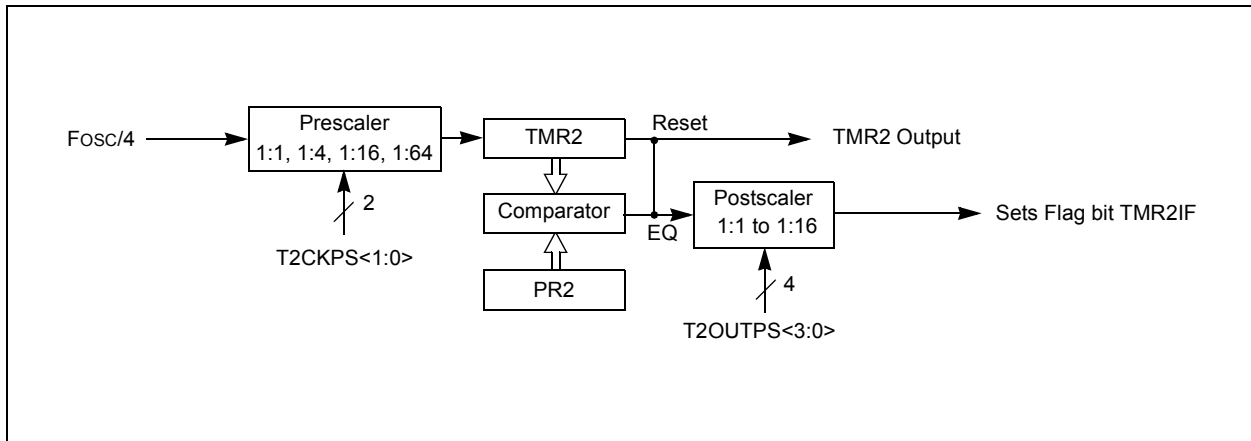
## 23.1 Timer2 Operation

The clock input to the Timer2 modules 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 23.2 "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)
- $\overline{MCLR}$ Reset
- Watchdog Timer (WDT) Reset
- Stack Overflow Reset
- Stack Underflow Reset
- `RESET` Instruction

> **Note:** TMR2 is not cleared when T2CON is written.

## 23.2 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.

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.

## 23.3 Timer2 Output

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

Timer2 can be optionally used as the shift clock source for the MSSP module operating in SPI mode. Additional information is provided in **Section 26.0 "Master Synchronous Serial Port (MSSP) Module"**

## 23.4 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.

## 23.5    Register Definitions: Timer2 Control

**REGISTER 23-1:    T2CON: TIMER2 CONTROL REGISTER**

| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-----|---------|---------|---------|---------|---------|---------|---------|
| — | 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

1111 = 1:16 Postscaler
1110 = 1:15 Postscaler
1101 = 1:14 Postscaler
1100 = 1:13 Postscaler
1011 = 1:12 Postscaler
1010 = 1:11 Postscaler
1001 = 1:10 Postscaler
1000 = 1:9 Postscaler
0111 = 1:8 Postscaler
0110 = 1:7 Postscaler
0101 = 1:6 Postscaler
0100 = 1:5 Postscaler
0011 = 1:4 Postscaler
0010 = 1:3 Postscaler
0001 = 1:2 Postscaler
0000 = 1:1 Postscaler

bit 2          **TMR2ON:** Timer2 On bit

1 =  Timer2 is on
0 =  Timer2 is off

bit 1-0        **T2CKPS<1:0>:** Timer2 Clock Prescale Select bits

11 = Prescaler is 64
10 = Prescaler is 16
01 = Prescaler is 4
00 = Prescaler is 1

**TABLE 23-1:** **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 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| CCP2CON | P2M<1:0> | | DC2B<1:0> | | CCP2M<3:0> | | | | 264 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| PR2 | Timer2 Module Period Register | | | | | | | | 197* |
| T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | 199 |
| TMR2 | Holding Register for the 8-bit TMR2 Register | | | | | | | | 197* |

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

     * Page provides register information.

       **Preliminary**       

## 24.0 PROGRAMMABLE SWITCH MODE CONTROL (PSMC)

The Programmable Switch Mode Controller (PSMC) is a high-performance Pulse Width Modulator (PWM) that can be configured to operate in one of several modes to support single or multiple phase applications.

A simplified block diagram indicating the relationship between inputs, outputs, and controls is shown in Figure 24-1.

This section begins with the fundamental aspects of the PSMC operation. A more detailed description of operation for each mode is located later in **Section 24.3 "Modes of Operation"**

Modes of operation include:

- Single-phase
- Complementary Single-phase
- Push-Pull
- Push-Pull 4-Bridge
- Complementary Push-Pull 4-Bridge
- Pulse Skipping
- Variable Frequency Fixed Duty Cycle
- Complementary Variable Frequency Fixed Duty Cycle
- ECCP Compatible modes
  - Full-Bridge
  - Full-Bridge Reverse
- 3-Phase 6-Step PWM

**FIGURE 24-1:    PSMC SIMPLIFIED BLOCK DIAGRAM**
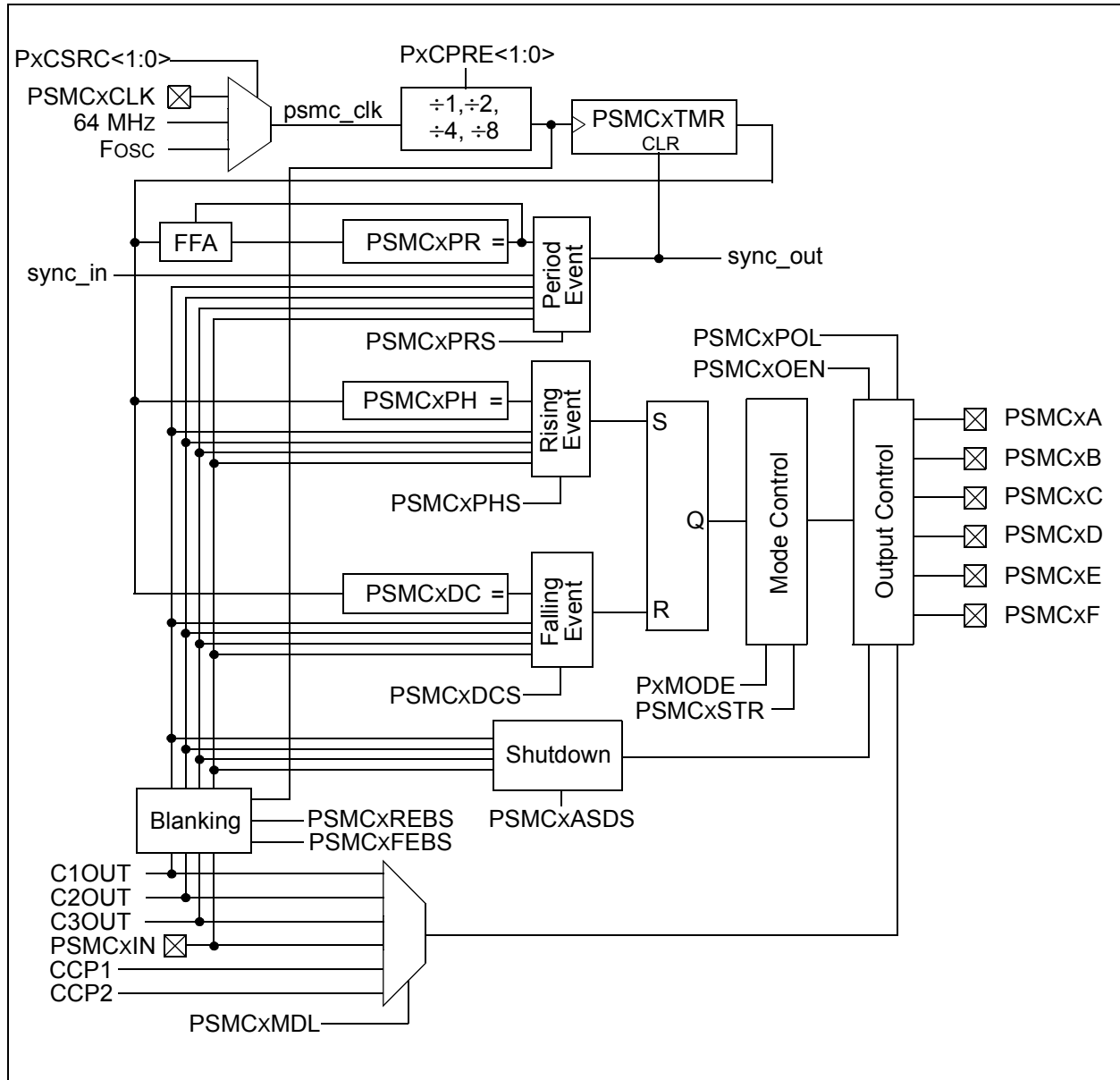
## 24.1 Fundamental Operation

PSMC operation is based on the sequence of three events:

- Period Event – Determines the frequency of the active signal.
- Rising Edge Event – Determines start of the active pulse. This is also referred to as the phase.
- Falling Edge Event – Determines the end of the active pulse. This is also referred to as the duty cycle.

The basic waveform generated from these events is shown in Figure 24-2.

**FIGURE 24-2: BASIC PWM WAVEFORM GENERATION**
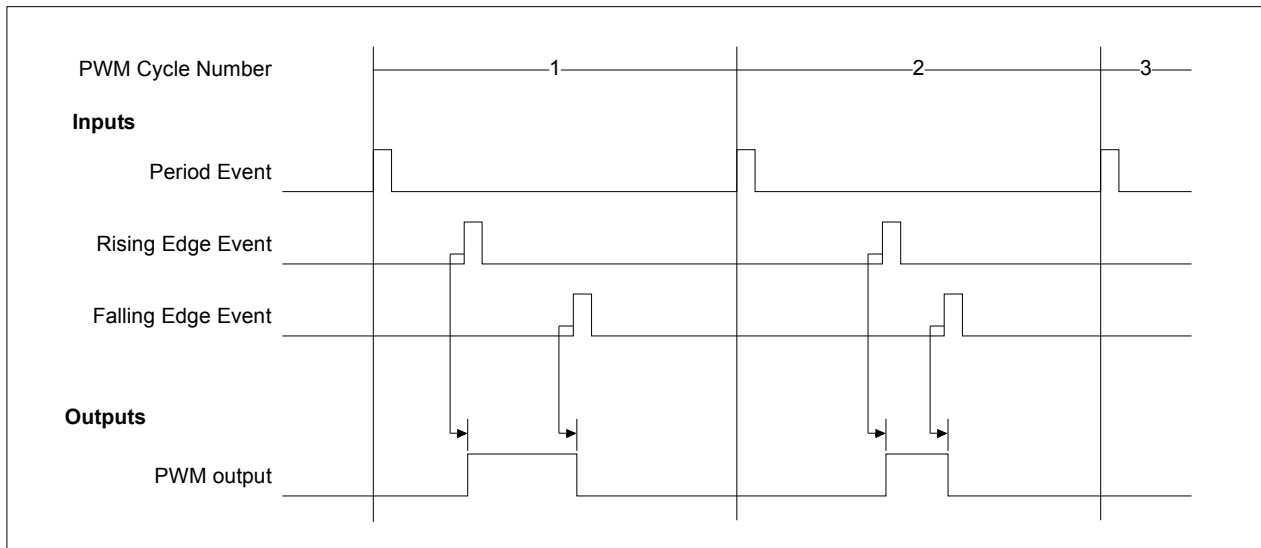


Each of the three types of events is triggered by a user selectable combination of synchronous timed and asynchronous external inputs.

Asynchronous event inputs may come directly from an input pin or through the comparators.

Synchronous timed events are determined from the PSMCxTMR counter, which is derived from internal clock sources. See **Section 24.2.5 "PSMC Time Base Clock Sources"** for more detail.

The active pulse stream can be further modulated by one of several internal or external sources:

- Register control bit
- Comparator output
- CCP output
- Input pin

User selectable deadtime can be inserted in the drive outputs to prevent shoot through of configurations with two devices connected in series between the supply rails.

Applications requiring very small frequency granularity control when the PWM frequency is large can do so with the fractional frequency control available in the variable frequency fixed Duty Cycle modes.

PSMC operation can be quickly terminated without software intervention by the auto-shutdown control. Auto-shutdown can be triggered by any combination of the following:

- PSMCxIN pin
- Comparator 1 output
- Comparator 2 output
- Comparator 3 output

### 24.1.1 PERIOD EVENT

The period event determines the frequency of the active pulse. Period event sources include any combination of the following:

- PSMCxTMR counter match
- PSMC input pin
- Comparator 1 output
- Comparator 2 output
- Comparator 3 output

Period event sources are selected with the PSMC Period Source (PSMCxPRS) register (Register 24-13).

**Section 24.2.1.2 "16-bit Period Register"** contains details on configuring the PSMCxTMR counter match for synchronous period events.

All period events cause the PSMCxTMR counter to reset on the counting clock edge immediately following the period event. The PSMCxTMR counter resumes counting from zero on the counting clock edge after the period event Reset.

During a period, the rising event and falling event are each permitted to occur only once. Subsequent rising or falling events that may occur within the period are suppressed, thereby preventing output chatter from spurious inputs.

### 24.1.2    RISING EDGE EVENT

The rising edge event determines the start of the active drive period. The rising edge event is also referred to as the phase because two synchronized PSMC peripherals may have different rising edge events relative to the period start, thereby creating a phase relationship between the two PSMC peripheral outputs.

Depending on the PSMC mode, one or more of the PSMC outputs will change in immediate response to the rising edge event. Rising edge event sources include any combination of the following:

- Synchronous:
  - PSMCxTMR time base counter match
- Asynchronous:
  - PSMC input pin
  - Comparator 1 output
  - Comparator 2 output
  - Comparator 3 output

Rising edge event sources are selected with the PSMC Phase Source (PSMCxPHS) register (Register 24-11).

For configuring the PSMCxTMR time base counter match for synchronous rising edge events, see **Section 24.2.1.3 "16-bit Phase Register"**.

The first rising edge event in a cycle period is the only one permitted to cause action. All subsequent rising edge events in the same period are suppressed to prevent the PSMC output from chattering in the presence of spurious event inputs. A rising edge event is also suppressed when it occurs after a falling edge event in the same period.

The rising edge event also triggers the start of two other timers when needed: falling edge blanking and dead-band period. For more detail refer to **Section 24.2.8 "Input Blanking"** and **Section 24.4 "Dead-Band Control"**.

When the rising edge event is delayed from the period start, the amount of delay subtracts from the total amount of time available for the drive duty cycle. For example, if the rising edge event is delayed by 10% of the period time, the maximum duty cycle for that period is 90%. A 100% duty cycle is still possible in this example, but duty cycles from 90% to 100% are not possible.

### 24.1.3    FALLING EDGE EVENT

The falling edge event determines the end of the active drive period. The falling edge event is also referred to as the duty cycle because varying the falling edge event, while keeping the rising edge event and period events fixed, varies the active drive duty cycle.

Depending on the PSMC mode, one or more of the PSMC outputs will change in immediate response to the falling edge event. Falling edge event sources include any combination of the following:

- Synchronous:
  - PSMCxTMR time base counter match
- Asynchronous:
  - PSMC input pin
  - Comparator 1 output
  - Comparator 2 output
  - Comparator 3 output

Falling edge event sources are selected with PSMC Duty Cycle Source (PSMCxDCS) register (Register 24-12).

For configuring the PSMCxTMR time base counter match for synchronous falling edge events, see **Section 24.2.1.4 "16-bit Duty Cycle Register"**.

The first falling edge event in a cycle period is the only one permitted to cause action. All subsequent falling edge events in the same period are suppressed to prevent the PSMC output from chattering in the presence of spurious event inputs.

A falling edge event suppresses any subsequent rising edges that may occur in the same period. In other words, if an asynchronous falling event input should come late and occur early in the period, following that for which it was intended, the rising edge in that period will be suppressed. This will have a similar effect as pulse skipping.

The falling edge event also triggers the start of two other timers: rising edge blanking and dead-band period. For more detail refer to **Section 24.2.8 "Input Blanking"** and **Section 24.4 "Dead-Band Control"**.

## 24.2 Event Sources

There are two main sources for the period, rising edge and falling edge events:

- Synchronous input
  - Time base
- Asynchronous Inputs
  - Digital Inputs
  - Analog inputs

### 24.2.1 TIME BASE

The Time Base section consists of several smaller pieces.

- 16-bit time base counter
- 16-bit Period register
- 16-bit Phase register (rising edge event)
- 16-bit Duty Cycle register (falling edge event)
- Clock control
- Interrupt Generator

An example of a fully synchronous PWM waveform generated with the time base is shown in Figure 24-2.

The PSMCxLD bit of the PSMCxCON register is provided to synchronize changes to the event Count registers. Changes are withheld from taking action until the first period event Reset after the PSMCxLD bit is set. For example, to change the PWM frequency, while maintaining the same effective duty cycle, the Period and Duty Cycle registers need to be changed. The changes to all four registers take effect simultaneously on the period event Reset after the after the PSMCxLD bit is set.

#### 24.2.1.1 16-bit Counter (Time Base)

The PSMCxTMR is the counter used as a timing reference for each synchronous PWM period. The counter starts at 0000h and increments to FFFFh on the rising edge of the psmc_clk signal.

When the counter rolls over from FFFFh to 0000h without a period event occurring, the overflow interrupt will be generated, thereby setting the PxTOVIF bit of the PSMC Time Base Interrupt Control (PSMCxINT) register (Register 24-32).

The PSMCxTMR counter is reset on both synchronous and asynchronous period events.

The PSMCxTMR is accessible to software as two 8-bit registers:

- PSMC Time Base Counter Low (PSMCxTMRL) register (Register 24-17)
- PSMC PSMC Time Base Counter High (PSMCxTMRH) register (Register 24-18)

PSMCxTMR is reset to the default POR value when the PSMCxEN bit is cleared.

#### 24.2.1.2 16-bit Period Register

The PSMCxPR Period register is used to determine a synchronous period event referenced to the 16-bit PSMCxTMR digital counter. A match between the PSMCxTMR and PSMCxPR register values will generate a period event.

The match will generate a period match interrupt, thereby setting the PxTPRIF bit of the PSMC Time Base Interrupt Control (PSMCxINT) register (Register 24-32).

The 16-bit period value is accessible to software as two 8-bit registers:

- PSMC Period Count Low Byte (PSMCxPRL) register (Register 24-23)
- PSMC Period Count High Byte (PSMCxPRH) register (Register 24-24)

The 16-bit period value is double-buffered before it is presented to the 16-bit time base for comparison. The buffered registers are updated on the first period event Reset after the PSMCxLD bit of the PSMCxCON register is set.

The synchronous PWM period time can be determined from Equation 24-1.

#### EQUATION 24-1: PWM PERIOD

$$Period = \frac{PSMCxPR[15:0] + 1}{F_{psmc\_clk}}$$

#### 24.2.1.3 16-bit Phase Register

The PSMCxPH Phase register is used to determine a synchronous rising edge event referenced to the 16-bit PSMCxTMR digital counter. A match between the PSMCxTMR and the PSMCxPH register values will generate a rising edge event.

The match will generate a phase match interrupt, thereby setting the PxTPHIF bit of the PSMC Time Base Interrupt Control (PSMCxINT) register (Register 24-32).

The 16-bit phase value is accessible to software as two 8-bit registers:

- PSMC Phase Count Low Byte (PSMCxPHL) register (Register 24-32)
- PSMC Phase Count High Byte (PSMCxPHH) register (Register 24-32)

The 16-bit phase value is double-buffered before it is presented to the 16-bit PSMCxTMR for comparison. The buffered registers are updated on the first period event Reset after the PSMCxLD bit of the PSMCxCON register is set.

### 24.2.1.4    16-bit Duty Cycle Register

The PSMCxDC Duty Cycle register is used to determine a synchronous falling edge event referenced to the 16-bit PSMCxTMR digital counter. A match between the PSMCxTMR and PSMCxDC register values will generate a falling edge event.

The match will generate a duty cycle match interrupt, thereby setting the PxTDCIF bit of the PSMC Time Base Interrupt Control (PSMCxINT) register (Register 24-32).

The 16-bit duty cycle value is accessible to software as two 8-bit registers:

- PSMC Duty Cycle Count Low Byte (PSMCxDCL) register (Register 24-21)
- PSMC Duty Cycle Count High Byte (PSMCxDCH) register (Register 24-22)

The 16-bit duty cycle value is double-buffered before it is presented to the 16-bit time base for comparison. The buffered registers are updated on the first period event Reset after the PSMCxLD bit of the PSMCxCON register is set.

When the period, phase, and duty cycle are all determined from the time base, the effective PWM duty cycle can be expressed as shown in Equation 24-2.

**EQUATION 24-2:    PWM DUTY CYCLE**

$$DUTYCYCLE = \frac{PSMCxDC[15:0] - PSMCxPH[15:0]}{(PSMCxPR[15:0] + 1)}$$

### 24.2.2    0% DUTY CYCLE OPERATION USING TIME BASE

To configure the PWM for 0% duty cycle set PSMCxDC<15:0> = PSMCxPH<15:0>. This will trigger a falling edge event simultaneous with the rising edge event and prevent the PWM from being asserted.

### 24.2.3    100% DUTY CYCLE OPERATION USING TIME BASE

To configure the PWM for 100% duty cycle set PSMCxDC<15:0> > PSMCxPR<15:0>.

This will prevent a falling edge event from occurring as the PSMCxDC<15:0> value and the time base value PSMCxTMR<15:0> will never be equal.

### 24.2.4    TIME BASE INTERRUPT GENERATION

The Time Base section can generate four unique interrupts:

- Time Base Counter Overflow Interrupt
- Time Base Phase Register Match Interrupt
- Time Base Duty Cycle Register Match Interrupt
- Time Base Period Register Match Interrupt

Each interrupt has an interrupt flag bit and an interrupt enable bit. The interrupt flag bit is set anytime a given event occurs, regardless of the status of the enable bit.

Time base interrupt enables and flags are located in the PSMC Time Base Interrupt Control (PSMCxINT) register (Register 24-32).

PSMC time base interrupts also require that the PSMCxTIE bit in the PIE4 register and the PEIE and GIE bits in the INTCON register be set in order to generate an interrupt. The PSMCxTIF interrupt flag in the PIR4 register will only be set by a time base interrupt when one or more of the enable bits in the PSMCxINT register is set.

The interrupt flag bits need to be cleared in software. However, all PMSCx time base interrupt flags, except PSMCxTIF, are cleared when the PSMCxEN bit is cleared.

Interrupt bits that are set by software will generate an interrupt provided that the corresponding interrupt is enabled.

### 24.2.5    PSMC TIME BASE CLOCK SOURCES

There are 3 clock sources available to the module:

- Internal 64 MHz clock
- Fosc system clock
- External clock input pin

The clock source is selected with the PxCSRC<1:0> bits of the PSMCx Clock Control (PSMCxCLK) register (Register 24-5).

When the Internal 64 MHz clock is selected as the source, the HFINTOSC continues to operate and clock the PSMC circuitry in Sleep. However, the system clock to other peripherals and the CPU is suppressed.

The Internal 64 MHz clock utilizes the system clock 4x PLL. When the system clock source is external and the PSMC is using the Internal 64 MHz clock, the 4x PLL should not be used for the system clock.

### 24.2.6    CLOCK PRESCALER

There are four prescaler choices available to be applied to the selected clock:

- Divide by 1
- Divide by 2
- Divide by 4
- Divide by 8

The clock source is selected with the PxCPRE<1:0> bits of the PSMCx Clock Control (PSMCxCLK) register (Register 24-5).

The prescaler output is psmc_clk, which is the clock used by all of the other portions of the PSMC module.

**FIGURE 24-3: TIME BASE WAVEFORM GENERATION**



## 24.2.7 ASYNCHRONOUS INPUTS

The PSMC module supports asynchronous inputs alone or in combination with the synchronous inputs. asynchronous inputs include:

- Analog
  - Comparator 1 output
  - Comparator 2 output
  - Comparator 3 output
- Digital
  - PSMCxIN pin

### 24.2.7.1 Comparator Inputs

The outputs of any combination of the comparators may be used to trigger any of the three events as well as auto-shutdown.

The event triggers on the rising edge of the comparator output. Except for auto-shutdown, the event input is not level sensitive.

### 24.2.7.2 PSMCxIN Pin Input

The PSMCxIN pin may be used to trigger PSMC events. Data is passed through straight to the PSMC module without any synchronization to a system clock. This is so that input blanking may be applied to any external circuit using the module.

The event triggers on the rising edge of the PSMCxIN signal.

## 24.2.8 INPUT BLANKING

Input blanking is a function whereby the inputs from any selected asynchronous input may be driven inactive for a short period of time. This is to prevent electrical transients from the turn-on/off of power components from generating a false event.

Rising edge and falling edge blanking are controlled independently. The following features are available for blanking:

- Blanking mode
- Blanking time counters
- Blanking enable

There is no blanking available for a period event.

The following Blanking modes are available:

- Blanking disabled
- Immediate blanking

The Falling Edge Blanking mode is set with the PxFEBM<1:0> bits of the PSMCx Blanking Control (PSMCxBLNK) register (Register 24-8).

The Rising Edge Blanking mode is set with the PxREBM<1:0> bits of the PSMCx Blanking Control (PSMCxBLNK) register (Register 24-8).

### 24.2.8.1 Blanking Disabled

With blanking disabled, the asynchronous inputs are passed to PSMC module without any intervention.

### 24.2.8.2 Immediate Blanking

With Immediate blanking, a counter is used to determine the blanking period. The desired blanking time is measured in psmc_clk periods. A rising edge event will start incrementing the rising edge blanking counter. A falling edge event will start incrementing the falling edge blanking counter.

The rising edge blanking time is set with the PSMC Rising Edge Blanking Time (PSMCxBLKR) register (Register 24-28). The inputs to be blanked are selected with the PSMC Rising Edge Blanked Source (PSMCxREBS) register (Register 24-9). During rising edge blanking, the selected blanked sources are suppressed for falling edge as well as rising edge, auto-shutdown and period events.

The falling edge blanking time is set with the PSMC Falling Edge Blanking Time (PSMCxBLKF) register (Register 24-29). The inputs to be blanked are selected with the PSMC Falling Edge Blanked Source (PSMCxFEBS) register (Register 24-10). During falling edge blanking, the selected blanked sources are suppressed for rising edge, as well as falling edge, auto-shutdown, and period events.

The blanking counters are incremented on the rising edge of psmc_clk. Blanked sources are suppressed until the counter value equals the blanking time register causing the blanking to terminate.

As the rising and falling edge events are from asynchronous inputs, there may be some uncertainty in the actual blanking time implemented in each cycle. The maximum uncertainty is equal to one psmc_clk period.

## 24.2.9 OUTPUT WAVEFORM GENERATION

The PSMC PWM output waveform is generated based upon the different input events. However, there are several other factors that affect the PWM waveshapes:

- Output Control
  - Output Enable
  - Output Polarity
- Waveform Mode Selection
- Dead-band Control
- Steering control

## 24.2.10 OUTPUT CONTROL

### 24.2.10.1 Output Pin Enable

Each PSMC PWM output pin has individual output enable control.

When the PSMC output enable control is disabled, the module asserts no control over the pin. In this state, the pin can be used for general purpose I/O or other associate peripheral use.

When the PSMC output enable is enabled, the active PWM waveform is applied to the pin per the port priority selection.

PSMC output enable selections are made with the PSMC Output Enable Control (PSMCxOEN) register (Register 24-6).

### 24.2.10.2 Output Steering

PWM output will be presented only on pins for which output steering is enabled. The PSMC has up to 6 PWM outputs. The PWM signal in some modes can be steered to one or more of these outputs.

Steering differs from output enable in the following manner: When the output is enabled but the PWM steering to the corresponding output is not enabled, then general purpose output to the pin is disabled and the pin level will remain constantly in the inactive PWM state. Output steering is controlled with the PSMCS Steering Control 0 (PSMCxSTR0) register (Register 24-30).

Steering operates only in the following modes:

- Single-phase
- Complementary Single-phase
- 3-phase 6-step PWM

### 24.2.10.3 Polarity Control

Each PSMC output has individual output polarity control. Polarity is set with the PSMC Polarity Control (PSMCxPOL) register (Register 24-7).

# PIC16(L)F1782/3

## 24.3    Modes of Operation

All modes of operation use the period, rising edge, and falling edge events to generate the various PWM output waveforms.

The 3-phase 6-step PWM mode makes special use of the software controlled steering to generate the required waveform.

Modes of operation are selected with the PSMC Control (PSMCxCON) register (Register 24-1).

### 24.3.1    SINGLE-PHASE MODE

The single PWM is the most basic of all the waveshapes generated by the PSMC module. It consists of a single output that uses all three events (rising edge, falling edge and period events) to generate the waveform.

#### 24.3.1.1    Mode Features

• No dead-band control available
• PWM can be steered to any combination of the following PSMC outputs:
  - PSMCxA
  - PSMCxB
  - PSMCxC
  - PSMCxD
  - PSMCxE
  - PSMCxF
• Identical PWM waveform is presented to all pins for which steering is enabled.

#### 24.3.1.2    Waveform Generation

Rising Edge Event

• All outputs with PxSTR enabled are set to the active state

Falling Edge Event

• All outputs with PxSTR enabled are set to the inactive state

Code for setting up the PSMC generate the single-phase waveform shown in Figure 24-4, and given in Example 24-1.

EXAMPLE 24-1:    SINGLE-PHASE SETUP

```
; Single-phase PWM PSMC setup
; Fully synchronous operation
; Period = 10 us
; Duty cycle = 50%
    BANKSEL  PSMC1CON
    MOVLW  0x02       ; set period
    MOVWF  PSMC1PRH
    MOVLW  0x7F
    MOVWF  PSMC1PRL
    MOVLW  0x01       ; set duty cycle
    MOVWF  PSMC1DCH
    MOVLW  0x3F
    MOVWF  PSMC1DCL
    CLRF   PSMC1PHH   ; no phase offset
    CLRF   PSMC1PHL
    MOVLW  0x01       ; PSMC clock=64 MHz
    MOVWF  PSMC1CLK
; output on A, normal polarity
    BSF    PSMC1STR0,P1STRA
    BCF    PSMC1POL, P1POLA
    BSF    PSMC1OEN, P1OEA
; set time base as source for all events
    BSF    PSMC1PRS, P1PRST
    BSF    PSMC1PHS, P1PHST
    BSF    PSMC1DCS, P1DCST
; enable PSMC in Single-Phase Mode
; this also loads steering and time buffers
    MOVLW  B'11000000'
    BANKSEL  TRISC
    BCF    TRISC, 0   ; enable pin driver
```

FIGURE 24-4:    SINGLE PWM WAVEFORM - PSMCXSTR0 = 01H

**Preliminary**
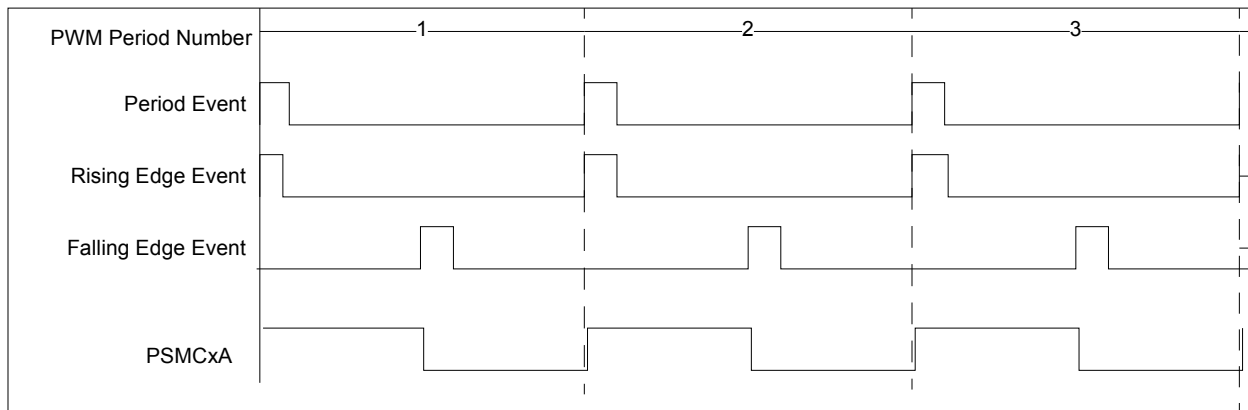
## 24.3.2 COMPLEMENTARY PWM

The complementary PWM uses the same events as the single PWM, but two waveforms are generated instead of only one.

The two waveforms are opposite in polarity to each other. The two waveforms may also have dead-band control as well.

### 24.3.2.1 Mode Features and Controls

• Dead-band control available
• PWM primary output can be steered to the following pins:
  - PSMCxA
  - PSMCxC
  - PSMCxE
• PWM complementary output can be steered to the following pins:
  - PSMCxB
  - PSMCxD
  - PSMCxE

### 24.3.2.2 Waveform Generation

Rising Edge Event

• Complementary output is set inactive
• Optional rising edge dead band is activated
• Primary output is set active

Falling Edge Event

• Primary output is set inactive
• Optional falling edge dead band is activated
• Complementary output is set active

Code for setting up the PSMC generate the complementary single-phase waveform shown in Figure 24-5, and given in Example 24-2.

**EXAMPLE 24-2: COMPLEMENTARY SINGLE-PHASE SETUP**

```
; Complementary Single-phase PWM PSMC setup
; Fully synchronous operation
; Period = 10 us
; Duty cycle = 50%
; Deadband = 93.75 +15.6/-0 ns
    BANKSEL   PSMC1CON
    MOVLW   0x02        ; set period
    MOVWF   PSMC1PRH
    MOVLW   0x7F
    MOVWF   PSMC1PRL
    MOVLW   0x01        ; set duty cycle
    MOVWF   PSMC1DCH
    MOVLW   0x3F
    MOVWF   PSMC1DCL
    CLRF    PSMC1PHH    ; no phase offset
    CLRF    PSMC1PHL
    MOVLW   0x01        ; PSMC clock=64 MHz
    MOVWF   PSMC1CLK
; output on A, normal polarity
    MOVLW   B'00000011'; A and B enables
    MOVWF   PSMC1OEN
    MOVWF   PSMC1STR0
    CLRF    PSMC1POL
; set time base as source for all events
    BSF     PSMC1PRS, P1PRST
    BSF     PSMC1PHS, P1PHST
    BSF     PSMC1DCS, P1DCST
; set rising and falling dead-band times
    MOVLW   D'6'
    MOVWF   PSMC1DBR
    MOVWF   PSMC1DBF
; enable PSMC in Complementary Single Mode
; this also loads steering and time buffers
; and enables rising and falling deadbands
    MOVLW   B'11110001'
    BANKSEL TRISC
    BCF     TRISC, 0   ; enable pin drivers
    BCF     TRISC, 1
```

**FIGURE 24-5: COMPLEMENTARY PWM WAVEFORM – PSMCXSTR0 = 03H**
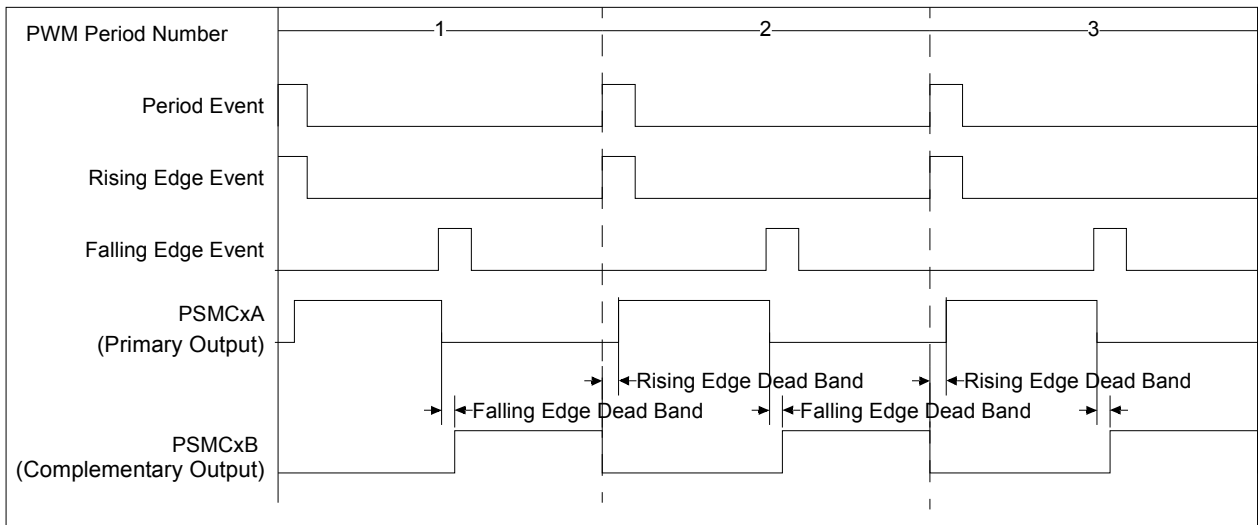
## 24.3.3    PUSH-PULL PWM

The push-pull PWM is used to drive transistor bridge circuits. It uses at least two outputs and generates PWM signals that alternate between the two outputs in even and odd cycles.

Variations of the push-pull waveform include four outputs with two outputs being complementary or two sets of two identical outputs. Refer to Sections 24.3.4 through 24.3.6 for the other Push-Pull modes.

### 24.3.3.1    Mode Features

• No dead-band control available
• No steering control available
• Output is on the following two pins only:
  - PSMCxA
  - PSMCxB

> **Note:** This is a subset of the 6-pin output of the push-pull PWM output, which is why pin functions are fixed in these positions, so they are compatible with that mode. See **Section 24.3.6 "Push-Pull PWM with 4 Full-Bridge and Complementary Outputs"**

### 24.3.3.2    Waveform Generation

Odd numbered period rising edge event:

• PSMCxA is set active

Odd numbered period falling edge event:

• PSMCxA is set inactive

Even numbered period rising edge event:

• PSMCxB is set active

Even numbered period falling edge event:

• PSMCxB is set inactive

Code for setting up the PSMC generate the complementary single-phase waveform shown in Figure 24-6, and given in Example 24-3.

### EXAMPLE 24-3:    PUSH-PULL SETUP

```
; Push-Pull PWM PSMC setup
; Fully synchronous operation
; Period = 10 us
; Duty cycle = 50% (25% each phase)
    BANKSEL  PSMC1CON
    MOVLW    0x02       ; set period
    MOVWF    PSMC1PRH
    MOVLW    0x7F
    MOVWF    PSMC1PRL
    MOVLW    0x01       ; set duty cycle
    MOVWF    PSMC1DCH
    MOVLW    0x3F
    MOVWF    PSMC1DCL
    CLRF     PSMC1PHH   ; no phase offset
    CLRF     PSMC1PHL
    MOVLW    0x01       ; PSMC clock=64 MHz
    MOVWF    PSMC1CLK
; output on A and B, normal polarity
    MOVLW    B'00000011'
    MOVWF    PSMC1OEN
    CLRF     PSMC1POL
; set time base as source for all events
    BSF      PSMC1PRS, P1PRST
    BSF      PSMC1PHS, P1PHST
    BSF      PSMC1DCS, P1DCST
; enable PSMC in Push-Pull Mode
; this also loads steering and time buffers
    MOVLW    B'11000010'
    BANKSEL  TRISC
    BCF      TRISC, 0   ; enable pin drivers
    BCF      TRISC, 1
```

### FIGURE 24-6:    PUSH-PULL PWM WAVEFORM
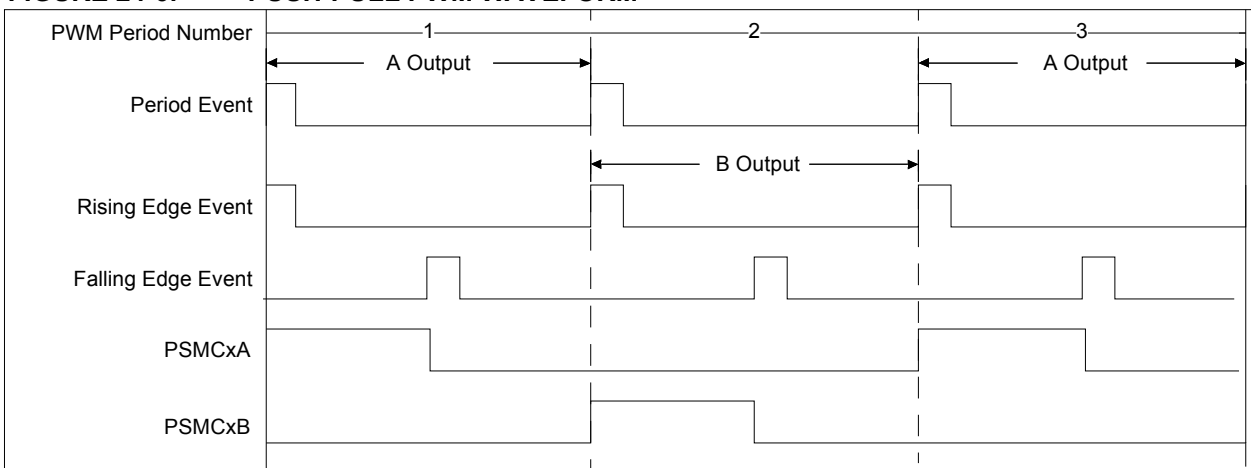
### 24.3.4 PUSH-PULL PWM WITH COMPLEMENTARY OUTPUTS

The complementary push-pull PWM is used to drive transistor bridge circuits as well as synchronous switches on the secondary side of the bridge. The PWM waveform is output on four pins presented as two pairs of two-output signals with a normal and complementary output in each pair. Dead band can be inserted between the normal and complementary outputs at the transition times.

#### 24.3.4.1 Mode Features

- Dead-band control is available
- No steering control available
- Primary PWM output is only on:
  - PSMCxA
  - PSMCxE
- Complementary PWM output is only on:
  - PSMCxB
  - PSMCxF

**Note:** This is a subset of the 6-pin output of the push-pull PWM output, which is why pin functions are fixed in these positions, so they are compatible with that mode. See **Section 24.3.6 "Push-Pull PWM with 4 Full-Bridge and Complementary Outputs"**.

#### 24.3.4.2 Waveform Generation

Push-Pull waveforms generate alternating outputs on the output pairs. Therefore, there are two sets of rising edge events and two sets of falling edge events

Odd numbered period rising edge event:

- PSMCxE is set inactive
- Dead-band rising is activated (if enabled)
- PSMCxA is set active

Odd numbered period falling edge odd event:

- PSMCxA is set inactive
- Dead-band falling is activated (if enabled)
- PSMCxE is set active

Even numbered period rising edge event:

- PSMCxF is set inactive
- Dead-band rising is activated (if enabled)
- PSMCxB is set active

Even numbered period falling edge event:

- PSMCxB is set inactive
- Dead-band falling is activated (if enabled)
- PSMCxF is set active

**FIGURE 24-7: PUSH-PULL WITH COMPLEMENTARY OUTPUTS PWM WAVEFORM**
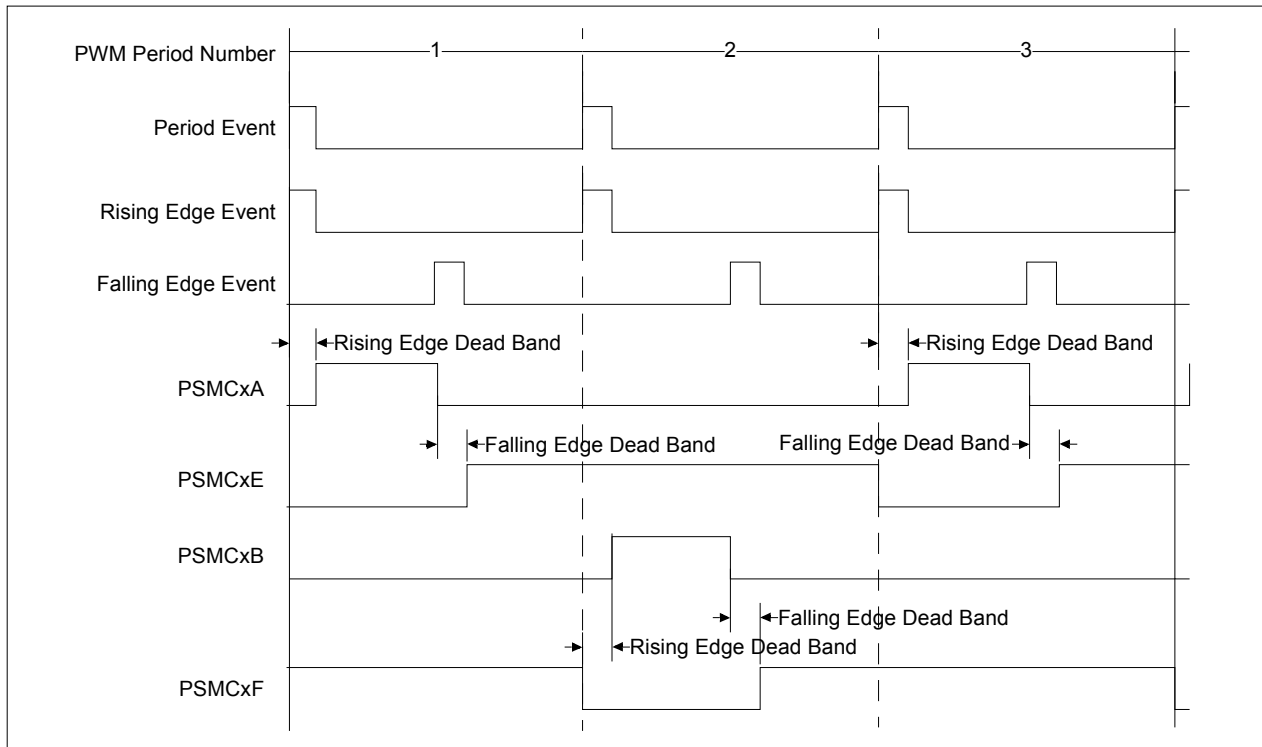
# PIC16(L)F1782/3

### 24.3.5 PUSH-PULL PWM WITH 4 FULL-BRIDGE OUTPUTS

The full-bridge push-pull PWM is used to drive transistor bridge circuits as well as synchronous switches on the secondary side of the bridge.

#### 24.3.5.1 Mode Features

• No Dead-band control
• No Steering control available
• PWM is output on the following four pins only:
  - PSMCxA
  - PSMCxB
  - PSMCxC
  - PSMCxD

> **Note:** PSMCxA and PSMCxC are identical waveforms, and PSMCxB and PSMCxD are identical waveforms.

> **Note:** This is a subset of the 6-pin output of the push-pull PWM output, which is why pin functions are fixed in these positions, so they are compatible with that mode. See **Section 24.3.6 "Push-Pull PWM with 4 Full-Bridge and Complementary Outputs"**.

#### 24.3.5.2 Waveform generation

Push-pull waveforms generate alternating outputs on the output pairs. Therefore, there are two sets of rising edge events and two sets of falling edge events.

Odd numbered period rising edge event:

• PSMCxOUT0 and PSMCxOUT2 is set active

Odd numbered period falling edge event:

• PSMCxOUT0 and PSMCxOUT2 is set inactive

Even numbered period rising edge event:

• PSMCxOUT1 and PSMCxOUT3 is set active

Even numbered period falling edge event:

• PSMCxOUT1 and PSMCxOUT3 is set inactive

**FIGURE 24-8:       PUSH-PULL PWM WITH 4 FULL-BRIDGE OUTPUTS**



**Preliminary** © 2011-2012 Microchip Technology Inc.
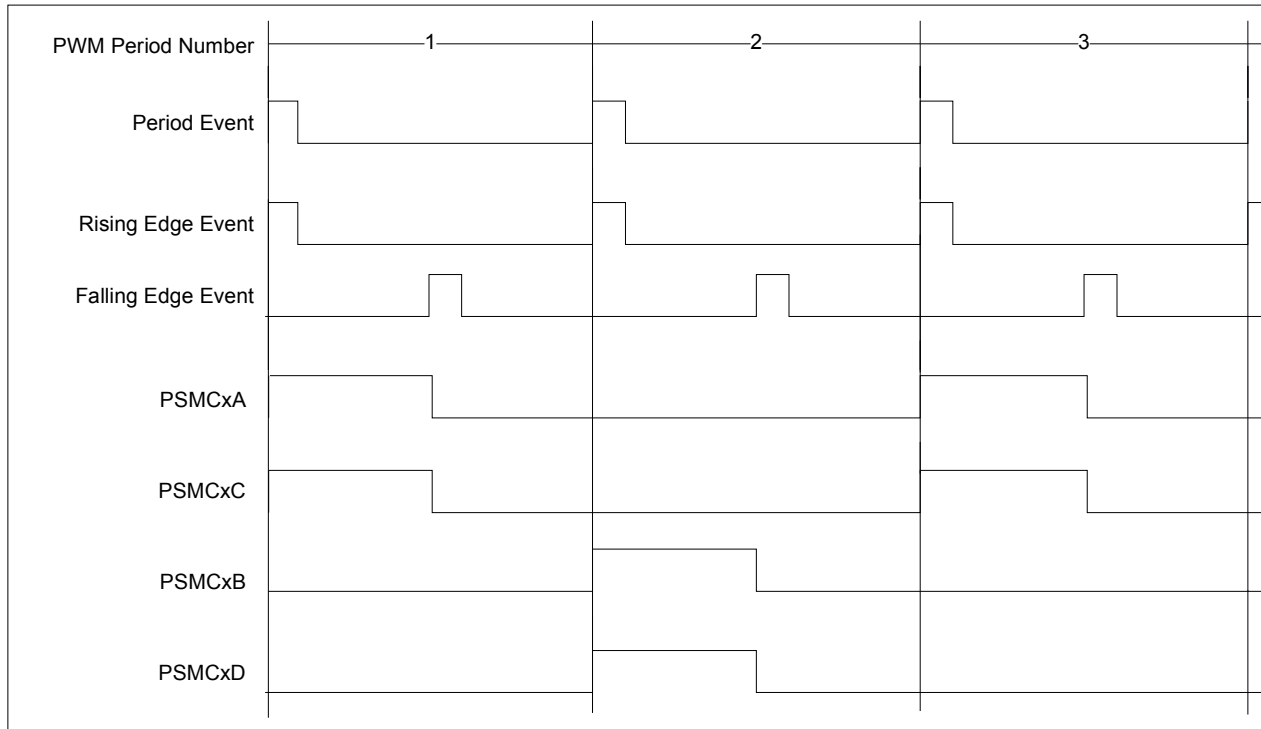
## 24.3.6 PUSH-PULL PWM WITH 4 FULL-BRIDGE AND COMPLEMENTARY OUTPUTS

The push-pull PWM is used to drive transistor bridge circuits as well as synchronous switches on the secondary side of the bridge. It uses six outputs and generates PWM signals with dead band that alternate between the six outputs in even and odd cycles.

### 24.3.6.1 Mode Features and Controls

- Dead-band control is available
- No steering control available
- Primary PWM is output on the following four pins:
  - PSMCxA
  - PSMCxB
  - PSMCxC
  - PSMCxD
- Complementary PWM is output on the following two pins:
  - PSMCxE
  - PSMCxF

> **Note:** PSMCxA and PSMCxC are identical waveforms, and PSMCxB and PSMCxD are identical waveforms.

### 24.3.6.2 Waveform Generation

Push-pull waveforms generate alternating outputs on two sets of pin. Therefore, there are two sets of rising edge events and two sets of falling edge events

Odd numbered period rising edge event:

- PSMCxE is set inactive
- Dead-band rising is activated (if enabled)
- PSMCxA and PSMCxC are set active

Odd numbered period falling edge event:

- PSMCxA and PSMCxC are set inactive
- Dead-band falling is activated (if enabled)
- PSMCxE is set active

Even numbered period rising edge event:

- PSMCxF is set inactive
- Dead-band rising is activated (if enabled)
- PSMCxB and PSMCxD are set active

Even numbered period falling edge event:

- PSMCxB and PSMCxOUT3 are set inactive
- Dead-band falling is activated (if enabled)
- PSMCxF is set active

**FIGURE 24-9:** **PUSH-PULL 4 FULL-BRIDGE AND COMPLEMENTARY PWM**
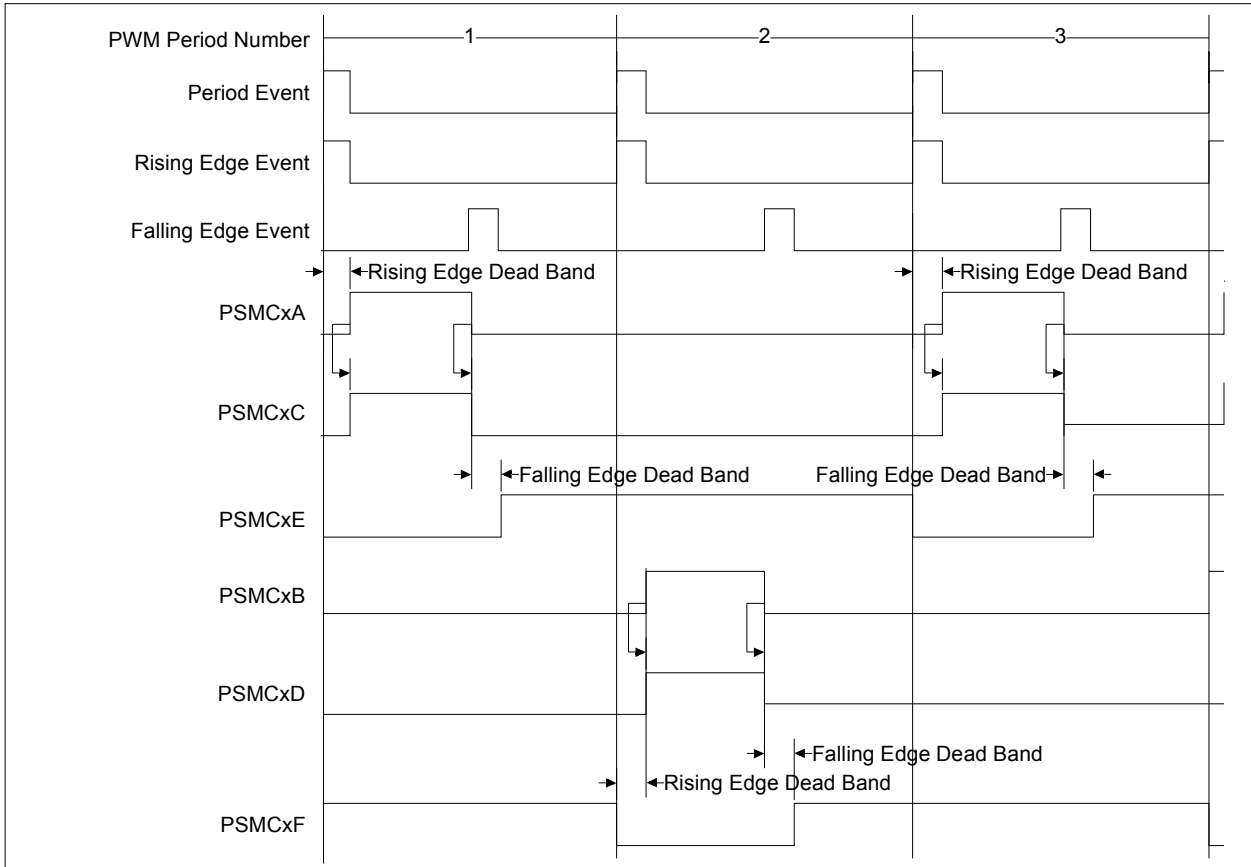


## 24.3.7    PULSE-SKIPPING PWM

The pulse-skipping PWM is used to generate a series of fixed-length pulses that can be triggered at each period event. A rising edge event will be generated when any enabled asynchronous rising edge input is active when the period event occurs, otherwise no event will be generated.

The rising edge event occurs based upon the value in the PSMCxPH register pair.

The falling edge event always occurs according to the enabled event inputs without qualification between any two inputs.

### 24.3.7.1    Mode Features

• No dead-band control available
• No steering control available
• PWM is output to only one pin:
  - PSMCxA

### 24.3.7.2    Waveform Generation

Rising Edge Event

If any enabled asynchronous rising edge event = 1 when there is a period event, then upon the next synchronous rising edge event:

• PSMCxA is set active

Falling Edge Event

• PSMCxA is set inactive

> **Note:** To use this mode, an external source must be used for the determination of whether or not to generate the set pulse. If the phase time base is used, it will either always generate a pulse or never generate a pulse based on the PSMCxPH value.

**FIGURE 24-10: PULSE-SKIPPING PWM WAVEFORM**
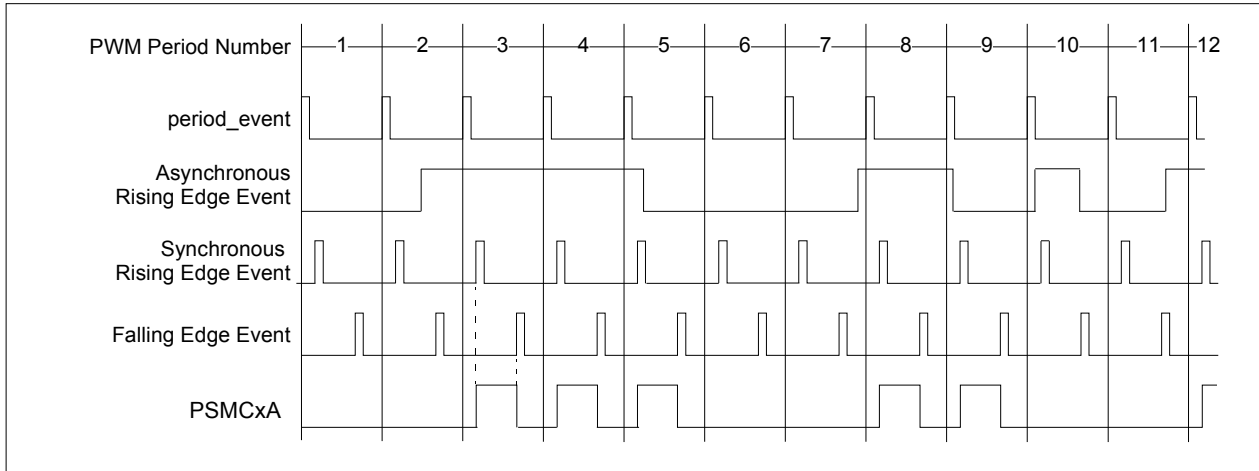
# PIC16(L)F1782/3

### 24.3.8 PULSE-SKIPPING PWM WITH COMPLEMENTARY OUTPUTS

The pulse-skipping PWM is used to generate a series of fixed-length pulses that may or not be triggered at each period event. If any of the sources enabled to generate a rising edge event are high when a period event occurs, a pulse will be generated. If the rising edge sources are low at the period event, no pulse will be generated.

The rising edge occurs based upon the value in the PSMCxPH register pair.

The falling edge event always occurs according to the enabled event inputs without qualification between any two inputs.

#### 24.3.8.1 Mode Features

• Dead-band control is available
• No steering control available
• Primary PWM is output on only PSMCxA.
• Complementary PWM is output on only PSMCxB.

#### 24.3.8.2 Waveform Generation

Rising Edge Event

If any enabled asynchronous rising edge event = 1 when there is a period event, then upon the next synchronous rising edge event:

• Complementary output is set inactive
• Dead-band rising is activated (if enabled)
• Primary output is set active

Falling Edge Event

• Primary output is set inactive
• Dead-band falling is activated (if enabled)
• Complementary output is set active

> **Note:** To use this mode, an external source must be used for the determination of whether or not to generate the set pulse. If the phase time base is used, it will either always generate a pulse or never generate a pulse based on the PSMCxPH value.

**FIGURE 24-11:** PULSE SKIPPING WITH COMPLEMENTARY OUTPUT PWM WAVEFORM



**Preliminary** © 2011-2012 Microchip Technology Inc.
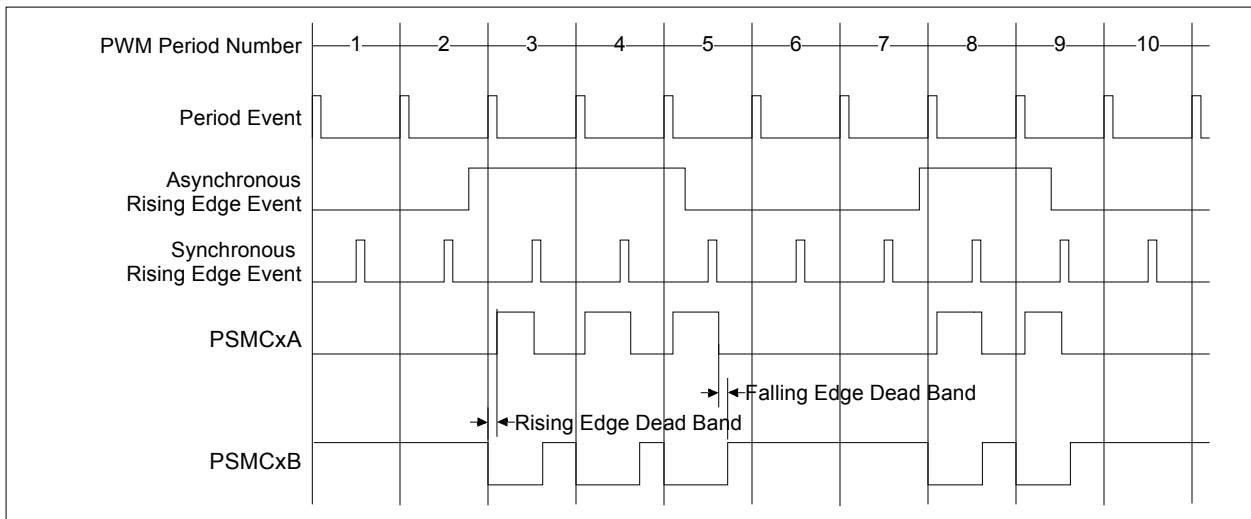
### 24.3.9 ECCP COMPATIBLE FULL-BRIDGE PWM

This mode of operation is designed to match the Full-Bridge mode from the ECCP module. It is called ECCP compatible as the term "full-bridge" alone has different connotations in regards to the output waveforms.

Full-Bridge Compatible mode uses the same waveform events as the single PWM mode to generate the output waveforms.

There are both Forward and Reverse modes available for this operation, again to match the ECCP implementation. Direction is selected with the mode control bits.

#### 24.3.9.1 Mode Features

• Dead-band control available on direction switch
  - Changing from forward to reverse uses the falling edge dead-band counters.
  - Changing from reverse to forward uses the rising edge dead-band counters.
• No steering control available
• PWM is output on the following four pins only:
  - PSMCxA
  - PSMCxB
  - PSMCxC
  - PSMCxD

#### 24.3.9.2 Waveform Generation - Forward

In this mode of operation, three of the four pins are static. PSMCxA is the only output that changes based on rising edge and falling edge events.

Static Signal Assignment

• Outputs set to active state
  - PSMCxD
• Outputs set to inactive state
  - PSMCxB
  - PSMCxC

Rising Edge Event

• PSMCxA is set active

Falling Edge Event

• PSMCxA is set inactive

#### 24.3.9.3 Waveform Generation - Reverse

In this mode of operation, three of the four pins are static. Only PSMCxB toggles based on rising edge and falling edge events.

Static Signal Assignment

• Outputs set to active state
  - PSMCxC
• Outputs set to inactive state
  - PSMCxA
  - PSMCxD

Rising Edge Event

• PSMCxB is set active

Falling Edge Event

• PSMCxB is set inactive

**FIGURE 24-12: ECCP COMPATIBLE FULL-BRIDGE PWM WAVEFORM - PSMCXSTR0 = 0FH**
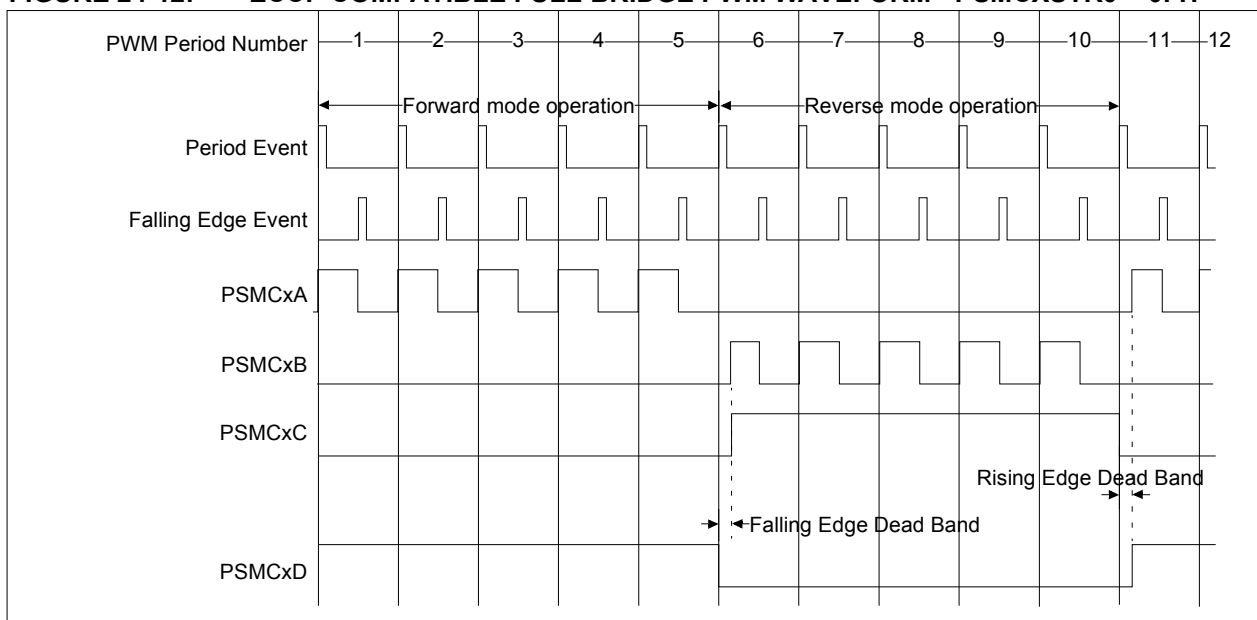
# PIC16(L)F1782/3

### 24.3.10 VARIABLE FREQUENCY - FIXED DUTY CYCLE PWM

This mode of operation is quite different from all of the other modes. It uses only the period event for waveform generation. At each period event, the PWM output is toggled.

The rising edge and falling edge events are unused in this mode.

#### 24.3.10.1 Mode Features

- No dead-band control available
- No steering control available
- Fractional Frequency Adjust
  - Fine period adjustments are made with the PSMC Fractional Frequency Adjust (PSMCxFFA) register (Register 24-27)
- PWM is output on the following pin only:
  - PSMCxA

#### 24.3.10.2 Waveform Generation

Period Event

- Output of PSMCxA is toggled
- FFA counter is incremented by the 4-bit value in PSMCxF FA

**FIGURE 24-13:** VARIABLE FREQUENCY – FIXED DUTY CYCLE PWM WAVEFORM

**Preliminary**

© 2011-2012 Microchip Technology Inc.
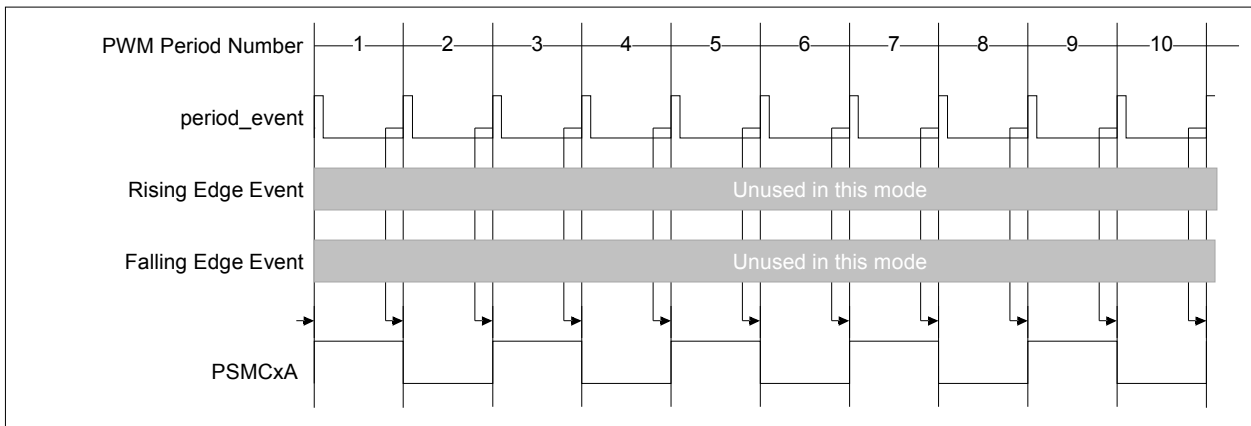
### 24.3.11 VARIABLE FREQUENCY - FIXED DUTY CYCLE PWM WITH COMPLEMENTARY OUTPUTS

This mode is the same as the single output Fixed Duty Cycle mode except a complementary output with dead-band control is generated.

The rising edge and falling edge events are unused in this mode. Therefore, a different triggering mechanism is required for the dead-band counters.

A period events that generate a rising edge on PSMCxA use the rising edge dead-band counters.

A period events that generate a falling edge on PSMCxA use the falling edge dead-band counters.

#### 24.3.11.1 Mode Features

- Dead-band control is available
- No steering control available
- Fractional Frequency Adjust
  - Fine period adjustments are made with the PSMC Fractional Frequency Adjust (PSMCxFFA) register (Register 24-27)
- Primary PWM is output to the following pins:
  - PSMCxA
  - PSMCxC
  - PSMCxE
- Complementary PWM is output to the following pins:
  - PSMCxB
  - PSMCxD
  - PSMCxF

#### 24.3.11.2 Waveform Generation

Period Event

When output is going inactive to active:

- Complementary output is set inactive
- FFA counter is incremented by the 4-bit value in PSMCFFA register.
- Dead-band rising is activated (if enabled)
- Primary output is set active

When output is going active to inactive:

- Primary output is set inactive
- FFA counter is incremented by the 4-bit value in PSMCFFA register
- Dead-band falling is activated (if enabled)
- Complementary output is set active

**FIGURE 24-14:** **VARIABLE FREQUENCY – FIXED DUTY CYCLE PWM WITH COMPLEMENTARY OUTPUTS WAVEFORM**
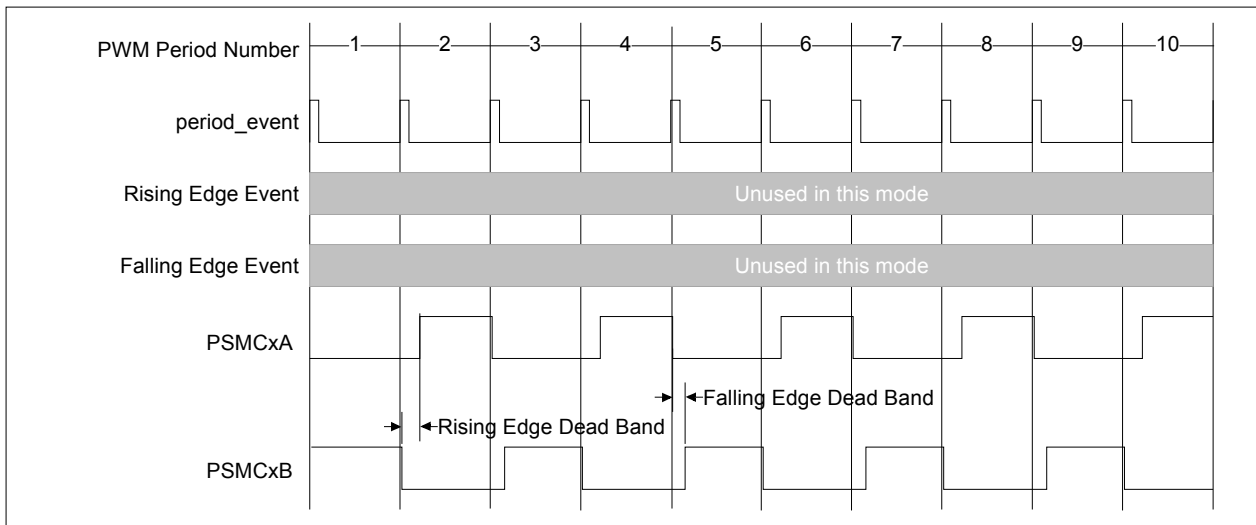
### 24.3.12    3-PHASE PWM

The 3-Phase mode of operation is used in 3-phase power supply and motor drive applications configured as three half-bridges. A half-bridge configuration consists of two power driver devices in series, between the positive power rail (high side) and negative power rail (low side). The three outputs come from the junctions between the two drivers in each half-bridge. When the steering control selects a phase drive, power flows from the positive rail through a high-side power device to the load and back to the power supply through a low-side power device.

In this mode of operation, all six PSMC outputs are used, but only two are active at a time.

The two active outputs consist of a high-side driver and low-side driver output.

#### 24.3.12.1    Mode Features

- No dead-band control is available
- PWM can be steered to the following six pairs:
  - PSMCxA and PSMCxD
  - PSMCxA and PSMCxF
  - PSMCxC and PSMCxF
  - PSMCxC and PSMCxB
  - PSMCxE and PSMCxB
  - PSMCxE and PSMCxD

#### 24.3.12.2    Waveform Generation

3-phase steering has a more complex waveform generation scheme than the other modes. There are several factors which go into what waveforms are created.

The PSMC outputs are grouped into 3 sets of drivers: one for each phase. Each phase has two associated PWM outputs: one for the high-side drive and one for the low-side drive.

High Side drives are indicated by 1H, 2H and 3H.

Low Side drives are indicated by 1L, 2L, 3L.

Phase grouping is mapped as shown in Table 24-1. There are six possible phase drive combinations. Each phase drive combination activates two of the six outputs and deactivates the other four. Phase drive is selected with the steering control as shown in Table 24-2.

**TABLE 24-1:    PHASE GROUPING**

| PSMC grouping | |
|---|---|
| PSMCxA | 1H |
| PSMCxB | 1L |
| PSMCxC | 2H |
| PSMCxD | 2L |
| PSMCxE | 3H |
| PSMCxF | 3L |

**TABLE 24-2:    3-PHASE STEERING CONTROL**

| PSMC outputs | | PSMCxSTR0 Value[1] | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 00h | 01h | 02h | 04h | 08h | 10h | 20h |
| PSMCxA | 1H | inactive | **active** | **active** | inactive | inactive | inactive | inactive |
| PSMCxB | 1L | inactive | inactive | inactive | inactive | **active** | **active** | inactive |
| PSMCxC | 2H | inactive | inactive | inactive | **active** | **active** | inactive | inactive |
| PSMCxD | 2L | inactive | **active** | inactive | inactive | inactive | inactive | **active** |
| PSMCxE | 3H | inactive | inactive | inactive | inactive | inactive | **active** | **active** |
| PSMCxF | 3L | inactive | inactive | **active** | **active** | inactive | inactive | inactive |

**Note  1:**  Steering for any value other than those shown will default to the output combination of the Least Significant steering bit that is set.

**High/Low Side Modulation Enable**

It is also possible to enable the PWM output on the low side or high side drive independently using the PxLSMEN and PXHSMEN bits of the PSMC Steering Control 1 (PSMCxSTR1) register (Register 24-31).

When the PxHSMEN bit is set, the active-high side output listed in Table 24-2 is modulated using the normal rising edge and falling edge events.

When the PxLSMEN bit is set, the active-low side output listed in Table 24-2 is modulated using the normal rising edge and falling edge events.

When both the PxHSMEN and PxLSMEN bits are cleared, the active outputs listed in Table 24-2 go immediately to the rising edge event states and do not change.

Rising Edge Event

- Active outputs are set to their active states

Falling Edge Event

- Active outputs are set to their inactive state

**PIC16(L)F1782/3**

**FIGURE 24-15:**     **3-PHASE PWM STEERING WAVEFORM (PXHSMEN = 0 AND PXLSMEN = 1)**
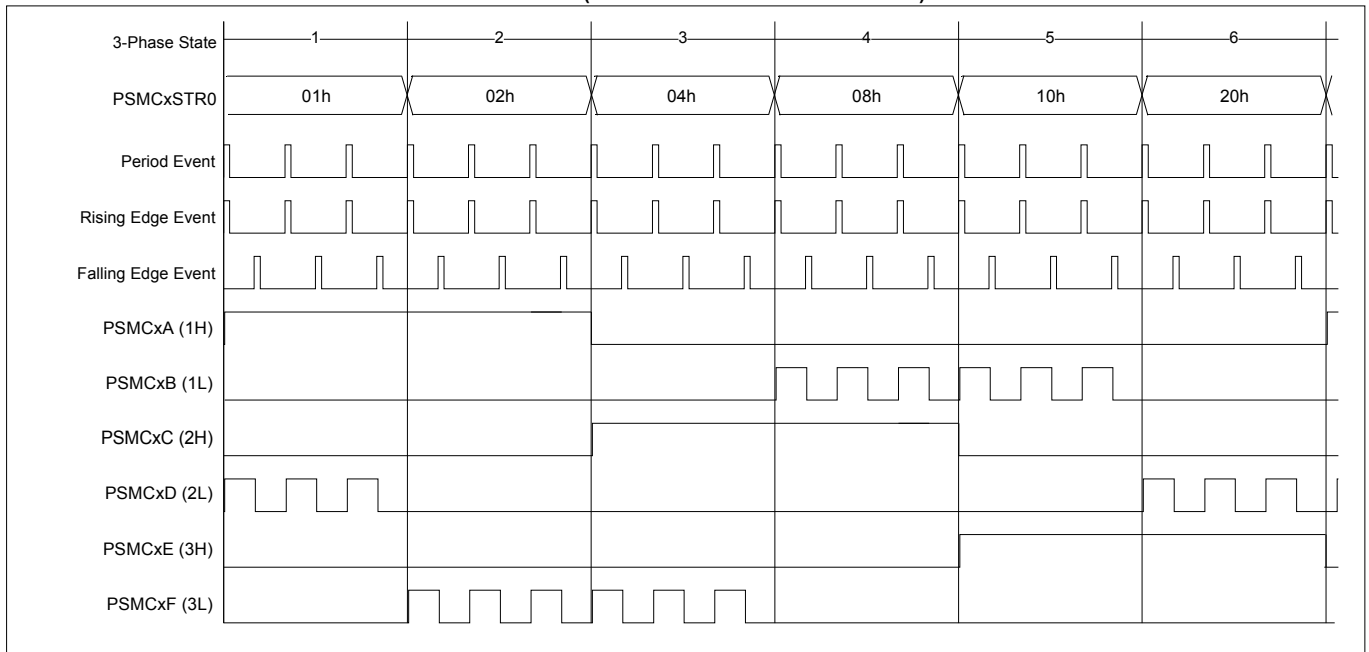
## 24.4 Dead-Band Control

The dead-band control provides non-overlapping PWM signals to prevent shoot-through current in series connected power switches. Dead-band control is available only in modes with complementary drive and when changing direction in the ECCP compatible Full-Bridge modes.

The module contains independent 8-bit dead-band counters for rising edge and falling edge dead-band control.

### 24.4.1 DEAD-BAND TYPES

There are two separate dead-band generators available, one for rising edge events and the other for falling edge events.

#### 24.4.1.1 Rising Edge Dead Band

Rising edge dead-band control is used to delay the turn-on of the primary switch driver from when the complementary switch driver is turned off.

Rising edge dead band is initiated with the rising edge event.

Rising edge dead-band time is adjusted with the PSMC Rising Edge Dead-Band Time (PSMCxDBR) register (Register 24-25).

If the PSMCxDBR register value is changed when the PSMC is enabled, the new value does not take effect until the first period event after the PSMCxLD bit is set.

#### 24.4.1.2 Falling Edge Dead Band

Falling edge dead-band control is used to delay the turn-on of the complementary switch driver from when the primary switch driver is turned off.

Falling edge dead band is initiated with the falling edge event.

Falling edge dead-band time is adjusted with the PSMC Falling Edge Dead-Band Time (PSMCxDBF) register (Register 24-26).

If the PSMCxDBF register value is changed when the PSMC is enabled, the new value does not take effect until the first period event after the PSMCxLD bit is set.

### 24.4.2 DEAD-BAND ENABLE

When a mode is selected that may use dead-band control, dead-band timing is enabled by setting one of the enable bits in the PSMC Control (PSMCxCON) register (Register 24-1).

Rising edge dead band is enabled with the PxDBRE bit.

Rising edge dead band is enabled with the PxDBFE bit.

Enable changes take effect immediately.

### 24.4.3 DEAD-BAND CLOCK SOURCE

The dead-band counters are incremented on every rising edge of the psmc_clk signal.

### 24.4.4 DEAD-BAND UNCERTAINTY

When the rising and falling edge events that trigger the dead-band counters come from asynchronous inputs, there will be uncertainty in the actual dead-band time of each cycle. The maximum uncertainty is equal to one psmc_clk period. The one clock of uncertainty may still be introduced, even when the dead-band count time is cleared to zero.

### 24.4.5 DEAD-BAND OVERLAP

There are two cases of dead-band overlap and each is treated differently due to system requirements.

#### 24.4.5.1 Rising to Falling Overlap

In this case, the falling edge event occurs while the rising edge dead-band counter is still counting. The following sequence occurs:

1. Dead-band rising count is terminated.
2. Dead-band falling count is initiated.
3. Primary output is suppressed.

#### 24.4.5.2 Falling to Rising Overlap

In this case, the rising edge event occurs while the falling edge dead-band counter is still counting. The following sequence occurs:

1. Dead-band falling count is terminated.
2. Dead-band rising count is initiated.
3. Complementary output is suppressed.

#### 24.4.5.3 Rising Edge-to-Rising Edge or Falling Edge-to-Falling Edge

In cases where one of the two dead-band counters is set for a short period, or disabled all together, it is possible to get rising-to-rising or falling-to-falling overlap. When this is the case, the following sequence occurs:

1. Dead-band count is terminated.
2. Dead-band count is restarted.
3. Output waveform control freezes in the present state.
4. Restarted dead-band count completes.
5. Output control resumes normally.

## 24.5 Output Steering

Output Steering allows for PWM signals generated by the PSMC module to be placed on different pins under software control. Synchronized steering will hold steering changes until the first period event after the PSMCxLD bit is set. Unsynchronized steering changes will take place immediately.

Output steering is available in the following modes:

• 3-phase PWM
• Single PWM
• Complementary PWM

### 24.5.1 3-PHASE STEERING

3-phase steering is available in the 3-Phase Modulation mode only. For more details on 3-phase steering refer to **Section 24.3.12 "3-Phase PWM"**.

### 24.5.2 SINGLE PWM STEERING

In Single PWM Steering mode, the single PWM signal can be routed to any combination of the PSMC output pins. Examples of unsynchronized single PWM steering are shown in Figure 24-16.

**FIGURE 24-16:** SINGLE PWM STEERING WAVEFORM (NO SYNCHRONIZATION)



With synchronization disabled, it is possible to get glitches on the PWM outputs.
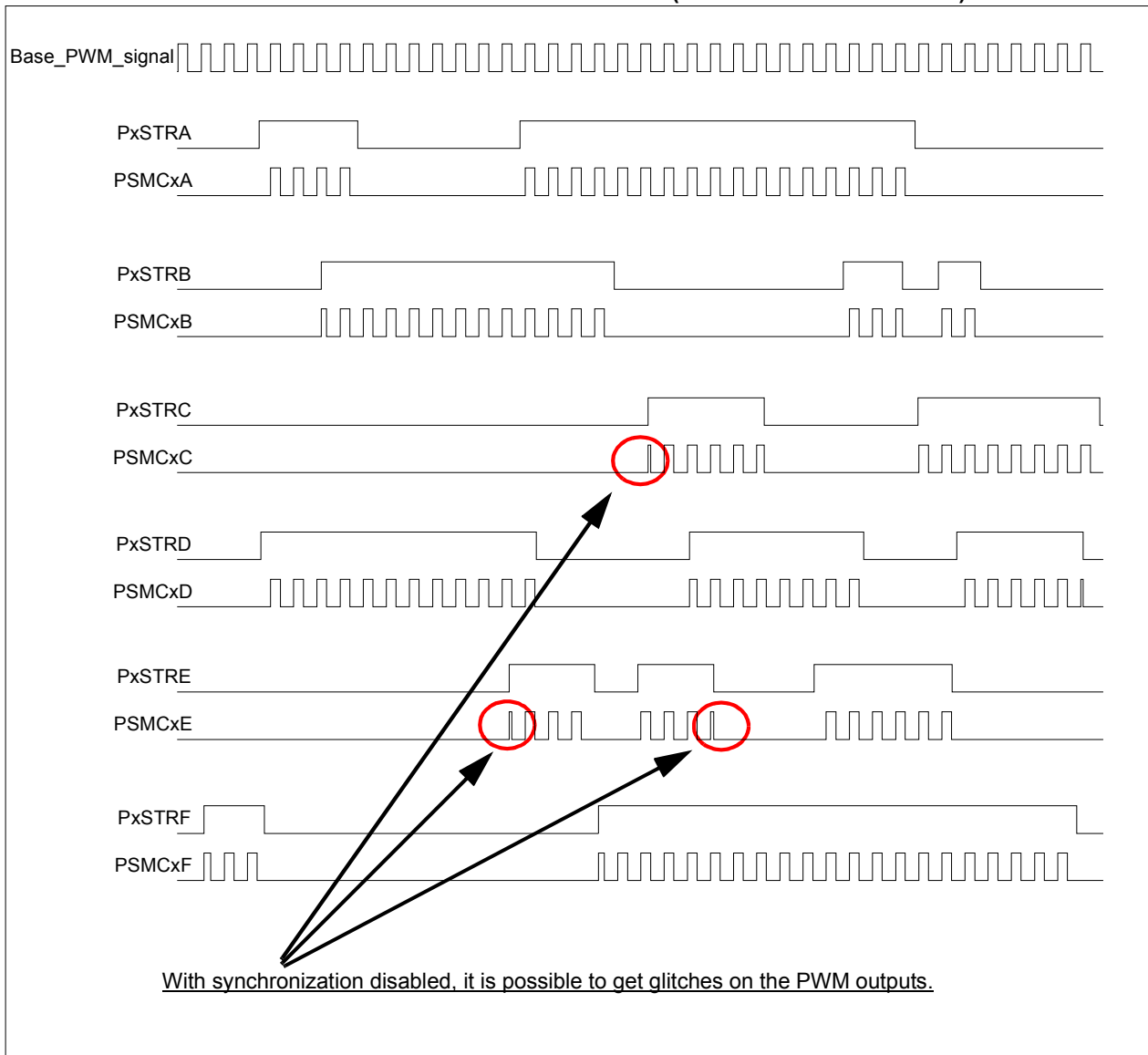
# PIC16(L)F1782/3

### 24.5.3 COMPLEMENTARY PWM STEERING

In Complementary PWM Steering mode, the primary PWM signal (non-complementary) and complementary signal can be steered according to their respective type.

Primary PWM signal can be steered to any of the following outputs:

- PSMCxA
- PSMCxC
- PSMCxE

The complementary PWM signal can be steered to any of the following outputs:

- PSMCxB
- PSMCxD
- PSMCxE

Examples of unsynchronized complementary steering are shown in Figure 24-17.

**FIGURE 24-17:** **COMPLEMENTARY PWM STEERING WAVEFORM (NO SYNCHRONIZATION, ZERO DEAD-BAND TIME)**



Arrows indicate where a change in the steering bit automatically forces a change in the corresponding PSMC output.

**Preliminary**

© 2011-2012 Microchip Technology Inc.
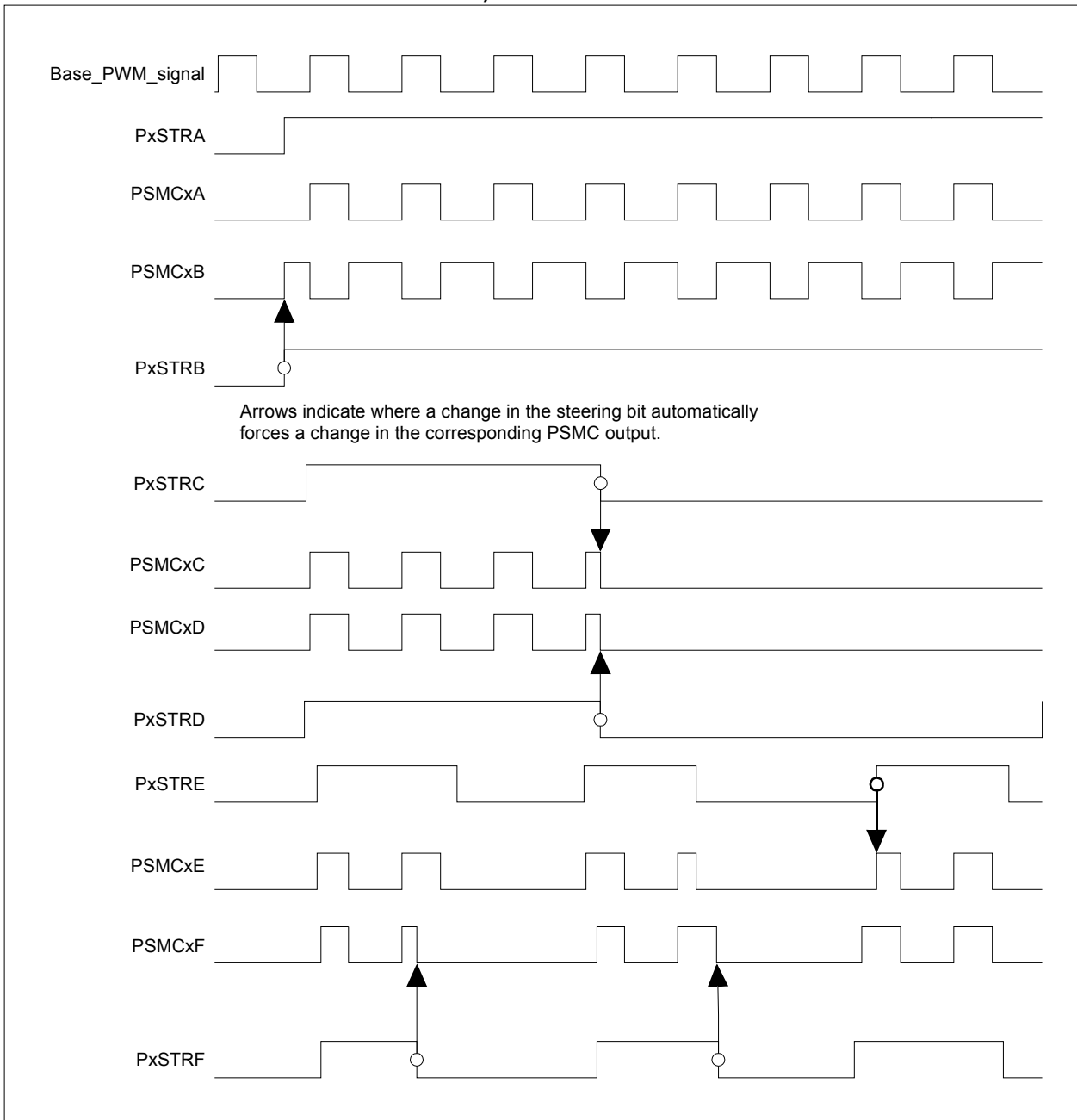
## 24.5.4 SYNCHRONIZED PWM STEERING

In Single, Complementary and 3-phase PWM modes, it is possible to synchronize changes to steering selections with the period event. This is so that PWM outputs do not change in the middle of a cycle and therefore, disrupt operation of the application.

Steering synchronization is enabled by setting the PxSSYNC bit of the PSMC Steering Control 1 (PSMCxSTR1) register (Register 24-31).

When synchronized steering is enabled while the PSMC module is enabled, steering changes do not take effect until the first period event after the PSMCxLD bit is set.

Examples of synchronized steering are shown in Figure 24-18.

## 24.5.5 INITIALIZING SYNCHRONIZED STEERING

If synchronized steering is to be used, special care should be taken to initialize the PSMC Steering Control 0 (PSMCxSTR0) register (Register 24-30) in a safe configuration before setting either the PSMCxEN or PSMCxLD bits. When either of those bits are set, the PSMCxSTR0 value at that time is loaded into the synchronized steering output buffer. The buffer load occurs even if the PxSSYNC bit is low. When the PxSSYNC bit is set, the outputs will immediately go to the drive states in the preloaded buffer.

**FIGURE 24-18:** **PWM STEERING WITH SYNCHRONIZATION WAVEFORM**
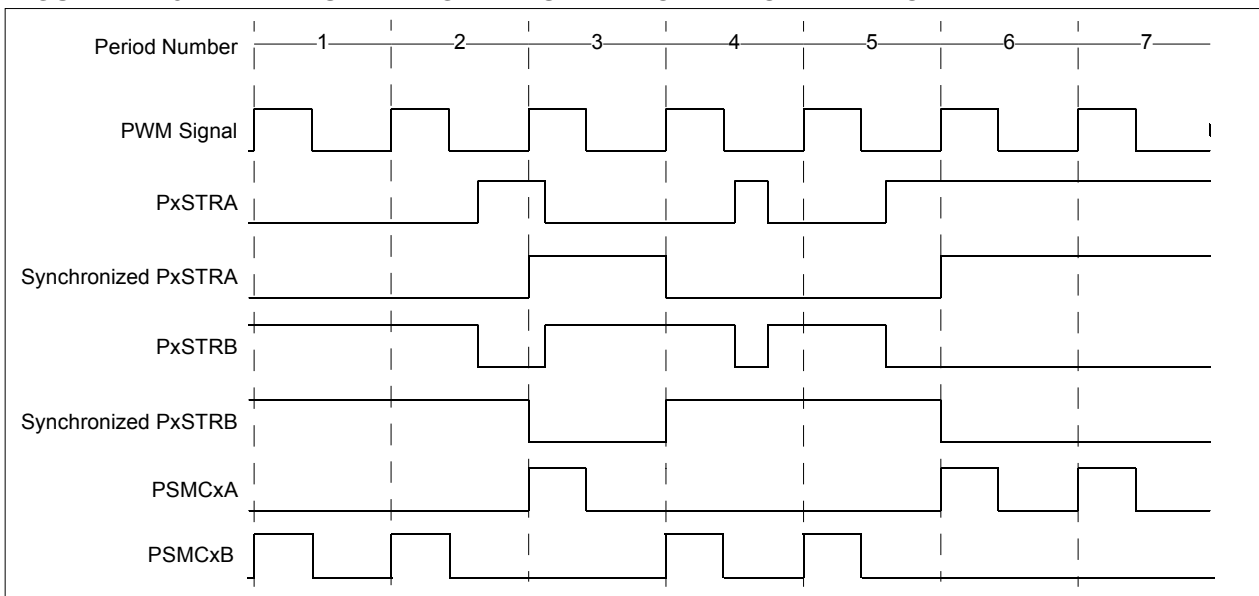
# PIC16(L)F1782/3

## 24.6 PSMC Modulation (Burst Mode)

PSMC Modulation is a method to stop/start PWM operation of the PSMC without having to disable the module. It also allows other modules to control the operational period of the PSMC. This is also referred to as Burst mode.

This is a method to implement PWM dimming.

### 24.6.1 MODULATION ENABLE

The modulation function is enabled by setting the PxMDLEN bit of PSMC Modulation Control (PSMCxMDL) register (Register 24-2).

When modulation is enabled, the modulation source controls when the PWM signals are active and inactive.

When modulation is disabled, the PWM signals operate continuously, regardless of the selected modulation source.

### 24.6.2 MODULATION SOURCES

There are multiple sources that can be used for modulating the PSMC. However, unlike the PSMC input sources, only one modulation source can be selected at a time. Modulation sources include:

- PSMCxIN Pin
- Any CCP output
- Any Comparator output
- PxMDLBIT of the PSMCxMDL register

### 24.6.2.1 PxMDLBIT Bit

The PxMDLBIT bit of the PSMC Modulation Control (PSMCxMDL) register (Register 24-2) allows for software modulation control without having to enable/disable other module functions.

### 24.6.3 MODULATION EFFECT ON PWM SIGNALS

When modulation starts, the PSMC begins operation on a new period, just as if it had rolled over from one period to another during continuous operation.

When modulation stops, its operation depends on the type of waveform being generated.

In Operation modes other than Fixed Duty Cycle, the PSMC completes its current PWM period and then freezes the module. The PSMC output pins are forced into the default inactive state ready for use when modulation starts.

In Fixed Duty Cycle mode operation, the PSMC continues to operate until the period event changes the PWM to its inactive state, at which point the PSMC module is frozen. The PSMC output pins are forced into the default inactive state ready for use when modulation starts.

**FIGURE 24-19:** **PSMC MODULATION WAVEFORM**

**Preliminary**
© 2011-2012 Microchip Technology Inc.
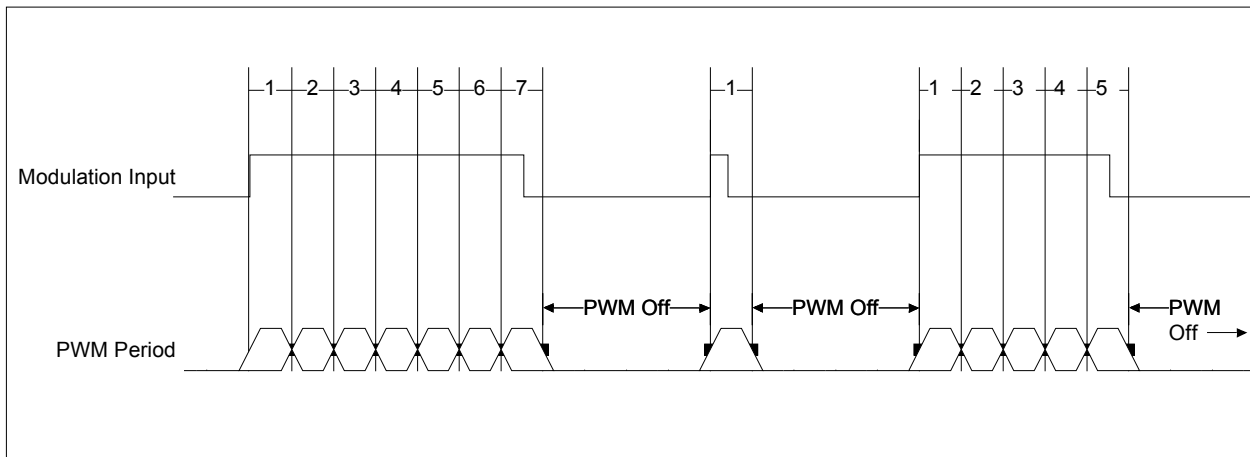
## 24.7 Auto-Shutdown

Auto-shutdown is a method to immediately override the PSMC output levels with specific overrides that allow for safe shutdown of the application.

Auto-shutdown includes a mechanism to allow the application to restart under different conditions.

Auto-shutdown is enabled with the PxASDEN bit of the PSMC Auto-shutdown Control (PSMCxASDC) register (Register 24-14). All auto-shutdown features are enabled when PxASDEN is set and disabled when cleared.

### 24.7.1 SHUTDOWN

There are two ways to generate a shutdown event:

• Manual
• External Input

#### 24.7.1.1 Manual Override

The auto-shutdown control register can be used to manually override the pin functions. Setting the PxASE bit of the PSMC Auto-shutdown Control (PSMCxASDC) register (Register 24-14) generates a software shut-down event.

The auto-shutdown override will persist as long as PxASE remains set.

#### 24.7.1.2 External Input Source

Any of the given sources that are available for event generation are also available for system shut-down. This is so that external circuitry can monitor and force a shutdown without any software overhead. Auto-shutdown sources are selected with the PSMC Auto-shutdown Source (PSMCxASDS) register (Register 24-16).

When any of the selected external auto-shutdown sources go high, the PxASE bit is set and an auto-shutdown interrupt is generated.

| Note: | The external shutdown sources are level sensitive, not edge sensitive. The shutdown condition will persist as long as the circuit is driving the appropriate logic level. |
|---|---|

### 24.7.2 PIN OVERRIDE LEVELS

The logic levels driven to the output pins during an auto-shutdown event are determined by the PSMC Auto-shutdown Output Level (PSMCxASDL) register (Register 24-15).

#### 24.7.2.1 PIN Override Enable

Setting the PxASDOV bit of the PSMC Auto-shutdown Control (PSMCxASDC) register (Register 24-14) will also force the override levels onto the pins, exactly like what happens when the auto-shutdown is used. However, whereas setting PxASE causes an auto-shutdown interrupt, setting PxASDOV does not generate an interrupt.

### 24.7.3 RESTART FROM AUTO-SHUTDOWN

After an auto-shutdown event has occurred, there are two ways for the module to resume operation:

• Manual restart
• Automatic restart

The restart method is selected with the PxARSEN bit of the PSMC Auto-shutdown Control (PSMCxASDC) register (Register 24-14).

#### 24.7.3.1 Manual Restart

When PxARSEN is cleared, and once the PxASDE bit is set, it will remain set until cleared by software.

The PSMC will restart on the period event after PxASDE bit is cleared in software.

#### 24.7.3.2 Auto-Restart

When PxARSEN is set, the PxASDE bit will clear automatically when the source causing the Reset and no longer asserts the shut-down condition.

The PSMC will restart on the next period event after the auto-shutdown condition is removed.

Examples of manual and automatic restart are shown in Figure 24-20.

| Note: | Whether manual or auto-restart is selected, the PxASDE bit cannot be cleared in software when the auto-shutdown condition is still present. |
|---|---|

# PIC16(L)F1782/3

**FIGURE 24-20:    AUTO-SHUTDOWN AND RESTART WAVEFORM**

**Preliminary**

© 2011-2012 Microchip Technology Inc.
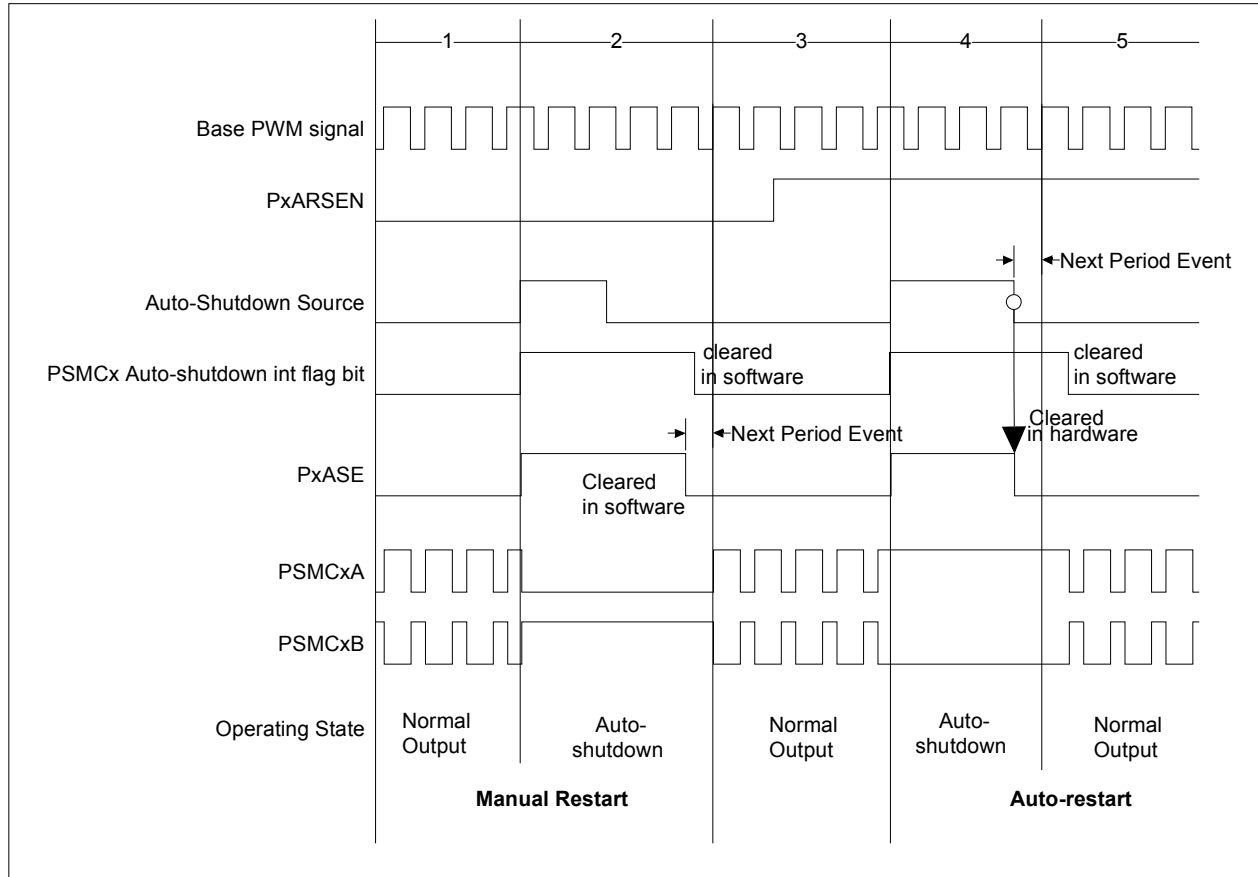
## 24.8 PSMC Synchronization

It is possible to synchronize the periods of two or more PSMC modules together, provided that both modules are on the same device.

Synchronization is achieved by sending a sync signal from the master PSMC module to the desired slave modules. This sync signal generates a period event in each slave module, thereby aligning all slaves with the master. This is useful when an application requires different PWM signal generation from each module but the waveforms must be consistent within a PWM period.

### 24.8.1 SYNCHRONIZATION SOURCES

The synchronization source can be any PSMC module on the same device. For example, in a device with two PSMC modules, the possible sources for each device is as shown below:

- Sources for PSMC1
  - PSMC2
- Sources for PSMC2
  - PSMC1

**FIGURE 24-21: PSMC SYNCHRONIZATION - SYNC OUTPUT TO PIN**
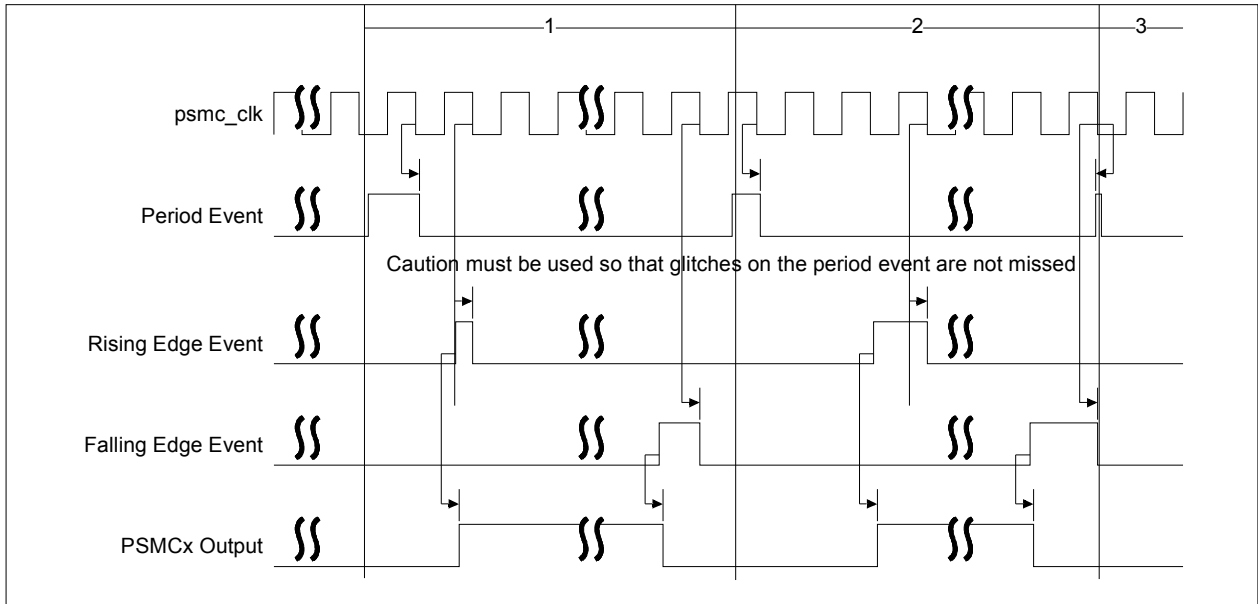


#### 24.8.1.1 PSMC Internal Connections

The sync signal from the master PSMC module is essentially that module's period event trigger. The slave PSMC modules receive and process the sync signal as an additional period event input.

Enabling a module as a slave recipient is done with the PxSYNC bits of the PSMC Synchronization Control 1 (PSMC1SYNC) register (Register 24-3) and the PSMC Synchronization Control 2 (PSMC2SYNC) register (Register 24-4).

#### 24.8.1.2 Synchronization Skid

At high frequencies (i.e., 64 MHz clock), it is possible for slave modules to lag synchronization by a maximum of one clock period.

## 24.9 Fractional Frequency Adjust (FFA)

FFA is a method by which PWM resolution can be improved on 50% fixed duty cycle signals. Higher resolution is achieved by altering the PWM period by a single count for calculated intervals. This increased resolution is based upon the PWM frequency averaged over a large number of PWM periods. For example, if the period event time is increased by one psmc_clk period ($T_{PSMC\_CLK}$) every N events, then the effective resolution of the average event period is $T_{PSMC\_CLK}/N$.

When active, after every period event the FFA hardware adds the PSMCxFFA value with the previously accumulated result. Each time the addition causes an overflow, the period event time is increased by one. Refer to Figure 24-22.

**FIGURE 24-22:** FFA BLOCK DIAGRAM.
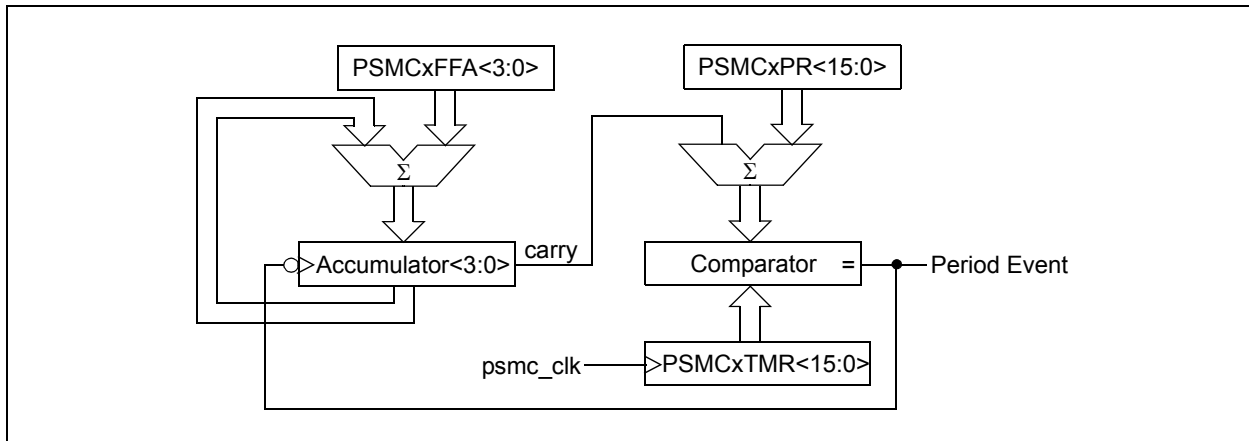


The FFA function is only available when using one of the two Fixed Duty Cycle modes of operation. In fixed duty cycle operation each PWM period is comprised of two period events. That is why the PWM periods in Table 24-3 example calculations are multiplied by 2 as opposed to the normal period calculations for normal mode operation.

The extra resolution gained by the FFA is based upon the number of bits in the FFA register and the psmc_clk frequency. The parameters of interest are:

- $T_{PWM}$ – this is the lower bound of the PWM period that will be adjusted
- $T_{PWM}+1$ – this is the upper bound of the PWM period that will be adjusted. This is used to help determine the step size for each increment of the FFA register
- $T_{RESOLUTION}$ – each increment of the FFA register will add this amount of period to average PWM frequency

**TABLE 24-3:** FRACTIONAL FREQUENCY ADJUST CALCULATIONS

| Parameter | Value |
|---|---|
| $F_{PSMC\_CLK}$ | 64 MHz |
| $T_{PSMC\_CLK}$ | 15.625 ns |
| PSMCxPR<15:0> | 00FFh = 255 |
| $T_{PWM}$ | = (PSMCxPR<15:0>+1)*2*$T_{PSMC\_CLK}$<br>= 256*2*15.625ns<br>= 8 us |
| $F_{PWM}$ | 125 kHz |
| $T_{PWM}+1$ | = (PSMCxPR<15:0>+2)*2*$T_{PSMC\_CLK}$<br>= 257*2*15.625ns<br>= 8.03125 us |
| $F_{PWM}+1$ | = 124.513 kHz |
| $T_{RESOLUTION}$ | = ($T_{PWM}+1-T_{PWM}$)/$2^{FFA-Bits}$<br>= (8.03125us - 8.0 us)/16<br>= 0.03125us/16<br>~ 1.95 ns |
| $F_{RESOLUTION}$ | ($F_{PWM}+1-F_{PWM}$)/$2^{FFA-Bits}$<br>~ -30.4 Hz |

**TABLE 24-4: SAMPLE FFA OUTPUT PERIODS/FREQUENCIES**

| FFA number | Output Frequency (kHz) | Step Size (Hz) |
|:---:|:---:|:---:|
| 0 | 125.000 | 0 |
| 1 | 124.970 | -30.4 |
| 2 | 124.939 | -60.8 |
| 3 | 124.909 | -91.2 |
| 4 | 124.878 | -121.6 |
| 5 | 124.848 | -152.0 |
| 6 | 124.818 | -182.4 |
| 7 | 124.787 | -212.8 |
| 8 | 124.757 | -243.2 |
| 9 | 124.726 | -273.6 |
| 10 | 124.696 | -304.0 |
| 11 | 124.666 | -334.4 |
| 12 | 124.635 | -364.8 |
| 13 | 124.605 | -395.2 |
| 14 | 124.574 | -425.6 |
| 15 | 124.544 | -456.0 |

## 24.10 Register Updates

There are 10 double-buffered registers that can be updated "on the fly". However, due to the asynchronous nature of the potential updates, a special hardware system is used for the updates.

There are two operating cases for the PSMC:

- module is enabled
- module is disabled

### 24.10.1 DOUBLE BUFFERED REGISTERS

The double-buffered registers that are affected by the special hardware update system are:

- PSMCxPRL
- PSMCxPRH
- PSMCxDCL
- PSMCxDCH
- PSMCxPHL
- PSMCxPHH
- PSMCxDBR
- PSMCxDBF
- PSMCxBLKR
- PSMCxBLKF
- PSMCxSTR0 (when the PxSSYNC bit is set)

### 24.10.2 MODULE DISABLED UPDATES

When the PSMC module is disabled (PSMCxEN = 0), any write to one of the buffered registers will also write directly to the buffer. This means that all buffers are loaded and ready for use when the module is enabled.

### 24.10.3 MODULE ENABLED UPDATES

When the PSMC module is enabled (PSMCxEN = 1), the PSMCxLD bit of the PSMC Control (PSMCxCON) register (Register 24-1) must be used.

When the PSMCxLD bit is set, the transfer from the register to the buffer occurs on the next period event. The PSMCxLD bit is automatically cleared by hardware after the transfer to the buffers is complete.

The reason that the PSMCxLD bit is required is that depending on the customer application and operation conditions, all 10 registers may not be updated in one PSMC period. If the buffers are loaded at different times (i.e., DCL gets updated, but DCH does not OR DCL and DCL are updated by PRH and PRL are not), then unintended operation may occur.

The sequence for loading the buffer registers when the PSMC module is enabled is as follows:

1. Software updates all registers.
2. Software sets the PSMCxLD bit.
3. Hardware updates all buffers on the next period event.
4. Hardware clears PSMCxLD bit.

## 24.11 Operation During Sleep

The PSMC continues to operate in sleep with the following clock sources:

- Internal 64 MHz
- External clock

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 24.12 Register Definitions: PSMC Control

**REGISTER 24-1:** **PSMCxCON – PSMC CONTROL REGISTER**

| R/W-0/0 | R/W/HC-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|------------|---------|---------|---------|---------|---------|---------|
| PSMCxEN | PSMCxLD | PxDBFE | PxDBRE | PxMODE<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 Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **PSMCxEN:** PSMC Module Enable bit

1 = PSMCx module is enabled
0 = PSMCx module is disabled

bit 6 **PSMCxLD:** PSMC Load Buffer Enable bit

1 = PSMCx registers are ready to be updated with the appropriate register contents
0 = PSMCx buffer update complete

bit 5 **PxDBFE:** PSMC Falling Edge Dead-Band Enable bit

1 = PSMCx falling edge dead band enabled
0 = PSMCx falling edge dead band disabled

bit 4 **PxDBRE:** PSMC Rising Edge Dead-Band Enable bit

1 = PSMCx rising edge dead band enabled
0 = PSMCx rising edge dead band disabled

bit 3-0 **PxMODE<3:0>** PSMC Operating Mode bits

1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = 3-phase steering PWM
1011 = Fixed duty cycle, variable frequency, complementary PWM
1010 = Fixed duty cycle, variable frequency, single PWM
1001 = ECCP compatible Full-Bridge forward output
1000 = ECCP compatible Full-Bridge reverse output
0111 = Pulse-skipping with complementary output
0110 = Pulse-skipping PWM output
0101 = Push-pull with 4-full-bridge outputs and complementary outputs
0100 = Push-pull with 4-full-bridge outputs
0011 = Push-pull with complementary outputs
0010 = Push-pull output
0001 = Single PWM with complementary output (with PWM steering capability)
0000 = Single PWM waveform generation (with PWM steering capability)

# PIC16(L)F1782/3

## REGISTER 24-2: PSMCxMDL – PSMC MODULATION CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|-----|---------|---------|---------|---------|
| PxMDLEN | PxMDLPOL | PxMDLBIT | — | PxMSRC<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 Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **PxMDLEN:** PSMC Periodic Modulation Mode Enable bit

1 = PSMCx is active when input signal selected by PxMSRC<3:0> is in its active state (see PxMPOL)
0 = PSMCx module is always active

bit 6 **PxMDLPOL:** PSMC Periodic Modulation Polarity bit

1 = PSMCx is active when the PSMCx Modulation source output equals logic '0' (active-low)
0 = PSMCx is active when the PSMCx Modulation source output equals logic '1' (active-high)

bit 5 **PxMDLBIT:** PSMC Periodic Modulation Software Control bit

PxMDLEN = 1 AND PxMSRC<3:0> = 0000
1 = PSMCx is active when the PxMPOL equals logic '0'
0 = PSMCx is active when the PxMPOL equals logic '1'
PxMDLEN = 0 OR (PxMEN = 1 and PxMSRC<3:0> <> '0000')
Does not affect module operation

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

bit 3-0 **PxMSRC<3:0>** PSMC Periodic Modulation Source Selection bits

1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = Reserved
1011 = Reserved
1010 = Reserved
1001 = Reserved
1000 = PSMCx Modulation Source is PSMCxIN pin
0111 = Reserved
0110 = PSMCx Modulation Source is CCP2
0101 = PSMCx Modulation Source is CCP1
0100 = Reserved
0011 = PSMCx Modulation Source is Comparator 3 output
0010 = PSMCx Modulation Source is Comparator 2 output
0001 = PSMCx Modulation Source is Comparator 1 output
0000 = PSMCx Modulation Source is PxMDLBIT register bit

**Preliminary**

**REGISTER 24-3:** **PSMC1SYNC – PSMC1 SYNCHRONIZATION CONTROL REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | P1SYNC<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 7-2    **Unimplemented:** Read as '0'

bit 1-0    **P1SYNC<1:0>:** PSMC1 Period Synchronization Mode bits
10 = PSMC1 is synchronized with the PSMC2 module
01 = Reserved - Do not use
00 = PSMC1 is not synchronized with any other PSMC module

**REGISTER 24-4:** **PSMC2SYNC – PSMC2 SYNCHRONIZATION CONTROL REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | P2SYNC<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 7-2    **Unimplemented:** Read as '0'

bit 1-0    **P2SYNC<1:0>:** PSMC2 Period Synchronization Mode bits
10 = Reserved - Do not use
01 = PSMC2 is synchronized with the PSMC1 module
00 = PSMC2 is not synchronized with any other PSMC module

# PIC16(L)F1782/3

**REGISTER 24-5: PSMCxCLK – PSMC CLOCK CONTROL REGISTER**

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | PxCPRE<1:0> | | — | — | PxCSRC<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 7-6      **Unimplemented:** Read as '0'

bit 5-4      **PxCPRE<1:0>:** PSMCx Clock Prescaler Selection bits

         11 = PSMCx Clock frequency/8
         10 = PSMCx Clock frequency/4
         01 = PSMCx Clock frequency/2
         00 = PSMCx Clock frequency/1

bit 3-2      **Unimplemented:** Read as '0'

bit 1-0      **PxCSRC<1:0>:** PSMCx Clock Source Selection bits

         11 = Reserved
         10 = PSMCxCLK pin
         01 = 64 MHz clock in from PLL
         00 = F<sub>OSC</sub> system clock

---

**REGISTER 24-6: PSMCxOEN – PSMC OUTPUT ENABLE CONTROL REGISTER**

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | PxOEF[1] | PxOEE[1] | PxOED[1] | PxOEC[1] | PxOEB | PxOEA |
| 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      **Unimplemented:** Read as '0'

bit 5-0      **PxOEy:** PSMCx Output y Enable bit[1]

         1 = PWM output is active on PSMCx output y pin
         0 = PWM output is not active, normal port functions in control of pin

**Note 1:** These bits are not implemented on PSMC2.

**REGISTER 24-7:** **PSMCxPOL – PSMC POLARITY CONTROL REGISTER**

| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | PxPOLIN | PxPOLF[1] | PxPOLE[1] | PxPOLD[1] | PxPOLC[1] | PxPOLB | PxPOLA |
| 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 **PxPOLIN:** PSMCxIN Polarity bit
1 = PSMCxIN input is active-low
0 = PSMCxIN input is active-high

bit 5-0 **PxPOLy:** PSMCx Output y Polarity bit[1]
1 = PWM PSMCx output y is active-low
0 = PWM PSMCx output y is active-high

**Note 1:** These bits are not implemented on PSMC2.

**REGISTER 24-8:** **PSMCxBLNK – PSMC BLANKING CONTROL REGISTER**

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | PxFEBM1 | PxFEBM0 | — | — | PxREBM1 | PxREBM0 |
| 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 **Unimplemented:** Read as '0'

bit 5-4 **PxFEBM<1:0>** PSMC Falling Edge Blanking Mode bits
11 = Reserved - do not use
10 = Reserved - do not use
01 = Immediate blanking
00 = No blanking

bit 3-2 **Unimplemented:** Read as '0'

bit 1-0 **PxREBM<1:0>** PSMC Rising Edge Blanking Mode bits
11 = Reserved - do not use
10 = Reserved - do not use
01 = Immediate blanking
00 = No blanking

# PIC16(L)F1782/3

### REGISTER 24-9: PSMCxREBS – PSMC RISING EDGE BLANKED SOURCE REGISTER

| R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 |
|---------|-----|-----|-----|---------|---------|---------|-----|
| PxREBSIN | — | — | — | PxREBSC3 | PxREBSC2 | PxREBSC1 | — |
| 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 **PxREBSIN:** PSMCx Rising Edge Event Blanked from PSMCxIN pin
1 = PSMCxIN pin cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = PSMCxIN pin is not blanked

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **PxREBSC3:** PSMCx Rising Edge Event Blanked from Comparator 3
1 = Comparator 3 cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = Comparator 3 is not blanked

bit 2 **PxREBSC2:** PSMCx Rising Edge Event Blanked from Comparator 2
1 = Comparator 2 cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = Comparator 2 is not blanked

bit 1 **PxREBSC1:** PSMCx Rising Edge Event Blanked from Comparator 1
1 = Comparator 1 cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = Comparator 1 is not blanked

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

### REGISTER 24-10: PSMCxFEBS – PSMC FALLING EDGE BLANKED SOURCE REGISTER

| R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 |
|---------|-----|-----|-----|---------|---------|---------|-----|
| PxFEBSIN | — | — | — | PxFEBSC3 | PxFEBSC2 | PxFEBSC1 | — |
| 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 **PxFEBSIN:** PSMCx Falling Edge Event Blanked from PSMCxIN pin
1 = PSMCxIN pin cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = PSMCxIN pin is not blanked

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **PxFEBSC3:** PSMCx Falling Edge Event Blanked from Comparator 3
1 = Comparator 3 cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = Comparator 3 is not blanked

bit 2 **PxFEBSC2:** PSMCx Falling Edge Event Blanked from Comparator 2
1 = Comparator 2 cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = Comparator 2 is not blanked

bit 1 **PxFEBSC1:** PSMCx Falling Edge Event Blanked from Comparator 1
1 = Comparator 1 cannot cause a rising or falling event for the duration indicated by the PSMCxBLNK register
0 = Comparator 1 is not blanked

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

## REGISTER 24-11: PSMCxPHS – PSMC PHASE SOURCE REGISTER[1]

| R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|-----|-----|-----|---------|---------|---------|---------|
| PxPHSIN | — | — | — | PxPHSC3 | PxPHSC2 | PxPHSC1 | PxPHST |
| 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 **PxPHSIN:** PSMCx Rising Edge Event occurs on PSMCxIN pin
1 = Rising edge event will occur when PSMCxIN pin goes true
0 = PSMCxIN pin will not cause rising edge event

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **PxPHSC3:** PSMCx Rising Edge Event occurs on Comparator 3 output
1 = Rising edge event will occur when Comparator 3 output goes true
0 = Comparator 3 will not cause rising edge event

bit 2 **PxPHSC2:** PSMCx Rising Edge Event occurs on Comparator 2 output
1 = Rising edge event will occur when Comparator 2 output goes true
0 = Comparator 2 will not cause rising edge event

bit 1 **PxPHSC1:** PSMCx Rising Edge Event occurs on Comparator 1 output
1 = Rising edge event will occur when Comparator 1 output goes true
0 = Comparator 1 will not cause rising edge event

bit 0 **PxPHST:** PSMCx Rising Edge Event occurs on Time Base match
1 = Rising edge event will occur when PSMCxTMR = PSMCxPH
0 = Time base will not cause rising edge event

**Note 1:** Sources are not mutually exclusive: more than one source can cause a rising edge event.

# PIC16(L)F1782/3

**REGISTER 24-12:   PSMCxDCS – PSMC DUTY CYCLE SOURCE REGISTER[1]**

| R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|-----|-----|-----|---------|---------|---------|---------|
| PxDCSIN | — | — | — | PxDCSC3 | PxDCSC2 | PxDCSC1 | PxDCST |
| 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      **PxDCSIN:** PSMCx Falling Edge Event occurs on PSMCxIN pin
           1 =   Falling edge event will occur when PSMCxIN pin goes true
           0 =   PSMCxIN pin will not cause falling edge event

bit 6-4    **Unimplemented:** Read as '0'

bit 3      **PxDCSC3:** PSMCx Falling Edge Event occurs on Comparator 3 output
           1 =   Falling edge event will occur when Comparator 3 output goes true
           0 =   Comparator 3 will not cause falling edge event

bit 2      **PxDCSC2:** PSMCx Falling Edge Event occurs on Comparator 2 output
           1 =   Falling edge event will occur when Comparator 2 output goes true
           0 =   Comparator 2 will not cause falling edge event

bit 1      **PxDCSC1:** PSMCx Falling Edge Event occurs on Comparator 1 output
           1 =   Falling edge event will occur when Comparator 1 output goes true
           0 =   Comparator 1 will not cause falling edge event

bit 0      **PxDCST:** PSMCx Falling Edge Event occurs on Time Base match
           1 =   Falling edge event will occur when PSMCxTMR = PSMCxDC
           0 =   Time base will not cause falling edge event

**Note 1:**   Sources are not mutually exclusive: more than one source can cause a falling edge event.

         **Preliminary**          © 2011-2012 Microchip Technology Inc.

**REGISTER 24-13: PSMCxPRS – PSMC PERIOD SOURCE REGISTER**[1]

| R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|-----|-----|-----|---------|---------|---------|---------|
| PxPRSIN | — | — | — | PxPRSC3 | PxPRSC2 | PxPRSC1 | PxPRST |
| 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 **PxPRSIN:** PSMCx Period Event occurs on PSMCxIN pin
  1 = Period event will occur and PSMCxTMR will reset when PSMCxIN pin goes true
  0 = PSMCxIN pin will not cause period event

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **PxPRSC3:** PSMCx Period Event occurs on Comparator 3 output
  1 = Period event will occur and PSMCxTMR will reset when Comparator 3 output goes true
  0 = Comparator 3 will not cause period event

bit 2 **PxPRSC2:** PSMCx Period Event occurs on Comparator 2 output
  1 = Period event will occur and PSMCxTMR will reset when Comparator 2 output goes true
  0 = Comparator 2 will not cause period event

bit 1 **PxPRSC1:** PSMCx Period Event occurs on Comparator 1 output
  1 = Period event will occur and PSMCxTMR will reset when Comparator 1 output goes true
  0 = Comparator 1 will not cause period event

bit 0 **PxPRST:** PSMCx Period Event occurs on Time Base match
  1 = Period event will occur and PSMCxTMR will reset when PSMCxTMR = PSMCxPR
  0 = Time base will not cause period event

**Note 1:** Sources are not mutually exclusive: more than one source can force the period event and reset the PSMCxTMR.

# PIC16(L)F1782/3

**REGISTER 24-14: PSMCxASDC – PSMC AUTO-SHUTDOWN CONTROL REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
|---------|---------|---------|-----|-----|-----|-----|---------|
| PxASE | PxASDEN | PxARSEN | — | — | — | — | PxASDOV |
| 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      **PxASE:** PWM Auto-Shutdown Event Status bit[1]

         1 = A shutdown event has occurred, PWM outputs are inactive and in their shutdown states
         0 = PWM outputs are operating normally

bit 6      **PxASDEN:** PWM Auto-Shutdown Enable bit

         1 = Auto-shutdown is enabled. If any of the sources in PSMCxASDS assert a logic '1', then the outputs will go into their auto-shutdown state and PSMCxSIF flag will be set.
         0 = Auto-shutdown is disabled

bit 5      **PxARSEN:** PWM Auto-Restart Enable bit

         1 = PWM restarts automatically when the shutdown condition is removed.
         0 = The PxASE bit must be cleared in firmware to restart PWM after the auto-shutdown condition is cleared.

bit 4-1      **Unimplemented:** Read as '0'

bit 0      **PxASDOV:** PWM Auto-Shutdown Override bit

         <u>PxASDEN = 1:</u>
         1 = Force PxASDL[n] levels on the PSMCx[n] pins without causing a PSMCxSIF interrupt
         0 = Normal PWM and auto-shutdown execution

         <u>PxASDEN = 0:</u>
         No effect

**Note 1:** PASE bit may be set in software. When this occurs the functionality is the same as that caused by hardware.

**Preliminary**      © 2011-2012 Microchip Technology Inc.

**REGISTER 24-15: PSMCxASDL – PSMC AUTO-SHUTDOWN OUTPUT LEVEL REGISTER**

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | PxASDLF[1] | PxASDLE[1] | PxASDLD[1] | PxASDLC[1] | PxASDLB | PxASDLA |
| 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 **Unimplemented:** Read as '0'

bit 5 **PxASDLF:** PSMCx Output F Auto-Shutdown Pin Level bit[1]

1 = When auto-shutdown is asserted, pin PSMCxF will drive logic '1'
0 = When auto-shutdown is asserted, pin PSMCxF will drive logic '0'

bit 4 **PxASDLE:** PSMCx Output E Auto-Shutdown Pin Level bit[1]

1 = When auto-shutdown is asserted, pin PSMCxE will drive logic '1'
0 = When auto-shutdown is asserted, pin PSMCxE will drive logic '0'

bit 3 **PxASDLD:** PSMCx Output D Auto-Shutdown Pin Level bit[1]

1 = When auto-shutdown is asserted, pin PSMCxD will drive logic '1'
0 = When auto-shutdown is asserted, pin PSMCxD will drive logic '0'

bit 2 **PxASDLC:** PSMCx Output C Auto-Shutdown Pin Level bit[1]

1 = When auto-shutdown is asserted, pin PSMCxC will drive logic '1'
0 = When auto-shutdown is asserted, pin PSMCxC will drive logic '0'

bit 1 **PxASDLB:** PSMCx Output B Auto-Shutdown Pin Level bit

1 = When auto-shutdown is asserted, pin PSMCxB will drive logic '1'
0 = When auto-shutdown is asserted, pin PSMCxB will drive logic '0'

bit 0 **PxASDLA:** PSMCx Output A Auto-Shutdown Pin Level bit

1 = When auto-shutdown is asserted, pin PSMCxA will drive logic '1'
0 = When auto-shutdown is asserted, pin PSMCxA will drive logic '0'

**Note 1:** These bits are not implemented on PSMC2.

## REGISTER 24-16: PSMCxASDS – PSMC AUTO-SHUTDOWN SOURCE REGISTER

| R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 |
|---------|-----|-----|-----|---------|---------|---------|-----|
| PxASDSIN | — | — | — | PxASDSC3 | PxASDSC2 | PxASDSC1 | — |
| 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        **PxASDSIN:** Auto-shutdown occurs on PSMCxIN pin
                 1 =    Auto-shutdown will occur when PSMCxIN pin goes true
                 0 =    PSMCxIN pin will not cause auto-shutdown

bit 6-4      **Unimplemented:** Read as '0'

bit 3        **PxASDSC3:** Auto-shutdown occurs on Comparator 3 output
                 1 =    Auto-shutdown will occur when Comparator 3 output goes true
                 0 =    Comparator 3 will not cause auto-shutdown

bit 2        **PxASDSC2:** Auto-shutdown occurs on Comparator 2 output
                 1 =    Auto-shutdown will occur when Comparator 2 output goes true
                 0 =    Comparator 2 will not cause auto-shutdown

bit 1        **PxASDSC1:** Auto-shutdown occurs on Comparator 1 output
                 1 =    Auto-shutdown will occur when Comparator 1 output goes true
                 0 =    Comparator 1 will not cause auto-shutdown

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

## REGISTER 24-17: PSMCxTMRL – PSMC TIME BASE COUNTER LOW REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxTMRL<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 | |

bit 7-0      **PSMCxTMRL<7:0>:** 16-bit PSMCx Time Base Counter Least Significant bits
                 =   PSMCxTMR<7:0>

## REGISTER 24-18: PSMCxTMRH – PSMC TIME BASE COUNTER HIGH REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxTMRH<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 | |

bit 7-0      **PSMCxTMRH<7:0>:** 16-bit PSMCx Time Base Counter Most Significant bits
                 =   PSMCxTMR<15:8>

**REGISTER 24-19: PSMCxPHL – PSMC PHASE COUNT LOW BYTE REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxPHL<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 | |

bit 7-0    **PSMCxPHL<7:0>:** 16-bit Phase Count Least Significant bits
   =   PSMCxPH<7:0>

**REGISTER 24-20: PSMCxPHH – PSMC PHASE COUNT HIGH BYTE REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxPHH<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 | |

bit 7-0    **PSMCxPHH<7:0>:** 16-bit Phase Count Most Significant bits
   =   PSMCxPH<15:8>

# PIC16(L)F1782/3

**REGISTER 24-21: PSMCxDCL – PSMC DUTY CYCLE COUNT LOW BYTE REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| PSMCxDCL<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 | |

bit 7-0 **PSMCxDCL<7:0>:** 16-bit Duty Cycle Count Least Significant bits
= PSMCxDC<7:0>

**REGISTER 24-22: PSMCxDCH – PSMC DUTY CYCLE COUNT HIGH REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| PSMCxDCH<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 | |

bit 7-0 **PSMCxDCH<7:0>:** 16-bit Duty Cycle Count Most Significant bits
= PSMCxDC<15:8>

**Preliminary** © 2011-2012 Microchip Technology Inc.

**REGISTER 24-23: PSMCxPRL – PSMC PERIOD COUNT LOW BYTE REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxPRL<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 | |

bit 7-0 **PSMCxPRL<7:0>:** 16-bit Period Time Least Significant bits
= PSMCxPR<7:0>

**REGISTER 24-24: PSMCxPRH – PSMC PERIOD COUNT HIGH BYTE REGISTER**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxPRH<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 | |

bit 7-0 **PSMCxPRH<7:0>:** 16-bit Period Time Most Significant bits
= PSMCxPR<15:8>

## REGISTER 24-25: PSMCxDBR – PSMC RISING EDGE DEAD-BAND TIME REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxDBR<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 | |

bit 7-0 **PSMCxDBR<7:0>:** Rising Edge Dead-Band Time
= Unsigned number of PSMCx psmc_clk clock periods in rising edge dead band

## REGISTER 24-26: PSMCxDBF – PSMC FALLING EDGE DEAD-BAND TIME REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxDBF<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 | |

bit 7-0 **PSMCxDBF<7:0>:** Falling Edge Dead-Band Time
= Unsigned number of PSMCx psmc_clk clock periods in falling edge dead band

## REGISTER 24-27: PSMCxFFA – PSMC FRACTIONAL FREQUENCY ADJUST REGISTER

| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-----|-----|-----|-----|---------|---------|---------|---------|
| — | — | — | — | | PSMCxFFA<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 Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-4 **Unimplemented:** Read as '0'
bit 3-0 **PSMCxFFA<3:0>:** Fractional Frequency Adjustment bits
= Unsigned number of fractional PSMCx psmc_clk clock periods to add to each period event time. The fractional time period = 1/(16*psmc_clk)

### REGISTER 24-28:  PSMCxBLKR – PSMC RISING EDGE BLANKING TIME REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxBLKR<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 | |

bit 7-0 **PSMCxBLKR<7:0>:** Rising Edge Blanking Time

= Unsigned number of PSMCx psmc_clk clock periods in rising edge blanking

### REGISTER 24-29:  PSMCxBLKF – PSMC FALLING EDGE BLANKING TIME REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | PSMCxBLKF<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 | |

bit 7-0 **PSMCxBLKF<7:0>:** Falling Edge Blanking Time bits

= Unsigned number of PSMCx psmc_clk clock periods in falling edge blanking

**REGISTER 24-30: PSMCxSTR0 – PSMC STEERING CONTROL REGISTER 0**

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| — | — | PxSTRF[2] | PxSTRE[2] | PxSTRD[2] | PxSTRC[2] | PxSTRB | PxSTRA |
| 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　　**Unimplemented:** Read as '0'

bit 5　　**PxSTRF**: PWM Steering PSMCxF Output Enable bit[2]

If PxMODE<3:0> = 0000 (Single-phase PWM):
1 = Single PWM output is active on pin PSMCxF
0 = Single PWM output is not active on pin PSMCxF. PWM drive is in inactive state

If PxMODE<3:0> = 0001 (Complementary Single-phase PWM):
1 = Complementary PWM output is active on pin PSMCxF
0 = Complementary PWM output is not active on pin PSMCxOUT5. PWM drive is in inactive state

IF PxMODE<3:0> = 1100 (3-phase Steering):[1]
1 = PSMCxD and PSMCxE are high. PSMCxA, PMSCxB, PSMCxC and PMSCxF are low.
0 = 3-phase output combination is not active

bit 4　　**PxSTRE**: PWM Steering PSMCxE Output Enable bit[2]

If PxMODE<3:0> = 000x (single-phase PWM or Complementary PWM):
1 = Single PWM output is active on pin PSMCxE
0 = Single PWM output is not active on pin PSMCxE. PWM drive is in inactive state

IF PxMODE<3:0> = 1100 (3-phase Steering):[1]
1 = PSMCxB and PSMCxE are high. PSMCxA, PMSCxC, PSMCxD and PMSCxF are low.
0 = 3-phase output combination is not active

bit 3　　**PxSTRD**: PWM Steering PSMCxD Output Enable bit[2]

If PxMODE<3:0> = 0000 (Single-phase PWM):
1 = Single PWM output is active on pin PSMCxD
0 = Single PWM output is not active on pin PSMCxD. PWM drive is in inactive state

If PxMODE<3:0> = 0001 (Complementary single-phase PWM):
1 = Complementary PWM output is active on pin PSMCxD
0 = Complementary PWM output is not active on pin PSMCxD. PWM drive is in inactive state

IF PxMODE<3:0> = 1100 (3-phase Steering):[1]
1 = PSMCxB and PSMCxC are high. PSMCxA, PMSCxD, PSMCxE and PMSCxF are low.
0 = 3-phase output combination is not active

bit 2　　**PxSTRC**: PWM Steering PSMCxC Output Enable bit[2]

If PxMODE<3:0> = 000x (Single-phase PWM or Complementary PWM):
1 = Single PWM output is active on pin PSMCxC
0 = Single PWM output is not active on pin PSMCxC. PWM drive is in inactive state

IF PxMODE<3:0> = 1100 (3-phase Steering):[1]
1 = PSMCxC and PSMCxF are high. PSMCxA, PMSCxB, PSMCxD and PMSCxE are low.
0 = 3-phase output combination is not active

**REGISTER 24-30: PSMCxSTR0 – PSMC STEERING CONTROL REGISTER 0**

bit 1          **PxSTRB**: PWM Steering PSMCxB Output Enable bit

<u>If PxMODE<3:0> = `0000` (Single-phase PWM):</u>
`1` = Single PWM output is active on pin PSMCxOUT1
`0` = Single PWM output is not active on pin PSMCxOUT1. PWM drive is in inactive state

<u>If PxMODE<3:0> = `0001` (Complementary Single-phase PWM):</u>
`1` = Complementary PWM output is active on pin PSMCxB
`0` = Complementary PWM output is not active on pin PSMCxB. PWM drive is in inactive state

<u>IF PxMODE<3:0> = `1100` (3-phase Steering):</u>**(1)**
`1` = PSMCxA and PSMCxF are high. PSMCxB, PMSCxC, PSMCxD and PMSCxE are low.
`0` = 3-phase output combination is not active

bit 0          **PxSTRA**: PWM Steering PSMCxA Output Enable bit

<u>If PxMODE<3:0> = `000x` (Single-phase PWM or Complementary PWM):</u>
`1` = Single PWM output is active on pin PSMCxA
`0` = Single PWM output is not active on pin PSMCxA. PWM drive is in inactive state

<u>IF PxMODE<3:0> = `1100` (3-phase Steering):</u>**(1)**
`1` = PSMCxA and PSMCxD are high. PSMCxB, PMSCxC, PSMCxE and PMSCxF are low.
`0` = 3-phase output combination is not active

**Note 1:** In 3-phase Steering mode, only one PSTRx bit should be set at a time. If more than one is set, then the lowest bit number steering combination has precedence.
      **2:** These bits are not implemented on PSMC2.

**REGISTER 24-31: PSMCxSTR1 – PSMC STEERING CONTROL REGISTER 1**

| R/W-0/0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---------|-----|-----|-----|-----|-----|---------|---------|
| PxSSYNC | — | — | — | — | — | PxLSMEN | PxHSMEN |
| 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 **PxSSYNC**: PWM Steering Synchronization bit

`1` = PWM outputs are updated on period boundary
`0` = PWM outputs are updated immediately

bit 6-2 **Unimplemented:** Read as '`0`'

bit 1 **PxLSMEN**: 3-Phase Steering Low Side Modulation Enable bit

PxMODE = `1100`:
`1` = Low side driver PSMCxB, PSMCxD and PSMCxF outputs are modulated according to PSMCxMDL when the output is high and driven low without modulation when the output is low.
`0` = PSMCxB, PSMCxD, and PSMCxF outputs are driven high and low by PSMCxSTR0 control without modulation.

PxMODE <> `1100`:
No effect on output

bit 0 **PxHSMEN**: 3-Phase Steering High Side Modulation Enable bit

PxMODE = `1100`:
`1` = High side driver PSMCxA, PSMCxC and PSMCxE outputs are modulated according to PSMCxMDL when the output is high and driven low without modulation when the output is low.
`0` = PSMCxA, PSMCxC and PSMCxE outputs are driven high and low by PSMCxSTR0 control without modulation.

PxMODE <> `1100`:
No effect on output

**Preliminary** © 2011-2012 Microchip Technology Inc.

## REGISTER 24-32: PSMCxINT – PSMC TIME BASE INTERRUPT CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| PxTOVIE | PxTPHIE | PxTDCIE | PxTPRIE | PxTOVIF | PxTPHIF | PxTDCIF | PxTPRIF |
| 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 **PxTOVIE**: PSMC Time Base Counter Overflow Interrupt Enable bit

1 = Time base counter overflow interrupts are enabled
0 = Time base counter overflow interrupts are disabled

bit 6 **PxTPHIE**: PSMC Time Base Phase Interrupt Enable bit

1 = Time base phase match interrupts are enabled
0 = Time base phase match interrupts are disabled

bit 5 **PxTDCIE**: PSMC Time Base Duty Cycle Interrupt Enable bit

1 = Time base duty cycle match interrupts are enabled
0 = Time base duty cycle match interrupts are disabled

bit 4 **PxTPRIE**: PSMC Time Base Period Interrupt Enable bit

1 = Time base period match interrupts are enabled
0 = Time base period match Interrupts are disabled

bit 3 **PxTOVIF**: PSMC Time Base Counter Overflow Interrupt Flag bit

1 = The 16-bit PSMCxTMR has overflowed from FFFFh to 0000h
0 = The 16-bit PSMCxTMR counter has not overflowed

bit 2 **PxTPHIF**: PSMC Time Base Phase Interrupt Flag bit

1 = The 16-bit PSMCxTMR counter has matched PSMCxPH<15:0>
0 = The 16-bit PSMCxTMR counter has not matched PSMCxPH<15:0>

bit 1 **PxTDCIF**: PSMC Time Base Duty Cycle Interrupt Flag bit

1 = The 16-bit PSMCxTMR counter has matched PSMCxDC<15:0>
0 = The 16-bit PSMCxTMR counter has not matched PSMCxDC<15:0>

bit 0 **PxTPRIF**: PSMC Time Base Period Interrupt Flag bit

1 = The 16-bit PSMCxTMR counter has matched PSMCxPR<15:0>
0 = The 16-bit PSMCxTMR counter has not matched PSMCxPR<15:0>

# PIC16(L)F1782/3

## TABLE 24-5: SUMMARY OF REGISTERS ASSOCIATED WITH PSMC

| Name | Bit7 | Bit6 | Bit5 | Bit4 | BIt3 | Bit2 | Bit1 | Bit0 | Register on Page |
|------|------|------|------|------|------|------|------|------|------------------|
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| ODCONC | ODC7 | ODC6 | ODC5 | ODC4 | ODC3 | ODC2 | ODC1 | ODC0 | 134 |
| PIE4 | — | — | PSMC2TIE | PSMC1TIE | — | — | PSMC2SIE | PSMC1SIE | 87 |
| PIR4 | — | — | PSMC2TIF | PSMC1TIF | — | — | PSMC2SIF | PSMC1SIF | 90 |
| PSMCxASDC | PxASE | PxASDEN | PxARSEN | — | — | — | — | PxASDOV | 242 |
| PSMCxASDL | — | — | PxASDLF[1] | PxASDLE[1] | PxASDLD[1] | PxASDLC[1] | PxASDLB | PxASDLA | 243 |
| PSMCxASDS | PxASDSIN | — | — | — | PxASDSC3 | PxASDSC2 | PxASDSC1 | — | 244 |
| PSMCxBLKF | PSMCxBLKF<7:0> | | | | | | | | 249 |
| PSMCxBLKR | PSMCxBLKR<7:0> | | | | | | | | 249 |
| PSMCxBLNK | — | — | PxFEBM1 | PxFEBM0 | — | — | PxREBM1 | PxREBM0 | 237 |
| PSMCxCLK | — | — | PxCPRE<1:0> | | — | — | PxCSRC<1:0> | | 236 |
| PSMCxCON | PSMCxEN | PSMCxLD | PxDBFE | PxDBRE | PxMODE<3:0> | | | | 233 |
| PSMCxDBF | PSMCxDBF<7:0> | | | | | | | | 248 |
| PSMCxDBR | PSMCxDBR<7:0> | | | | | | | | 248 |
| PSMCxDCH | PSMCxDC<15:8> | | | | | | | | 246 |
| PSMCxDCL | PSMCxDC<7:0> | | | | | | | | 246 |
| PSMCxDCS | PxDCSIN | — | — | — | PxDCSC3 | PxDCSC2 | PxDCSC1 | PxDCST | 240 |
| PSMCxFEBS | PxFEBSIN | — | — | — | PxFEBSC3 | PxFEBSC2 | PxFEBSC1 | — | 238 |
| PSMCxFFA | — | — | — | — | PSMCxFFA<3:0> | | | | 248 |
| PSMCxINT | PxTOVIE | PxTPHIE | PxTDCIE | PxTPRIE | PxTOVIF | PxTPHIF | PxTDCIF | PxTPRIF | 253 |
| PSMCxMDL | PxMDLEN | PxMDLPOL | PxMDLBIT | — | PxMSRC<3:0> | | | | 234 |
| PSMCxOEN | — | — | PxOEF[1] | PxOEE[1] | PxOED[1] | PxOEC[1] | PxOEB | PxOEA | 236 |
| PSMCxPHH | PSMCxPH<15:8> | | | | | | | | 245 |
| PSMCxPHL | PSMCxPH<7:0> | | | | | | | | 245 |
| PSMCxPHS | PxPHSIN | — | — | — | PxPHSC3 | PxPHSC2 | PxPHSC1 | PxPHST | 239 |
| PSMCxPOL | — | PxPOLIN | PxPOLF[1] | PxPOLE[1] | PxPOLD[1] | PxPOLC[1] | PxPOLB | PxPOLA | 237 |
| PSMCxPRH | PSMCxPR<15:8> | | | | | | | | 247 |
| PSMCxPRL | PSMCxPR<7:0> | | | | | | | | 247 |
| PSMCxPRS | PxPRSIN | — | — | — | PxPRSC3 | PxPRSC2 | PxPRSC1 | PxPRST | 241 |
| PSMCxREBS | PxREBSIN | — | — | — | PxREBSC3 | PxREBSC2 | PxREBSC1 | — | 238 |
| PSMCxSTR0 | — | — | PxSTRF[1] | PxSTRE[1] | PxSTRD[1] | PxSTRC[1] | PxSTRB | PxSTRA | 250 |
| PSMCxSTR1 | PxSSYNC | — | — | — | — | — | PxLSMEN | PxHSMEN | 252 |
| PSMCxSYNC | — | — | — | — | — | — | PxSYNC<1:0> | | 235 |
| PSMCxTMRH | PSMCxTMR<15:8> | | | | | | | | 244 |
| PSMCxTMRL | PSMCxTMR<7:0> | | | | | | | | 244 |
| SLRCONC | SLRC7 | SLRC6 | SLRC5 | SLRC4 | SLRC3 | SLCR2 | SRC1 | SLRC0 | 134 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by PSMC module.
**Note 1:** Unimplemented in PSMC2.

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 25.0 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 2 standard Capture/Compare/PWM modules (CCP1 and CCP2).

The Capture and Compare functions are identical for all CCP 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 CCPx module. 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. |

# PIC16(L)F1782/3

## 25.1 Capture Mode

The Capture mode function described in this section is available and identical for all CCP modules.

Capture mode makes use of the 16-bit Timer1 resource. When an event occurs on the CCPx pin, the 16-bit CCPRxH:CCPRxL register pair captures and stores the 16-bit value of the TMR1H:TMR1L 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 Interrupt Request Flag bit CCPxIF of the PIRx 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 25-1 shows a simplified diagram of the Capture operation.

### 25.1.1 CCP PIN CONFIGURATION

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

Also, the CCP2 pin function can be moved to alternative pins using the APFCON register. Refer to **Section 13.1 "Alternate Pin Function"** for more details.

| Note: | If the CCPx pin is configured as an output, a write to the port can cause a capture condition. |
|---|---|

**FIGURE 25-1:** CAPTURE MODE OPERATION BLOCK DIAGRAM



### 25.1.2 TIMER1 MODE RESOURCE

Timer1 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 22.0 "Timer1 Module with Gate Control"** for more information on configuring Timer1.

### 25.1.3 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 PIEx register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the PIRx register following any change in Operating mode.

| Note: | Clocking Timer1 from the system clock (F$_{OSC}$) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (F$_{OSC}$/4) or from an external clock source. |
|---|---|

### 25.1.4 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. Equation 25-1 demonstrates the code to perform this function.

**EXAMPLE 25-1:** CHANGING BETWEEN CAPTURE PRESCALERS

```
BANKSEL CCPxCON      ;Set Bank bits to point
                     ;to CCPxCON
CLRF    CCPxCON      ;Turn CCP module off
MOVLW   NEW_CAPT_PS  ;Load the W reg with
                     ;the new prescaler
                     ;move value and CCP ON
MOVWF   CCPxCON      ;Load CCPxCON with this
                     ;value
```

**Preliminary**

© 2011-2012 Microchip Technology Inc.

### 25.1.5 CAPTURE DURING SLEEP

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock (F$_{OSC}$/4), or by an external clock source.

When Timer1 is clocked by F$_{OSC}$/4, Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep when Timer1 is clocked by an external clock source.

### 25.1.6 ALTERNATE PIN LOCATIONS

This module incorporates I/O pins that can be moved to other locations with the use of the alternate pin function register APFCON. To determine which pins can be moved and what their default locations are upon a Reset, see **Section 13.1 "Alternate Pin Function"** for more information.

---

## 25.2 Compare Mode

The Compare mode function described in this section is available and identical for al CCP modules.

Compare mode makes use of the 16-bit Timer1 resource. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared·against the 16-bit value of the TMR1H:TMR1L 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 an Auto-conversion 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 25-2 shows a simplified diagram of the compare operation.

**FIGURE 25-2: COMPARE MODE OPERATION BLOCK DIAGRAM**
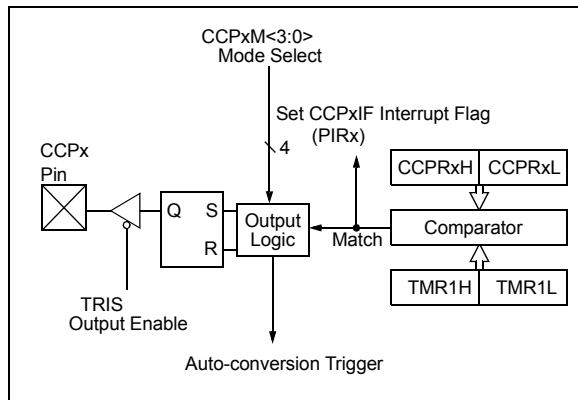


### 25.2.1 CCPX PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the associated TRIS bit.

The CCP2 pin function can be moved to alternate pins using the APFCON register (Register 13-1). Refer to **Section 13.1 "Alternate Pin Function"** 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. |
|---|---|

### 25.2.2 TIMER1 MODE RESOURCE

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See **Section 22.0 "Timer1 Module with Gate Control"** for more information on configuring Timer1.

| Note: | Clocking Timer1 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, TImer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source. |
|---|---|

### 25.2.3 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).

### 25.2.4 AUTO-CONVERSION TRIGGER

When Auto-conversion Trigger mode is chosen (CCPxM<3:0> = 1011), the CCPx module does the following:

- Resets Timer1
- Starts an ADC conversion if ADC is enabled

The CCPx module does not assert control of the CCPx pin in this mode.

The Auto-conversion Trigger output of the CCP occurs immediately upon a match between the TMR1H, TMR1L register pair and the CCPRxH, CCPRxL register pair. The TMR1H, TMR1L register pair is not reset until the next rising edge of the Timer1 clock. The Auto-conversion Trigger output starts an A/D conversion (if the A/D module is enabled). This allows the CCPRxH, CCPRxL register pair to effectively provide a 16-bit programmable period register for Timer1.

Refer to **Section 17.2.5 "Auto-Conversion Trigger"** for more information.

| Note 1: | The Auto-conversion Trigger from the CCP module does not set interrupt flag bit TMR1IF of the PIR1 register. |
|---|---|
| 2: | Removing the match condition by changing the contents of the CCPRxH and CCPRxL register pair, between the clock edge that generates the Auto-conversion Trigger and the clock edge that generates the Timer1 Reset, will preclude the Reset from occurring. |

**Preliminary** © 2011-2012 Microchip Technology Inc.

### 25.2.5 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.

### 25.2.6 ALTERNATE PIN LOCATIONS

This module incorporates I/O pins that can be moved to other locations with the use of the alternate pin function register APFCON. To determine which pins can be moved and what their default locations are upon a Reset, see **Section 13.1 "Alternate Pin Function"** for more information.

## 25.3 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 25-3 shows a typical waveform of the PWM signal.

### 25.3.1 STANDARD PWM OPERATION

The standard PWM function described in this section is available and identical for all CCP 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 registers
- T2CON registers
- CCPRxL registers
- CCPxCON registers

Figure 25-4 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 25-3:** **CCP PWM OUTPUT SIGNAL**
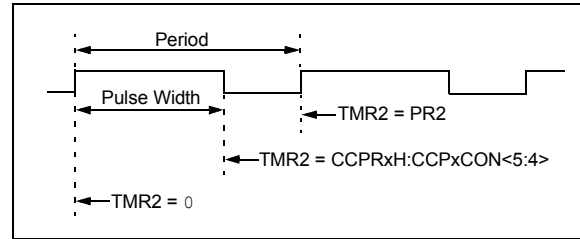


**FIGURE 25-4:** **SIMPLIFIED PWM BLOCK DIAGRAM**



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

**2:** In PWM mode, CCPR1H is a read-only register.

**Preliminary** © 2011-2012 Microchip Technology Inc.
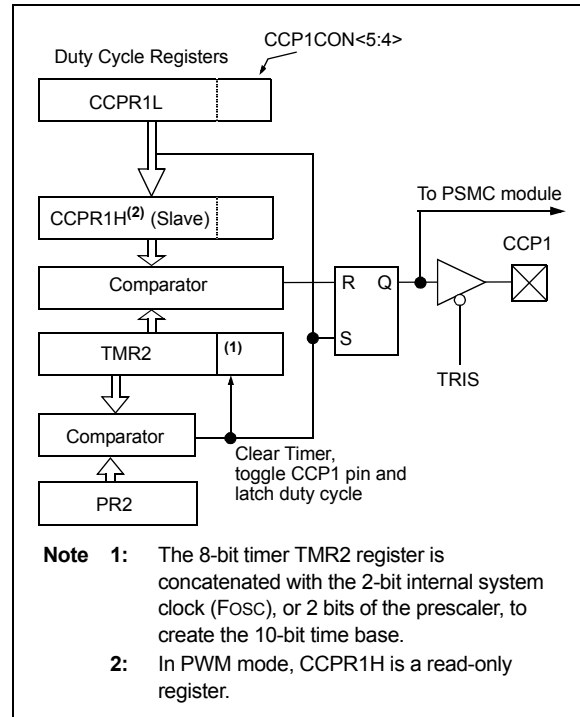
### 25.3.2    SETUP FOR PWM OPERATION

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 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 DCxBx bits of the CCPxCON register, with the PWM duty cycle value.
5. Configure and start Timer2:
   - Clear the TMR2IF interrupt flag bit of the PIRx register. See Note 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 below.
   - Enable the CCPx pin output driver by clearing the associated TRIS bit.

> **Note:** 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 6 may be ignored.

### 25.3.3    TIMER2 TIMER RESOURCE

The PWM standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

### 25.3.4    PWM PERIOD

The PWM period is specified by the PR2 register of Timer2. The PWM period can be calculated using the formula of Equation 25-1.

**EQUATION 25-1:    PWM PERIOD**

$$PWM\ Period = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

**Note 1:** $T_{OSC}$ = 1/$F_{OSC}$

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 23.1 "Timer2 Operation"**) is not used in the determination of the PWM frequency.

### 25.3.5    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 25-2 is used to calculate the PWM pulse width.

Equation 25-3 is used to calculate the PWM duty cycle ratio.

**EQUATION 25-2:    PULSE WIDTH**

$$Pulse\ Width = (CCPRxL{:}CCPxCON{<}5{:}4{>}) \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

**EQUATION 25-3:    DUTY CYCLE RATIO**

$$Duty\ Cycle\ Ratio = \frac{(CCPRxL{:}CCPxCON{<}5{:}4{>})}{4(PR2 + 1)}$$

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 ($F_{OSC}$), or 2 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 25-4).

### 25.3.6 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 25-4.

**EQUATION 25-4: PWM RESOLUTION**

$$Resolution = \frac{log[4(PR2 + 1)]}{log(2)} \ bits$$

**Note:** If the pulse width value is greater than the period the assigned PWM pin(s) will remain unchanged.

**TABLE 25-1: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (F$_{OSC}$ = 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 25-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (F$_{OSC}$ = 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 |

**Preliminary**

### 25.3.7 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.

### 25.3.8 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 6.0 "Oscillator Module (with Fail-Safe Clock Monitor)"** for additional details.

### 25.3.9 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

**TABLE 25-3: SUMMARY OF 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 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| APFCON | C2OUTSEL | CC1PSEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 119 |
| CCP1CON | P1M<1:0> | | DC1B<1:0> | | CCP1M<3:0> | | | | 264 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 84 |
| PIE2 | OSFIE | C2IE | C1IE | EEIE | BCL1IE | — | C3IE | CCP2IE | 86 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 88 |
| PIR2 | OSFIF | C2IF | C1IF | EEIF | BCL1IF | — | C3IF | CCP2IF | 89 |
| PR2 | Timer2 Period Register | | | | | | | | 197* |
| T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | 199 |
| TMR2 | Timer2 Module Register | | | | | | | | 197 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 122 |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used by the PWM.
* Page provides register information.

## 25.4 CCP Control Register

**REGISTER 25-1:** **CCPxCON: CCPx CONTROL REGISTER**

| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| — | — | 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       **Unimplemented:** Read as '0'

bit 5-4       **DCxB<1:0>:** PWM Duty Cycle Least Significant bits

<u>Capture mode:</u>
Unused

<u>Compare mode:</u>
Unused

<u>PWM mode:</u>
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0       **CCPxM<3:0>:** CCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets CCPx 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 (set CCPxIF)
1001 = Compare mode: clear output on compare match (set CCPxIF)
1010 = Compare mode: generate software interrupt only
1011 = Compare mode: Auto-conversion Trigger (sets CCPxIF bit (CCP2), starts A/D conversion if A/D module is enabled)[1]

11xx = PWM mode

## 26.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

### 26.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

• Serial Peripheral Interface (SPI)
• Inter-Integrated Circuit ($I^2C$™)

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 26-1 is a block diagram of the SPI interface module.

**FIGURE 26-1: MSSP BLOCK DIAGRAM (SPI MODE)**

# PIC16(L)F1782/3

The I$^2$C 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 26-2 is a block diagram of the I$^2$C interface module in Master mode. Figure 26-3 is a diagram of the I$^2$C interface module in Slave mode.

**FIGURE 26-2:**     **MSSP BLOCK DIAGRAM (I$^2$C™ MASTER MODE)**

**Preliminary**
© 2011-2012 Microchip Technology Inc.
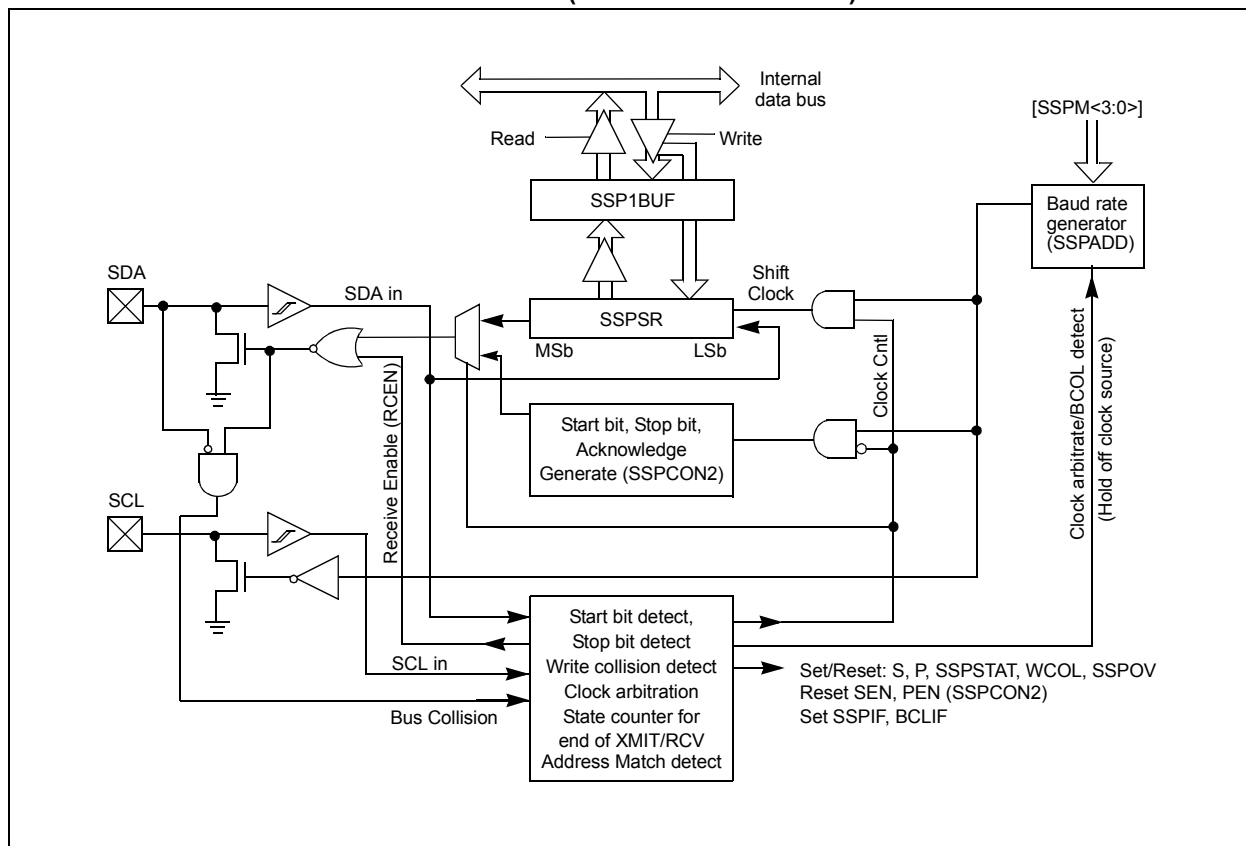
**FIGURE 26-3:** **MSSP BLOCK DIAGRAM (I²C™ SLAVE MODE)**
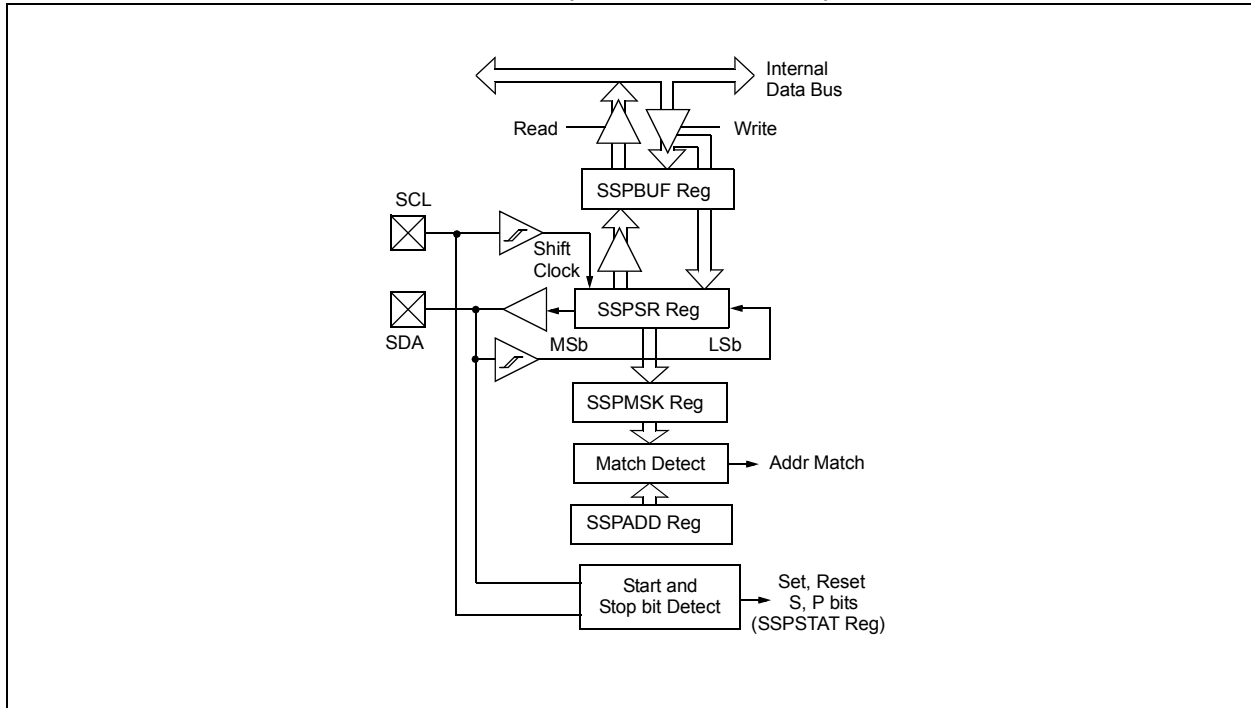
## 26.2 SPI Mode Overview

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 26-1 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 26-4 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 26-5 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 while the master device is sending out the MSb from its shift register (on

its SDO pin) and the slave device is reading this bit and saving it as the LSb of its shift register, that the slave device is also sending out the MSb from its shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its shift register.

After 8 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 26-4:** **SPI MASTER AND MULTIPLE SLAVE CONNECTION**



### 26.2.1 SPI MODE REGISTERS

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

- MSSP STATUS register (SSPSTAT)
- MSSP Control register 1 (SSPCON1)
- MSSP Control register 3 (SSPCON3)
- MSSP Data Buffer register (SSPBUF)
- MSSP Address register (SSPADD)
- MSSP Shift register (SSPSR)
  (Not directly accessible)

SSPCON1 and SSPSTAT are the control and STATUS registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

In one SPI master mode, SSPADD can be loaded with a value used in the Baud Rate Generator. More information on the Baud Rate Generator is available in **Section 26.7 "Baud Rate Generator"**.

SSPSR is the shift register used for shifting data in and out. SSPBUF provides indirect access to the SSPSR register. SSPBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPSR and SSPBUF together create a buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

## 26.2.2 SPI MODE OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

To enable the serial port, SSP Enable bit, SSPEN of the SSPCON1 register, must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPCONx registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI 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
- $\overline{SS}$ must have corresponding TRIS bit set

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

The MSSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready. Once the 8 bits of data have been received, that byte is moved to the SSPBUF register. Then, the Buffer Full Detect bit, BF of the SSPSTAT register, and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the write collision detect bit WCOL of the SSPCON1 register, will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPBUF register to complete successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. The Buffer Full bit, BF of the SSPSTAT register, indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur.

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

**FIGURE 26-5:     SPI MASTER/SLAVE CONNECTION**
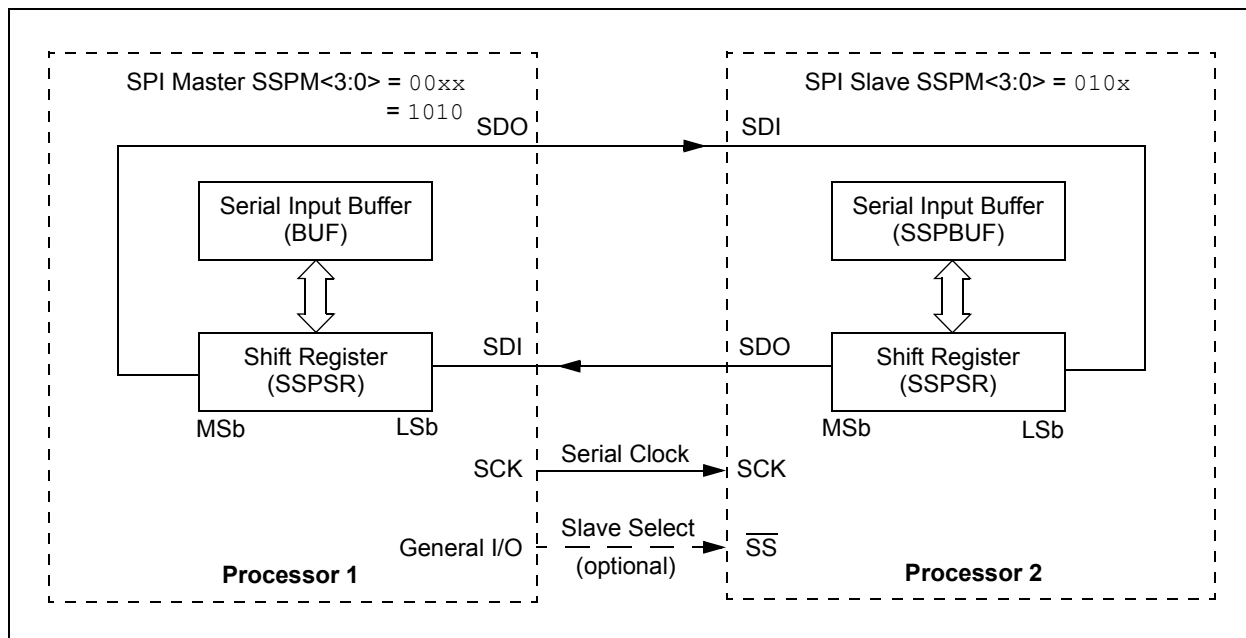
### 26.2.3 SPI MASTER MODE

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 26-5) is to broadcast data by the software protocol.

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

The clock polarity is selected by appropriately programming the CKP bit of the SSPCON1 register and the CKE bit of the SSPSTAT register. This then, would give waveforms for SPI communication as shown in Figure 26-6, Figure 26-8 and Figure 26-9, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- F$_{OSC}$/4 (or T$_{CY}$)
- F$_{OSC}$/16 (or 4 * T$_{CY}$)
- F$_{OSC}$/64 (or 16 * T$_{CY}$)
- Timer2 output/2
- Fosc/(4 * (SSPADD + 1))

Figure 26-6 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 SSPBUF is loaded with the received data is shown.

**FIGURE 26-6:** SPI MODE WAVEFORM (MASTER MODE)
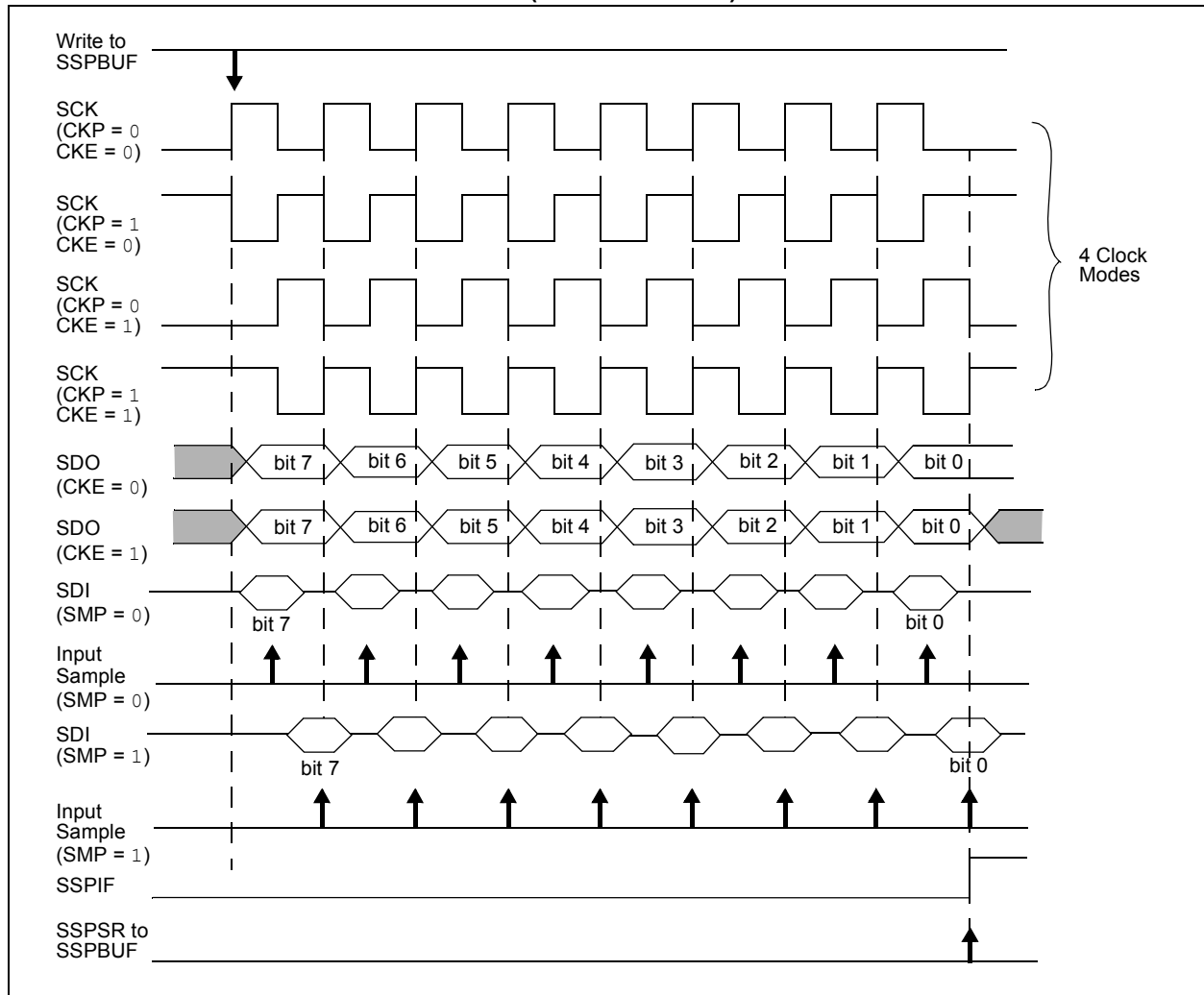
# PIC16(L)F1782/3

## 26.2.4 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 SSPCON1 register.

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

While in Sleep mode, the slave can transmit/receive data. 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.

### 26.2.4.1 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 26-7 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 SSPCON3 register will enable writes to the SSPBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

## 26.2.5 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 $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 0100).

When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven.

When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

| | |
|---|---|
| **Note 1:** | When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPCON1<3:0> = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to V<sub>DD</sub>. |
| **2:** | When the SPI is used in Slave mode with CKE set; the user must enable $\overline{SS}$ pin control. |
| **3:** | While operated in SPI Slave mode the SMP bit of the SSPSTAT register must remain clear. |

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the $\overline{SS}$ pin to a high level or clearing the SSPEN bit.

**FIGURE 26-7:** **SPI DAISY-CHAIN CONNECTION**



**FIGURE 26-8:** **SLAVE SELECT SYNCHRONOUS WAVEFORM**
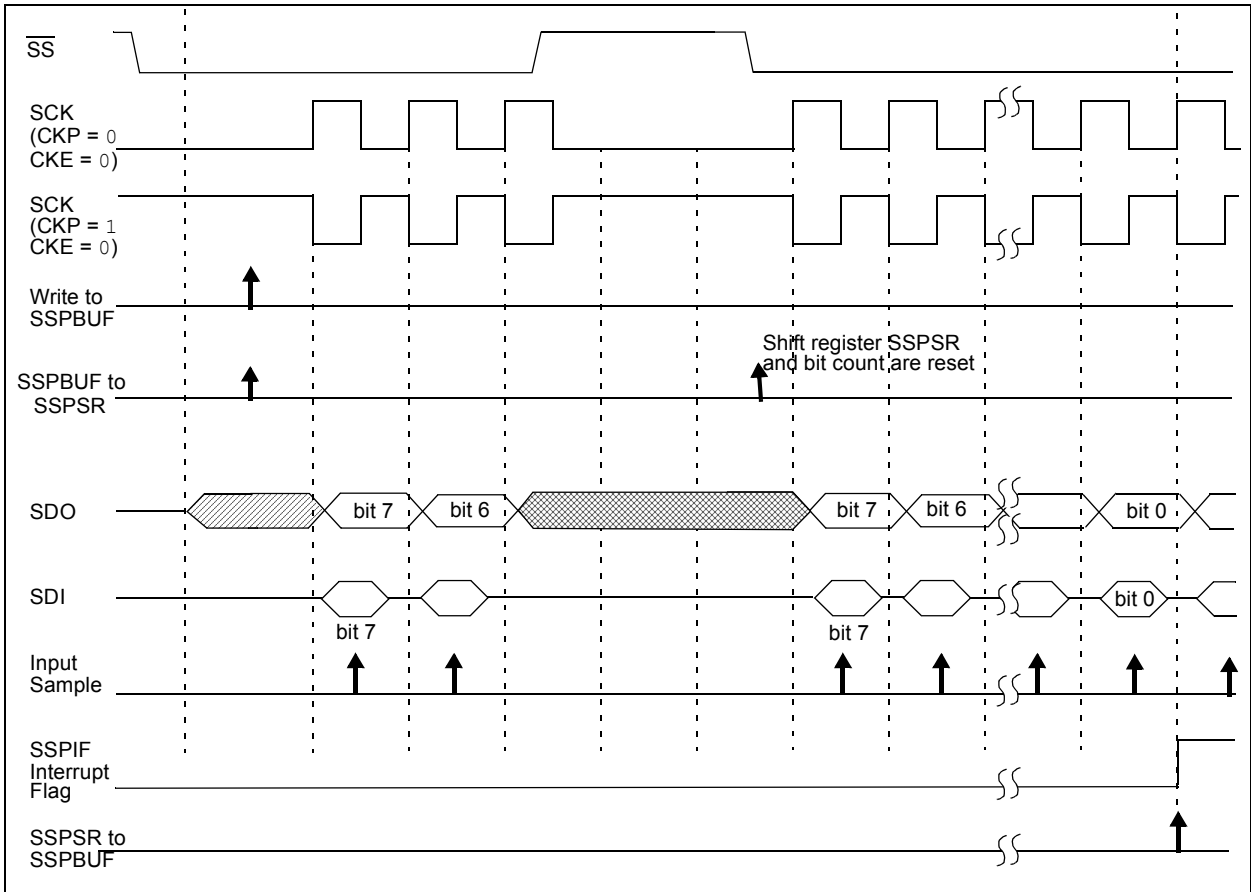
# PIC16(L)F1782/3

**FIGURE 26-9:** SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)
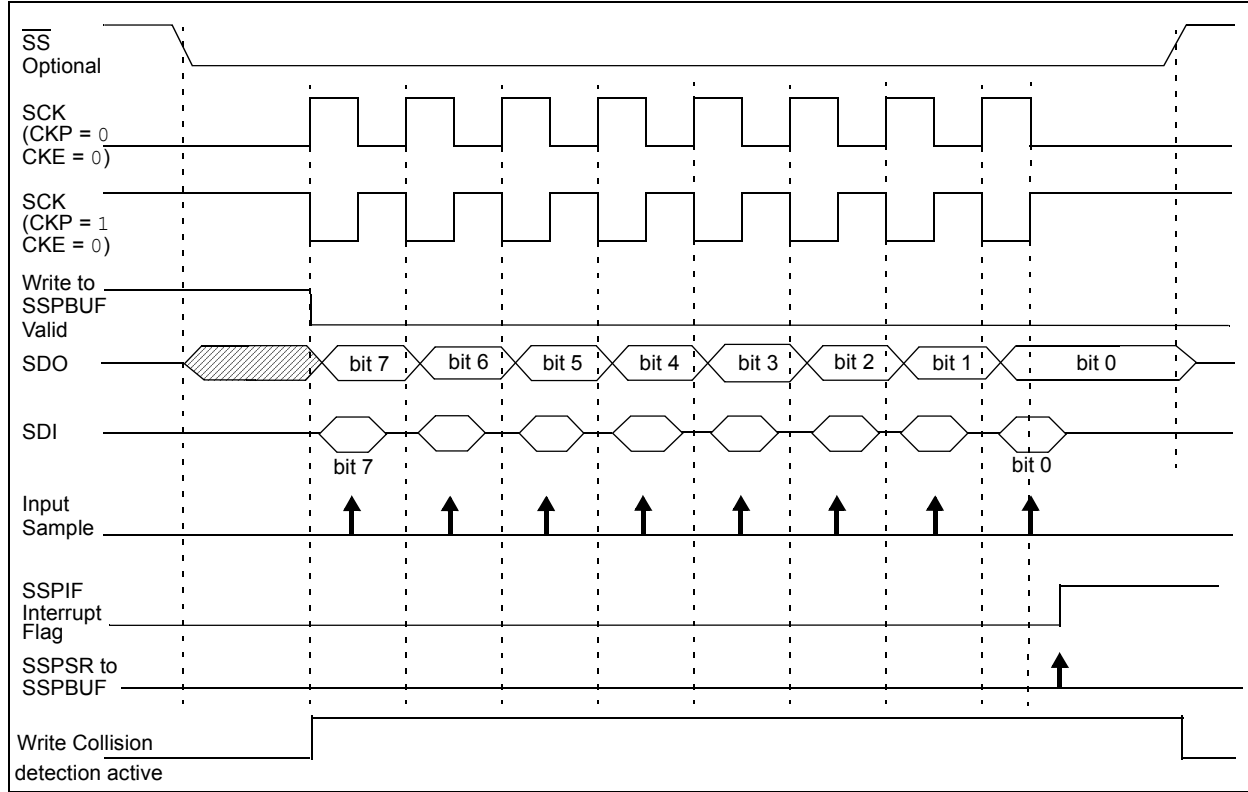


**FIGURE 26-10:** SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)

**Preliminary**
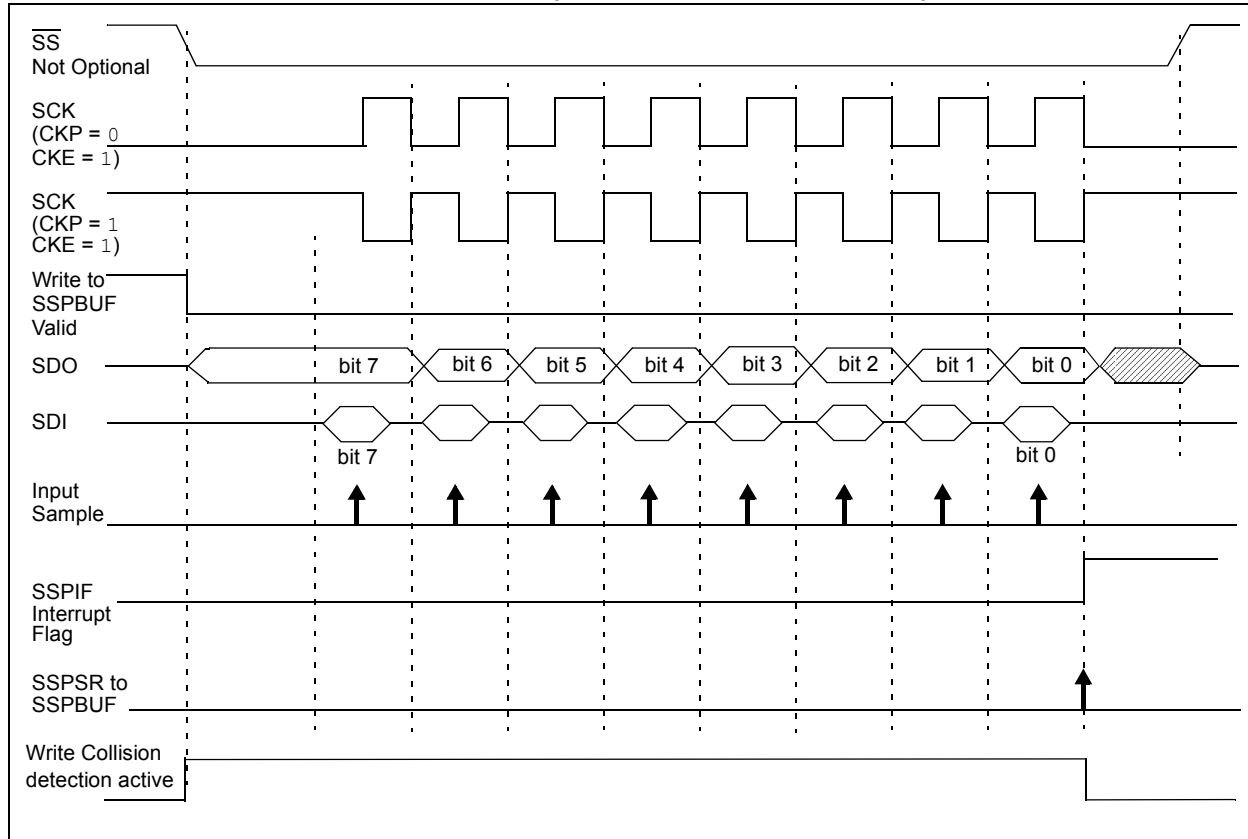
## 26.2.6    SPI OPERATION IN SLEEP MODE

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 8 bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

### TABLE 26-1:    SUMMARY OF 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 | ANSA7 | — | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 123 |
| APFCON | C2OUTSEL | CCP1SEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 119 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| SSPBUF | Synchronous Serial Port Receive Buffer/Transmit Register | | | | | | | | 269* |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 313 |
| SSPCON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 315 |
| SSPSTAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 312 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 122 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISA0 | 133 |

**Legend:**    — = Unimplemented location, read as '0'. Shaded cells are not used by the MSSP in SPI mode.
   *    Page provides register information.
**Note   1:**    PIC16(L)F1783 only.

# PIC16(L)F1782/3

## 26.3 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 26-11 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 26-11 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 26-11: I²C MASTER/ SLAVE CONNECTION
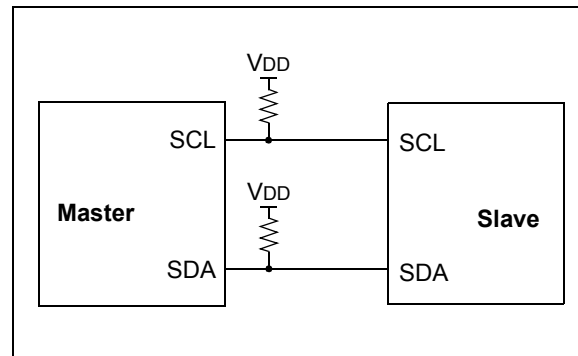


The Acknowledge bit (ACK) is an active-low signal, which holds the SDA line low to indicate to the transmitter that the slave device has received the transmitted data and is ready to receive more.

The transition of a data bit 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.

**Preliminary** © 2011-2012 Microchip Technology Inc.

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 gives 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.

## 26.3.1    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.

## 26.3.2    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.

## 26.4    I2C MODE OPERATION

All MSSP I2C communication is byte oriented and shifted out MSb first. Six SFR registers and 2 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.

### 26.4.1    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.

### 26.4.2    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 Philips I2C specification.

### 26.4.3    SDA AND SCL PINS

Selection of any I2C mode with the SSPEN 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.

### 26.4.4    SDA HOLD TIME

The hold time of the SDA pin is selected by the SDAHT bit of the SSPCON3 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 26-2:    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 SSPADD. |
| Write Request | Slave receives a matching address with R/$\overline{W}$ bit clear, and is ready to clock in data. |
| Read Request | Master sends an address byte with the R/$\overline{W}$ 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 hold 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. |

## 26.4.5 START CONDITION

The I²C specification defines a Start condition as a transition of SDA from a high to a 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 26-10 shows wave forms 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.

## 26.4.6 STOP CONDITION

A Stop condition is a transition of the SDA line from 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.

## 26.4.7 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/$\overline{\text{W}}$ bit set. The slave logic will then hold the clock and prepare to clock out data.

After a full match with R/$\overline{\text{W}}$ clear in 10-bit mode, a prior match flag is set and maintained. Until a Stop condition, a high address with R/$\overline{\text{W}}$ clear, or high address match fails.

## 26.4.8 START/STOP CONDITION INTERRUPT MASKING

The SCIE and PCIE bits of the SSPCON3 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 26-12: I²C START AND STOP CONDITIONS**
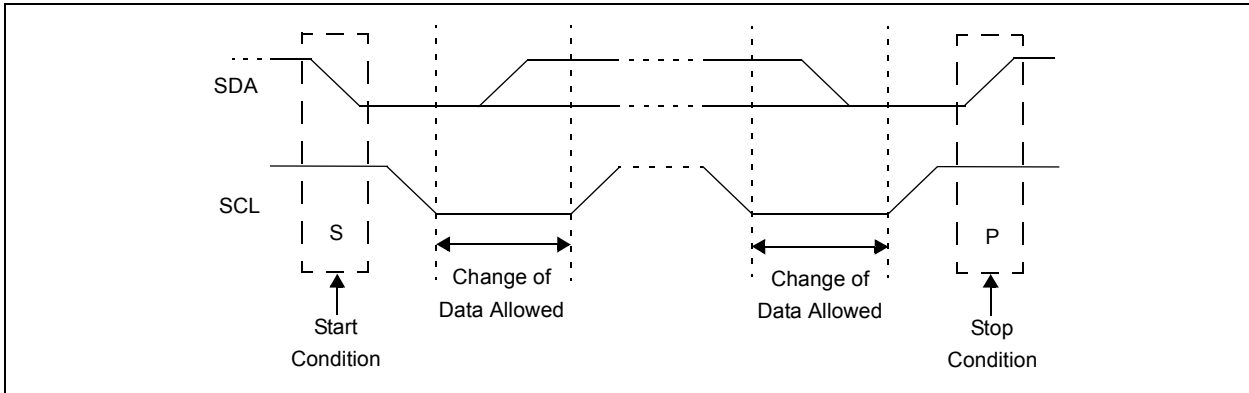


**FIGURE 26-13: I²C RESTART CONDITION**
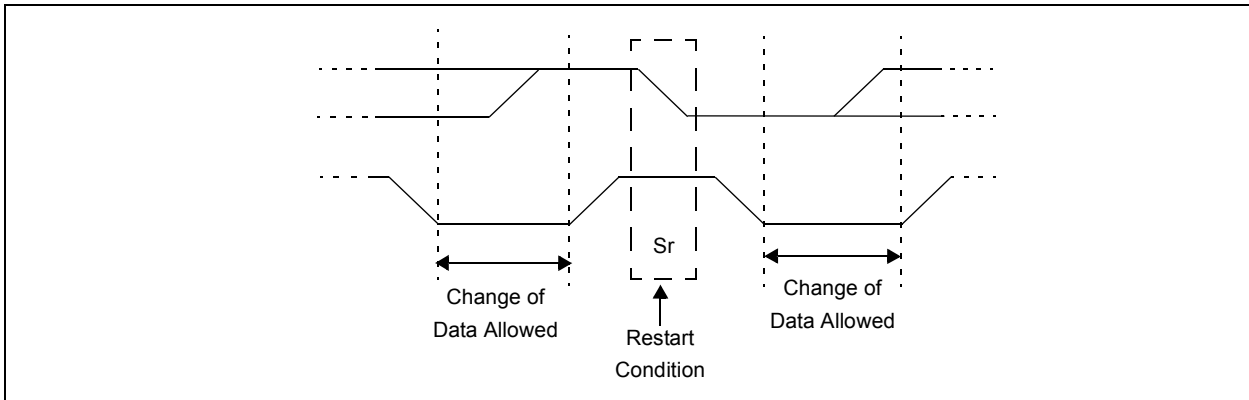
## 26.4.9 ACKNOWLEDGE SEQUENCE

The 9th SCL pulse for any transferred byte in I$^2$C 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 ($\overline{\text{ACK}}$) 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 $\overline{\text{ACK}}$ is placed in the ACKSTAT bit of the SSPCON2 register.

Slave software, when the AHEN and DHEN bits are set, allow the user to set the $\overline{\text{ACK}}$ value sent back to the transmitter. The ACKDT bit of the SSPCON2 register is set/cleared to determine the response.

Slave hardware will generate an $\overline{\text{ACK}}$ response if the AHEN and DHEN bits of the SSPCON3 register are clear.

There are certain conditions where an $\overline{\text{ACK}}$ will not be sent by the slave. If the BF bit of the SSPSTAT register or the SSPOV bit of the SSPCON1 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 SSPCON3 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.

## 26.5 I$^2$C SLAVE MODE OPERATION

The MSSP Slave mode operates in one of four modes selected in the SSPM bits of SSPCON1 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.

### 26.5.1 SLAVE MODE ADDRESSES

The SSPADD register (Register 26-6) 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 SSPBUF 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 SSP Mask register (Register 26-5) affects the address matching process. See **Section 26.5.9 "SSP Mask Register"** for more information.

#### 26.5.1.1 I$^2$C 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.

#### 26.5.1.2 I$^2$C 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 SSPADD register.

After the acknowledge of the high byte the UA bit is set and SCL is held low until the user updates SSPADD with the low address. The low address byte is clocked in and all 8 bits are compared to the low address value in SSPADD. Even if there is not an address match; SSPIF and UA are set, and SCL is held low until SSPADD is updated to receive a high byte again. When SSPADD 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/$\overline{\text{W}}$ 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.

## 26.5.2 SLAVE RECEPTION

When the R/$\overline{\text{W}}$ bit of a matching received address byte is clear, the R/$\overline{\text{W}}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and acknowledged.

When the overflow condition exists for a received address, then not Acknowledge is given. An overflow condition is defined as either bit BF of the SSPSTAT register is set, or bit SSPOV of the SSPCON1 register is set. The BOEN bit of the SSPCON3 register modifies this operation. For more information see Register 26-4.

An MSSP interrupt is generated for each transferred data byte. Flag bit, SSPIF, must be cleared by software.

When the SEN bit of the SSPCON2 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 SSPCON1 register, except sometimes in 10-bit mode. See **Section 26.2.3 "SPI Master Mode"** for more detail.

### 26.5.2.1 7-bit Addressing Reception

This section describes a standard sequence of events for the MSSP module configured as an I$^2$C Slave in 7-bit Addressing mode. All decisions made by hardware or software and their effect on reception. Figure 26-13 and Figure 26-14 is used as a visual reference for this description.

This is a step by step process of what typically must be done to accomplish I$^2$C communication.

1. Start bit detected.
2. S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{\text{W}}$ bit clear is received.
4. The slave pulls SDA low sending an $\overline{\text{ACK}}$ to the master, and sets SSPIF bit.
5. Software clears the SSPIF bit.
6. Software reads received address from SSPBUF 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 $\overline{\text{ACK}}$ to the master, and sets SSPIF bit.
10. Software clears SSPIF.
11. Software reads the received byte from SSPBUF clearing BF.
12. Steps 8-12 are repeated for all received bytes from the master.
13. Master sends Stop condition, setting P bit of SSPSTAT, and the bus goes idle.

### 26.5.2.2 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 $\overline{\text{ACK}}$ 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 I$^2$C communication. Figure 26-15 displays a module using both address and data holding. Figure 26-16 includes the operation with the SEN bit of the SSPCON2 register set.

1. S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
2. Matching address with R/$\overline{\text{W}}$ 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 SSPCON3 register to determine if the SSPIF was after or before the $\overline{\text{ACK}}$.
5. Slave reads the address value from SSPBUF, clearing the BF flag.
6. Slave sets $\overline{\text{ACK}}$ value clocked out to the master by setting ACKDT.
7. Slave releases the clock by setting CKP.
8. SSPIF is set after an $\overline{\text{ACK}}$, not after a NACK.
9. If SEN = 1 the slave hardware will stretch the clock after the $\overline{\text{ACK}}$.
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 SSPCON3 to determine the source of the interrupt.
13. Slave reads the received data from SSPBUF clearing BF.
14. Steps 7-14 are the same for each received data byte.
15. Communication is ended by either the slave sending an $\overline{\text{ACK}}$ = 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.

# PIC16(L)F1782/3

**FIGURE 26-14:** I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 0, DHEN = 0)



**Preliminary** © 2011-2012 Microchip Technology Inc.

**FIGURE 26-15:** I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)
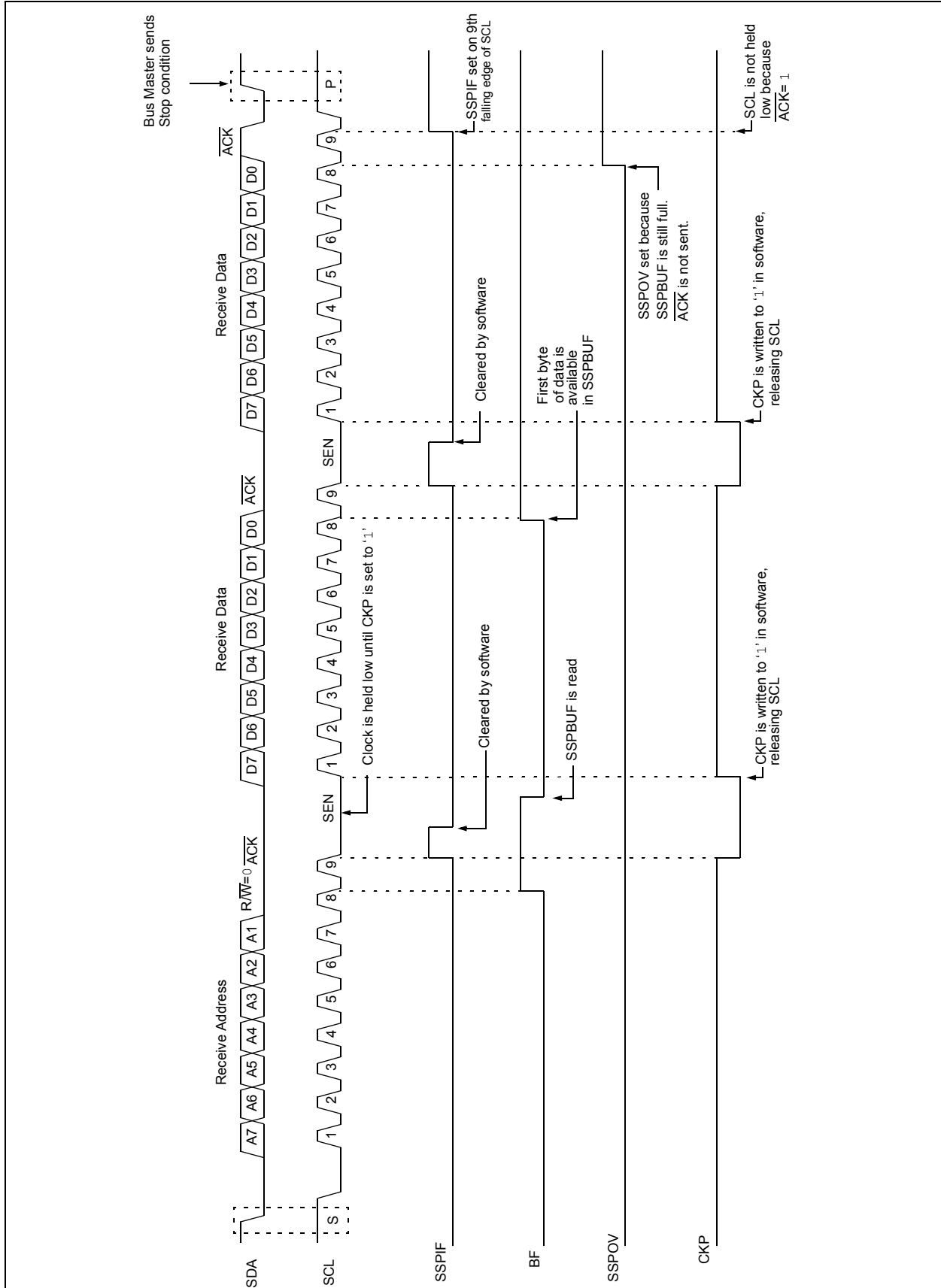
# PIC16(L)F1782/3

**FIGURE 26-16:** I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)



**Preliminary** © 2011-2012 Microchip Technology Inc.
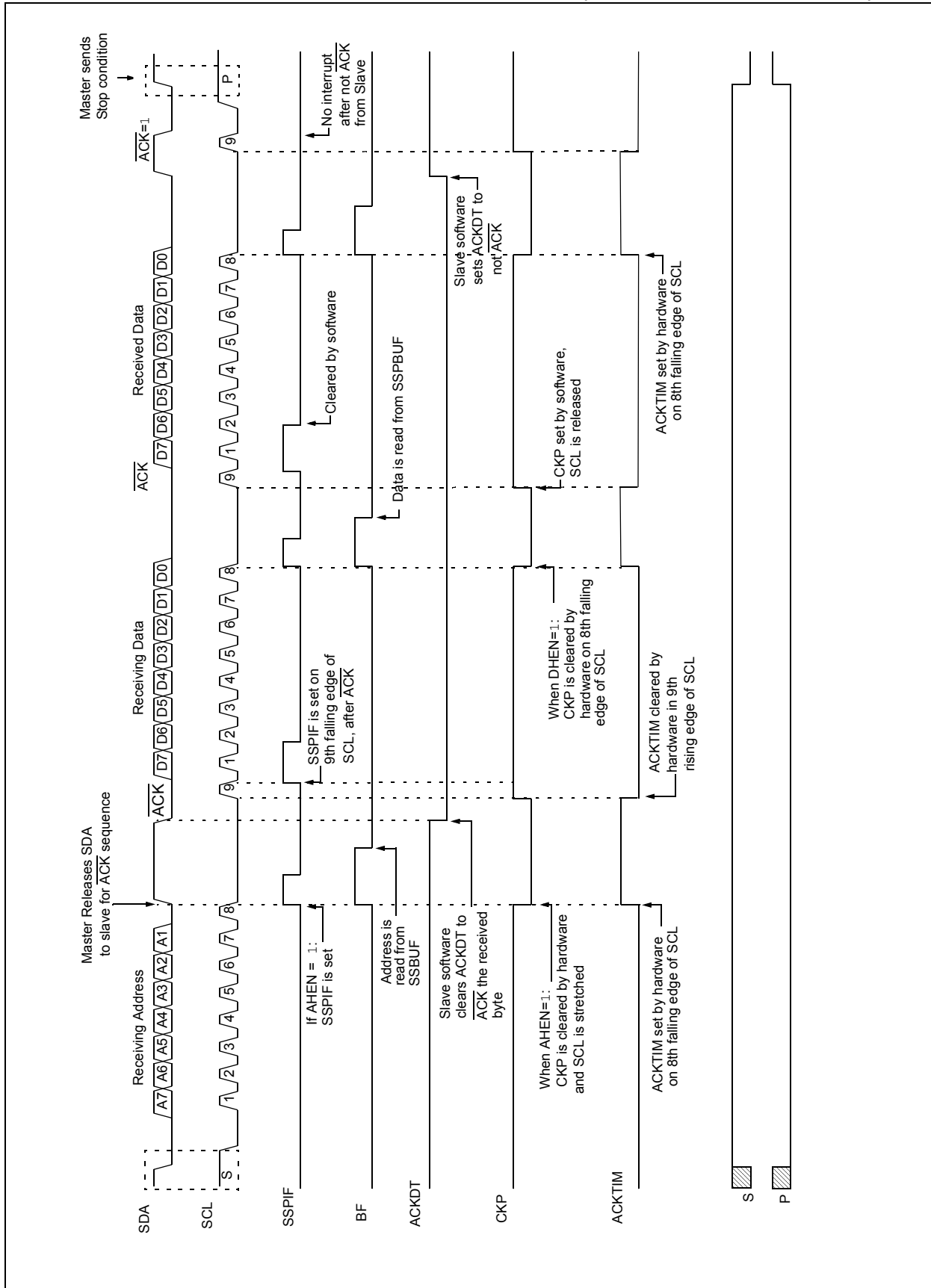
**FIGURE 26-17:** I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 1, DHEN = 1)
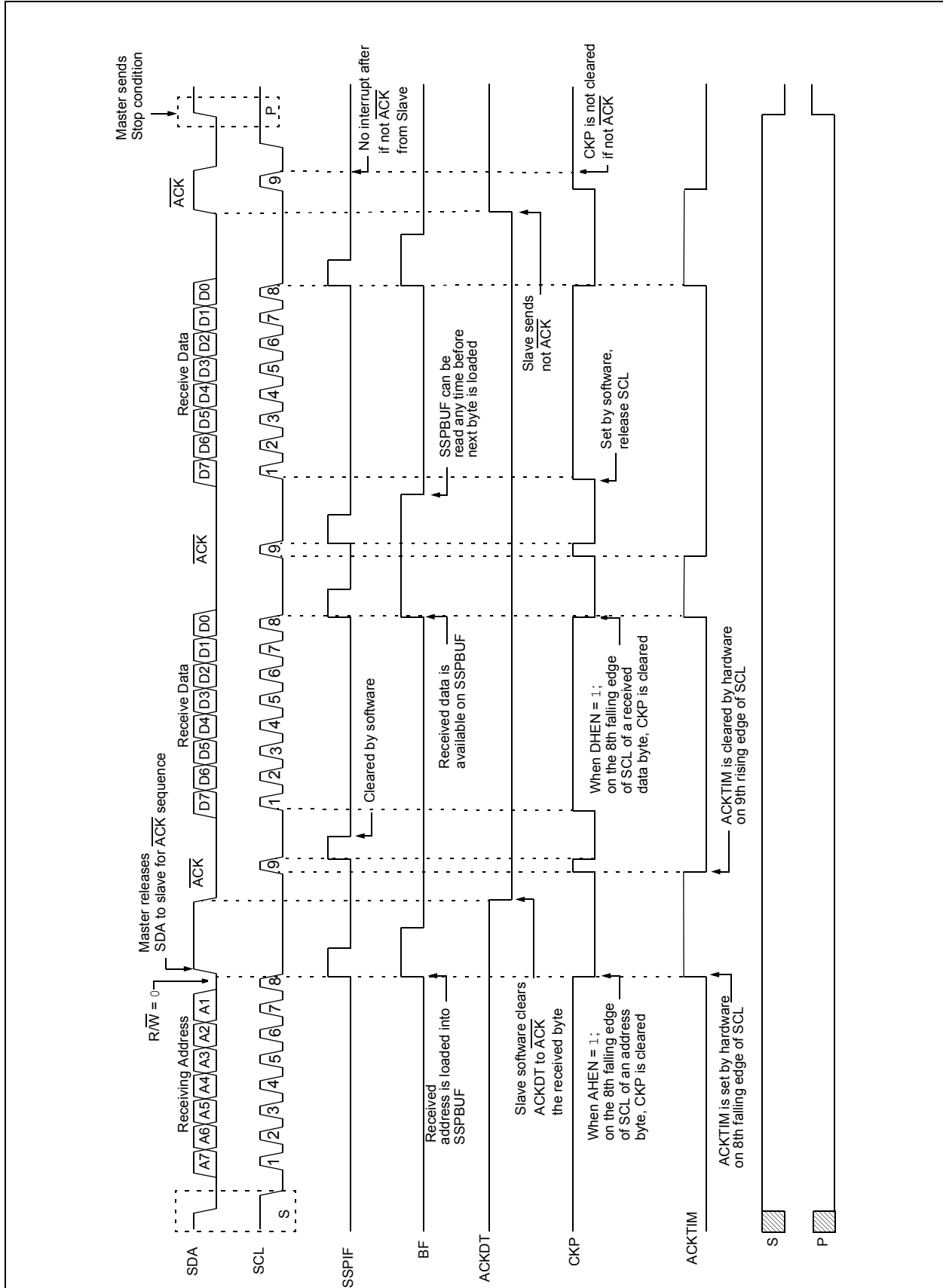
### 26.5.3 SLAVE TRANSMISSION

When the R/$\overline{\text{W}}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{\text{W}}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register, and an $\overline{\text{ACK}}$ pulse is sent by the slave on the ninth bit.

Following the $\overline{\text{ACK}}$, slave hardware clears the CKP bit and the SCL pin is held low (see **Section 26.5.6 "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 SSPBUF register which also loads the SSPSR register. Then the SCL pin should be released by setting the CKP bit of the SSPCON1 register. The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time.

The $\overline{\text{ACK}}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. This $\overline{\text{ACK}}$ value is copied to the ACKSTAT bit of the SSPCON2 register. If ACKSTAT is set (not $\overline{\text{ACK}}$), then the data transfer is complete. In this case, when the not $\overline{\text{ACK}}$ is latched by the slave, the slave goes idle and waits for another occurrence of the Start bit. If the SDA line was low ($\overline{\text{ACK}}$), the next transmit data must be loaded into the SSPBUF 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 SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

#### 26.5.3.1 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 SSPCON3 register is set, the BCLIF bit of the PIR 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.

#### 26.5.3.2 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 26-17 can be used as a reference to this list.

1. Master sends a Start condition on SDA and SCL.
2. S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{\text{W}}$ bit set is received by the Slave setting SSPIF bit.
4. Slave hardware generates an $\overline{\text{ACK}}$ and sets SSPIF.
5. SSPIF bit is cleared by user.
6. Software reads the received address from SSPBUF, clearing BF.
7. R/$\overline{\text{W}}$ is set so CKP was automatically cleared after the $\overline{\text{ACK}}$.
8. The slave software loads the transmit data into SSPBUF.
9. CKP bit is set releasing SCL, allowing the master to clock the data out of the slave.
10. SSPIF is set after the $\overline{\text{ACK}}$ 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 $\overline{\text{ACK}}$s 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 $\overline{\text{ACK}}$; 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.

 © 2011-2012 Microchip Technology Inc.

**FIGURE 26-18:** I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 0)
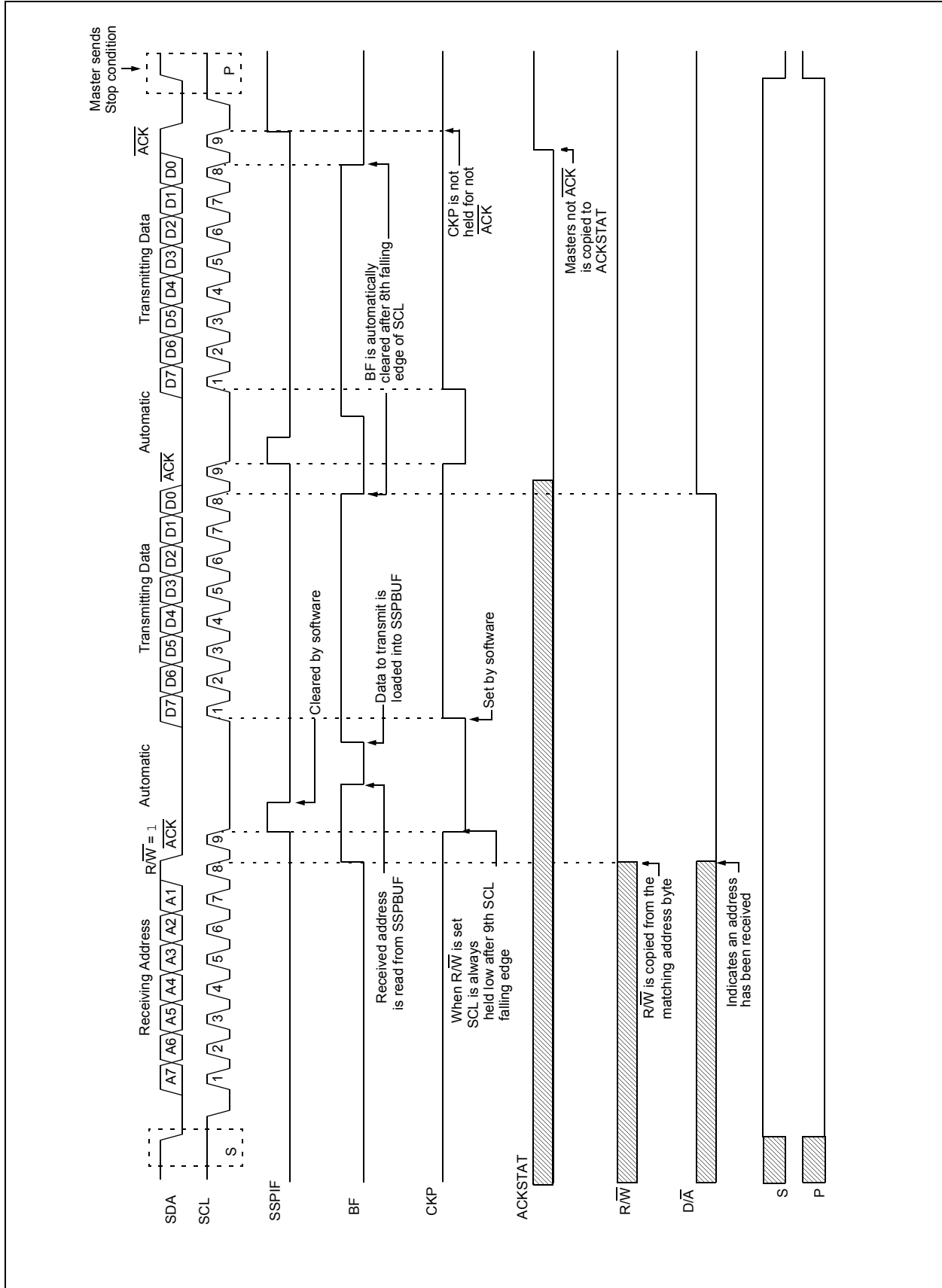
### 26.5.3.3 7-bit Transmission with Address Hold Enabled

Setting the AHEN bit of the SSPCON3 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 26-18 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 SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Master sends matching address with R/$\overline{\text{W}}$ 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 SSPCON3 register, and R/$\overline{\text{W}}$ and D/$\overline{\text{A}}$ of the SSPSTAT register to determine the source of the interrupt.
6. Slave reads the address value from the SSPBUF register clearing the BF bit.
7. Slave software decides from this information if it wishes to $\overline{\text{ACK}}$ or not $\overline{\text{ACK}}$ and sets the ACKDT bit of the SSPCON2 register accordingly.
8. Slave sets the CKP bit releasing SCL.
9. Master clocks in the $\overline{\text{ACK}}$ value from the slave.
10. Slave hardware automatically clears the CKP bit and sets SSPIF after the $\overline{\text{ACK}}$ if the R/$\overline{\text{W}}$ bit is set.
11. Slave software clears SSPIF.
12. Slave loads value to transmit to the master into SSPBUF setting the BF bit.

> **Note:** SSPBUF cannot be loaded until after the $\overline{\text{ACK}}$.

13. Slave sets the CKP bit releasing the clock.
14. Master clocks out the data from the slave and sends an $\overline{\text{ACK}}$ value on the 9th SCL pulse.
15. Slave hardware copies the $\overline{\text{ACK}}$ value into the ACKSTAT bit of the SSPCON2 register.
16. Steps 10-15 are repeated for each byte transmitted to the master from the slave.
17. If the master sends a not $\overline{\text{ACK}}$ the slave releases the bus allowing the master to send a Stop and end the communication.

> **Note:** Master must send a not $\overline{\text{ACK}}$ on the last byte to ensure that the slave releases the SCL line to receive a Stop.

**FIGURE 26-19:** $I^2C$ SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)
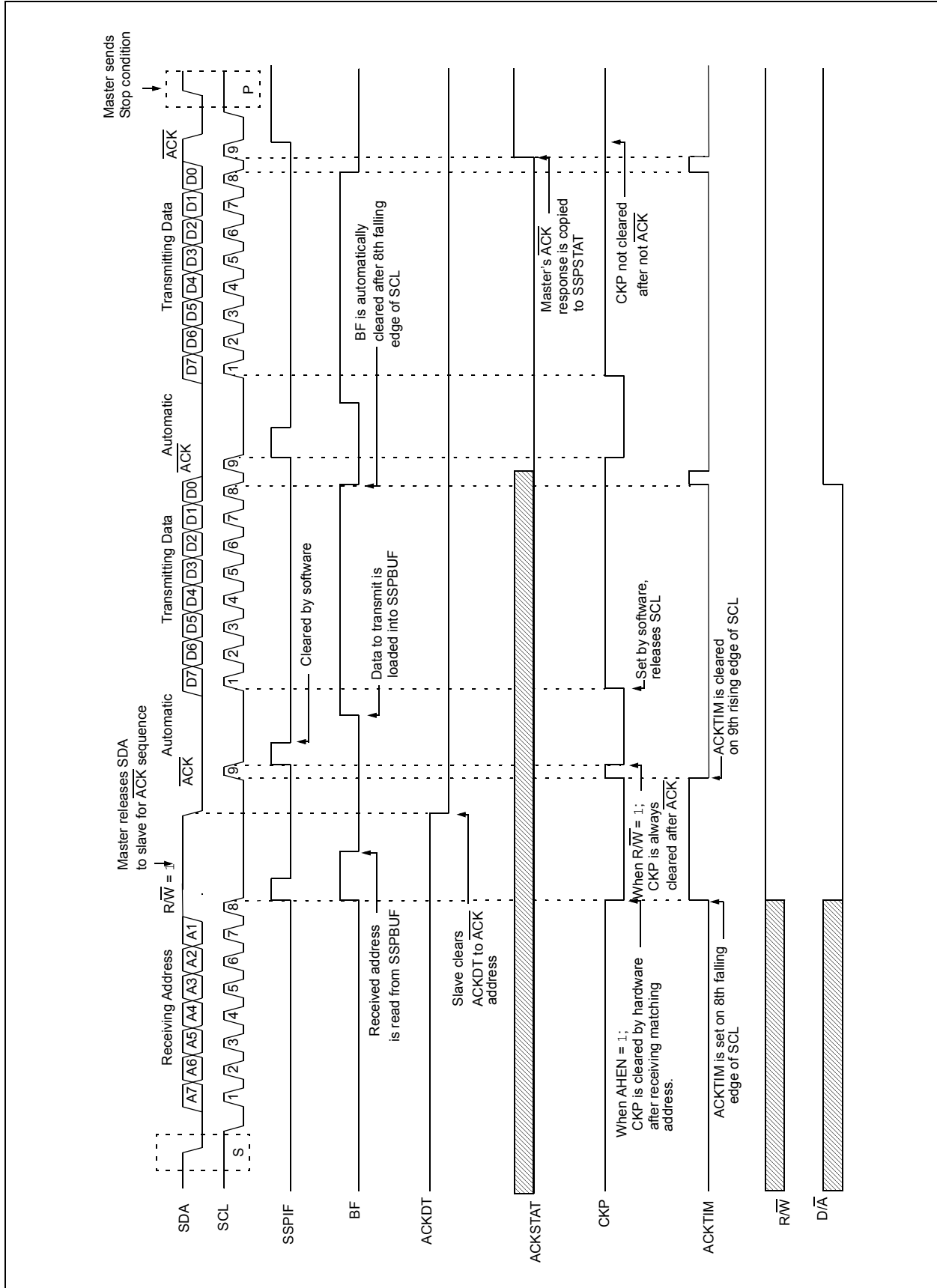
### 26.5.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSP module configured as an I$^2$C slave in 10-bit Addressing mode.

Figure 26-19 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 I$^2$C communication.

1.  Bus starts Idle.
2.  Master sends Start condition; S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3.  Master sends matching high address with R/$\overline{W}$ bit clear; UA bit of the SSPSTAT register is set.
4.  Slave sends $\overline{ACK}$ and SSPIF is set.
5.  Software clears the SSPIF bit.
6.  Software reads received address from SSPBUF clearing the BF flag.
7.  Slave loads low address into SSPADD, releasing SCL.
8.  Master sends matching low address byte to the slave; UA bit is set.

> **Note:** Updates to the SSPADD register are not allowed until after the $\overline{ACK}$ sequence.

9.  Slave sends $\overline{ACK}$ 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 SSPADD 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 SSPBUF clearing BF.
12. Slave loads high address into SSPADD.
13. Master clocks a data byte to the slave and clocks out the slaves $\overline{ACK}$ on the 9th SCL pulse; SSPIF is set.
14. If SEN bit of SSPCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPIF.
16. Slave reads the received byte from SSPBUF 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.

### 26.5.5 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 SSPADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCL line is held low are the same. Figure 26-20 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 26-21 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

**FIGURE 26-20:** **I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)**
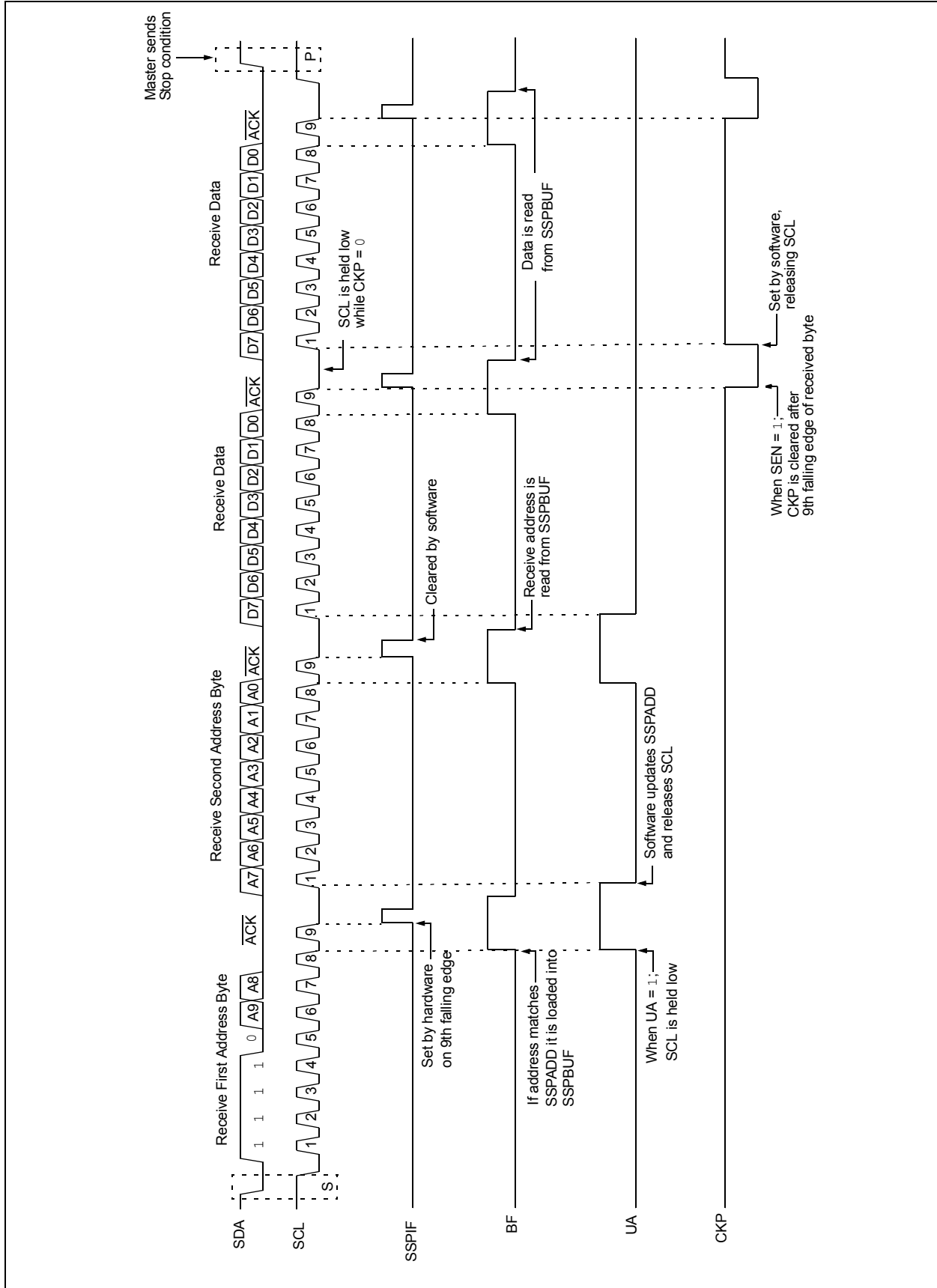
# PIC16(L)F1782/3

**FIGURE 26-21:** I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 0)



**Preliminary** © 2011-2012 Microchip Technology Inc.
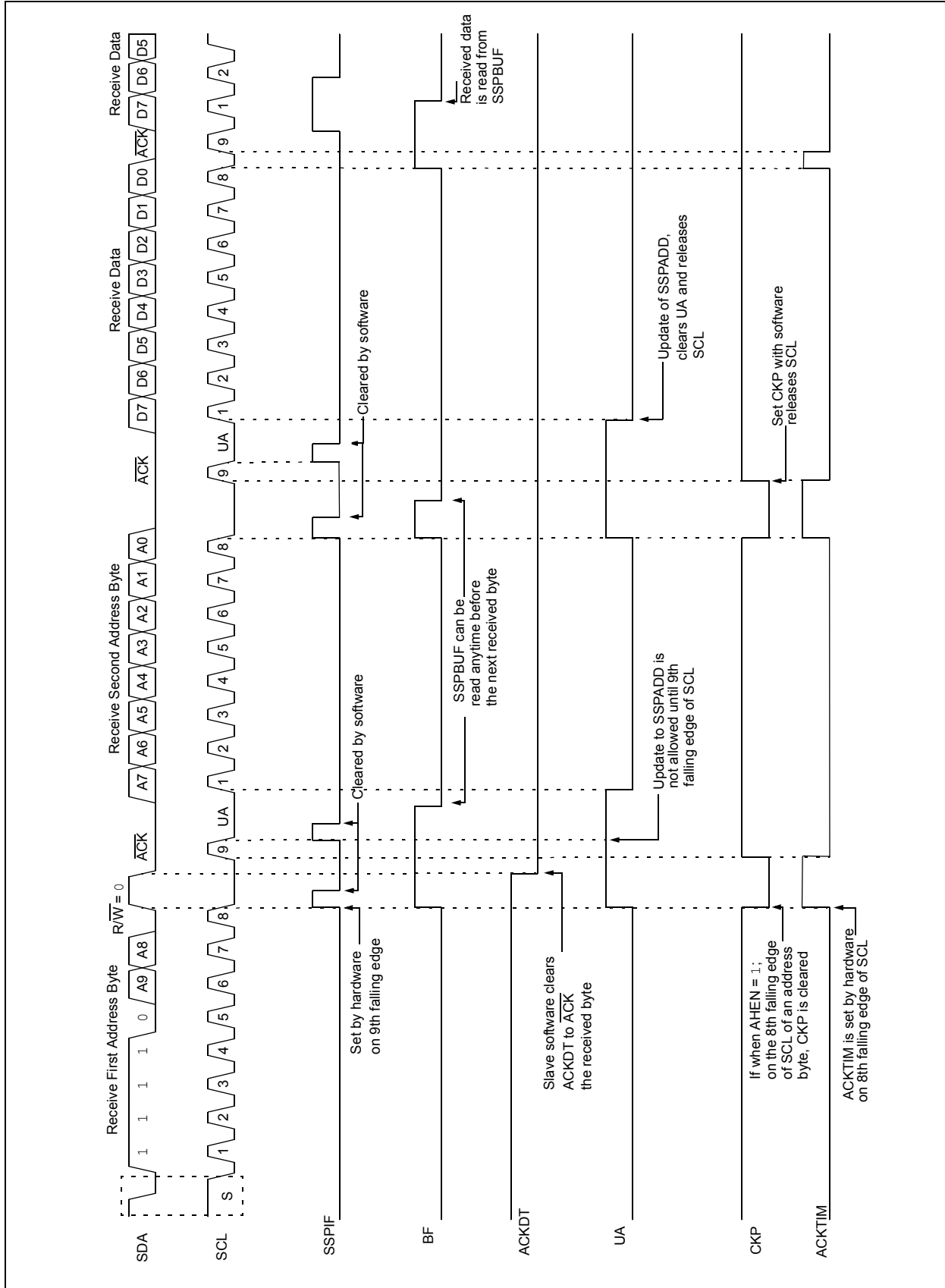
**FIGURE 26-22:** I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)
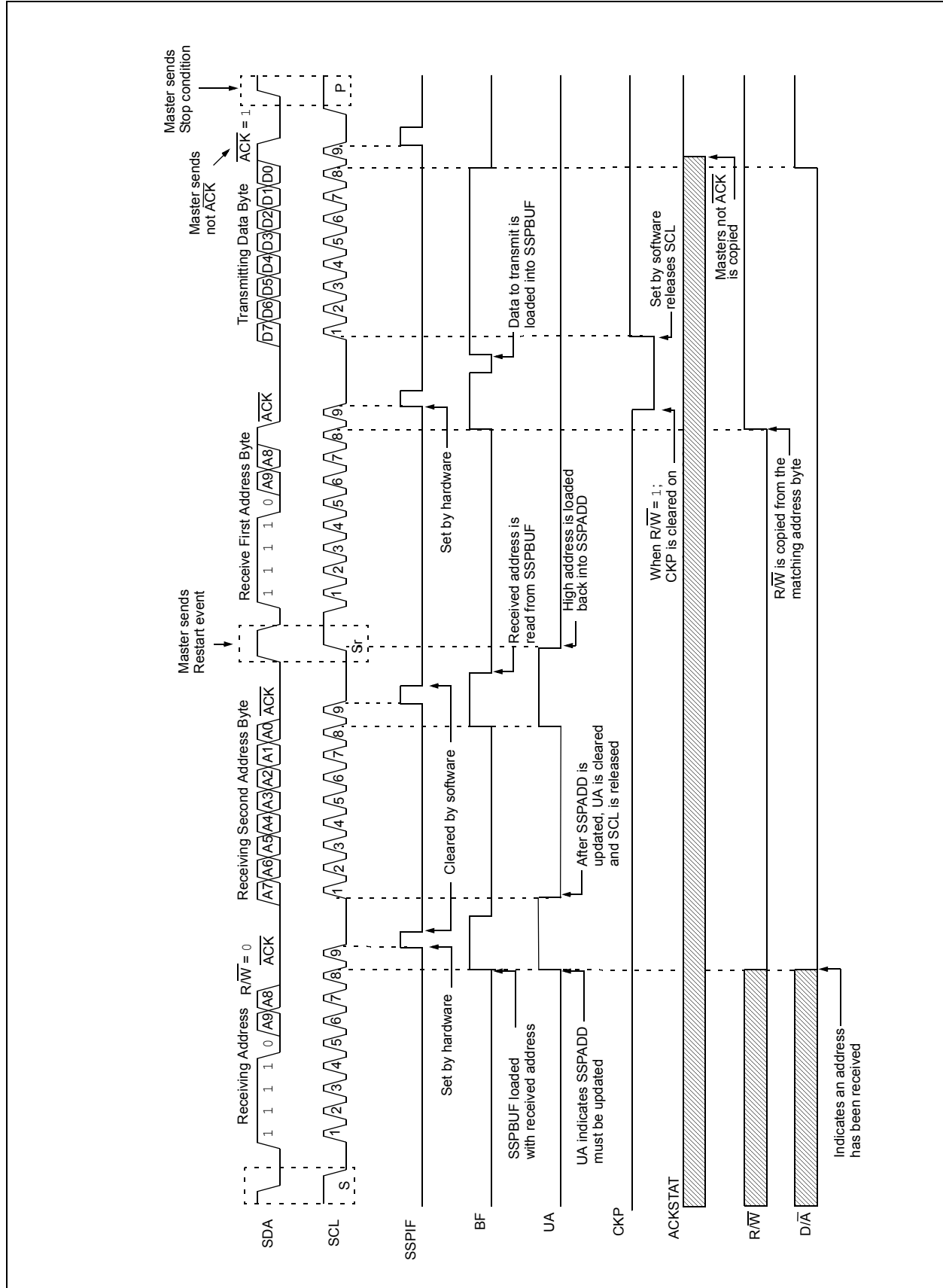
# PIC16(L)F1782/3

## 26.5.6 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 SSPCON1 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.

### 26.5.6.1 Normal Clock Stretching

Following an $\overline{ACK}$ if the R/$\overline{W}$ bit of SSPSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPBUF with data to transfer to the master. If the SEN bit of SSPCON2 is set, the slave hardware will always stretch the clock after the $\overline{ACK}$ sequence. Once the slave is ready; CKP is set by software and communication resumes.

> **Note 1:** The BF bit has no effect on if 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 SSPBUF was read before the 9th falling edge of SCL.
>
> **2:** Previous versions of the module did not stretch the clock for a transmission if SSPBUF was loaded before the 9th falling edge of SCL. It is now always cleared for read requests.

### 26.5.6.2 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 SSPADD.

> **Note:** Previous versions of the module did not stretch the clock if the second address byte did not match.

### 26.5.6.3 Byte NACKing

When AHEN bit of SSPCON3 is set; CKP is cleared by hardware after the 8th falling edge of SCL for a received matching address byte. When DHEN bit of SSPCON3 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.

## 26.5.7 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 I$^2$C master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I$^2$C 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 26-22).

## FIGURE 26-23: CLOCK SYNCHRONIZATION TIMING
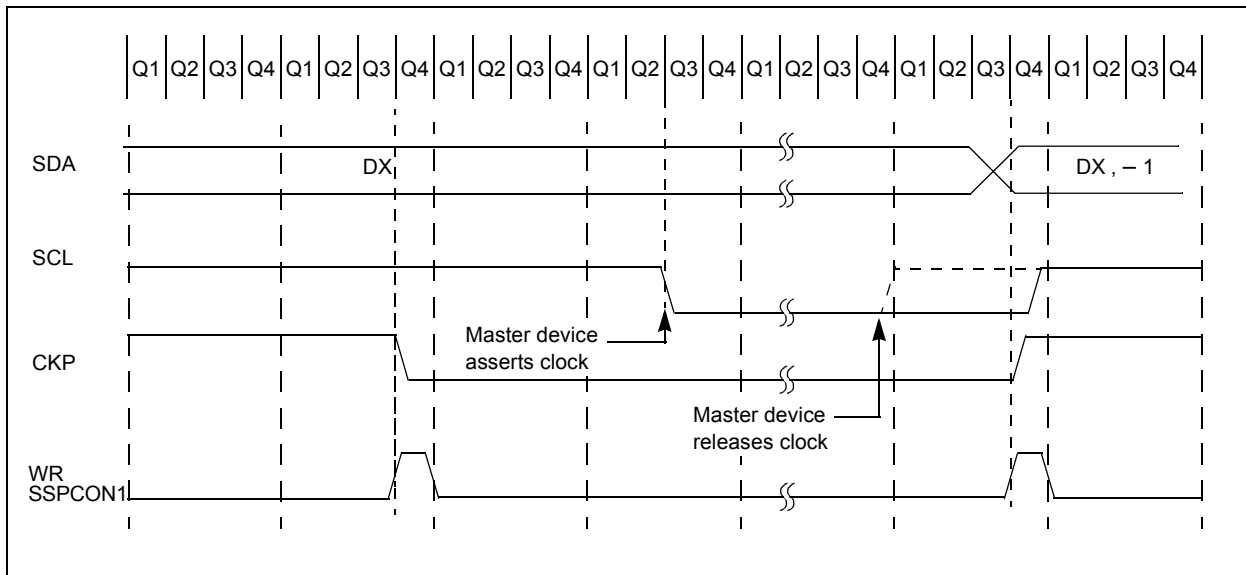
### 26.5.8 GENERAL CALL ADDRESS SUPPORT

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 SSPCON2 register is set, the slave module will automatically $\overline{ACK}$ the reception of this address regardless of the value stored in SSPADD. After the slave clocks in an address of all zeros with the R/$\overline{W}$ bit clear, an interrupt is generated and slave software can read SSPBUF and respond. Figure 26-23 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 SSPCON3 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 26-24: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE**
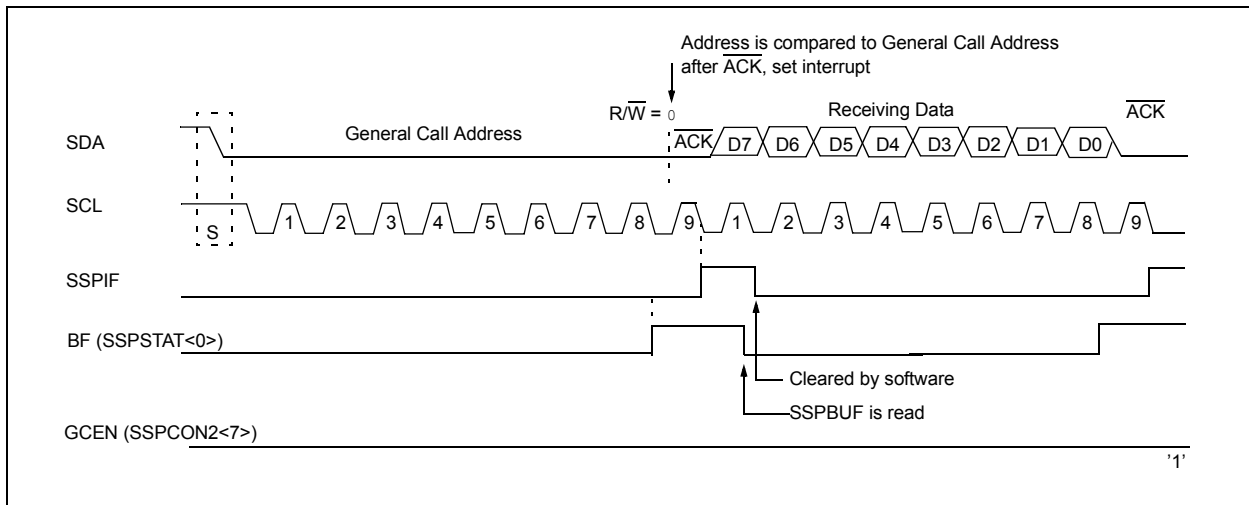


### 26.5.9 SSP MASK REGISTER

An SSP Mask (SSPMSK) register (Register 26-5) is available in I²C Slave mode as a mask for the value held in the SSPSR register during an address comparison operation. A zero ('0') bit in the SSPMSK register has the effect of making the corresponding bit 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 SSP operation until written with a mask value.

The SSP Mask register is active during:

• 7-bit Address mode: address compare of A<7:1>.

• 10-bit Address mode: address compare of A<7:0> only. The SSP mask has no effect during the reception of the first (high) byte of the address.

# PIC16(L)F1782/3

## 26.6 I²C Master Mode

Master mode is enabled by setting and clearing the appropriate SSPM bits in the SSPCON1 register and by setting the SSPEN bit. In Master mode, the SDA and SCK pins must be configured as inputs. The MSSP peripheral hardware will override the output driver TRIS controls when necessary to drive the pins low.

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 SSP Interrupt Flag bit, SSPIF, to be set (SSP 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 queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur
>
> **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.

### 26.6.1 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/$\overline{W}$) bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

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

A Baud Rate Generator is used to set the clock frequency output on SCL. See **Section 26.7 "Baud Rate Generator"** for more detail.

### 26.6.2 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 SSPADD<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 26-25).

**FIGURE 26-25: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**
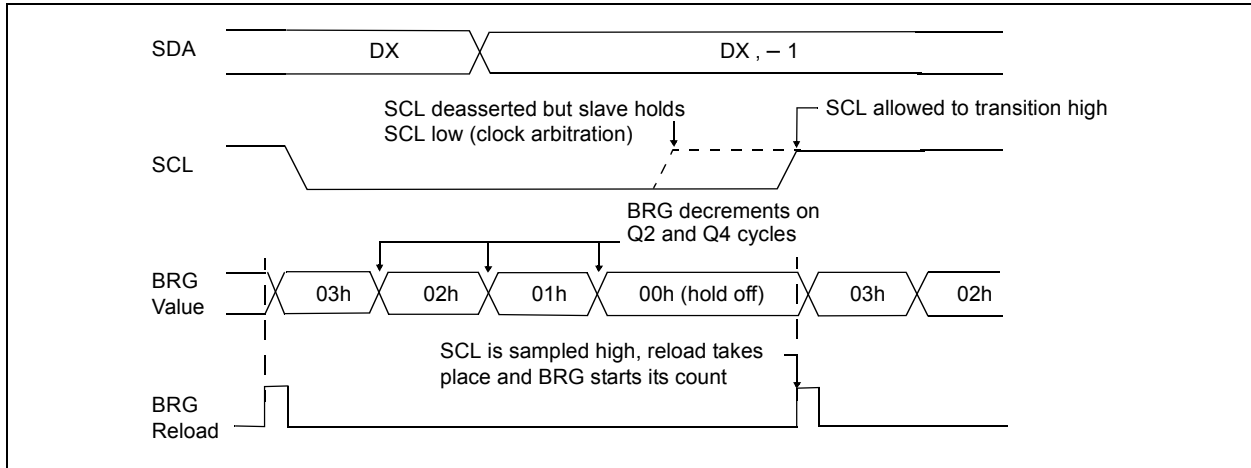


### 26.6.3 WCOL STATUS FLAG

If the user writes the SSPBUF 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 SSPBUF was attempted while the module was not idle.

| Note: | Because queueing of events is not allowed, writing to the lower 5 bits of SSPCON2 is disabled until the Start condition is complete. |
|---|---|

# PIC16(L)F1782/3

### 26.6.4 I$^2$C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Enable bit, SEN bit of the SSPCON2 register. If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (T$_{BRG}$), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the S bit of the SSPSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and resumes its count. When the Baud Rate Generator times out (T$_{BRG}$), the SEN bit of the SSPCON2 reg-

ister 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 I$^2$C module is reset into its Idle state.
>
> **2:** The Philips I$^2$C specification states that a bus collision cannot occur on a Start.

### FIGURE 26-26: FIRST START BIT TIMING



**Preliminary** © 2011-2012 Microchip Technology Inc.

### 26.6.5    I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit of the SSPCON2 register is programmed high and the 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 (T$_{BRG}$). 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 T$_{BRG}$. This action is then followed by assertion of the SDA pin (SDA = 0) for one T$_{BRG}$ while SCL is high. SCL is asserted low. Following this, the RSEN bit of the

SSPCON2 register will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit of the SSPSTAT register will be set. The SSPIF bit will not be set until the Baud Rate Generator has timed out.

---

**Note 1:** If RSEN is programmed while any other event is in progress, it will not take effect.

**2:** A bus collision during the Repeated Start condition occurs if:

- SDA is sampled low when SCL goes from low-to-high.
- SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data '1'.

---

**FIGURE 26-27:    REPEAT START CONDITION WAVEFORM**

# PIC16(L)F1782/3

## 26.6.6 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 SSPBUF register. This action will set the Buffer Full flag bit, BF and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted. SCL is held low for one Baud Rate Generator rollover count (T$_{BRG}$). Data should be valid before SCL is released high. When the SCL pin is released high, it is held that way for T$_{BRG}$. 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 $\overline{ACK}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{ACK}$ 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 SSPBUF, leaving SCL low and SDA unchanged (Figure 26-27).

After the write to the SSPBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/$\overline{W}$ 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 $\overline{ACK}$ bit is loaded into the ACKSTAT Status bit of the SSPCON2 register. Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

### 26.6.6.1 BF Status Flag

In Transmit mode, the BF bit of the SSPSTAT register is set when the CPU writes to SSPBUF and is cleared when all 8 bits are shifted out.

### 26.6.6.2 WCOL Status Flag

If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write does not occur).

WCOL must be cleared by software before the next transmission.

### 26.6.6.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit of the SSPCON2 register is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the slave does not Acknowledge ($\overline{ACK}$ = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

### 26.6.6.4 Typical transmit sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPCON2 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 SSPBUF with the slave address to transmit.
6. Address is shifted out the SDA pin until all 8 bits are transmitted. Transmission begins as soon as SSPBUF is written to.
7. The MSSP module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
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 SSPBUF with eight bits of data.
10. Data is shifted out the SDA pin until all 8 bits are transmitted.
11. The MSSP module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
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 SSPCON2 register. Interrupt is generated once the Stop/Restart condition is complete.

**Preliminary** © 2011-2012 Microchip Technology Inc.

**FIGURE 26-28:** I²C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)
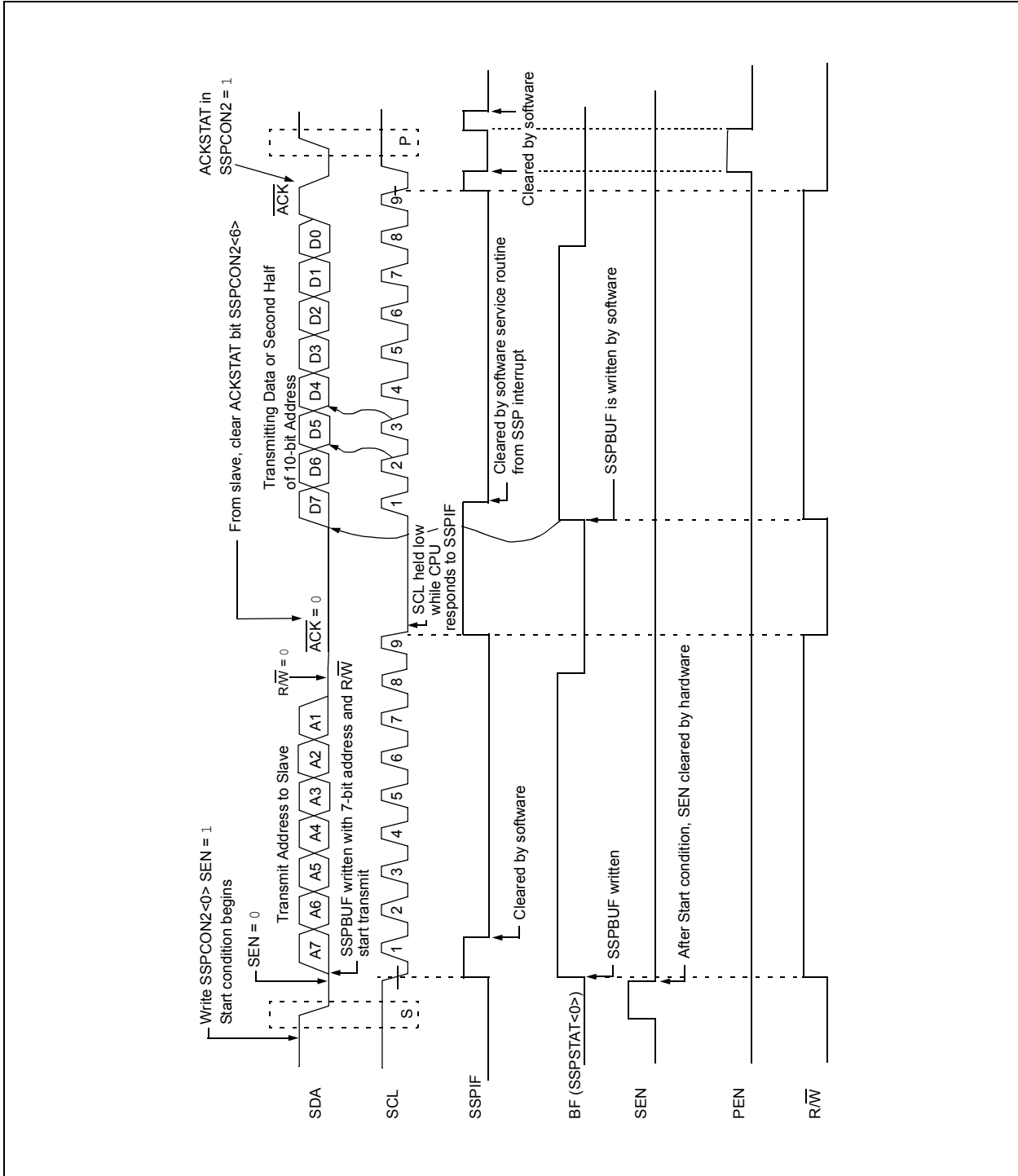
### 26.6.7    I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN bit of the SSPCON2 register.

> **Note:**    The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable, ACKEN bit of the SSPCON2 register.

#### 26.6.7.1    BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

#### 26.6.7.2    SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

#### 26.6.7.3    WCOL Status Flag

If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

#### 26.6.7.4    Typical Receive Sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPCON2 register.
2. SSPIF is set by hardware on completion of the Start.
3. SSPIF is cleared by software.
4. User writes SSPBUF with the slave address to transmit and the R/$\overline{\text{W}}$ bit set.
5. Address is shifted out the SDA pin until all 8 bits are transmitted. Transmission begins as soon as SSPBUF is written to.
6. The MSSP module shifts in the $\overline{\text{ACK}}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 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 SSPCON2 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 SSPUF, clears BF.
11. Master sets $\overline{\text{ACK}}$ value sent to slave in ACKDT bit of the SSPCON2 register and initiates the $\overline{\text{ACK}}$ by setting the ACKEN bit.
12. Masters $\overline{\text{ACK}}$ 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 $\overline{\text{ACK}}$ or Stop to end communication.

# PIC16(L)F1782/3

**FIGURE 26-29:** I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)
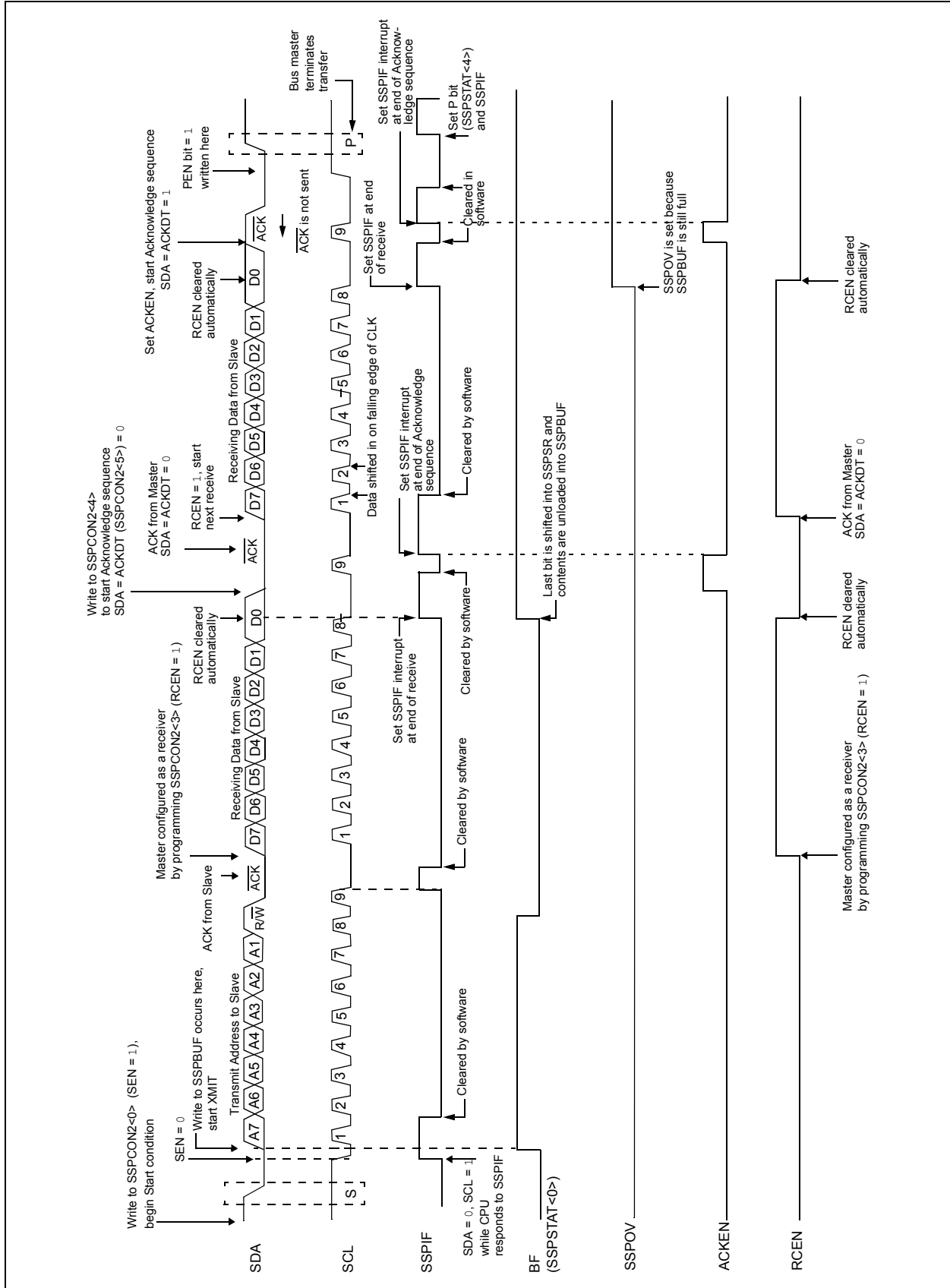
© 2011-2012 Microchip Technology Inc.          **Preliminary**          DS41579C-page 303

## 26.6.8 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN bit of the SSPCON2 register. When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (T$_{BRG}$) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for T$_{BRG}$. 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 26-29).

### 26.6.8.1 WCOL Status Flag

If the user writes the SSPBUF when an Acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write does not occur).

## 26.6.9 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 bit of the SSPCON2 register. At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCL pin will be brought high and one T$_{BRG}$ (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit of the SSPSTAT register is set. A T$_{BRG}$ later, the PEN bit is cleared and the SSPIF bit is set (Figure 26-30).

### 26.6.9.1 WCOL Status Flag

If the user writes the SSPBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

**FIGURE 26-30: ACKNOWLEDGE SEQUENCE WAVEFORM**



Note: T$_{BRG}$ = one Baud Rate Generator period.
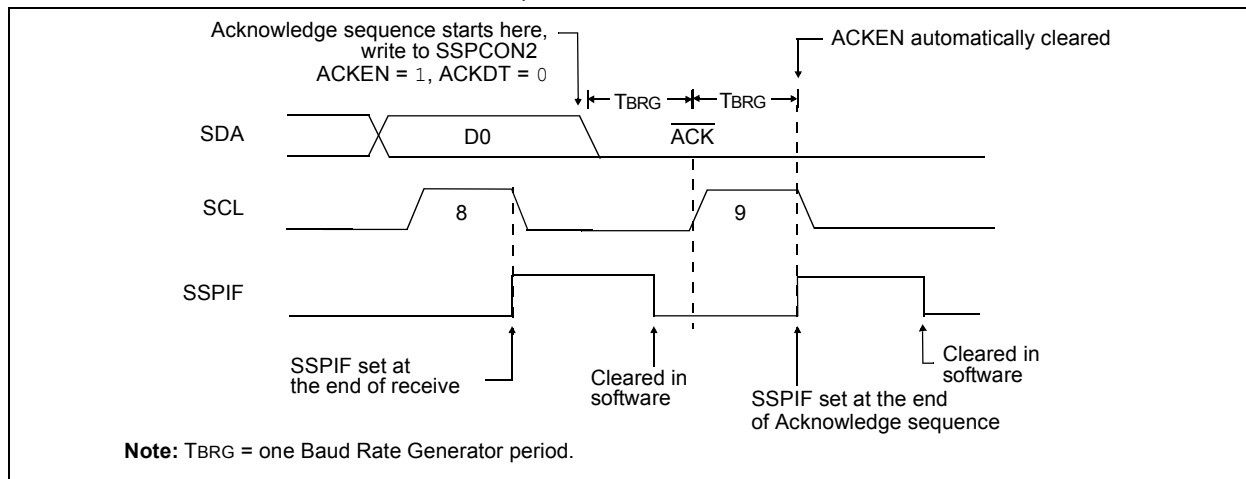
**FIGURE 26-31: STOP CONDITION RECEIVE OR TRANSMIT MODE**



Note: T$_{BRG}$ = one Baud Rate Generator period.

 © 2011-2012 Microchip Technology Inc.
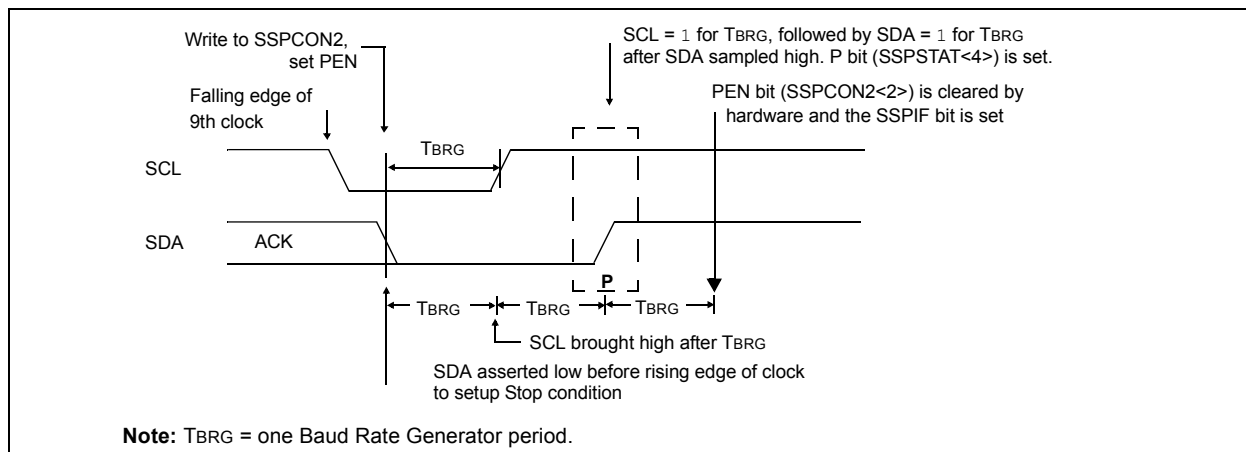
### 26.6.10 SLEEP OPERATION

While in Sleep mode, the I$^2$C slave module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

### 26.6.11 EFFECTS OF A RESET

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

### 26.6.12 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 I$^2$C bus may be taken when the P bit of the SSPSTAT register is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the SSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDA line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLIF bit.

The states where arbitration can be lost are:

• Address Transfer
• Data Transfer
• A Start Condition
• A Repeated Start Condition
• An Acknowledge Condition

### 26.6.13 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 I$^2$C port to its Idle state (Figure 26-31).

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 SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I$^2$C bus is free, the user can resume communication by asserting a Start condition.

If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I$^2$C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I$^2$C bus can be taken when the P bit is set in the SSPSTAT register, or the bus is Idle and the S and P bits are cleared.

**FIGURE 26-32:** BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE
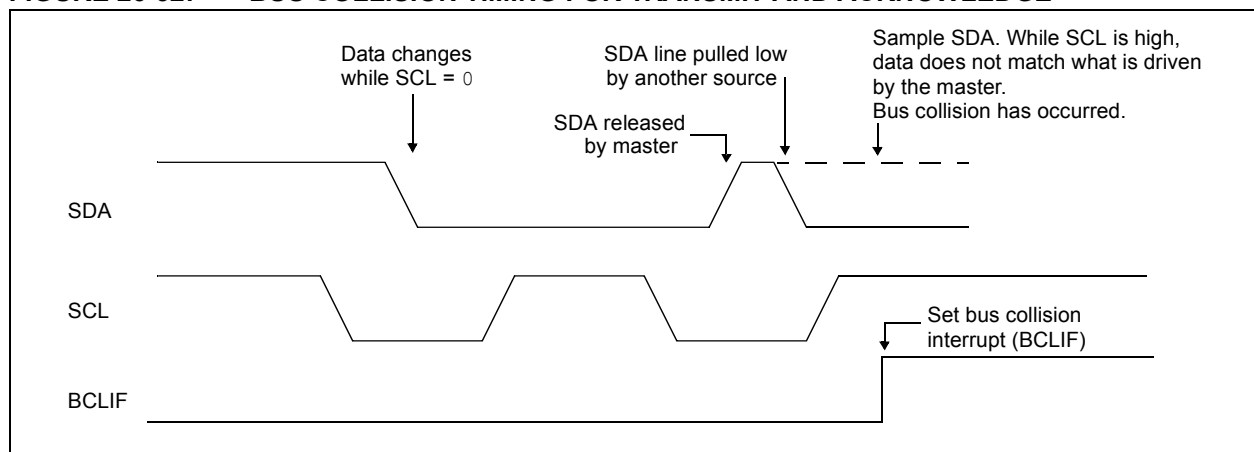
### 26.6.13.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the Start condition (Figure 26-32).

b) SCL is sampled low before SDA is asserted low (Figure 26-33).

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 26-32).

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 26-34). 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 26-33:** **BUS COLLISION DURING START CONDITION (SDA ONLY)**



**Preliminary** © 2011-2012 Microchip Technology Inc.
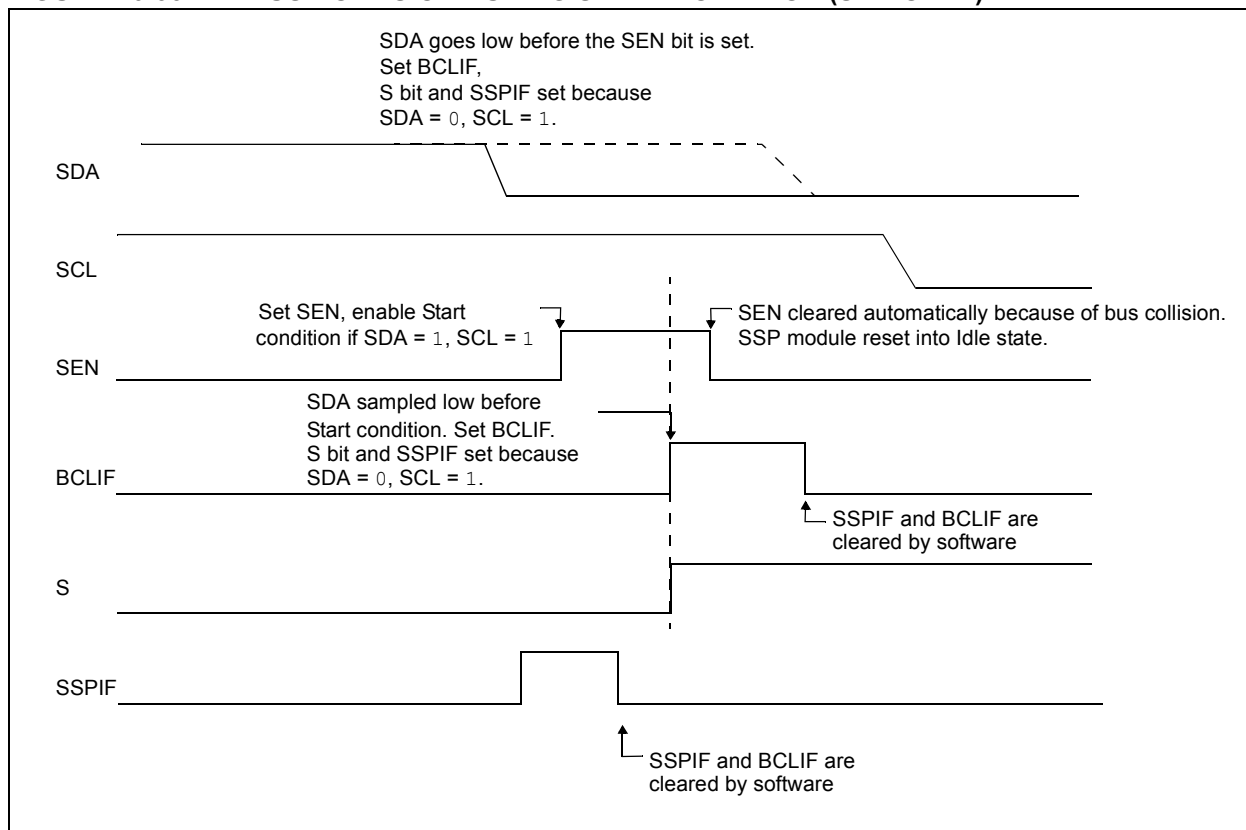
**FIGURE 26-34:** **BUS COLLISION DURING START CONDITION (SCL = 0)**
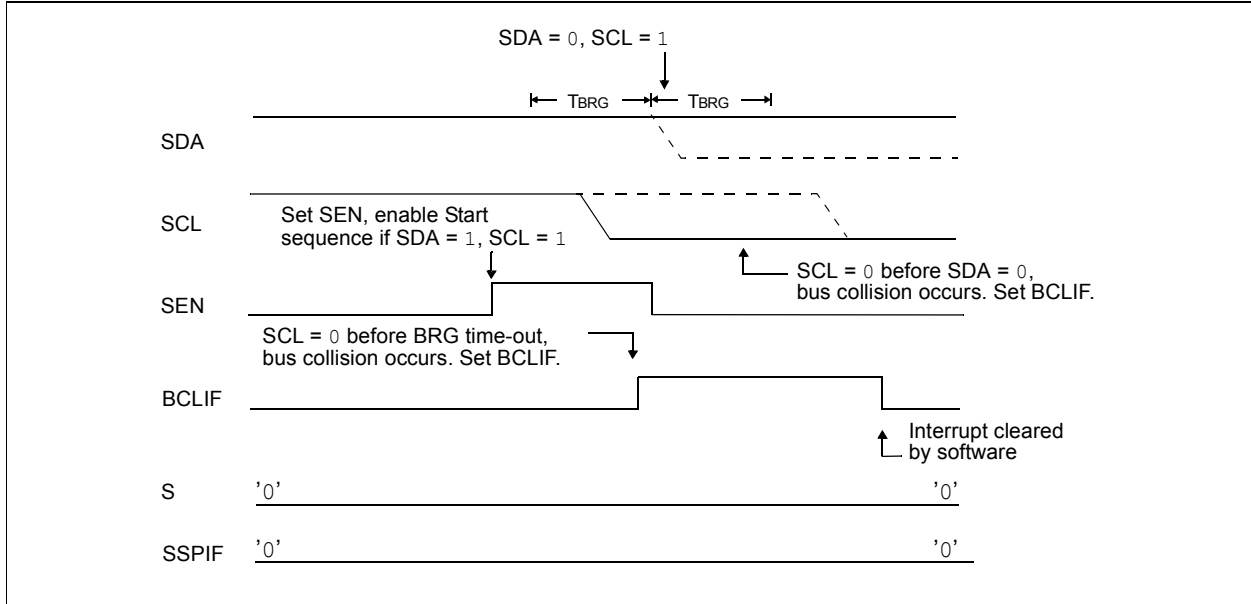


**FIGURE 26-35:** **BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION**
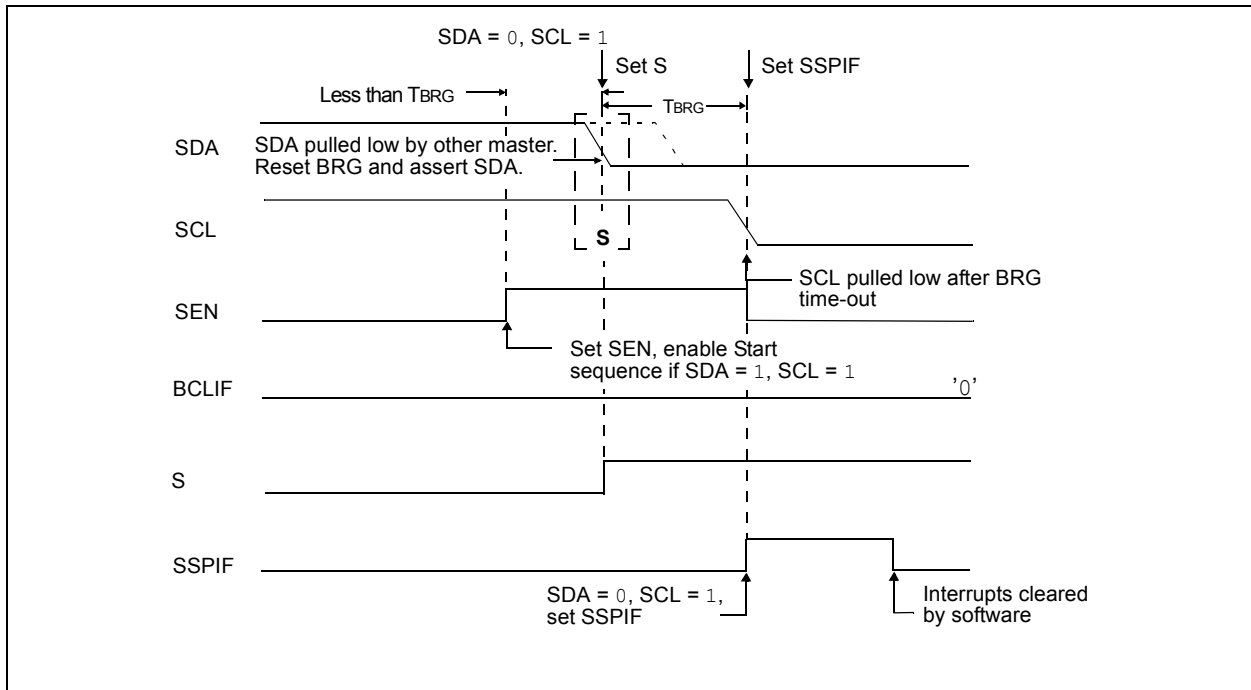
# PIC16(L)F1782/3

### 26.6.13.2    Bus Collision During a Repeated Start Condition

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

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

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

When the user releases SDA and the pin is allowed to float high, the BRG is loaded with SSPADD 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 26-35). 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 26-36.

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 26-36:    BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)
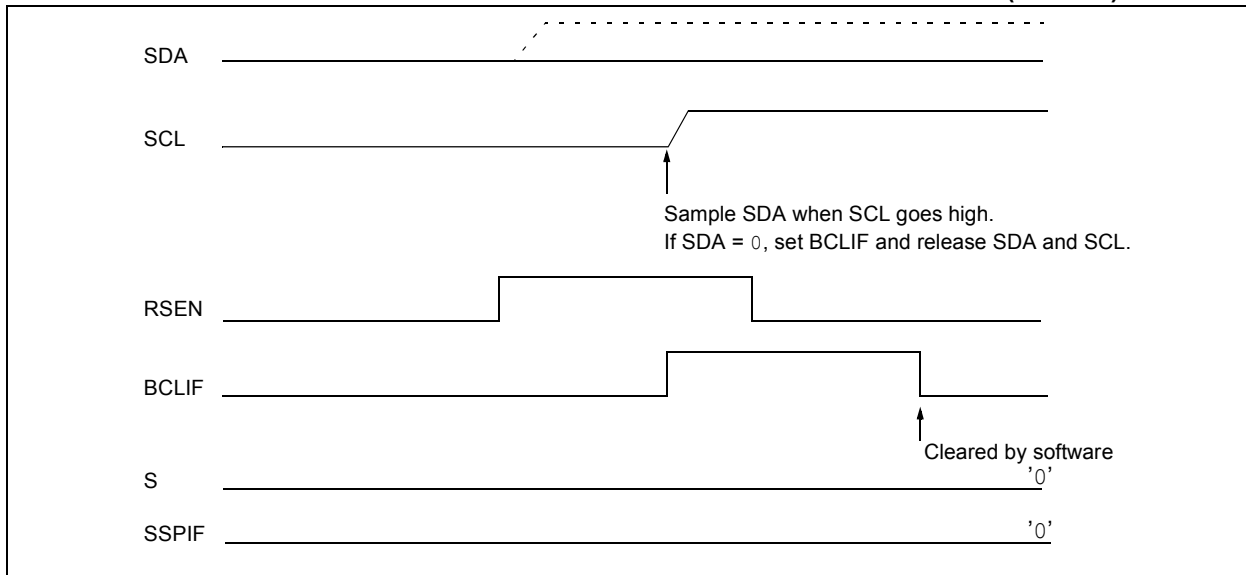


FIGURE 26-37:    BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



**Preliminary**    © 2011-2012 Microchip Technology Inc.
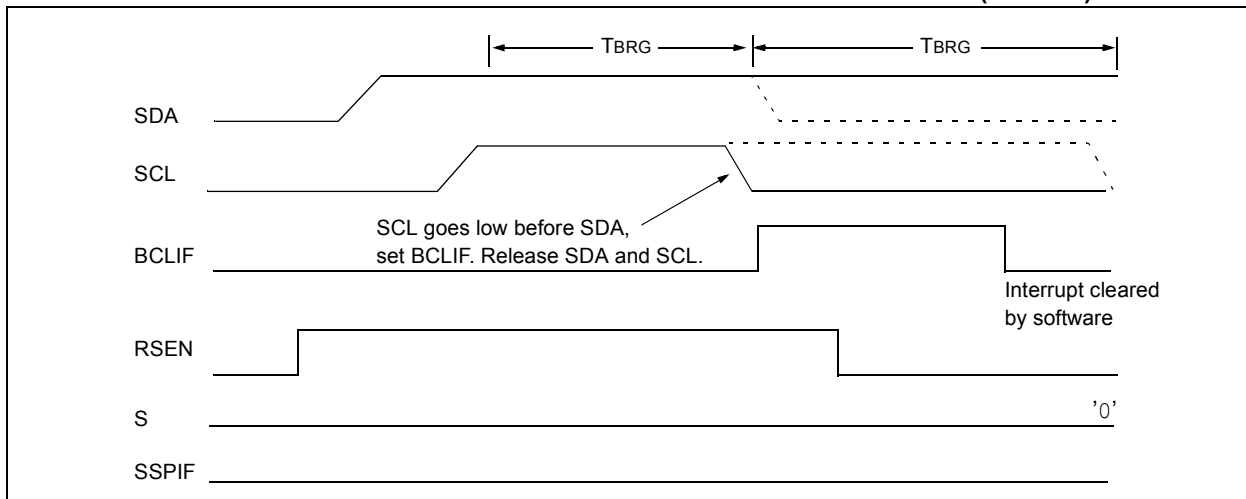
### 26.6.13.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

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

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

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD and counts down to 0. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 26-37). 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 26-38).

**FIGURE 26-38: BUS COLLISION DURING A STOP CONDITION (CASE 1)**
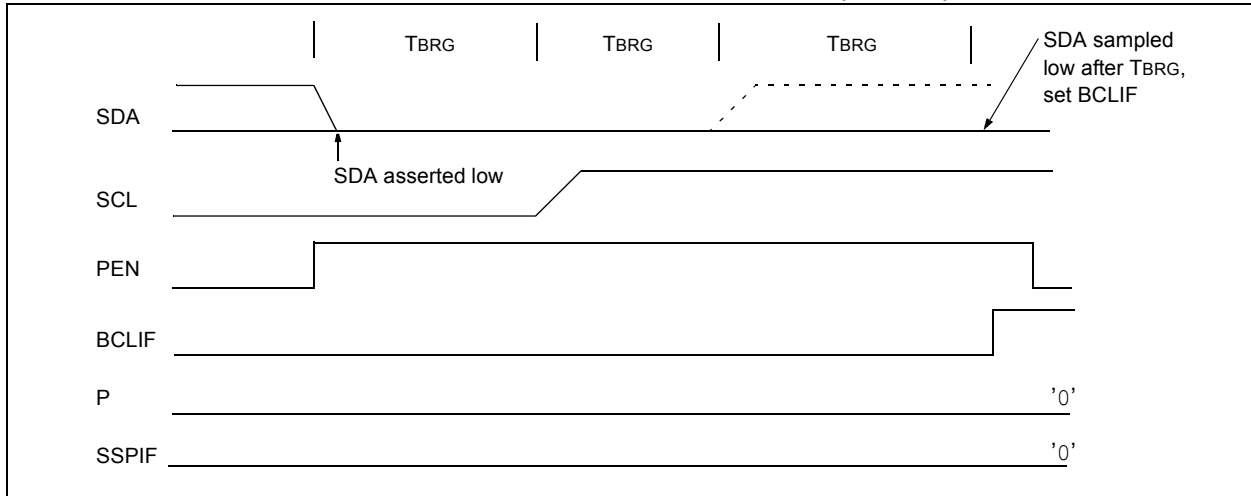


**FIGURE 26-39: BUS COLLISION DURING A STOP CONDITION (CASE 2)**
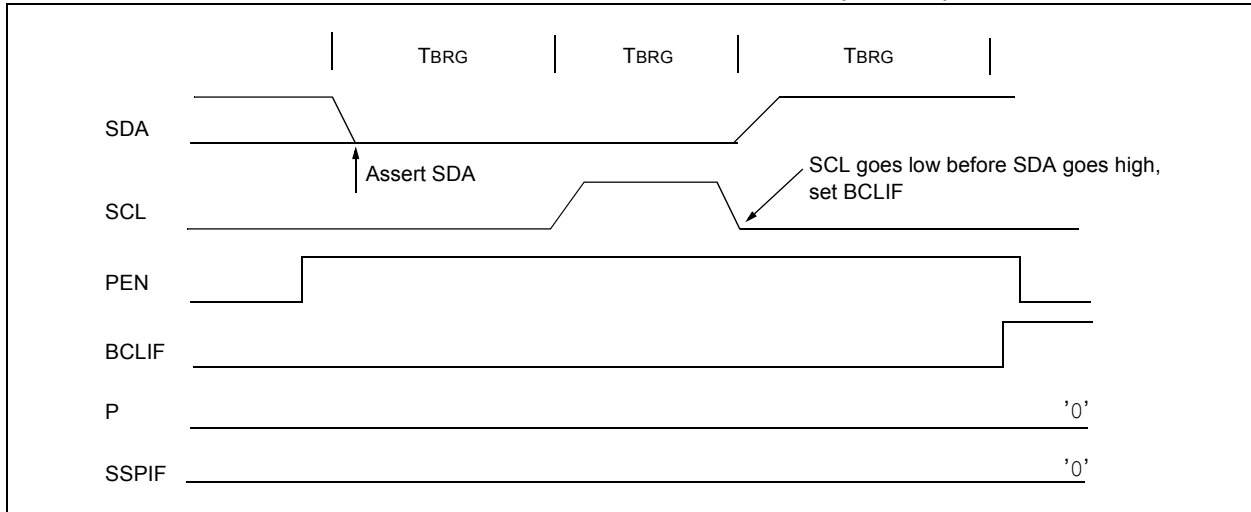
# PIC16(L)F1782/3

**TABLE 26-3:** **SUMMARY OF REGISTERS ASSOCIATED WITH I²C™ OPERATION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| APFCON | C2OUTSEL | CCP1SEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 119 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIE2 | OSFIE | C2IE | C1IE | EEIE | BCL1IE | — | C3IE | CCP2IE | 86 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| PIR2 | OSFIF | C2IF | C1IF | EEIF | BCL1IF | — | C3IF | CCP2IF | 89 |
| SSPADD | ADD<7:0> | | | | | | | | 316 |
| SSPBUF | Synchronous Serial Port Receive Buffer/Transmit Register | | | | | | | | 269* |
| SSPCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 313 |
| SSPCON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 314 |
| SSPCON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 313 |
| SSPMSK | MSK<7:0> | | | | | | | | 316 |
| SSPSTAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 312 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISA0 | 133 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by the MSSP module in I²C™ mode.
       * Page provides register information.
**Note** **1:** PIC16(L)F1783 only.

## 26.7 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 SSPADD register (Register 26-6). When a write occurs to SSPBUF, 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 26-39 triggers the value from SSPADD 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.

Table 26-4 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

**EQUATION 26-1:**

$$F_{CLOCK} = \frac{F_{OSC}}{(SSPxADD + 1)(4)}$$

**FIGURE 26-40: BAUD RATE GENERATOR BLOCK DIAGRAM**
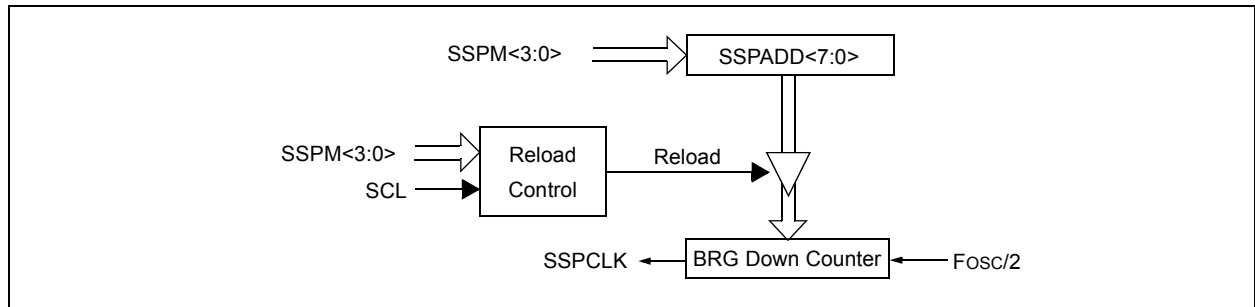


> **Note:** Values of 0x00, 0x01 and 0x02 are not valid for SSPADD when used as a Baud Rate Generator for I²C. This is an implementation limitation.

**TABLE 26-4: 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.

## 26.8    Register Definitions: MSSP Control

**REGISTER 26-1:    SSPSTAT: SSP STATUS REGISTER**

| R/W-0/0 | R/W-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
|---------|---------|-------|-------|-------|-------|-------|-------|
| SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | 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 I$^2$C 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 I$^2$C™ mode only:</u>
1 = Enable input logic so that thresholds are compliant with SMBus specification
0 = Disable SMBus specific inputs

bit 5      **D/$\overline{\text{A}}$:** 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, SSPEN 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, SSPEN 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/$\overline{\text{W}}$:** Read/Write bit information (I$^2$C mode only)
This bit holds the R/$\overline{\text{W}}$ 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 I$^2$C Slave mode:</u>
1 =   Read
0 =   Write
<u>In I$^2$C 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 SSPADD register
0 = Address does not need to be updated

bit 0      **BF:** Buffer Full Status bit

<u>Receive (SPI and I$^2$C modes):</u>
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty

<u>Transmit (I$^2$C mode only):</u>
1 = Data transmit in progress (does not include the $\overline{\text{ACK}}$ and Stop bits), SSPBUF is full
0 = Data transmit complete (does not include the $\overline{\text{ACK}}$ and Stop bits), SSPBUF is empty

## REGISTER 26-2:   SSPCON1: SSP CONTROL REGISTER 1

| R/C/HS-0/0 | R/C/HS-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| WCOL | SSPOV | SSPEN | CKP | SSPM<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 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 SSPBUF register was attempted while the I$^2$C conditions were not valid for a transmission to be started
0 = No collision
Slave mode:
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6
**SSPOV:** Receive Overflow Indicator bit[1]
In SPI mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPBUF, 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 SSPBUF register (must be cleared in software).
0 = No overflow
In I$^2$C mode:
1 = A byte is received while the SSPBUF 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 I$^2$C 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 I$^2$C Slave mode:
SCL release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)
In I$^2$C Master mode:
Unused in this mode

bit 3-0
**SSPM<3:0>:** Synchronous Serial Port Mode Select bits
0000 = SPI Master mode, clock = F$_{OSC}$/4
0001 = SPI Master mode, clock = F$_{OSC}$/16
0010 = SPI Master mode, clock = F$_{OSC}$/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ 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 = F$_{OSC}$ / (4 * (SSPADD+1))[4]
1001 = Reserved
1010 = SPI Master mode, clock = F$_{OSC}$/(4 * (SSPADD+1))[5]
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 SSPBUF 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:** SSPADD values of 0, 1 or 2 are not supported for I$^2$C mode.
**5:** SSPADD value of '0' is not supported. Use SSPM = 0000 instead.

# PIC16(L)F1782/3

**REGISTER 26-3:** **SSPCON2: SSP CONTROL REGISTER 2**

| R/W-0/0 | R-0/0 | R/W-0/0 | R/S/HS-0/0 | R/S/HS-0/0 | R/S/HS-0/0 | R/S/HS-0/0 | R/W/HS-0/0 |
|---------|-------|---------|------------|------------|------------|------------|------------|
| GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| 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 I$^2$C Slave mode only)
　　　　1 = Enable interrupt when a general call address (0x00 or 00h) is received in the SSPSR
　　　　0 = General call address disabled

bit 6　　**ACKSTAT:** Acknowledge Status bit (in I$^2$C mode only)
　　　　1 = Acknowledge was not received
　　　　0 = Acknowledge was received

bit 5　　**ACKDT:** Acknowledge Data bit (in I$^2$C 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:** Acknowledge Sequence Enable bit (in I$^2$C 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:** Receive Enable bit (in I$^2$C Master mode only)
　　　　1 = Enables Receive mode for I$^2$C
　　　　0 = Receive idle

bit 2　　**PEN:** Stop Condition Enable bit (in I$^2$C Master mode only)
　　　　SCKMSSP Release Control:
　　　　1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
　　　　0 = Stop condition Idle

bit 1　　**RSEN:** Repeated Start Condition Enable bit (in I$^2$C Master mode only)
　　　　1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
　　　　0 = Repeated Start condition Idle

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

**Note　1:** For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I$^2$C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

**Preliminary** © 2011-2012 Microchip Technology Inc.

## REGISTER 26-4: SSPCON3: SSP CONTROL REGISTER 3

| R-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|---------|---------|---------|---------|---------|---------|---------|
| 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$^2$C mode only)**(3)**

1 = Indicates the I$^2$C 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$^2$C 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$^2$C 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

In SPI Slave mode:**(1)**
    1 = SSPBUF 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 SSPSTAT register already set, SSPOV bit of the
        SSPCON1 register is set, and the buffer is not updated

In I$^2$C Master mode and SPI Master mode:
    This bit is ignored.

In I$^2$C Slave mode:
    1 = SSPBUF 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 = SSPBUF is only updated when SSPOV is clear

bit 3 **SDAHT:** SDA Hold Time Selection bit (I$^2$C 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$^2$C 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$^2$C Slave mode only)

1 = Following the 8th falling edge of SCL for a matching received address byte; CKP bit of the
    SSPCON1 register will be cleared and the SCL will be held low.
0 = Address holding is disabled

bit 0 **DHEN:** Data Hold Enable bit (I$^2$C Slave mode only)

1 = Following the 8th falling edge of SCL for a received data byte; slave hardware clears the CKP bit
    of the SSPCON1 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 SSPBUF.

**2:** This bit has no effect in Slave modes that Start and Stop condition detection is explicitly listed as enabled.

**3:** The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is set.

# PIC16(L)F1782/3

## REGISTER 26-5: SSPMSK: SSP MASK REGISTER

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | MSK<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 | |

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

bit 0 **MSK<0>:** Mask bit for I$^2$C Slave mode, 10-bit Address
I$^2$C Slave mode, 10-bit address (SSPM<3:0> = 0111 or 1111):
1 = The received address bit 0 is compared to SSPADD<0> to detect I$^2$C address match
0 = The received address bit 0 is not used to detect I$^2$C address match
I$^2$C Slave mode, 7-bit address, the bit is ignored

## REGISTER 26-6: SSPADD: MSSP ADDRESS AND BAUD RATE REGISTER (I$^2$C MODE)

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| | | | 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)/F$_{OSC}$

**10-Bit Slave mode — Most Significant Address Byte:**

bit 7-3 **Not used:** Unused for Most Significant Address byte. Bit state of this register is a "don't care". Bit pattern sent by master is fixed by I$^2$C specification and must be equal to '11110'. However, those bits are compared by hardware and are not affected by the value in this register.

bit 2-1 **ADD<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".

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 27.0 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 polarity in synchronous modes
- Sleep operation

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 27-1 and Figure 27-2.

**FIGURE 27-1:** **EUSART TRANSMIT BLOCK DIAGRAM**
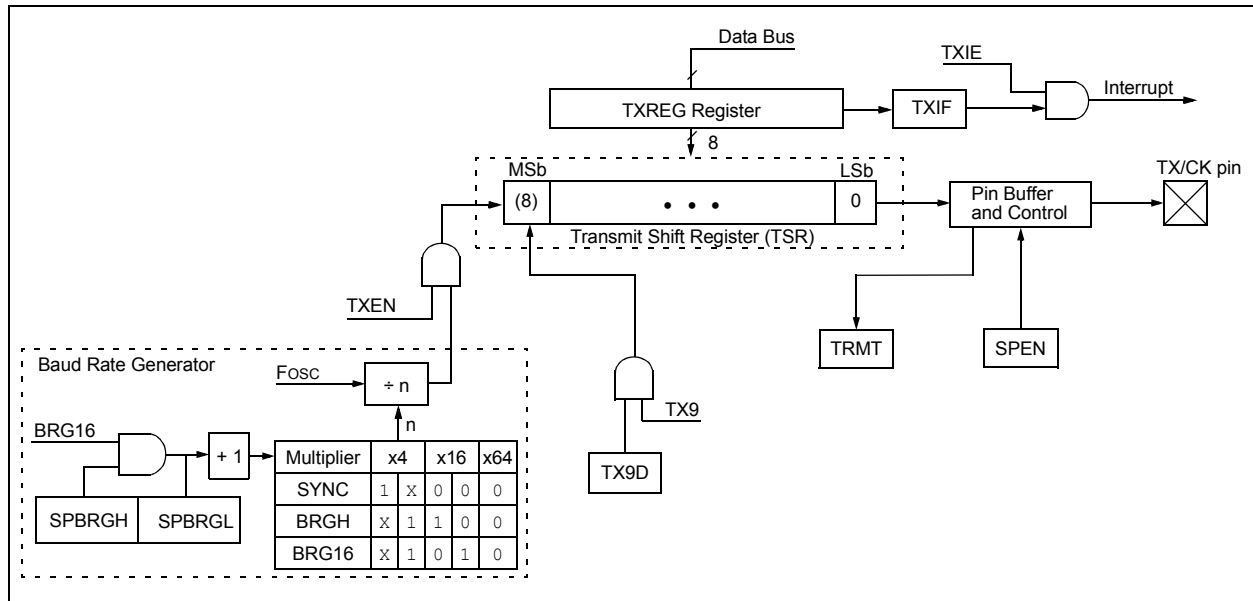
# PIC16(L)F1782/3

**FIGURE 27-2:** **EUSART RECEIVE BLOCK DIAGRAM**
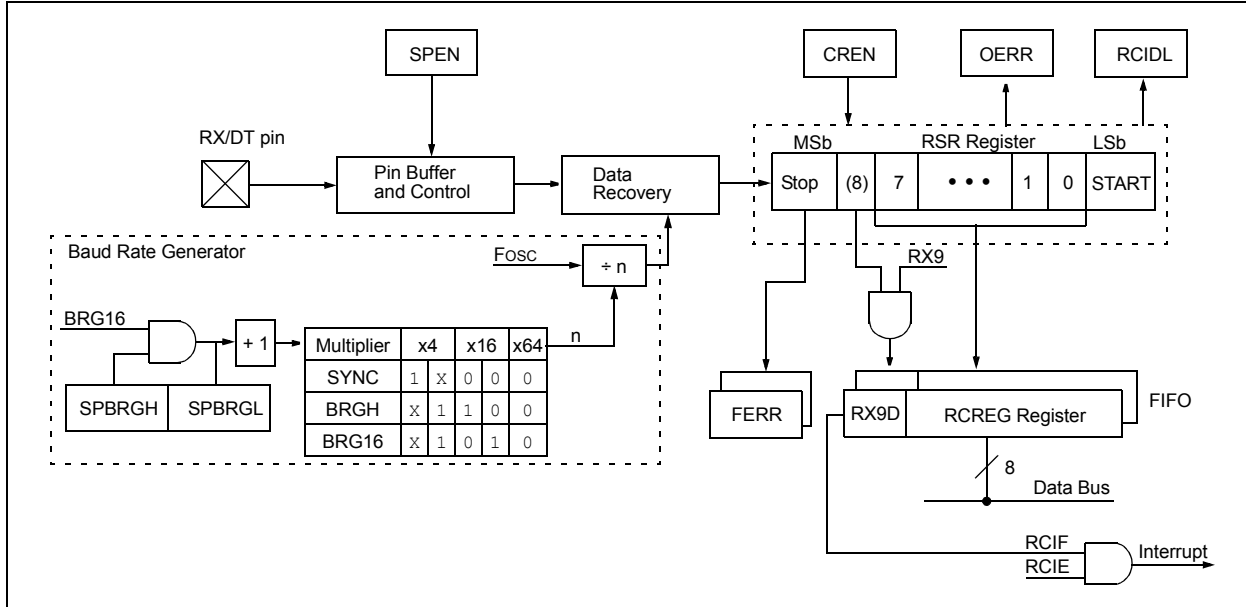


The operation of the EUSART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)

These registers are detailed in Register 27-1, Register 27-2 and Register 27-3, respectively.

When the receiver or transmitter section is not enabled then the corresponding RX or TX pin may be used for general purpose input and output.

## 27.1 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 V$_{OH}$ mark state which represents a '1' data bit, and a V$_{OL}$ 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 27-5 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.

### 27.1.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 27-1. The heart of the transmitter is the serial Transmit Shift Register (TSR), which is not directly accessible by software. The TSR obtains its data from the transmit buffer, which is the TXREG register.

#### 27.1.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- TXEN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the TXEN bit of the TXSTA register enables the transmitter circuitry of the EUSART. Clearing the SYNC bit of the TXSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTA register enables the EUSART and automatically configures the TX/CK I/O pin as an output. If the TX/CK pin is shared with an analog peripheral, the analog I/O function must be disabled by clearing the corresponding ANSEL bit.

> **Note:** The TXIF Transmitter Interrupt flag is set when the TXEN enable bit is set.

#### 27.1.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREG is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the TXREG until the Stop bit of the previous character has been transmitted. The pending character in the TXREG is then transferred to the TSR in one T$_{CY}$ immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXREG.

#### 27.1.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the SCKP bit of the BAUDxCON register. The default state of this bit is '0' which selects high true transmit Idle and data bits. Setting the SCKP bit to '1' will invert the transmit data resulting in low true Idle and data bits. The SCKP bit controls transmit data polarity in Asynchronous mode only. In Synchronous mode, the SCKP bit has a different function. See **Section 27.5.1.2 "Clock Polarity"**.

#### 27.1.1.4 Transmit Interrupt Flag

The TXIF interrupt flag bit of the PIR1 register is set whenever the EUSART transmitter is enabled and no character is being held for transmission in the TXREG. In other words, the TXIF bit is only clear when the TSR is busy with a character and a new character has been queued for transmission in the TXREG. The TXIF flag bit is not cleared immediately upon writing TXREG. TXIF becomes valid in the second instruction cycle following the write execution. Polling TXIF immediately following the TXREG write will return invalid results. The TXIF bit is read-only, it cannot be set or cleared by software.

The TXIF interrupt can be enabled by setting the TXIE interrupt enable bit of the PIE1 register. However, the TXIF flag bit will be set whenever the TXREG is empty, regardless of the state of TXIE enable bit.

To use interrupts when transmitting data, set the TXIE bit only when there is more data to send. Clear the TXIE interrupt enable bit upon writing the last character of the transmission to the TXREG.

### 27.1.1.5 TSR Status

The TRMT bit of the TXSTA register indicates the status of the TSR register. This is a read-only bit. The TRMT bit is set when the TSR register is empty and is cleared when a character is transferred to the TSR register from the TXREG. The TRMT bit remains clear until all bits have been shifted out of the TSR register. No interrupt logic is tied to this bit, so the user has 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.

### 27.1.1.6 Transmitting 9-Bit Characters

The EUSART supports 9-bit character transmissions. When the TX9 bit of the TXSTA register is set, the EUSART will shift 9 bits out for each character transmitted. The TX9D bit of the TXSTA register is the ninth, and Most Significant, data bit. When transmitting 9-bit data, the TX9D data bit must be written before writing the 8 Least Significant bits into the TXREG. All nine bits of data will be transferred to the TSR shift register immediately after the TXREG is written.

A special 9-bit Address mode is available for use with multiple receivers. See **Section 27.1.2.7 "Address Detection"** for more information on the address mode.

### 27.1.1.7 Asynchronous Transmission Set-up:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 27.4 "EUSART Baud Rate Generator (BRG)"**).

2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.

3. If 9-bit transmission is desired, set the TX9 control bit. A set ninth data bit will indicate that the 8 Least Significant data bits are an address when the receiver is set for address detection.

4. Set SCKP bit if inverted transmit is desired.

5. Enable the transmission by setting the TXEN control bit. This will cause the TXIF interrupt bit to be set.

6. If interrupts are desired, set the TXIE interrupt enable bit of the PIE1 register. An interrupt will occur immediately provided that the GIE and PEIE bits of the INTCON register are also set.

7. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.

8. Load 8-bit data into the TXREG register. This will start the transmission.

**FIGURE 27-3: ASYNCHRONOUS TRANSMISSION**



**Preliminary** © 2011-2012 Microchip Technology Inc.
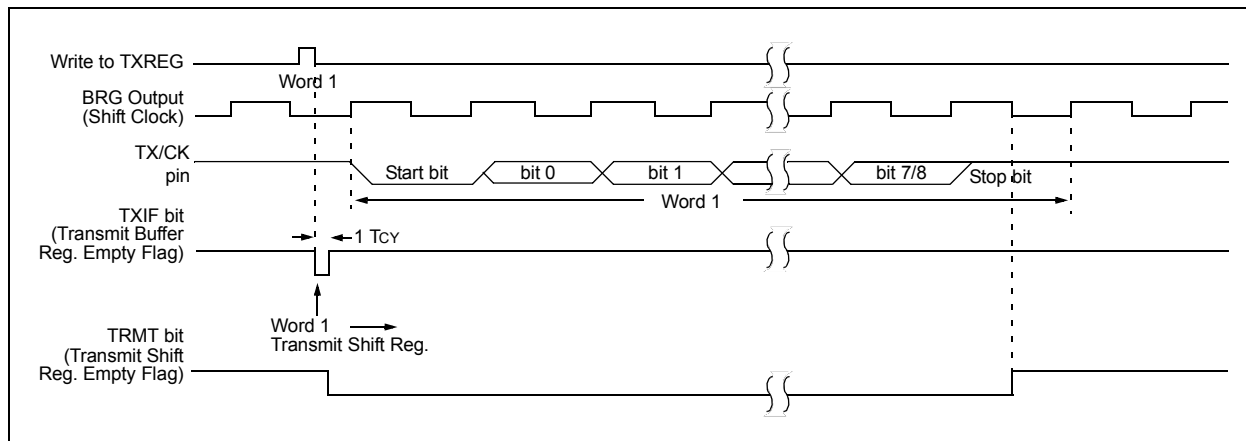
**FIGURE 27-4:** ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)



Note: This timing diagram shows two consecutive transmissions.
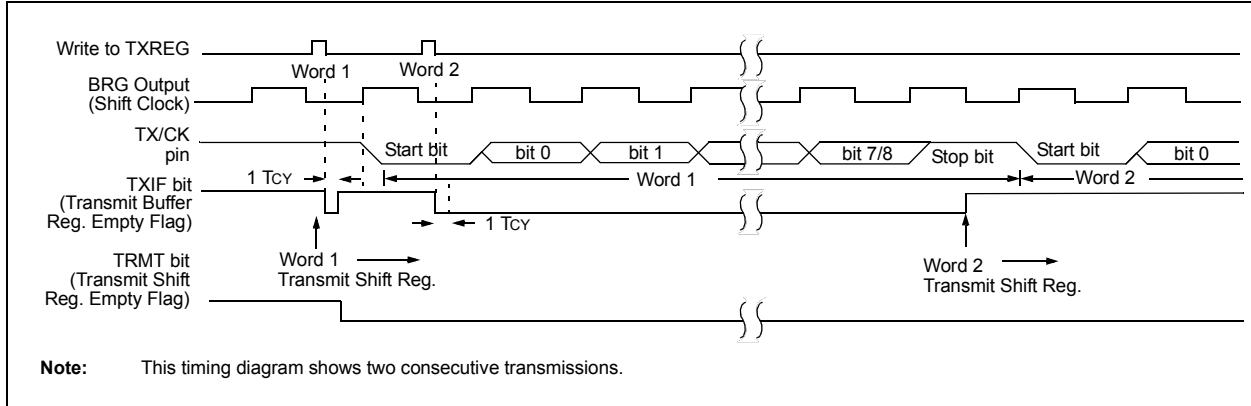
**TABLE 27-1:** SUMMARY OF REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| APFCON | C2OUTSEL | CC1PSEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 119 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 329 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 328 |
| SPBRGL | BRG<7:0> | | | | | | | | 330 |
| SPBRGH | BRG<15:8> | | | | | | | | 330 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |
| TXREG | EUSART Transmit Data Register | | | | | | | | 319* |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 327 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for asynchronous transmission.
   * Page provides register information.

## 27.1.2 EUSART ASYNCHRONOUS RECEIVER

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in Figure 27-2. The data is received on the RX/DT pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all 8 or 9 bits of the character have been shifted in, they are immediately transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must start servicing the EUSART receiver. The FIFO and RSR registers are not directly accessible by software. Access to the received data is via the RCREG register.

### 27.1.2.1 Enabling the Receiver

The EUSART receiver is enabled for asynchronous operation by configuring the following three control bits:

- CREN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the CREN bit of the RCSTA register enables the receiver circuitry of the EUSART. Clearing the SYNC bit of the TXSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTA register enables the EUSART. The programmer must set the corresponding TRIS bit to configure the RX/DT I/O pin as an input.

> **Note:** If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

### 27.1.2.2 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 27.1.2.4 "Receive Framing Error"** for more information on framing errors.

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the EUSART receive FIFO and the RCIF interrupt flag bit of the PIR1 register is set. The top character in the FIFO is transferred out of the FIFO by reading the RCREG register.

> **Note:** If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See **Section 27.1.2.5 "Receive Overrun Error"** for more information on overrun errors.

### 27.1.2.3 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 all of the following bits:

- RCIE, Interrupt Enable bit of the PIE1 register
- PEIE, Peripheral Interrupt Enable bit of the INTCON register
- GIE, Global Interrupt Enable bit of the INTCON register

The RCIF interrupt flag bit will be set when there is an unread character in the FIFO, regardless of the state of interrupt enable bits.

### 27.1.2.4 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error Status bit. A framing error indicates that a Stop bit was not seen at the expected time. The framing error status is accessed via the FERR bit of the RCSTA register. The FERR bit represents the status of the top unread character in the receive FIFO. Therefore, the FERR bit must be read before reading the RCREG.

The FERR bit is read-only and only applies to the top unread character in the receive FIFO. A framing error (FERR = 1) does not preclude reception of additional characters. It is not necessary to clear the FERR bit. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERR bit can be forced clear by clearing the SPEN bit of the RCSTA register which resets the EUSART. Clearing the CREN bit of the RCSTA register does not affect the FERR bit. A framing error by itself does not generate an interrupt.

> **Note:** If all receive characters in the receive FIFO have framing errors, repeated reads of the RCREG will not clear the FERR bit.

### 27.1.2.5 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before the FIFO is accessed. When this happens the OERR bit of the RCSTA register is set. The characters already in the FIFO buffer can be read but no additional characters will be received until the error is cleared. The error must be cleared by either clearing the CREN bit of the RCSTA register or by resetting the EUSART by clearing the SPEN bit of the RCSTA register.

### 27.1.2.6 Receiving 9-bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTA register is set the EUSART will shift 9 bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth and Most Significant data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the 8 Least Significant bits from the RCREG.

### 27.1.2.7 Address Detection

A special Address Detection mode is available for use when multiple receivers share the same transmission line, such as in RS-485 systems. Address detection is enabled by setting the ADDEN bit of the RCSTA register.

Address detection requires 9-bit character reception. When address detection is enabled, only characters with the ninth data bit set will be transferred to the receive FIFO buffer, thereby setting the RCIF interrupt bit. All other characters will be ignored.

Upon receiving an address character, user software determines if the address matches its own. Upon address match, user software must disable address detection by clearing the ADDEN bit before the next Stop bit occurs. When user software detects the end of the message, determined by the message protocol used, software places the receiver back into the Address Detection mode by setting the ADDEN bit.

# PIC16(L)F1782/3

## 27.1.2.8 Asynchronous Reception Set-up:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 27.4 "EUSART Baud Rate Generator (BRG)"**).

2. Clear the ANSEL bit for the RX pin (if applicable).

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 bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.

5. If 9-bit reception is desired, set the RX9 bit.

6. Enable reception by setting the CREN bit.

7. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.

8. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.

9. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register.

10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

## 27.1.2.9 9-bit Address Detection Mode Set-up

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 27.4 "EUSART Baud Rate Generator (BRG)"**).

2. Clear the ANSEL bit for the RX pin (if applicable).

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 bit of the PIE1 register and the GIE and PEIE 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. Enable reception by setting the CREN bit.

8. The RCIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.

9. Read the RCSTA register to get the error flags. The ninth data bit will always be set.

10. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register. Software determines if this is the device's address.

11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

### FIGURE 27-5: ASYNCHRONOUS RECEPTION



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

**Preliminary**

© 2011-2012 Microchip Technology Inc.
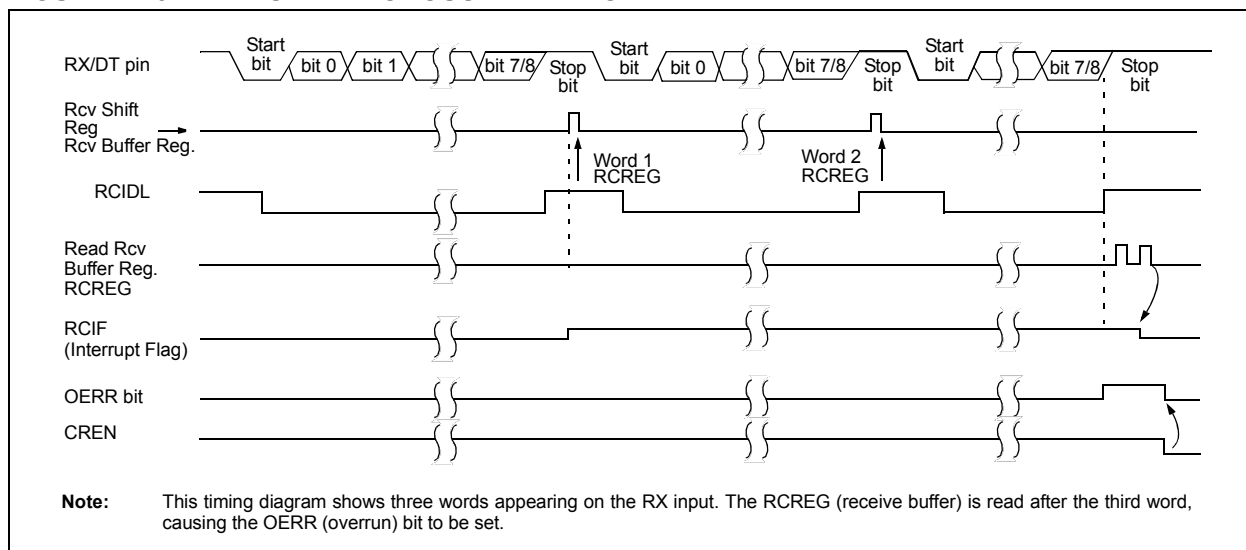
**TABLE 27-2: SUMMARY OF REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| APFCON | C2OUTSEL | CC1PSEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 119 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 329 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| RCREG | EUSART Receive Data Register | | | | | | | | 322* |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 328 |
| SPBRGL | BRG<7:0> | | | | | | | | 330 |
| SPBRGH | BRG<15:8> | | | | | | | | 330 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 327 |

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

\* Page provides register information.

## 27.2 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (INTOSC). However, the INTOSC frequency may drift as V$_{DD}$ 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 INTOSC output. Adjusting the value in the OSCTUNE register allows for fine resolution changes to the system clock source. See **Section 6.2.2 "Internal Clock Sources"** 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 27.4.1 "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.

## 27.3    Register Definitions: EUSART Control

### REGISTER 27-1:    TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

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

| Legend: | | |
|---------|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| 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       **CSRC:** Clock Source Select bit
<u>Asynchronous mode</u>:
Don't care
<u>Synchronous mode</u>:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

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

bit 5       **TXEN:** Transmit Enable bit[1]
1 = Transmit enabled
0 = Transmit disabled

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

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

bit 2       **BRGH:** High Baud Rate Select bit
<u>Asynchronous mode</u>:
1 = High speed
0 = Low speed
<u>Synchronous mode</u>:
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.

# PIC16(L)F1782/3

## REGISTER 27-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER [1]

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-0/0 | R-0/0 | R-0/0 |
|---------|---------|---------|---------|---------|-------|-------|-------|
| 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' |
| 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     **SPEN:** Serial Port Enable bit
          1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
          0 = Serial port disabled (held in Reset)

bit 6     **RX9:** 9-bit Receive Enable bit
          1 = Selects 9-bit reception
          0 = Selects 8-bit reception

bit 5     **SREN:** Single Receive Enable bit
          Asynchronous mode:
          Don't care
          Synchronous mode – Master:
          1 = Enables single receive
          0 = Disables single receive
          This bit is cleared after reception is complete.
          Synchronous mode – Slave
          Don't care

bit 4     **CREN:** Continuous Receive Enable bit
          Asynchronous mode:
          1 = Enables receiver
          0 = Disables receiver
          Synchronous mode:
          1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
          0 = Disables continuous receive

bit 3     **ADDEN:** Address Detect Enable bit
          Asynchronous mode 9-bit (RX9 = 1):
          1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set
          0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
          Asynchronous mode 8-bit (RX9 = 0):
          Don't care

bit 2     **FERR:** Framing Error bit
          1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
          0 = No framing error

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

bit 0     **RX9D:** Ninth bit of Received Data
          This can be address/data bit or a parity bit and must be calculated by user firmware.

**Preliminary**          © 2011-2012 Microchip Technology Inc.

## REGISTER 27-3: BAUDCON: BAUD RATE CONTROL REGISTER

| R-0/0 | R-1/1 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 |
|-------|-------|-----|---------|---------|-----|---------|---------|
| ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN |
| 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 **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 received and the receiver is receiving
Synchronous mode:
Don't care

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

bit 4 **SCKP**: Synchronous Clock Polarity Select bit

Asynchronous mode:

1 = Transmit inverted data to the TX/CK pin
0 = Transmit non-inverted data to the TX/CK pin

Synchronous mode:
1 = Data is clocked on rising edge of the clock
0 = Data is clocked on falling edge of the clock

bit 3 **BRG16:** 16-bit Baud Rate Generator bit

1 = 16-bit Baud Rate Generator is used
0 = 8-bit Baud Rate Generator is used

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

bit 1 **WUE:** Wake-up Enable bit

Asynchronous mode:

1 = Receiver is waiting for a falling edge. No character will be received, byte RCIF will be set. WUE will automatically clear after RCIF is set.
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

# PIC16(L)F1782/3

## 27.4 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 BAUDCON register selects 16-bit mode.

The SPBRGH, SPBRGL register pair determines the period of the free running baud rate timer. In Asynchronous mode the multiplier of the baud rate period is determined by both the BRGH bit of the TXSTA register and the BRG16 bit of the BAUDCON register. In Synchronous mode, the BRGH bit is ignored.

Table 27-3 contains the formulas for determining the baud rate. Example 27-1 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 27-3. It may be advantageous to use the high baud rate (BRGH = `1`), or the 16-bit BRG (BRG16 = `1`) to reduce the baud rate error. The 16-bit BRG mode is used to achieve slow baud rates for fast oscillator frequencies.

Writing a new value to the SPBRGH, SPBRGL 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 27-1: CALCULATING BAUD RATE ERROR**

For a device with $F_{OSC}$ of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$Desired\ Baud\ Rate = \frac{F_{OSC}}{64([SPBRGH:SPBRGL]+1)}$$

Solving for SPBRGH:SPBRGL:

$$X = \frac{\frac{F_{OSC}}{Desired\ Baud\ Rate}}{64} - 1$$

$$= \frac{\frac{16000000}{9600}}{64} - 1$$

$$= [25.042] = 25$$

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

$$= 9615$$

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

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

**Preliminary**

**TABLE 27-3: BAUD RATE FORMULAS**

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|---|---|---|---|---|
| SYNC | BRG16 | BRGH | | |
| 0 | 0 | 0 | 8-bit/Asynchronous | $F_{OSC}/[64\,(n+1)]$ |
| 0 | 0 | 1 | 8-bit/Asynchronous | $F_{OSC}/[16\,(n+1)]$ |
| 0 | 1 | 0 | 16-bit/Asynchronous | |
| 0 | 1 | 1 | 16-bit/Asynchronous | $F_{OSC}/[4\,(n+1)]$ |
| 1 | 0 | x | 8-bit/Synchronous | |
| 1 | 1 | x | 16-bit/Synchronous | |

**Legend:** x = Don't care, n = value of SPBRGH, SPBRGL register pair

**TABLE 27-4: SUMMARY OF REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 329 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 328 |
| SPBRGL | BRG<7:0> | | | | | | | | 330 |
| SPBRGH | BRG<15:8> | | | | | | | | 330 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 327 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for the baud rate generator.

\* Page provides register information.

# PIC16(L)F1782/3

## TABLE 27-5: BAUD RATES FOR ASYNCHRONOUS MODES

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fosc = 32.000 MHz | | | Fosc = 20.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1200 | — | — | — | 1221 | 1.73 | 255 | 1200 | 0.00 | 239 | 1200 | 0.00 | 143 |
| 2400 | 2404 | 0.16 | 207 | 2404 | 0.16 | 129 | 2400 | 0.00 | 119 | 2400 | 0.00 | 71 |
| 9600 | 9615 | 0.16 | 51 | 9470 | -1.36 | 32 | 9600 | 0.00 | 29 | 9600 | 0.00 | 17 |
| 10417 | 10417 | 0.00 | 47 | 10417 | 0.00 | 29 | 10286 | -1.26 | 27 | 10165 | -2.42 | 16 |
| 19.2k | 19.23k | 0.16 | 25 | 19.53k | 1.73 | 15 | 19.20k | 0.00 | 14 | 19.20k | 0.00 | 8 |
| 57.6k | 55.55k | -3.55 | 3 | — | — | — | 57.60k | 0.00 | 7 | 57.60k | 0.00 | 2 |
| 115.2k | — | — | — | — | — | — | — | — | — | — | — | — |

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fosc = 8.000 MHz | | | Fosc = 4.000 MHz | | | Fosc = 3.6864 MHz | | | Fosc = 1.000 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | 300 | 0.16 | 207 | 300 | 0.00 | 191 | 300 | 0.16 | 51 |
| 1200 | 1202 | 0.16 | 103 | 1202 | 0.16 | 51 | 1200 | 0.00 | 47 | 1202 | 0.16 | 12 |
| 2400 | 2404 | 0.16 | 51 | 2404 | 0.16 | 25 | 2400 | 0.00 | 23 | — | — | — |
| 9600 | 9615 | 0.16 | 12 | — | — | — | 9600 | 0.00 | 5 | — | — | — |
| 10417 | 10417 | 0.00 | 11 | 10417 | 0.00 | 5 | — | — | — | — | — | — |
| 19.2k | — | — | — | — | — | — | 19.20k | 0.00 | 2 | — | — | — |
| 57.6k | — | — | — | — | — | — | 57.60k | 0.00 | 0 | — | — | — |
| 115.2k | — | — | — | — | — | — | — | — | — | — | — | — |

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fosc = 32.000 MHz | | | Fosc = 20.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1200 | — | — | — | — | — | — | — | — | — | — | — | — |
| 2400 | — | — | — | — | — | — | — | — | — | — | — | — |
| 9600 | 9615 | 0.16 | 207 | 9615 | 0.16 | 129 | 9600 | 0.00 | 119 | 9600 | 0.00 | 71 |
| 10417 | 10417 | 0.00 | 191 | 10417 | 0.00 | 119 | 10378 | -0.37 | 110 | 10473 | 0.53 | 65 |
| 19.2k | 19.23k | 0.16 | 103 | 19.23k | 0.16 | 64 | 19.20k | 0.00 | 59 | 19.20k | 0.00 | 35 |
| 57.6k | 57.14k | -0.79 | 34 | 56.82k | -1.36 | 21 | 57.60k | 0.00 | 19 | 57.60k | 0.00 | 11 |
| 115.2k | 117.64k | 2.12 | 16 | 113.64k | -1.36 | 10 | 115.2k | 0.00 | 9 | 115.2k | 0.00 | 5 |

**Preliminary**

© 2011-2012 Microchip Technology Inc.

**TABLE 27-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)**

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fosc = 8.000 MHz | | | Fosc = 4.000 MHz | | | Fosc = 3.6864 MHz | | | Fosc = 1.000 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | — | — | — | — | — | — | 300 | 0.16 | 207 |
| 1200 | — | — | — | 1202 | 0.16 | 207 | 1200 | 0.00 | 191 | 1202 | 0.16 | 51 |
| 2400 | 2404 | 0.16 | 207 | 2404 | 0.16 | 103 | 2400 | 0.00 | 95 | 2404 | 0.16 | 25 |
| 9600 | 9615 | 0.16 | 51 | 9615 | 0.16 | 25 | 9600 | 0.00 | 23 | — | — | — |
| 10417 | 10417 | 0.00 | 47 | 10417 | 0.00 | 23 | 10473 | 0.53 | 21 | 10417 | 0.00 | 5 |
| 19.2k | 19231 | 0.16 | 25 | 19.23k | 0.16 | 12 | 19.2k | 0.00 | 11 | — | — | — |
| 57.6k | 55556 | -3.55 | 8 | — | — | — | 57.60k | 0.00 | 3 | — | — | — |
| 115.2k | — | — | — | — | — | — | 115.2k | 0.00 | 1 | — | — | — |

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fosc = 32.000 MHz | | | Fosc = 20.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | 300.0 | 0.00 | 6666 | 300.0 | -0.01 | 4166 | 300.0 | 0.00 | 3839 | 300.0 | 0.00 | 2303 |
| 1200 | 1200 | -0.02 | 3332 | 1200 | -0.03 | 1041 | 1200 | 0.00 | 959 | 1200 | 0.00 | 575 |
| 2400 | 2401 | -0.04 | 832 | 2399 | -0.03 | 520 | 2400 | 0.00 | 479 | 2400 | 0.00 | 287 |
| 9600 | 9615 | 0.16 | 207 | 9615 | 0.16 | 129 | 9600 | 0.00 | 119 | 9600 | 0.00 | 71 |
| 10417 | 10417 | 0.00 | 191 | 10417 | 0.00 | 119 | 10378 | -0.37 | 110 | 10473 | 0.53 | 65 |
| 19.2k | 19.23k | 0.16 | 103 | 19.23k | 0.16 | 64 | 19.20k | 0.00 | 59 | 19.20k | 0.00 | 35 |
| 57.6k | 57.14k | -0.79 | 34 | 56.818 | -1.36 | 21 | 57.60k | 0.00 | 19 | 57.60k | 0.00 | 11 |
| 115.2k | 117.6k | 2.12 | 16 | 113.636 | -1.36 | 10 | 115.2k | 0.00 | 9 | 115.2k | 0.00 | 5 |

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Fosc = 8.000 MHz | | | Fosc = 4.000 MHz | | | Fosc = 3.6864 MHz | | | Fosc = 1.000 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (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 | — | — | — |

# PIC16(L)F1782/3

## TABLE 27-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 32.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 18.432 MHz | | | FOSC = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | 300.0 | 0.00 | 26666 | 300.0 | 0.00 | 16665 | 300.0 | 0.00 | 15359 | 300.0 | 0.00 | 9215 |
| 1200 | 1200 | 0.00 | 6666 | 1200 | -0.01 | 4166 | 1200 | 0.00 | 3839 | 1200 | 0.00 | 2303 |
| 2400 | 2400 | 0.01 | 3332 | 2400 | 0.02 | 2082 | 2400 | 0.00 | 1919 | 2400 | 0.00 | 1151 |
| 9600 | 9604 | 0.04 | 832 | 9597 | -0.03 | 520 | 9600 | 0.00 | 479 | 9600 | 0.00 | 287 |
| 10417 | 10417 | 0.00 | 767 | 10417 | 0.00 | 479 | 10425 | 0.08 | 441 | 10433 | 0.16 | 264 |
| 19.2k | 19.18k | -0.08 | 416 | 19.23k | 0.16 | 259 | 19.20k | 0.00 | 239 | 19.20k | 0.00 | 143 |
| 57.6k | 57.55k | -0.08 | 138 | 57.47k | -0.22 | 86 | 57.60k | 0.00 | 79 | 57.60k | 0.00 | 47 |
| 115.2k | 115.9k | 0.64 | 68 | 116.3k | 0.94 | 42 | 115.2k | 0.00 | 39 | 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 | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (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 | 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 | — | — | — |

**Preliminary**

## 27.4.1 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 BAUDCON register starts the auto-baud calibration sequence (Figure 27-6). While the ABD sequence takes place, the EUSART state machine is held in Idle. On the first rising edge of the receive line, after the Start bit, the SPBRG begins counting up using the BRG counter clock as shown in Table 27-6. The fifth rising edge will occur on the RX pin at the end of the eighth bit period. At that time, an accumulated value totaling the proper BRG period is left in the SPBRGH, SPBRGL register pair, the ABDEN bit is automatically cleared and the RCIF interrupt flag is set. The value in the RCREG needs to be read to clear the RCIF interrupt. RCREG content should be discarded. When calibrating for modes that do not use the SPBRGH register the user can verify that the SPBRGL register did not overflow by checking for 00h in the SPBRGH register.

The BRG auto-baud clock is determined by the BRG16 and BRGH bits as shown in Table 27-6. During ABD, both the SPBRGH and SPBRGL registers are used as a 16-bit counter, independent of the BRG16 bit setting. While calibrating the baud rate period, the SPBRGH and SPBRGL 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 27.4.3 "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 SPBRGH:SPBRGL register pair.

**TABLE 27-6: BRG COUNTER CLOCK RATES**

| BRG16 | BRGH | BRG Base Clock | BRG ABD Clock |
|-------|------|----------------|---------------|
| 0 | 0 | $F_{OSC}/64$ | $F_{OSC}/512$ |
| 0 | 1 | $F_{OSC}/16$ | $F_{OSC}/128$ |
| 1 | 0 | $F_{OSC}/16$ | $F_{OSC}/128$ |
| 1 | 1 | $F_{OSC}/4$ | $F_{OSC}/32$ |

**Note:** During the ABD sequence, SPBRGL and SPBRGH registers are both used as a 16-bit counter, independent of BRG16 setting.

**FIGURE 27-6: AUTOMATIC BAUD RATE CALIBRATION**



**Note 1:** The ABD sequence requires the **EUSART** module to be configured in Asynchronous mode.
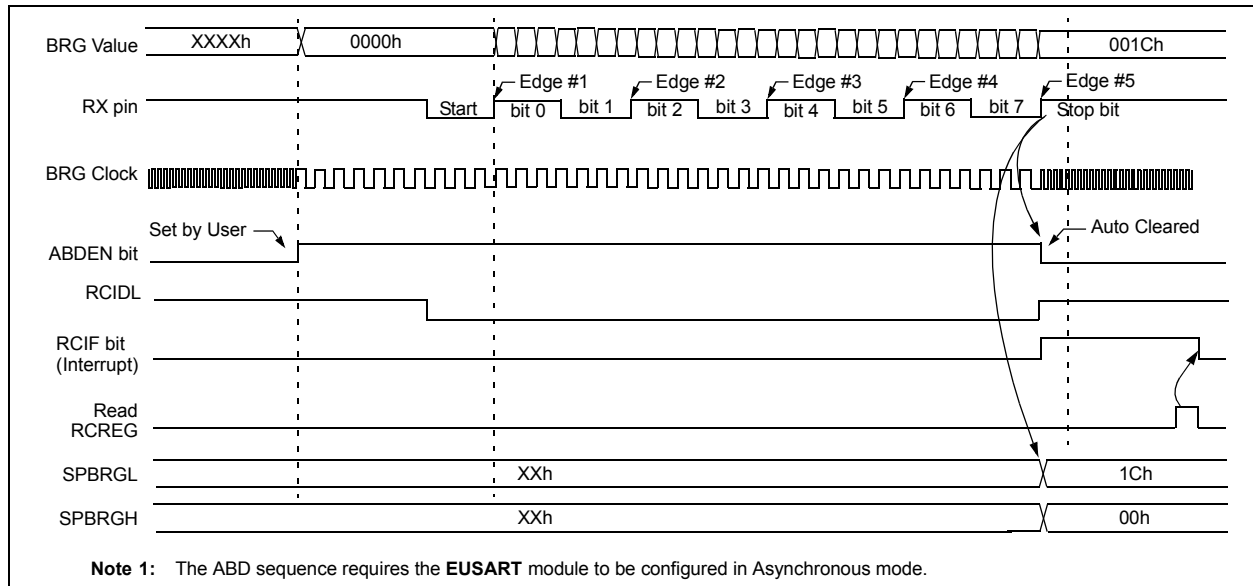
# PIC16(L)F1782/3

## 27.4.2 AUTO-BAUD OVERFLOW

During the course of automatic baud detection, the ABDOVF bit of the BAUDCON register will be set if the baud rate counter overflows before the fifth rising edge is detected on the RX pin. The ABDOVF bit indicates that the counter has exceeded the maximum count that can fit in the 16 bits of the SPBRGH:SPBRGL register pair. After the ABDOVF bit has been set, the counter continues to count until the fifth rising edge is detected on the RX pin. Upon detecting the fifth RX edge, the hardware will set the RCIF interrupt flag and clear the ABDEN bit of the BAUDCON register. The RCIF flag can be subsequently cleared by reading the RCREG register. The ABDOVF flag of the BAUDCON register can be cleared by software directly.

To terminate the auto-baud process before the RCIF flag is set, clear the ABDEN bit then clear the ABDOVF bit of the BAUDCON register. The ABDOVF bit will remain set if the ABDEN bit is not cleared first.

## 27.4.3 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 BAUDCON register. Once set, the normal receive sequence on RX/DT is disabled, and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a Sync Break or a wake-up signal character for the LIN protocol.)

The EUSART module generates an RCIF interrupt coincident with the wake-up event. The interrupt is generated synchronously to the Q clocks in normal CPU operating modes (Figure 27-7), and asynchronously if the device is in Sleep mode (Figure 27-8). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared by the low-to-high transition on the RX line at the end of the Break. This signals to the user that the Break event is over. At this point, the EUSART module is in Idle mode waiting to receive the next character.

## 27.4.3.1 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 Start-up Time

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

### WUE Bit

The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared in hardware by a rising edge on RX/DT. The interrupt condition is then cleared in software by reading the RCREG register and discarding its contents.

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

**FIGURE 27-7:** **AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION**



**Note 1:** The **EUSART** remains in Idle while the WUE bit is set.
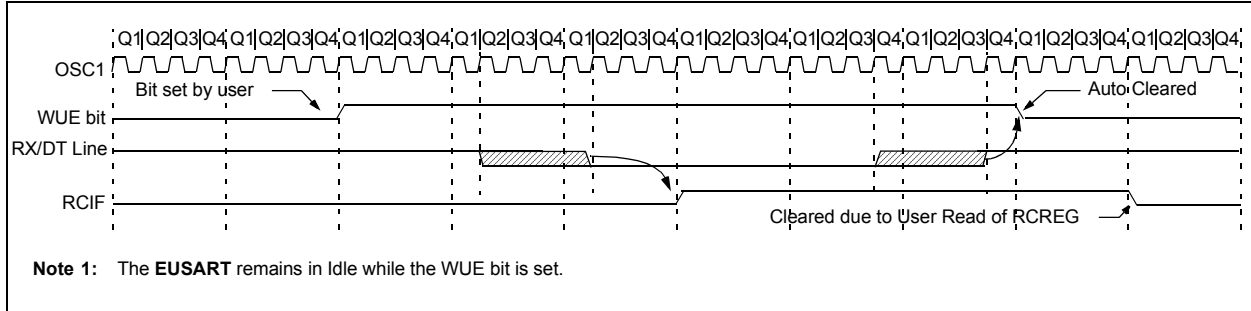
**FIGURE 27-8:** **AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**



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

  **2:** The **EUSART** remains in Idle while the WUE bit is set.
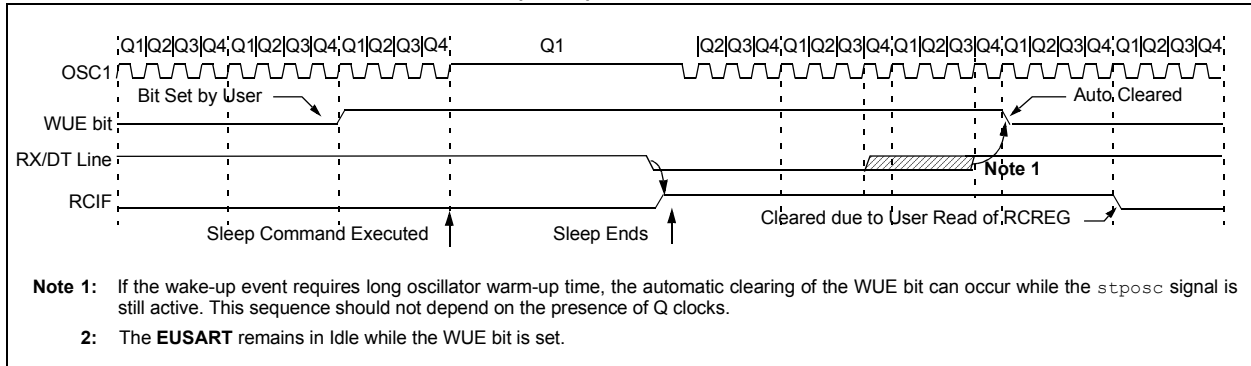
### 27.4.4 BREAK CHARACTER SEQUENCE

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 TXSTA register. The Break character transmission is then initiated by a write to the TXREG. The value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

The TRMT bit of the TXSTA register indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 27-9 for the timing of the Break character sequence.

#### 27.4.4.1 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 TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware and the Sync character is then transmitted.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

### 27.4.5 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 RCSTA register and the Received data as indicated by RCREG. The Baud Rate Generator is assumed to have been initialized to the expected baud rate.

A Break character has been received when;

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

The second method uses the Auto-Wake-up feature described in **Section 27.4.3 "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 BAUDCON register before placing the EUSART in Sleep mode.

**FIGURE 27-9:     SEND BREAK CHARACTER SEQUENCE**



**Preliminary** © 2011-2012 Microchip Technology Inc.
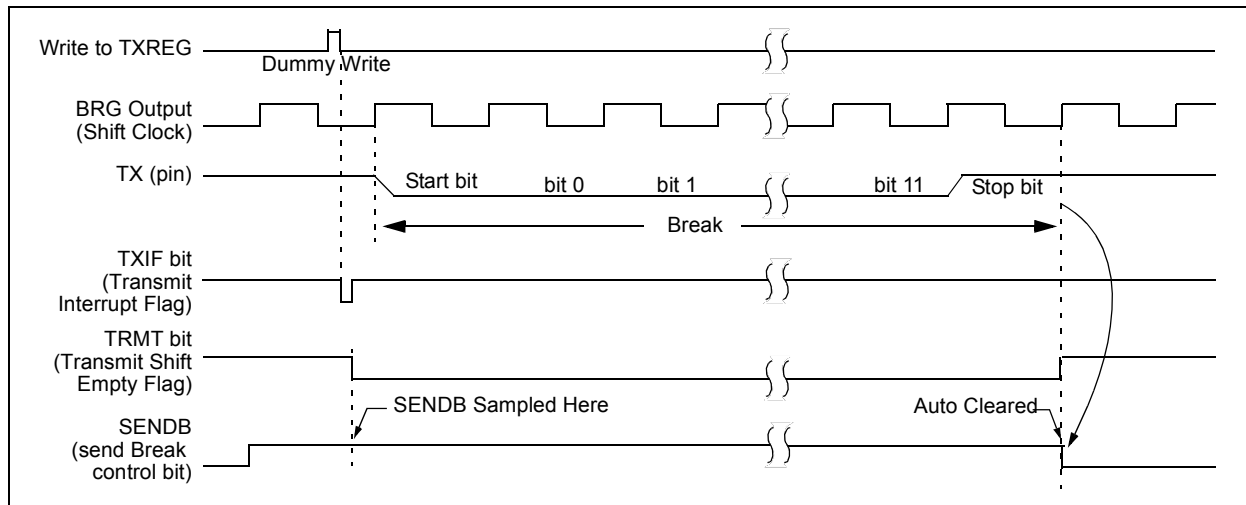
## 27.5  EUSART Synchronous Mode

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.

### 27.5.1  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 TXSTA register configures the device for synchronous operation. Setting the CSRC bit of the TXSTA register configures the device as a master. Clearing the SREN and CREN bits of the RCSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTA register enables the EUSART.

#### 27.5.1.1  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.

#### 27.5.1.2  Clock Polarity

A clock polarity option is provided for Microwire compatibility. Clock polarity is selected with the SCKP bit of the BAUDCON register. Setting the SCKP bit sets the clock Idle state as high. When the SCKP bit is set, the data changes on the falling edge of each clock.

Clearing the SCKP bit sets the Idle state as low. When the SCKP bit is cleared, the data changes on the rising edge of each clock.

#### 27.5.1.3  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 TXREG register. If the TSR still contains all or part of a previous character the new character data is held in the TXREG until the last bit of the previous character has been transmitted. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREG is immediately transferred to the TSR. The transmission of the character commences immediately following the transfer of the data to the TSR from the TXREG.

Each data bit changes on the leading edge of the master clock and remains valid until the subsequent leading clock edge.

| Note: | The TSR register is not mapped in data memory, so it is not available to the user. |
|---|---|

#### 27.5.1.4  Synchronous Master Transmission Set-up:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 27.4 "EUSART Baud Rate Generator (BRG)"**).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Disable Receive mode by clearing bits SREN and CREN.
4. Enable Transmit mode by setting the TXEN bit.
5. If 9-bit transmission is desired, set the TX9 bit.
6. If interrupts are desired, set the TXIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
7. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
8. Start transmission by loading data to the TXREG register.

**FIGURE 27-10:** **SYNCHRONOUS TRANSMISSION**



Note: Sync Master mode, SPBRGL = 0, continuous transmission of two 8-bit words.
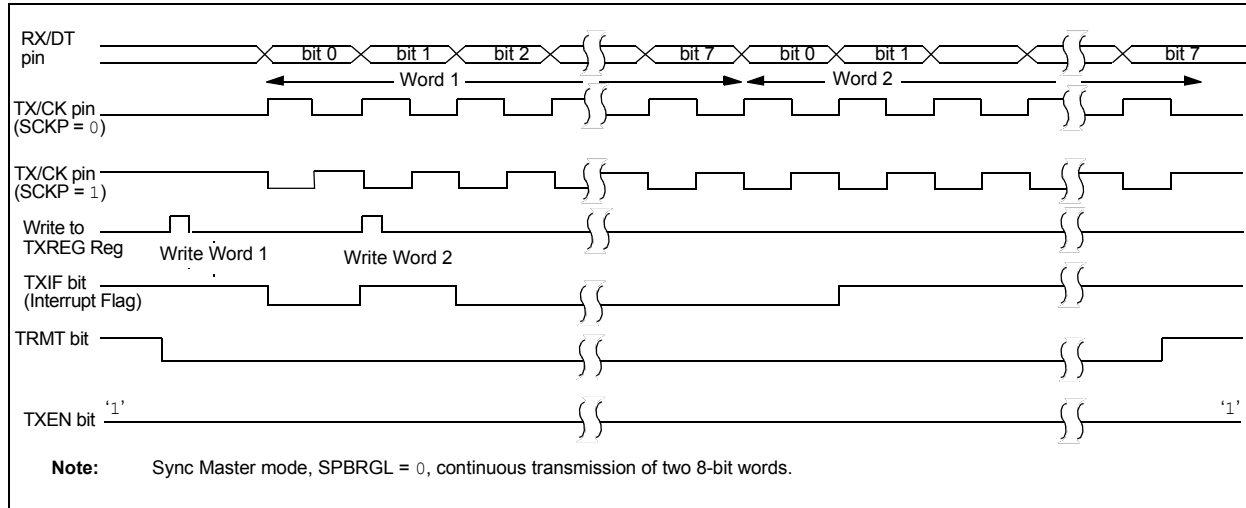
**FIGURE 27-11:** **SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**
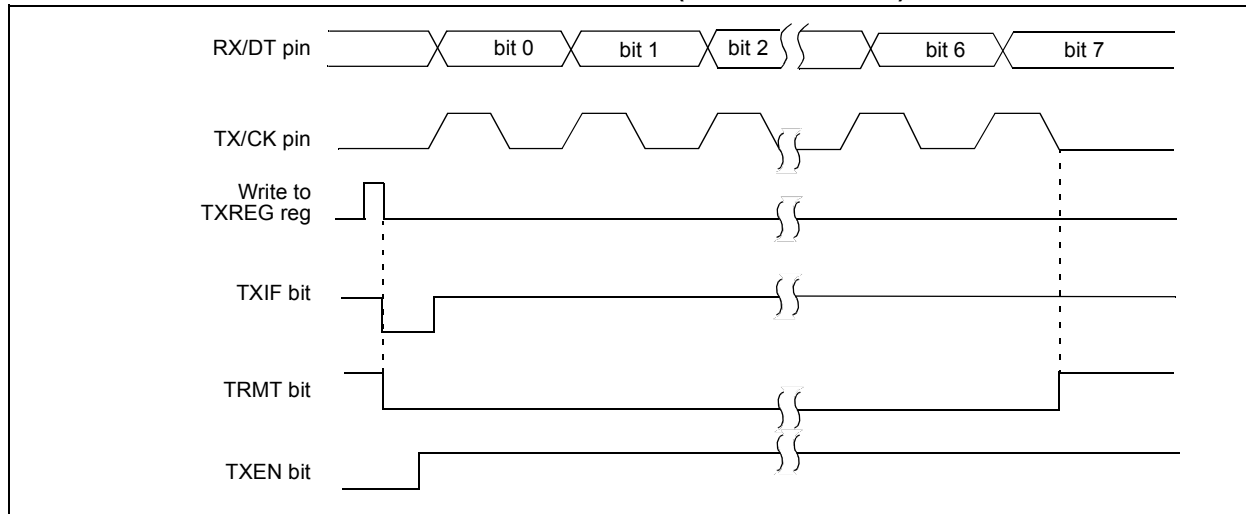


**TABLE 27-7:** **SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| APFCON | C2OUTSEL | CC1PSEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 119 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 329 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 328 |
| SPBRGL | BRG<7:0> | | | | | | | | 330 |
| SPBRGH | BRG<15:8> | | | | | | | | 330 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |
| TXREG | EUSART Transmit Data Register | | | | | | | | 319* |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 327 |

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

**Preliminary** © 2011-2012 Microchip Technology Inc.

### 27.5.1.5    Synchronous Master Reception

Data is received at the RX/DT pin. The RX/DT pin output driver is automatically disabled when the EUSART is configured for synchronous master receive operation.

In Synchronous mode, reception is enabled by setting either the Single Receive Enable bit (SREN of the RCSTA register) or the Continuous Receive Enable bit (CREN of the RCSTA register).

When SREN is set and CREN is clear, only as many clock cycles are generated as there are data bits in a single character. The SREN bit is automatically cleared at the completion of one character. When CREN is set, clocks are continuously generated until CREN is cleared. If CREN is cleared in the middle of a character the CK clock stops immediately and the partial character is discarded. If SREN and CREN are both set, then SREN is cleared at the completion of the first character and CREN takes precedence.

To initiate reception, set either SREN or CREN. Data is sampled at the RX/DT pin on the trailing edge of the TX/CK clock pin and is shifted into the Receive Shift Register (RSR). When a complete character is received into the RSR, the RCIF bit is set and the character is automatically transferred to the two character receive FIFO. The Least Significant eight bits of the top character in the receive FIFO are available in RCREG. The RCIF bit remains set as long as there are unread characters in the receive FIFO.

> **Note:** If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

### 27.5.1.6    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 is automatically disabled 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.

> **Note:** If the device is configured as a slave and the TX/CK function is on an analog pin, the corresponding ANSEL bit must be cleared.

### 27.5.1.7    Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before RCREG is read to access the FIFO. When this happens the OERR bit of the RCSTA register is set. Previous data in the FIFO will not be overwritten. The two characters in the FIFO buffer can be read, however, no additional characters

will be received until the error is cleared. The OERR bit can only be cleared by clearing the overrun condition. If the overrun error occurred when the SREN bit is set and CREN is clear then the error is cleared by reading RCREG. If the overrun occurred when the CREN bit is set then the error condition is cleared by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

### 27.5.1.8    Receiving 9-bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTA register is set the EUSART will shift 9-bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth, and Most Significant, data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the 8 Least Significant bits from the RCREG.

### 27.5.1.9    Synchronous Master Reception Set-up:

1. Initialize the SPBRGH, SPBRGL register pair for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.

2. Clear the ANSEL bit for the RX pin (if applicable).

3. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.

4. Ensure bits CREN and SREN are clear.

5. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.

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 RCSTA 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 RCREG register.

11. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

# PIC16(L)F1782/3

**FIGURE 27-12:** **SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**



**Note:** Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0.
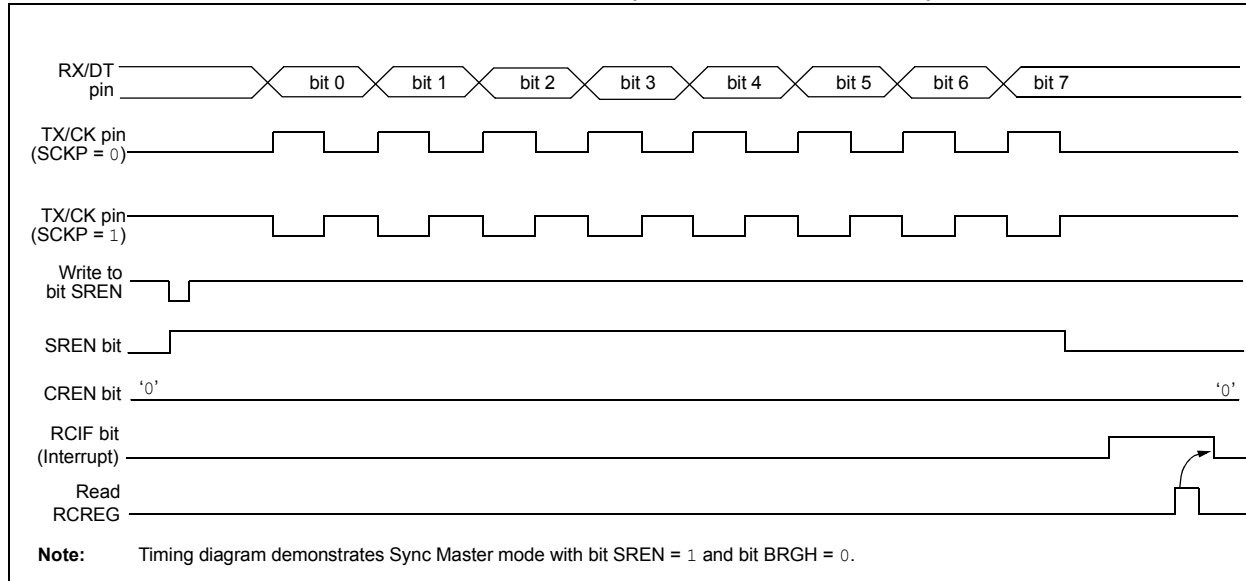
**TABLE 27-8:** **SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| APFCON | C2OUTSEL | CC1PSEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 119 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 329 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| RCREG | EUSART Receive Data Register | | | | | | | | 322* |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 328 |
| SPBRGL | BRG<7:0> | | | | | | | | 330 |
| SPBRGH | BRG<15:8> | | | | | | | | 330 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 327 |

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

**Preliminary**

## 27.5.2 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 TXSTA register configures the device for synchronous operation. Clearing the CSRC bit of the TXSTA register configures the device as a slave. Clearing the SREN and CREN bits of the RCSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTA register enables the EUSART.

### 27.5.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see **Section 27.5.1.3 "Synchronous Master Transmission"**), except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

1. The first character will immediately transfer to the TSR register and transmit.
2. The second word will remain in TXREG register.
3. The TXIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXREG register will transfer the second character to the TSR and the TXIF bit will now be set.
5. If the PEIE and TXIE bits are set, the interrupt will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will call the Interrupt Service Routine.

### 27.5.2.2 Synchronous Slave Transmission Set-up:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for the CK pin (if applicable).
3. Clear the CREN and SREN bits.
4. If interrupts are desired, set the TXIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
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 8 bits to the TXREG register.

**TABLE 27-9: SUMMARY OF 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 |
|---|---|---|---|---|---|---|---|---|---|
| APFCON | C2OUTSEL | CC1PSEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 119 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 329 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 328 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |
| TXREG | EUSART Transmit Data Register | | | | | | | | 319* |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 327 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for synchronous slave transmission.
 * Page provides register information.

# PIC16(L)F1782/3

### 27.5.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (**Section 27.5.1.5 "Synchronous Master Reception"**), with the following exceptions:

• Sleep
• CREN bit is always set, therefore the receiver is never Idle
• SREN bit, which is a "don't care" in Slave mode

A character may be received while in Sleep mode by setting the CREN bit prior to entering Sleep. Once the word is received, the RSR register will transfer the data to the RCREG register. If the RCIE enable bit is set, the interrupt generated will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will branch to the interrupt vector.

### 27.5.2.4 Synchronous Slave Reception Set-up:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for both the CK and DT pins (if applicable).
3. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
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 RCSTA register.
8. Retrieve the 8 Least Significant bits from the receive FIFO by reading the RCREG register.
9. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

### TABLE 27-10: SUMMARY OF 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 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|------------------|
| APFCON | C2OUTSEL | CC1PSEL | SDOSEL | SCKSEL | SDISEL | TXSEL | RXSEL | CCP2SEL | 119 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 329 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | ADIF | RCIF | TXIF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 88 |
| RCREG | EUSART Receive Data Register | | | | | | | | 322* |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 328 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 133 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 327 |

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

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 27.6 EUSART Operation During Sleep

The EUSART will remain active during Sleep only in the Synchronous Slave mode. All other modes require the system clock and therefore cannot generate the necessary signals to run the Transmit or Receive Shift registers during Sleep.

Synchronous Slave mode uses an externally generated clock to run the Transmit and Receive Shift registers.

### 27.6.1 SYNCHRONOUS RECEIVE DURING SLEEP

To receive during Sleep, all the following conditions must be met before entering Sleep mode:

• RCSTA and TXSTA Control registers must be configured for Synchronous Slave Reception (see **Section 27.5.2.4 "Synchronous Slave Reception Set-up:"**).
• If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
• The RCIF interrupt flag must be cleared by reading RCREG to unload any pending characters in the receive buffer.

Upon entering Sleep mode, the device will be ready to accept data and clocks on the RX/DT and TX/CK pins, respectively. When the data word has been completely clocked in by the external device, the RCIF interrupt flag bit of the PIR1 register will be set. Thereby, waking the processor from Sleep.

Upon waking from Sleep, the instruction following the SLEEP instruction will be executed. If the Global Interrupt Enable (GIE) bit of the INTCON register is also set, then the Interrupt Service Routine at address 004h will be called.

### 27.6.2 SYNCHRONOUS TRANSMIT DURING SLEEP

To transmit during Sleep, all the following conditions must be met before entering Sleep mode:

• RCSTA and TXSTA Control registers must be configured for Synchronous Slave Transmission (see **Section 27.5.2.2 "Synchronous Slave Transmission Set-up:"**).
• The TXIF interrupt flag must be cleared by writing the output data to the TXREG, thereby filling the TSR and transmit buffer.
• If interrupts are desired, set the TXIE bit of the PIE1 register and the PEIE bit of the INTCON register.
• Interrupt enable bits TXIE of the PIE1 register and PEIE of the INTCON register must set.

Upon entering Sleep mode, the device will be ready to accept clocks on TX/CK pin and transmit data on the RX/DT pin. When the data word in the TSR has been completely clocked out by the external device, the pending byte in the TXREG will transfer to the TSR and the TXIF flag will be set. Thereby, waking the processor from Sleep. At this point, the TXREG is available to accept another character for transmission, which will clear the TXIF flag.

Upon waking from Sleep, the instruction following the SLEEP instruction will be executed. If the Global Interrupt Enable (GIE) bit is also set then the Interrupt Service Routine at address 0004h will be called.

### 27.6.3 ALTERNATE PIN LOCATIONS

This module incorporates I/O pins that can be moved to other locations with the use of the alternate pin function register, APFCON. To determine which pins can be moved and what their default locations are upon a Reset, see **Section 13.1 "Alternate Pin Function"** for more information.

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 28.0 IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

ICSP™ programming allows customers to manufacture circuit boards with unprogrammed devices. Programming can be done after the assembly process allowing the device to be programmed with the most recent firmware or a custom firmware. Five pins are needed for ICSP™ programming:

• ICSPCLK
• ICSPDAT
• MCLR/VPP
• VDD
• VSS

In Program/Verify mode the program memory, user IDs and the Configuration Words are programmed through serial communications. The ICSPDAT pin is a bidirectional I/O used for transferring the serial data and the ICSPCLK pin is the clock input. For more information on ICSP™ refer to the **"PIC16(L)F178X Memory Programming Specification"** (DS41457).

### 28.1 High-Voltage Programming Entry Mode

The device is placed into High-Voltage Programming Entry mode by holding the ICSPCLK and ICSPDAT pins low then raising the voltage on MCLR/VPP to VIHH.

### 28.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using VDD only, without high voltage. When the LVP bit of Configuration Words is set to '1', the low-voltage ICSP programming entry is enabled. To disable the Low-Voltage ICSP mode, the LVP bit must be programmed to '0'.

Entry into the Low-Voltage Programming Entry mode requires the following steps:

1. MCLR is brought to VIL.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, MCLR must be held at VIL for as long as Program/Verify mode is to be maintained.

If low-voltage programming is enabled (LVP = 1), the MCLR Reset function is automatically enabled and cannot be disabled. See **Section 5.5 "MCLR"** for more information.

The LVP bit can only be reprogrammed to '0' by using the High-Voltage Programming mode.

### 28.3 Common Programming Interfaces

Connection to a target device is typically done through an ICSP™ header. A commonly found connector on development tools is the RJ-11 in the 6P6C (6 pin, 6 connector) configuration. See Figure 28-1.

**FIGURE 28-1: ICD RJ-11 STYLE CONNECTOR INTERFACE**
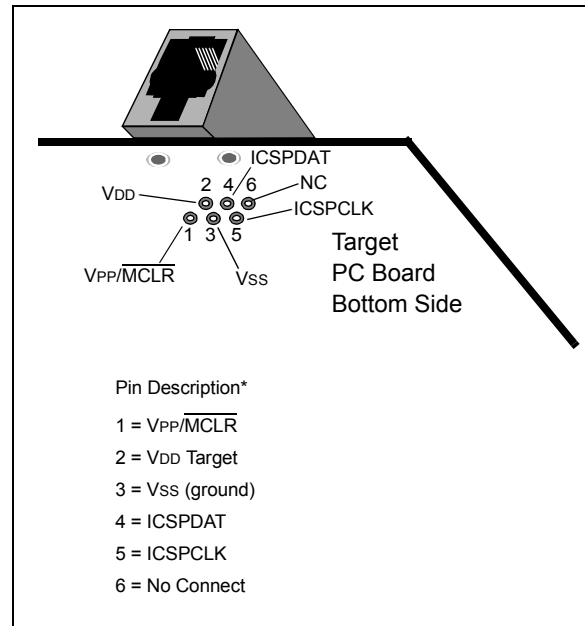


Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 28-2.

# PIC16(L)F1782/3

**FIGURE 28-2:** **PICkit™ PROGRAMMER STYLE CONNECTOR INTERFACE**



Pin 1 Indicator

Pin Description*

1 = VPP/$\overline{\text{MCLR}}$

2 = VDD Target

3 = VSS (ground)

4 = ICSPDAT

5 = ICSPCLK

6 = No Connect

\* The 6-pin header (0.100" spacing) accepts 0.025" square pins.
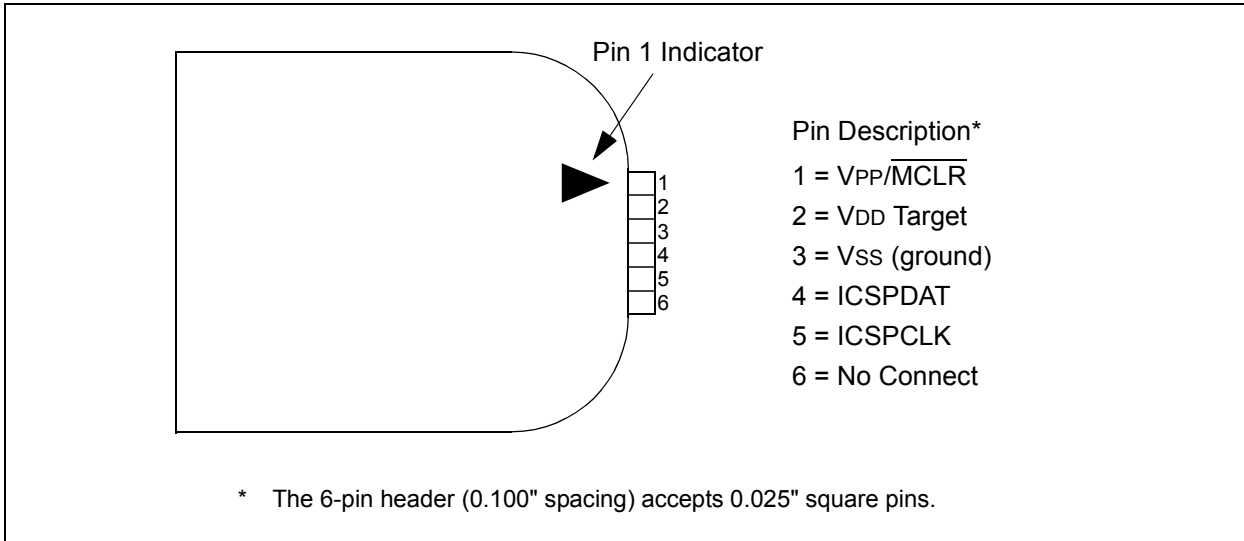
For additional interface recommendations, refer to your specific device programmer manual prior to PCB design.

It is recommended that isolation devices be used to separate the programming pins from other circuitry. The type of isolation is highly dependent on the specific application and may include devices such as resistors, diodes, or even jumpers. See Figure 28-3 for more information.

**FIGURE 28-3:** **TYPICAL CONNECTION FOR ICSP™ PROGRAMMING**



External Programming Signals

**VDD**

Device to be Programmed

VDD

VPP

$\overline{\text{MCLR}}$/VPP

VSS

VSS

Data

ICSPDAT

Clock

ICSPCLK

To Normal Connections

\* Isolation devices (as required).

**Preliminary**
© 2011-2012 Microchip Technology Inc.
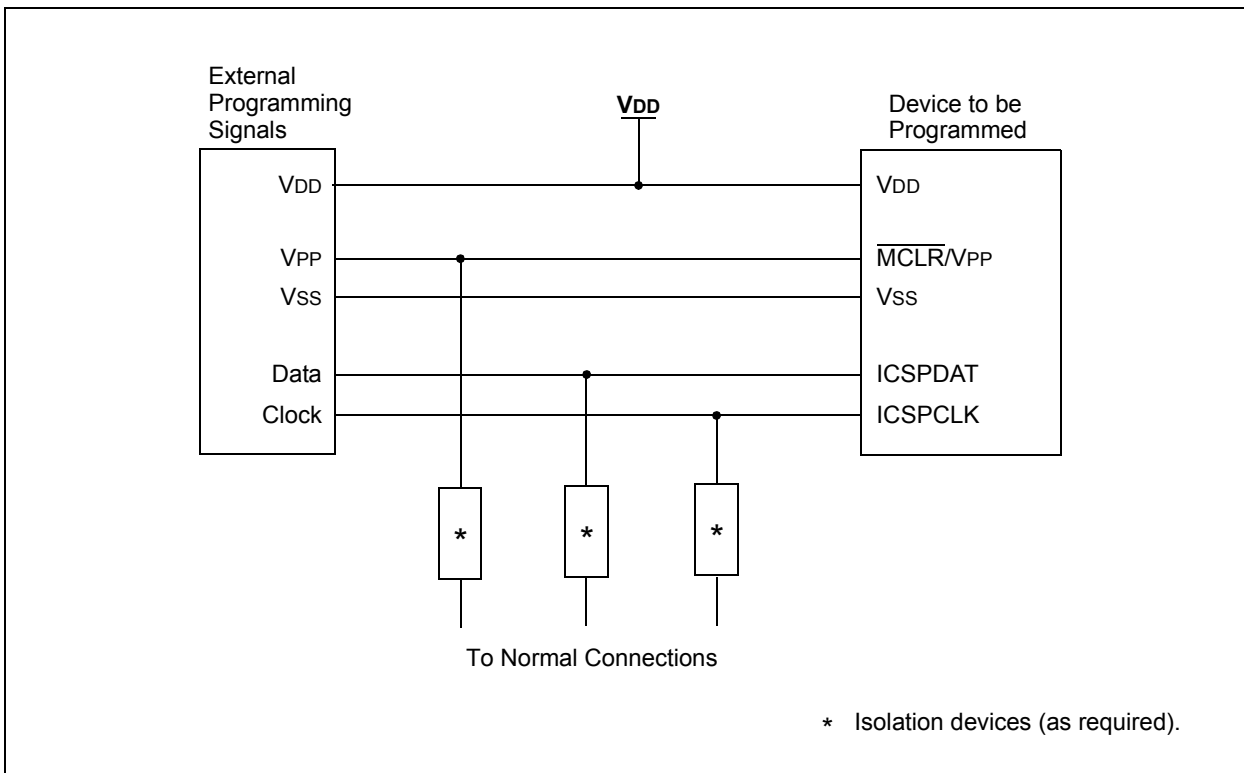
## 29.0 INSTRUCTION SET SUMMARY

Each instruction is a 14-bit word containing the operation code (opcode) and all required operands. The opcodes are broken into three broad categories.

- Byte Oriented
- Bit Oriented
- Literal and Control

The literal and control category contains the most varied instruction word format.

Table 29-3 lists the instructions recognized by the MPASM™ assembler.

All instructions are executed within a single instruction cycle, with the following exceptions, which may take two or three cycles:

- Subroutine takes two cycles (CALL, CALLW)
- Returns from interrupts or subroutines take two cycles (RETURN, RETLW, RETFIE)
- Program branching takes two cycles (GOTO, BRA, BRW, BTFSS, BTFSC, DECFSZ, INCSFZ)
- One additional instruction cycle will be used when any instruction references an indirect file register and the file select register is pointing to program memory.

One instruction cycle consists of 4 oscillator cycles; for an oscillator frequency of 4 MHz, this gives a nominal instruction execution rate of 1 MHz.

All instruction examples use the format '0xhh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

## 29.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the instruction, or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

### TABLE 29-1: OPCODE FIELD DESCRIPTIONS

| Field | Description |
|---|---|
| f | Register file address (0x00 to 0x7F) |
| W | Working register (accumulator) |
| b | Bit address within an 8-bit file register |
| k | Literal field, constant data or label |
| x | Don't care location (= 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. |
| d | Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1. |
| n | FSR or INDF number. (0-1) |
| mm | Pre-post increment-decrement mode selection |

### TABLE 29-2: ABBREVIATION DESCRIPTIONS

| Field | Description |
|---|---|
| PC | Program Counter |
| $\overline{TO}$ | Time-out bit |
| C | Carry bit |
| DC | Digit carry bit |
| Z | Zero bit |
| $\overline{PD}$ | Power-down bit |

# PIC16(L)F1782/3

**FIGURE 29-1:** **GENERAL FORMAT FOR INSTRUCTIONS**

**Byte-oriented file register operations**

```
13            8  7  6              0
┌─────────────┬────┬──────────────┐
│   OPCODE    │ d  │  f (FILE #)   │
└─────────────┴────┴──────────────┘
```

d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

**Bit-oriented file register operations**

```
13           10 9   7  6           0
┌─────────────┬──────┬─────────────┐
│   OPCODE    │b(BIT#)│ f (FILE #)  │
└─────────────┴──────┴─────────────┘
```

b = 3-bit bit address
f = 7-bit file register address

**Literal and control operations**

General

```
13            8  7                 0
┌─────────────┬───────────────────┐
│   OPCODE    │    k (literal)     │
└─────────────┴───────────────────┘
```

k = 8-bit immediate value

CALL and GOTO instructions only

```
13      11 10                      0
┌─────────┬─────────────────────────┐
│ OPCODE  │      k (literal)         │
└─────────┴─────────────────────────┘
```

k = 11-bit immediate value

MOVLP instruction only

```
13              7  6               0
┌───────────────┬──────────────────┐
│    OPCODE     │   k (literal)     │
└───────────────┴──────────────────┘
```

k = 7-bit immediate value

MOVLB instruction only

```
13                5  4             0
┌─────────────────┬────────────────┐
│     OPCODE      │   k (literal)   │
└─────────────────┴────────────────┘
```

k = 5-bit immediate value

BRA instruction only

```
13            9  8                 0
┌─────────────┬───────────────────┐
│   OPCODE    │    k (literal)     │
└─────────────┴───────────────────┘
```

k = 9-bit immediate value

FSR Offset instructions

```
13            7  6  5              0
┌─────────────┬────┬───────────────┐
│   OPCODE    │ n  │  k (literal)   │
└─────────────┴────┴───────────────┘
```

n = appropriate FSR
k = 6-bit immediate value

FSR Increment instructions

```
13                  3  2  1  0
┌───────────────────┬────┬────────┐
│      OPCODE       │ n  │m (mode)│
└───────────────────┴────┴────────┘
```

n = appropriate FSR
m = 2-bit mode value

OPCODE only

```
13                                0
┌──────────────────────────────────┐
│             OPCODE                │
└──────────────────────────────────┘
```

**Preliminary**

© 2011-2012 Microchip Technology Inc.

**TABLE 29-3: PIC16(L)F1782/3 INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 14-Bit Opcode MSb | | | LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| BYTE-ORIENTED FILE REGISTER OPERATIONS | | | | | | | | | |
| ADDWF | f, d | Add W and f | 1 | 00 | 0111 | dfff | ffff | C, DC, Z | 2 |
| ADDWFC | f, d | Add with Carry W and f | 1 | 11 | 1101 | dfff | ffff | C, DC, Z | 2 |
| ANDWF | f, d | AND W with f | 1 | 00 | 0101 | dfff | ffff | Z | 2 |
| ASRF | f, d | Arithmetic Right Shift | 1 | 11 | 0111 | dfff | ffff | C, Z | 2 |
| LSLF | f, d | Logical Left Shift | 1 | 11 | 0101 | dfff | ffff | C, Z | 2 |
| LSRF | f, d | Logical Right Shift | 1 | 11 | 0110 | dfff | ffff | C, Z | 2 |
| CLRF | f | Clear f | 1 | 00 | 0001 | lfff | ffff | Z | 2 |
| CLRW | – | Clear W | 1 | 00 | 0001 | 0000 | 00xx | Z | |
| COMF | f, d | Complement f | 1 | 00 | 1001 | dfff | ffff | Z | 2 |
| DECF | f, d | Decrement f | 1 | 00 | 0011 | dfff | ffff | Z | 2 |
| INCF | f, d | Increment f | 1 | 00 | 1010 | dfff | ffff | Z | 2 |
| IORWF | f, d | Inclusive OR W with f | 1 | 00 | 0100 | dfff | ffff | Z | 2 |
| MOVF | f, d | Move f | 1 | 00 | 1000 | dfff | ffff | Z | 2 |
| MOVWF | f | Move W to f | 1 | 00 | 0000 | 1fff | ffff | | 2 |
| RLF | f, d | Rotate Left f through Carry | 1 | 00 | 1101 | dfff | ffff | C | 2 |
| RRF | f, d | Rotate Right f through Carry | 1 | 00 | 1100 | dfff | ffff | C | 2 |
| SUBWF | f, d | Subtract W from f | 1 | 00 | 0010 | dfff | ffff | C, DC, Z | 2 |
| SUBWFB | f, d | Subtract with Borrow W from f | 1 | 11 | 1011 | dfff | ffff | C, DC, Z | 2 |
| SWAPF | f, d | Swap nibbles in f | 1 | 00 | 1110 | dfff | ffff | | 2 |
| XORWF | f, d | Exclusive OR W with f | 1 | 00 | 0110 | dfff | ffff | Z | 2 |
| BYTE ORIENTED SKIP OPERATIONS | | | | | | | | | |
| DECFSZ | f, d | Decrement f, Skip if 0 | 1(2) | 00 | 1011 | dfff | ffff | | 1, 2 |
| INCFSZ | f, d | Increment f, Skip if 0 | 1(2) | 00 | 1111 | dfff | ffff | | 1, 2 |
| BIT-ORIENTED FILE REGISTER OPERATIONS | | | | | | | | | |
| BCF | f, b | Bit Clear f | 1 | 01 | 00bb | bfff | ffff | | 2 |
| BSF | f, b | Bit Set f | 1 | 01 | 01bb | bfff | ffff | | 2 |
| BIT-ORIENTED SKIP OPERATIONS | | | | | | | | | |
| BTFSC | f, b | Bit Test f, Skip if Clear | 1 (2) | 01 | 10bb | bfff | ffff | | 1, 2 |
| BTFSS | f, b | Bit Test f, Skip if Set | 1 (2) | 01 | 11bb | bfff | ffff | | 1, 2 |
| LITERAL OPERATIONS | | | | | | | | | |
| ADDLW | k | Add literal and W | 1 | 11 | 1110 | kkkk | kkkk | C, DC, Z | |
| ANDLW | k | AND literal with W | 1 | 11 | 1001 | kkkk | kkkk | Z | |
| IORLW | k | Inclusive OR literal with W | 1 | 11 | 1000 | kkkk | kkkk | Z | |
| MOVLB | k | Move literal to BSR | 1 | 00 | 0000 | 001k | kkkk | | |
| MOVLP | k | Move literal to PCLATH | 1 | 11 | 0001 | 1kkk | kkkk | | |
| MOVLW | k | Move literal to W | 1 | 11 | 0000 | kkkk | kkkk | | |
| SUBLW | k | Subtract W from literal | 1 | 11 | 1100 | kkkk | kkkk | C, DC, Z | |
| XORLW | k | Exclusive OR literal with W | 1 | 11 | 1010 | kkkk | kkkk | Z | |

**Note 1:** If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

**2:** If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

**TABLE 29-3:** **PIC16(L)F1782/3 INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 14-Bit Opcode MSb | | | LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| **CONTROL OPERATIONS** | | | | | | | | | |
| BRA | k | Relative Branch | 2 | 11 | 001k | kkkk | kkkk | | |
| BRW | – | Relative Branch with W | 2 | 00 | 0000 | 0000 | 1011 | | |
| CALL | k | Call Subroutine | 2 | 10 | 0kkk | kkkk | kkkk | | |
| CALLW | – | Call Subroutine with W | 2 | 00 | 0000 | 0000 | 1010 | | |
| GOTO | k | Go to address | 2 | 10 | 1kkk | kkkk | kkkk | | |
| RETFIE | k | Return from interrupt | 2 | 00 | 0000 | 0000 | 1001 | | |
| RETLW | k | Return with literal in W | 2 | 11 | 0100 | kkkk | kkkk | | |
| RETURN | – | Return from Subroutine | 2 | 00 | 0000 | 0000 | 1000 | | |
| **INHERENT OPERATIONS** | | | | | | | | | |
| CLRWDT | – | Clear Watchdog Timer | 1 | 00 | 0000 | 0110 | 0100 | $\overline{TO}$, $\overline{PD}$ | |
| NOP | – | No Operation | 1 | 00 | 0000 | 0000 | 0000 | | |
| OPTION | – | Load OPTION_REG register with W | 1 | 00 | 0000 | 0110 | 0010 | | |
| RESET | – | Software device Reset | 1 | 00 | 0000 | 0000 | 0001 | | |
| SLEEP | – | Go into Standby mode | 1 | 00 | 0000 | 0110 | 0011 | $\overline{TO}$, $\overline{PD}$ | |
| TRIS | f | Load TRIS register with W | 1 | 00 | 0000 | 0110 | 0fff | | |
| **C-COMPILER OPTIMIZED** | | | | | | | | | |
| ADDFSR | n, k | Add Literal k to FSRn | 1 | 11 | 0001 | 0nkk | kkkk | | |
| MOVIW | n mm | Move Indirect FSRn to W with pre/post inc/dec modifier, mm | 1 | 00 | 0000 | 0001 | 0nmm | Z | **2, 3** |
| | k[n] | Move INDFn to W, Indexed Indirect. | 1 | 11 | 1111 | 0nkk | kkkk | Z | **2** |
| MOVWI | n mm | Move W to Indirect FSRn with pre/post inc/dec modifier, mm | 1 | 00 | 0000 | 0001 | 1nmm | | **2, 3** |
| | k[n] | Move W to INDFn, Indexed Indirect. | 1 | 11 | 1111 | 1nkk | kkkk | | **2** |

**Note 1:** If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

    **2:** If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

    **3:** See Table in the MOVIW and MOVWI instruction descriptions.

## 29.2    Instruction Descriptions

| ADDFSR | Add Literal to FSRn |
|---|---|
| Syntax: | [ *label* ] ADDFSR   FSRn, k |
| Operands: | $-32 \le k \le 31$<br>$n \in [\,0, 1\,]$ |
| Operation: | FSR(n) + k → FSR(n) |
| Status Affected: | None |
| Description: | The signed 6-bit literal 'k' is added to the contents of the FSRnH:FSRnL register pair.<br><br>FSRn is limited to the range 0000h - FFFFh. Moving beyond these bounds will cause the FSR to wrap-around. |

| ADDLW | Add literal and W |
|---|---|
| Syntax: | [ *label* ]  ADDLW    k |
| Operands: | $0 \le k \le 255$ |
| Operation: | (W) + k → (W) |
| Status Affected: | C, DC, Z |
| Description: | The contents of the W register are added to the eight-bit literal 'k' and the result is placed in the W register. |

| ADDWF | Add W and f |
|---|---|
| Syntax: | [ *label* ]  ADDWF    f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (W) + (f) → (destination) |
| Status Affected: | C, DC, Z |
| Description: | Add the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'. |

| ADDWFC | ADD W and CARRY bit to f |
|---|---|
| Syntax: | [ *label* ] ADDWFC      f {,d} |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (W) + (f) + (C) → dest |
| Status Affected: | C, DC, Z |
| 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'. |

| ANDLW | AND literal with W |
|---|---|
| Syntax: | [ *label* ]  ANDLW    k |
| Operands: | $0 \le k \le 255$ |
| Operation: | (W) .AND. (k) → (W) |
| Status Affected: | Z |
| Description: | The contents of W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register. |

| ANDWF | AND W with f |
|---|---|
| Syntax: | [ *label* ]  ANDWF    f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (W) .AND. (f) → (destination) |
| Status Affected: | Z |
| Description: | AND the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'. |

| ASRF | Arithmetic Right Shift |
|---|---|
| Syntax: | [ *label* ] ASRF    f {,d} |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (f<7>)→ dest<7><br>(f<7:1>) → dest<6:0>,<br>(f<0>) → C, |
| Status Affected: | C, Z |
| Description: | The contents of register 'f' are shifted one bit to the right through the Carry flag. The MSb remains unchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'. |

| **BCF** | **Bit Clear f** |
|---|---|
| Syntax: | [ *label* ] BCF    f,b |
| Operands: | $0 \leq f \leq 127$<br>$0 \leq b \leq 7$ |
| Operation: | $0 \rightarrow$ (f\<b\>) |
| Status Affected: | None |
| Description: | Bit 'b' in register 'f' is cleared. |

| **BRA** | **Relative Branch** |
|---|---|
| Syntax: | [ *label* ] BRA   label<br>[ *label* ] BRA   $+k |
| Operands: | $-256 \leq$ label - PC + 1 $\leq 255$<br>$-256 \leq k \leq 255$ |
| Operation: | $(PC) + 1 + k \rightarrow PC$ |
| Status Affected: | None |
| Description: | Add the signed 9-bit literal 'k' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + k. This instruction is a two-cycle instruction. This branch has a limited range. |

| **BRW** | **Relative Branch with W** |
|---|---|
| Syntax: | [ *label* ] BRW |
| Operands: | None |
| Operation: | $(PC) + (W) \rightarrow PC$ |
| Status Affected: | None |
| Description: | Add the contents of W (unsigned) to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + (W). This instruction is a two-cycle instruction. |

| **BSF** | **Bit Set f** |
|---|---|
| Syntax: | [ *label* ] BSF    f,b |
| Operands: | $0 \leq f \leq 127$<br>$0 \leq b \leq 7$ |
| Operation: | $1 \rightarrow$ (f\<b\>) |
| Status Affected: | None |
| Description: | Bit 'b' in register 'f' is set. |

| **BTFSC** | **Bit Test f, Skip if Clear** |
|---|---|
| Syntax: | [ *label* ] BTFSC   f,b |
| Operands: | $0 \leq f \leq 127$<br>$0 \leq b \leq 7$ |
| Operation: | skip if (f\<b\>) = 0 |
| Status Affected: | None |
| Description: | If bit 'b' in register 'f' is '1', the next instruction is executed.<br>If bit 'b', in register 'f', is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2-cycle instruction. |

| **BTFSS** | **Bit Test f, Skip if Set** |
|---|---|
| Syntax: | [ *label* ] BTFSS   f,b |
| Operands: | $0 \leq f \leq 127$<br>$0 \leq b < 7$ |
| Operation: | skip if (f\<b\>) = 1 |
| Status Affected: | None |
| Description: | If bit 'b' in register 'f' is '0', the next instruction is executed.<br>If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction. |

**Preliminary**

| CALL | Call Subroutine |
|---|---|
| Syntax: | [ *label* ]  CALL  k |
| Operands: | $0 \leq k \leq 2047$ |
| Operation: | (PC)+ 1→ TOS,<br>k → PC<10:0>,<br>(PCLATH<6:3>) → PC<14:11> |
| Status Affected: | None |
| Description: | Call Subroutine. First, return address (PC + 1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction. |

| CALLW | Subroutine Call With W |
|---|---|
| Syntax: | [ *label* ] CALLW |
| Operands: | None |
| Operation: | (PC) +1 → TOS,<br>(W) → PC<7:0>,<br>(PCLATH<6:0>) → PC<14:8> |
| Status Affected: | None |
| Description: | Subroutine call with W.  First, the return address (PC + 1) is pushed onto the return stack. Then, the contents of W is loaded into PC<7:0>, and the contents of PCLATH into PC<14:8>. CALLW is a two-cycle instruction. |

| CLRF | Clear f |
|---|---|
| Syntax: | [ *label* ]  CLRF   f |
| Operands: | $0 \leq f \leq 127$ |
| Operation: | 00h → (f)<br>1 → Z |
| Status Affected: | Z |
| Description: | The contents of register 'f' are cleared and the Z bit is set. |

| CLRW | Clear W |
|---|---|
| Syntax: | [ *label* ]  CLRW |
| Operands: | None |
| Operation: | 00h → (W)<br>1 → Z |
| Status Affected: | Z |
| Description: | W register is cleared. Zero bit (Z) is set. |

| CLRWDT | Clear Watchdog Timer |
|---|---|
| Syntax: | [ *label* ]  CLRWDT |
| Operands: | None |
| Operation: | 00h → WDT<br>0 → WDT prescaler,<br>$1 \rightarrow \overline{TO}$<br>$1 \rightarrow \overline{PD}$ |
| Status Affected: | $\overline{TO}$, $\overline{PD}$ |
| Description: | CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT.<br>Status bits $\overline{TO}$ and $\overline{PD}$ are set. |

| COMF | Complement f |
|---|---|
| Syntax: | [ *label* ]  COMF   f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | $\overline{(f)} \rightarrow$ (destination) |
| Status Affected: | Z |
| Description: | The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. |

| DECF | Decrement f |
|---|---|
| Syntax: | [ *label* ]  DECF f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (f) - 1 → (destination) |
| Status Affected: | Z |
| Description: | Decrement register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'. |

| DECFSZ | Decrement f, Skip if 0 |
|---|---|
| Syntax: | [ *label* ]   DECFSZ   f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (f) - 1 $\rightarrow$ (destination);<br>skip if result = 0 |
| Status Affected: | None |
| Description: | The contents of register 'f' are decremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.<br>If the result is '1', the next instruction is executed. If the result is '0', then a NOP is executed instead, making it a 2-cycle instruction. |

| INCFSZ | Increment f, Skip if 0 |
|---|---|
| Syntax: | [ *label* ]   INCFSZ   f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (f) + 1 $\rightarrow$ (destination),<br>skip if result = 0 |
| Status Affected: | None |
| Description: | The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.<br>If the result is '1', the next instruction is executed. If the result is '0', a NOP is executed instead, making it a  2-cycle instruction. |

| GOTO | Unconditional Branch |
|---|---|
| Syntax: | [ *label* ]   GOTO   k |
| Operands: | $0 \leq k \leq 2047$ |
| Operation: | k $\rightarrow$ PC<10:0><br>PCLATH<6:3> $\rightarrow$ PC<14:11> |
| Status Affected: | None |
| Description: | GOTO is an unconditional branch. The eleven-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two-cycle instruction. |

| IORLW | Inclusive OR literal with W |
|---|---|
| Syntax: | [ *label* ]   IORLW   k |
| Operands: | $0 \leq k \leq 255$ |
| Operation: | (W) .OR. k $\rightarrow$ (W) |
| Status Affected: | Z |
| Description: | The contents of the W register are OR'ed with the eight-bit literal 'k'. The result is placed in the W register. |

| INCF | Increment f |
|---|---|
| Syntax: | [ *label* ]   INCF   f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (f) + 1 $\rightarrow$ (destination) |
| Status Affected: | Z |
| Description: | The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'. |

| IORWF | Inclusive OR W with f |
|---|---|
| Syntax: | [ *label* ]   IORWF   f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (W) .OR. (f) $\rightarrow$ (destination) |
| Status Affected: | Z |
| Description: | Inclusive OR the W register with register 'f'. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'. |

| LSLF | Logical Left Shift |
|------|--------------------|
| Syntax: | [ *label* ] LSLF    f {,d} |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (f<7>) → C<br>(f<6:0>) → dest<7:1><br>0 → dest<0> |
| Status Affected: | C, Z |
| Description: | The contents of register 'f' are shifted one bit to the left through the Carry flag. A '0' is shifted into the LSb. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'. |
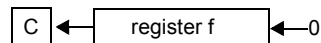
C ← register f ←0

| LSRF | Logical Right Shift |
|------|---------------------|
| Syntax: | [ *label* ] LSRF    f {,d} |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | 0 → dest<7><br>(f<7:1>) → dest<6:0>,<br>(f<0>) → C, |
| Status Affected: | C, Z |
| Description: | The contents of register 'f' are shifted one bit to the right through the Carry flag. A '0' is shifted into the MSb. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'. |
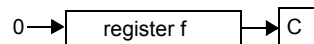
0 → register f → C

| MOVF | Move f |
|------|--------|
| Syntax: | [ *label* ]    MOVF   f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (f) → (dest) |
| Status Affected: | Z |
| Description: | The contents of register f is moved to a destination dependent upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected. |
| Words: | 1 |
| Cycles: | 1 |
| Example: | MOVF    FSR,  0 |

After Instruction
    W  =  value in FSR register
    Z  =  1

**Preliminary**

# PIC16(L)F1782/3

**MOVIW**  Move INDFn to W

| | |
|---|---|
| Syntax: | [ *label* ] MOVIW ++FSRn<br>[ *label* ] MOVIW --FSRn<br>[ *label* ] MOVIW FSRn++<br>[ *label* ] MOVIW FSRn--<br>[ *label* ] MOVIW k[FSRn] |
| Operands: | $n \in [0,1]$<br>$mm \in [00,01,10,11]$<br>$-32 \le k \le 31$ |
| Operation: | INDFn $\rightarrow$ W<br>Effective address is determined by<br>• FSR + 1 (preincrement)<br>• FSR - 1 (predecrement)<br>• FSR + k (relative offset)<br>After the Move, the FSR value will be either:<br>• FSR + 1 (all increments)<br>• FSR - 1 (all decrements)<br>• Unchanged |
| Status Affected: | Z |

| Mode | Syntax | mm |
|---|---|---|
| Preincrement | ++FSRn | 00 |
| Predecrement | --FSRn | 01 |
| Postincrement | FSRn++ | 10 |
| Postdecrement | FSRn-- | 11 |

| | |
|---|---|
| Description: | This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it.<br><br>Note: The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn.<br><br>FSRn is limited to the range 0000h - FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wrap-around. |

**MOVLB**  Move literal to BSR

| | |
|---|---|
| Syntax: | [ *label* ] MOVLB  k |
| Operands: | $0 \le k \le 15$ |
| Operation: | k $\rightarrow$ BSR |
| Status Affected: | None |
| Description: | The five-bit literal 'k' is loaded into the Bank Select Register (BSR). |

**MOVLP**  Move literal to PCLATH

| | |
|---|---|
| Syntax: | [ *label* ] MOVLP  k |
| Operands: | $0 \le k \le 127$ |
| Operation: | k $\rightarrow$ PCLATH |
| Status Affected: | None |
| Description: | The seven-bit literal 'k' is loaded into the PCLATH register. |

**MOVLW**  Move literal to W

| | |
|---|---|
| Syntax: | [ *label* ]   MOVLW  k |
| Operands: | $0 \le k \le 255$ |
| Operation: | k $\rightarrow$ (W) |
| Status Affected: | None |
| Description: | The eight-bit literal 'k' is loaded into W register. The "don't cares" will assemble as '0's. |
| Words: | 1 |
| Cycles: | 1 |
| Example: | MOVLW   0x5A |

After Instruction
W   =   0x5A

**MOVWF**  Move W to f

| | |
|---|---|
| Syntax: | [ *label* ]   MOVWF   f |
| Operands: | $0 \le f \le 127$ |
| Operation: | (W) $\rightarrow$ (f) |
| Status Affected: | None |
| Description: | Move data from W register to register 'f'. |
| Words: | 1 |
| Cycles: | 1 |
| Example: | MOVWF   OPTION_REG |

Before Instruction
OPTION_REG = 0xFF
W = 0x4F
After Instruction
OPTION_REG = 0x4F
W = 0x4F

**Preliminary** © 2011-2012 Microchip Technology Inc.

| MOVWI | Move W to INDFn |
|---|---|

| | |
|---|---|
| Syntax: | [ *label* ] MOVWI ++FSRn<br>[ *label* ] MOVWI --FSRn<br>[ *label* ] MOVWI FSRn++<br>[ *label* ] MOVWI FSRn--<br>[ *label* ] MOVWI k[FSRn] |
| Operands: | $n \in [0,1]$<br>$mm \in [00,01, 10, 11]$<br>$-32 \leq k \leq 31$ |
| Operation: | $W \rightarrow INDFn$<br>Effective address is determined by<br>• FSR + 1 (preincrement)<br>• FSR - 1 (predecrement)<br>• FSR + k (relative offset)<br>After the Move, the FSR value will be either:<br>• FSR + 1 (all increments)<br>• FSR - 1 (all decrements)<br>Unchanged |
| Status Affected: | None |

| Mode | Syntax | mm |
|---|---|---|
| Preincrement | ++FSRn | 00 |
| Predecrement | --FSRn | 01 |
| Postincrement | FSRn++ | 10 |
| Postdecrement | FSRn-- | 11 |

| | |
|---|---|
| Description: | This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it.<br><br>Note: The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn.<br><br>FSRn is limited to the range 0000h - FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wrap-around.<br><br>The increment/decrement operation on FSRn WILL NOT affect any Status bits. |

| NOP | No Operation |
|---|---|

| | |
|---|---|
| Syntax: | [ *label* ]   NOP |
| Operands: | None |
| Operation: | No operation |
| Status Affected: | None |
| Description: | No operation. |
| Words: | 1 |
| Cycles: | 1 |
| Example: | NOP |

| OPTION | Load OPTION_REG Register with W |
|---|---|

| | |
|---|---|
| Syntax: | [ *label* ]   OPTION |
| Operands: | None |
| Operation: | (W) $\rightarrow$ OPTION_REG |
| Status Affected: | None |
| Description: | Move data from W register to OPTION_REG register. |
| Words: | 1 |
| Cycles: | 1 |
| Example: | OPTION |

Before Instruction
OPTION_REG = 0xFF
W = 0x4F
After Instruction
OPTION_REG = 0x4F
W = 0x4F

| RESET | Software Reset |
|---|---|

| | |
|---|---|
| Syntax: | [ *label* ]   RESET |
| Operands: | None |
| Operation: | Execute a device Reset. Resets the $\overline{RI}$ flag of the PCON register. |
| Status Affected: | None |
| Description: | This instruction provides a way to execute a hardware Reset by software. |

| **RETFIE** | **Return from Interrupt** |
|---|---|
| Syntax: | [ *label* ]   RETFIE |
| Operands: | None |
| Operation: | TOS → PC,<br>$1 \rightarrow$ GIE |
| Status Affected: | None |
| Description: | Return from Interrupt. Stack is POPed and Top-of-Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two-cycle instruction. |
| Words: | 1 |
| Cycles: | 2 |
| Example: | RETFIE |

```
After Interrupt
        PC  =   TOS
        GIE =   1
```

| **RETURN** | **Return from Subroutine** |
|---|---|
| Syntax: | [ *label* ]   RETURN |
| Operands: | None |
| Operation: | TOS → PC |
| Status Affected: | None |
| Description: | Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction. |

| **RETLW** | **Return with literal in W** |
|---|---|
| Syntax: | [ *label* ]   RETLW  k |
| Operands: | $0 \le k \le 255$ |
| Operation: | k → (W);<br>TOS → PC |
| Status Affected: | None |
| Description: | The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction. |
| Words: | 1 |
| Cycles: | 2 |
| Example: | |

```
        CALL TABLE;W contains table
           ;offset value
   •       ;W now has table value
TABLE   •
        •
        ADDWF PC ;W = offset
        RETLW k1 ;Begin table
        RETLW k2 ;
        •
        •
        •
        RETLW kn ; End of table
```

```
Before Instruction
        W   =   0x07
After Instruction
        W   =   value of k8
```

| **RLF** | **Rotate Left f through Carry** |
|---|---|
| Syntax: | [ *label* ]    RLF   f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | See description below |
| Status Affected: | C |
| 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 the W register. If 'd' is '1', the result is stored back in register 'f'. |



| | |
|---|---|
| Words: | 1 |
| Cycles: | 1 |
| Example: | RLF   REG1,0 |
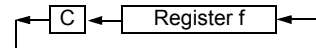
```
Before Instruction
        REG1    =   1110 0110
        C       =   0
After Instruction
        REG1    =   1110 0110
        W       =   1100 1100
        C       =   1
```

| RRF | Rotate Right f through Carry |
|---|---|
| Syntax: | [ *label* ]  RRF  f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | See description below |
| Status Affected: | C |
| Description: | The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'. |



| SLEEP | Enter Sleep mode |
|---|---|
| Syntax: | [ *label* ]  SLEEP |
| Operands: | None |
| Operation: | 00h $\rightarrow$ WDT,<br>0 $\rightarrow$ WDT prescaler,<br>1 $\rightarrow$ $\overline{TO}$,<br>0 $\rightarrow$ $\overline{PD}$ |
| Status Affected: | $\overline{TO}$, $\overline{PD}$ |
| Description: | The power-down Status bit, $\overline{PD}$ is cleared. Time-out Status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared.<br>The processor is put into Sleep mode with the oscillator stopped. |

| SUBLW | Subtract W from literal |
|---|---|
| Syntax: | [ *label* ]  SUBLW  k |
| Operands: | $0 \leq k \leq 255$ |
| Operation: | k - (W) $\rightarrow$ (W) |
| Status Affected: | C, DC, Z |
| Description: | The W register is subtracted (2's complement method) from the eight-bit literal 'k'. The result is placed in the W register. |

| C = 0 | W > k |
|---|---|
| C = 1 | W $\leq$ k |
| DC = 0 | W<3:0> > k<3:0> |
| DC = 1 | W<3:0> $\leq$ k<3:0> |

| SUBWF | Subtract W from f |
|---|---|
| Syntax: | [ *label* ]  SUBWF  f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (f) - (W) $\rightarrow$ (destination) |
| Status Affected: | C, DC, Z |
| Description: | Subtract (2's complement method) W register from register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f. |

| C = 0 | W > f |
|---|---|
| C = 1 | W $\leq$ f |
| DC = 0 | W<3:0> > f<3:0> |
| DC = 1 | W<3:0> $\leq$ f<3:0> |

| SUBWFB | Subtract W from f with Borrow |
|---|---|
| Syntax: | SUBWFB  f {,d} |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (f) – (W) – ($\overline{B}$) $\rightarrow$ dest |
| Status Affected: | C, DC, Z |
| Description: | Subtract W and the BORROW flag (CARRY) 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'. |

| SWAPF | Swap Nibbles in f |
|---|---|
| Syntax: | [ *label* ]　SWAPF f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (f<3:0>) $\rightarrow$ (destination<7:4>),<br>(f<7:4>) $\rightarrow$ (destination<3:0>) |
| Status Affected: | None |
| Description: | The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed in register 'f'. |

| TRIS | Load TRIS Register with W |
|---|---|
| Syntax: | [ *label* ]　TRIS f |
| Operands: | $5 \leq f \leq 7$ |
| Operation: | (W) $\rightarrow$ TRIS register 'f' |
| Status Affected: | None |
| Description: | Move data from W register to TRIS register.<br>When 'f' = 5, TRISA is loaded.<br>When 'f' = 6, TRISB is loaded.<br>When 'f' = 7, TRISC is loaded. |

| XORLW | Exclusive OR literal with W |
|---|---|
| Syntax: | [ *label* ]　XORLW　k |
| Operands: | $0 \leq k \leq 255$ |
| Operation: | (W) .XOR. k $\rightarrow$ (W) |
| Status Affected: | Z |
| Description: | The contents of the W register are XOR'ed with the eight-bit literal 'k'. The result is placed in the W register. |

| XORWF | Exclusive OR W with f |
|---|---|
| Syntax: | [ *label* ]　XORWF　f,d |
| Operands: | $0 \leq f \leq 127$<br>$d \in [0,1]$ |
| Operation: | (W) .XOR. (f) $\rightarrow$ (destination) |
| Status Affected: | Z |
| Description: | Exclusive OR the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'. |

**Preliminary**

## 30.0 ELECTRICAL SPECIFICATIONS

### Absolute Maximum Ratings[†]

Ambient temperature under bias.................................................................................................... -40°C to +125°C

Storage temperature ..................................................................................................... -65°C to +150°C

Voltage on VDD with respect to VSS, PIC16F1782/3 ........................................................ -0.3V to +6.5V

Voltage on VCAP pin with respect to VSS, PIC16F1782/3 ................................................ -0.3V to +4.0V

Voltage on VDD with respect to VSS, PIC16LF1782/3 ....................................................... -0.3V to +4.0V

Voltage on MCLR with respect to Vss ................................................................................ -0.3V to +9.0V

Voltage on all other pins with respect to VSS ...................................................... -0.3V to (VDD + 0.3V)

Total power dissipation[(1)] ............................................................................................................ 800 mW

Maximum current out of VSS pin, -40°C ≤ TA ≤ +85°C for industrial............................................. 170 mA

Maximum current out of VSS pin, -40°C ≤ TA ≤ +125°C for extended ......................................... 70 mA

Maximum current into VDD pin, -40°C ≤ TA ≤ +85°C for industrial.............................................. 85 mA

Maximum current into VDD pin, -40°C ≤ TA ≤ +125°C for extended ............................................ 35 mA

Clamp current, IK (VPIN < 0 or VPIN > VDD)............................................................................... ± 20 mA

Maximum output current sunk by any I/O pin................................................................................ 25 mA

Maximum output current sourced by any I/O pin .......................................................................... 25 mA

**Note 1:** Power dissipation is calculated as follows: PDIS = VDD x {IDD – ∑IOH} + ∑ {(VDD – VOH) x IOH} + ∑(VOl x IOL).

---

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure above maximum rating conditions for extended periods may affect device reliability.

---

# PIC16(L)F1782/3

**FIGURE 30-1:** **PIC16F1782/3 VOLTAGE FREQUENCY GRAPH, -40°C ≤ T<sub>A</sub> ≤ +125°C**



**Note 1:** The shaded region indicates the permissible combinations of voltage and frequency.
**2:** Refer to Table 30-1 for each Oscillator mode's supported frequencies.
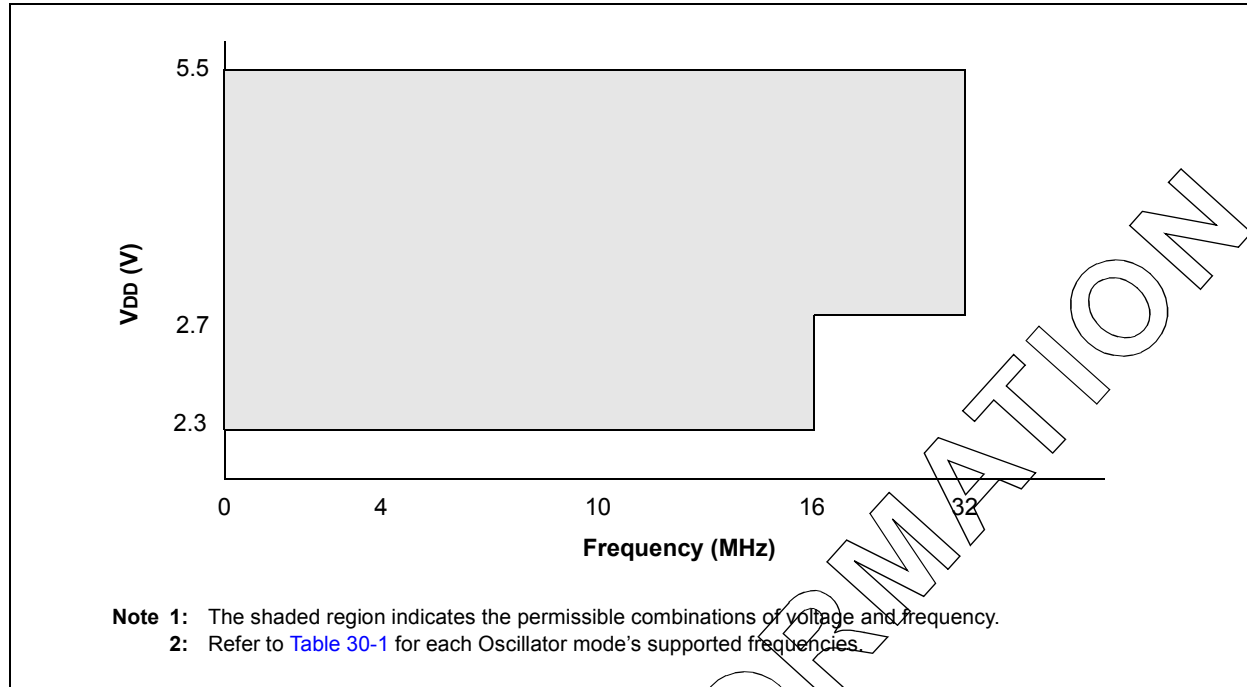
**FIGURE 30-2:** **PIC16LF1782/3 VOLTAGE FREQUENCY GRAPH, -40°C ≤ T<sub>A</sub> ≤ +125°C**



**Note 1:** The shaded region indicates the permissible combinations of voltage and frequency.
**2:** Refer to Table 30-1 for each Oscillator mode's supported frequencies.

**Preliminary** © 2011-2012 Microchip Technology Inc.
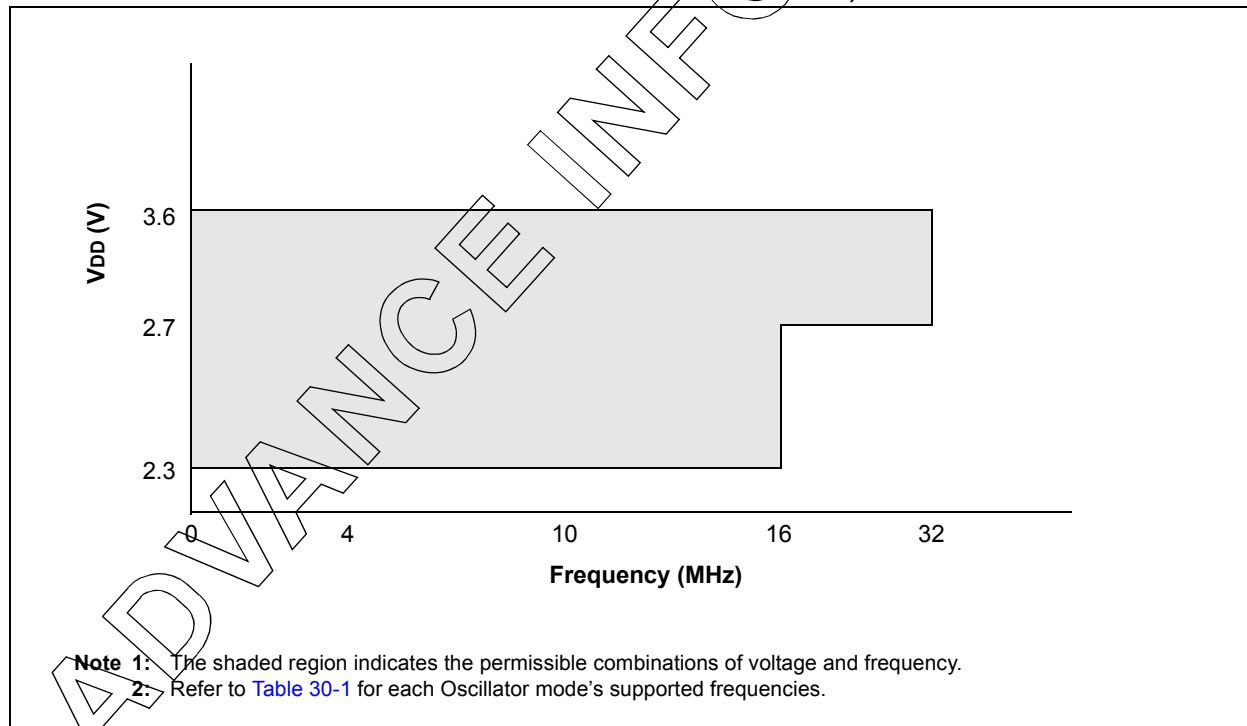
**FIGURE 30-3:** HFINTOSC FREQUENCY ACCURACY OVER DEVICE V$_{DD}$ AND TEMPERATURE

## 30.1 DC Characteristics: PIC16(L)F1782/3-I/E (Industrial, Extended)

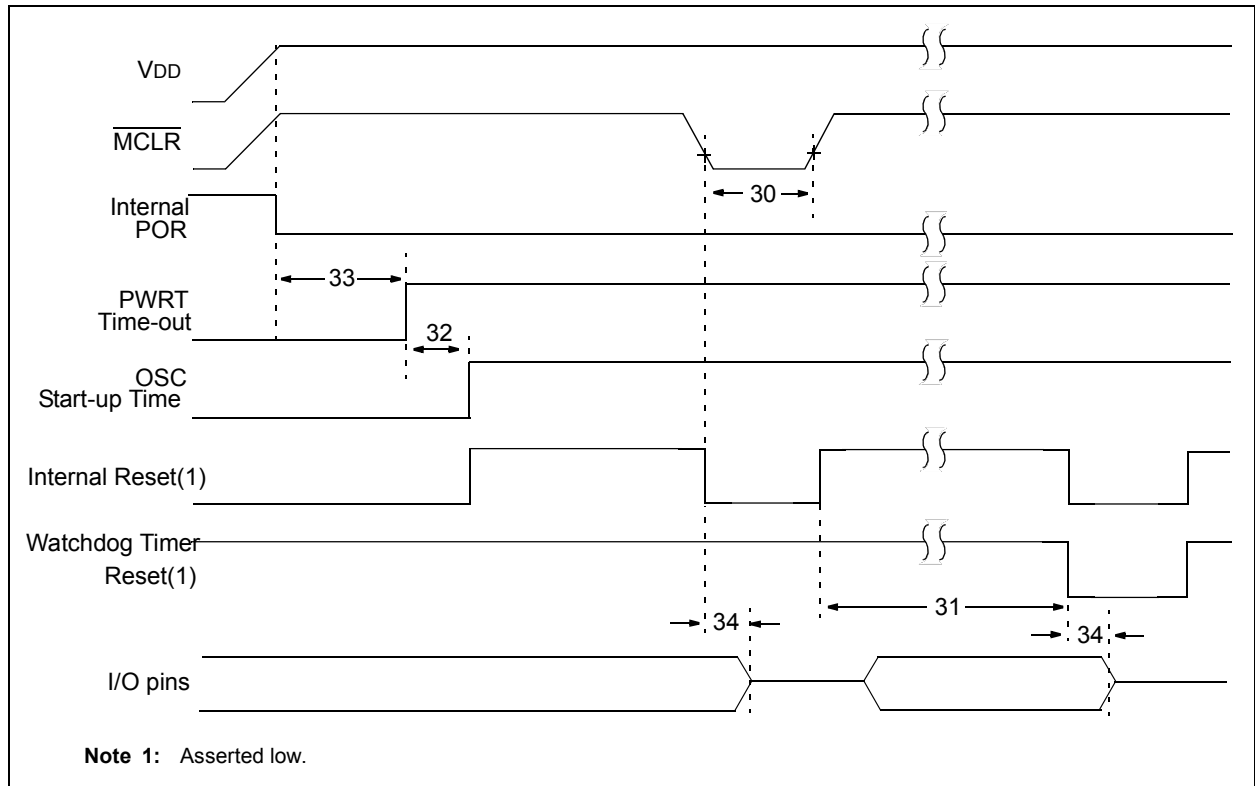| PIC16LF1782/3 | | | | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature  -40°C ≤ TA ≤ +85°C for industrial<br>  -40°C ≤ TA ≤ +125°C for extended | |
|---|---|---|---|---|---|---|---|
| **PIC16F1782/3** | | | | | | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature  -40°C ≤ TA ≤ +85°C for industrial<br>  -40°C ≤ TA ≤ +125°C for extended | |
| Param. No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| D001 | VDD | **Supply Voltage (VDDMIN, VDDMAX)** | | | | | |
| | | PIC16LF1782/3 | 1.8<br>2.7 | —<br>— | 3.6<br>3.6 | V<br>V | FOSC ≤ 16 MHz:<br>FOSC ≤ 32 MHz (**Note 2**) |
| D001 | | PIC16F1782/3 | 2.3<br>2.7 | —<br>— | 5.5<br>5.5 | V<br>V | FOSC ≤ 16 MHz:<br>FOSC ≤ 32 MHz (**Note 2**) |
| D002* | VDR | **RAM Data Retention Voltage[1]** | | | | | |
| | | PIC16LF1782/3 | 1.5 | — | — | V | Device in Sleep mode |
| D002* | | PIC16F1782/3 | 1.7 | — | — | V | Device in Sleep mode |
| | VPOR* | **Power-on Reset Release Voltage** | — | 1.6 | — | V | |
| | VPORR* | **Power-on Reset Rearm Voltage** | | | | | |
| | | PIC16LF1782/3 | — | 0.8 | — | V | Device in Sleep mode |
| | | PIC16F1782/3 | — | 1.5 | — | V | Device in Sleep mode |
| D003 | VFVR | **Fixed Voltage Reference Voltage** | -3 | — | 3 | % | 1.024V, VDD ≥ 2.5V<br>2.048V, VDD ≥ 2.5V<br>4.096V, VDD ≥ 4.75V |
| D004* | SVDD | **VDD Rise Rate** to ensure internal Power-on Reset signal | 0.05 | — | — | V/ms | See **Section 5.1 "Power-On Reset (POR)"** for details. |

\*  These parameters are characterized but not tested.

†  Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.

**2:** PLL required for 32 MHz operation.

**3:** For proper operation, the minimum value of the ADC positive voltage reference must be 1.8V or greater. When selecting the FVR or the VREF+ pin as the source of the ADC positive voltage reference, be aware that the voltage must be 1.8V or greater.

**Preliminary**

**FIGURE 30-4:** **POR AND POR REARM WITH SLOW RISING V**DD



**Note 1:** When NPOR is low, the device is held in Reset.
**2:** TPOR 1 μs typical.
**3:** TVLOW 2.7 μs typical.

## 30.2 DC Characteristics: PIC16(L)F1782/3-I/E (Industrial, Extended)

| PIC16LF1782/3 | Standard Operating Conditions (unless otherwise stated) Operating temperature   -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended |
|---|---|
| PIC16F1782/3 | Standard Operating Conditions (unless otherwise stated) Operating temperature   -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended |

| Param No. | Device Characteristics | Min. | Typ† | Max. | Units | Conditions | |
|---|---|---|---|---|---|---|---|
| | | | | | | $V_{DD}$ | Note |
| **Supply Current ($I_{DD}$)**[1, 2] | | | | | | | |
| D009 | LDO Regulator | — | 75 | — | μA | — | High Power mode, normal operation |
| | | — | 15 | — | μA | — | Sleep VREGCON<1> = 0 |
| | | — | 0.3 | — | μA | — | Sleep VREGCON<1> = 1 |
| D010 | | — | 8 | 16 | μA | 1.8 | $F_{OSC}$ = 32 kHz LP Oscillator mode **(Note 4)**, -40°C ≤ TA ≤ +85°C |
| | | — | 12 | 20 | μA | 3.0 | |
| D010 | | — | 18 | 63 | μA | 2.3 | $F_{OSC}$ = 32 kHz LP Oscillator mode **(Note 4, 5)**, -40°C ≤ TA ≤ +85°C |
| | | — | 20 | 74 | μA | 3.0 | |
| | | — | 22 | 75 | μA | 5.0 | |
| D012 | | — | 160 | 650 | μA | 1.8 | $F_{OSC}$ = 4 MHz XT Oscillator mode |
| | | — | 320 | 1000 | μA | 3.0 | |
| D012 | | — | 260 | 700 | μA | 2.3 | $F_{OSC}$ = 4 MHz XT Oscillator mode **(Note 5)** |
| | | — | 330 | 1100 | μA | 3.0 | |
| | | — | 380 | 1200 | μA | 5.0 | |

**Note 1:** The test conditions for all $I_{DD}$ measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to $V_{DD}$; $\overline{MCLR}$ = $V_{DD}$; WDT disabled.

**2:** The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

**3:** For RC oscillator configurations, current through $R_{EXT}$ is not included. The current through the resistor can be extended by the formula $I_R = V_{DD}/2R_{EXT}$ (mA) with $R_{EXT}$ in kΩ.

**4:** FVR and BOR are disabled.

**5:** 0.1 μF capacitor on $V_{CAP}$.

**6:** 8 MHz crystal oscillator with 4x PLL enabled.

**Preliminary**
© 2011-2012 Microchip Technology Inc.

## 30.2    DC Characteristics: PIC16(L)F1782/3-I/E (Industrial, Extended) (Continued)

| PIC16LF1782/3 | **Standard Operating Conditions (unless otherwise stated)** Operating temperature     –40°C ≤ TA ≤ +85°C for industrial     –40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|
| PIC16F1782/3 | **Standard Operating Conditions (unless otherwise stated)** Operating temperature     –40°C ≤ TA ≤ +85°C for industrial     –40°C ≤ TA ≤ +125°C for extended | | | | | |

| Param No. | Device Characteristics | Min. | Typ† | Max. | Units | Conditions | |
|---|---|---|---|---|---|---|---|
| | | | | | | $V_{DD}$ | Note |
| | **Supply Current (IDD)**[1, 2] | | | | | | |
| D014 | | — | 125 | 550 | μA | 1.8 | $F_{OSC}$ = 4 MHz EC Oscillator mode Medium-Power mode |
| | | — | 280 | 1100 | μA | 3.0 | |
| D014 | | — | 220 | 650 | μA | 2.3 | $F_{OSC}$ = 4 MHz EC Oscillator mode **(Note 5)** Medium-Power mode |
| | | — | 290 | 1000 | μA | 3.0 | |
| | | — | 350 | 1200 | μA | 5.0 | |
| D015 | | — | 2.1 | 6.2 | mA | 3.0 | $F_{OSC}$ = 32 MHz EC Oscillator High-Power mode |
| | | — | 2.5 | 7.5 | mA | 3.6 | |
| D015 | | — | 2.1 | 6.5 | mA | 3.0 | $F_{OSC}$ = 32 MHz EC Oscillator High-Power mode **(Note 5)** |
| | | — | 2.2 | 7.5 | mA | 5.0 | |

**Note 1:**   The test conditions for all IDD measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD; MCLR = VDD; WDT disabled.

**2:**   The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

**3:**   For RC oscillator configurations, current through REXT is not included. The current through the resistor can be extended by the formula IR = VDD/2REXT (mA) with REXT in kΩ.

**4:**   FVR and BOR are disabled.

**5:**   0.1 μF capacitor on VCAP.

**6:**   8 MHz crystal oscillator with 4x PLL enabled.

# PIC16(L)F1782/3

## 30.2 DC Characteristics: PIC16(L)F1782/3-I/E (Industrial, Extended) (Continued)

| PIC16LF1782/3 | **Standard Operating Conditions (unless otherwise stated)** Operating temperature  -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC16F1782/3** | **Standard Operating Conditions (unless otherwise stated)** Operating temperature  -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended | | | | | | |

| Param No. | Device Characteristics | Min. | Typ† | Max. | Units | Conditions | |
|---|---|---|---|---|---|---|---|
| | | | | | | VDD | Note |
| | **Supply Current (IDD)**[1,2] | | | | | | |
| D017 | | — | 130 | 180 | μA | 1.8 | FOSC = 500 kHz MFINTOSC mode |
| | | — | 150 | 250 | μA | 3.0 | |
| D017 | | — | 150 | 250 | μA | 2.3 | FOSC = 500 kHz MFINTOSC mode **(Note 5)** |
| | | — | 170 | 330 | μA | 3.0 | |
| | | — | 220 | 430 | μA | 5.0 | |
| D019 | | — | 0.8 | 2.2 | mA | 1.8 | FOSC = 16 MHz HFINTOSC mode |
| | | — | 1.2 | 3.7 | mA | 3.0 | |
| D019 | | — | 1.0 | 2.3 | mA | 2.3 | FOSC = 16 MHz HFINTOSC mode **(Note 5)** |
| | | — | 1.3 | 3.9 | mA | 3.0 | |
| | | — | 1.4 | 4.1 | mA | 5.0 | |
| D020 | | — | 2.1 | 6.2 | mA | 3.0 | FOSC = 32 MHz HFINTOSC mode |
| | | — | 2.5 | 7.5 | mA | 3.6 | |
| D020 | | — | 2.1 | 6.5 | mA | 3.0 | FOSC = 32 MHz HFINTOSC mode |
| | | — | 2.2 | 7.5 | mA | 5.0 | |
| D022 | | — | 2.1 | 6.2 | mA | 3.0 | FOSC = 32 MHz HS Oscillator mode **(Note 6)** |
| | | — | 2.5 | 7.5 | mA | 3.6 | |
| D022 | | — | 2.1 | 6.5 | mA | 3.0 | FOSC = 32 MHz HS Oscillator mode **(Note 5, 6)** |
| | | — | 2.2 | 7.5 | mA | 5.0 | |

**Note 1:** The test conditions for all IDD measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD; MCLR = VDD; WDT disabled.

**2:** The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be extended by the formula IR = VDD/2REXT (mA) with REXT in kΩ.

**4:** FVR and BOR are disabled.

**5:** 0.1 μF capacitor on VCAP.

**6:** 8 MHz crystal oscillator with 4x PLL enabled.

**Preliminary**

## 30.3 DC Characteristics: PIC16(L)F1782/3-I/E (Power-Down)

| PIC16LF1782/3 | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended |
|---|---|
| PIC16F1782/3 | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended |

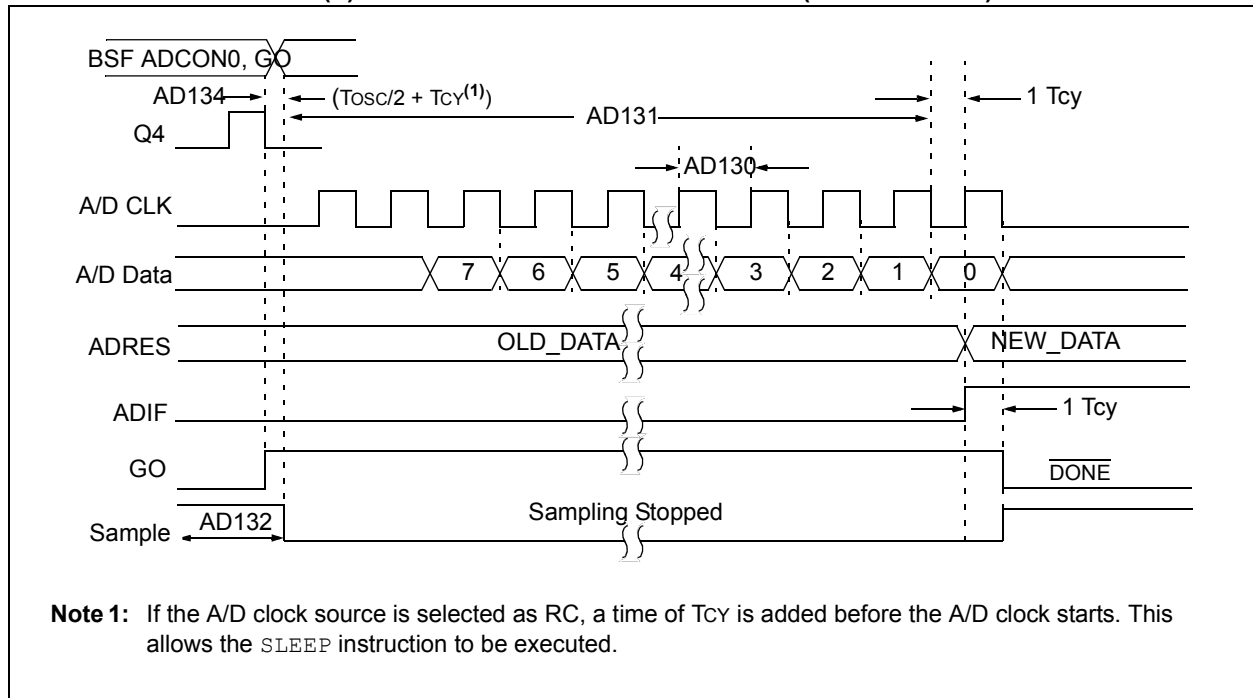| Param No. | Device Characteristics | Min. | Typ† | Max. +85°C | Max. +125°C | Units | Conditions VDD | Note |
|---|---|---|---|---|---|---|---|---|
| **Power-down Base Current (IPD)[2]** | | | | | | | | |
| D023 | | — | 0.05 | 1.0 | 8.0 | μA | 1.8 | WDT, BOR, FVR, and T1OSC disabled, all Peripherals Inactive |
| | | — | 0.08 | 2.0 | 9.0 | μA | 3.0 | |
| D023 | | — | 0.3 | 2 | 11 | μA | 2.3 | WDT, BOR, FVR, and T1OSC disabled, all Peripherals Inactive |
| | | — | 0.4 | 3 | 12 | μA | 3.0 | |
| | | — | 0.5 | 6 | 15 | μA | 5.0 | |
| D024 | | — | 0.5 | 6 | 14 | μA | 1.8 | LPWDT Current **(Note 1)** |
| | | — | 0.8 | 7 | 17 | μA | 3.0 | |
| D024 | | — | 0.8 | 6 | 15 | μA | 2.3 | LPWDT Current **(Note 1)** |
| | | — | 0.9 | 7 | 20 | μA | 3.0 | |
| | | — | 1.0 | 8 | 22 | μA | 5.0 | |
| D025 | | — | 15 | 25 | — | μA | 1.8 | FVR Current |
| | | — | 18 | 30 | — | μA | 3.0 | |
| D025 | | — | 18 | 33 | — | μA | 2.3 | FVR Current **(Note 4)** |
| | | — | 19 | 35 | — | μA | 3.0 | |
| | | — | 20 | 37 | — | μA | 5.0 | |
| D026 | | — | 7.5 | 17 | 20 | μA | 3.0 | BOR Current **(Note 1)** |
| D026 | | — | 40 | 77 | 30 | μA | 3.0 | BOR Current **(Note 1, Note 4)** |
| | | — | 87 | 20 | 40 | μA | 5.0 | |
| D027 | | — | 1 | 4 | 8 | μA | 3.0 | LPBOR Current **(Note 1)** |
| D028 | | — | 0.5 | 5 | 9 | μA | 1.8 | SOSC Current **(Note 1)** |
| | | — | 0.8 | 8.5 | 12 | μA | 3.0 | |
| D028 | | — | 1.1 | 6 | 10 | μA | 2.3 | SOSC Current **(Note 1)** |
| | | — | 1.3 | 8.5 | 20 | μA | 3.0 | |
| | | — | 1.4 | 10 | 25 | μA | 5.0 | |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Legend:** TBD = To Be Determined

**Note 1:** The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.

**2:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD, and VREGCON = 0x03.

**3:** A/D oscillator source is FRC.

**4:** 0.1 μF capacitor on VCAP.

# PIC16(L)F1782/3

## 30.3   DC Characteristics: PIC16(L)F1782/3-I/E (Power-Down) (Continued)

| PIC16LF1782/3 | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature     -40°C ≤ TA ≤ +85°C for industrial<br>                                      -40°C ≤ TA ≤ +125°C for extended | | | | | | |
|---|---|---|---|---|---|---|---|
| PIC16F1782/3 | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature     -40°C ≤ TA ≤ +85°C for industrial<br>                                      -40°C ≤ TA ≤ +125°C for extended | | | | | | |

| Param<br>No. | Device Characteristics | Min. | Typ† | Max.<br>+85°C | Max.<br>+125°C | Units | Conditions | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | VDD | Note |
| | **Power-down Base Current (IPD)**[2] | | | | | | | |
| D029 | | — | 9 | 18 | — | μA | 1.8 | A/D Current **(Note 1, Note 3)**, no conversion in progress |
| | | — | 11 | 22 | — | μA | 3.0 | |
| D029 | | — | 12 | 24 | — | μA | 2.3 | A/D Current **(Note 1, Note 3)**, no conversion in progress |
| | | — | 14 | 28 | — | μA | 3.0 | |
| | | — | 15 | 30 | — | μA | 5.0 | |
| D030 | | — | TBD | — | — | μA | 1.8 | A/D Current **(Note 1, Note 3)**, conversion in progress |
| | | — | TBD | — | — | μA | 3.0 | |
| D030 | | — | TBD | — | — | μA | 2.3 | A/D Current **(Note 1, Note 3, Note 4)**, conversion in progress |
| | | — | TBD | — | — | μA | 3.0 | |
| | | — | TBD | — | — | μA | 5.0 | |
| D031 | | — | 250 | — | — | μA | 3.0 | Op Amp (High power) |
| | | — | 280 | — | — | μA | 3.6 | |
| D031 | | — | 230 | — | — | μA | 2.3 | Op Amp (High power) |
| | | — | 250 | — | — | μA | 3.0 | |
| | | — | 350 | — | — | μA | 5.0 | |
| D032 | | — | 250 | — | — | μA | 1.8 | Comparator, High-Power mode |
| | | — | 300 | — | — | μA | 3.0 | |
| D032 | | — | 280 | — | — | μA | 2.3 | Comparator, High-Power mode |
| | | — | 300 | — | — | μA | 3.0 | |
| | | — | 310 | — | — | μA | 5.0 | |
| D033 | | — | TBD | — | — | μA | 3.0 | PSMC (64 MHz) |
| | | — | TBD | — | — | μA | 3.6 | |
| D033 | | — | TBD | — | — | μA | 2.3 | |
| | | — | TBD | — | — | μA | 3.0 | |
| | | — | TBD | — | — | μA | 5.0 | |

\*       These parameters are characterized but not tested.

†       Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Legend:**    TBD = To Be Determined

**Note   1:**   The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.

**2:**   The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD, and VREGCON = 0x03.

**3:**   A/D oscillator source is FRC.

**4:**   0.1 μF capacitor on VCAP.

**Preliminary**

# PIC16(L)F1782/3

## 30.4   DC Characteristics: PIC16(L)F1782/3-I/E

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| | VIL | **Input Low Voltage** | | | | | |
| | | I/O PORT: | | | | | |
| D034 | | with TTL buffer | — | — | 0.8 | V | 4.5V ≤ VDD ≤ 5.5V |
| D034A | | | — | — | 0.15 VDD | V | 1.8V ≤ VDD ≤ 4.5V |
| D035 | | with Schmitt Trigger buffer | — | — | 0.2 VDD | V | 2.0V ≤ VDD ≤ 5.5V |
| | | with I²C™ levels | — | — | 0.3 VDD | V | |
| | | with SMBus levels | — | — | 0.8 | V | 2.7V ≤ VDD ≤ 5.5V |
| D036 | | $\overline{MCLR}$, OSC1 (RC mode)[1] | — | — | 0.2 VDD | V | |
| D036A | | OSC1 (HS mode) | — | — | 0.3 VDD | V | |
| | VIH | **Input High Voltage** | | | | | |
| | | I/O ports: | | | | | |
| D040 | | with TTL buffer | 2.0 | — | — | V | 4.5V ≤ VDD ≤ 5.5V |
| D040A | | | 0.25 VDD + 0.8 | — | — | V | 1.8V ≤ VDD ≤ 4.5V |
| D041 | | with Schmitt Trigger buffer | 0.8 VDD | — | — | V | 2.0V ≤ VDD ≤ 5.5V |
| | | with I²C™ levels | 0.7 VDD | — | — | V | |
| | | with SMBus levels | 2.1 | — | — | V | 2.7V ≤ VDD ≤ 5.5V |
| D042 | | $\overline{MCLR}$ | 0.8 VDD | — | — | V | |
| D043A | | OSC1 (HS mode) | 0.7 VDD | — | — | V | |
| D043B | | OSC1 (RC mode) | 0.9 VDD | — | — | V | **(Note 1)** |
| | IIL | **Input Leakage Current**[2] | | | | | |
| D060 | | I/O ports | — | ± 5 | ± 125 | nA | VSS ≤ VPIN ≤ VDD, Pin at high-impedance @ 85°C |
| | | | | ± 5 | ± 1000 | nA | 125°C |
| D061 | | $\overline{MCLR}$[3] | — | ± 50 | ± 200 | nA | VSS ≤ VPIN ≤ VDD @ 85°C |
| | IPUR | **Weak Pull-up Current** | | | | | |
| D070* | | | 25 | 100 | 200 | | VDD = 3.3V, VPIN = VSS |
| | | | 25 | 140 | 300 | µA | VDD = 5.0V, VPIN = VSS |
| | VOL | **Output Low Voltage**[4] | | | | | |
| D080 | | I/O ports | — | — | 0.6 | V | IOL = 8mA, VDD = 5V<br>IOL = 6mA, VDD = 3.3V<br>IOL = 1.8mA, VDD = 1.8V |
| | VOH | **Output High Voltage**[4] | | | | | |
| D090 | | I/O ports | VDD - 0.7 | — | — | V | IOH = 3.5mA, VDD = 5V<br>IOH = 3mA, VDD = 3.3V<br>IOH = 1mA, VDD = 1.8V |

\*   These parameters are characterized but not tested.

†   Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:**   In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended to use an external clock in RC mode.

**2:**   Negative current is defined as current sourced by the pin.

**3:**   The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**4:**   Including OSC2 in CLKOUT mode.

# PIC16(L)F1782/3

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ T$_A$ ≤ +85°C for industrial -40°C ≤ T$_A$ ≤ +125°C for extended | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| | | **Capacitive Loading Specs on Output Pins** | | | | | |
| D101* | COSC2 | OSC2 pin | — | — | 15 | pF | In XT, HS and LP modes when external clock is used to drive OSC1 |
| D101A* | C$_{IO}$ | All I/O pins | — | — | 50 | pF | |
| | | **V$_{CAP}$ Capacitor Charging** | | | | | |
| D102 | | Charging current | — | 200 | — | µA | |
| D102A | | Source/sink capability when charging complete | — | 0.0 | — | mA | |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended to use an external clock in RC mode.

**2:** Negative current is defined as current sourced by the pin.

**3:** The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**4:** Including OSC2 in CLKOUT mode.

## 30.5 Memory Programming Requirements

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated) Operating temperature   -40°C ≤ TA ≤ +125°C | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| | | **Program Memory Programming Specifications** | | | | | |
| D110 | VIHH | Voltage on MCLR/VPP/RE3 pin | 8.0 | — | 9.0 | V | (Note 3) |
| D111 | IDDP | Supply Current during Programming | — | — | 10 | mA | |
| D112 | | VDD for Bulk Erase | 2.7 | — | VDDMAX | V | |
| D113 | VPEW | VDD for Write or Row Erase | VDDMIN | — | VDDMAX | V | |
| D114 | IPPPGM | Current on MCLR/VPP during Erase/Write | — | — | 1.0 | mA | |
| D115 | IDDPGM | Current on VDD during Erase/Write | — | | 5.0 | mA | |
| | | **Data EEPROM Memory** | | | | | |
| D116 | ED | Byte Endurance | 100K | — | — | E/W | -40°C to +85°C |
| D117 | VDRW | VDD for Read/Write | VDDMIN | — | VDDMAX | V | |
| D118 | TDEW | Erase/Write Cycle Time | — | 4.0 | 5.0 | ms | |
| D119 | TRETD | Characteristic Retention | — | 40 | — | Year | Provided no other specifications are violated |
| D120 | TREF | Number of Total Erase/Write Cycles before Refresh[2] | 100K | — | — | E/W | -40°C to +85°C |
| | | **Program Flash Memory** | | | | | |
| D121 | EP | Cell Endurance | 10K | — | — | E/W | -40°C to +85°C (**Note 1**) |
| D122 | VPR | VDD for Read | VDDMIN | — | VDDMAX | V | |
| D123 | TIW | Self-timed Write Cycle Time | — | 2 | 2.5 | ms | |
| D124 | 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:** Self-write and Block Erase.

2: Refer to **Section 12.2 "Using the Data EEPROM"** for a more detailed discussion on data EEPROM endurance.

3: Required only if single-supply programming is disabled.

# PIC16(L)F1782/3

## 30.6 Thermal Considerations

**Standard Operating Conditions (unless otherwise stated)**
Operating temperature  -40°C ≤ TA ≤ +125°C

| Param No. | Sym. | Characteristic | Typ. | Units | Conditions |
|---|---|---|---|---|---|
| TH01 | θJA | Thermal Resistance Junction to Ambient | 60 | °C/W | 28-pin SPDIP package |
| | | | 80 | °C/W | 28-pin SOIC package |
| | | | 90 | °C/W | 28-pin SSOP package |
| | | | 27.5 | °C/W | 28-pin UQFN 4x4mm package |
| | | | 27.5 | °C/W | 28-pin QFN 6x6mm package |
| TH02 | θJC | Thermal Resistance Junction to Case | 31.4 | °C/W | 28-pin SPDIP package |
| | | | 24 | °C/W | 28-pin SOIC package |
| | | | 24 | °C/W | 28-pin SSOP package |
| | | | 24 | °C/W | 28-pin UQFN 4x4mm package |
| | | | 24 | °C/W | 28-pin QFN 6x6mm package |
| TH03 | TJMAX | Maximum Junction Temperature | 150 | °C | |
| TH04 | PD | Power Dissipation | — | W | PD = PINTERNAL + PI/O |
| TH05 | PINTERNAL | Internal Power Dissipation | — | W | PINTERNAL = IDD x VDD[1] |
| TH06 | PI/O | I/O Power Dissipation | — | W | PI/O = Σ (IOL * VOL) + Σ (IOH * (VDD - VOH)) |
| TH07 | PDER | Derated Power | — | W | PDER = PDMAX (TJ - TA)/θJA[2] |

**Note 1:** IDD is current to run the chip alone without driving any load on the output pins.
    **2:** TA = Ambient Temperature
    **3:** TJ = Junction Temperature

**Preliminary**

## 30.7 Timing Parameter Symbology

The timing parameter symbols have been created with one of the following formats:

1. TppS2ppS
2. TppS

| T | | |
|---|---|---|
| F | Frequency | T | Time |

Lowercase letters (pp) and their meanings:

| pp | | | | |
|---|---|---|---|---|
| cc | CCP1 | osc | OSC1 |
| ck | CLKOUT | rd | $\overline{RD}$ |
| cs | $\overline{CS}$ | rw | $\overline{RD}$ or $\overline{WR}$ |
| di | SDI | sc | SCK |
| do | SDO | ss | $\overline{SS}$ |
| dt | Data in | t0 | T0CKI |
| io | I/O PORT | t1 | T1CKI |
| mc | $\overline{MCLR}$ | wr | $\overline{WR}$ |

Uppercase letters and their meanings:

| S | | | | |
|---|---|---|---|---|
| F | Fall | P | Period |
| H | High | R | Rise |
| I | Invalid (High-impedance) | V | Valid |
| L | Low | Z | High-impedance |

**FIGURE 30-5: LOAD CONDITIONS**

Load Condition

Pin — CL — Vss

**Legend:** CL = 50 pF for all pins, 15 pF for OSC2 output
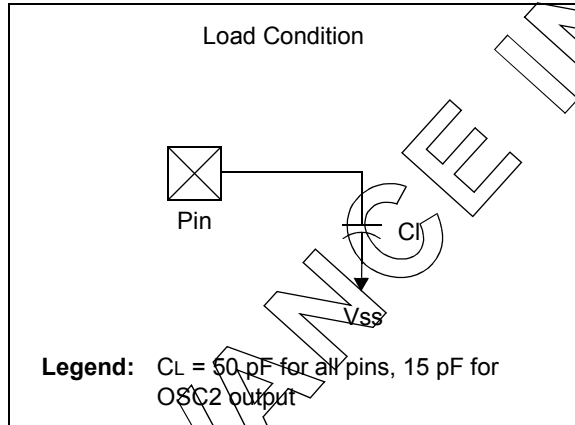
# PIC16(L)F1782/3

## 30.8 AC Characteristics: PIC16(L)F1782/3-I/E

### FIGURE 30-6: CLOCK TIMING
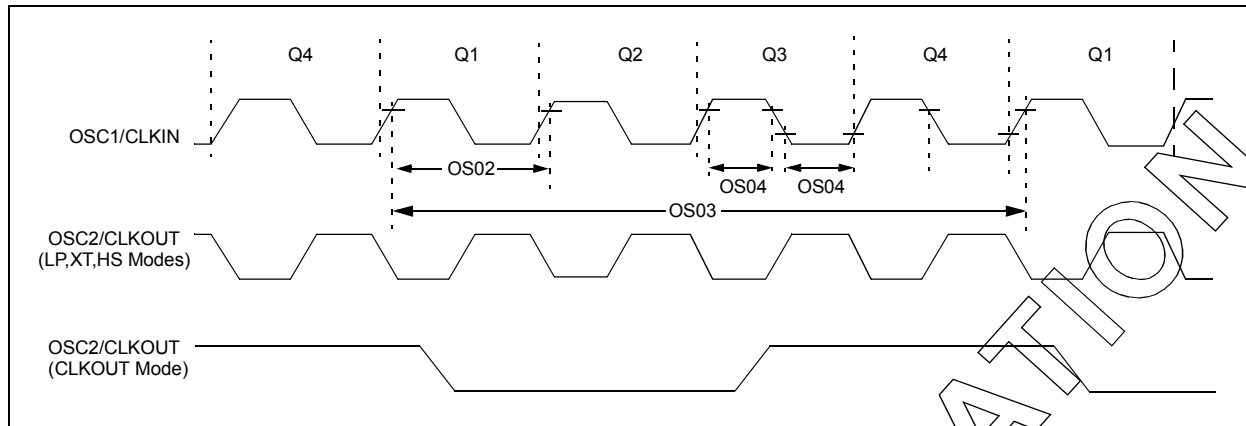


### TABLE 30-1: CLOCK OSCILLATOR TIMING REQUIREMENTS

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature    -40°C ≤ TA ≤ +125°C | | | | | | | |
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ†** | **Max.** | **Units** | **Conditions** |
| OS01 | FOSC | External CLKIN Frequency[1] | DC | — | 0.5 | MHz | EC Oscillator mode (low) |
| | | | DC | — | 4 | MHz | EC Oscillator mode (medium) |
| | | | DC | — | 20 | MHz | EC Oscillator mode (high) |
| | | Oscillator Frequency[1] | — | 32.768 | — | kHz | LP Oscillator mode |
| | | | 0.1 | — | 4 | MHz | XT Oscillator mode |
| | | | 1 | — | 4 | MHz | HS Oscillator mode |
| | | | 1 | — | 20 | MHz | HS Oscillator mode, VDD > 2.3V |
| | | | DC | — | 4 | MHz | RC Oscillator mode, VDD > 2.0V |
| OS02 | TOSC | External CLKIN Period[1] | 27 | — | ∞ | μs | LP Oscillator mode |
| | | | 250 | — | ∞ | ns | XT Oscillator mode |
| | | | 50 | — | ∞ | ns | HS Oscillator mode |
| | | | 50 | — | ∞ | ns | EC Oscillator mode |
| | | Oscillator Period[1] | — | 30.5 | — | μs | LP Oscillator mode |
| | | | 250 | — | 10,000 | ns | XT Oscillator mode |
| | | | 50 | — | 1,000 | ns | HS Oscillator mode |
| | | | 250 | — | — | ns | RC Oscillator mode |
| OS03 | TCY | Instruction Cycle Time[1] | 200 | TCY | DC | ns | TCY = 4/FOSC |
| OS04* | TosH, TosL | External CLKIN High, External CLKIN Low | 2 | — | — | μs | LP oscillator |
| | | | 100 | — | — | ns | XT oscillator |
| | | | 20 | — | — | ns | HS oscillator |
| OS05* | TosR, TosF | External CLKIN Rise, External CLKIN Fall | 0 | — | ∞ | ns | LP oscillator |
| | | | 0 | — | ∞ | ns | XT oscillator |
| | | | 0 | — | ∞ | ns | HS oscillator |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min" values with an external clock applied to OSC1 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

**Preliminary**     © 2011-2012 Microchip Technology Inc.

## TABLE 30-2: OSCILLATOR PARAMETERS

| Standard Operating Conditions (unless otherwise stated) Operating Temperature -40°C ≤ TA ≤ +125°C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Freq. Tolerance | Min. | Typ† | Max. | Units | Conditions |
| OS08 | HFOSC | Internal Calibrated HFINTOSC Frequency[2] | ±2% | — | 16.0 | — | MHz | 0°C ≤ TA ≤ +60°C, VDD ≥ 2.5V |
| | | | ±3% | — | 16.0 | — | MHz | 60°C ≤ TA ≤ 85°C, VDD ≥ 2.5V |
| | | | ±5% | — | 16.0 | — | MHz | -40°C ≤ TA ≤ +125°C |
| OS08A | MFOSC | Internal Calibrated MFINTOSC Frequency[2] | ±2% | — | 500 | — | kHz | 0°C ≤ TA ≤ +60°C, VDD ≥ 2.5V |
| | | | ±3% | — | 500 | — | kHz | 60°C ≤ TA ≤ 85°C, VDD ≥ 2.5V |
| | | | ±5% | — | 500 | — | kHz | -40°C ≤ TA ≤ +125°C |
| OS09 | LFOSC | Internal LFINTOSC Frequency | — | — | 31 | — | kHz | -40°C ≤ TA ≤ +125°C |
| OS10* | TIOSC ST | HFINTOSC Wake-up from Sleep Start-up Time | — | — | 5 | 8 | μs | |
| | | MFINTOSC Wake-up from Sleep Start-up Time | — | — | 20 | 30 | μs | |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min" values with an external clock applied to the OSC1 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

**2:** To ensure these oscillator frequency tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

**3:** By design.

## TABLE 30-3: PLL CLOCK TIMING SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) Operating Temperature -40°C ≤ TA ≤ +125°C, 2.7V ≤ VDD ≤ 5.5V | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| F10 | FOSC | Oscillator Frequency Range | 4 | — | 8 | MHz | |
| F11 | FSYS | On-Chip VCO System Frequency | 16 | — | 32 | MHz | |
| F12 | TRC | PLL Start-up Time (Lock Time) | — | — | 2 | ms | |
| F13* | ΔCLK | CLKOUT Stability (Jitter) | -0.25% | — | +0.25% | % | |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

# PIC16(L)F1782/3

**FIGURE 30-7:** **CLKOUT AND I/O TIMING**



**Preliminary**
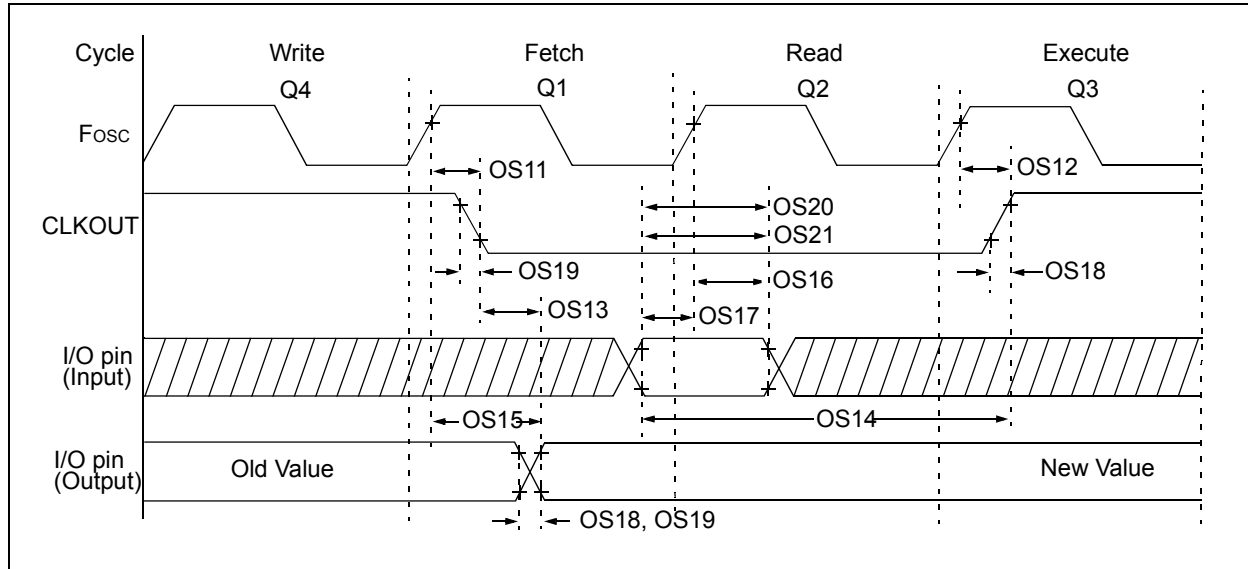
**TABLE 30-4:** **CLKOUT AND I/O TIMING PARAMETERS**

| | | **Standard Operating Conditions (unless otherwise stated)** | | | | | |
|---|---|---|---|---|---|---|---|
| | | Operating Temperature -40°C ≤ TA ≤ +125°C | | | | | |
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ†** | **Max.** | **Units** | **Conditions** |
| OS11 | TosH2ckL | Fosc↑ to CLKOUT↓ [1] | — | — | 70 | ns | VDD = 3.3-5.0V |
| OS12 | TosH2ckH | Fosc↑ to CLKOUT↑ [1] | — | — | 72 | ns | VDD = 3.3-5.0V |
| OS13 | TckL2ioV | CLKOUT↓ to Port out valid[1] | — | — | 20 | ns | |
| OS14 | TioV2ckH | Port input valid before CLKOUT↑[1] | TOSC + 200 ns | — | — | ns | |
| OS15 | TosH2ioV | Fosc↑ (Q1 cycle) to Port out valid | — | 50 | 70* | ns | VDD = 3.3-5.0V |
| OS16 | TosH2ioI | Fosc↑ (Q2 cycle) to Port input invalid (I/O in hold time) | 50 | — | — | ns | VDD = 3.3-5.0V |
| OS17 | TioV2osH | Port input valid to Fosc↑ (Q2 cycle) (I/O in setup time) | 20 | — | — | ns | |
| OS18 | TioR | Port output rise time[2] | — | 40 | 72 | ns | VDD = 1.8V |
| | | | — | 15 | 32 | | VDD = 3.3-5.0V |
| OS19 | TioF | Port output fall time[2] | — | 28 | 55 | ns | VDD = 1.8V |
| | | | — | 15 | 30 | | VDD = 3.3-5.0V |
| OS20* | Tinp | INT pin input high or low time | 25 | — | — | ns | |
| OS21* | Tioc | Interrupt-on-change new input level time | 25 | — | — | ns | |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated.

**Note 1:** Measurements are taken in RC mode where CLKOUT output is 4 x TOSC.

**2:** Includes OSC2 in CLKOUT mode.

**FIGURE 30-8:** **RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**



**Note 1:** Asserted low.

**FIGURE 30-9:** **BROWN-OUT RESET TIMING AND CHARACTERISTICS**



**Note 1:** 64 ms delay only if PWRTE bit in the Configuration Word register is programmed to '0'.
2 ms delay if PWRTE = 0 and VREGEN = 1.

**Preliminary** © 2011-2012 Microchip Technology Inc.
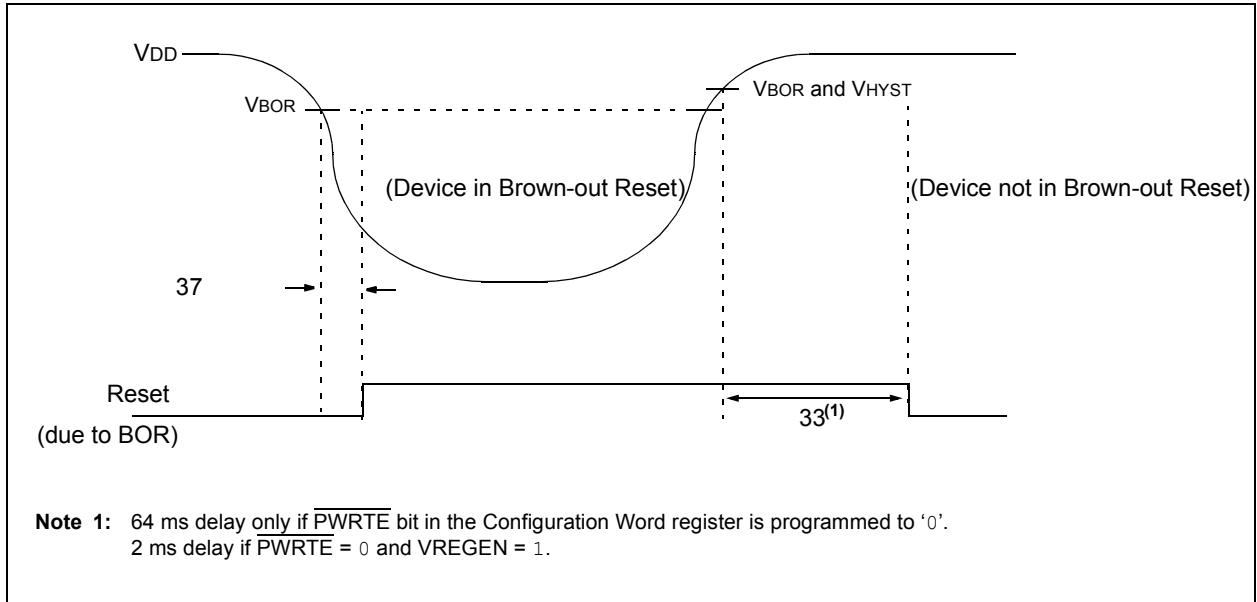
**TABLE 30-5:** **RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET PARAMETERS**

| **Standard Operating Conditions (unless otherwise stated)** Operating Temperature -40°C ≤ TA ≤ +125°C |
|---|

| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 30 | TMCL | $\overline{\text{MCLR}}$ Pulse Width (low) | 2<br>5 | —<br>— | —<br>— | μs<br>μs | VDD = 3.3-5V, -40°C to +85°C<br>VDD = 3.3-5V |
| 31 | TWDTLP | Low-Power Watchdog Timer Time-out Period | 10 | 16 | 27 | ms | VDD = 3.3V-5V<br>1:16 Prescaler used |
| 32 | TOST | Oscillator Start-up Timer Period[1], [2] | — | 1024 | — | Tosc | **(Note 3)** |
| 33* | TPWRT | Power-up Timer Period, $\overline{\text{PWRTE}}$ = 0 | 40 | 65 | 140 | ms | |
| 34* | TIOZ | I/O high-impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset | — | — | 2.0 | μs | |
| 35 | VBOR | Brown-out Reset Voltage | 2.55<br>2.30<br>1.80 | 2.70<br>2.45<br>1.90 | 2.85<br>2.6<br>2.10 | V<br>V<br>V | BORV = 0<br>BORV=1 (F device)<br>BORV=1 (LF device) |
| 35A | VLPBOR | Low-Power Brown-out | 1.8 | 2.1 | 2.5 | V | LPBOR = 1 |
| 36* | VHYST | Brown-out Reset Hysteresis | 0 | 25 | 75 | mV | -40°C to +85°C |
| 37* | TBORDC | Brown-out Reset DC Response Time | 1 | 3 | 5 | μs | VDD ≤ VBOR |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min" values with an external clock applied to the OSC1 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

**2:** By design.

**3:** Period of the slower clock.

**4:** To ensure these voltage tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

**FIGURE 30-10:** **TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**
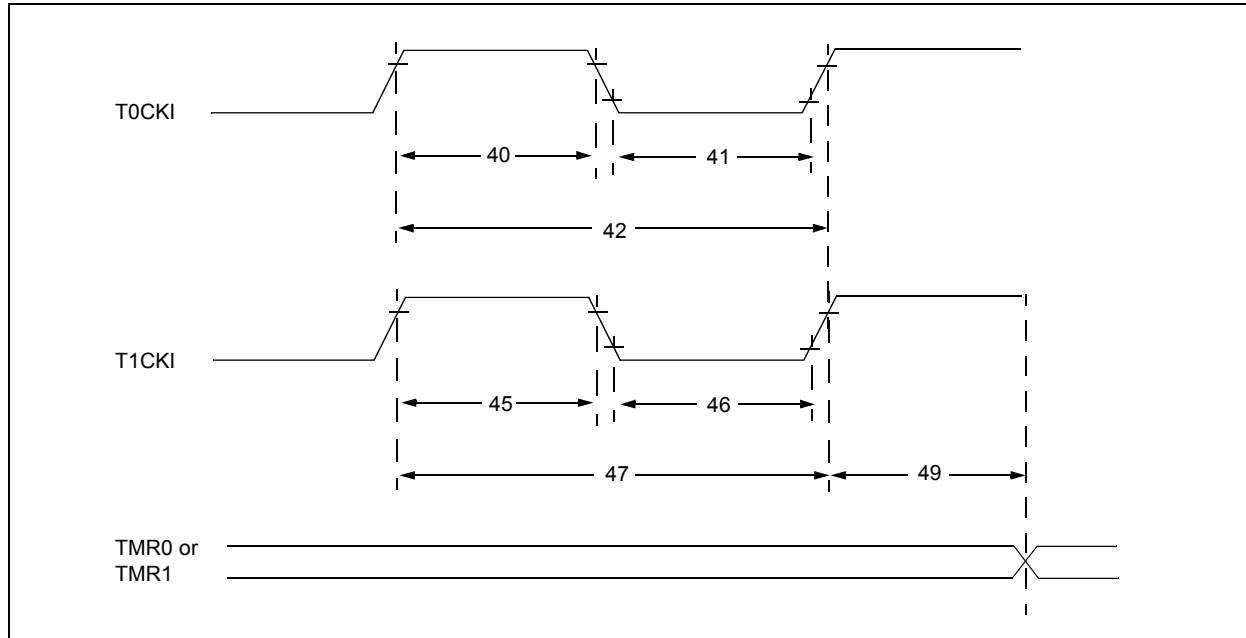


**TABLE 30-6:** **TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

**Standard Operating Conditions (unless otherwise stated)**
Operating Temperature  -40°C ≤ T$_A$ ≤ +125°C

| Param No. | Sym. | Characteristic | | Min. | Typ† | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| 40* | T$_T$0H | T0CKI High Pulse Width | No Prescaler | 0.5 T$_{CY}$ + 20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 41* | T$_T$0L | T0CKI Low Pulse Width | No Prescaler | 0.5 T$_{CY}$ + 20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 42* | T$_T$0P | T0CKI Period | | Greater of: 20 or $\frac{T_{CY} + 40}{N}$ | — | — | ns | N = prescale value (2, 4, ..., 256) |
| 45* | T$_T$1H | T1CKI High Time | Synchronous, No Prescaler | 0.5 T$_{CY}$ + 20 | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 46* | T$_T$1L | T1CKI Low Time | Synchronous, No Prescaler | 0.5 T$_{CY}$ + 20 | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 47* | T$_T$1P | T1CKI Input Period | Synchronous | Greater of: 30 or $\frac{T_{CY} + 40}{N}$ | — | — | ns | N = prescale value (1, 2, 4, 8) |
| | | | Asynchronous | 60 | — | — | ns | |
| 48 | F$_T$1 | Timer1 Oscillator Input Frequency Range (oscillator enabled by setting bit T1OSCEN) | | 32.4 | 32.768 | 33.1 | kHz | |
| 49* | TCKEZ$_{TMR1}$ | Delay from External Clock Edge to Timer Increment | | 2 T$_{OSC}$ | — | 7 T$_{OSC}$ | — | Timers in Sync mode |

\*    These parameters are characterized but not tested.

†    Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

 © 2011-2012 Microchip Technology Inc.

### FIGURE 30-11: CAPTURE/COMPARE/PWM TIMINGS (CCP)


CCPx
(Capture mode)
CC01
CC02
CC03

**Note:** Refer to Figure 30.5 for load conditions.
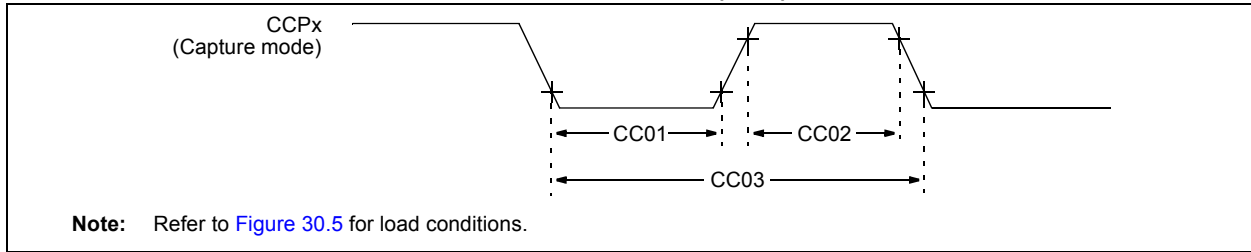
### TABLE 30-7: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)

**Standard Operating Conditions (unless otherwise stated)**
Operating Temperature -40°C ≤ TA ≤ +125°C

| Param No. | Sym. | Characteristic | | Min. | Typ† | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| CC01* | TccL | CCPx Input Low Time | No Prescaler | $0.5T_{CY} + 20$ | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC02* | TccH | CCPx Input High Time | No Prescaler | $0.5T_{CY} + 20$ | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC03* | TccP | CCPx Input Period | | $\frac{3T_{CY} + 40}{N}$ | — | — | ns | N = prescale value (1, 4 or 16) |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Downloaded from Datasheet.su

# PIC16(L)F1782/3

**TABLE 30-8:** **PIC16(L)F1782/3 A/D CONVERTER (ADC) 12-BIT DIFFERENTIAL CHARACTERISTICS:**

| **Operating Conditions** | | | | | | | |
|---|---|---|---|---|---|---|---|
| VDD = 3V, Temp. = 25°C, Single-ended 2 μs TAD, VREF+ = 3V, VREF- = VSS | | | | | | | |
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| AD01 | NR | Resolution | — | — | 10 | bit | |
| AD02 | EIL | Integral Error | — | ±1 | ±1.6 | LSb | |
| AD03 | EDL | Differential Error | — | ±1 | ±1.4 | LSb | No missing codes |
| AD04 | EOFF | Offset Error | — | ±1 | ±2 | LSb | |
| AD05 | EGN | Gain Error | — | ±1 | ±2 | LSb | |
| AD06 | VREF | Reference Voltage[3] | 1.8 | — | VDD | V | VREF = (VREF+ minus VREF-) (**Note 5**) |
| AD07 | VAIN | Full-Scale Range | — | — | VREF | V | |
| AD08 | ZAIN | Recommended Impedance of Analog Voltage Source | — | — | 10 | kΩ | Can go higher if external 0.01μF capacitor is present on input pin. |
| | | | | | | | |
| AD09 | NR | Resolution | — | — | 12 | bit | |
| AD10 | EIL | Integral Error | — | ±2 | — | LSb | |
| AD11 | EDL | Differential Error | — | ±2 | — | LSb | |
| AD12 | EOFF | Offset Error | — | ±1 | — | LSb | |
| AD13 | EGN | Gain Error | — | ±1 | — | LSb | |
| AD14 | VREF | Reference Voltage[3] | 1.8 | — | VDD | V | VREF = (VREF+ minus VREF-) (**Note 5**) |
| AD15 | VAIN | Full-Scale Range | — | — | VREF | V | |
| AD16 | ZAIN | Recommended Impedance of Analog Voltage Source | — | — | 10 | kΩ | Can go higher if external 0.01μF capacitor is present on input pin. |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Total Absolute Error includes integral, differential, offset and gain errors.

**2:** The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

**3:** ADC VREF is from external VREF, VDD pin or FVR, whichever is selected as reference input.

**4:** When ADC is off, it will not consume any current other than leakage current. The power-down current specification includes any such leakage from the ADC module.

**5:** FVR voltage selected must be 2.048V or 4.096V.

**TABLE 30-9:** **PIC16(L)F1782/3 A/D CONVERSION REQUIREMENTS**

| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Operating temperature    -40°C ≤ TA ≤ +125°C | | | | | | | |
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| AD130* | TAD | A/D Clock Period | 1.0 | — | 9.0 | μs | TOSC-based |
| | | A/D Internal RC Oscillator Period | 1.0 | 1.6 | 6.0 | μs | ADCS<1:0> = 11 (ADRC mode) |
| AD131 | TCNV | Conversion Time (not including Acquisition Time)[1] | — | 15 | — | TAD | Set GO/DONE bit to conversion complete |
| AD132* | TACQ | Acquisition Time | — | 5.0 | — | μs | |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The ADRES register may be read on the following TCY cycle.

**Preliminary** © 2011-2012 Microchip Technology Inc.

**FIGURE 30-12: PIC16(L)F1782/3 A/D CONVERSION TIMING (NORMAL MODE)**



**Note 1:** If the A/D clock source is selected as RC, a time of $T_{CY}$ is added before the A/D clock starts. This allows the `SLEEP` instruction to be executed.
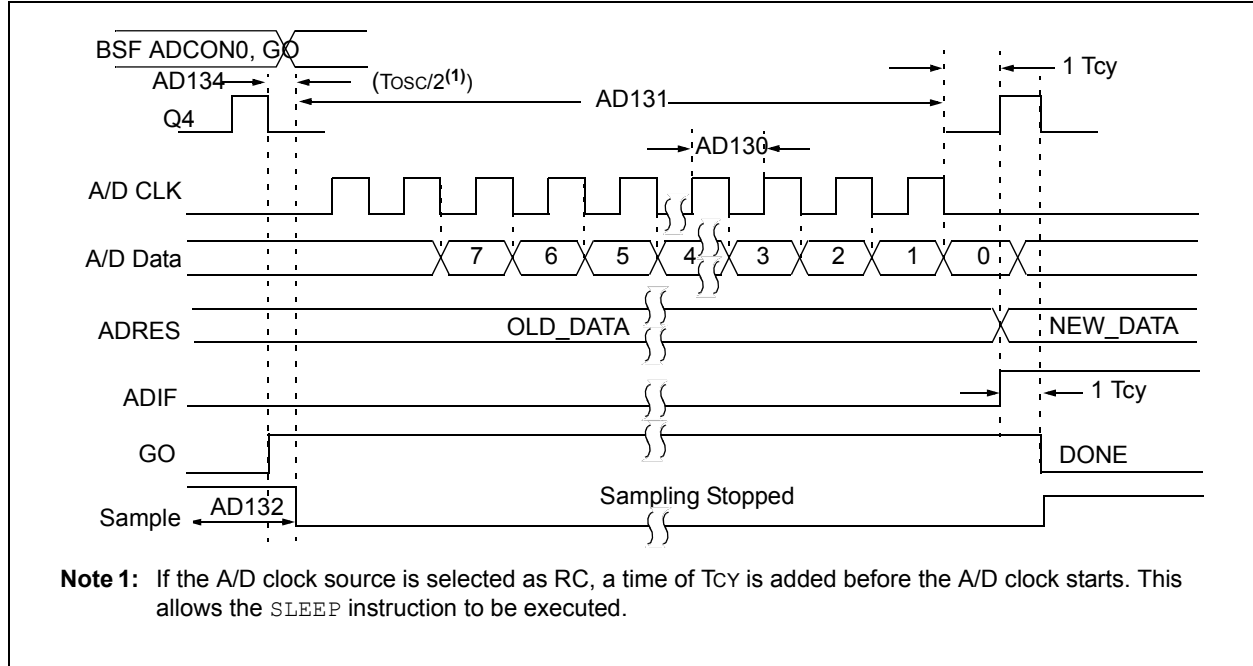
**FIGURE 30-13: PIC16(L)F1782/3 A/D CONVERSION TIMING (SLEEP MODE)**



**Note 1:** If the A/D clock source is selected as RC, a time of $T_{CY}$ is added before the A/D clock starts. This allows the `SLEEP` instruction to be executed.

# PIC16(L)F1782/3

## TABLE 30-10: OPERATIONAL AMPLIFIER (OPA)

| DC CHARACTERISTICS | | | | | | | Standard Operating Conditions (unless otherwise stated): $V_{DD}$ = 3.0 Temperature 25°C, High-Power Mode |
|---|---|---|---|---|---|---|---|
| Param No. | Symbol | Parameters | Min | Typ | Max | Units | Conditions |
| OPA01 | GBWP | Gain Bandwidth Product | — | 4.3 | — | MHz | High-Power mode |
| OPA02 | $T_{ON}$ | Turn on Time | — | 10 | — | µs | |
| OPA03 | $P_M$ | Phase Margin | — | 60 | — | degrees | |
| OPA04 | $S_R$ | Slew Rate | — | 3 | — | V/µs | |
| OPA05 | $O_{FF}$ | Offset | — | ±2 | ±5 | mV | |
| OPA06 | $C_{MRR}$ | Common Mode Rejection Ratio | 60 | 70 | — | dB | |
| OPA07 | $A_{OL}$ | Open Loop Gain | — | 90 | — | dB | |

## TABLE 30-11: COMPARATOR SPECIFICATIONS

| Operating Conditions: $V_{DD}$ = 3.0V, Temperature = 25°C (unless otherwise stated). | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristics | Min. | Typ. | Max. | Units | Comments |
| CM01 | $V_{IOFF}$ | Input Offset Voltage | — | ±2.5 | ±5 | mV | High Power mode $V_{ICM}$ = $V_{DD}$/2 |
| CM02 | $V_{ICM}$ | Input Common Mode Voltage | 0 | — | $V_{DD}$ | V | |
| CM03 | CMRR | Common Mode Rejection Ratio | 40 | 50 | — | dB | |
| CM04A | | Response Time Rising Edge | — | 60 | 85 | ns | High-Power mode measured at $V_{DD}$/2 100 mV Overdrive |
| CM04B | $T_{RESP}$ | Response Time Falling Edge | — | 60 | 90 | ns | High-Power mode measured at $V_{DD}$/2 100 mV Overdrive |
| CM04C | | Response Time Rising Edge | — | 85 | — | ns | Low-Power mode measured at $V_{DD}$/2 100 mV Overdrive |
| CM04D | | Response Time Falling Edge | — | 85 | — | ns | Low-Power mode measured at $V_{DD}$/2 100 mV Overdrive |
| CM05 | Tmc2ov | Comparator Mode Change to Output Valid* | — | — | 10 | µs | |
| CM06 | $C_{HYSTER}$ | Comparator Hysteresis | 20 | 45 | 75 | mV | Hystersis ON, High Power measured at $V_{DD}$/2 **(Note 2)** |

    \*     These parameters are characterized but not tested.

**Note 1:**   Response time measured with one comparator input at $V_{DD}$/2, while the other input transitions from $V_{SS}$ to $V_{DD}$.

    **2:**   Comparator Hysteresis is available when the CxHYS bit of the CMxCON0 register is enabled.

© 2011-2012 Microchip Technology Inc.

## TABLE 30-12: DIGITAL-TO-ANALOG CONVERTER (DAC) SPECIFICATIONS

| Param No. | Sym. | Characteristics | Min. | Typ. | Max. | Units | Comments |
|---|---|---|---|---|---|---|---|
| DAC01* | C$_{LSB}$ | Step Size | — | V$_{DD}$/256 | — | V | |
| DAC02* | C$_{ACC}$ | Absolute Accuracy | — | — | ± 1 | LSb | |
| DAC03* | C$_R$ | Unit Resistor Value (R) | — | 600 | — | Ω | |
| DAC04* | C$_{ST}$ | Settling Time[1] | — | — | 10 | µs | |

**\*** These parameters are characterized but not tested.

**Legend:** TBD = To Be Determined

**Note 1:** Settling time measured while DACR<7:0> transitions from '`0x00`' to '`0xFF`'.

## FIGURE 30-14: EUSART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING



**Note:** Refer to Figure 30-5 for load conditions.
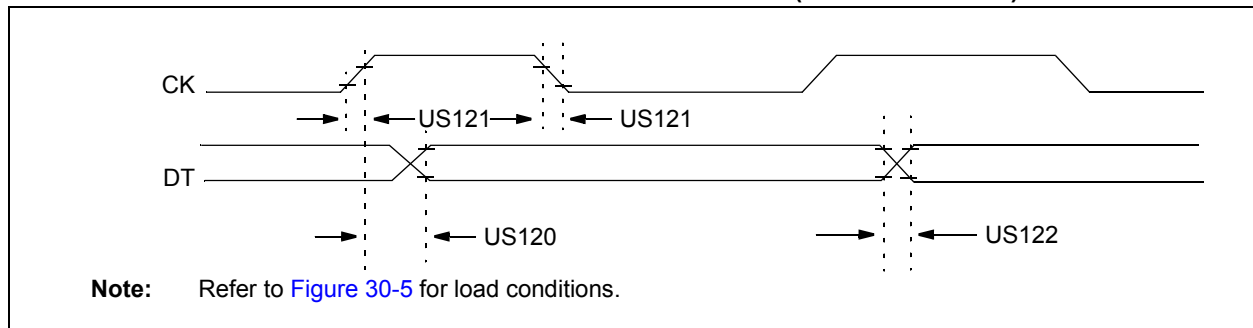
## TABLE 30-13: EUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS

| Param. No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| | | **Standard Operating Conditions (unless otherwise stated)** Operating Temperature    -40°C ≤ T$_A$ ≤ +125°C | | | | | |
| US120 | T$_{CKH2DTV}$ | SYNC XMIT (Master and Slave) Clock high to data-out valid | 3.0-5.5V | — | 80 | ns | |
| | | | 1.8-5.5V | — | 100 | ns | |
| US121 | T$_{CKRF}$ | Clock out rise time and fall time (Master mode) | 3.0-5.5V | — | 45 | ns | |
| | | | 1.8-5.5V | — | 50 | ns | |
| US122 | T$_{DTRF}$ | Data-out rise time and fall time | 3.0-5.5V | — | 45 | ns | |
| | | | 1.8-5.5V | — | 50 | ns | |

## FIGURE 30-15: EUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



**Note:** Refer to Figure 30-5 for load conditions.
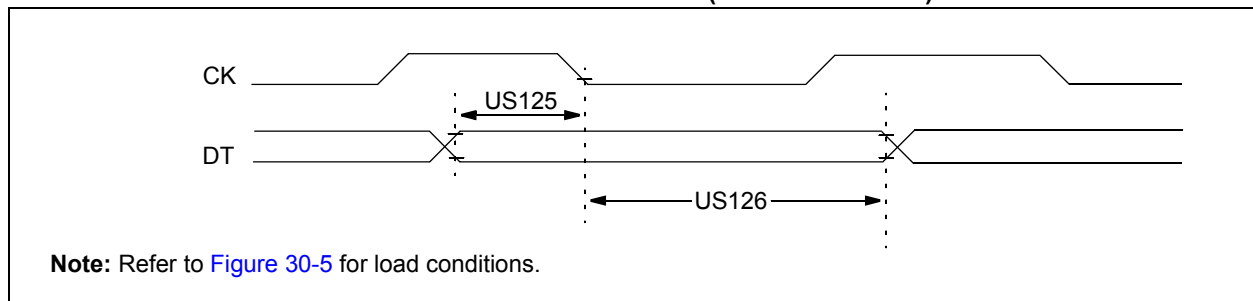
**TABLE 30-14: EUSART SYNCHRONOUS RECEIVE REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| **Standard Operating Conditions (unless otherwise stated)** <br> Operating Temperature    -40°C ≤ TA ≤ +125°C | | | | | | |
| US125 | TDTV2CKL | SYNC RCV (Master and Slave) <br> Data-hold before CK ↓ (DT hold time) | 10 | — | ns | |
| US126 | TCKL2DTL | Data-hold after CK ↓ (DT hold time) | 15 | — | ns | |

**Preliminary**                    © 2011-2012 Microchip Technology Inc.

**FIGURE 30-16:** **SPI MASTER MODE TIMING (CKE = 0, SMP = 0)**



**Note:** Refer to Figure 30-5 for load conditions.
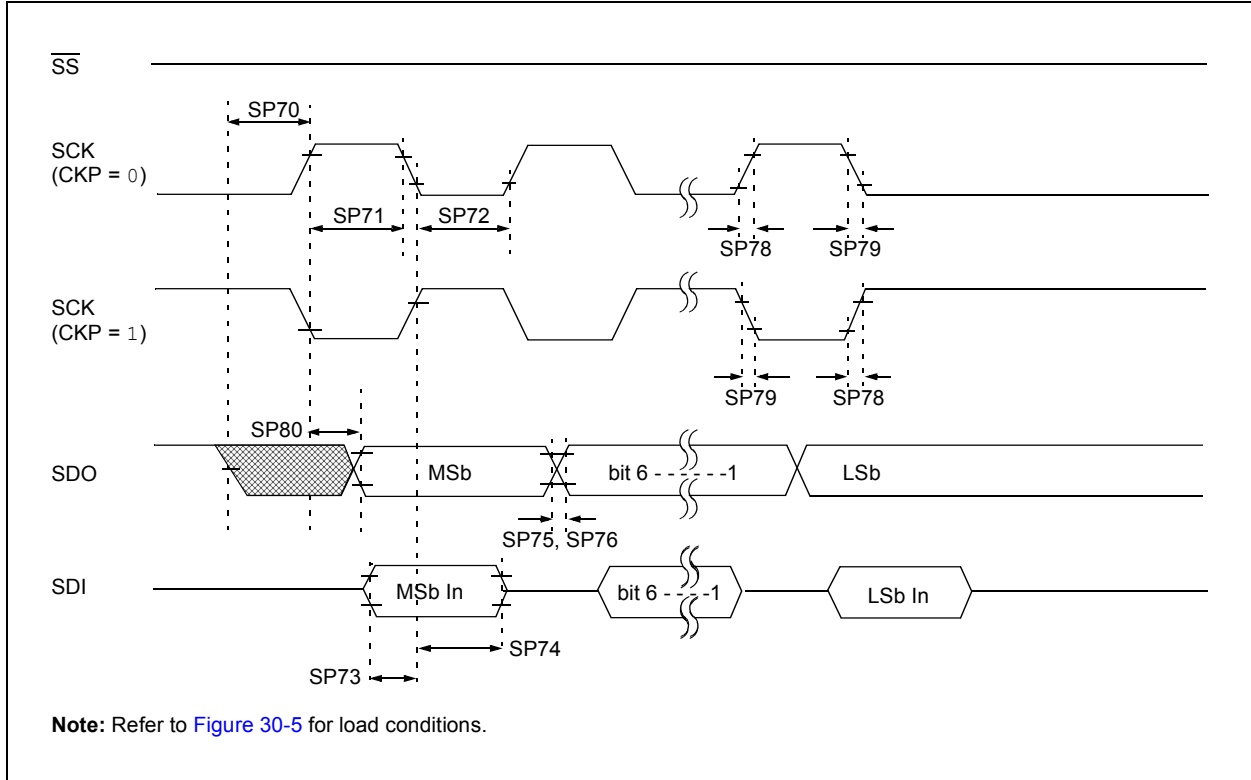
**FIGURE 30-17:** **SPI MASTER MODE TIMING (CKE = 1, SMP = 1)**



**Note:** Refer to Figure 30-5 for load conditions.
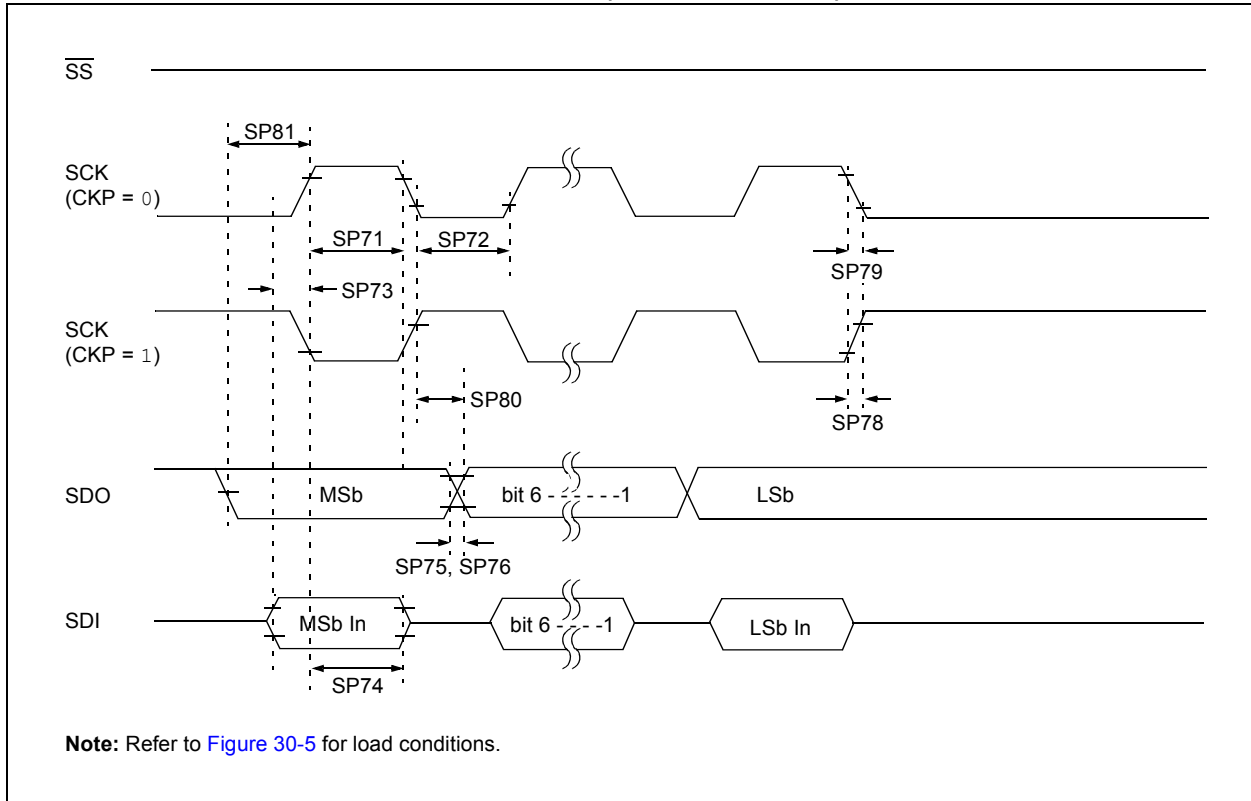
# PIC16(L)F1782/3

**FIGURE 30-18:** **SPI SLAVE MODE TIMING (CKE = 0)**



**Note:** Refer to Figure 30-5 for load conditions.
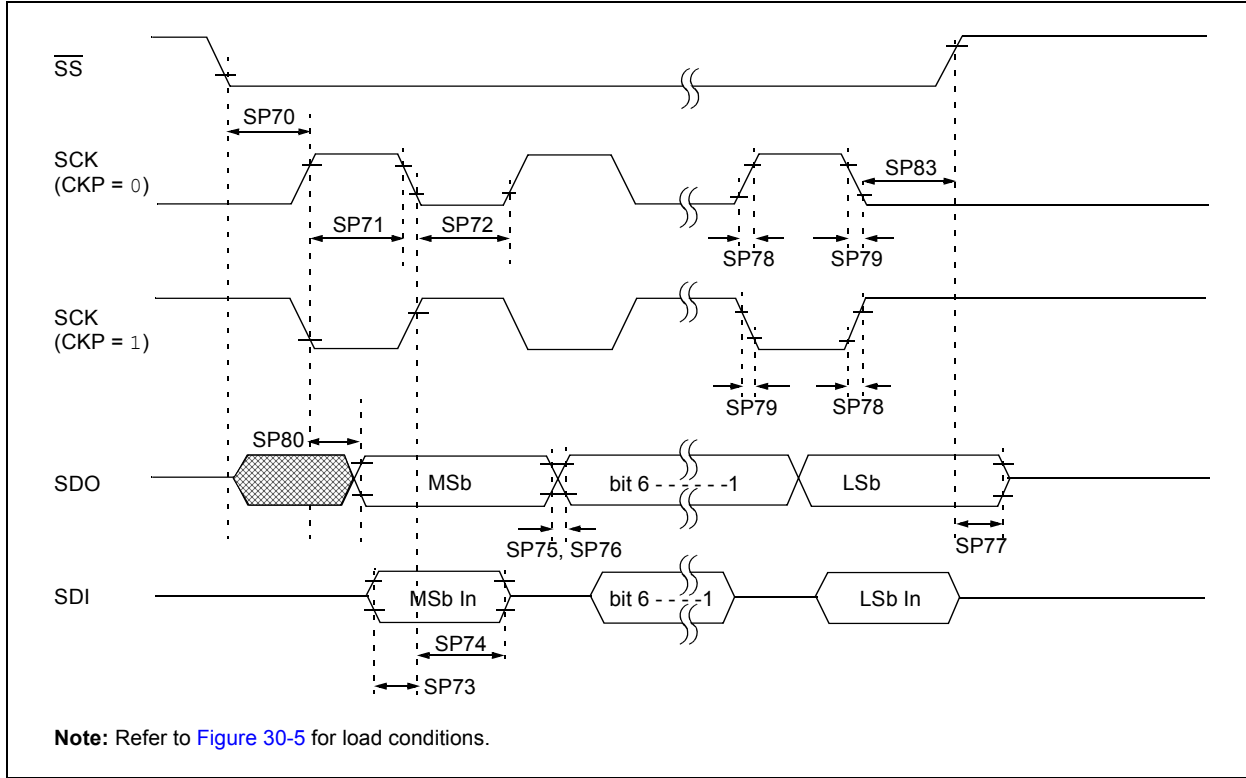
**FIGURE 30-19:** **SPI SLAVE MODE TIMING (CKE = 1)**



**Note:** Refer to Figure 30-5 for load conditions.

---

**Preliminary**
© 2011-2012 Microchip Technology Inc.
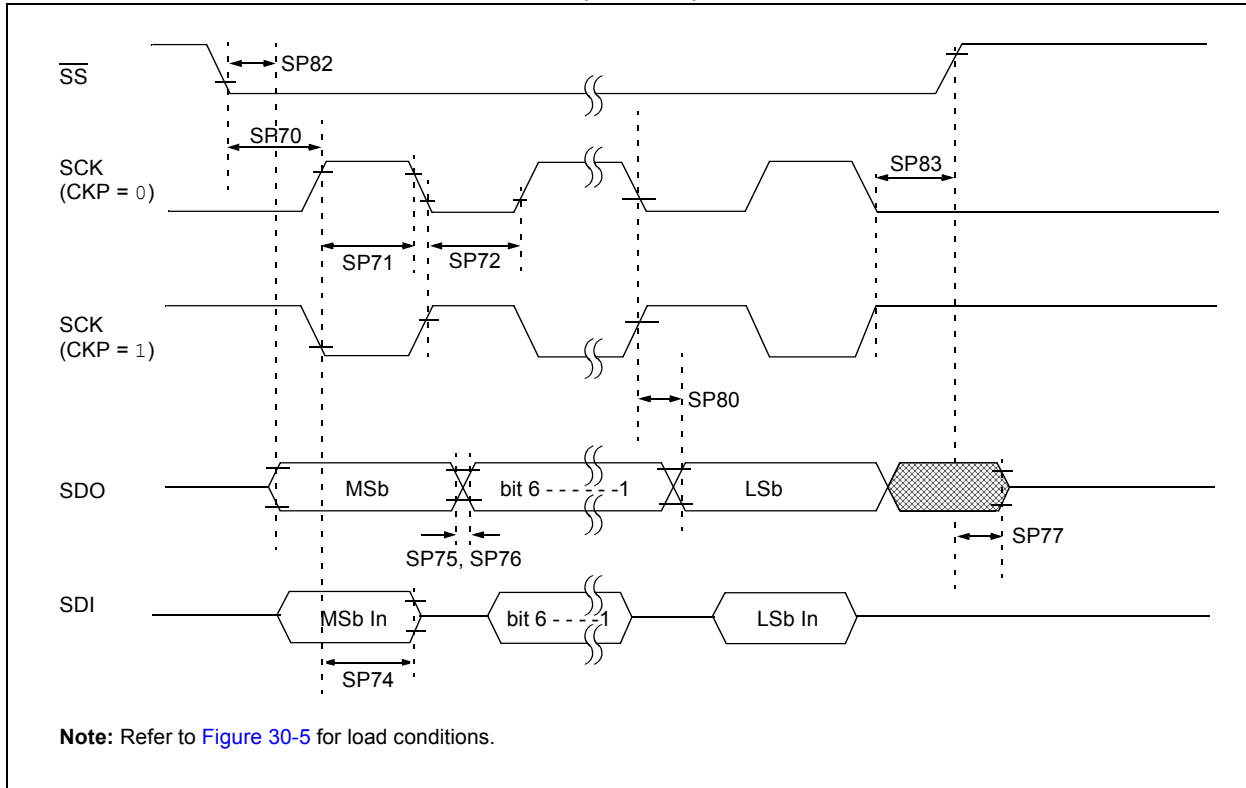
## TABLE 30-15: SPI MODE REQUIREMENTS

| Param No. | Symbol | Characteristic | | Min. | Typ† | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| SP70* | TssL2scH, TssL2scL | $\overline{SS}\downarrow$ to SCK↓ or SCK↑ input | | TCY | — | — | ns | |
| SP71* | TscH | SCK input high time (Slave mode) | | TCY + 20 | — | — | ns | |
| SP72* | TscL | SCK input low time (Slave mode) | | TCY + 20 | — | — | ns | |
| SP73* | TDIV2scH, TDIV2scL | Setup time of SDI data input to SCK edge | | 100 | — | — | ns | |
| SP74* | TscH2DIL, TscL2DIL | Hold time of SDI data input to SCK edge | | 100 | — | — | ns | |
| SP75* | TDOR | SDO data output rise time | 3.0-5.5V | — | 10 | 25 | ns | |
| | | | 1.8-5.5V | — | 25 | 50 | ns | |
| SP76* | TDOF | SDO data output fall time | | — | 10 | 25 | ns | |
| SP77* | TssH2DOZ | $\overline{SS}\uparrow$ to SDO output high-impedance | | 10 | — | 50 | ns | |
| SP78* | TscR | SCK output rise time (Master mode) | 3.0-5.5V | — | 10 | 25 | ns | |
| | | | 1.8-5.5V | — | 25 | 50 | ns | |
| SP79* | TscF | SCK output fall time (Master mode) | | — | 10 | 25 | ns | |
| SP80* | TscH2DOV, TscL2DOV | SDO data output valid after SCK edge | 3.0-5.5V | — | — | 50 | ns | |
| | | | 1.8-5.5V | — | — | 145 | ns | |
| SP81* | TDOV2scH, TDOV2scL | SDO data output setup to SCK edge | | Tcy | — | — | ns | |
| SP82* | TssL2DOV | SDO data output valid after $\overline{SS}\downarrow$ edge | | — | — | 50 | ns | |
| SP83* | TscH2ssH, TscL2ssH | $\overline{SS}\uparrow$ after SCK edge | | 1.5TCY + 40 | — | — | ns | |

\* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

## FIGURE 30-20: I²C™ BUS START/STOP BITS TIMING



**Note**: Refer to Figure 30-5 for load conditions.
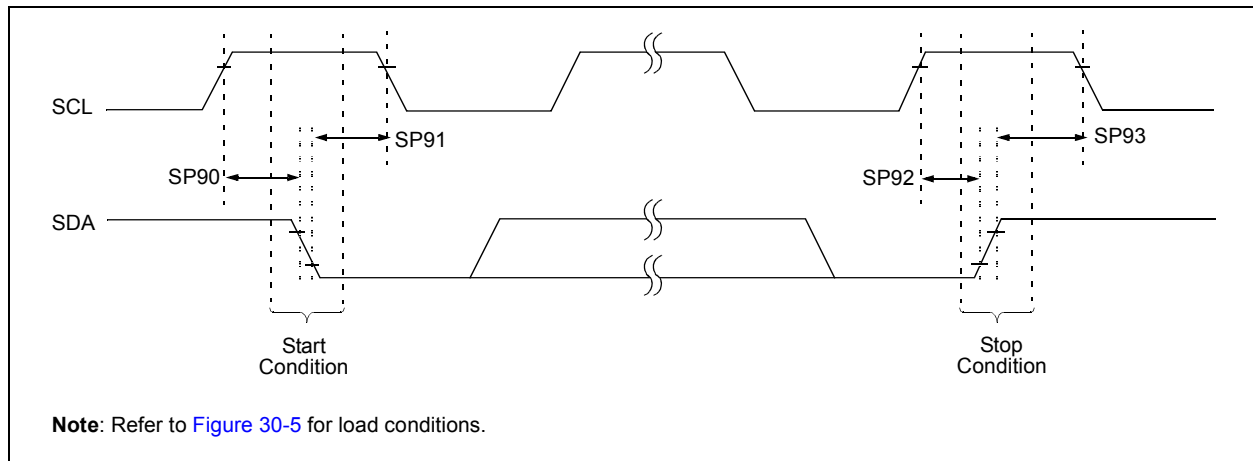
# PIC16(L)F1782/3

**TABLE 30-16: I²C™ BUS START/STOP BITS REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min. | Typ | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|---|
| SP90* | TSU:STA | Start condition Setup time | 100 kHz mode | 4700 | — | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 600 | — | — | | |
| SP91* | THD:STA | Start condition Hold time | 100 kHz mode | 4000 | — | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 600 | — | — | | |
| SP92* | TSU:STO | Stop condition Setup time | 100 kHz mode | 4700 | — | — | ns | |
| | | | 400 kHz mode | 600 | — | — | | |
| SP93 | THD:STO | Stop condition Hold time | 100 kHz mode | 4000 | — | — | ns | |
| | | | 400 kHz mode | 600 | — | — | | |

\*    These parameters are characterized but not tested.

**FIGURE 30-21:     I²C™ BUS DATA TIMING**



**Note:** Refer to Figure 30-5 for load conditions.

**Preliminary**                      © 2011-2012 Microchip Technology Inc.
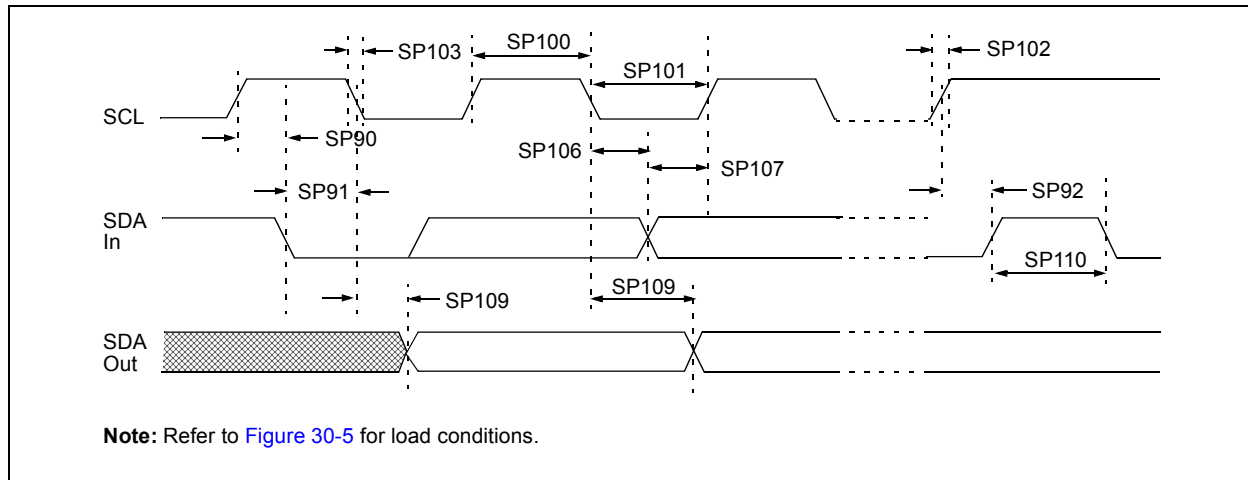
**TABLE 30-17: I²C™ BUS DATA REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| SP100* | T$_{HIGH}$ | Clock high time | 100 kHz mode | 4.0 | — | μs | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 0.6 | — | μs | Device must operate at a minimum of 10 MHz |
| | | | SSP module | 1.5T$_{CY}$ | — | | |
| SP101* | T$_{LOW}$ | Clock low time | 100 kHz mode | 4.7 | — | μs | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 1.3 | — | μs | Device must operate at a minimum of 10 MHz |
| | | | SSP module | 1.5T$_{CY}$ | — | | |
| SP102* | T$_R$ | SDA and SCL rise time | 100 kHz mode | — | 1000 | ns | |
| | | | 400 kHz mode | 20 + 0.1C$_B$ | 300 | ns | C$_B$ is specified to be from 10-400 pF |
| SP103* | T$_F$ | SDA and SCL fall time | 100 kHz mode | — | 250 | ns | |
| | | | 400 kHz mode | 20 + 0.1C$_B$ | 250 | ns | C$_B$ is specified to be from 10-400 pF |
| SP106* | T$_{HD:DAT}$ | Data input hold time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | μs | |
| SP107* | T$_{SU:DAT}$ | Data input setup time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| SP109* | T$_{AA}$ | Output valid from clock | 100 kHz mode | — | 3500 | ns | **(Note 1)** |
| | | | 400 kHz mode | — | — | ns | |
| SP110* | T$_{BUF}$ | 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 | |
| SP111 | C$_B$ | Bus capacitive loading | | — | 400 | pF | |

\*   These parameters are characterized but not tested.

**Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

    **2:** A Fast mode (400 kHz) I²C™ bus device can be used in a Standard mode (100 kHz) I²C bus system, but the requirement T$_{SU:DAT}$ ≥ 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 T$_R$ max. + T$_{SU:DAT}$ = 1000 + 250 = 1250 ns (according to the Standard mode I²C bus specification), before the SCL line is released.

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 31.0 DC AND AC CHARACTERISTICS GRAPHS AND CHARTS

The graphs and tables provided in this section are for **design guidance** and are **not tested**.

In some graphs or tables, the data presented are **outside specified operating range** (i.e., outside specified V$_{DD}$ range). This is for **information only** and devices are ensured to operate properly only within the specified range.

| | |
|---|---|
| **Note:** | The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range. |

**"Typical" represents the mean of the distribution at 25°C. "Maximum", "Max.", "Minimum" or "Min." represents (mean + 3σ) or (mean - 3σ) respectively, where σ is a standard deviation, over each temperature range.**

# PIC16(L)F1782/3

**FIGURE 31-1:** **I**$_{PD}$ **BASE, PIC16LF1782/3 ONLY**
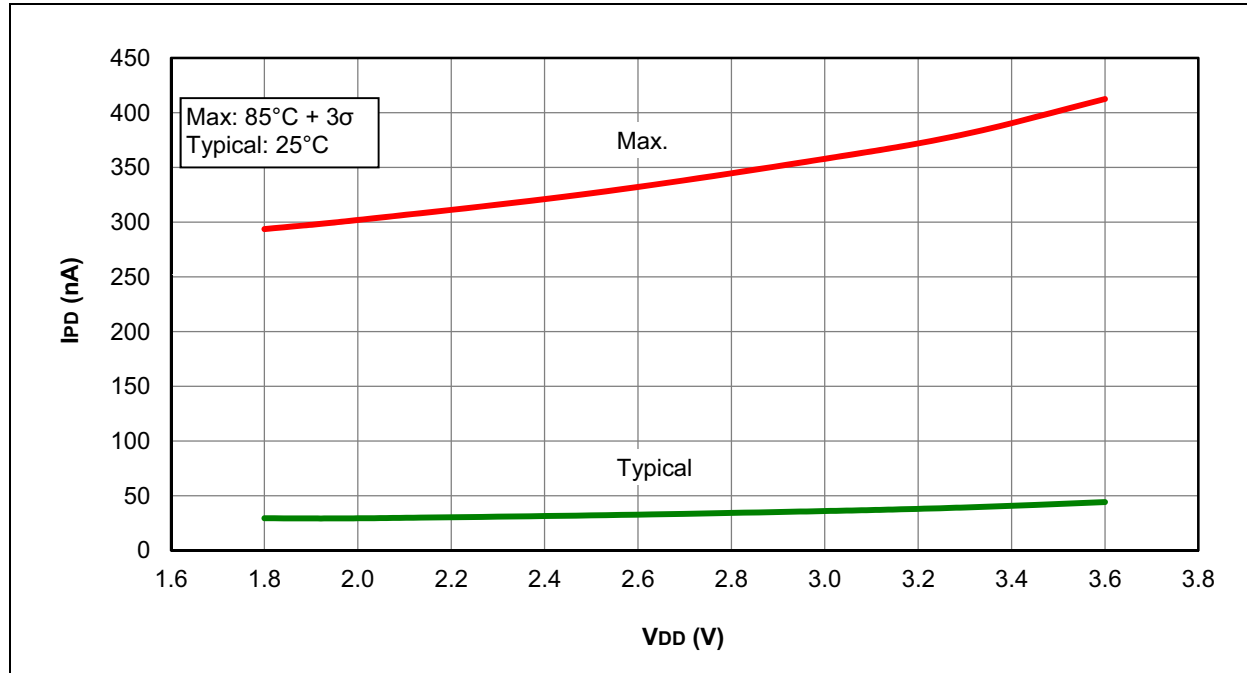


**FIGURE 31-2:** **I**$_{PD}$ **BASE, LOW-POWER SLEEP MODE (VREGPM = 1), PIC16F1782/3 ONLY**



**Preliminary** © 2011-2012 Microchip Technology Inc.
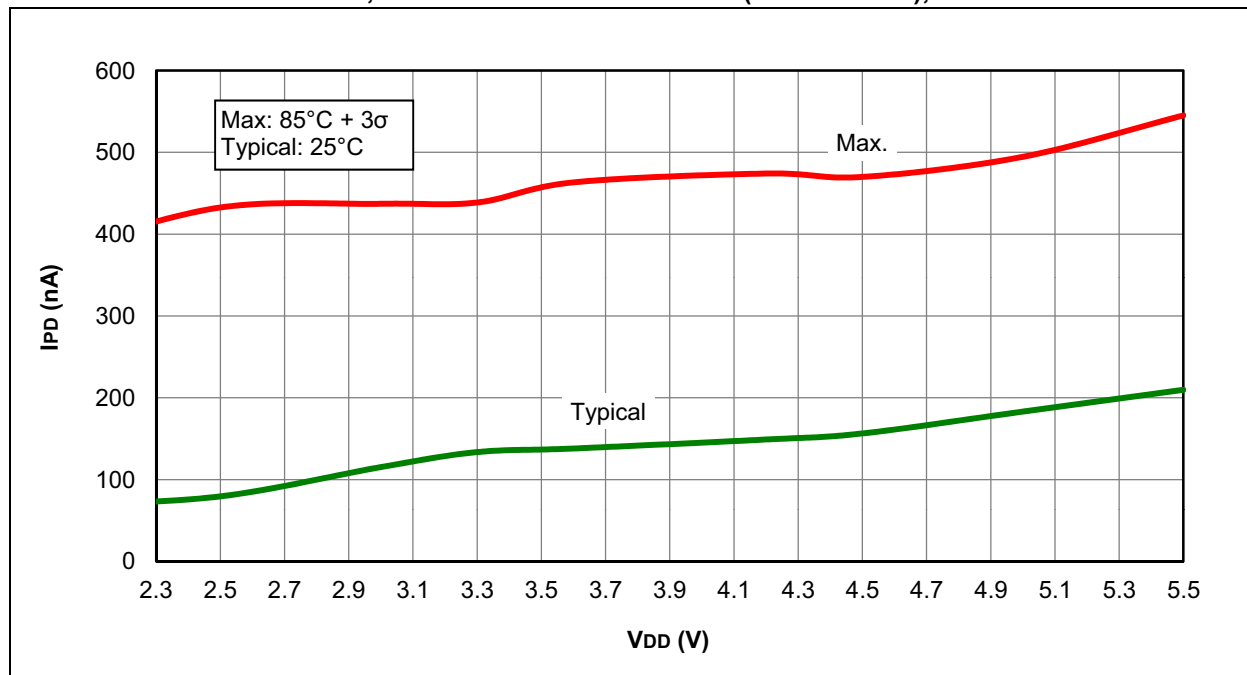
**FIGURE 31-3:** **I**<sub>PD</sub>**, WATCH DOG TIMER (WDT), PIC16LF1782/3 ONLY**
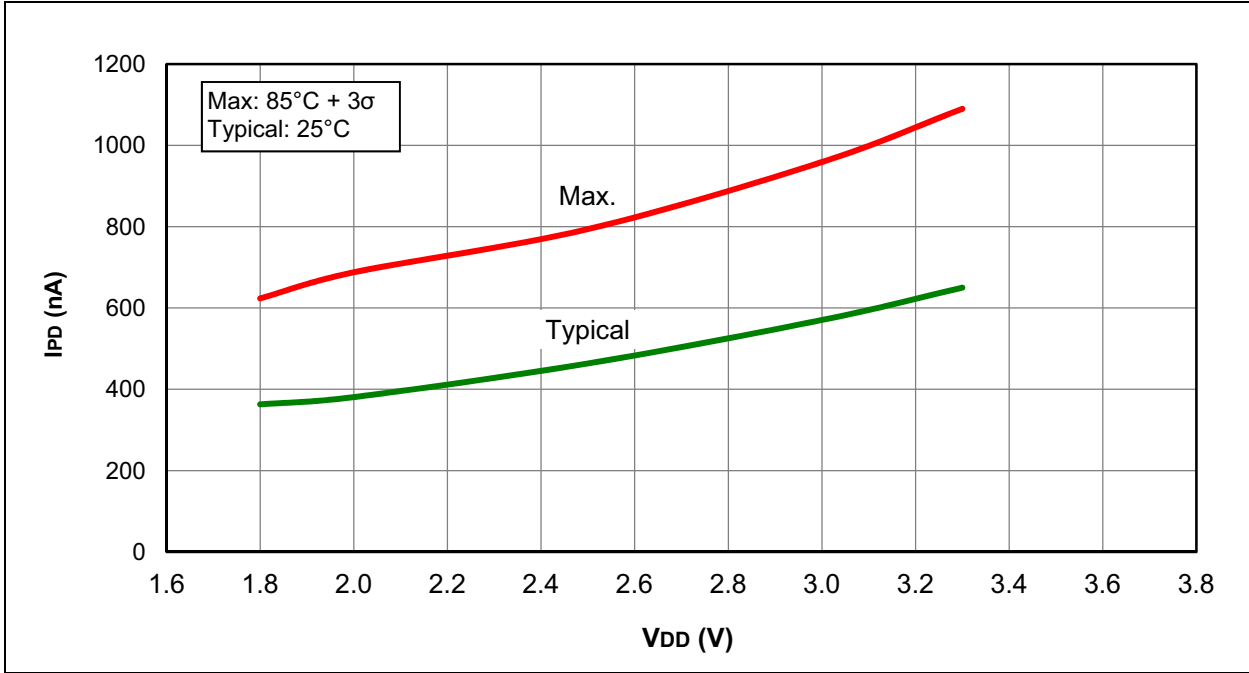


**FIGURE 31-4:** **I**<sub>PD</sub>**, WATCH DOG TIMER (WDT), PIC16F1782/3 ONLY**

# PIC16(L)F1782/3

**Preliminary** © 2011-2012 Microchip Technology Inc.
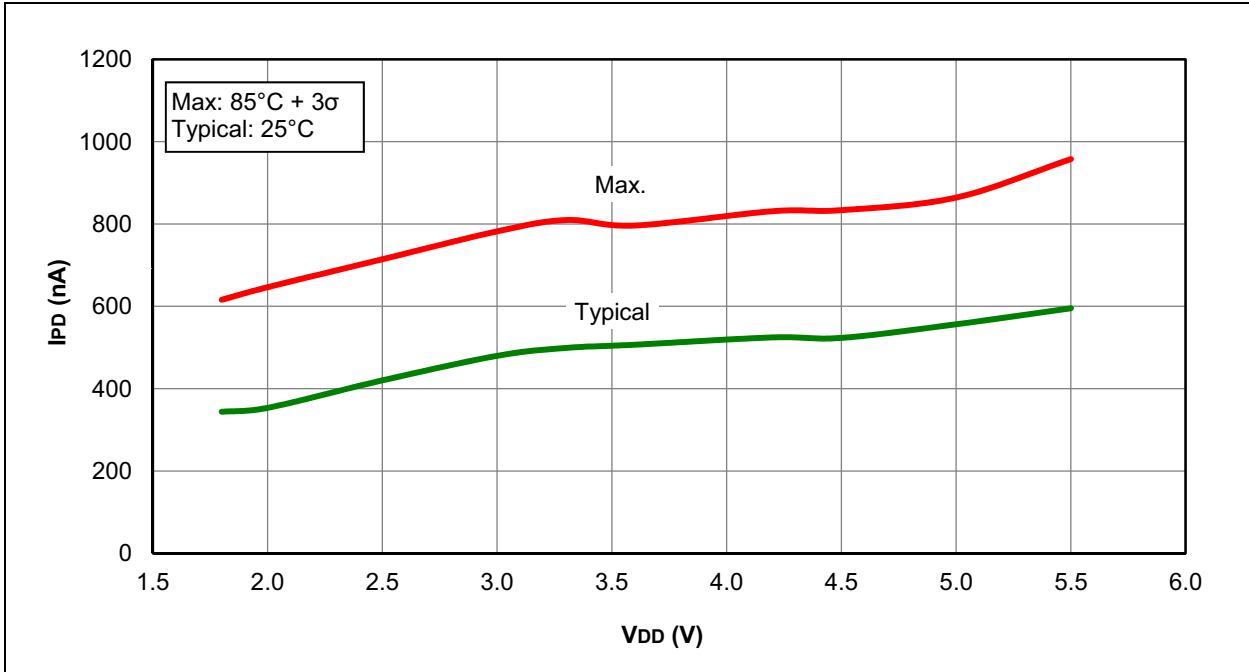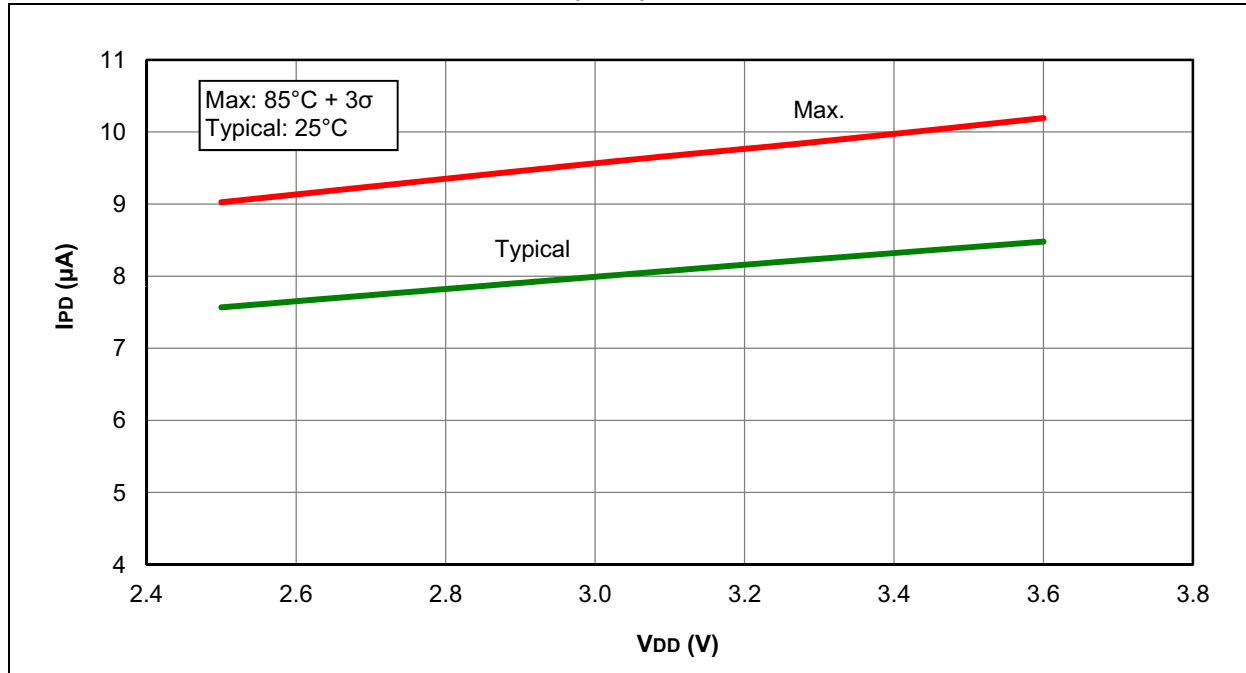
**FIGURE 31-7:** **I**PD, **LOW-POWER BROWN-OUT RESET (LPBOR), PIC16LF1782/3 ONLY**
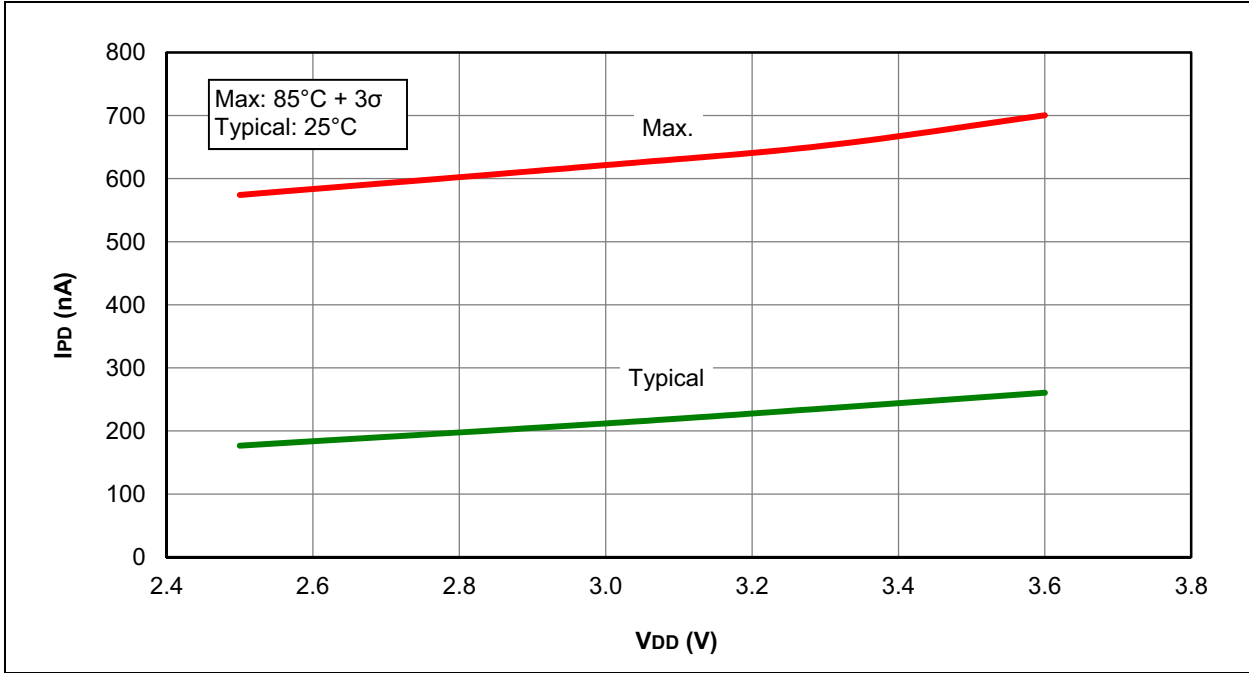


**FIGURE 31-8:** **I**PD, **LOW-POWER BROWN-OUT RESET (LPBOR), PIC16F1782/3 ONLY**
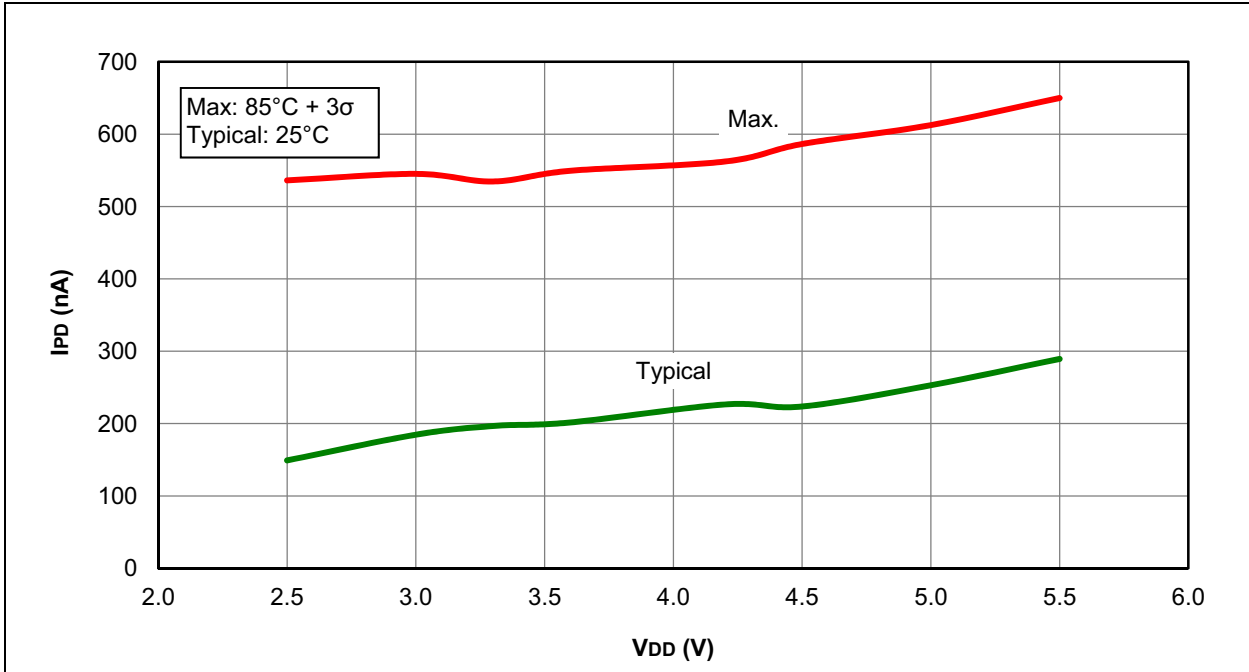
# PIC16(L)F1782/3

**FIGURE 31-9:** IPD, TIMER1 OSCILLATOR, FOSC = 32 kHz, PIC16LF1782/3 ONLY
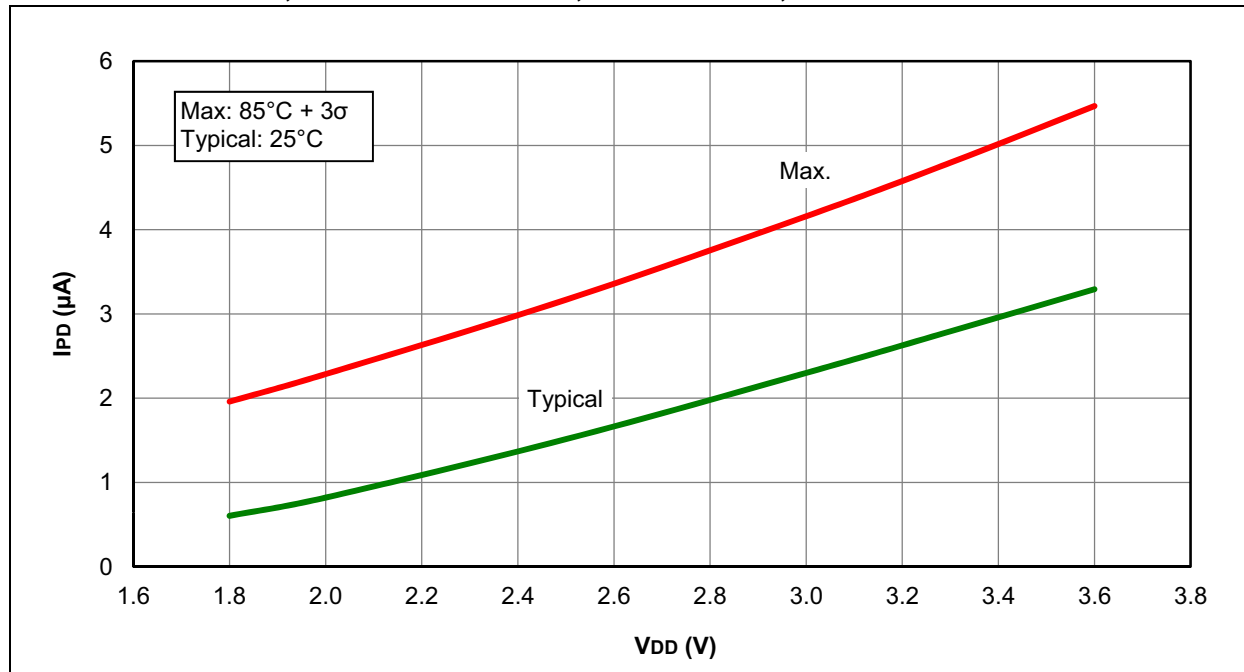


**FIGURE 31-10:** IPD, TIMER1 OSCILLATOR, FSC = 32 kHz, PIC16F1782/3 ONLY



**Preliminary**     © 2011-2012 Microchip Technology Inc.
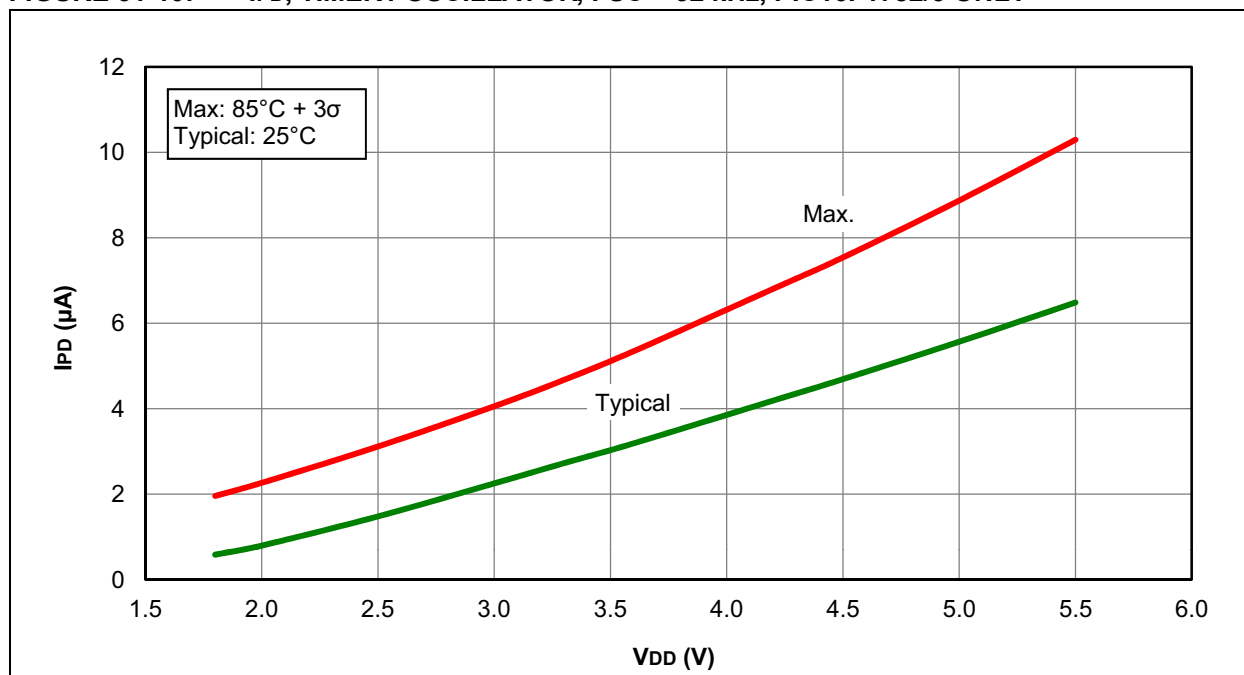
# PIC16(L)F1782/3

**FIGURE 31-11:** **IPD, VOH vs. IOH OVER TEMPERATURE, VDD = 5.0V, PIC16F1782/3 ONLY**
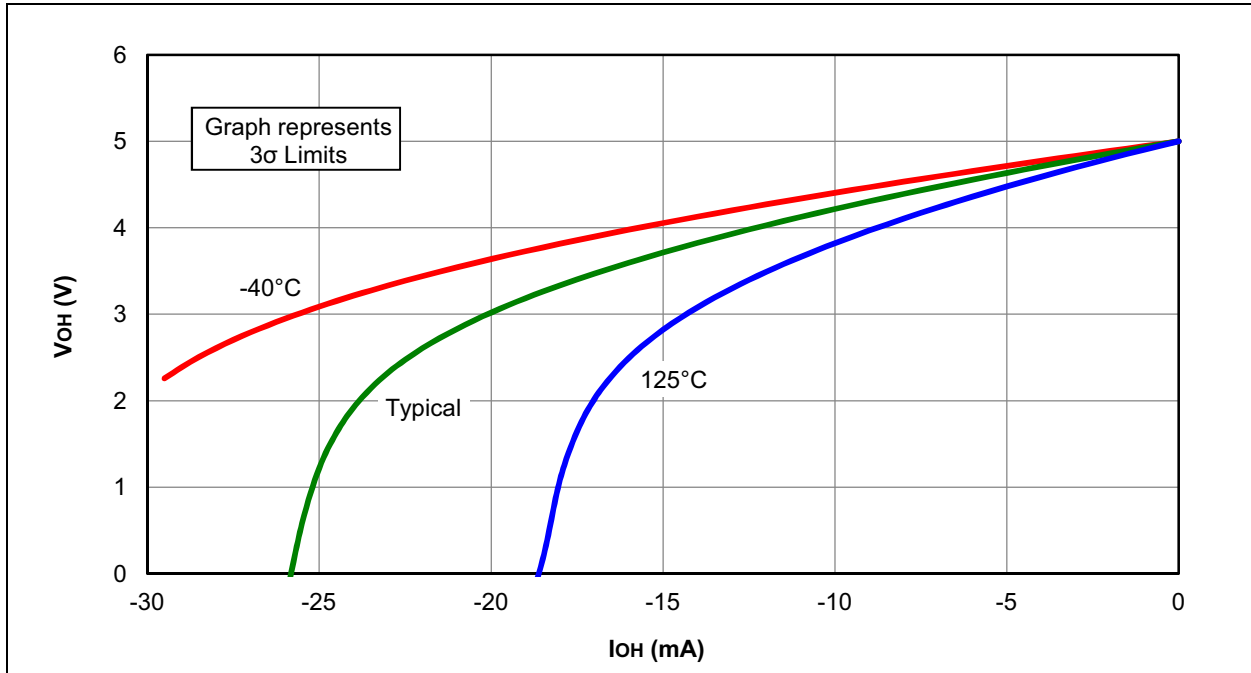


**FIGURE 31-12:** **IPD, VOL vs. IOL OVER TEMPERATURE, VDD = 5.0V, PIC16F1782/3 ONLY**
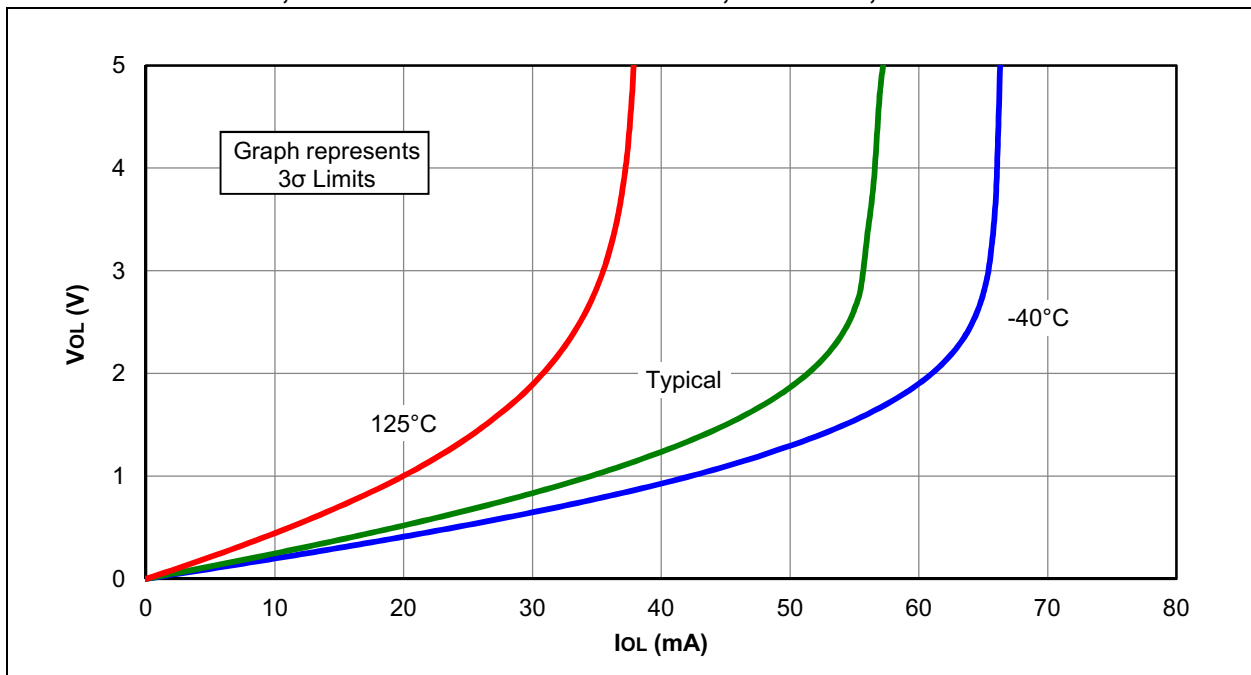
# PIC16(L)F1782/3

**FIGURE 31-13:** **I**PD, **V**OH **vs. I**OH **OVER TEMPERATURE, V**DD **= 3.0V**
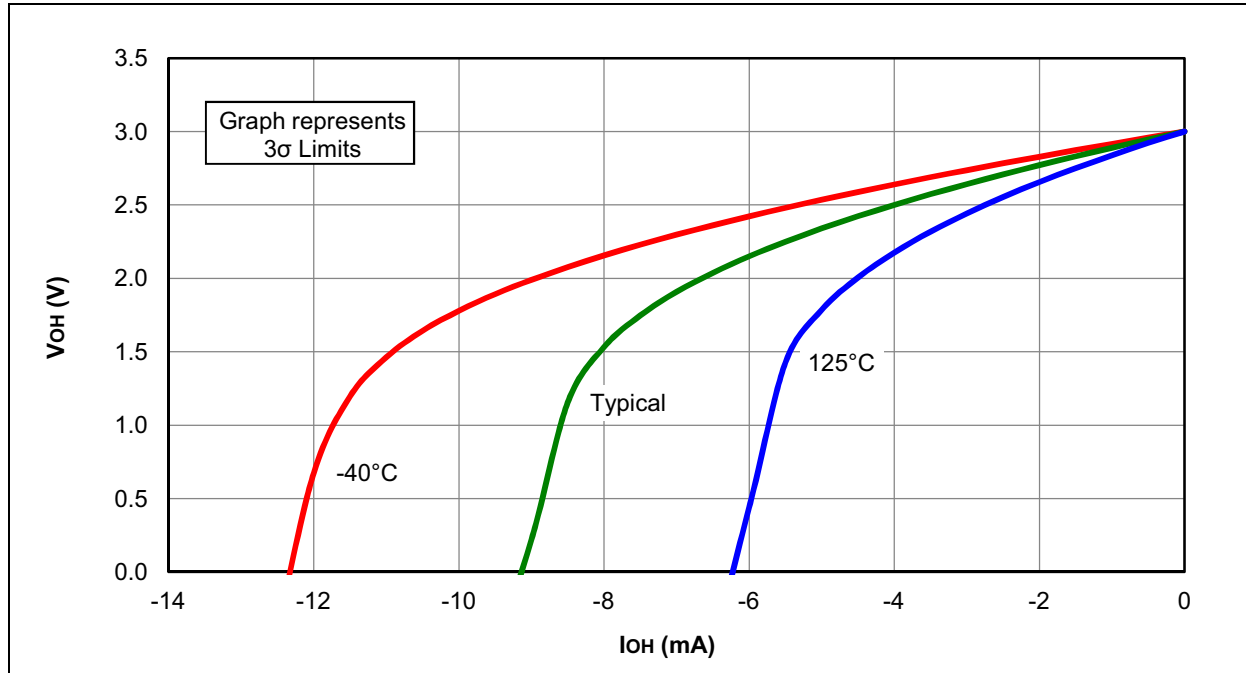


**FIGURE 31-14:** **I**PD, **V**OL **vs. I**OL **OVER TEMPERATURE, V**DD **= 3.0V**

**Preliminary**                    © 2011-2012 Microchip Technology Inc.
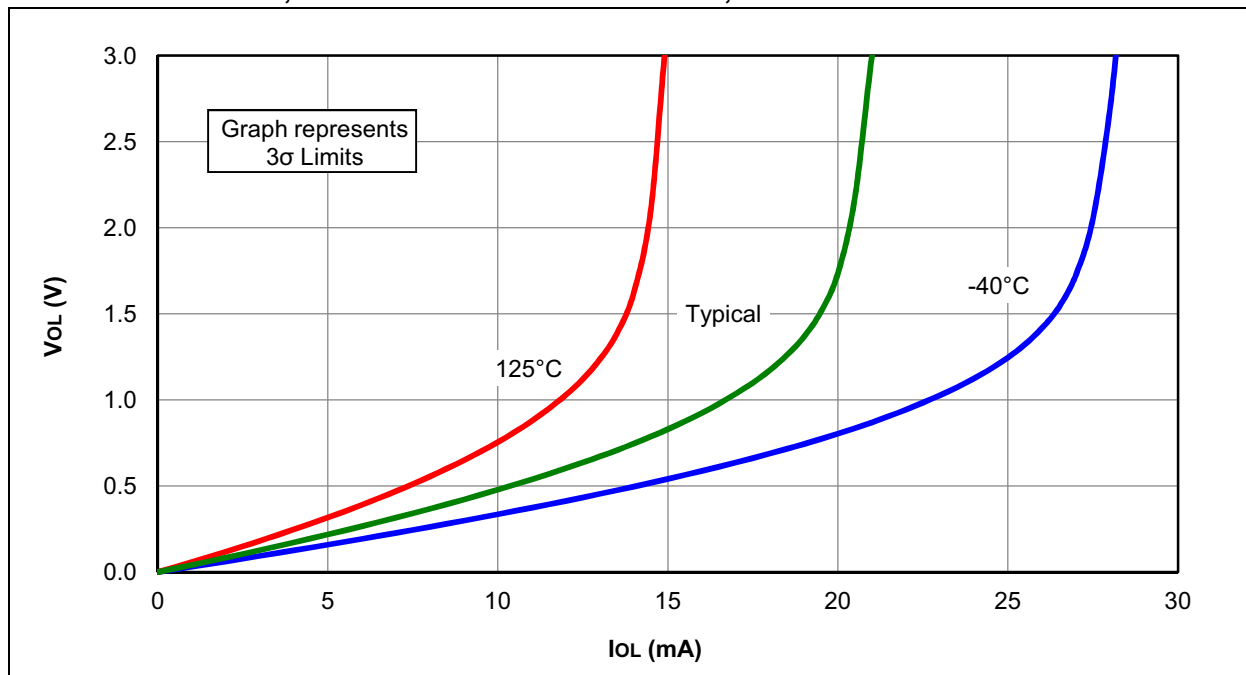
**FIGURE 31-15:** IPD, VOL vs. IOL OVER TEMPERATURE, VDD = 1.8V, PIC16LF1782/3 ONLY
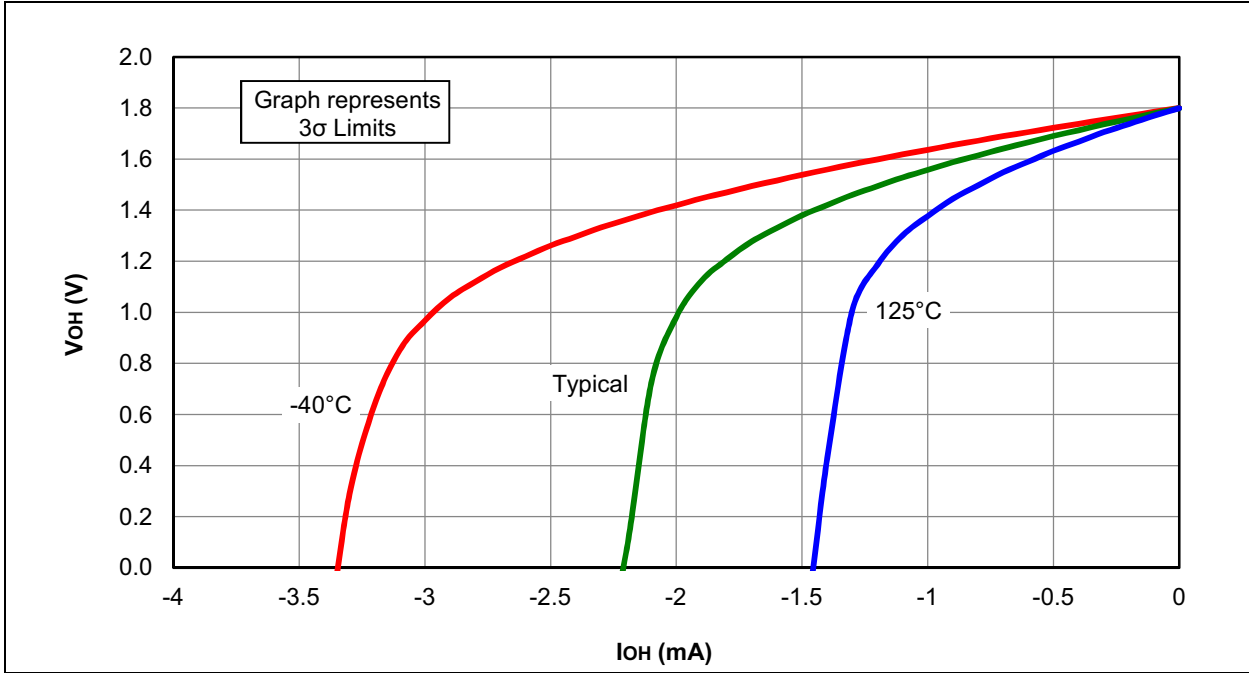


**FIGURE 31-16:** IPD, VOL vs. IOL OVER TEMPERATURE, VDD = 1.8V, PIC16LF1782/3 ONLY
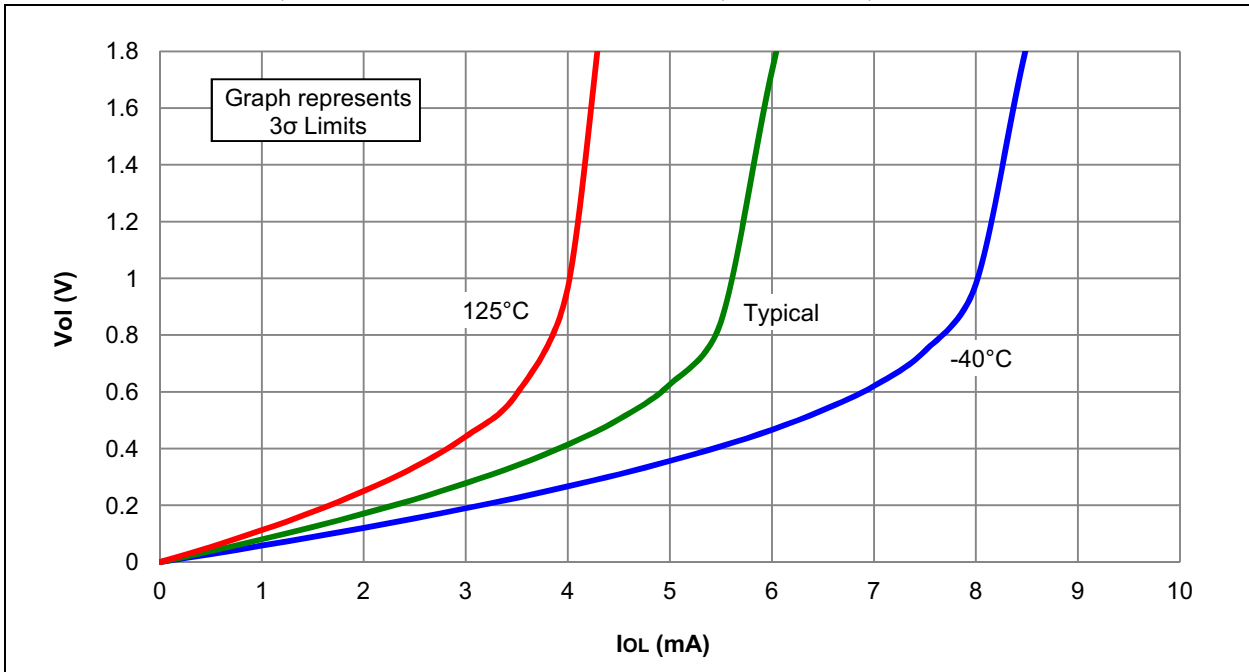
# PIC16(L)F1782/3

**FIGURE 31-17:** POR RELEASE VOLTAGE
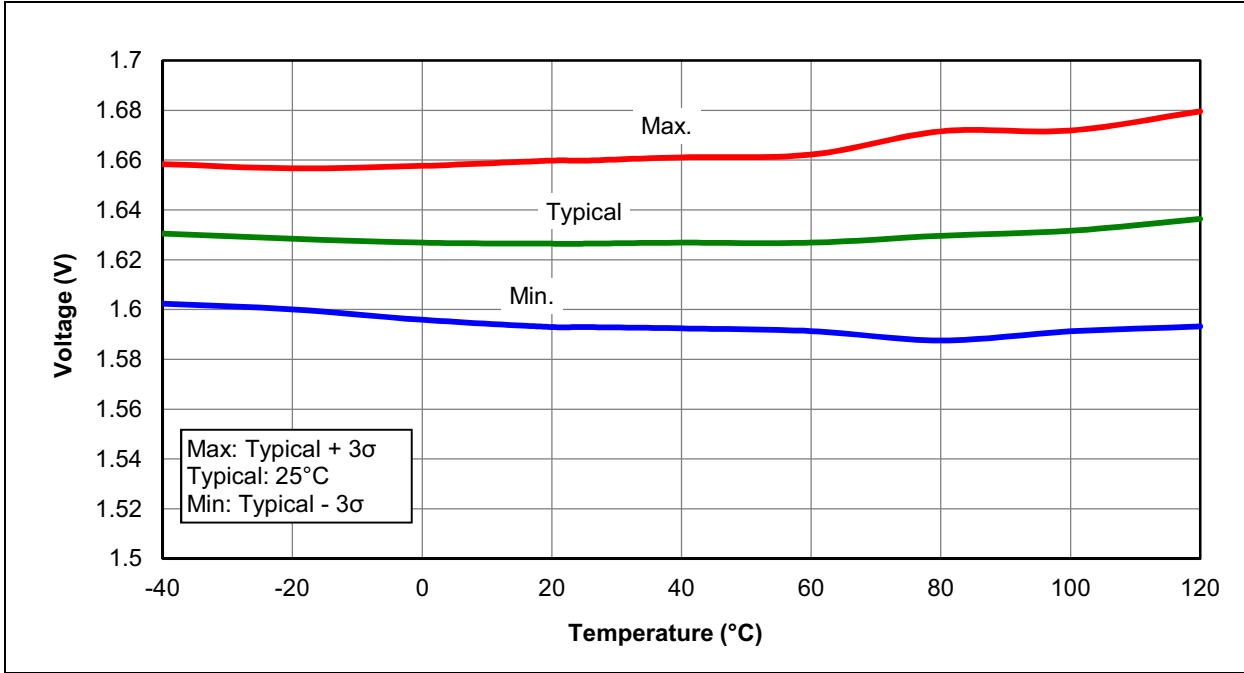


**FIGURE 31-18:** POR REARM VOLTAGE, NORMAL POWER MODE (VREGPM1 = 0),
PIC16F1782/3 ONLY
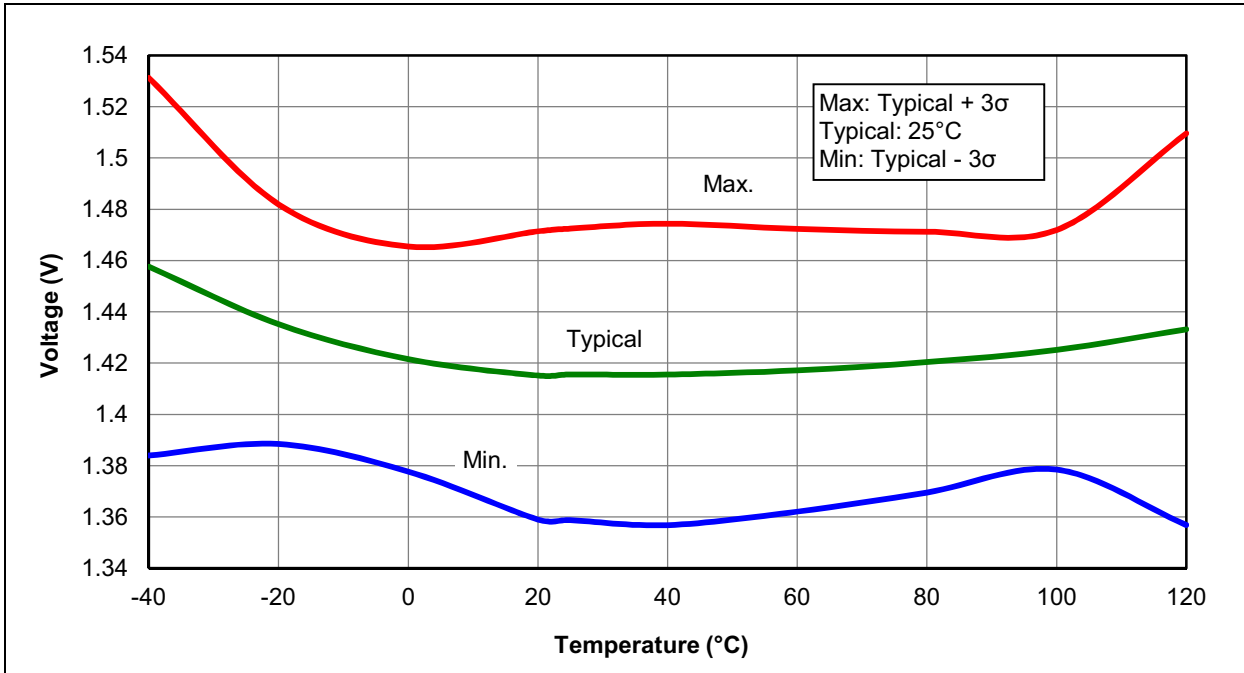


**Preliminary**   © 2011-2012 Microchip Technology Inc.
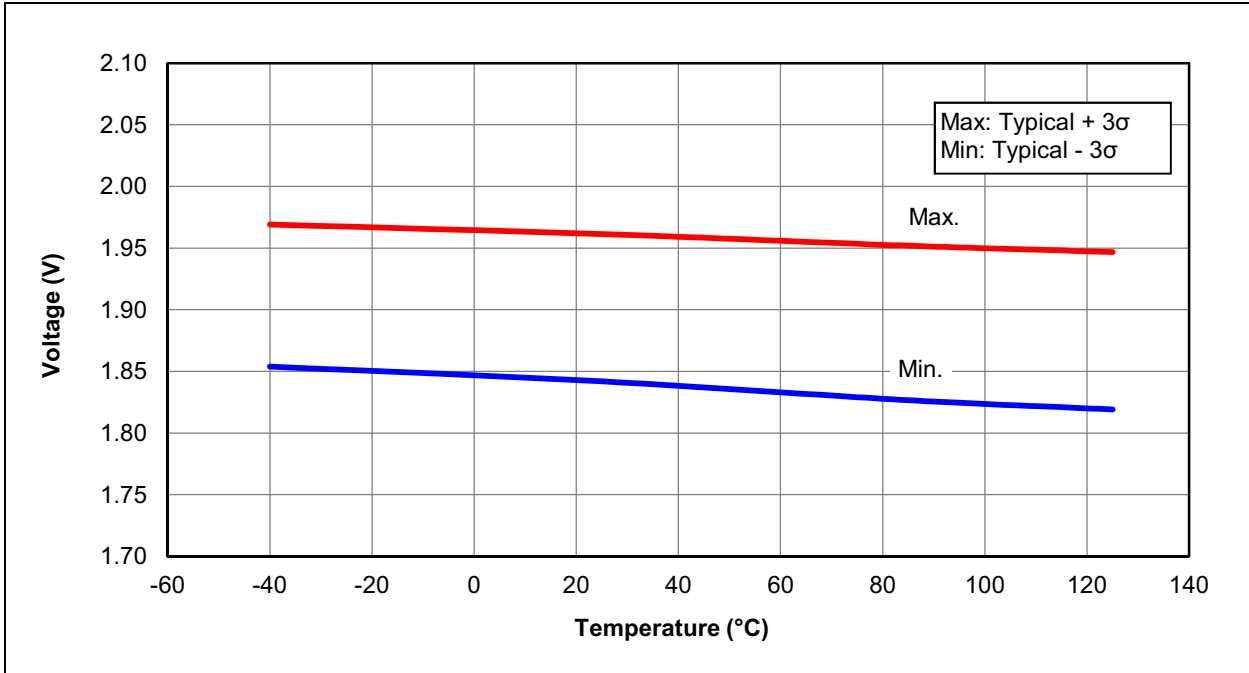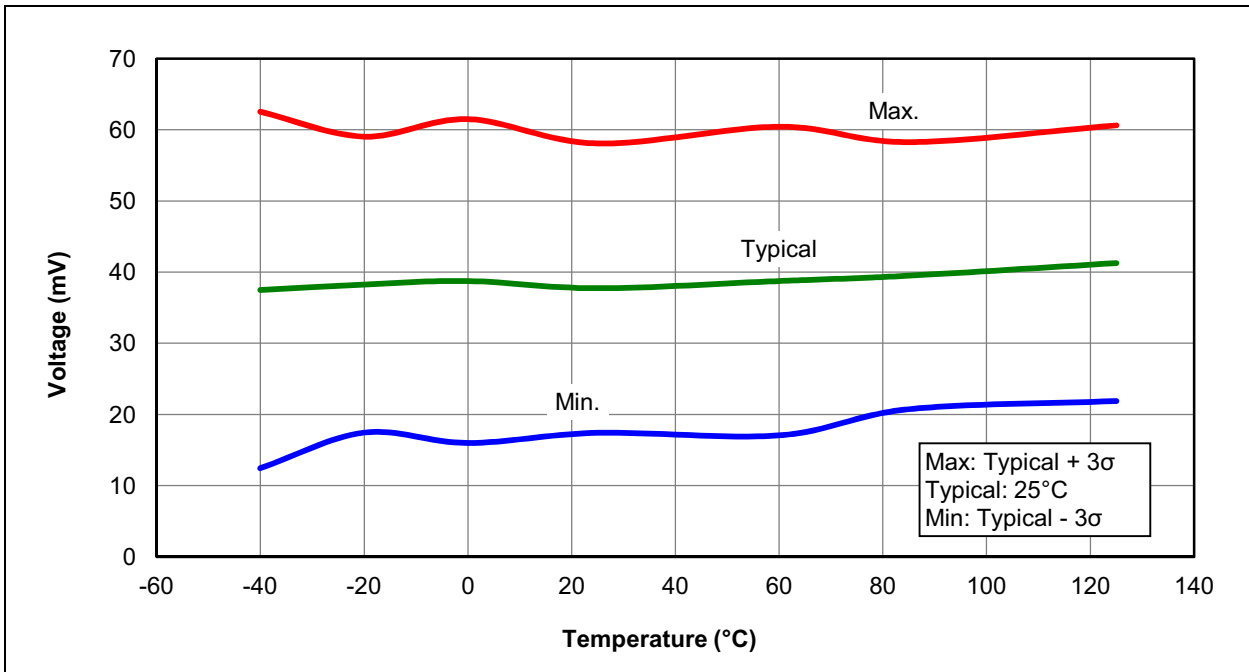
# PIC16(L)F1782/3

# PIC16(L)F1782/3

**FIGURE 31-21:** **BROWN-OUT RESET VOLTAGE, HIGH TRIP POINT (BORV = 0)**

FIGURE 31-21: BROWN-OUT RESET VOLTAGE, HIGH TRIP POINT (BORV = 0)
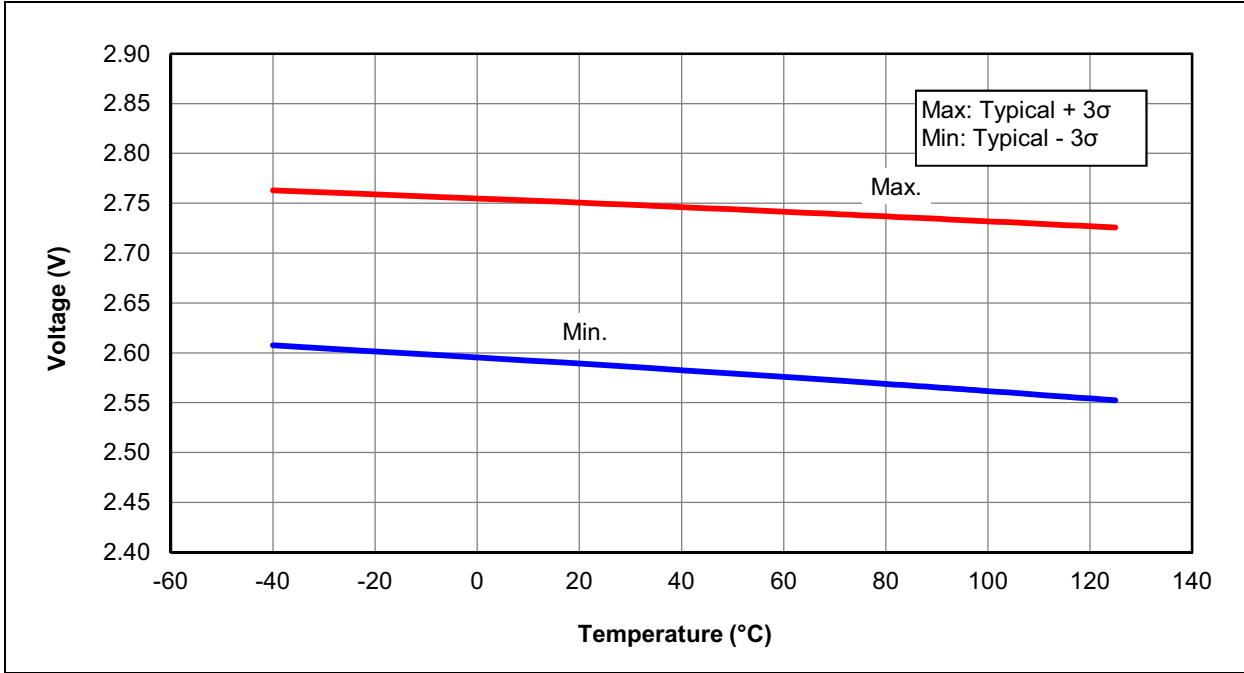


FIGURE 31-22: BROWN-OUT RESET HYSTERESIS, HIGH TRIP POINT (BORV = 0)

**Preliminary**
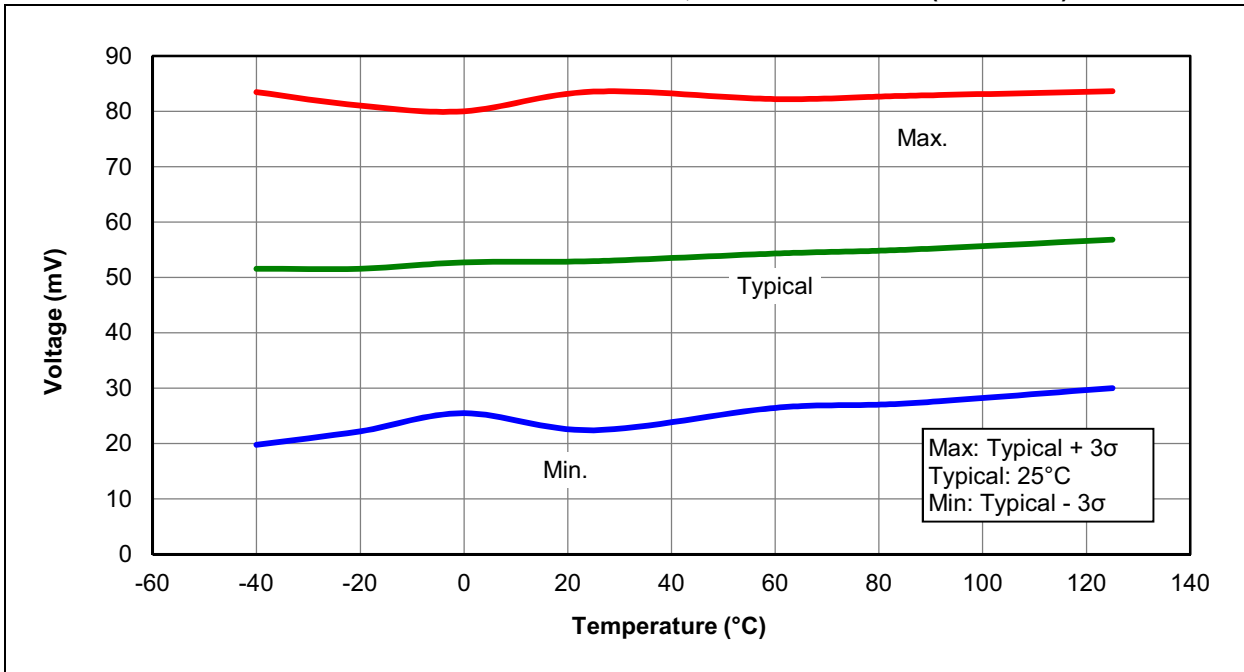
**FIGURE 31-23:** **LPBOR RESET VOLTAGE**
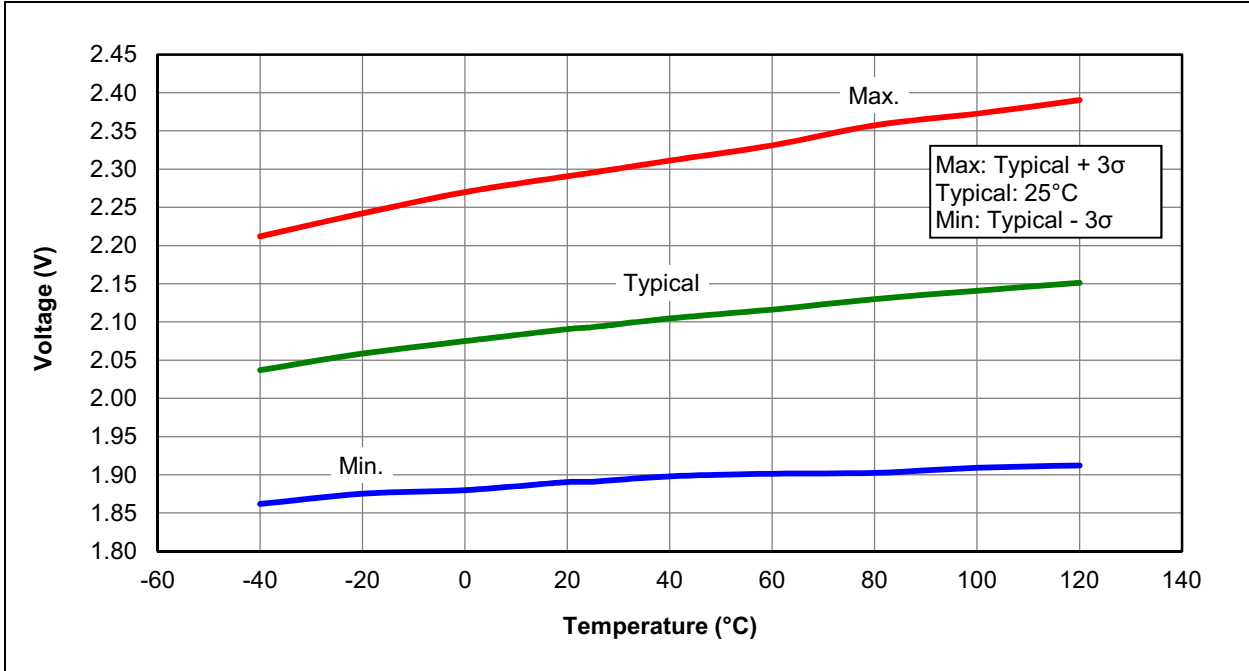


**FIGURE 31-24:** **LPBOR RESET HYSTERESIS**
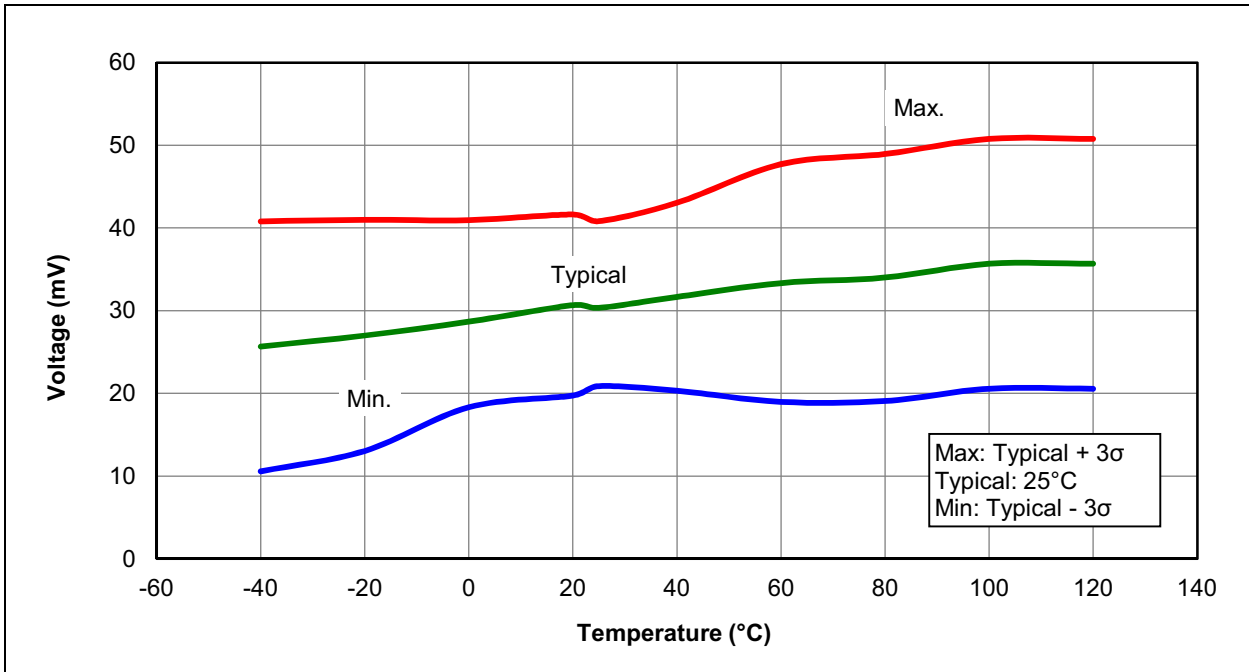
# PIC16(L)F1782/3

**FIGURE 31-25:** WDT TIME-OUT PERIOD
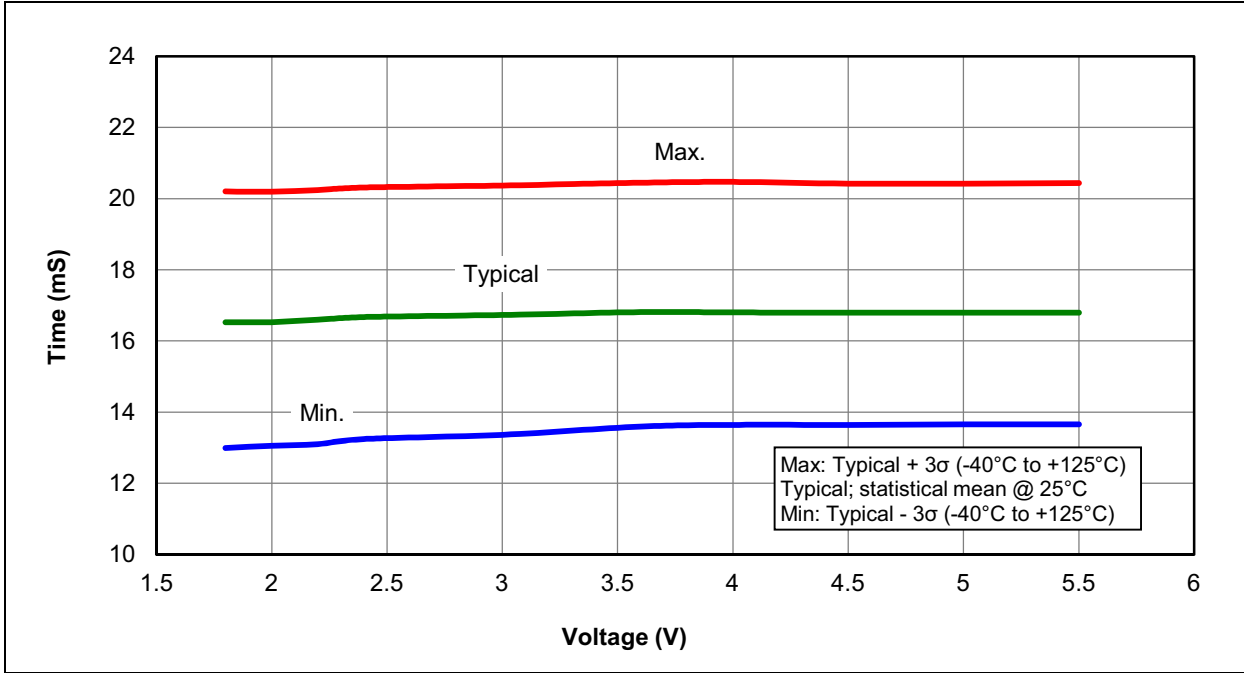


**FIGURE 31-26:** PWRT PERIOD



**Preliminary**
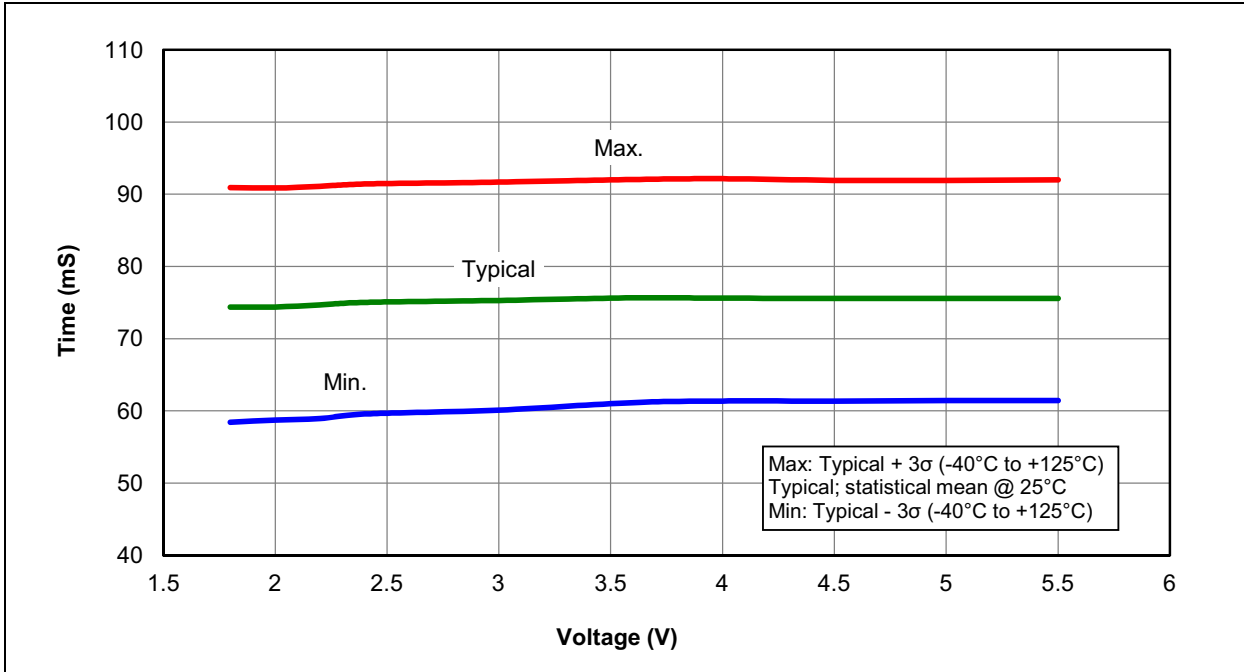
# PIC16(L)F1782/3

**FIGURE 31-27:** ADC 12-BIT MODE, SINGE-ENDED DNL, V$_{DD}$ = 3.0V, TAD = 4$\mu$S, 25°C
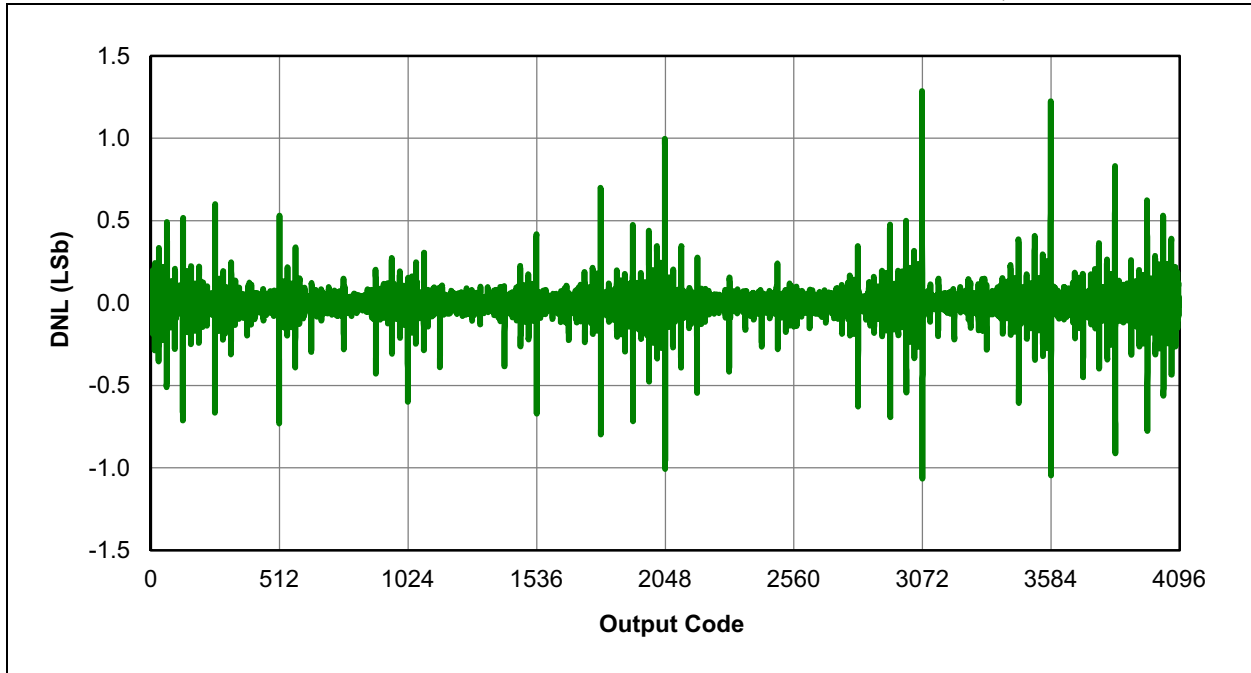


**FIGURE 31-28:** ADC 12-BIT MODE, SINGE-ENDED DNL, V$_{DD}$ = 5.5V, TAD = 4$\mu$S, 25°C



© 2011-2012 Microchip Technology Inc. **Preliminary** DS41579C-page 411
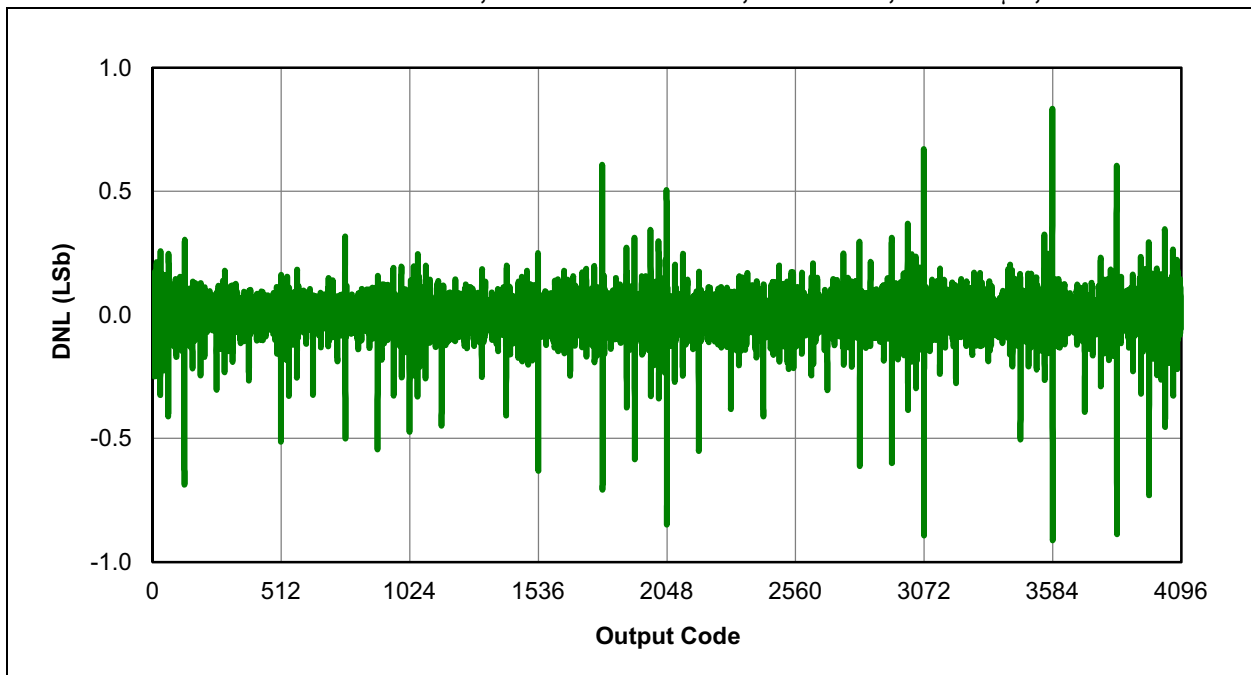
# PIC16(L)F1782/3

**FIGURE 31-29:** COMPARATOR HYSTERESIS, NORMAL SPEED MODE (CxSP = 1), V<sub>DD</sub> = 1.8V, TYPICAL MEASURED VALUES, PIC16LF1782/3 ONLY
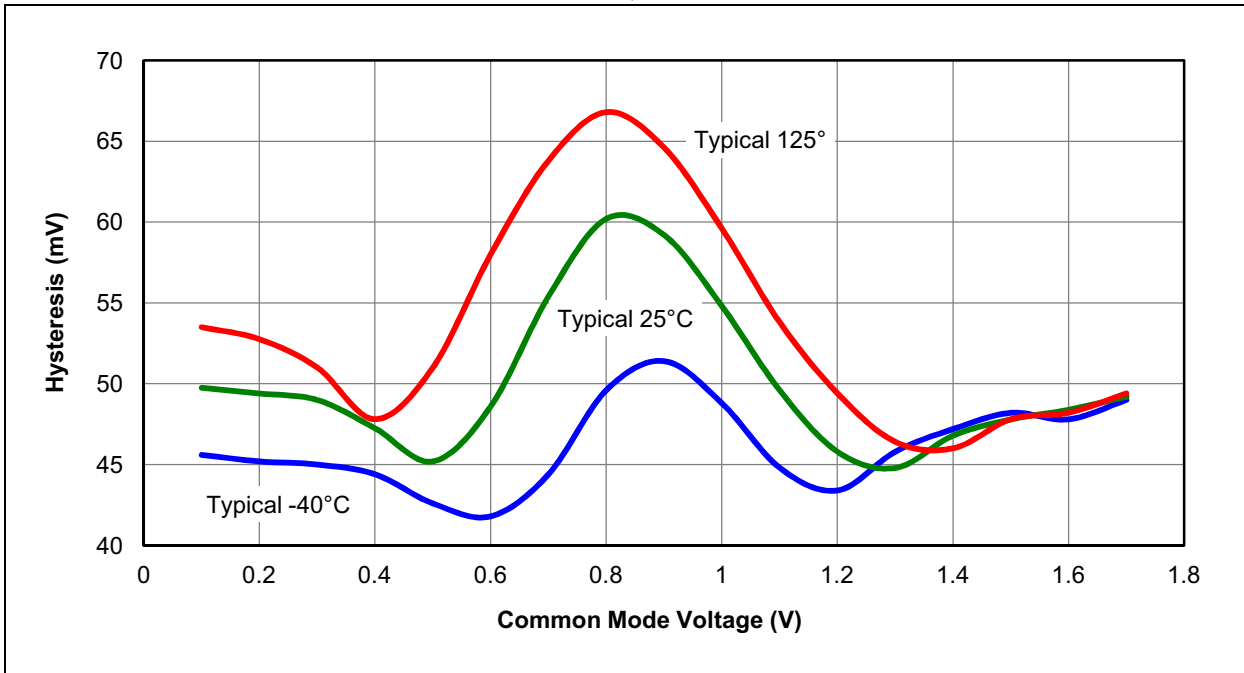


**FIGURE 31-30:** COMPARATOR HYSTERESIS, NORMAL SPEED MODE (CxSP = 1), V<sub>DD</sub> = 3.6V, TYPICAL MEASURED VALUES

**Preliminary**

© 2011-2012 Microchip Technology Inc.
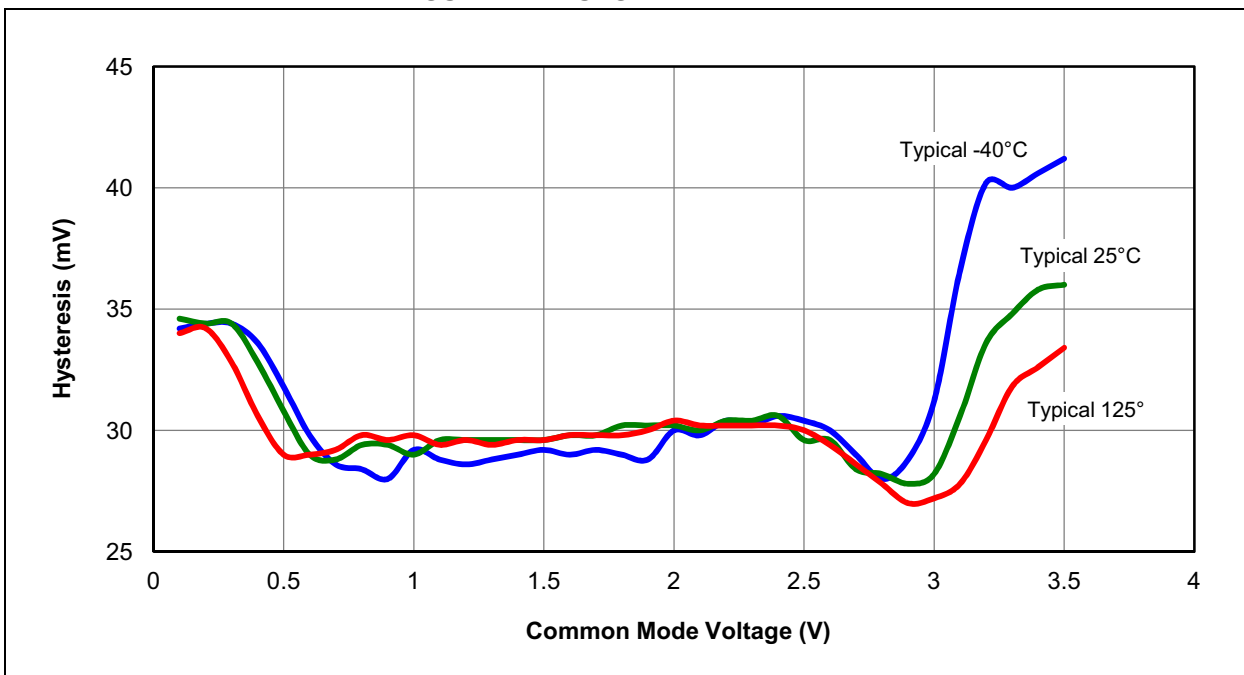
# PIC16(L)F1782/3

**FIGURE 31-31:** COMPARATOR HYSTERESIS, NORMAL SPEED MODE (CxSP = 1), V<sub>DD</sub> = 5.5V, TYPICAL MEASURED VALUES, PIC16F1782/3 ONLY
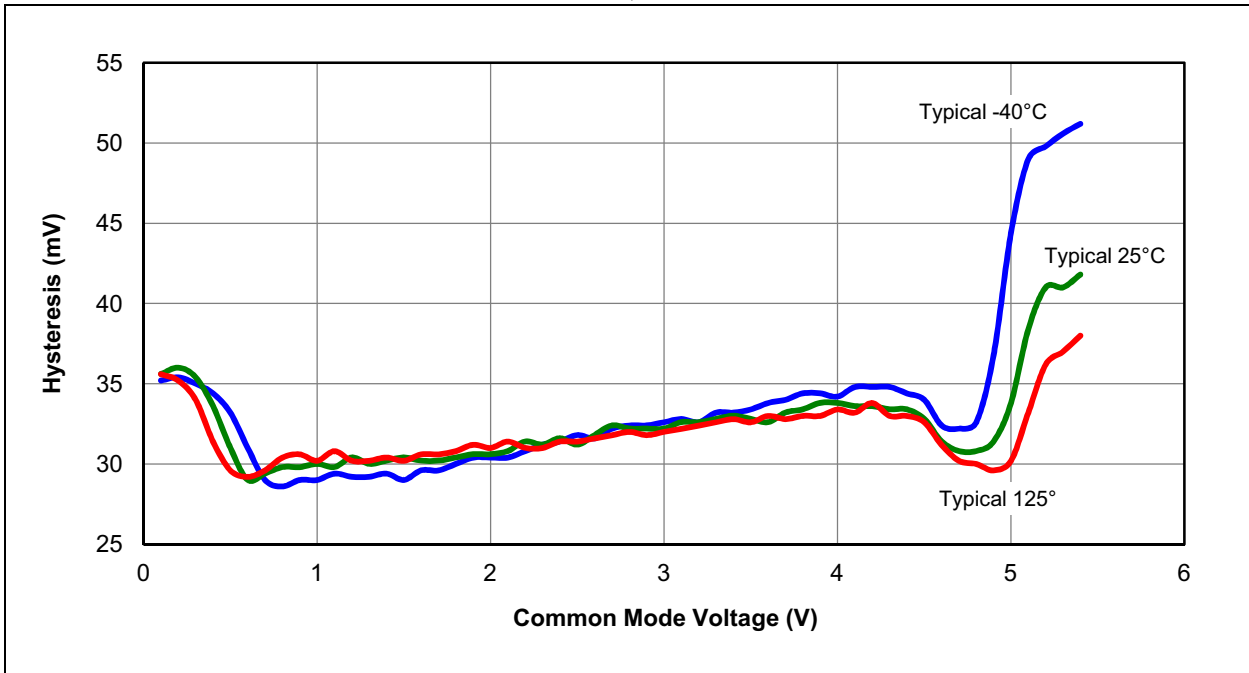


**FIGURE 31-32:** COMPARATOR RESPONSE TIME OVER TEMPERATURE, NORMAL POWER MODE (CxSP = 1), TYPICAL MEASURED VALUES

© 2011-2012 Microchip Technology Inc. **Preliminary** DS41579C-page 413

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.
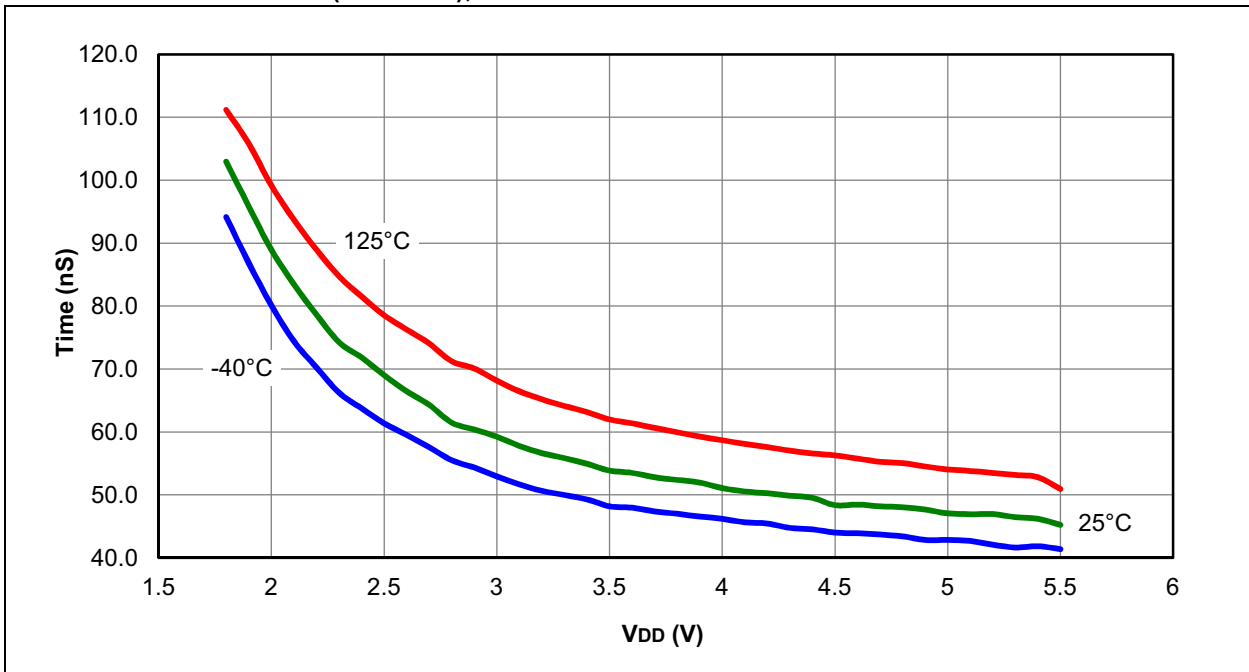
## 32.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers and dsPIC® digital signal controllers are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB C Compiler for Various Device Families
  - HI-TECH C® for Various Device Families
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- Device Programmers
  - PICkit™ 2 Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits, and Starter Kits

## 32.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - In-Circuit Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either C or assembly)
- One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (C or assembly)
  - Mixed C and assembly
  - Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

## 32.2 MPLAB C Compilers for Various Device Families

The MPLAB C Compiler code development systems are complete ANSI C compilers for Microchip's PIC18, PIC24 and PIC32 families of microcontrollers and the dsPIC30 and dsPIC33 families of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

## 32.3 HI-TECH C for Various Device Families

The HI-TECH C Compiler code development systems are complete ANSI C compilers for Microchip's PIC family of microcontrollers and the dsPIC family of digital signal controllers. These compilers provide powerful integration capabilities, omniscient code generation and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

The compilers include a macro assembler, linker, preprocessor, and one-step driver, and can run on multiple platforms.

## 32.4 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 IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

## 32.5 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

## 32.6 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

**Preliminary**

## 32.7 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C 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.

## 32.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs PIC® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The 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 (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 32.9 MPLAB ICD 3 In-Circuit Debugger System

MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost effective high-speed hardware debugger/programmer for Microchip Flash Digital Signal Controller (DSC) and microcontroller (MCU) devices. It debugs and programs PIC® Flash microcontrollers and dsPIC® DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (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.

## 32.10 PICkit 3 In-Circuit Debugger/ Programmer and PICkit 3 Debug Express

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 Integrated Development Environment (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 an 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™.

The PICkit 3 Debug Express include the PICkit 3, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

# PIC16(L)F1782/3

## 32.11 PICkit 2 Development Programmer/Debugger and PICkit 2 Debug Express

The PICkit™ 2 Development Programmer/Debugger is a low-cost development tool with an easy to use interface for programming and debugging Microchip's Flash families of microcontrollers. The full featured Windows® programming interface supports baseline (PIC10F, PIC12F5xx, PIC16F5xx), midrange (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30, dsPIC33, and PIC32 families of 8-bit, 16-bit, and 32-bit microcontrollers, and many Microchip Serial EEPROM products. With Microchip's powerful MPLAB Integrated Development Environment (IDE) the PICkit™ 2 enables in-circuit debugging on most PIC® microcontrollers. In-Circuit-Debugging runs, halts and single steps the program while the PIC microcontroller is embedded in the application. When halted at a breakpoint, the file registers can be examined and modified.

The PICkit 2 Debug Express include the PICkit 2, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

## 32.12 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at V$_{DDMIN}$ and V$_{DDMAX}$ for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages and a modular, detachable socket assembly to support various package types. The ICSP™ cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices and incorporates an MMC card for file storage and data applications.

## 32.13 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

**Preliminary**

## 33.0   PACKAGING INFORMATION

### 33.1   Package Marking Information

28-Lead SPDIP (.300")

Example

```
XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX
      YYWWNNN
```

```
PIC16F1782
-I/SP  e3
      1204017
```
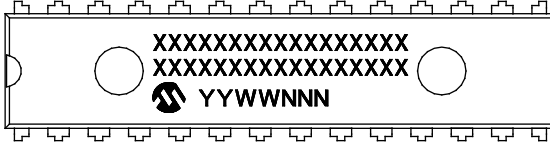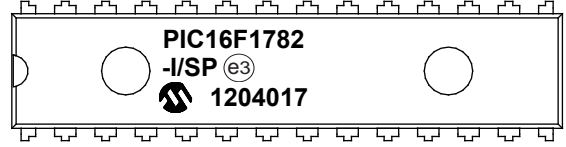
28-Lead SOIC (7.50 mm)

Example

```
XXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXX

        YYWWNNN
```

```
PIC16F1782
-I/SO  e3


        1204017
```
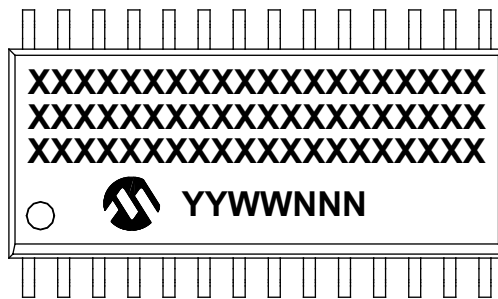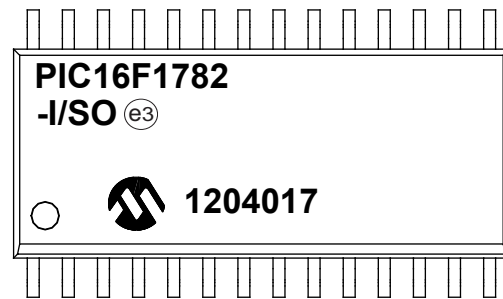
28-Lead SSOP (5.30 mm)

Example

```
XXXXXXXXXXXX
XXXXXXXXXXXX
        YYWWNNN
```

```
PIC16F1782
-I/SS  e3
        1204017
```

| Legend: | XX...X | Customer-specific information |
| --- | --- | --- |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week '01') |
| | NNN | Alphanumeric traceability code |
| | e3 | Pb-free JEDEC designator for Matte Tin (Sn) |
| | * | This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package. |
| Note: | | In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information. |

## Package Marking Information (Continued)

28-Lead QFN (6x6 mm)

Example

PIN 1

XXXXXXXX
XXXXXXXX
YYWWNNN

PIN 1

16F1782

-I/ML (e3)

120417
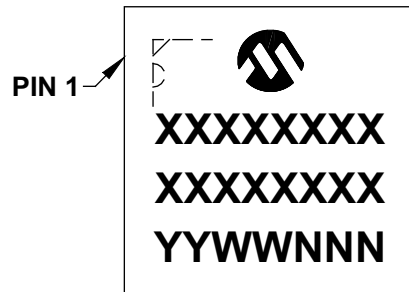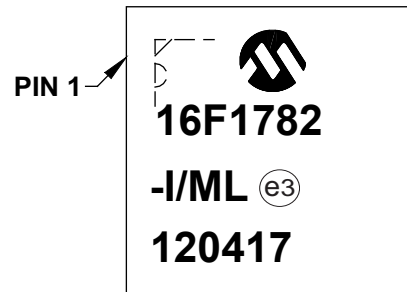
28-Lead UQFN (4x4x0.5 mm)

Example

PIN 1

● XXXXX
XXXXXX
XXXXXX
YWWNNN

PIN 1

● PIC16
LF1782
I/MV
204017

| Legend: | XX...X | Customer-specific information |
|---|---|---|
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week '01') |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC designator for Matte Tin (Sn) |
| | * | This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package. |

**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

## 33.2    Package Details

The following sections give the technical details of the packages.

### 28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

> **Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging
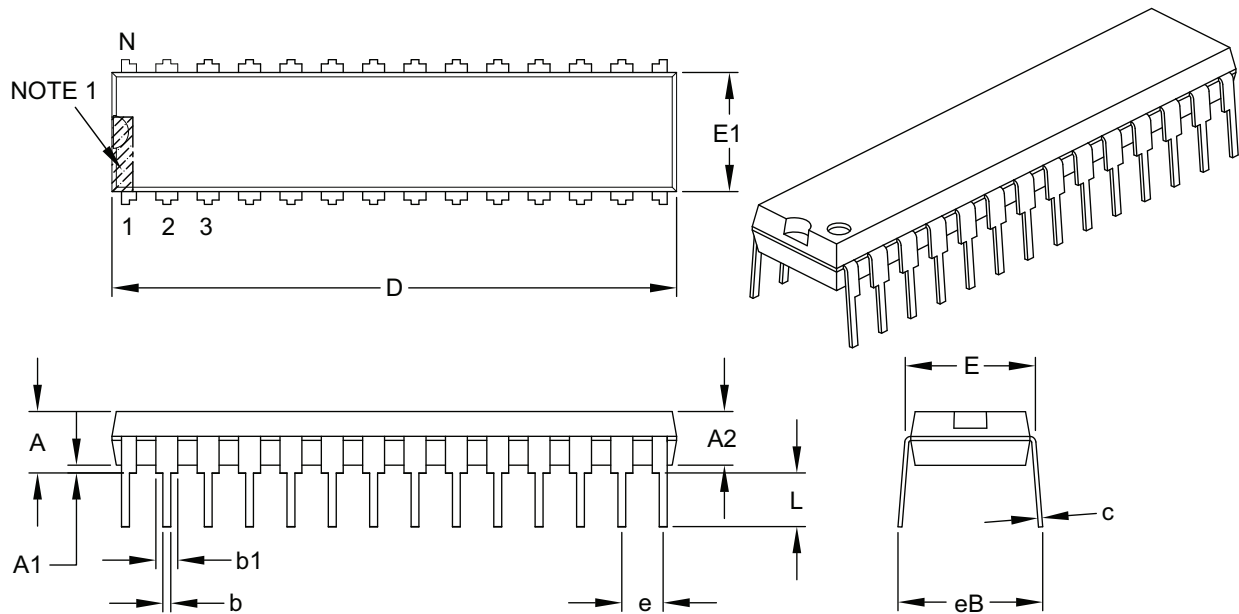


|                           | Units | INCHES |     |     |
|---------------------------|-------|--------|-----|-----|
| Dimension Limits          |       | MIN    | NOM | MAX |
| Number of Pins            | N     |        | 28  |     |
| Pitch                     | e     |        | .100 BSC |  |
| Top to Seating Plane      | A     | –      | –   | .200 |
| Molded Package Thickness  | A2    | .120   | .135 | .150 |
| Base to Seating Plane     | A1    | .015   | –   | –   |
| Shoulder to Shoulder Width | E    | .290   | .310 | .335 |
| Molded Package Width      | E1    | .240   | .285 | .295 |
| Overall Length            | D     | 1.345  | 1.365 | 1.400 |
| Tip to Seating Plane      | L     | .110   | .130 | .150 |
| Lead Thickness            | c     | .008   | .010 | .015 |
| Upper Lead Width          | b1    | .040   | .050 | .070 |
| Lower Lead Width          | b     | .014   | .018 | .022 |
| Overall Row Spacing  §    | eB    | –      | –   | .430 |

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.
      BSC: Basic Dimension. Theoretically exact value shown without tolerances.

*Microchip Technology Drawing C04-070B*

# PIC16(L)F1782/3

## 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

**TOP VIEW**

**SIDE VIEW**

**VIEW A-A**

Microchip Technology Drawing C04-052C Sheet 1 of 2

**Preliminary**

## 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



**VIEW C**
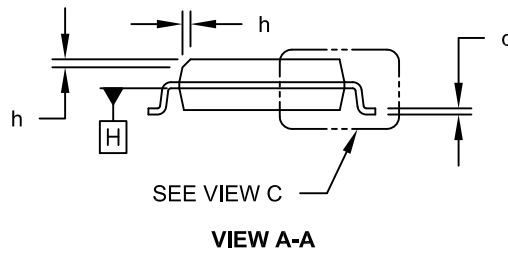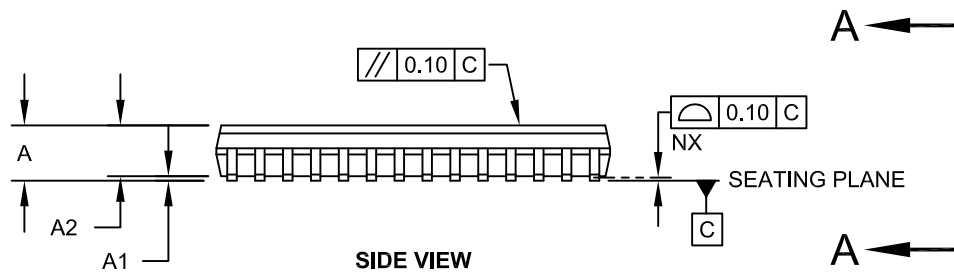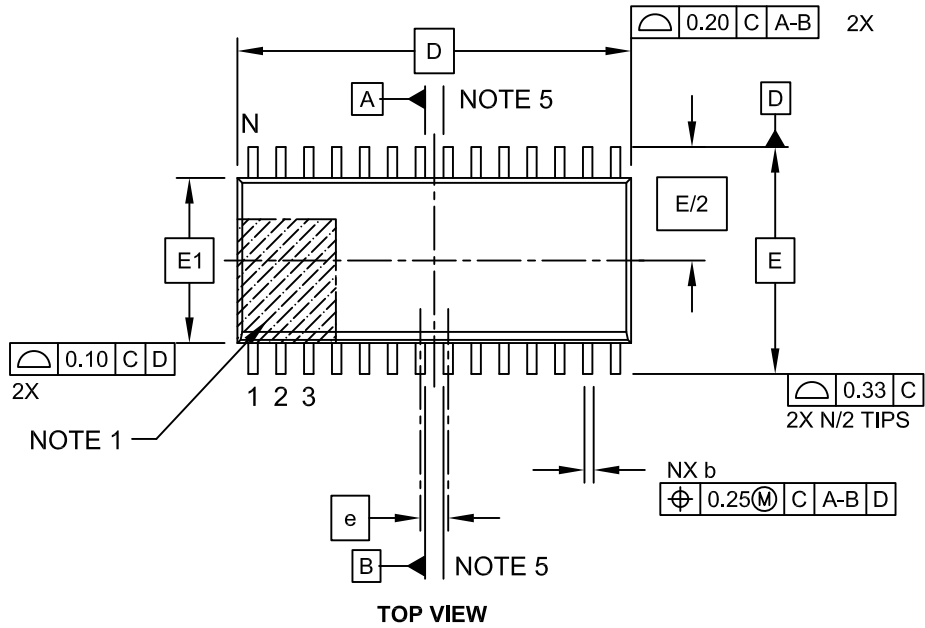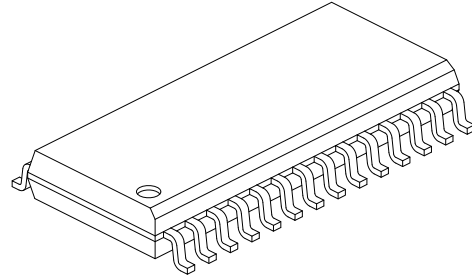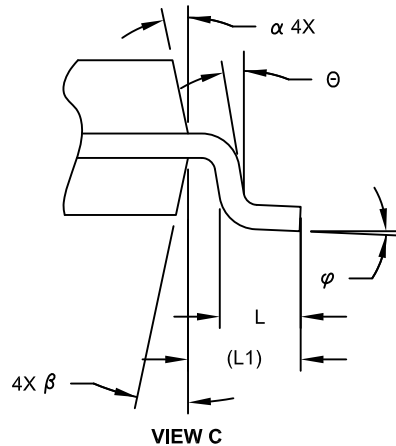
| | Units | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | 1.27 BSC | |
| Overall Height | A | - | - | 2.65 |
| Molded Package Thickness | A2 | 2.05 | - | - |
| Standoff        § | A1 | 0.10 | - | 0.30 |
| Overall Width | 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 | $\Theta$ | 0° | - | - |
| Foot Angle | $\varphi$ | 0° | - | 8° |
| Lead Thickness | c | 0.18 | - | 0.33 |
| Lead Width | b | 0.31 | - | 0.51 |
| Mold Draft Angle Top | $\alpha$ | 5° | - | 15° |
| Mold Draft Angle Bottom | $\beta$ | 5° | - | 15° |

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

*Microchip Technology Drawing C04-052C Sheet 2 of 2*

# PIC16(L)F1782/3

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
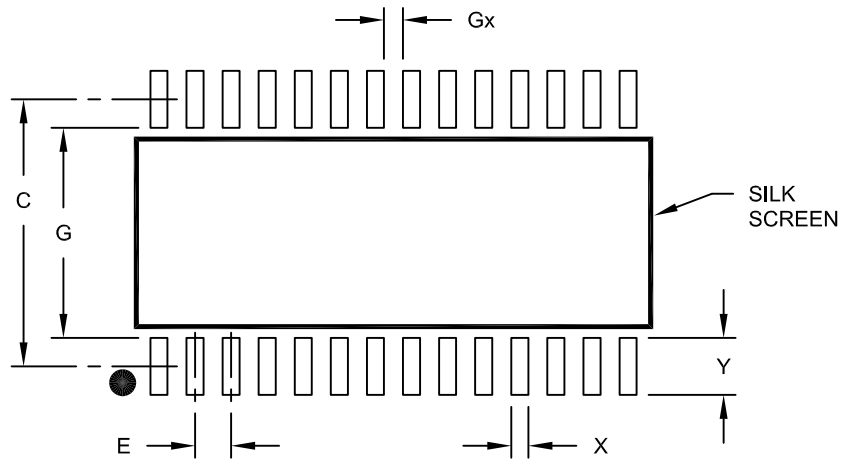


RECOMMENDED LAND PATTERN

| | 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

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 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
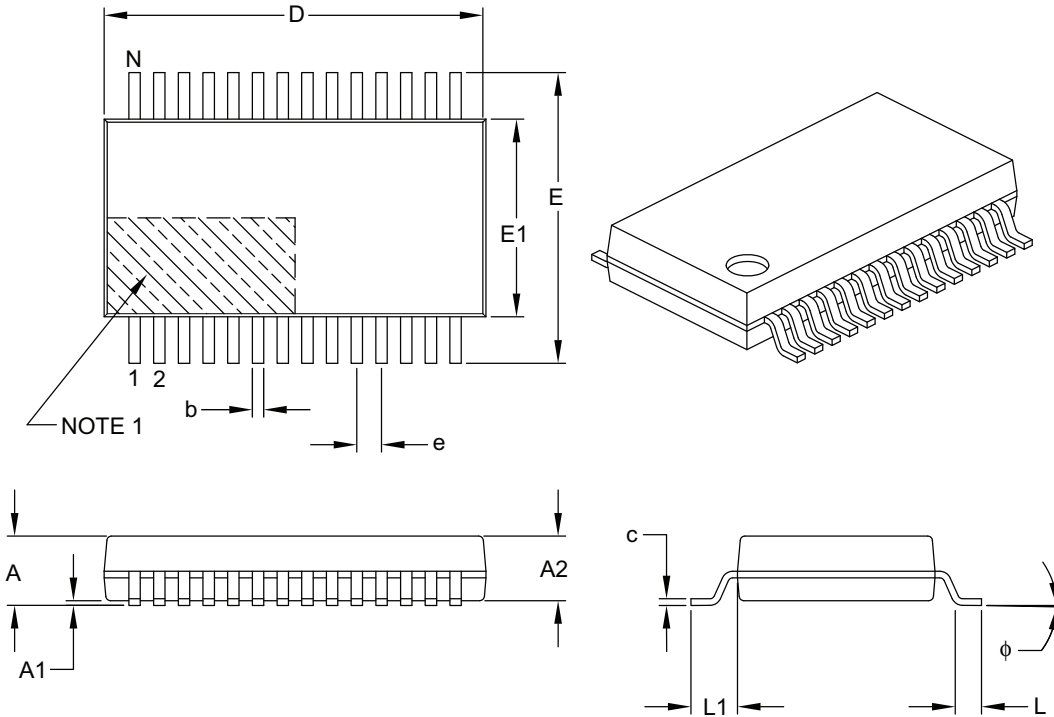


| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | – | – | 2.00 |
| Molded Package Thickness | A2 | 1.65 | 1.75 | 1.85 |
| Standoff | A1 | 0.05 | – | – |
| Overall Width | E | 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.

Microchip Technology Drawing C04-073B

# PIC16(L)F1782/3

**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



SILK SCREEN

RECOMMENDED LAND PATTERN
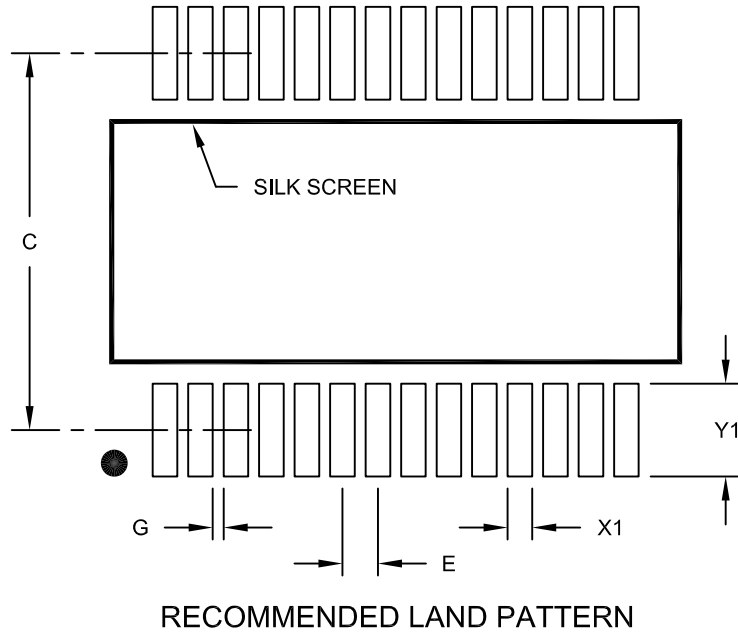
| | Units | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.65 BSC | |
| Contact Pad Spacing | C | | 7.20 | |
| Contact Pad Width (X28) | X1 | | | 0.45 |
| Contact Pad Length (X28) | Y1 | | | 1.75 |
| Distance Between Pads | G | 0.20 | | |

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
    BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

---

**Preliminary** © 2011-2012 Microchip Technology Inc.

**28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN]
with 0.55 mm Contact Length**

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at
http://www.microchip.com/packaging



**TOP VIEW**

**BOTTOM VIEW**
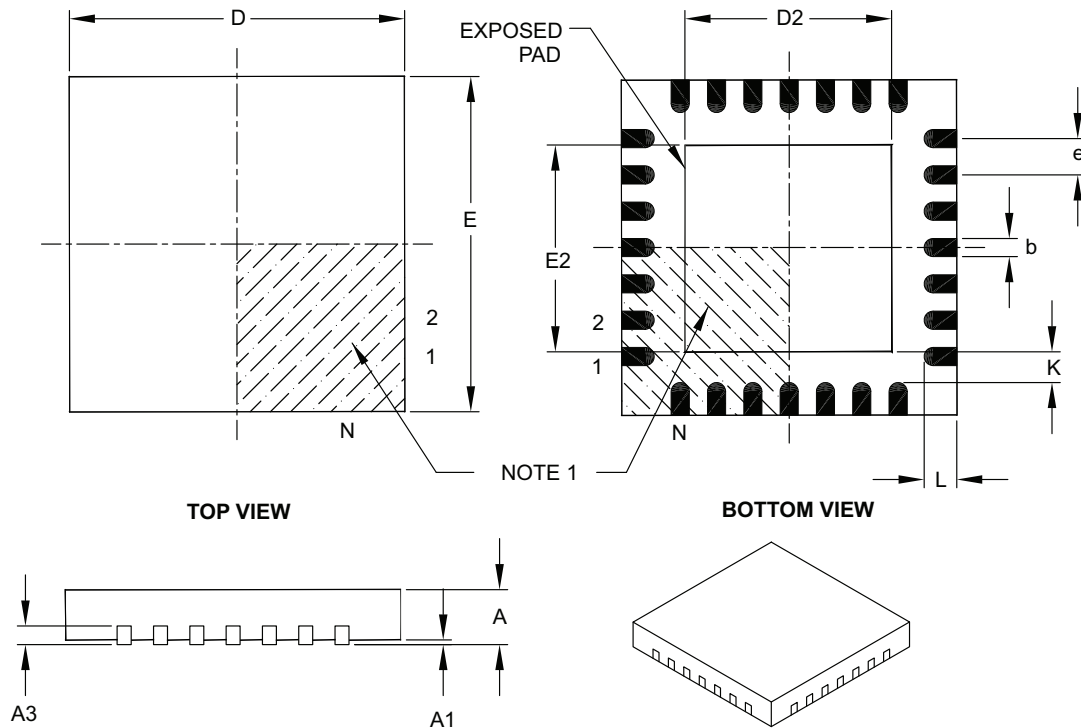
| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Contact Thickness | A3 | | 0.20 REF | |
| Overall Width | E | | 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 |
| Contact Width | b | 0.23 | 0.30 | 0.35 |
| Contact Length | L | 0.50 | 0.55 | 0.70 |
| Contact-to-Exposed Pad | K | 0.20 | – | – |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-105B

# PIC16(L)F1782/3

**28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN]**
**with 0.55 mm Contact Length**

RECOMMENDED LAND PATTERN

| | Units | | MILLIMETERS | |
|---|---|---|---|---|
| | Dimension Limits | MIN | NOM | MAX |
| Contact Pitch | E | | 0.65 BSC | |
| Optional Center Pad Width | W2 | | | 4.25 |
| Optional Center Pad Length | T2 | | | 4.25 |
| Contact Pad Spacing | C1 | | 5.70 | |
| Contact Pad Spacing | C2 | | 5.70 | |
| Contact Pad Width (X28) | X1 | | | 0.37 |
| Contact Pad Length (X28) | Y1 | | | 1.00 |
| 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. C04-2105A

**Preliminary**
© 2011-2012 Microchip Technology Inc.

## 28-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) – 4x4x0.5 mm Body [UQFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at
http://www.microchip.com/packaging



**TOP VIEW**

**BOTTOM VIEW**
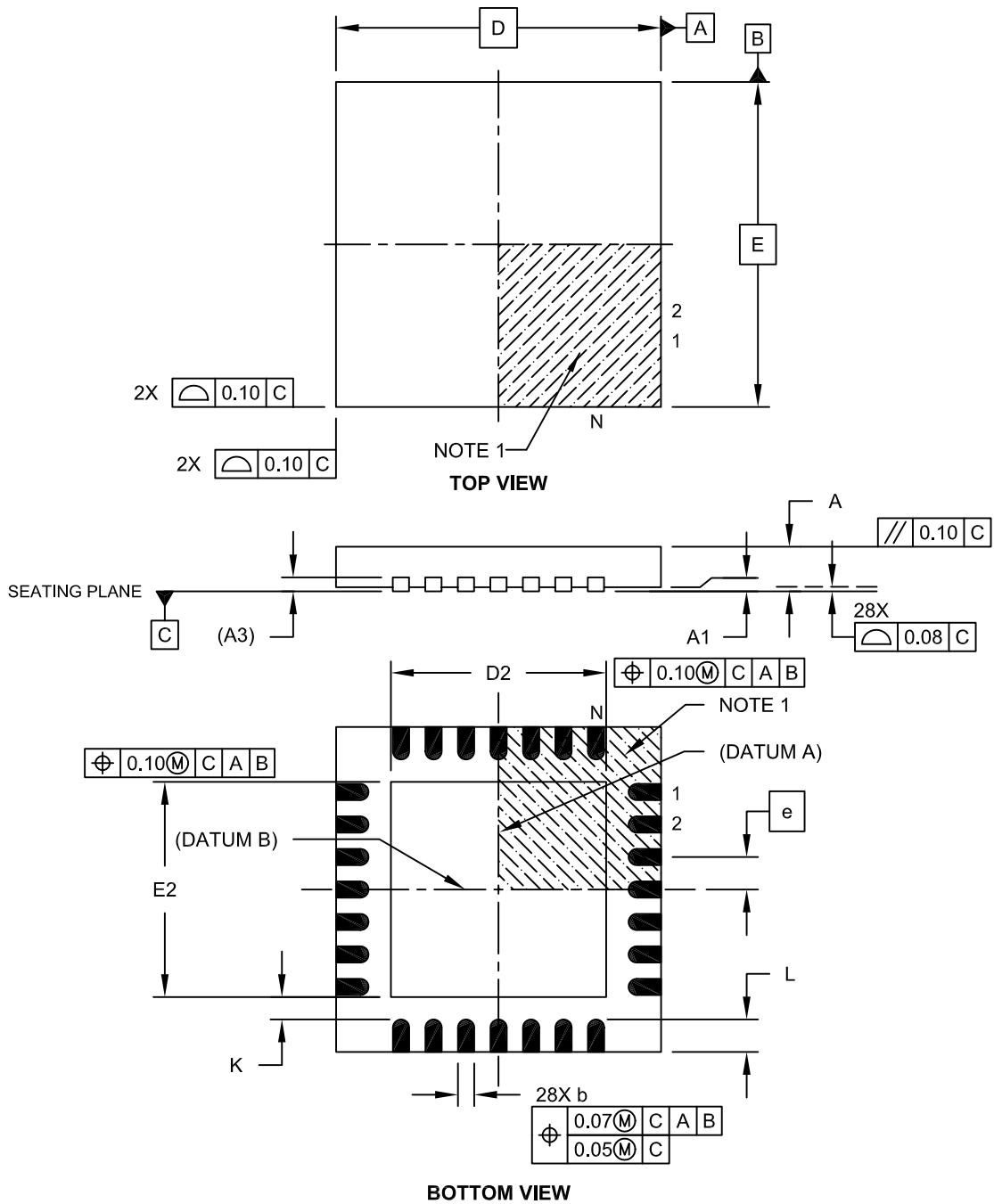
Microchip Technology Drawing  C04-152A Sheet 1 of 2

# PIC16(L)F1782/3

## 28-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) – 4x4x0.5 mm Body [UQFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging
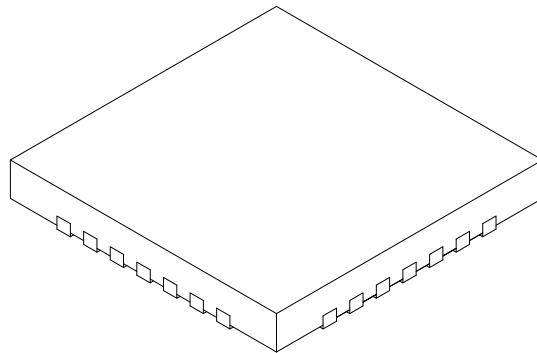


| | Units | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | 0.40 BSC | |
| Overall Height | A | 0.45 | 0.50 | 0.55 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Contact Thickness | A3 | | 0.127 REF | |
| Overall Width | E | | 4.00 BSC | |
| Exposed Pad Width | E2 | 2.55 | 2.65 | 2.75 |
| Overall Length | D | | 4.00 BSC | |
| Exposed Pad Length | D2 | 2.55 | 2.65 | 2.75 |
| Contact Width | b | 0.15 | 0.20 | 0.25 |
| Contact Length | L | 0.30 | 0.40 | 0.50 |
| Contact-to-Exposed Pad | K | 0.20 | - | - |

**Notes:**
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing  C04-152A Sheet 2 of 2

**Preliminary** © 2011-2012 Microchip Technology Inc.

## APPENDIX A: DATA SHEET REVISION HISTORY

### Revision A (04/2011)

Original release.

### Revision B (06/2011)

Revised Section 18.0; Revised Table 30-8; Add Operational Amplifier Table.

### Revision C (03/2012)

Electrical Specifications update.

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

# INDEX

# PIC16(L)F1782/3

# PIC16(L)F1782/3

**Preliminary**

**NOTES:**

**Preliminary** © 2011-2012 Microchip Technology Inc.

## 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
- Development Systems Information Line

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the web site at: http://microchip.com/support**

# PIC16(L)F1782/3

## READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

TO:     Technical Publications Manager                                    Total Pages Sent _____
RE:     Reader Response

From:  Name _____
        Company _____
        Address _____
        City / State / ZIP / Country _____
        Telephone: (_____) _____ - _____        FAX: (_____) _____ - _____

Application (optional):

Would you like a reply? ____ Y ____ N

Device:  PIC16(L)F1782/3                              Literature Number:  DS41579C

Questions:

1.  What are the best features of this document?

    _____

    _____

2.  How does this document meet your hardware and software development needs?

    _____

    _____

3.  Do you find the organization of this document easy to follow? If not, why?

    _____

    _____

4.  What additions to the document do you think would enhance the structure and subject?

    _____

    _____

5.  What deletions from the document could be made without affecting the overall usefulness?

    _____

    _____

6.  Is there any incorrect or misleading information (what and where)?

    _____

    _____

7.  How would you improve this document?

    _____

    _____

© 2011-2012 Microchip Technology Inc.

## PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| PART NO. | [X]<sup>(1)</sup> | - | X | /XX | XXX |
|----------|-----------|---|---|-----|-----|
| Device | Tape and Reel Option | | Temperature Range | Package | Pattern |

| | |
|---|---|
| **Device:** | PIC16F1782, PIC16LF1782, PIC16F1783, PIC16LF1783 |

| **Tape and Reel Option:** | Blank | = Standard packaging (tube or tray) |
|---|---|---|
| | T | = Tape and Reel<sup>(1)</sup> |

| **Temperature Range:** | I | = -40°C to +85°C | (Industrial) |
|---|---|---|---|
| | E | = -40°C to +125°C | (Extended) |

| **Package:** | ML | = QFN |
|---|---|---|
| | MV | = UQFN |
| | SP | = SPDIP |
| | SO | = SOIC |
| | SS | = SSOP |

| **Pattern:** | QTP, SQTP, Code or Special Requirements (blank otherwise) |
|---|---|

**Examples:**

a) PIC16LF1782T- I/MV 301
   Tape and Reel,
   Industrial temperature,
   UQFN package,
   QTP pattern #301

b) PIC16LF1783- I/P
   Industrial temperature
   SPDIP package

c) PIC16F1783- E/SS
   Extended temperature,
   SSOP package

**Note 1:** Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.

---

© 2011-2012 Microchip Technology Inc.          **Preliminary**          DS41579C-page 443

![Microchip logo]

# 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

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Cleveland**
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

**Indianapolis**
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

**Santa Clara**
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

**Toronto**
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

## ASIA/PACIFIC

**Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

**Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**China - Beijing**
Tel: 86-10-8569-7000
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: 86-571-2819-3187
Fax: 86-571-2819-3189

**China - Hong Kong SAR**
Tel: 852-2401-1200
Fax: 852-2401-3431

**China - Nanjing**
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

**China - Qingdao**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**China - Shanghai**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**China - Shenyang**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**China - Shenzhen**
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

**China - Wuhan**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**China - 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: 91-20-2566-1512
Fax: 91-20-2566-1513

**Japan - Osaka**
Tel: 81-66-152-7160
Fax: 81-66-152-9310

**Japan - Yokohama**
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

**Korea - Daegu**
Tel: 82-53-744-4301
Fax: 82-53-744-4302

**Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

**Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**Taiwan - Hsin Chu**
Tel: 886-3-5778-366
Fax: 886-3-5770-955

**Taiwan - Kaohsiung**
Tel: 886-7-536-4818
Fax: 886-7-330-9305

**Taiwan - Taipei**
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

**Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

## EUROPE

**Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**Denmark - Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**UK - Wokingham**
Tel: 44-118-921-5869
Fax: 44-118-921-5820

11/29/11