# PIC16LF1566/1567

## 28/40/44-Pin Flash, 8-Bit Microcontrollers with XLP Technology

### Description

The PIC16LF1566/1567 microcontrollers deliver unique on-chip features for the design of mTouch® solutions and general purpose applications in 28/40/44-pin count packages. Two 10-bit high-speed ADCs with automated hardware CVD modules connect up to 34 analog channels to achieve a total sampling rate of 600k samples per second. This family provides mutual capacitance output drivers on all analog channels, two PWMs, two MSSP modules with low input voltage options and one EUSART, which makes this family an excellent solution to implement low-power and noise-robust capacitive sensing and other front-end sampling applications with minimal software overhead.

### Core Features

- C Compiler Optimized RISC Architecture
- Only 49 Instructions
- Operating Speed:
  - 0-32 MHz clock input
  - 125 ns minimum instruction cycle
- Interrupt Capability
- 16-Level Deep Hardware Stack
- Up to Three 8-bit Timers
- One 16-bit Timer
- Power-on Reset (POR)
- Power-up Timer (PWRT)
- Low-Power Brown-Out Reset (LPBOR)
- Programmable Watchdog Timer (WDT) up to 256s
- Programmable Code Protection

### Memory

- Up to 8k Words Flash Program Memory
- 1024 Bytes Data SRAM Memory
- Direct, Indirect and Relative Addressing modes

### Operating Characteristics

- Operating Voltage Range:
  - 1.8V to 3.6V
- Temperature Range:
  - Industrial: -40°C to 85°C
  - Extended: -40°C to 125°C

### eXtreme Low-Power (XLP) Features

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

### Digital Peripherals

- PWM: Two 10-bit Pulse-Width Modulators
  - Output on up to five pins per PWM at the same time
- Dual Master Synchronous Serial Port (MSSP) with SPI and $I^2C$:
  - 7-bit address masking
  - SMBus/PMBus™ compatibility
  - Configurable low input voltage threshold for $I^2C$
- Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART):
  - RS-232, R-485, and LIN compatible
  - Auto-Baud Detect
  - Auto-wake-up on start
- Up to 35 I/O Pins and One Input Pin:
  - Individually programmable pull-ups
  - Interrupt-on-Change with edge-select

### Intelligent Analog Peripherals

- Dual 10-Bit Analog-to-Digital Converter (ADC):
  - Up to 35 external channels
  - Conversion available during Sleep
  - Temperature indicator
  - Simultaneous sampling on two ADCs
  - Connect multiple channels together for sampling
  - External conversion trigger
  - Fixed Voltage Reference as a channel
  - External pin as positive ADC voltage reference
  - Combined 600k samples per second
- Hardware Capacitive Voltage Divider (CVD)
  - Double-sample conversions
  - Two sets of result registers
  - 7-bit precharge timer
  - 7-bit acquisition timer
  - Two guard ring output drives
  - Mutual capacitance TX output on any analog channel
  - 30 pF adjustable sample and hold capacitor
- Internal Voltage Reference Module

### Clocking Structure

- 16 MHz Internal Oscillator Block:
  - ±1% at calibration
  - Selectable frequency range from 0 to 32 MHz
- 31 kHz Low-Power Internal Oscillator
- External Oscillator Block with:
  - Two external clock modes up to 32 MHz
- Oscillator Start-up Timer (OST)

### Programming/Debug Features

- In-Circuit Debug Integrated On-Chip
- Emulation Header for Advanced Debug:
  - Provides trace, background debug and up to 32 hardware break points
- In-Circuit Serial Programming™ (ICSP™) via Two Pin

**TABLE 1: PIC16LF1566/1567 FAMILY TYPES**

| Device | Data Sheet Index | Program Memory Flash (words) | Data EEPROM (bytes) | SRAM (bytes) | I/Os (1) | 10-bit ADCs(4) | Analog Channels(2)(3) CVD RX Channels | CVD TX Channels(5) | Timers 8/16-bit | EUSART | MSSP | PWM | Debug |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PIC12LF1552 | (A) | 2048 | 0 | 256 | 6 | 1 | 4 | 1 | 1 / 0 | - | 1 | - | - |
| PIC16LF1554 | (B) | 4096 | 0 | 256 | 12 | 2 | 10 | 2 | 2 / 1 | 1 | 1 | 2 | I |
| PIC16LF1559 | (B) | 8192 | 0 | 512 | 18 | 2 | 16 | 2 | 2 / 1 | 1 | 1 | 2 | I |
| PIC16LF1566 | (C) | 8192 | 0 | 1024 | 25 | 2 | 23 | 23 | 3 / 1 | 1 | 2 | 2 | I |
| PIC16LF1567 | (C) | 8192 | 0 | 1024 | 36 | 2 | 34 | 34 | 3 / 1 | 1 | 2 | 2 | I |

**Note 1:** The $\overline{\text{MCLR}}$ pin is input-only.

**2:** Analog channels are split between the available ADCs.

**3:** Maximum usable analog channels assuming one pin must be assigned to output.

**4:** If $V_{DD}$ > 2.4V, ADC may be overclocked 4x ($T_{AD}$ = 0.25 µs).

**5:** Includes functionality of ADxGRDA output pin.

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

**A:** DS40001674 PIC12LF1552 Data Sheet, 8-Pin Flash, 8-Bit Microcontrollers

**B:** DS40001761 PIC16LF1554/1559 Data Sheet, 20-Pin Flash, 8-Bit Microcontrollers with XLP Technology

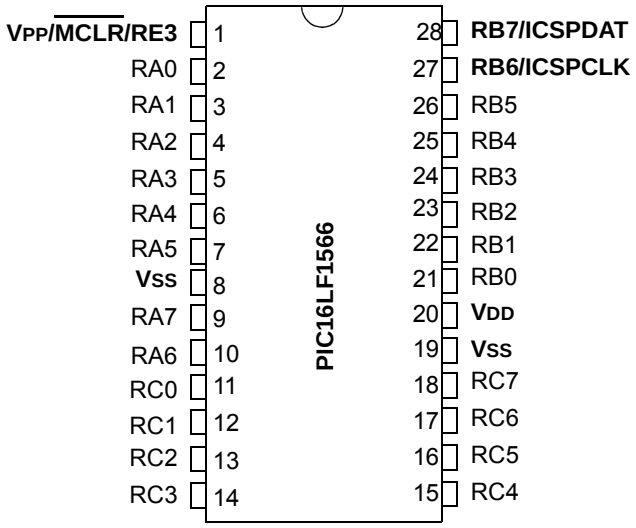
**C:** DS40001817 PIC16LF1566/1567 Data Sheet 28/40/44-Pin Flash, 8-Bit Microcontrollers with XLP Technology

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


© 2015-2016 Microchip Technology Inc.

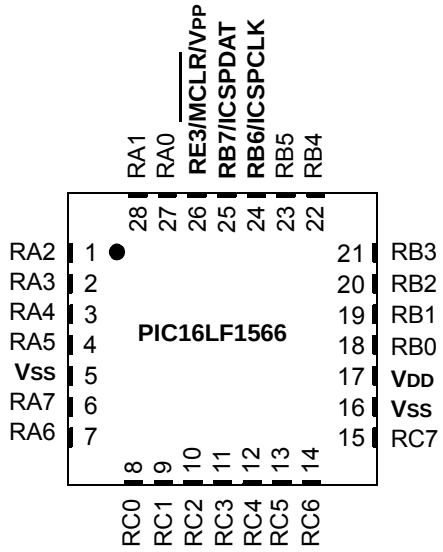
## PIN DIAGRAMS

**FIGURE 1: 28-PIN SPDIP, SOIC, SSOP DIAGRAM FOR PIC16LF1566**



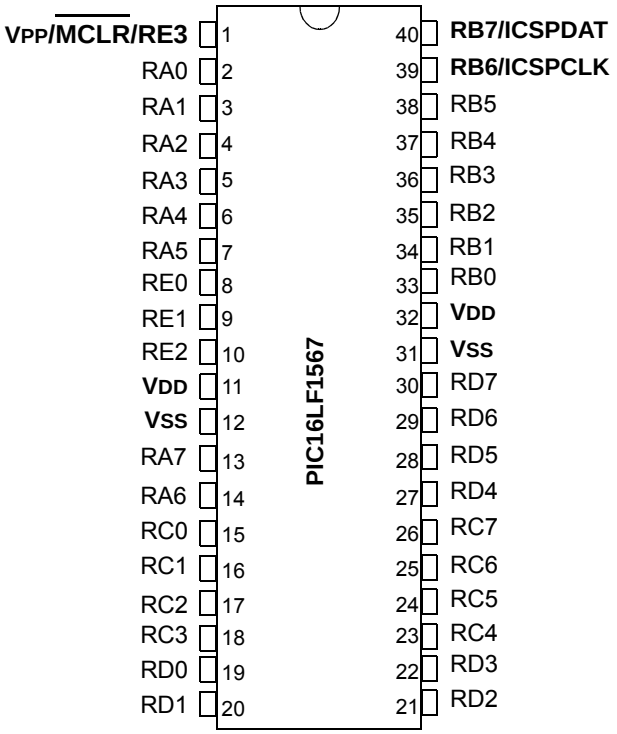

**Note:** See Table 2 for the pin allocation tables.

**FIGURE 2: 28-PIN UQFN DIAGRAM FOR PIC16LF1566**



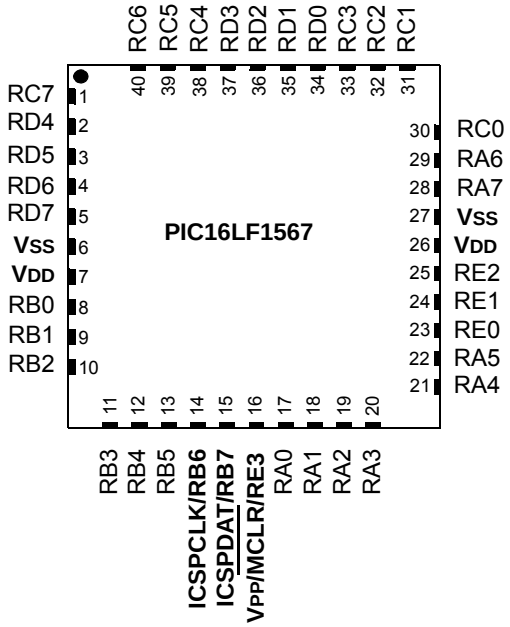

**Note:** See Table 2 or the pin allocation tables.

**FIGURE 3: 40-PIN PDIP DIAGRAM FOR PIC16LF1567**




**Note:** See Table 3 for the pin allocation tables.

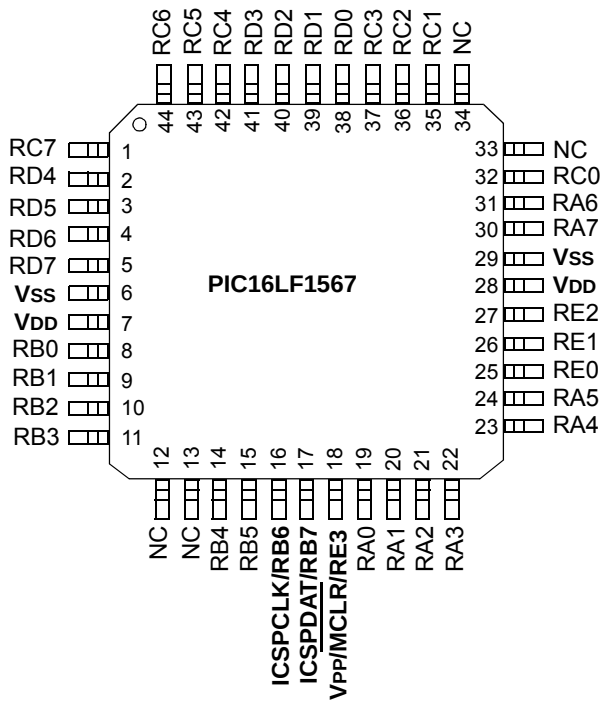
**FIGURE 4: 40-PIN UQFN DIAGRAM FOR PIC16LF1567**




**Note:** See Table 3 for the pin allocation tables.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 5: 44-PIN TQFP DIAGRAM FOR PIC16LF1567**




**Note:** See Table 3 for the pin allocation table.

## PIN ALLOCATION TABLES

**TABLE 2: 28-PIN ALLOCATION TABLE (PIC16LF1566)**

| I/O | 28-Pin SPDIP/SOIC/SSOP | 28-Pin UQFN | Analog Channel | ADC and CVD | Timers | PWM | EUSART | MSSP | Interrupt | Pull-up | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 2 | 27 | AN20 | — | — | PWM10 | — | SS1(1) | — | — | — |
| RA1 | 3 | 28 | AN10 | — | — | PWM11 | — | SS2 | — | — | — |
| RA2 | 4 | 1 | AN0 | VREF- | — | PWM12 | — | — | — | — | — |
| RA3 | 5 | 2 | AN1 | VREF+ | — | PWM13 | — | — | — | — | — |
| RA4 | 6 | 3 | AN2 | — | T0CKI | — | — | — | — | — | — |
| RA5 | 7 | 4 | AN21 | — | — | — | — | SS1(1) | — | — | — |
| RA6 | 10 | 7 | AN22 | ADTRIG | — | — | — | — | — | — | CLKOUT |
| RA7 | 9 | 6 | AN11 | — | — | — | — | — | — | — | CLKIN |
| RB0 | 21 | 18 | AN16 | — | — | PWM20 | — | — | INT<br>IOC | Y | — |
| RB1 | 22 | 19 | AN27 | — | — | PWM21 | — | — | IOC | Y | — |
| RB2 | 23 | 20 | AN17 | — | — | PWM22 | — | — | IOC | Y | — |
| RB3 | 24 | 21 | AN28 | — | — | PWM23 | — | — | IOC | Y | — |
| RB4 | 25 | 22 | AN18 | AD1GRDA(1)<br>AD2GRDA(1) | — | — | — | — | IOC | Y | — |
| RB5 | 26 | 23 | AN29 | AD1GRDA(1)<br>AD2GRDA(1) | T1G | — | — | — | IOC | Y | — |
| RB6 | 27 | 24 | AN19 | AD1GRDB(1)<br>AD2GRDB(1) | — | — | — | — | IOC | Y | ICSPCLK<br>ICDCLK |
| RB7 | 28 | 25 | AN40 | AD1GRDB(1)<br>AD2GRDB(1) | — | — | — | — | IOC | Y | ICSPDAT<br>ICDDAT |
| RC0 | 11 | 8 | AN12 | — | T1CKI | — | — | SDO2 | — | — | — |
| RC1 | 12 | 9 | AN23 | — | — | PWM2 | — | SCL2<br>SCK2 | — | — | — |
| RC2 | 13 | 10 | AN13 | — | — | PWM1 | — | SDA2<br>SDI2 | — | — | — |
| RC3 | 14 | 11 | AN24 | — | — | — | — | SCL1<br>SCK1 | — | — | — |
| RC4 | 15 | 12 | AN14 | — | — | — | — | SDA1<br>SDI1 | — | — | — |
| RC5 | 16 | 13 | AN25 | — | — | — | — | SDO1<br>$I^2CLVL$ | — | — | — |


© 2015-2016 Microchip Technology Inc.

**TABLE 2: 28-PIN ALLOCATION TABLE (PIC16LF1566) (CONTINUED)**

| I/O | 28-Pin SPDIP/SOIC/SSOP | 28-Pin UQFN | Analog Channel | ADC and CVD | Timers | PWM | EUSART | MSSP | Interrupt | Pull-up | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RC6 | 17 | 14 | AN15 | — | — | — | TX CK | — | — | — | — |
| RC7 | 18 | 15 | AN26 | — | — | — | RX DT | — | — | — | — |
| RE3 | 1 | 26 | — | — | — | — | — | — | — | Y | $\overline{MCLR}$ $V_{PP}$ |
| $V_{DD}$ | 20 | 17 | — | — | — | — | — | — | — | — | $V_{DD}$ |
| $V_{SS}$ | 8 | 5 | — | — | — | — | — | — | — | — | $V_{SS}$ |
| $V_{SS}$ | 19 | 16 | — | — | — | — | — | — | — | — | $V_{SS}$ |

**Note 1:** Pin functions can be assigned to one of two pin locations via software.

**TABLE 3: 40/44-PIN ALLOCATION TABLE (PIC16LF1567)**

| I/O | 40-Pin PDIP | 40-Pin UQFN | 44-Pin TQFP | Analog Channel | ADC and CVD | Timers | PWM | EUSART | MSSP | Interrupt | Pull-Up | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 2 | 17 | 19 | AN20 | — | — | PWM10 | — | SS1(1) | — | — | — |
| RA1 | 3 | 18 | 20 | AN10 | — | — | PWM11 | — | SS2 | — | — | — |
| RA2 | 4 | 19 | 21 | AN0 | VREF- | — | PWM12 | — | — | — | — | — |
| RA3 | 5 | 20 | 22 | AN1 | VREF+ | | PWM13 | — | — | — | — | — |
| RA4 | 6 | 21 | 23 | AN2 | — | T0CKI | — | — | — | — | — | — |
| RA5 | 7 | 22 | 24 | AN21 | — | — | — | — | SS1(1) | — | — | — |
| RA6 | 14 | 29 | 31 | AN22 | ADTRIG | — | — | — | — | — | — | CLKOUT |
| RA7 | 13 | 28 | 30 | AN11 | — | — | — | — | — | — | — | CLKIN |
| RB0 | 33 | 8 | 8 | AN16 | — | — | PWM20 | — | — | INT IOC | Y | — |
| RB1 | 34 | 9 | 9 | AN27 | — | — | PWM21 | — | — | IOC | Y | — |
| RB2 | 35 | 10 | 10 | AN17 | — | — | PWM22 | — | — | IOC | Y | — |
| RB3 | 36 | 11 | 11 | AN28 | — | — | PWM23 | — | — | IOC | Y | — |
| RB4 | 37 | 12 | 14 | AN18 | AD1GRDA(1) AD2GRDA(1) | — | — | — | — | IOC | Y | — |
| RB5 | 38 | 13 | 15 | AN29 | AD1GRDA(1) AD2GRDA(1) | T1G | — | — | — | IOC | Y | — |
| RB6 | 39 | 14 | 16 | AN19 | AD1GRDB(1) AD2GRDB(1) | — | — | — | — | IOC | Y | ICSPCLK ICDCLK |
| RB7 | 40 | 15 | 17 | AN40 | AD1GRDB(1) AD2GRDB(1) | — | — | — | — | IOC | Y | ICSPDAT ICDDAT |
| RC0 | 15 | 30 | 32 | AN12 | — | T1CKI | — | — | SDO2 | — | — | — |
| RC1 | 16 | 31 | 35 | AN23 | — | — | PWM2 | — | SCL2 SCK2 | — | — | — |
| RC2 | 17 | 32 | 36 | AN13 | — | — | PWM1 | — | SDA2 SDI2 | — | — | — |
| RC3 | 18 | 33 | 37 | AN24 | — | — | — | — | SCL1 SCK1 | — | — | — |
| RC4 | 23 | 38 | 42 | AN14 | — | — | — | — | SDA1 SDI1 | — | — | — |
| RC5 | 24 | 39 | 43 | AN25 | — | — | — | — | SDO1 I2CLVL | — | — | — |
| RC6 | 25 | 40 | 44 | AN15 | — | — | — | TX CK | — | — | — | — |
| RC7 | 26 | 1 | 1 | AN26 | — | — | — | RX DT | — | — | — | — |
| RD0 | 19 | 34 | 38 | AN42 | — | — | — | — | — | — | — | — |


© 2015-2016 Microchip Technology Inc.

**TABLE 3: 40/44-PIN ALLOCATION TABLE (PIC16LF1567) (CONTINUED)**

| I/O | 40-Pin PDIP | 40-Pin UQFN | 44-Pin TQFP | Analog Channel | ADC and CVD | Timers | PWM | EUSART | MSSP | Interrupt | Pull-Up | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RD1 | 20 | 35 | 39 | AN32 | — | — | — | — | — | — | — | — |
| RD2 | 21 | 36 | 40 | AN43 | — | — | — | — | — | — | — | — |
| RD3 | 22 | 37 | 41 | AN33 | — | — | — | — | — | — | — | — |
| RD4 | 27 | 2 | 2 | AN34 | — | — | — | — | — | — | — | — |
| RD5 | 28 | 3 | 3 | AN44 | — | — | — | — | — | — | — | — |
| RD6 | 29 | 4 | 4 | AN35 | — | — | — | — | — | — | — | — |
| RD7 | 30 | 5 | 5 | AN45 | — | — | — | — | — | — | — | — |
| RE0 | 8 | 23 | 25 | AN30 | — | — | — | — | — | — | — | — |
| RE1 | 9 | 24 | 26 | AN41 | — | — | — | — | — | — | — | — |
| RE2 | 10 | 25 | 27 | AN31 | — | — | — | — | — | — | — | — |
| RE3 | 1 | 16 | 18 | — | — | — | — | — | — | — | Y | $\overline{\text{MCLR}}$ $V_{PP}$ |
| $V_{DD}$ | 11 | 7 | 7 | — | — | — | — | — | — | — | — | $V_{DD}$ |
| $V_{DD}$ | — | 26 | 28 | — | — | — | — | — | — | — | — | $V_{DD}$ |
| $V_{SS}$ | 12 | 6 | 6 | — | — | — | — | — | — | — | — | $V_{SS}$ |
| $V_{SS}$ | 31 | 27 | 29 | — | — | — | — | — | — | — | — | $V_{SS}$ |

**Note 1:** Pin functions can be assigned to one of two pin locations via software.

## Table of Contents


1.0 Device Overview ..... 11
2.0 Enhanced Mid-Range CPU ..... 20
3.0 Memory Organization ..... 22
4.0 Device Configuration ..... 58
5.0 Oscillator Module ..... 63
6.0 Resets ..... 71
7.0 Interrupts ..... 79
8.0 Power-Down Mode (Sleep) ..... 89
9.0 Watchdog Timer (WDT) ..... 91
10.0 Flash Program Memory Control ..... 95
11.0 I/O Ports ..... 112
12.0 Interrupt-on-Change ..... 131
13.0 Fixed Voltage Reference (FVR) ..... 135
14.0 Temperature Indicator Module ..... 137
15.0 Analog-to-Digital Converter (ADC) Module ..... 139
16.0 Hardware Capacitive Voltage Divider (CVD) Module ..... 153
17.0 Timer0 Module ..... 179
18.0 Timer1 Module with Gate Control ..... 182
19.0 Timer2/4 Modules ..... 193
20.0 Master Synchronous Serial Port (MSSP1 and MSSP2) Module ..... 197
21.0 Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) ..... 255
22.0 Pulse-Width Modulation (PWM) Module ..... 282
23.0 In-Circuit Serial Programming™ (ICSP™) ..... 289
24.0 Instruction Set Summary ..... 291
25.0 Electrical Specifications ..... 305
26.0 DC and AC Characteristics Graphs and Charts ..... 328
27.0 Development Support ..... 329
28.0 Packaging Information ..... 333
Appendix A: Data Sheet Revision History ..... 352
The Microchip Website ..... 353
Customer Change Notification Service ..... 353
Customer Support ..... 353
Product Identification System ..... 354


### TO OUR VALUED CUSTOMERS

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

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com**. We welcome your feedback.

#### Most Current Data Sheet

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

**http://www.microchip.com**

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

#### Errata

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

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

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

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

#### Customer Notification System

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

 © 2015-2016 Microchip Technology Inc.

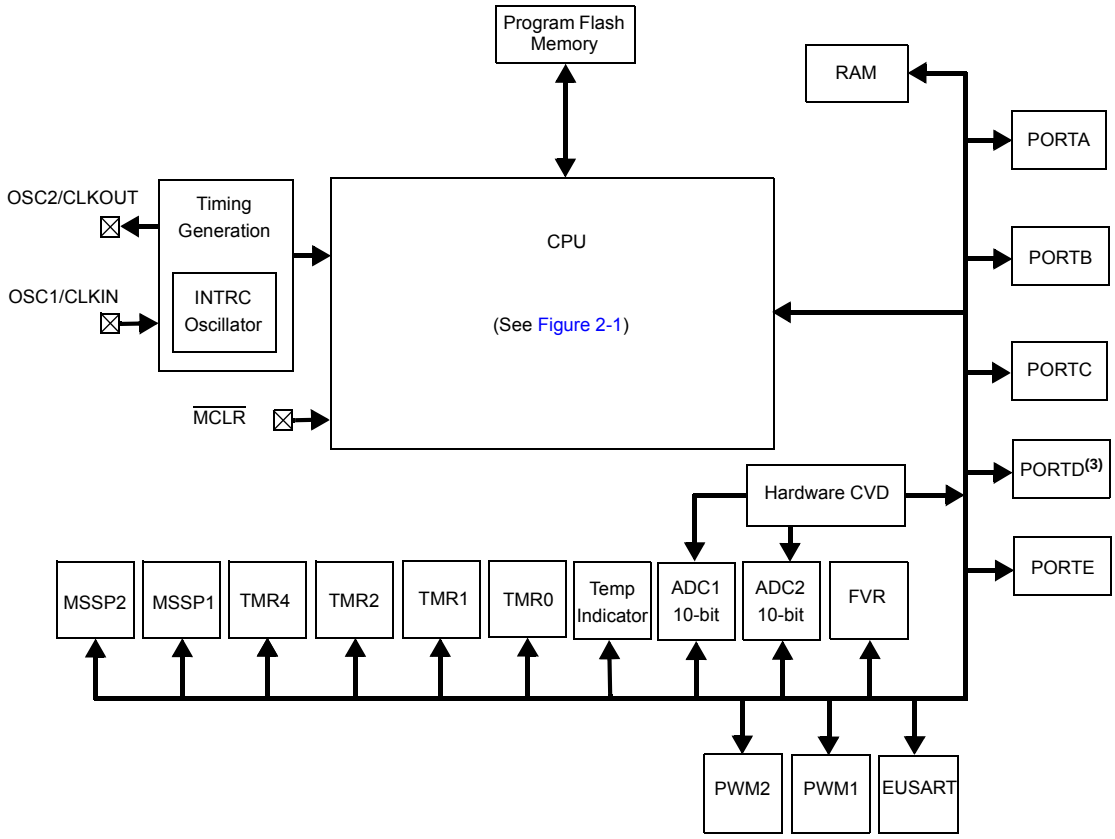
## 1.0 DEVICE OVERVIEW

The PIC16LF1566/1567 devices are described within this data sheet. The block diagram of these devices is shown in Figure 1-1, the available peripherals are shown in Table 1-1 and the pinout descriptions are shown in Table 1-2 and Table 1-3.

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

| Peripheral | | PIC16LF1566 | PIC16LF1567 |
|---|---|---|---|
| Analog-to-Digital Converter (ADC) | | | |
| | ADC1 | ● | ● |
| | ADC2 | ● | ● |
| Hardware Capacitive Voltage Divider (CVD) | | ● | ● |
| Enhanced Universal Synchronous/Asynchronous Receiver/Transmitter (EUSART) | | ● | ● |
| Fixed Voltage Reference (FVR) | | ● | ● |
| Temperature Indicator | | ● | ● |
| Master Synchronous Serial Ports | | | |
| | MSSP1 | ● | ● |
| | MSSP2 | ● | ● |
| PWM Modules | | | |
| | PWM1 | ● | ● |
| | PWM2 | ● | ● |
| Timers | | | |
| | Timer0 | ● | ● |
| | Timer1 | ● | ● |
| | Timer2 | ● | ● |
| | Timer4 | ● | ● |

**FIGURE 1-1: PIC16LF1566/1567 BLOCK DIAGRAM(1,2)**




**Note 1:** See applicable chapters for more information on peripherals.
**2:** See Table 1-1 for peripherals available on specific devices.
**3:** PIC16LF1567 only.

 © 2015-2016 Microchip Technology Inc.

**TABLE 1-2: PIC16LF1566 PINOUT DESCRIPTION**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RA0/AN20/PWM10/SS1(1) | RA0 | TTL | CMOS | General Purpose I/O. |
| | AN20 | AN | — | ADC Channel Input for ADC2. |
| | PWM10 | — | CMOS | PWM Output for PWM1. |
| | SS1 | ST | — | Slave Select Input for MSSP1. |
| RA1/AN10/PWM11/SS2 | RA1 | TTL | CMOS | General Purpose I/O. |
| | AN10 | AN | — | ADC Channel Input for ADC1. |
| | PWM11 | — | CMOS | PWM Output for PWM1. |
| | SS2 | ST | — | Slave Select Input for MSSP2. |
| RA2/AN0/PWM12 | RA2 | TTL | CMOS | General Purpose I/O. |
| | AN0 | AN | — | ADC Channel Input for both ADC1 and ADC2. |
| | PWM12 | — | CMOS | PWM Output for PWM1. |
| | VREF- | AN | — | ADC Negative Voltage Reference Input |
| RA3/AN1/ $V_{REF+}$/PWM13 | RA3 | TTL | CMOS | General Purpose I/O. |
| | AN1 | AN | — | ADC Channel Input for both ADC1 and ADC2. |
| | $V_{REF+}$ | AN | — | ADC Positive Voltage Reference Input. |
| | PWM13 | — | CMOS | PWM Output for PWM1. |
| RA4/AN2/T0CKI | RA4 | TTL | CMOS | General Purpose I/O. |
| | AN2 | AN | — | ADC Channel Input for both ADC1 and ADC2. |
| | T0CKI | ST | — | Timer0 Clock Input. |
| RA5/AN21/SS1(1) | RA5 | TTL | CMOS | General Purpose I/O. |
| | AN21 | AN | — | ADC Channel Input for ADC2. |
| | SS1 | ST | — | Slave Select Input for MSSP1. |
| RA6/AN22/ADTRIG/CLKOUT | RA6 | TTL | CMOS | General Purpose I/O. |
| | AN22 | AN | — | ADC Channel Input for ADC2. |
| | ADTRIG | ST | — | ADC Conversion Trigger Input. |
| | CLKOUT | — | CMOS | $F_{OSC}$/4 Output. |
| RA7/AN11/CLKIN | RA7 | TTL | CMOS | General Purpose I/O. |
| | AN11 | AN | — | ADC Channel Input for ADC1. |
| | CLKIN | CMOS | — | External Clock Input (EC mode). |
| RB0/AN16/PWM20/INT | RB0 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN16 | AN | — | ADC Channel Input for ADC1. |
| | PWM20 | — | CMOS | PWM Output for PWM2. |
| | INT | ST | — | External Interrupt. |
| RB1/AN27/PWM21 | RB1 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN27 | AN | — | ADC Channel Input for ADC2. |
| | PWM21 | — | CMOS | PWM Output for PWM2. |
| RB2/AN17/PWM22 | RB2 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN17 | AN | — | ADC Channel Input for ADC1. |
| | PWM22 | — | CMOS | PWM Output for PWM2. |

**TABLE 1-2: PIC16LF1566 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RB3/AN28/PWM23 | RB3 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN28 | AN | — | ADC Channel Input for ADC2. |
| | PWM23 | — | CMOS | PWM Output for PWM2. |
| RB4/AN18/AD1GRDA(1)/AD2GRDA(1) | RB4 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN18 | AN | — | ADC Channel Input for ADC1. |
| | AD1GRDA | — | CMOS | ADC1 Guard Ring Output A. |
| | AD2GRDA | — | CMOS | ADC2 Guard Ring Output A. |
| RB5/AN29/AD1GRDA(1)/AD2GRDA(1)/T1G | RB5 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN29 | AN | — | ADC Channel Input for ADC2. |
| | AD1GRDA | — | CMOS | ADC1 Guard Ring Output A. |
| | AD2GRDA | — | CMOS | ADC2 Guard Ring Output A. |
| | T1G | ST | — | Timer1 Gate Input. |
| RB6/AN19/AD1GRDB(1)/AD2GRDB(1)/ ICSPCLK/ICDCLK | RB6 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN19 | AN | — | ADC Channel Input for ADC1. |
| | AD1GRDB | — | CMOS | ADC1 Guard Ring Output B. |
| | AD2GRDB | — | CMOS | ADC2 Guard Ring Output B. |
| | ICSPCLK | ST | CMOS | ICSP™ Programming Clock. |
| | ICDCLK | ST | CMOS | In-Circuit Debug Clock. |
| RB7/AN40/AD1GRDB(1)/AD2GRDB(1)/ ICSPDAT/ICDDAT | RB7 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN40 | AN | — | ADC Channel Input for ADC2. |
| | AD1GRDB | — | CMOS | ADC1 Guard Ring Output B. |
| | AD2GRDB | — | CMOS | ADC2 Guard Ring Output B. |
| | ICSPDAT | ST | CMOS | ICSP™ Data I/O. |
| | ICDDAT | ST | CMOS | In-Circuit Debug Data. |
| RC0/AN12/T1CKI/SDO2 | RC0 | TTL | CMOS | General Purpose I/O. |
| | AN12 | AN | — | ADC Channel Input for ADC1. |
| | T1CKI | ST | — | Timer1 Clock Input. |
| | SDO2 | — | CMOS | SPI Data Output for MSSP2. |
| RC1/AN23/PWM2/SCL2/SCK2 | RC1 | TTL | CMOS | General Purpose I/O. |
| | AN23 | AN | — | ADC Channel Input for ADC2. |
| | PWM2 | — | CMOS | PWM Output for PWM2. |
| | SCL2 | $I^2C$ | OD | $I^2C$ Clock for MSSP2. |
| | SCK2 | ST | CMOS | SPI Clock for MSSP2. |
| RC2/AN13/PWM1/SDA2/SDI2 | RC2 | TTL | CMOS | General Purpose I/O. |
| | AN13 | AN | — | ADC Channel Input for ADC1. |
| | PWM1 | — | CMOS | PWM Output for PWM1. |
| | SDA2 | $I^2C$ | OD | $I^2C$ Data for MSSP2. |
| | SDI2 | CMOS | — | SPI Data Input for MSSP2. |
| RC3/AN24/SCL1/SCK1 | RC3 | TTL | CMOS | General Purpose I/O. |
| | AN24 | AN | — | ADC Channel Input for ADC2. |
| | SCL1 | $I^2C$ | OD | $I^2C$ Clock for MSSP1. |
| | SCK1 | ST | CMOS | SPI Clock for MSSP1. |

 © 2015-2016 Microchip Technology Inc.

**TABLE 1-2: PIC16LF1566 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RC4/AN14/SDA1/SDI1 | RC4 | TTL | CMOS | General Purpose I/O. |
| | AN14 | AN | — | ADC Channel Input for ADC1. |
| | SDA1 | $I^2C$ | OD | $I^2C$ Data for MSSP1. |
| | SDI1 | CMOS | — | SPI Data Input for MSSP1. |
| RC5/AN25/SDO1/I2CLVL | RC5 | TTL | CMOS | General Purpose I/O. |
| | AN25 | AN | — | ADC Channel Input for ADC2. |
| | SDO1 | — | CMOS | SPI Data Output for MSSP1. |
| | I2CLVL | AN | — | $I^2C$ Voltage Level Input. |
| RC6/AN15/TX/CK | RC6 | TTL | — | General Purpose I/O. |
| | AN15 | AN | — | ADC Channel Input for ADC1. |
| | TX | — | CMOS | EUSART Asynchronous Transmit. |
| | CK | ST | CMOS | EUSART Synchronous Clock. |
| RC7/AN26/RX/DT | RC7 | TTL | CMOS | General Purpose I/O. |
| | AN26 | AN | — | ADC Channel Input for ADC2. |
| | RX | ST | — | EUSART Asynchronous Input. |
| | DT | ST | CMOS | EUSART Synchronous Data. |
| RE3/$V_{PP}$/$\overline{MCLR}$ | RE3 | TTL | — | General Purpose Input with WPU. |
| | $V_{PP}$ | HV | — | Programming Voltage. |
| | $\overline{MCLR}$ | ST | — | Master Clear with Internal Pull-up. |

**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 $I^2C$ = Schmitt Trigger input with $I^2C$ levels
HV = High Voltage XTAL = Crystal

**Note 1:** Alternate pin function selected with the APFCON (Register 11-1) register.

**TABLE 1-3: PIC16LF1567 PINOUT DESCRIPTION**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RA0/AN20/PWM10/SS1(1) | RA0 | TTL | CMOS | General Purpose I/O. |
| | AN20 | AN | — | ADC Channel Input for ADC2. |
| | PWM10 | — | CMOS | PWM Output for PWM1. |
| | SS1 | ST | — | Slave Select Input for MSSP1. |
| RA1/AN10/PWM11/SS2 | RA1 | TTL | CMOS | General Purpose I/O. |
| | AN10 | AN | — | ADC Channel Input for ADC1. |
| | PWM11 | — | CMOS | PWM Output for PWM1. |
| | SS2 | ST | — | Slave Select Input for MSSP2. |
| RA2/AN0/PWM12 | RA2 | TTL | CMOS | General Purpose I/O. |
| | AN0 | AN | — | ADC Channel Input for both ADC1 and ADC2. |
| | $V_{REF-}$ | AN | — | ADC Negative Voltage Reference Input |
| | PWM12 | — | CMOS | PWM Output for PWM1. |
| RA3/AN1/$V_{REF+}$/PWM13 | RA3 | TTL | CMOS | General Purpose I/O. |
| | AN1 | AN | — | ADC Channel Input for both ADC1 and ADC2. |
| | $V_{REF+}$ | AN | — | ADC Positive Voltage Reference Input. |
| | PWM13 | — | CMOS | PWM Output for PWM1. |
| RA4/AN2/T0CKI | RA4 | TTL | CMOS | General Purpose I/O. |
| | AN2 | AN | — | ADC Channel Input for both ADC1 and ADC2. |
| | T0CKI | ST | — | Timer0 Clock Input. |
| RA5/AN21/SS1(1) | RA5 | TTL | CMOS | General Purpose I/O. |
| | AN21 | AN | — | ADC Channel Input for ADC2. |
| | SS1 | ST | — | Slave Select Input for MSSP1. |
| RA6/AN22/ADTRIG/CLKOUT | RA6 | TTL | CMOS | General Purpose I/O. |
| | AN22 | AN | — | ADC Channel Input for ADC2. |
| | ADTRIG | ST | — | ADC Conversion Trigger Input. |
| | CLKOUT | — | CMOS | $F_{OSC}$/4 Output. |
| RA7/AN11/CLKIN | RA7 | TTL | CMOS | General Purpose I/O. |
| | AN11 | AN | — | ADC Channel Input for ADC1. |
| | CLKIN | CMOS | — | External Clock Input (EC mode). |
| RB0/AN16/PWM20/INT | RB0 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN16 | AN | — | ADC Channel Input for ADC1. |
| | PWM20 | — | CMOS | PWM Output for PWM2. |
| | INT | ST | — | External Interrupt. |
| RB1/AN27/PWM21 | RB1 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN27 | AN | — | ADC Channel Input for ADC2. |
| | PWM21 | — | CMOS | PWM Output for PWM2. |
| RB2/AN17/PWM22 | RB2 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN17 | AN | — | ADC Channel Input for ADC1. |
| | PWM22 | — | CMOS | PWM Output for PWM2. |

 © 2015-2016 Microchip Technology Inc.

**TABLE 1-3: PIC16LF1567 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RB3/AN28/PWM23 | RB3 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN28 | AN | — | ADC Channel Input for ADC2. |
| | PWM23 | — | CMOS | PWM Output for PWM2. |
| RB4/AN18/AD1GRDA(1)/AD2GRDA(1) | RB4 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN18 | AN | — | ADC Channel Input for ADC1. |
| | AD1GRDA | — | CMOS | ADC1 Guard Ring Output A. |
| | AD2GRDA | — | CMOS | ADC2 Guard Ring Output A. |
| RB5/AN29/AD1GRDA(1)/AD2GRDA(1)/ T1G | RB5 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN29 | AN | — | ADC Channel Input for ADC2. |
| | AD1GRDA | — | CMOS | ADC1 Guard Ring Output A. |
| | AD2GRDA | — | CMOS | ADC2 Guard Ring Output A. |
| | T1G | ST | — | Timer1 Gate Input |
| RB6/AN19/AD1GRDB(1)/AD2GRDB(1)/ ICSPCLK/ICDCLK | RB6 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN19 | AN | — | ADC Channel Input for ADC1. |
| | AD1GRDB | — | CMOS | ADC1 Guard Ring Output B. |
| | AD2GRDB | — | CMOS | ADC2 Guard Ring Output B. |
| | ICSPCLK | ST | CMOS | ICSP™ Programming Clock. |
| | ICDCLK | ST | CMOS | In-Circuit Debug Clock. |
| RB7/AN40/AD1GRDB(1)/AD2GRDB(1)/ ICSPDAT/ICDDAT | RB7 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN40 | AN | — | ADC Channel Input for ADC2. |
| | AD1GRDB | — | CMOS | ADC1 Guard Ring Output B. |
| | AD2GRDB | — | CMOS | ADC2 Guard Ring Output B. |
| | ICSPDAT | ST | CMOS | ICSP™ Data I/O. |
| | ICDDAT | ST | CMOS | In-Circuit Debug Data. |
| RC0/AN12/T1CKI/SDO2 | RC0 | TTL | CMOS | General Purpose I/O. |
| | AN12 | AN | — | ADC Channel Input for ADC1. |
| | T1CKI | ST | — | Timer1 Clock Input. |
| | SDO2 | — | CMOS | SPI Data Output for MSSP2. |
| RC1/AN23/PWM2/SCL2/SCK2 | RC1 | TTL | CMOS | General Purpose I/O. |
| | AN23 | AN | — | ADC Channel Input for ADC2. |
| | PWM2 | — | CMOS | PWM Output for PWM2. |
| | SCL2 | $I^2C$ | OD | $I^2C$ Clock for MSSP2. |
| | SCK2 | ST | CMOS | SPI Clock for MSSP2. |
| RC2/AN13/PWM1/SDA2/SDI2 | RC2 | TTL | CMOS | General Purpose I/O. |
| | AN13 | AN | — | ADC Channel Input for ADC1. |
| | PWM1 | — | CMOS | PWM Output for PWM1. |
| | SDA2 | $I^2C$ | OD | $I^2C$ Data for MSSP2. |
| | SDI2 | CMOS | — | SPI Data Input for MSSP2. |
| RC3/AN24/SCL1/SCK1 | RC3 | TTL | CMOS | General Purpose I/O. |
| | AN24 | AN | — | ADC Channel Input for ADC2. |
| | SCL1 | $I^2C$ | OD | $I^2C$ Clock for MSSP1. |
| | SCK1 | ST | CMOS | SPI Clock for MSSP1. |

**TABLE 1-3: PIC16LF1567 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RC4/AN14/SDA1/SDI1 | RC4 | TTL | CMOS | General Purpose I/O. |
| | AN14 | AN | — | ADC Channel Input for ADC1. |
| | SDA1 | $I^2C$ | OD | $I^2C$ Data for MSSP1. |
| | SDI1 | CMOS | — | SPI Data Input for MSSP1. |
| RC5/AN25/SDO1/I2CLVL | RC5 | TTL | CMOS | General Purpose I/O. |
| | AN25 | AN | — | ADC Channel Input for ADC2. |
| | SDO1 | — | CMOS | SPI Data Output for MSSP1. |
| | I2CLVL | AN | — | $I^2C$ Voltage Level Input. |
| RC6/AN15/TX/CK | RC6 | TTL | CMOS | General Purpose I/O. |
| | AN15 | AN | — | ADC Channel Input for ADC1. |
| | TX | — | CMOS | EUSART Asynchronous Transmit. |
| | CK | ST | CMOS | EUSART Synchronous Clock. |
| RC7/AN26/RX/DT | RC7 | TTL | CMOS | General Purpose I/O. |
| | AN26 | AN | — | ADC Channel Input for ADC2. |
| | RX | ST | — | EUSART Asynchronous Input. |
| | DT | ST | CMOS | EUSART Synchronous Data. |
| RD0/AN42 | RD0 | TTL | CMOS | General Purpose I/O. |
| | AN42 | AN | — | ADC Channel Input for ADC2. |
| RD1/AN32 | RD1 | TTL | CMOS | General Purpose I/O. |
| | AN32 | AN | — | ADC Channel Input for ADC1. |
| RD2/AN43 | RD2 | TTL | CMOS | General Purpose I/O. |
| | AN43 | AN | — | ADC Channel Input for ADC2. |
| RD3/AN33 | RD3 | TTL | CMOS | General Purpose I/O. |
| | AN33 | AN | — | ADC Channel Input for ADC1. |
| RD4/AN34 | RD4 | TTL | CMOS | General Purpose I/O. |
| | AN34 | AN | — | ADC Channel Input for ADC1. |
| RD5/AN44 | RD5 | TTL | CMOS | General Purpose I/O. |
| | AN44 | AN | — | ADC Channel Input for ADC2. |
| RD6/AN35 | RD6 | TTL | CMOS | General Purpose I/O. |
| | AN35 | AN | — | ADC Channel Input for ADC1. |
| RD7/AN45 | RD7 | TTL | CMOS | General Purpose I/O. |
| | AN45 | AN | — | ADC Channel Input for ADC2. |
| RE0/AN30 | RE0 | TTL | CMOS | General Purpose I/O. |
| | AN30 | AN | — | ADC Channel Input for ADC1. |
| RE1/AN41 | RE1 | TTL | CMOS | General Purpose I/O. |
| | AN41 | AN | — | ADC Channel Input for ADC2. |
| RE2/AN31 | RE2 | TTL | CMOS | General Purpose I/O. |
| | AN31 | AN | — | ADC Channel Input for ADC1. |
| RE3/VPP/$\overline{MCLR}$ | RE3 | TTL | — | General Purpose Input with WPU. |
| | VPP | HV | — | Programming Voltage. |
| | $\overline{MCLR}$ | ST | — | Master Clear with Internal Pull-up. |

 © 2015-2016 Microchip Technology Inc.

**TABLE 1-3: PIC16LF1567 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|

**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; $I^2C$ = Schmitt Trigger input with $I^2C$ levels
HV = High Voltage; XTAL = Crystal

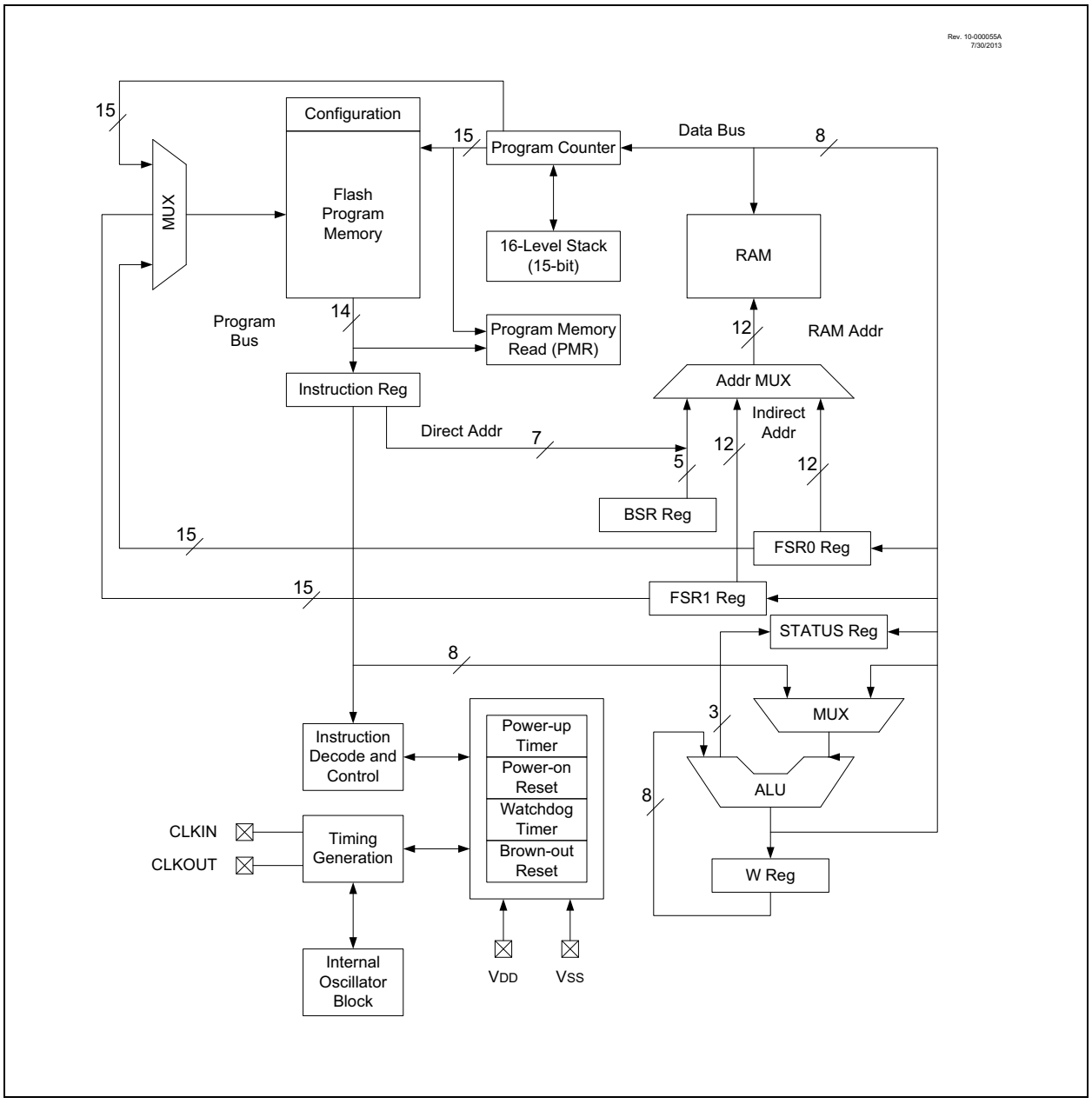
**Note 1:** Alternate pin function selected with the APFCON (Register 11-1) register.

## 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

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




 © 2015-2016 Microchip Technology Inc.

## 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 **Section 7.5 "Automatic Context Saving"**, for more information.

## 2.2 16-Level Stack with Overflow and Underflow

These devices have a hardware 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 **Section 3.4 "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 3.5 "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 24.0 "Instruction Set Summary"** for more details.

## 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

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. 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 Figure 3-1).

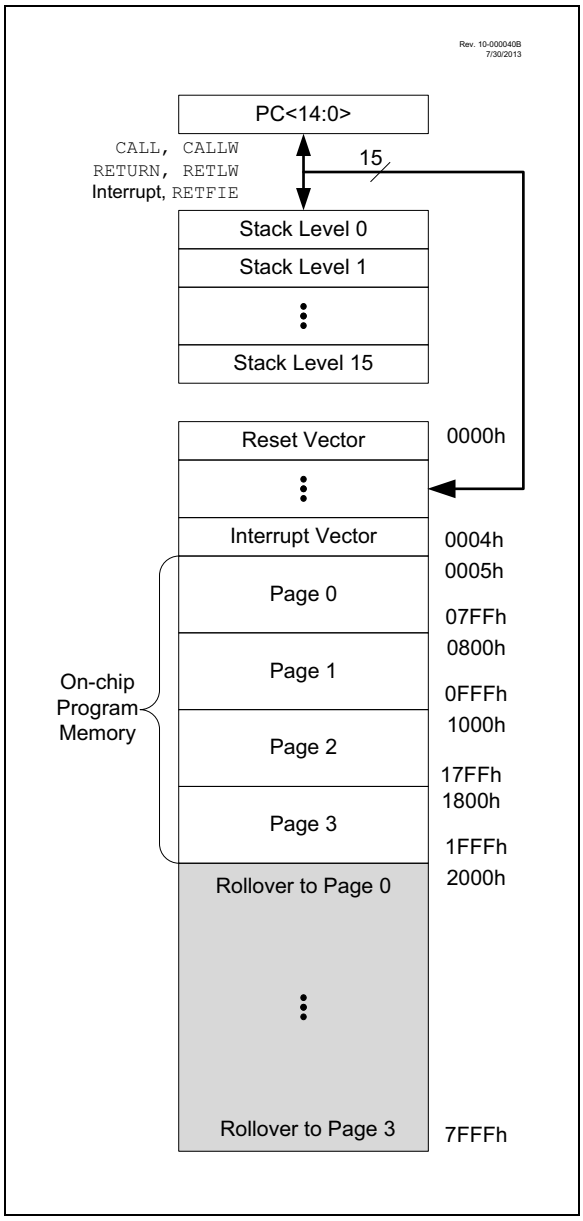
**TABLE 3-1: DEVICE SIZES AND ADDRESSES**

| Device | Program Memory Space (Words) | Last Program Memory Address | High-Endurance Flash Memory Address Range (1) |
|---|---|---|---|
| PIC16LF1566 | 8,192 | 1FFFh | 1F80h-1FFFh |
| PIC16LF1567 | 8,192 | 1FFFh | 1F80h-1FFFh |

**Note 1:** High-endurance Flash applies to low byte of each address in the range.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 3-1: PROGRAM MEMORY MAP AND STACK FOR PIC16LF1566/1567**




### 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 eight 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 operator will set bit 7 if a label points to a location in program memory.

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

```
constants
     DW          DATA0    ; First constant
     DW          DATA1    ; Second constant
     DW          DATA2
     DW          DATA3
my_function
     ;… LOTS OF CODE…
     MOVLW    DATA_INDEX
     ADDLW    LOW constants
     MOVWF    FSR1L
     MOVLW    HIGH constants; MSb is set
automatically
     MOVWF    FSR1H
     BTFSC    STATUS,C     ; carry from ADDLW?
     INCF     FSR1H,f      ; yes
     MOVIW    0[FSR1]
;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 3-2):

- 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 3.5 "Indirect Addressing"** for more information.

Data memory uses a 12-bit address. The upper five bits of the address define the Bank address and the lower seven 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 3-10.

**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 24.0 "Instruction Set Summary"**).

**Note 1:** The C and DC bits operate as $\overline{Borrow}$ and Digit $\overline{Borrow}$ out bits, respectively, in subtraction.


© 2015-2016 Microchip Technology Inc.

**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/$\overline{\text{Digit Borrow}}$ 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/$\overline{\text{Borrow}}$ 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 $\overline{\text{Borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high-order or low-order bit of the source register.

### 3.2.2 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.

### 3.2.3 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).

#### 3.2.3.1 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 3.5.2 "Linear Data Memory"** for more information.

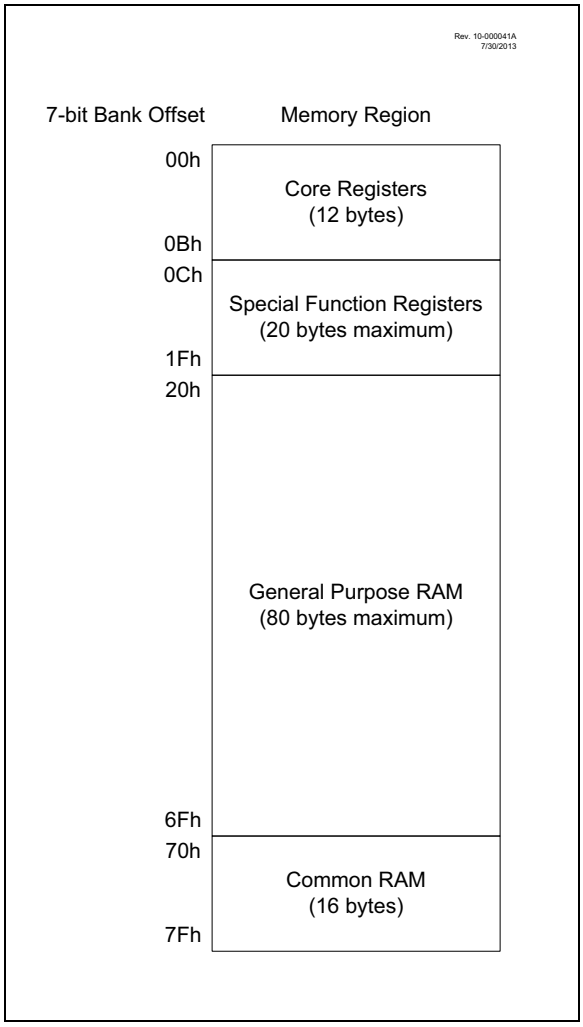
### 3.2.4 COMMON RAM

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

### 3.2.5 DEVICE MEMORY MAPS

The memory maps for PIC16LF1554/1559 are as shown in Table 3-3 through Table 3-7.

**FIGURE 3-2: BANKED MEMORY PARTITIONING**




 © 2015-2016 Microchip Technology Inc.

## TABLE 3-3: PIC16LF1566 MEMORY MAP, BANKS 0-7

| | BANK 0 | | BANK 1 | | BANK 2 | | BANK 3 | | BANK 4 | | BANK 5 | | BANK 6 | | BANK 7 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000h<br>001h<br>002h<br>003h<br>004h<br>005h<br>006h<br>007h<br>008h<br>009h<br>00Ah<br>00Bh | CPU Core Register, see Table 3-2 for specifics | | | | | | | | | | | | | | | |
| 00Ch | PORTA | 08Ch | TRISA | 10Ch | LATA | 18Ch | ANSELA | 20Ch | — | 28Ch | — | 30Ch | — | 38Ch | — | |
| 00Dh | PORTB | 08Dh | TRISB | 10Dh | LATB | 18Dh | ANSELB | 20Dh | WPUB | 28Dh | — | 30Dh | — | 38Dh | — | |
| 00Eh | PORTC | 08Eh | TRISC | 10Eh | LATC | 18Eh | ANSELC | 20Eh | — | 28Eh | — | 30Eh | — | 38Eh | — | |
| 00Fh | — | 08Fh | — | 10Fh | — | 18Fh | — | 20Fh | — | 28Fh | — | 30Fh | — | 38Fh | — | |
| 010h | PORTE | 090h | — | 110h | — | 190h | — | 210h | WPUE | 290h | — | 310h | — | 390h | — | |
| 011h | PIR1 | 091h | PIE1 | 111h | — | 191h | PMADRL | 211h | SSP1BUF | 291h | — | 311h | — | 391h | — | |
| 012h | PIR2 | 092h | PIE2 | 112h | — | 192h | PMADRH | 212h | SSP1ADD | 292h | — | 312h | — | 392h | — | |
| 013h | — | 093h | — | 113h | — | 193h | PMDATL | 213h | SSP1MSK | 293h | — | 313h | — | 393h | — | |
| 014h | — | 094h | — | 114h | — | 194h | PMDATH | 214h | SSP1STAT | 294h | — | 314h | — | 394h | IOCBP | |
| 015h | TMR0 | 095h | OPTION_REG | 115h | — | 195h | PMCON1 | 215h | SSP1CON1 | 295h | — | 315h | — | 395h | IOCBN | |
| 016h | TMR1L | 096h | PCON | 116h | BORCON | 196h | PMCON2 | 216h | SSP1CON2 | 296h | — | 316h | — | 396h | IOCBF | |
| 017h | TMR1H | 097h | WDTCON | 117h | FVRCON | 197h | — | 217h | SSP1CON3 | 297h | — | 317h | — | 397h | — | |
| 018h | T1CON | 098h | — | 118h | — | 198h | — | 218h | SSPLVL | 298h | — | 318h | — | 398h | — | |
| 019h | T1GCON | 099h | OSCCON | 119h | — | 199h | RCREG | 219h | SSP2BUF | 299h | — | 319h | — | 399h | — | |
| 01Ah | TMR2 | 09Ah | OSCSTAT | 11Ah | — | 19Ah | TXREG | 21Ah | SSP2ADD | 29Ah | — | 31Ah | — | 39Ah | — | |
| 01Bh | PR2 | 09Bh | ADRESL/ | 11Bh | — | 19Bh | SPBRGL | 21Bh | SSP2MSK | 29Bh | — | 31Bh | — | 39Bh | — | |
| 01Ch | T2CON | 09Ch | ADRESH | 11Ch | — | 19Ch | SPBRGH | 21Ch | SSP2STAT | 29Ch | — | 31Ch | — | 39Ch | — | |
| 01Dh | — | 09Dh | ADCON0 | 11Dh | APFCON | 19Dh | RCSTA | 21Dh | SSP2CON1 | 29Dh | — | 31Dh | — | 39Dh | — | |
| 01Eh | — | 09Eh | ADCON1 | 11Eh | — | 19Eh | TXSTA | 21Eh | SSP2CON2 | 29Eh | — | 31Eh | — | 39Eh | — | |
| 01Fh | — | 09Fh | ADCON2 | 11Fh | — | 19Fh | BAUDCON | 21Fh | SSP2CON3 | 29Fh | — | 31Fh | — | 39Fh | — | |
| 020h<br>06Fh | General Purpose Register 96 Bytes | 0A0h<br>0EFh | General Purpose Register 80 Bytes | 120h<br>16Fh | General Purpose Register 80 Bytes | 1A0h<br>1EFh | General Purpose Register 80 Bytes | 220h<br>26Fh | General Purpose Register 80 Bytes | 2A0h<br>2EFh | General Purpose Register 80 Bytes | 320h<br>36Fh | General Purpose Register 80 Bytes | 3A0h<br>3EFh | General Purpose Register 80 Bytes | |
| 070h<br>07Fh | | 0F0h<br>0FFh | Accesses 70h – 7Fh | 170h<br>17Fh | Accesses 70h – 7Fh | 1F0h<br>1FFh | Accesses 70h – 7Fh | 270h<br>27Fh | Accesses 70h – 7Fh | 2F0h<br>2FFh | Accesses 70h – 7Fh | 370h<br>37Fh | Accesses 70h – 7Fh | 3F0h<br>3FFh | Accesses 70h – 7Fh | |

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

**Note 1:** These ADC registers are the same as the registers in Bank 14.

© 2015-2016 Microchip Technology Inc.

## TABLE 3-4: PIC16LF1567 MEMORY MAP, BANKS 0-7

| | BANK 0 | | BANK 1 | | BANK 2 | | BANK 3 | | BANK 4 | | BANK 5 | | BANK 6 | | BANK 7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000h | CPU Core Register, see Table 3-2 for specifics | | | | | | | | | | | | | | |
| 001h | | | | | | | | | | | | | | | |
| 002h | | | | | | | | | | | | | | | |
| 003h | | | | | | | | | | | | | | | |
| 004h | | | | | | | | | | | | | | | |
| 005h | | | | | | | | | | | | | | | |
| 006h | | | | | | | | | | | | | | | |
| 007h | | | | | | | | | | | | | | | |
| 008h | | | | | | | | | | | | | | | |
| 009h | | | | | | | | | | | | | | | |
| 00Ah | | | | | | | | | | | | | | | |
| 00Bh | | | | | | | | | | | | | | | |
| 00Ch | PORTA | 08Ch | TRISA | 10Ch | LATA | 18Ch | ANSELA | 20Ch | | 28Ch | — | 30Ch | — | 38Ch | — |
| 00Dh | PORTB | 08Dh | TRISB | 10Dh | LATB | 18Dh | ANSELB | 20Dh | WPUB | 28Dh | — | 30Dh | — | 38Dh | — |
| 00Eh | PORTC | 08Eh | TRISC | 10Eh | LATC | 18Eh | ANSELC | 20Eh | — | 28Eh | — | 30Eh | — | 38Eh | — |
| 00Fh | PORTD | 08Fh | TRISD | 10Fh | LATD | 18Fh | ANSELD | 20Fh | — | 28Fh | — | 30Fh | — | 38Fh | — |
| 010h | PORTE | 090h | TRISE | 110h | LATE | 190h | ANSELE | 210h | WPUE | 290h | — | 310h | — | 390h | — |
| 011h | PIR1 | 091h | PIE1 | 111h | — | 191h | PMADRL | 211h | SSP1BUF | 291h | — | 311h | — | 391h | — |
| 012h | PIR2 | 092h | PIE2 | 112h | — | 192h | PMADRH | 212h | SSP1ADD | 292h | — | 312h | — | 392h | — |
| 013h | — | 093h | — | 113h | — | 193h | PMDATL | 213h | SSP1MSK | 293h | — | 313h | — | 393h | — |
| 014h | — | 094h | — | 114h | — | 194h | PMDATH | 214h | SSP1STAT | 294h | — | 314h | — | 394h | IOCBP |
| 015h | TMR0 | 095h | OPTION_REG | 115h | — | 195h | PMCON1 | 215h | SSP1CON1 | 295h | — | 315h | — | 395h | IOCBN |
| 016h | TMR1L | 096h | PCON | 116h | BORCON | 196h | PMCON2 | 216h | SSP1CON2 | 296h | — | 316h | — | 396h | IOCBF |
| 017h | TMR1H | 097h | WDTCON | 117h | FVRCON | 197h | — | 217h | SSP1CON3 | 297h | — | 317h | — | 397h | — |
| 018h | T1CON | 098h | — | 118h | — | 198h | — | 218h | SSPLVL | 298h | — | 318h | — | 398h | — |
| 019h | T1GCON | 099h | OSCCON | 119h | — | 199h | RCREG | 219h | SSP2BUF | 299h | — | 319h | — | 399h | — |
| 01Ah | TMR2 | 09Ah | OSCSTAT | 11Ah | — | 19Ah | TXREG | 21Ah | SSP2ADD | 29Ah | — | 31Ah | — | 39Ah | — |
| 01Bh | PR2 | 09Bh | ADRESL | 11Bh | — | 19Bh | SPBRGL | 21Bh | SSP2MSK | 29Bh | — | 31Bh | — | 39Bh | — |
| 01Ch | T2CON | 09Ch | ADRESH | 11Ch | — | 19Ch | SPBRGH | 21Ch | SSP2STAT | 29Ch | — | 31Ch | — | 39Ch | — |
| 01Dh | — | 09Dh | ADCON0 | 11Dh | APFCON | 19Dh | RCSTA | 21Dh | SSP2CON1 | 29Dh | — | 31Dh | — | 39Dh | — |
| 01Eh | — | 09Eh | ADCON1 | 11Eh | — | 19Eh | TXSTA | 21Eh | SSP2CON2 | 29Eh | — | 31Eh | — | 39Eh | — |
| 01Fh | — | 09Fh | ADCON2 | 11Fh | — | 19Fh | BAUDCON | 21Fh | SSP2CON3 | 29Fh | — | 31Fh | — | 39Fh | — |
| 020h | General Purpose Register 96 Bytes | 0A0h | General Purpose Register 80 Bytes | 120h | General Purpose Register 80 Bytes | 1A0h | General Purpose Register 80 Bytes | 220h | General Purpose Register 80 Bytes | 2A0h | General Purpose Register 80 Bytes | 320h | General Purpose Register 80 Bytes | 3A0h | General Purpose Register 80 Bytes |
| 06Fh | | 0EFh | | 16Fh | | 1EFh | | 26Fh | | 2EFh | | 36Fh | | 3EFh | |
| 070h | | 0F0h | Accesses 70h – 7Fh | 170h | Accesses 70h – 7Fh | 1F0h | Accesses 70h – 7Fh | 270h | Accesses 70h – 7Fh | 2F0h | Accesses 70h – 7Fh | 370h | Accesses 70h – 7Fh | 3F0h | Accesses 70h – 7Fh |
| 07Fh | | 0FFh | | 17Fh | | 1FFh | | 27Fh | | 2FFh | | 37Fh | | 3FFh | |

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

**Note 1:** These ADC registers are the same as the registers in Bank 14.



Preliminary

© 2015-2016 Microchip Technology Inc.

## TABLE 3-5: PIC16LF1566/1567 MEMORY MAP, BANKS 8-15

| | BANK 8 | | BANK 9 | | BANK 10 | | BANK 11 | | BANK 12 | | BANK 13 | | BANK 14 | | BANK 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 400h | CPU Core Register, see Table 3-2 for specifics | | | | | | | | | | | | | | |
| 401h | | | | | | | | | | | | | | | |
| 402h | | | | | | | | | | | | | | | |
| 403h | | | | | | | | | | | | | | | |
| 404h | | | | | | | | | | | | | | | |
| 405h | | | | | | | | | | | | | | | |
| 406h | | | | | | | | | | | | | | | |
| 407h | | | | | | | | | | | | | | | |
| 408h | | | | | | | | | | | | | | | |
| 409h | | | | | | | | | | | | | | | |
| 40Ah | | | | | | | | | | | | | | | |
| 40Bh | | | | | | | | | | | | | | | |
| 40Ch | — | 48Ch | — | 50Ch | — | 58Ch | — | 60Ch | — | 68Ch | — | 70Ch | — | 78Ch | — |
| 40Dh | — | 48Dh | — | 50Dh | — | 58Dh | — | 60Dh | — | 68Dh | — | 70Dh | — | 78Dh | — |
| 40Eh | — | 48Eh | — | 50Eh | — | 58Eh | — | 60Eh | — | 68Eh | — | 70Eh | — | 78Eh | — |
| 40Fh | — | 48Fh | — | 50Fh | — | 58Fh | — | 60Fh | — | 68Fh | — | 70Fh | — | 78Fh | — |
| 410h | — | 490h | — | 510h | — | 590h | — | 610h | — | 690h | — | 710h | — | 790h | — |
| 411h | — | 491h | — | 511h | — | 591h | — | 611h | PWM1DCL | 691h | ADCTX | 711h | AD1CON0 | 791h | AD2CON0 |
| 412h | — | 492h | — | 512h | — | 592h | — | 612h | PWM1DCH | 692h | AD1TX0 | 712h | ADCOMCON | 792h | — |
| 413h | — | 493h | — | 513h | — | 593h | — | 613h | PWM1CON | 693h | AD1TX1 | 713h | AD1CON2 | 793h | AD2CON2 |
| 414h | — | 494h | — | 514h | — | 594h | — | 614h | PWM2DCL | 694h | AD2TX0 | 714h | AD1CON3 | 794h | AD2CON3 |
| 415h | TMR4 | 495h | — | 515h | — | 595h | — | 615h | PWM2DCH | 695h | AD2TX1 | 715h | ADSTAT | 795h | — |
| 416h | PR4 | 496h | — | 516h | — | 596h | — | 616h | PWM2CON | 696h | — | 716h | AD1PRECON | 796h | AD2PRECON |
| 417h | T4CON | 497h | — | 517h | — | 597h | — | 617h | — | 697h | — | 717h | AD1ACQCON | 797h | AD2ACQCON |
| 418h | — | 498h | — | 518h | — | 598h | — | 618h | — | 698h | — | 718h | AD1GRD | 798h | AD2GRD |
| 419h | — | 499h | — | 519h | — | 599h | — | 619h | — | 699h | — | 719h | AD1CAPCON | 799h | AD2CAPCON |
| 41Ah | — | 49Ah | — | 51Ah | — | 59Ah | — | 61Ah | — | 69Ah | — | 71Ah | AAD1RES0L | 79Ah | AAD2RES0L |
| 41Bh | — | 49Bh | — | 51Bh | — | 59Bh | — | 61Bh | — | 69Bh | — | 71Bh | AAD1RES0H | 79Bh | AAD2RES0H |
| 41Ch | — | 49Ch | — | 51Ch | — | 59Ch | — | 61Ch | — | 69Ch | — | 71Ch | AAD1RES1L | 79Ch | AAD2RES1L |
| 41Dh | — | 49Dh | — | 51Dh | — | 59Dh | — | 61Dh | PWMTMRS | 69Dh | — | 71Dh | AAD1RES1H | 79Dh | AAD2RES1H |
| 41Eh | — | 49Eh | — | 51Eh | — | 59Eh | — | 61Eh | PWM1AOE | 69Eh | — | 71Eh | AD1CH0 | 79Eh | AD2CH0 |
| 41Fh | — | 49Fh | — | 51Fh | — | 59Fh | — | 61Fh | PWM2AOE | 69Fh | — | 71Fh | AD1CH1 | 79Fh | AD2CH1 |
| 420h | General Purpose Register 80 Bytes | 4A0h | General Purpose Register 80 Bytes | 520h | General Purpose Register 80 Bytes | 5A0h | General Purpose Register 80 Bytes | 620h<br>64Fh | General Purpose Register 48 Bytes | 6A0h | Unimplemented Read as '0' | 720h | Unimplemented Read as '0' | 7A0h | Unimplemented Read as '0' |
| 46Fh | | 4EFh | | 56Fh | | 5EFh | | 650h<br>66Fh | Unimplemented Read as '0' | 6EFh | | 76Fh | | 7EFh | |
| 470h | Accesses 70h – 7Fh | 4F0h | Accesses 70h – 7Fh | 570h | Accesses 70h – 7Fh | 5F0h | Accesses 70h – 7Fh | 670h | Accesses 70h – 7Fh | 6F0h | Accesses 70h – 7Fh | 770h | Accesses 70h – 7Fh | 7F0h | Accesses 70h – 7Fh |
| 47Fh | | 4FFh | | 57Fh | | 5FFh | | 67Fh | | 6FFh | | 77Fh | | 7FFh | |

**Note 1:** These ADC registers are the same as the registers in Bank 1.

© 2015-2016 Microchip Technology Inc.

**TABLE 3-6: PIC16LF1566/1567 MEMORY MAP, BANKS 16-23**

| | BANK 16 | | BANK 17 | | BANK 18 | | BANK 19 | | BANK 20 | | BANK 21 | | BANK 22 | | BANK 23 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 800h<br>801h<br>802h<br>803h<br>804h<br>805h<br>806h<br>807h<br>808h<br>809h<br>80Ah<br>80Bh | CPU Core Register, see Table 3-2 for specifics | | | | | | | | | | | | | | | |
| 80Ch | — | 88Ch | — | 90Ch | — | 98Ch | — | A0Ch | — | A8Ch | — | B0Ch | — | B8Ch | — | |
| 80Dh | — | 88Dh | — | 90Dh | — | 98Dh | — | A0Dh | — | A8Dh | — | B0Dh | — | B8Dh | — | |
| 80Eh | — | 88Eh | — | 90Eh | — | 98Eh | — | A0Eh | — | A8Eh | — | B0Eh | — | B8Eh | — | |
| 80Fh | — | 88Fh | — | 90Fh | — | 98Fh | — | A0Fh | — | A8Fh | — | B0Fh | — | B8Fh | — | |
| 810h | — | 890h | — | 910h | — | 990h | — | A10h | — | A90h | — | B10h | — | B90h | — | |
| 811h | — | 891h | — | 911h | — | 991h | — | A11h | — | A91h | — | B11h | — | B91h | — | |
| 812h | — | 892h | — | 912h | — | 992h | — | A12h | — | A92h | — | B12h | — | B92h | — | |
| 813h | — | 893h | — | 913h | — | 993h | — | A13h | — | A93h | — | B13h | — | B93h | — | |
| 814h | — | 894h | — | 914h | — | 994h | — | A14h | — | A94h | — | B14h | — | B94h | — | |
| 815h | — | 895h | — | 915h | — | 995h | — | A15h | — | A95h | — | B15h | — | B95h | — | |
| 816h | — | 896h | — | 916h | — | 996h | — | A16h | — | A96h | — | B16h | — | B96h | — | |
| 817h | — | 897h | — | 917h | — | 997h | — | A17h | — | A97h | — | B17h | — | B97h | — | |
| 818h | — | 898h | — | 918h | — | 998h | — | A18h | — | A98h | — | B18h | — | B98h | — | |
| 819h | — | 899h | — | 919h | — | 999h | — | A19h | — | A99h | — | B19h | — | B99h | — | |
| 81Ah | — | 89Ah | — | 91Ah | — | 99Ah | — | A1Ah | — | A9Ah | — | B1Ah | — | B9Ah | — | |
| 81Bh | — | 89Bh | — | 91Bh | — | 99Bh | — | A1Bh | — | A9Bh | — | B1Bh | — | B9Bh | — | |
| 81Ch | — | 89Ch | — | 91Ch | — | 99Ch | — | A1Ch | — | A9Ch | — | B1Ch | — | B9Ch | — | |
| 81Dh | — | 89Dh | — | 91Dh | — | 99Dh | — | A1Dh | — | A9Dh | — | B1Dh | — | B9Dh | — | |
| 81Eh | — | 89Eh | — | 91Eh | — | 99Eh | — | A1Eh | — | A9Eh | — | B1Eh | — | B9Eh | — | |
| 81Fh | — | 89Fh | — | 91Fh | — | 99Fh | — | A1Fh | — | A9Fh | — | B1Fh | — | B9Fh | — | |
| 820h<br>86Fh | Unimplemented Read as '0' | 8A0h<br>8EFh | Unimplemented Read as '0' | 920h<br>96Fh | Unimplemented Read as '0' | 9A0h<br>9EFh | Unimplemented Read as '0' | A20h<br>A6Fh | Unimplemented Read as '0' | AA0h<br>AEFh | Unimplemented Read as '0' | B20h<br>B6Fh | Unimplemented Read as '0' | BA0h<br>BEFh | Unimplemented Read as '0' | |
| 870h<br>87Fh | Accesses 70h – 7Fh | 8F0h<br>8FFh | Accesses 70h – 7Fh | 970h<br>97Fh | Accesses 70h – 7Fh | 9F0h<br>9FFh | Accesses 70h – 7Fh | A70h<br>A7Fh | Accesses 70h – 7Fh | AF0h<br>AFFh | Accesses 70h – 7Fh | B70h<br>B7Fh | Accesses 70h – 7Fh | BF0h<br>BFFh | Accesses 70h – 7Fh | |



Preliminary

© 2015-2016 Microchip Technology Inc.

**TABLE 3-7: PIC16LF1566/1567 MEMORY MAP, BANKS 24-31**

| | BANK 24 | | BANK 25 | | BANK 26 | | BANK 27 | | BANK 28 | | BANK 29 | | BANK 30 | | BANK 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C00h<br>C01h<br>C02h<br>C03h<br>C04h<br>C05h<br>C06h<br>C07h<br>C08h<br>C09h<br>C0Ah<br>C0Bh | CPU Core Register, see Table 3-2 for specifics | | | | | | | | | | | | | | |
| C0Ch | — | C8Ch | — | D0Ch | — | D8Ch | — | E0Ch | — | E8Ch | — | F0Ch | — | F8Ch | See Table 3-8 and Table 3-9 for register mapping details |
| C0Dh | — | C8Dh | — | D0Dh | — | D8Dh | — | E0Dh | — | E8Dh | — | F0Dh | — | F8Dh | |
| C0Eh | — | C8Eh | — | D0Eh | — | D8Eh | — | E0Eh | — | E8Eh | — | F0Eh | — | F8Eh | |
| C0Fh | — | C8Fh | — | D0Fh | — | D8Fh | — | E0Fh | — | E8Fh | — | F0Fh | — | F8Fh | |
| C10h | — | C90h | — | D10h | — | D90h | — | E10h | — | E90h | — | F10h | — | F90h | |
| C11h | — | C91h | — | D11h | — | D91h | — | E11h | — | E91h | — | F11h | — | F91h | |
| C12h | — | C92h | — | D12h | — | D92h | — | E12h | — | E92h | — | F12h | — | F92h | |
| C13h | — | C93h | — | D13h | — | D93h | — | E13h | — | E93h | — | F13h | — | F93h | |
| C14h | — | C94h | — | D14h | — | D94h | — | E14h | — | E94h | — | F14h | — | F94h | |
| C15h | — | C95h | — | D15h | — | D95h | — | E15h | — | E95h | — | F15h | — | F95h | |
| C16h | — | C96h | — | D16h | — | D96h | — | E16h | — | E96h | — | F16h | — | F96h | |
| C17h | — | C97h | — | D17h | — | D97h | — | E17h | — | E97h | — | F17h | — | F97h | |
| C18h | — | C98h | — | D18h | — | D98h | — | E18h | — | E98h | — | F18h | — | F98h | |
| C19h | — | C99h | — | D19h | — | D99h | — | E19h | — | E99h | — | F19h | — | F99h | |
| C1Ah | — | C9Ah | — | D1Ah | — | D9Ah | — | E1Ah | — | E9Ah | — | F1Ah | — | F9Ah | |
| C1Bh | — | C9Bh | — | D1Bh | — | D9Bh | — | E1Bh | — | E9Bh | — | F1Bh | — | F9Bh | |
| C1Ch | — | C9Ch | — | D1Ch | — | D9Ch | — | E1Ch | — | E9Ch | — | F1Ch | — | F9Ch | |
| C1Dh | — | C9Dh | — | D1Dh | — | D9Dh | — | E1Dh | — | E9Dh | — | F1Dh | — | F9Dh | |
| C1Eh | — | C9Eh | — | D1Eh | — | D9Eh | — | E1Eh | — | E9Eh | — | F1Eh | — | F9Eh | |
| C1Fh | — | C9Fh | — | D1Fh | — | D9Fh | — | E1Fh | — | E9Fh | — | F1Fh | — | F9Fh | |
| C20h<br>C6Fh | Unimplemented Read as '0' | CA0h<br>CEFh | Unimplemented Read as '0' | D20h<br>D6Fh | Unimplemented Read as '0' | DA0h<br>DEFh | Unimplemented Read as '0' | E20h<br>E6Fh | Unimplemented Read as '0' | EA0h<br>EEFh | Unimplemented Read as '0' | F20h<br>F6Fh | Unimplemented Read as '0' | FA0h<br>FEFh | |
| C70h<br>CFFh | Accesses 70h – 7Fh | CF0h<br>CFFh | Accesses 70h – 7Fh | D70h<br>D7Fh | Accesses 70h – 7Fh | DF0h<br>DFFh | Accesses 70h – 7Fh | E70h<br>E7Fh | Accesses 70h – 7Fh | EF0h<br>EFFh | Accesses 70h – 7Fh | F70h<br>F7Fh | Accesses 70h – 7Fh | F0h<br>FFFh | Accesses 70h – 7Fh |

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

© 2015-2016 Microchip Technology Inc.

**TABLE 3-8: PIC16LF1566/1567 MEMORY MAP, BANK 31**

| Address | Bank 31 |
|---|---|
| F80h | INDF0 |
| F81h | INDF1 |
| F82h | PCL |
| F83h | STATUS |
| F84h | FSR0L |
| F85h | FSR0H |
| F86h | FSR1L |
| F87h | FSR1H |
| F88h | BSR |
| F89h | WREG |
| F8Ah | PCLATH |
| F8Bh | INTCON |
| F8Ch | ICDIO |
| F8Dh | ICDCON0 |
| F8Eh | — |
| F8Fh | — |
| F90h | — |
| F91h | ICDSTAT |
| F92h | — |
| F93h | — |
| F94h | — |
| F95h | — |
| F96h | ICDINSTL |
| F97h | ICDINSTH |
| F98h | — |
| F99h | — |
| F9Ah | — |
| F9Bh | — |
| F9Ch | ICDBK0CON |
| F9Dh | ICDBK0L |
| F9Eh | ICDBK0H |
| F9Fh | Unimplemented Read as '0' |
| FE2h | |

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

**TABLE 3-9: PIC16LF1566/1567 MEMORY MAP, BANK 31**

| Address | Bank 31 |
|---|---|
| FE3h | BSRICDSHAD |
| 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 |
| FF0h | Unimplemented Read as '0' |
| FFFh | |

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


© 2015-2016 Microchip Technology Inc.

### 3.2.6 CORE FUNCTION REGISTERS SUMMARY

The Core Function registers listed in Table 3-10 can be addressed from any Bank.

**TABLE 3-10: 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 | — | — | — | $\overline{TO}$ | $\overline{PD}$ | 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 | — | — | — | BSR<4:0> | | | | | ---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 3-11: 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** | | | | | | | | | | | |
| 000h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 001h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 002h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 003h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 004h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 005h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 006h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 007h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 008h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 009h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 00Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 00Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 00Ch | PORTA | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | xxxx xxxx | xxxx xxxx |
| 00Dh | PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | xxxx xxxx | xxxx xxxx |
| 00Eh | PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | xxxx xxxx | xxxx xxxx |
| 00Fh | Unimplemented(4) | | | | | | | | | | |
| 00Fh | PORTD(2) | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxx xxxx | xxxx xxxx |
| 010h | PORTE | — | — | — | — | RE3 | — | — | — | ----x--- | ----x--- |
| 010h | PORTE(2) | | | | | RE3 | RE2 | RE1 | RE0 | ----xxxx | ----xxxx |
| 011h | PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 0000 0000 | 0000 0000 |
| 012h | PIR2 | — | AD2IF | — | — | BCL1IF | BCL2IF | TMR4IF | — | -0-- 000- | -0-- 000- |
| 013h | Unimplemented | | | | | | | | | | |
| 014h | Unimplemented | | | | | | | | | | |
| 015h | TMR0 | TMR0 | | | | | | | | xxxx xxxx | uuuu uuuu |
| 016h | TMR1L | TMR1L | | | | | | | | xxxx xxxx | uuuu uuuu |
| 017h | TMR1H | TMR1H | | | | | | | | xxxx xxxx | uuuu uuuu |
| 018h | T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | — | $\overline{T1SYNC}$ | — | TMR1ON | 0000 -0-0 | uuuu -u-u |
| 019h | T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/$\overline{DONE}$ | T1GVAL | — | T1GSS | 0000 0x-0 | uuuu ux-u |
| 01Ah | TMR2 | TMR2 | | | | | | | | 0000 0000 | 0000 0000 |
| 01Bh | PR2 | PR2 | | | | | | | | 1111 1111 | 1111 1111 |
| 01Ch | T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | -000 0000 | -000 0000 |
| 01Dh | — | Unimplemented | | | | | | | | — | — |
| 01Eh | — | Unimplemented | | | | | | | | — | — |
| 01Fh | — | Unimplemented | | | | | | | | — | — |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

 © 2015-2016 Microchip Technology Inc.

**TABLE 3-11: 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 1** | | | | | | | | | | | |
| 080h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 081h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 082h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 083h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 084h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 085h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 086h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 087h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 088h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 089h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 08Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 08Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 08Ch | TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 1111 1111 | 1111 1111 |
| 08Dh | TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 1111 1111 | 1111 1111 |
| 08Eh | TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 1111 1111 | 1111 1111 |
| 08Fh | Unimplemented | | | | | | | | | | |
| 090h | Unimplemented | | | | | | | | | | |
| 08Fh | TRISD(2) | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 1111 1111 | 1111 1111 |
| 090h | TRISE(2) | — | — | — | — | —(5) | TRISE2 | TRISE1 | TRISE0 | ----1111 | ----1111 |
| 091h | PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 0000 0000 | 0000 0000 |
| 092h | PIE2 | — | AD2IE | — | — | BCL1IE | BCL2IE | TMR4IE | — | -0-- 000- | -0-- 000- |
| 093h | Unimplemented | | | | | | | | | | |
| 094h | Unimplemented | | | | | | | | | | |
| 095h | OPTION_REG | $\overline{WPUEN}$ | INTEDG | TMR0CS | TMR0SE | PSA | PS<2:0> | | | 1111 1111 | 1111 1111 |
| 096h | PCON | STKOVF | STKUNF | — | $\overline{RWDT}$ | $\overline{RMCLR}$ | $\overline{RI}$ | $\overline{POR}$ | $\overline{BOR}$ | 00-1 11qq | qq-q qquu |
| 097h | WDTCON | — | — | WDTPS<4:0> | | | | | SWDTEN | --01 0110 | --01 0110 |
| 098h | Unimplemented | | | | | | | | | | |
| 099h | OSCCON | SPLLEN | IRCF<3:0> | | | | — | SCS<1:0> | | 0011 1-00 | 0011 1-00 |
| 09Ah | OSCSTAT | — | PLLSR | — | HFIOFR | — | — | LFIOFR | HFIOFS | -0-0 --0q | -q-q --0q |
| 09Bh | ADRESL/ AD1RES0L(3) | ADRESL | | | | | | | | xxxx xxxx | uuuu uuuu |
| 09Ch | ADRESH/ AD1RES0H(3) | ADRESH | | | | | | | | xxxx xxxx | uuuu uuuu |
| 09Dh | ADCON0/ AD1CON0(3) | CHS15 | CHS14 | CHS13 | CHS12 | CHS11 | CHS10 | GO/$\overline{DONE}$1 | AD1ON | 0000 0000 | 0000 0000 |
| 09Eh | ADCON1/ ADCOMCON(3) | ADFM | ADCS<2:0> | | | ADNREF | GO/ $\overline{DONE}$_ALL | ADPREF<1:0> | | 0000 0000 | 0000 0000 |
| 09Fh | ADCON2/ AD1CON2(3) | — | TRIGSEL<2:0> | | | — | — | — | — | -000 ---- | -000 ---- |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

**TABLE 3-11: 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** | | | | | | | | | | | |
| 100h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 101h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 102h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 103h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 104h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 105h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 106h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 107h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 108h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 109h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 10Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 10Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 10Ch | LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | xxxx xxxx | uuuu uuuu |
| 10Dh | LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | xxxx xxxx | uuuu uuuu |
| 10Eh | LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | xxxx xxxx | uuuu uuuu |
| 10Fh | Unimplemented | | | | | | | | | | |
| | LATD(2) | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | xxxx xxxx | uuuu uuuu |
| 110h | Unimplemented | | | | | | | | | | |
| | LATE(2) | — | — | — | — | — | LATE2 | LATE1 | LATE0 | -----xxx | ---- -uuu |
| 111h | — | Unimplemented | | | | | | | | — | — |
| 112h | — | Unimplemented | | | | | | | | — | — |
| 113h | — | Unimplemented | | | | | | | | — | — |
| 114h | — | Unimplemented | | | | | | | | — | — |
| 115h | — | Unimplemented | | | | | | | | — | — |
| 116h | BORCON | SBOREN | BORFS | — | — | — | — | — | BORRDY | 10-- ---q | uu-- ---u |
| 117h | FVRCON | FVREN | FVRRDY | TSEN | TSRNG | — | — | ADFVR<1:0> | | 0q00 --00 | 0q00 --00 |
| 118h | — | Unimplemented | | | | | | | | — | — |
| 119h | — | Unimplemented | | | | | | | | — | — |
| 11Ah | — | Unimplemented | | | | | | | | — | — |
| 11Bh | — | Unimplemented | | | | | | | | — | — |
| 11Ch | — | Unimplemented | | | | | | | | — | — |
| 11Dh | APFCON | — | — | SSSEL | — | — | — | GRDBSEL | GRDASEL | --0---00 | --0---00 |
| 11Eh | — | Unimplemented | | | | | | | | — | — |
| 11Fh | — | Unimplemented | | | | | | | | — | — |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

 © 2015-2016 Microchip Technology Inc.

**TABLE 3-11: 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 3** | | | | | | | | | | | |
| 180h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 181h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 182h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 183h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 184h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 185h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 186h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 187h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 188h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 189h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 18Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 18Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 18Ch | ANSELA | ANSA7 | ANSA6 | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 1111 1111 | 1111 1111 |
| 18Dh | ANSELB | ANSB7 | ANSB6 | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 1111 1111 | 1111 1111 |
| 18Eh | ANSELC(3) | ANSC7 | ANSC6 | ANSC5 | ANSC4 | ANSC3 | ANSC2 | ANSC1 | ANSC0 | 1111 1111 | 1111 1111 |
| 18Fh | Unimplemented | | | | | | | | | | |
| | ANSELD(2) | ANSD7 | ANSD6 | ANSD5 | ANSD4 | ANSD3 | ANSD2 | ANSD1 | ANSD0 | 1111 1111 | 1111 1111 |
| 190h | Unimplemented | | | | | | | | | | |
| | ANSELE(2) | — | — | — | — | — | ANSE2 | ANSE1 | ANSE0 | -----111 | -----111 |
| 191h | PMADRL | PMADRL | | | | | | | | 0000 0000 | 0000 0000 |
| 192h | PMADRH | — | PMADRH | | | | | | | 1000 0000 | 1000 0000 |
| 193h | PMDATL | PMDATL | | | | | | | | xxxx xxxx | uuuu uuuu |
| 194h | PMDATH | — | — | PMDATH | | | | | | --xx xxxx | --uu uuuu |
| 195h | PMCON1 | — | CFGS | LWLO | FREE | WRERR | WREN | WR | RD | -000 x000 | -000 q000 |
| 196h | PMCON2 | Program Memory Control Register 2 | | | | | | | | 0000 0000 | 0000 0000 |
| 197h | — | Unimplemented | | | | | | | | — | — |
| 198h | — | Unimplemented | | | | | | | | — | — |
| 199h | RCREG | RCREG | | | | | | | | xxxx xxxx | xxxx xxxx |
| 19Ah | TXREG | TXREG | | | | | | | | xxxx xxxx | xxxx xxxx |
| 19Bh | SPBRGL | BRG<7:0> | | | | | | | | xxxx xxxx | xxxx xxxx |
| 19Ch | SPBRGH | BRG<15:8> | | | | | | | | xxxx xxxx | xxxx xxxx |
| 19Dh | RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000x |
| 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 = depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations unimplemented, read as '0'.

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

**TABLE 3-11: 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** | | | | | | | | | | | |
| 200h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 201h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 202h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 203h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 204h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 205h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 206h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 207h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 208h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 209h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 20Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 20Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 20Ch | — | Unimplemented | | | | | | | | — | — |
| 20Dh | WPUB(3) | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 | 1111 1111 | 1111 1111 |
| 20Eh | — | Unimplemented | | | | | | | | — | — |
| 20Fh | — | Unimplemented | | | | | | | | — | — |
| 210h | WPUE(3) | — | — | — | — | WPUE3 | — | — | — | ----1--- | ----1--- |
| 211h | SSP1BUF | MSSPx Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 212h | SSP1ADD | ADD | | | | | | | | 0000 0000 | 0000 0000 |
| 213h | SSP1MSK | MSK | | | | | | | | 1111 1111 | 1111 1111 |
| 214h | SSP1STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 0000 0000 | 0000 0000 |
| 215h | SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 0000 0000 |
| 216h | SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | 0000 0000 |
| 217h | SSP1CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 0000 0000 | 0000 0000 |
| 218h | SSPLVL | — | — | — | S2ILS | — | — | — | S1ILS | ---0---0 | – |
| 219h | SSP2BUF | MSSPx Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | uuuu uuuu |
| 21Ah | SSP2ADD | ADD | | | | | | | | 0000 0000 | 0000 0000 |
| 21Bh | SSP2MSK | MSK | | | | | | | | 1111 1111 | 1111 1111 |
| 21Ch | SSP2STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 0000 0000 | 0000 0000 |
| 21Dh | SSP2CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 0000 0000 |
| 21Eh | SSP2CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 | 0000 0000 |
| 21Fh | SSP2CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 0000 0000 | 0000 0000 |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.


© 2015-2016 Microchip Technology Inc.

**TABLE 3-11: 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 5** | | | | | | | | | | | |
| 280h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 281h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 282h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 283h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 284h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 285h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 286h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 287h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 288h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 289h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 28Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 28Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 28Ch | — | Unimplemented | | | | | | | | — | — |
| 28Dh | — | Unimplemented | | | | | | | | — | — |
| 28Eh | — | Unimplemented | | | | | | | | — | — |
| 28Fh | — | Unimplemented | | | | | | | | — | — |
| 290h | — | Unimplemented | | | | | | | | — | — |
| 291h | — | Unimplemented | | | | | | | | — | — |
| 292h | — | Unimplemented | | | | | | | | — | — |
| 293h | — | Unimplemented | | | | | | | | — | — |
| 294h | — | Unimplemented | | | | | | | | — | — |
| 295h | — | Unimplemented | | | | | | | | — | — |
| 296h | — | Unimplemented | | | | | | | | — | — |
| 297h | — | Unimplemented | | | | | | | | — | — |
| 298h | — | Unimplemented | | | | | | | | — | — |
| 299h | — | Unimplemented | | | | | | | | — | — |
| 29Ah | — | Unimplemented | | | | | | | | — | — |
| 29Bh | — | Unimplemented | | | | | | | | — | — |
| 29Ch | — | Unimplemented | | | | | | | | — | — |
| 29Dh | — | Unimplemented | | | | | | | | — | — |
| 29Eh | — | Unimplemented | | | | | | | | — | — |
| 29Fh | — | Unimplemented | | | | | | | | — | — |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

**TABLE 3-11: 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 6** | | | | | | | | | | | |
| 300h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 301h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 302h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 303h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 304h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 305h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 306h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 307h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 308h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 309h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 30Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 30Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 30Ch | — | Unimplemented | | | | | | | | — | — |
| 30Dh | — | Unimplemented | | | | | | | | — | — |
| 30Eh | — | Unimplemented | | | | | | | | — | — |
| 30Fh | — | Unimplemented | | | | | | | | — | — |
| 310h | — | Unimplemented | | | | | | | | — | — |
| 311h | — | Unimplemented | | | | | | | | — | — |
| 312h | — | Unimplemented | | | | | | | | — | — |
| 313h | — | Unimplemented | | | | | | | | — | — |
| 314h | — | Unimplemented | | | | | | | | — | — |
| 315h | — | Unimplemented | | | | | | | | — | — |
| 316h | — | Unimplemented | | | | | | | | — | — |
| 317h | — | Unimplemented | | | | | | | | — | — |
| 318h | — | Unimplemented | | | | | | | | — | — |
| 319h | — | Unimplemented | | | | | | | | — | — |
| 31Ah | — | Unimplemented | | | | | | | | — | — |
| 31Bh | — | Unimplemented | | | | | | | | — | — |
| 31Ch | — | Unimplemented | | | | | | | | — | — |
| 31Dh | — | Unimplemented | | | | | | | | — | — |
| 31Eh | — | Unimplemented | | | | | | | | — | — |
| 31Fh | — | Unimplemented | | | | | | | | — | — |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

 © 2015-2016 Microchip Technology Inc.

**TABLE 3-11: 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** | | | | | | | | | | | |
| 380h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 381h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 382h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 383h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 384h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 385h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 386h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 387h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 388h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 389h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 38Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 38Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 38Ch | — | Unimplemented | | | | | | | | — | — |
| 38Dh | — | Unimplemented | | | | | | | | — | — |
| 38Eh | — | Unimplemented | | | | | | | | — | — |
| 38Fh | — | Unimplemented | | | | | | | | — | — |
| 390h | — | Unimplemented | | | | | | | | — | — |
| 391h | IOCBP | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 | 0000 0000 | 0000 0000 |
| 392h | IOCBN | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 | 0000 0000 | 0000 0000 |
| 393h | IOCBF | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 | 0000 0000 | 0000 0000 |
| 394h | — | Unimplemented | | | | | | | | — | — |
| 395h | — | Unimplemented | | | | | | | | — | — |
| 396h | — | Unimplemented | | | | | | | | — | — |
| 397h | — | Unimplemented | | | | | | | | — | — |
| 398h | — | Unimplemented | | | | | | | | — | — |
| 399h | — | Unimplemented | | | | | | | | — | — |
| 39Ah | — | Unimplemented | | | | | | | | — | — |
| 39Bh | — | Unimplemented | | | | | | | | — | — |
| 39Ch | — | Unimplemented | | | | | | | | — | — |
| 39Dh | — | Unimplemented | | | | | | | | — | — |
| 39Eh | — | Unimplemented | | | | | | | | — | — |
| 39Fh | — | Unimplemented | | | | | | | | — | — |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

**TABLE 3-11: 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 8** | | | | | | | | | | | |
| 400h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 401h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 402h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 403h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 404h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 405h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 406h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 407h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 408h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 409h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 40Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 40Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 40Ch to 414h | — | Unimplemented | | | | | | | | — | — |
| 415h | TMR4 | TMR4 | | | | | | | | 0000 0000 | 0000 0000 |
| 416h | PR4 | PR4 | | | | | | | | 11111111 | 11111111 |
| 417h | T4CON | — | T4OUTPS | | | | TMR4ON | T4CKPS | | -000 0000 | -000 0000 |
| 418h to 41Fh | — | Unimplemented | | | | | | | | — | — |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

 © 2015-2016 Microchip Technology Inc.

**TABLE 3-11: 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Banks 9-11** | | | | | | | | | | | |
| x00h/ x80h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| x00h/ x81h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| x02h/ x82h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| x03h/ x83h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| x04h/ x84h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| x05h/ x85h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| x06h/ x86h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| x07h/ x87h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| x08h/ x88h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| x09h/ x89h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| x0Ah/ x8Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| x0Bh/ x8Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 000u |
| x0Ch/ x8Ch — x1Fh/ x9Fh | — | Unimplemented | | | | | | | | — | — |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

**TABLE 3-11: 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 12** | | | | | | | | | | | |
| 600h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 601h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 602h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 603h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 604h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 605h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 606h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 607h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 608h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 609h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 60Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 60Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 60Ch | — | Unimplemented | | | | | | | | — | — |
| 60Dh | — | Unimplemented | | | | | | | | — | — |
| 60Eh | — | Unimplemented | | | | | | | | — | — |
| 60Fh | — | Unimplemented | | | | | | | | — | — |
| 610h | — | Unimplemented | | | | | | | | — | — |
| 611h | PWM1DCL | PWM1DCL<7:6> | | — | — | — | — | — | — | xx-- ---- | uu-- ---- |
| 612h | PWM1DCH | PWM1DCH | | | | | | | | xxxx xxxx | uuuu uuuu |
| 613h | PWM1CON | PWM1EN | PWM1OE | PWM1OUT | PWM1POL | — | — | — | — | 00x0 ---- | 00x0 ---- |
| 614h | PWM2DCL | PWM2DCL<7:6> | | — | — | — | — | — | — | xx-- ---- | uu-- ---- |
| 615h | PWM2DCH | PWM2DCH | | | | | | | | xxxx xxxx | uuuu uuuu |
| 616h | PWM2CON | PWM2EN | PWM2OE | PWM2OUT | PWM2POL | — | — | — | — | 00x0 ---- | 0x00 ---- |
| 617h | — | Unimplemented | | | | | | | | — | — |
| 618h | — | Unimplemented | | | | | | | | — | — |
| 619h | — | Unimplemented | | | | | | | | — | — |
| 61Ah | — | Unimplemented | | | | | | | | — | — |
| 61Bh | — | Unimplemented | | | | | | | | — | — |
| 61Ch | — | Unimplemented | | | | | | | | — | — |
| 61Dh | PWMTMRS | | | | | | P2TSEL | | P1TSEL | -----0-0 | -----0-0 |
| 61Eh | PWM1AOE | | | | | PWM1OE | | | | ----0000 | ----0000 |
| 61Fh | PWM2AOE | | | | | PWM2OE | | | | ----0000 | ----0000 |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

 © 2015-2016 Microchip Technology Inc.

**TABLE 3-11: 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Banks 13** | | | | | | | | | | | |
| 680h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 681h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 682h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 683h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 684h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 685h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 686h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 687h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 688h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 689h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 68Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 68Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 000u |
| 68Ch to 690h | — | Unimplemented | | | | | | | | — | — |
| 691h | ADCTX | — | A2TX2 | A2TX1 | A2TX0 | — | A1TX2 | A1TX1 | A1TX0 | -xxx-xxx | -uuu-uuu |
| 692h | AD1TX0 | TX17 | TX16 | TX15 | TX14 | TX13 | TX12 | TX11 | TX10 | xxxx xxxx | uuuu uuuu |
| 693h | AD1TX1 | — | — | — | — | — | — | TX19 | TX18 | ------xx | ------uu |
| | AD1TX1(2) | TX35 | TX34 | TX33 | TX32 | TX31 | TX30 | TX19 | TX18 | xxxx xxxx | uuuu uuuu |
| 694h | AD2TX0 | TX27 | TX26 | TX25 | TX24 | TX23 | TX22 | TX21 | TX20 | xxxx xxxx | uuuu uuuu |
| 695h | AD2TX1 | — | — | — | — | — | TX40 | TX29 | TX28 | -----xxx | -----uuu |
| | AD2TX1(2) | TX45 | TX44 | TX43 | TX42 | TX41 | TX40 | TX29 | TX28 | xxxx xxxx | uuuu uuuu |
| 696h to 69Fh | — | | | | | | | | | — | — |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

**TABLE 3-11: 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 14** | | | | | | | | | | | |
| 700h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 701h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 702h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 703h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 704h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 705h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 706h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 707h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 708h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 709h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 70Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 70Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 000u |
| 70Ch | — | Unimplemented | | | | | | | | — | — |
| 70Dh | — | Unimplemented | | | | | | | | — | — |
| 70Eh | — | Unimplemented | | | | | | | | — | — |
| 70Fh | — | Unimplemented | | | | | | | | — | — |
| 710h | — | Unimplemented | | | | | | | | — | — |
| 711h | AD1CON0 | CHS15 | CHS14 | CHS13 | CHS12 | CHS11 | CHS10 | GO/$\overline{DONE}$1 | AD1ON | 0000 0000 | 0000 0000 |
| 712h | ADCOMCON | ADFM | ADCS<2:0> | | | ADNREF | GO/$\overline{DONE}$_ALL | ADPREF<1:0> | | 0000 0000 | 0000 0000 |
| 713h | AD1CON2 | — | TRIGSEL<2:0> | | | — | — | — | — | -000 ---- | -000 ---- |
| 714h | AD1CON3 | AD1EPPOL | AD1IPPOL | — | — | — | — | AD1IPEN | AD1DSEN | 00-- --00 | 00-- --00 |
| 715h | ADSTAT | — | AD2CONV | AD2STG<1:0> | | — | AD1CONV | AD1STG | | -000 -000 | -000 -000 |
| 716h | AD1PRECON | — | ADPRE<6:0> | | | | | | | -000 0000 | -000 0000 |
| 717h | AD1ACQCON | — | ADACQ<6:0> | | | | | | | -000 0000 | -000 0000 |
| 718h | AD1GRD | GRD1BOE | GRD1AOE | GRD1POL | — | — | — | — | TX1POL | 000- ---0 | 000- ---0 |
| 719h | AD1CAPCON | — | — | — | — | ADDCAP<3:0> | | | | ---- 0000 | ---- 0000 |
| 71Ah | AAD1RES0L | ADRESL | | | | | | | | xxxx xxxx | uuuu uuuu |
| 71Bh | AAD1RES0H | ADRESH | | | | | | | | xxxx xxxx | uuuu uuuu |
| 71Ch | AAD1RES1L | ADRESL | | | | | | | | xxxx xxxx | uuuu uuuu |
| 71Dh | AAD1RES1H | ADRESH | | | | | | | | xxxx xxxx | uuuu uuuu |
| 71Eh | AD1CH0 | CHS17 | CH16 | CH15 | CH14 | CH13 | CH12 | CH11 | CH10 | 0000 0000 | 0000 0000 |
| 71Fh | AD1CH1 | — | — | — | — | — | — | CH19 | CH18 | ------00 | ------00 |
| | AD1CH1(2) | CH35 | CH34 | CH33 | CH32 | CH31 | CH30 | CH19 | CH18 | 0000 0000 | 0000 0000 |

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

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

**2:** PIC16LF1567.

**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.

**4:** PIC16LF1566 only.

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

 © 2015-2016 Microchip Technology Inc.

**TABLE 3-11: 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 15** | | | | | | | | | | | |
| 780h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 781h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 782h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 783h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 784h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 785h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 786h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 787h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 788h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 789h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 78Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 78Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 000u |
| 78Ch | — | Unimplemented | | | | | | | | — | — |
| 78Dh | — | Unimplemented | | | | | | | | — | — |
| 78Eh | — | Unimplemented | | | | | | | | — | — |
| 78Fh | — | Unimplemented | | | | | | | | — | — |
| 790h | — | Unimplemented | | | | | | | | — | — |
| 791h | AD2CON0 | CHS<5:0> | | | | | | GO/$\overline{DONE}$2 | AD2CON | 0000 0000 | 0000 0000 |
| 792h | — | Unimplemented | | | | | | | | — | — |
| 793h | AD2CON2 | — | TRIGSEL<2:0> | | | — | — | — | — | -000 ---- | -000 ---- |
| 794h | AD2CON3 | AD2EPPOL | AD2IPPOL | — | — | — | — | AD2IPEN | AD2DSEN | 00-- --00 | 00-- --00 |
| 795h | — | Unimplemented | | | | | | | | — | — |
| 796h | AD2PRECON | — | ADPRE<6:0> | | | | | | | -000 0000 | -000 0000 |
| 797h | AD2ACQCON | — | ADACQ<6:0> | | | | | | | -000 0000 | -000 0000 |
| 798h | AD2GRD | GRD2BOE | GRD2AOE | GRD2POL | — | — | — | — | TX2POL | 000- ---x | 000- ---u |
| 799h | AD2CAPCON | — | — | — | — | ADD2CAP<3:0> | | | | ---- 0000 | ---- 0000 |
| 79Ah | AAD2RES0L | ADRESL | | | | | | | | xxxx xxxx | uuuu uuuu |
| 79Bh | AAD2RES0H | ADRESH | | | | | | | | xxxx xxxx | uuuu uuuu |
| 79Ch | AAD2RES1L | ADRESL | | | | | | | | xxxx xxxx | uuuu uuuu |
| 79Dh | AAD2RES1H | ADRESH | | | | | | | | xxxx xxxx | uuuu uuuu |
| 79Eh | AD2CH0 | CH27 | CH26 | CH25 | CH24 | CH23 | CH22 | CH21 | CH20 | 0000 0000 | 0000 0000 |
| 79Fh | AD2CH1 | — | — | — | — | — | CH40 | CH29 | CH28 | -----000 | -----000 |
| | AD2CH1(2) | CH45 | CH44 | CH43 | CH42 | CH41 | CH40 | CH29 | CH28 | 00000000 | 00000000 |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

**TABLE 3-11: 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Banks 16-30** | | | | | | | | | | | |
| x00h/ x80h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| x00h/ x81h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| x02h/ x82h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| x03h/ x83h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| x04h/ x84h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| x05h/ x85h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| x06h/ x86h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| x07h/ x87h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| x08h/ x88h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| x09h/ x89h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| x0Ah/ x8Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| x0Bh/ x8Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

 © 2015-2016 Microchip Technology Inc.

**TABLE 3-11: 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** | | | | | | | | | | | |
| F80h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| F81h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| F82h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| F83h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| F84h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| F85h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| F86h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| F87h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| F88h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| F89h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| F8Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| F8Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| F8Ch | ICDIO | PORT_ ICDDAT | PORT_ ICDCLK | LAT_ ICDDAT | LAT_ ICDCLK | TRIS_ ICDDAT | TRIS_ ICDCLK | — | — | xxxxxx-- | |
| F8Dh | ICDCON0 | INBUG | FREEZ | SSTEP | — | DBGINEX | — | — | RSTVEC | xxx-x--x | |
| F8Eh to F90h | — | Unimplemented | | | | | | | | — | — |
| F91h | ICDSTAT | TRP1HLTF | TRP0HLTF | — | — | — | — | USRHLTF | — | xx----x- | |
| F92h to F95h | — | Unimplemented | | | | | | | | — | — |
| F96h | ICDINSTL | DBGIN7 | DBGIN6 | DBGIN5 | DBGIN4 | DBGIN3 | DBGIN2 | DBGIN1 | DBGIN0 | xxxxxxxx | |
| F97h | ICDINSTH | — | — | DBGIN13 | DBGIN12 | DBGIN11 | DBGIN10 | DBGIN9 | DBGIN8 | --xxxxxx | |
| F98h to F9Bh | — | Unimplemented | | | | | | | | — | — |
| F9Ch | ICDBK0CON | BKEN | — | — | — | — | — | — | BKHLT | x------x | |
| F9Dh | ICDBK0L | BKA7 | BKA6 | BKA5 | BKA4 | BKA3 | BKA2 | BKA1 | BKA0 | xxxxxxxx | |
| F9Eh | ICDBK0H | — | BKA14 | BKA13 | BKA12 | BKA11 | BKA10 | BKA9 | BKA8 | -xxxxxxx | |
| F9Fh | — | Unimplemented | | | | | | | | — | — |
| FA0h to FBFh | — | Unimplemented | | | | | | | | — | — |
| FC0h to FCFh | — | Unimplemented | | | | | | | | — | — |
| FD0h to FE2h | — | Unimplemented | | | | | | | | — | — |
| FE3h | BSRICDSHAD | — | — | — | BSR_ICDSHAD | | | | | ---xxxxx | – |
| FE4h | STATUS_- SHAD | — | — | — | — | — | Z_SHAD | DC_SHAD | C_SHAD | ---- -xxx | ---- -uuu |
| FE5h | WREG_SHAD | WREG_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu |
| FE6h | BSR_SHAD | — | — | — | BSR_SHAD | | | | | ---x xxxx | ---u uuuu |
| FE7h | PCLATH_- SHAD | — | PCLATH_SHAD | | | | | | | -xxx xxxx | uuuu uuuu |
| FE8h | FSR0L_SHAD | FSR0L_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu |
| FE9h | FSR0H_SHAD | FSR0H_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu |
| FEAh | FSR1L_SHAD | FSR1L_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu |
| FEBh | FSR1H_SHAD | FSR1H_SHAD | | | | | | | | xxxx xxxx | uuuu uuuu |

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

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

**TABLE 3-11: 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| FECh | — | Unimplemented | | | | | | | | — | — |
| FEDh | STKPTR | — | — | — | STKPTR | | | | | ---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 = depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations unimplemented, read as '0'.

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

**2:** PIC16LF1567.

**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.

**4:** PIC16LF1566 only.

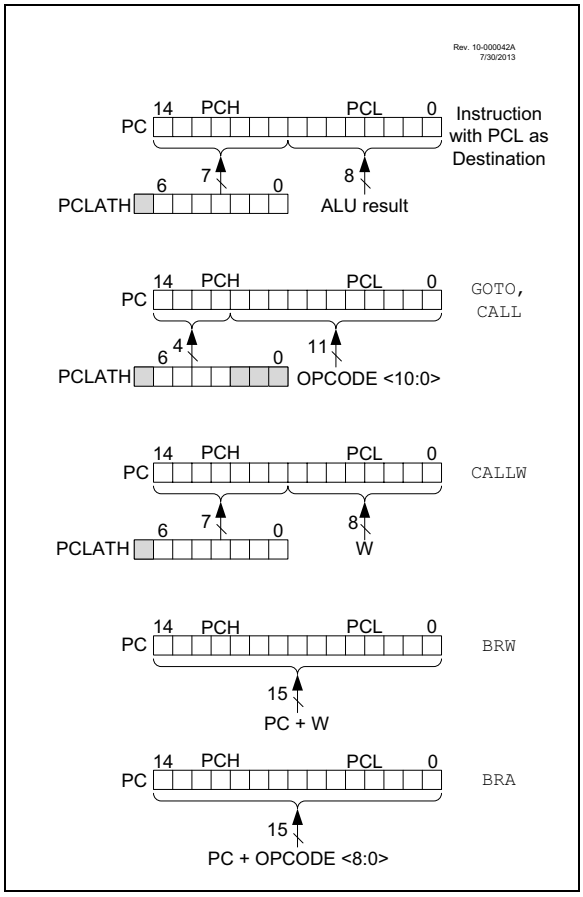
**5:** Unimplemented, read as '1'.

 © 2015-2016 Microchip Technology Inc.

## 3.3 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 3-3 shows the five situations for the loading of the PC.

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




### 3.3.1 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 seven bits to the PCLATH register. When the lower eight 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.

### 3.3.2 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).

### 3.3.3 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.

### 3.3.4 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.

## 3.4 Stack

All devices have a 16-level x 15-bit wide hardware stack (refer to Figure 3-4 through Figure 3-7). 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 1:** | 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. |
|---|---|

### 3.4.1 ACCESSING THE STACK

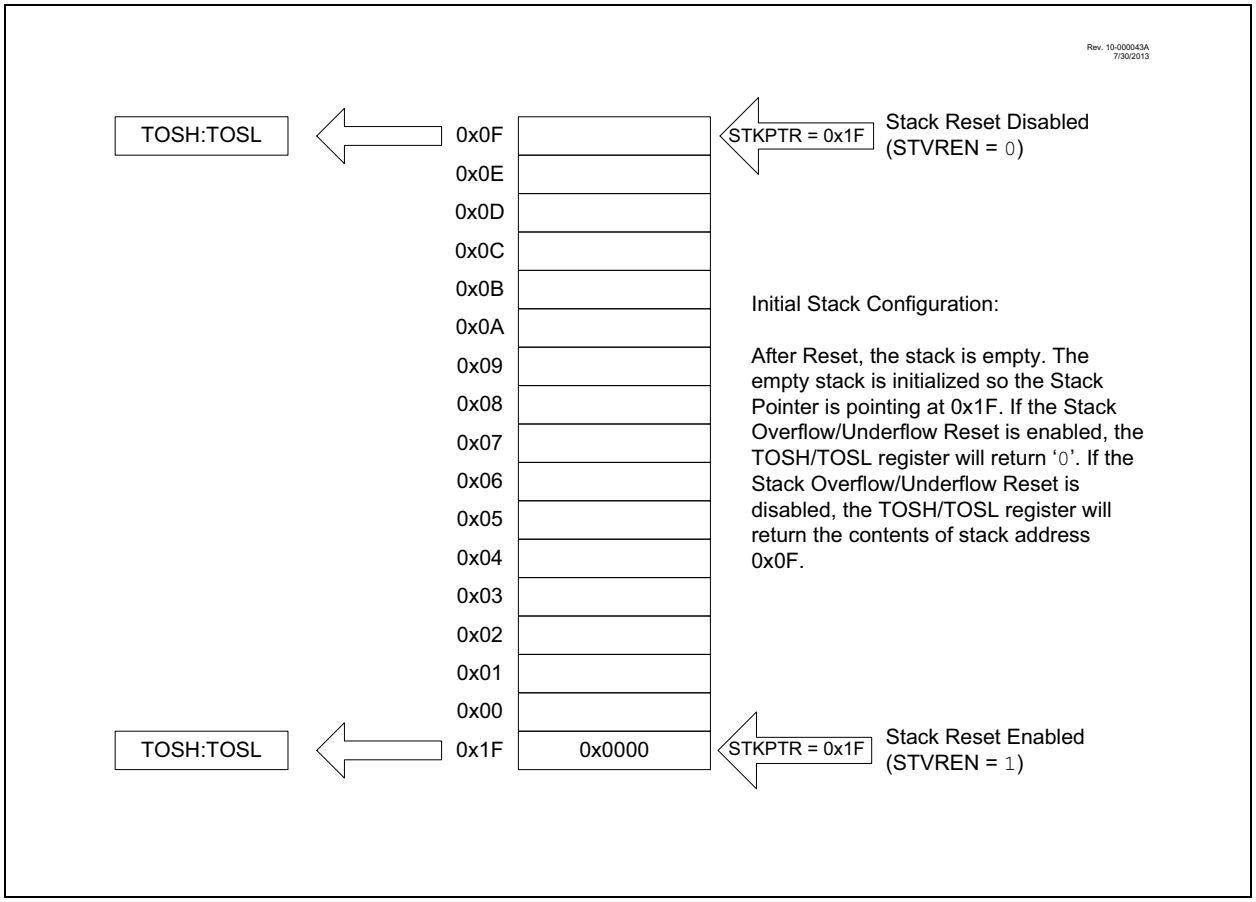
The stack is available through the TOSH, TOSL and STKPTR registers. STKPTR is the current value of the Stack Pointer. TOSH:TOSL register pair points to the TOP of the stack. Both registers are read/writable. TOS is split into TOSH and TOSL due to the 15-bit size of the PC. To access the stack, adjust the value of STKPTR, which will position TOSH:TOSL, then read/write to TOSH:TOSL. STKPTR is five 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 3-4 through Figure 3-7 for examples of accessing the stack.

**FIGURE 3-4: ACCESSING THE STACK EXAMPLE 1**




 © 2015-2016 Microchip Technology Inc.

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

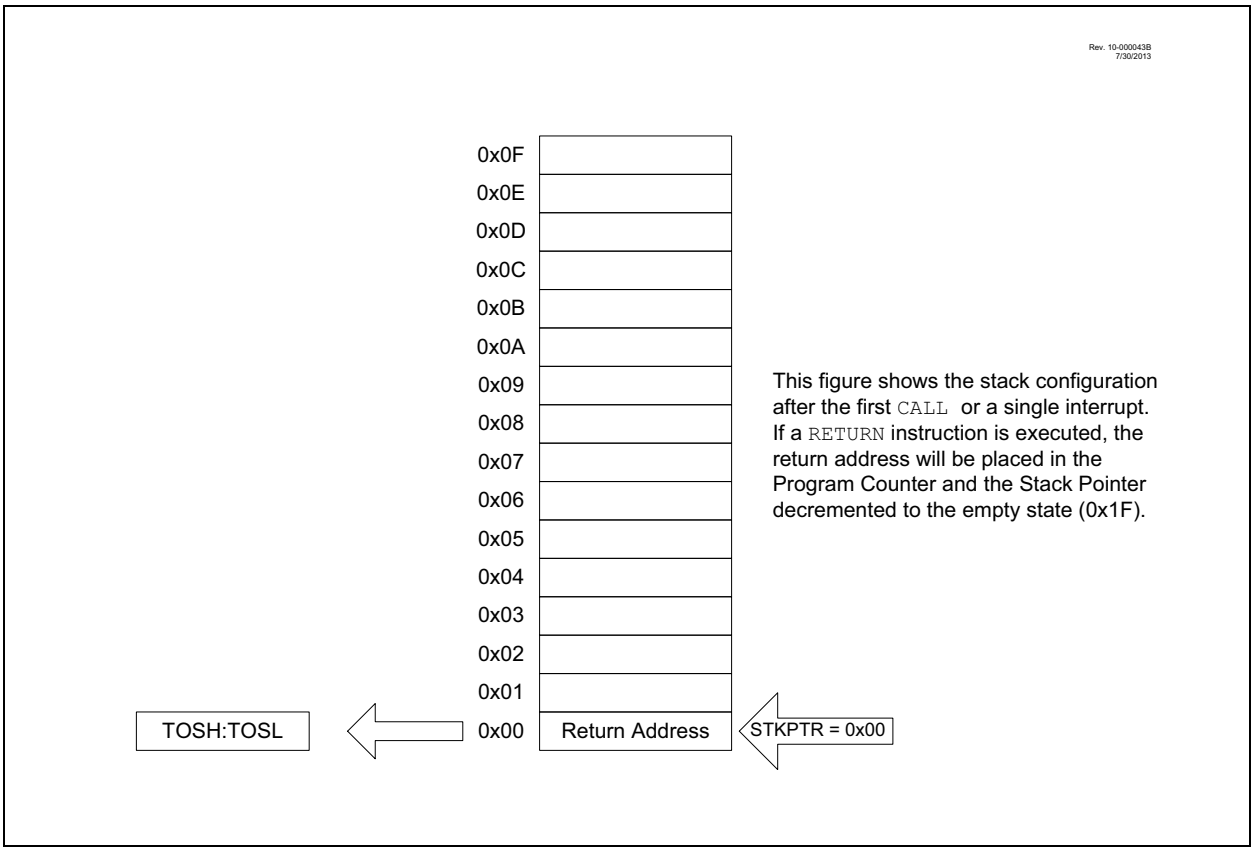



**FIGURE 3-6: ACCESSING THE STACK EXAMPLE 3**

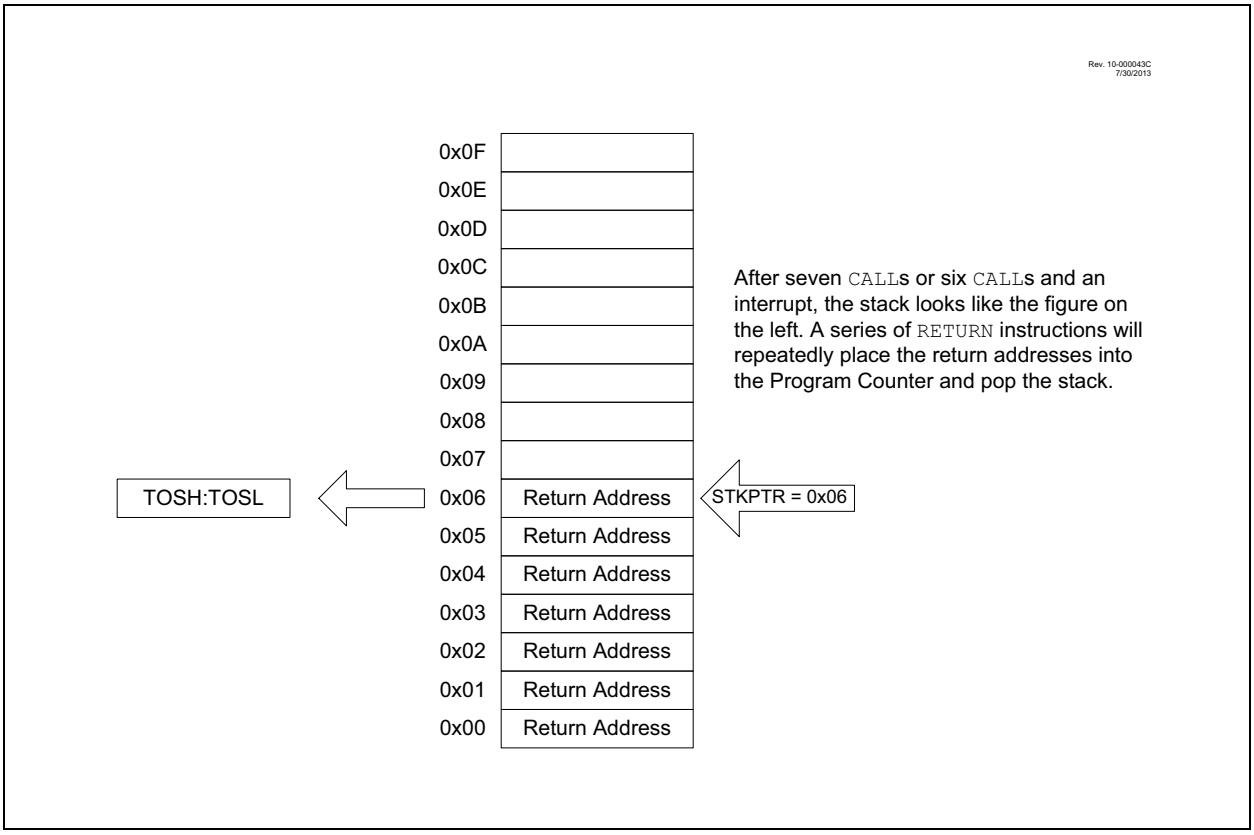
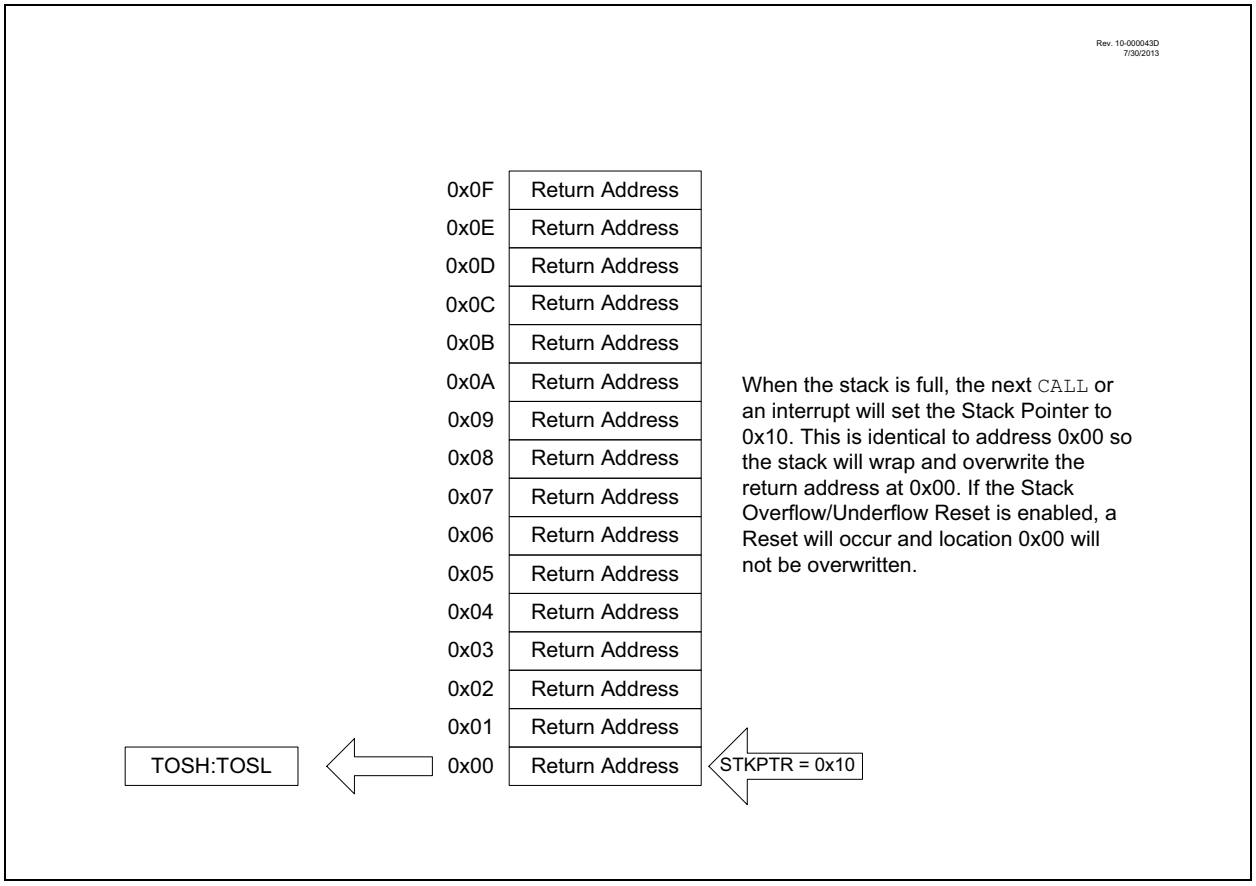


**FIGURE 3-7: ACCESSING THE STACK EXAMPLE 4**




### 3.4.2 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.

 © 2015-2016 Microchip Technology Inc.

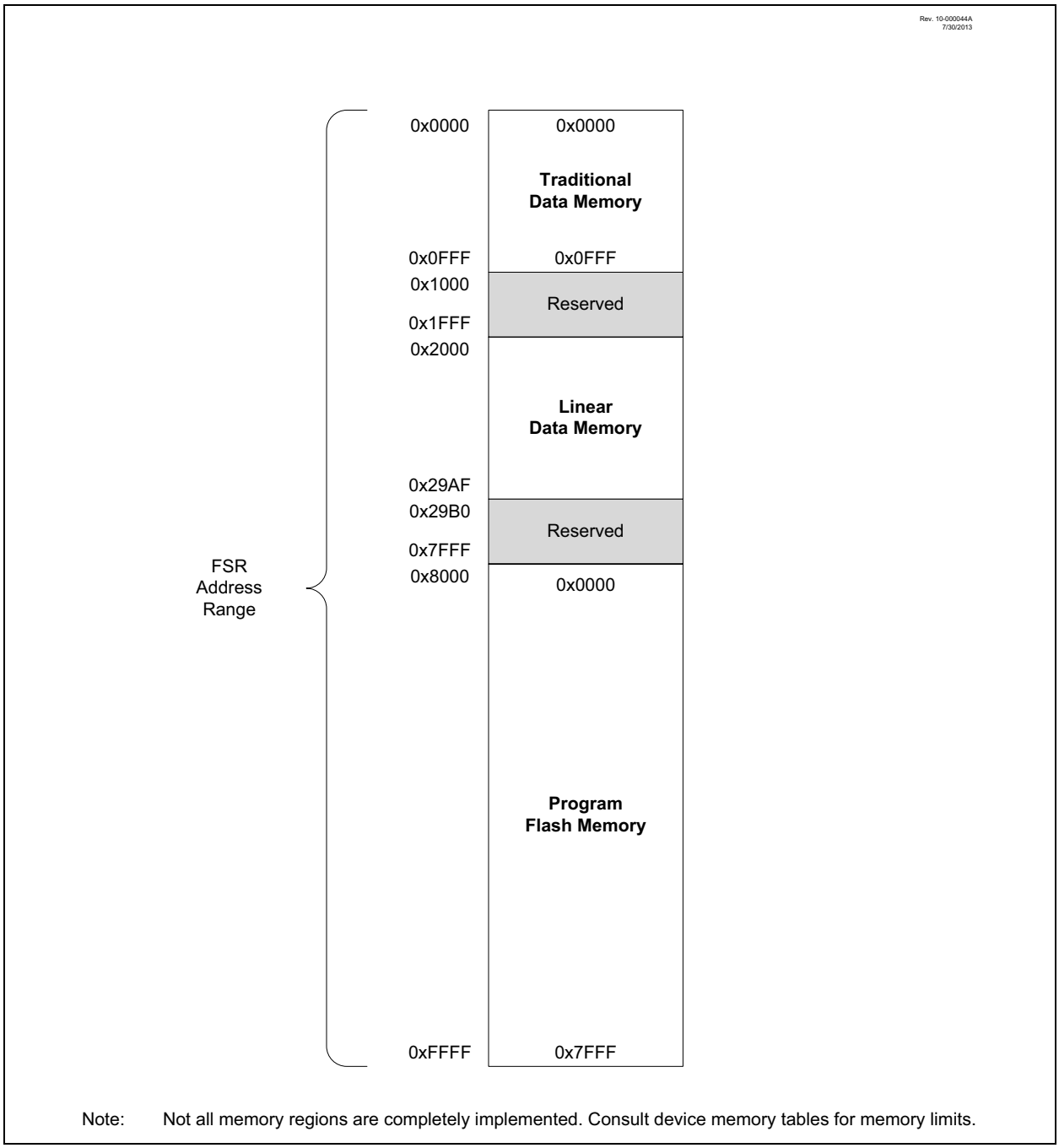
## 3.5 Indirect Addressing

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

**FIGURE 3-8: INDIRECT ADDRESSING**

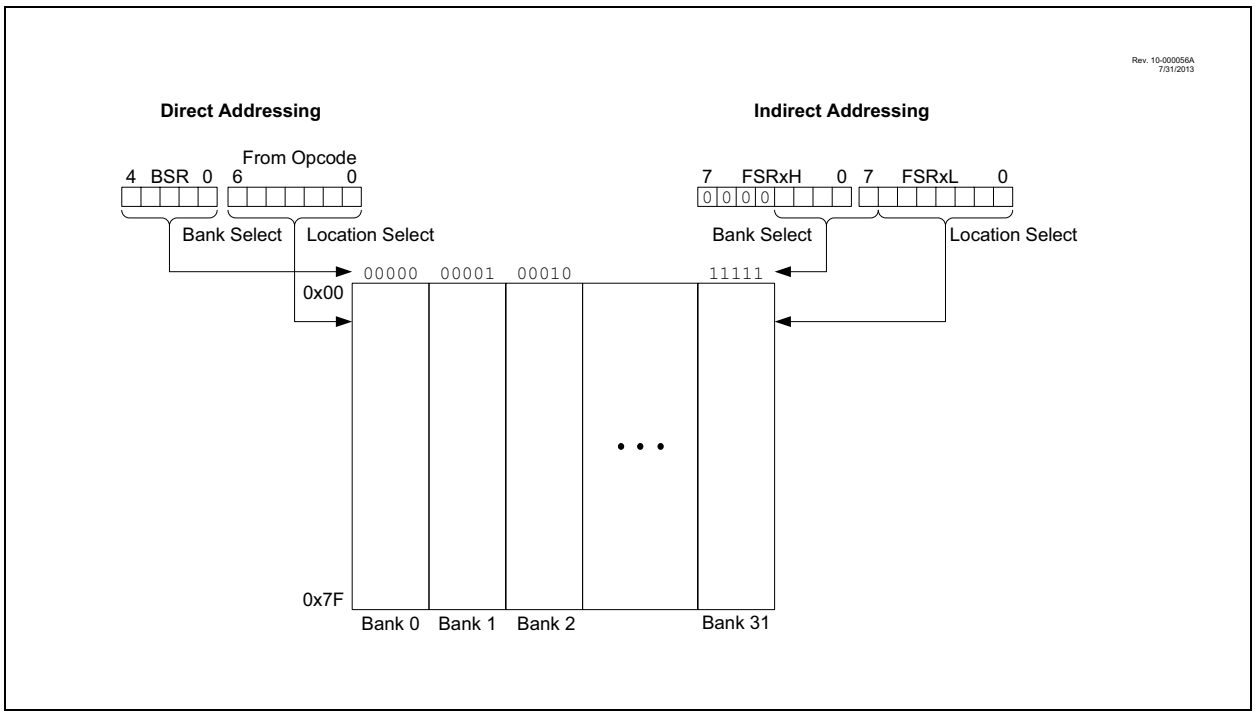



Note: Not all memory regions are completely implemented. Consult device memory tables for memory limits.

### 3.5.1 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 3-9: TRADITIONAL DATA MEMORY MAP**




 © 2015-2016 Microchip Technology Inc.

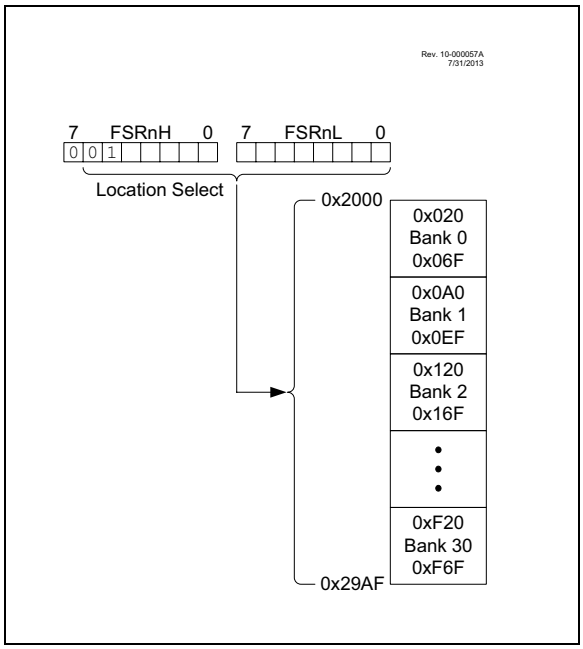
### 3.5.2 LINEAR DATA MEMORY

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.

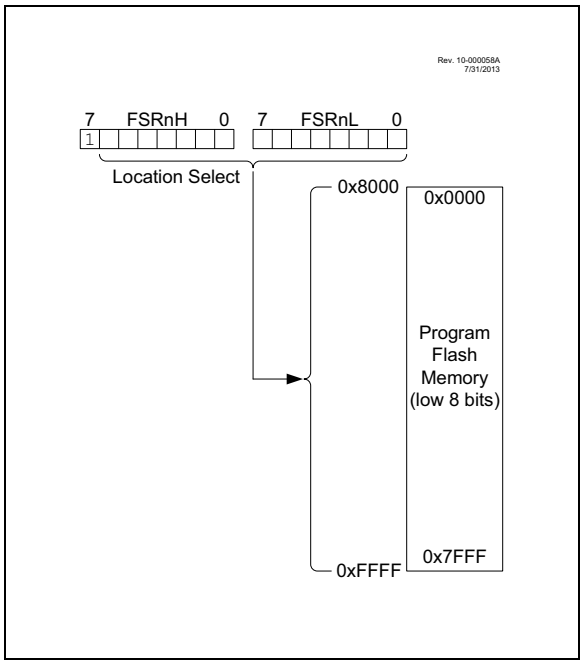
**FIGURE 3-10: LINEAR DATA MEMORY MAP**




### 3.5.3 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 eight 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 3-11: PROGRAM FLASH MEMORY MAP**



## 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'. |

 © 2015-2016 Microchip Technology Inc.

## 4.2 Register Definitions: Configuration Words

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

| U-1 | U-1 | R/P-1 | R/P-1 | R/P-1 | U-1 |
|---|---|---|---|---|---|
| — | — | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | — |
| bit 13 | | | | | bit 8 |

| R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | U-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | — | FOSC<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 13-12 **Unimplemented:** Read as '1'

bit 11 **$\overline{\text{CLKOUTEN}}$:** Clock Out Enable bit
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(1)
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 **Unimplemented:** Read as '1'

bit 7 **$\overline{\text{CP}}$:** Code Protection bit(2)
1 = Program memory code protection is disabled
0 = Program memory code protection is enabled

bit 6 **MCLRE:** $\overline{\text{MCLR}}$/VPP Pin Function Select bit
If LVP bit = 1:
This bit is ignored.
If LVP bit = 0:
1 = $\overline{\text{MCLR}}$/VPP pin function is $\overline{\text{MCLR}}$; Weak pull-up enabled.
0 = $\overline{\text{MCLR}}$/VPP pin function is digital input; $\overline{\text{MCLR}}$ internally disabled; Weak pull-up under control of WPUA3 bit.

bit 5 **$\overline{\text{PWRTE}}$:** Power-Up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 4-3 **WDTE<1:0>:** Watchdog Timer Enable bits
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

bit 2 **Unimplemented:** Read as '1'

bit 1-0 **FOSC<1:0>:** Oscillator Selection bits
11 = ECH: External Clock, High-Power mode: on CLKIN pin
10 = ECM: External Clock, Medium Power mode: on CLKIN pin
01 = ECL: External Clock, Low-Power mode: on CLKIN pin
00 = INTOSC oscillator: I/O function on CLKIN

**Note 1:** Enabling Brown-out Reset does not automatically enable Power-up Timer.
**2:** Once enabled, code-protect can only be disabled by bulk erasing the device.

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

| R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | U-1 |
|---|---|---|---|---|---|
| LVP | $\overline{\text{DEBUG}}$ | $\overline{\text{LPBOR}}$ | BORV | STVREN | — |
| bit 13 | | | | | bit 8 |

| U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | 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 $\overline{\text{MCLR}}$ must be used for programming

bit 12 **$\overline{\text{DEBUG}}$:** In-Circuit Debugger Mode bit(2)
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 **$\overline{\text{LPBOR}}$:** 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(3)
1 = Brown-out Reset voltage ($V_{BOR}$), low trip point selected
0 = Brown-out Reset voltage ($V_{BOR}$), 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 8-2 **Unimplemented:** Read as ‘1’

bit 1-0 **WRT<1:0>:** Flash Memory Self-Write Protection bits
8 kW Flash memory
11 = Write protection off
10 = 000h to 01FFh write protected, 0200h to 1FFFh may be modified
01 = 000h to 0FFFh write protected, 1000h to 1FFFh may be modified
00 = 000h to 1FFFh write protected, no addresses may be modified

**Note 1:** The LVP bit cannot be programmed to ‘0’ when Programming mode is entered via LVP.
**2:** 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’.
**3:** See $V_{BOR}$ parameter for specific trip point voltages.


© 2015-2016 Microchip Technology Inc.

## 4.3 Code Protection

Code protection allows the device to be protected from unauthorized access. Internal access to the program memory is 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.

## 4.4 Write Protection

Write protection allows the device to be protected from unintended self-writes. Applications, such as bootloader 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 10.4 "User ID, Device ID and Configuration Word Access"** for more information on accessing these memory locations. For more information on checksum calculation, see the **"***PIC16LF1566/1567 Memory Programming Specification*" (DS40001796).

## 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 10.4 "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.

**REGISTER 4-3: DEVICEID: DEVICE ID REGISTER(1)**

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

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| DEV<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | | |
| '0' = Bit is cleared | '1' = Bit is set | x = Bit is unknown |

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

| Device | DEV<13:0> Values |
|---|---|
| PIC16LF1566 | 11 0000 0100 0110 (3046h) |
| PIC16LF1567 | 11 0000 0100 0111 (3047h) |

**Note 1:** This location cannot be written.

**REGISTER 4-4: REVISIONID: REVISION ID REGISTER(1)**

| R | R | R | R | R | R |
|---|---|---|---|---|---|
| REV<13:8> | | | | | |
| bit 13 | | | | | bit 8 |

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| REV<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | | |
| '0' = Bit is cleared | '1' = Bit is set | x = Bit is unknown |

bit 13-0 **REV<13:0>:** Revision ID bits
These bits are used to identify the device revision.

**Note 1:** This location cannot be written.


© 2015-2016 Microchip Technology Inc.

# 5.0 OSCILLATOR MODULE

## 5.1 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 5-1 illustrates a block diagram of the oscillator module.

Clock sources can be supplied from external clock oscillators. 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.

The oscillator module can be configured in one of the following clock modes.

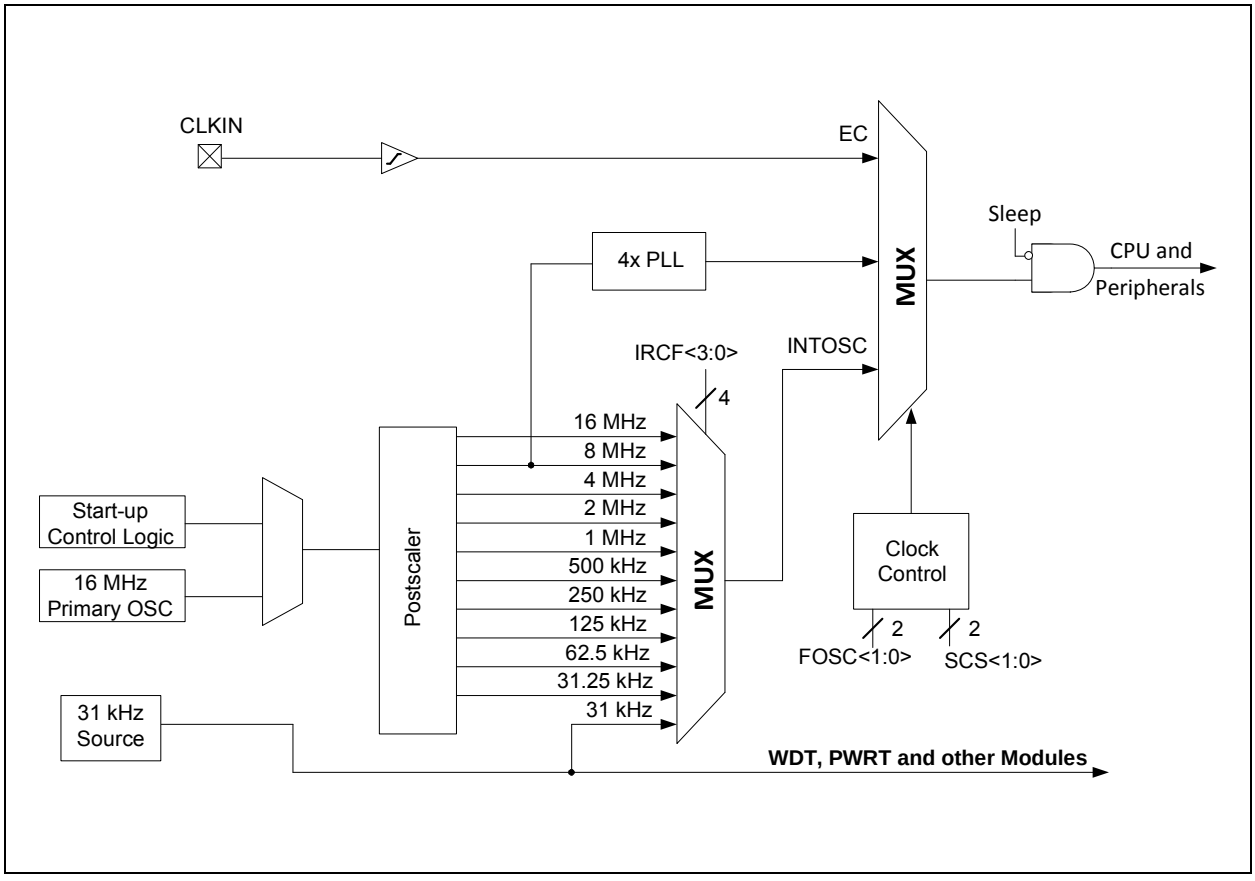
1. ECL – External Clock Low-Power mode (0 MHz to 0.5 MHz)
2. ECM – External Clock Medium Power mode (0.5 MHz to 4 MHz)
3. ECH – External Clock High-Power mode (4 MHz to 20 MHz)
4. INTOSC – Internal oscillator (31 kHz to 32 MHz)

Clock source modes are selected by the FOSC<1:0> 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 INTOSC internal oscillator block produces low and high-frequency clock sources, designated LFINTOSC and HFINTOSC (see Internal Oscillator Block, Figure 5-1). A wide selection of device clock frequencies may be derived from these clock sources.

**FIGURE 5-1: SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM**



## 5.2 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).

Internal clock sources are contained within the oscillator module. The oscillator block has two internal oscillators that are used to generate two system clock sources: the 16 MHz High-Frequency Internal Oscillator (HFINTOSC) 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 5.3 "Clock Switching"** for additional information.

### 5.2.1 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<1:0> bits in the Configuration Words to select an external clock source that will be used as the default system clock upon a device Reset.
- Clear the SCS<1:0> bits in the OSCCON register to switch the system clock source to:
  - An external clock source determined by the value of the FOSC bits.

See **Section 5.3 "Clock Switching"** for more information.

#### 5.2.1.1 EC Mode

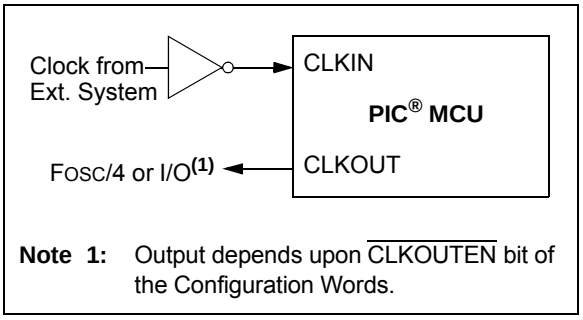
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the CLKIN input. CLKOUT is available for general purpose I/O or CLKOUT. Figure 5-2 shows the pin connections for EC mode.

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

- High power, 4-20 MHz (FOSC = 11)
- Medium power, 0.5-4 MHz (FOSC = 10)
- Low power, 0-0.5 MHz (FOSC = 01)

When EC mode is selected, 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 5-2: EXTERNAL CLOCK (EC) MODE OPERATION**




**Note 1:** Output depends upon $\overline{\text{CLKOUTEN}}$ bit of the Configuration Words.

 © 2015-2016 Microchip Technology Inc.

### 5.2.2 INTERNAL CLOCK SOURCES

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

- Program the FOSC<1:0> bits in Configuration Words to select the INTOSC clock source, which will be used as the default system clock upon a device Reset.
- Set the SCS<1:0> bits in the OSCCON register to '1x' to switch the system clock source to the internal oscillator during run-time. See **Section 5.3 "Clock Switching"** for more information.

In INTOSC mode, the CLKIN pin is available for general purpose I/O. The CLKOUT pin is available for general purpose I/O or CLKOUT.

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

The internal oscillator block has two independent oscillators.

1. The HFINTOSC (High-Frequency Internal Oscillator) is factory calibrated and operates at 16 MHz.
2. The LFINTOSC (Low-Frequency Internal Oscillator) is uncalibrated and operates at 31 kHz.

#### 5.2.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a factory calibrated 16 MHz internal clock source.

The outputs of the HFINTOSC connects to a prescaler and multiplexer (see Figure 5-1). 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 5.2.2.4 "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<1:0> = 00, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '1x'.

A fast start-up 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 Stable bit (HFIOFS) of the OSCSTAT register indicates when the HFINTOSC is running within 0.5% of its final value.

#### 5.2.2.2 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 5-1). Select 31 kHz, via software, using the IRCF<3:0> bits of the OSCCON register. See **Section 5.2.2.4 "Internal Oscillator Clock Switch Timing"** for more information. The LFINTOSC is also the source for the Power-up Timer (PWRT) and Watchdog Timer (WDT).

The LFINTOSC is enabled by selecting 31 kHz (IRCF<3:0> bits of the OSCCON register = 000x) 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 LF frequency, and
- FOSC<1:0> = 00, 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)

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

#### 5.2.2.3 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 outputs of the 16 MHz HFINTOSC postscaler and the LFINTOSC connect to a multiplexer (see Figure 5-1). The Internal Oscillator Frequency Select bits IRCF<3:0> of the OSCCON register select the frequency. 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.

#### 5.2.2.4 Internal Oscillator Clock Switch Timing

When switching between the HFINTOSC and the LFINTOSC, the new oscillator may already be shut down to save power (see Figure 5-3). 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 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. Clock switch is complete.

See Figure 5-3 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.

Start-up delay specifications are located in the oscillator tables of **Section 25.0 "Electrical Specifications"**.

#### 5.2.2.5 32 MHz Internal Oscillator Frequency Selection

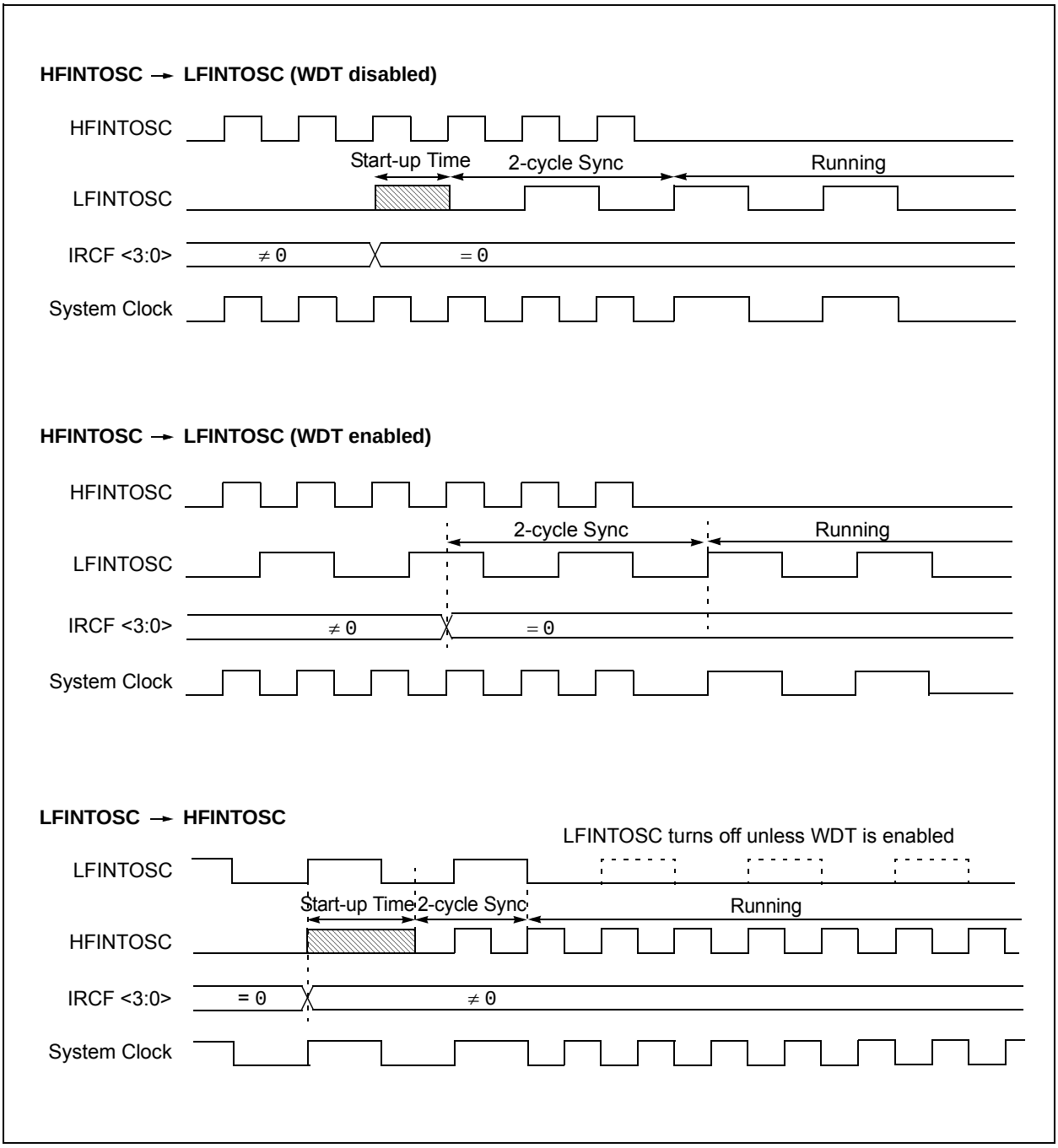
The Internal Oscillator Block can be used with the 4x PLL to produce a 32 MHz internal system clock source. The following settings are required to use the 32 MHz internal clock source:

- The FOSC bits in Configuration Word 1 must be set to use the INTOSC source as the device system clock (FOSC<1:0> = 00).
- The SCS bits in the OSCCON register must be cleared to use the clock determined by FOSC<1:0> in Configuration Word 1 (SCS<1:0> = 00).
- The IRCF bits in the OSCCON register must be set to the 8 MHz HFINTOSC set to use (IRCF<3:0> = 1110).
- The SPLLEN bit in the OSCCON register must be set to enable the 4x PLL.

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.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 5-3: INTERNAL OSCILLATOR SWITCH TIMING**



## 5.3 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
- Internal Oscillator Block (INTOSC)

### 5.3.1 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<1:0> bits in the Configuration Words.
- 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.

When switching between clock sources, a delay is required to allow the new clock to stabilize. These oscillator delays are shown in Table 5-1.

**TABLE 5-1: OSCILLATOR SWITCHING DELAYS**

| Switch From | Switch To | Frequency | Oscillator Delay |
|---|---|---|---|
| Sleep | LFINTOSC(1)<br>MFINTOSC(1)<br>HFINTOSC(1) | 31 kHz<br>31.25 kHz-500 kHz<br>31.25 kHz-16 MHz | Oscillator Warm-Up Delay TWARM(2) |
| Sleep/POR | EC(1) | DC – 32 MHz | 2 cycles |
| LFINTOSC | EC(1) | DC – 32 MHz | 1 cycle of each |
| Any clock source | MFINTOSC(1)<br>HFINTOSC | 31.25 kHz-500 MHz<br>31.25 kHz-16 MHz | 2 μs (approx.) |
| Any clock source | LFINTOSC | 31 kHz | 1 cycle of each |
| PLL inactive | PLL active | 16-32 MHz | 2 ms (approx.) |

**Note 1:** PLL inactive
**2:** See **Section 25.0 “Electrical Specifications”**

 © 2015-2016 Microchip Technology Inc.

## 5.4 Register Definitions: Oscillator Control

**REGISTER 5-1: 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
1 = 4x PLL Is enabled
0 = 4x PLL is disabled

bit 6-3 **IRCF<3:0>:** Internal Oscillator Frequency Select bits
1111 = 16 MHz
1110 = 8 MHz
1101 = 4 MHz
1100 = 2 MHz
1011 = 1 MHz
1010 = 500 kHz(1)
1001 = 250 kHz(1)
1000 = 125 kHz(1)
0111 = 500 kHz (default upon Reset)
0110 = 250 kHz
0101 = 125 kHz
0100 = 62.5 kHz
001x = 31.25 kHz
000x = 31 kHz (LFINTOSC)

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

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

**Note 1:** Duplicate frequency derived from HFINTOSC.

**REGISTER 5-2: OSCSTAT: OSCILLATOR STATUS REGISTER**

| U-0 | R-0/q | U-0 | R-0/q | U-0 | U-0 | R-0/q | R-0/q |
|---|---|---|---|---|---|---|---|
| — | PLLSR | — | HFIOFR | — | — | 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 **Unimplemented:** Read as ‘0’

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

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **HFIOFR:** High-Frequency Internal Oscillator Ready bit
1 = 16 MHz Internal Oscillator (HFINTOSC) is ready
0 = 16 MHz Internal Oscillator (HFINTOSC) is not ready

bit 3-2 **Unimplemented:** Read as ‘0’

bit 1 **LFIOFR:** Low-Frequency Internal Oscillator Ready bit
1 = 31 kHz Internal Oscillator (LFINTOSC) is ready
0 = 31 kHz Internal Oscillator (LFINTOSC) is not ready

bit 0 **HFIOFS:** High-Frequency Internal Oscillator Stable bit
1 = 16 MHz Internal Oscillator (HFINTOSC) is stable
0 = 16 MHz Internal Oscillator (HFINTOSC) is not yet stable

**TABLE 5-2: 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> | | 69 |
| OSCSTAT | — | PLLSR | — | HFIOFR | — | — | LFIOFR | HFIOFS | 70 |

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

**TABLE 5-3: 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 | — | — | — | — | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | — | 59 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | — | FOSC<1:0> | | |

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

 © 2015-2016 Microchip Technology Inc.

## 6.0 RESETS

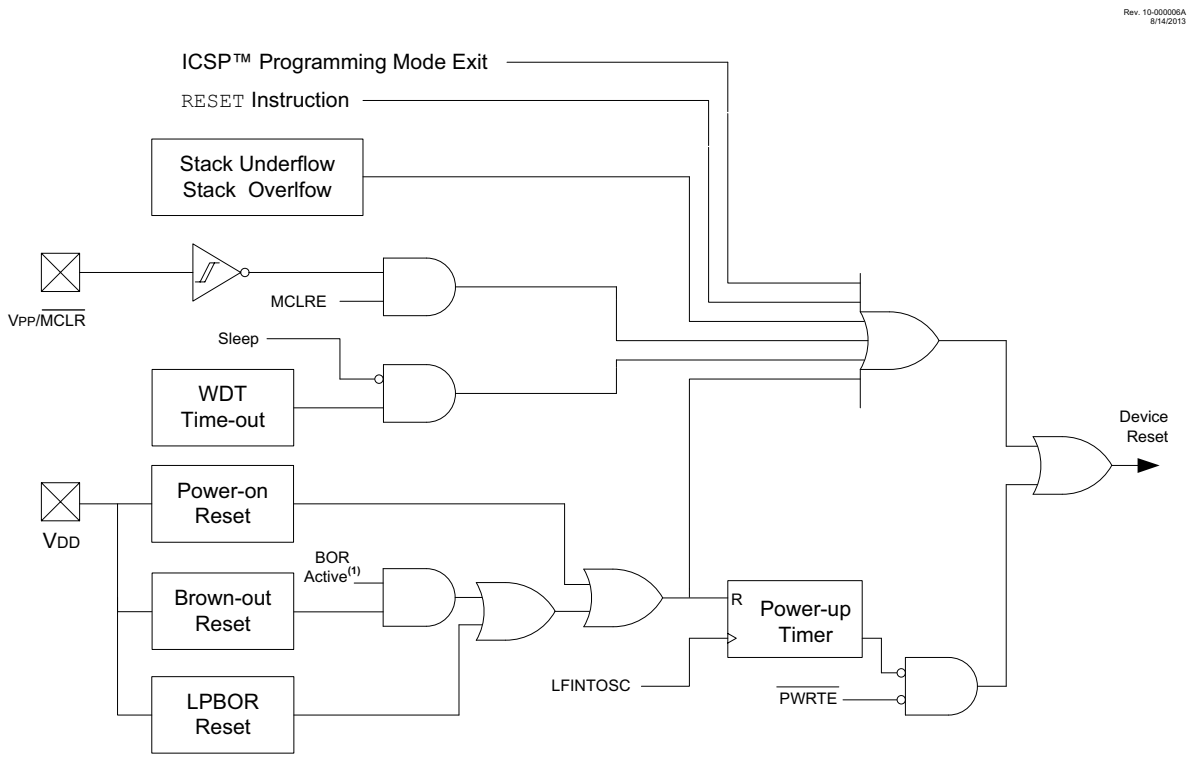
There are multiple ways to reset this device:

- 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 6-1.

**FIGURE 6-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**




**Note 1:** See Table 6-1 for BOR active conditions.

## 6.1 Power-on Reset (POR)

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

### 6.1.1 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 VDD to rise to an acceptable level. The Power-up Timer is enabled by clearing the 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"* (DS00000607).

## 6.2 Brown-out Reset (BOR)

The BOR circuit holds the device in Reset when VDD 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 6-1 for more information.

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

A VDD noise rejection filter prevents the BOR from triggering on small events. If VDD falls below VBOR for a duration greater than parameter TBORDC, the device will reset. See Figure 6-2 for more information.

### 6.2.1 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 VDD is higher than the BOR threshold.

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

### 6.2.2 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 VDD 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.

### 6.2.3 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 VDD 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.

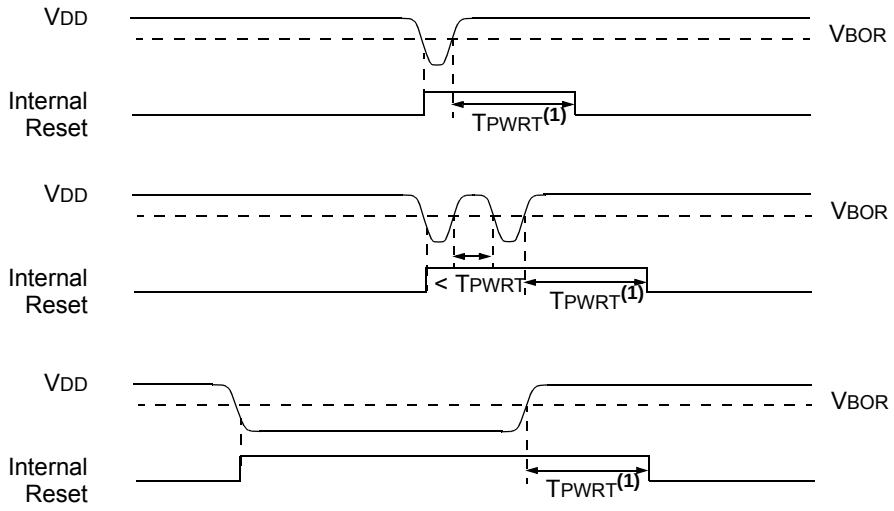
**TABLE 6-1: BOR OPERATING MODES**

| 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.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 6-2: BROWN-OUT SITUATIONS**




**Note 1:** TPWRT delay only if $\overline{\text{PWRTE}}$ bit is programmed to '0'.

## 6.3 Register Definitions: BOR Control

**REGISTER 6-1: 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
If BOREN <1:0> in Configuration Words = 01:
1 = BOR Enabled
0 = BOR Disabled
If BOREN <1:0> in Configuration Words ≠ 01:
SBOREN is read/write, but has no effect on the BOR

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

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.

## 6.4 Low-Power Brown-out Reset (LPBOR)

The Low-Power Brown-out Reset (LPBOR) operates like the BOR to detect low voltage conditions on the VDD pin. When too low of a voltage is detected, the device is held in Reset. When this occurs, a register bit (BOR) is changed to indicate that a BOR Reset has occurred. The $\overline{BOR}$ bit in PCON is used for both BOR and the LPBOR. Refer to Register 6-2.

The LPBOR voltage threshold (VLPBOR) has a wider tolerance than the BOR (VBOR), but requires much less current (LPBOR current) to operate. The LPBOR is intended for use when the BOR is configured as disabled (BOREN = 00) or disabled in Sleep mode (BOREN = 10).

Refer to Figure 6-1 to see how the LPBOR interacts with other modules.

### 6.4.1 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.

## 6.5 $\overline{MCLR}$

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 6-2).

**TABLE 6-2: $\overline{MCLR}$ CONFIGURATION**

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

### 6.5.1 $\overline{MCLR}$ 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. |
|---|---|

### 6.5.2 $\overline{MCLR}$ 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 11.3 "PORTA Registers"** for more information.

## 6.6 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 9.0 "Watchdog Timer (WDT)"** for more information.

## 6.7 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 6-4 for default conditions after a RESET instruction has occurred.

## 6.8 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 3.4.2 "Overflow/Underflow Reset"** for more information.

## 6.9 Programming Mode Exit

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

## 6.10 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.

## 6.11 Start-up Sequence

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

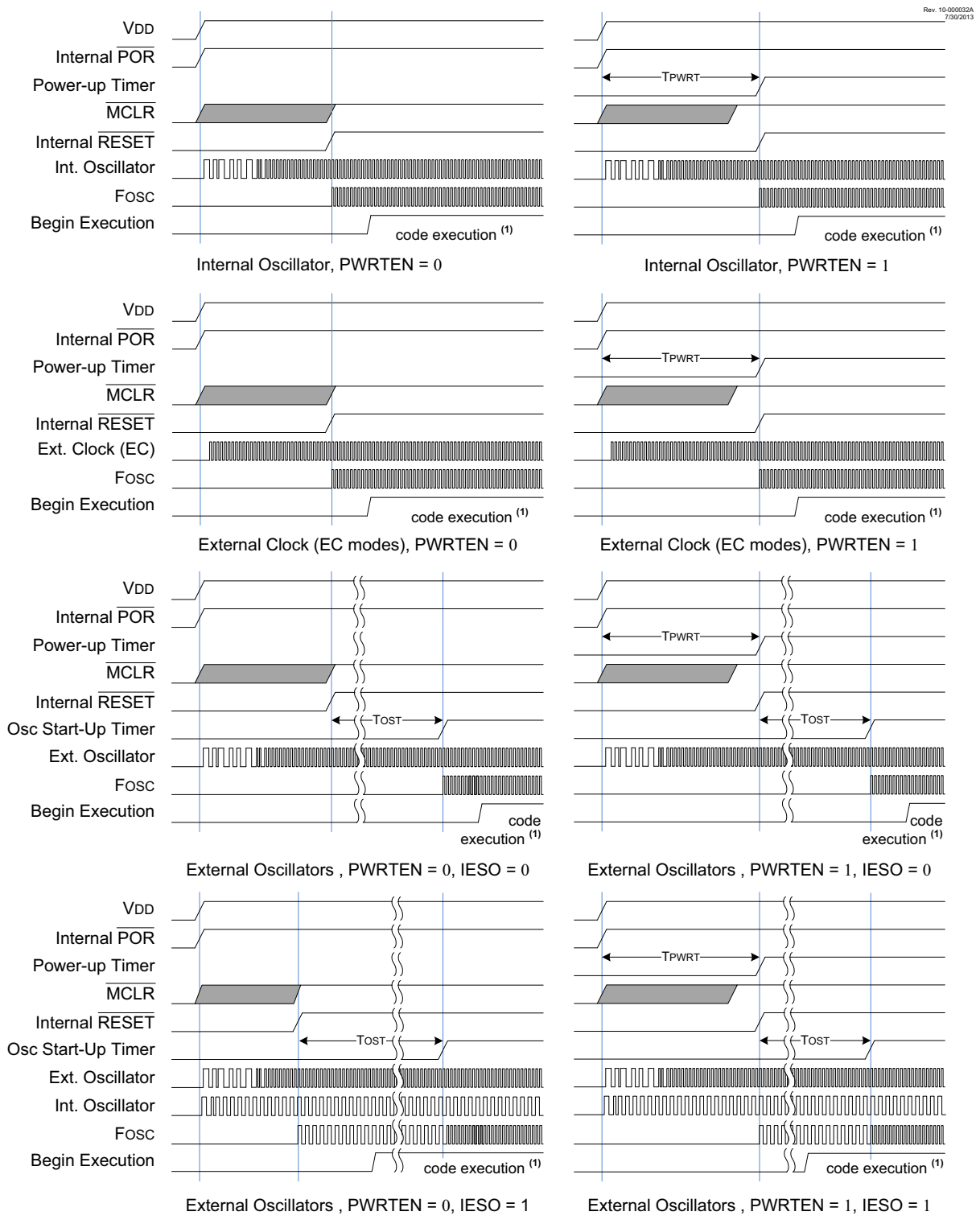
1. Power-up Timer runs to completion (if enabled).
2. $\overline{MCLR}$ must be released (if enabled).

The total time-out will vary based on oscillator configuration and Power-up Timer configuration. See **Section 5.0 "Oscillator Module"** for more information.

The Power-up Timer runs independently of $\overline{MCLR}$ Reset. If $\overline{MCLR}$ is kept low long enough, the Power-up Timer will expire. Upon bringing $\overline{MCLR}$ high, the device will begin execution after 10 FOSC cycles (see Figure 6-3). This is useful for testing purposes or to synchronize more than one device operating in parallel.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 6-3: RESET START-UP SEQUENCE**




**Note 1:** Code execution begins 10 FOSC cycles after the FOSC clock is released.

## 6.12 Determining the Cause of a Reset

Upon any Reset, multiple bits in the STATUS and PCON registers are updated to indicate the cause of the Reset. Table 6-3 and Table 6-4 show the Reset conditions of these registers.

**TABLE 6-3: RESET STATUS BITS AND THEIR SIGNIFICANCE**

| STKOVF | STKUNF | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | 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 6-4: RESET CONDITION FOR SPECIAL REGISTERS**

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

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

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

 © 2015-2016 Microchip Technology Inc.

## 6.13 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 6-2.

## 6.14 Register Definitions: Power Control

**REGISTER 6-2: 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 | -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 **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 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 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 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)

**TABLE 6-5: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS(1)**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| BORCON | SBOREN | BORFS | — | — | — | — | — | BORRDY | 73 |
| PCON | STKOVF | STKUNF | — | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 77 |
| STATUS | — | — | — | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | Z | DC | C | 25 |
| WDTCON | — | — | WDTPS<4:0> | | | | | SWDTEN | 93 |

**Legend:** — = unimplemented bit, reads as '0'. Shaded cells are not used by Resets.
**Note 1:** Other (non Power-up) Resets include $\overline{\text{MCLR}}$ Reset and Watchdog Timer Reset during normal operation.

**TABLE 6-6: SUMMARY OF CONFIGURATION WORD WITH RESETS**

| 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 | — | — | — | — | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | — | 59 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | — | FOSC<1:0> | | |
| CONFIG2 | 13:8 | — | — | LVP | $\overline{\text{DEBUG}}$ | $\overline{\text{LPBOR}}$ | BORV | STVREN | — | 60 |
| | 7:0 | — | — | — | — | — | — | WRT<1:0> | | |

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


© 2015-2016 Microchip Technology Inc.

## 7.0 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.

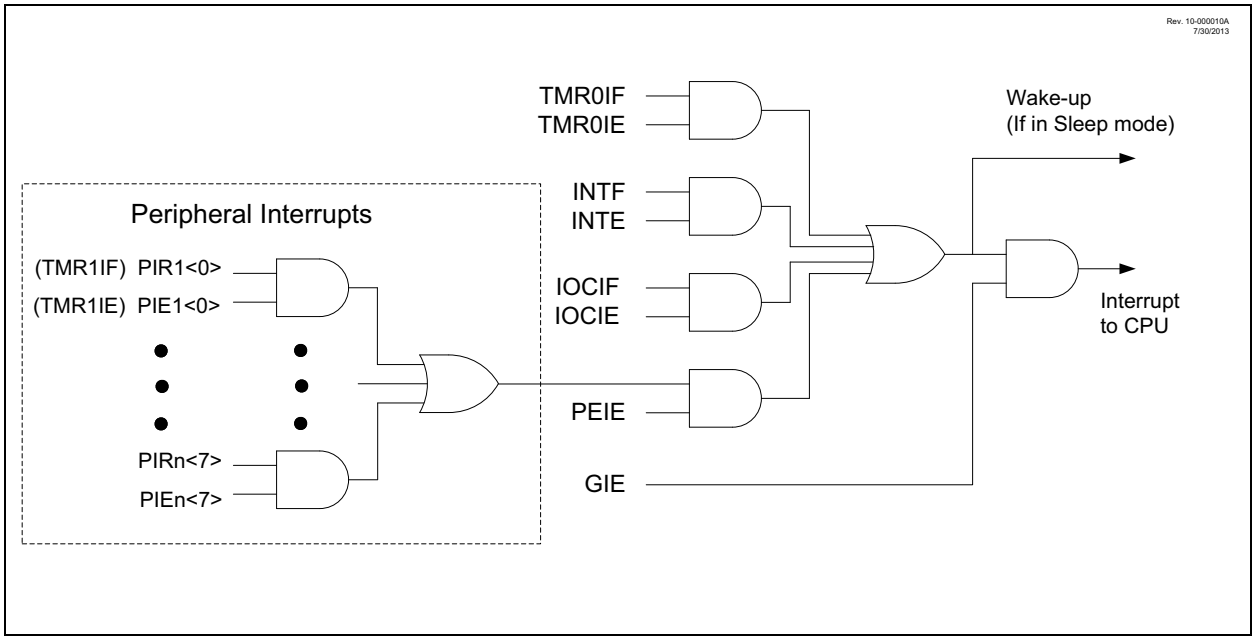
This chapter contains the following information for interrupts:

- 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 7-1.

**FIGURE 7-1: INTERRUPT LOGIC**



## 7.1 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 and 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 7.5 "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. |

 © 2015-2016 Microchip Technology Inc.

## 7.2 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 three or four instruction cycles. For asynchronous interrupts, the latency is three to five instruction cycles, depending on when the interrupt occurs. See Figure 7-2 and Figure 7-3 for more details.

**FIGURE 7-2: INTERRUPT LATENCY**

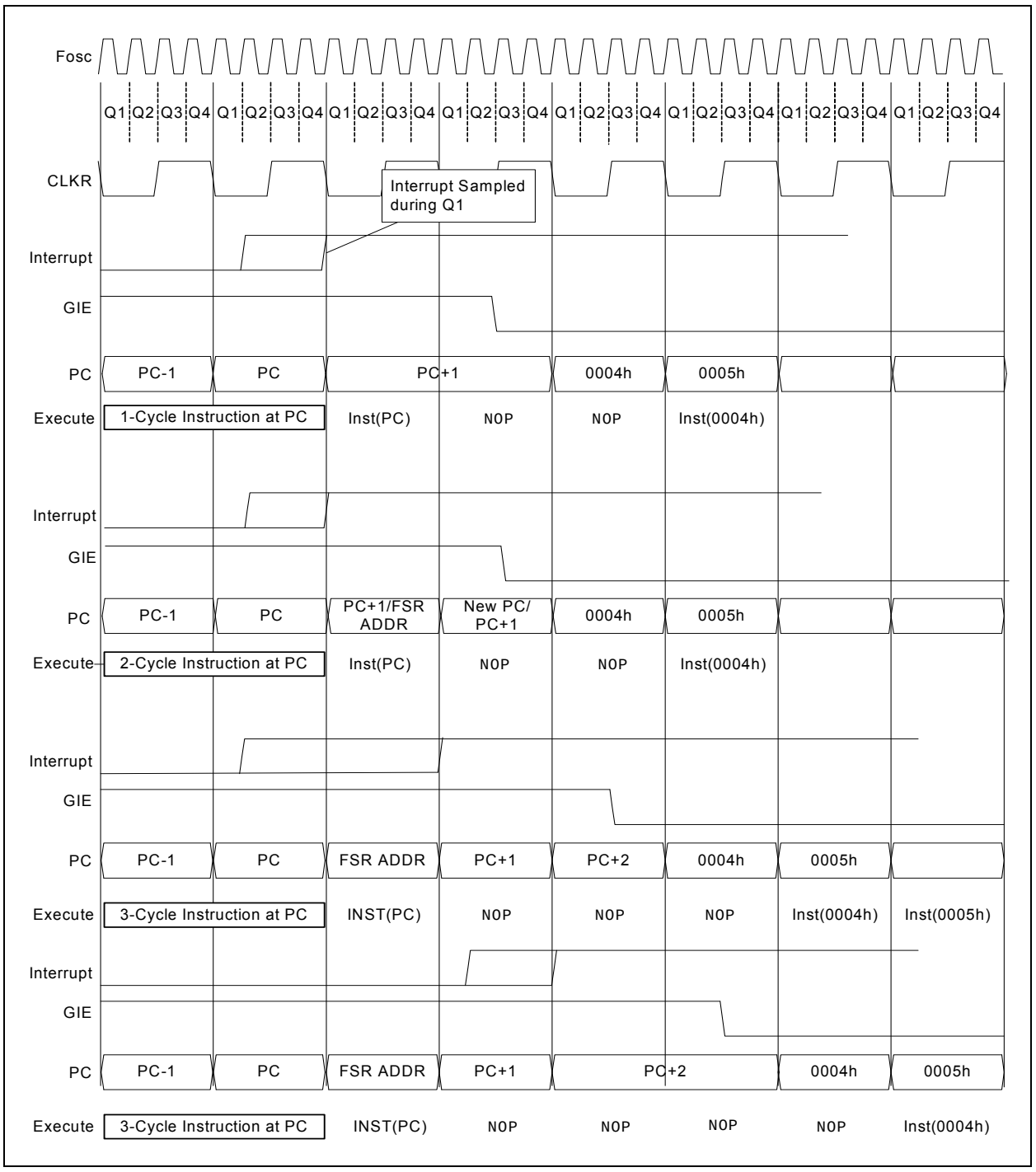
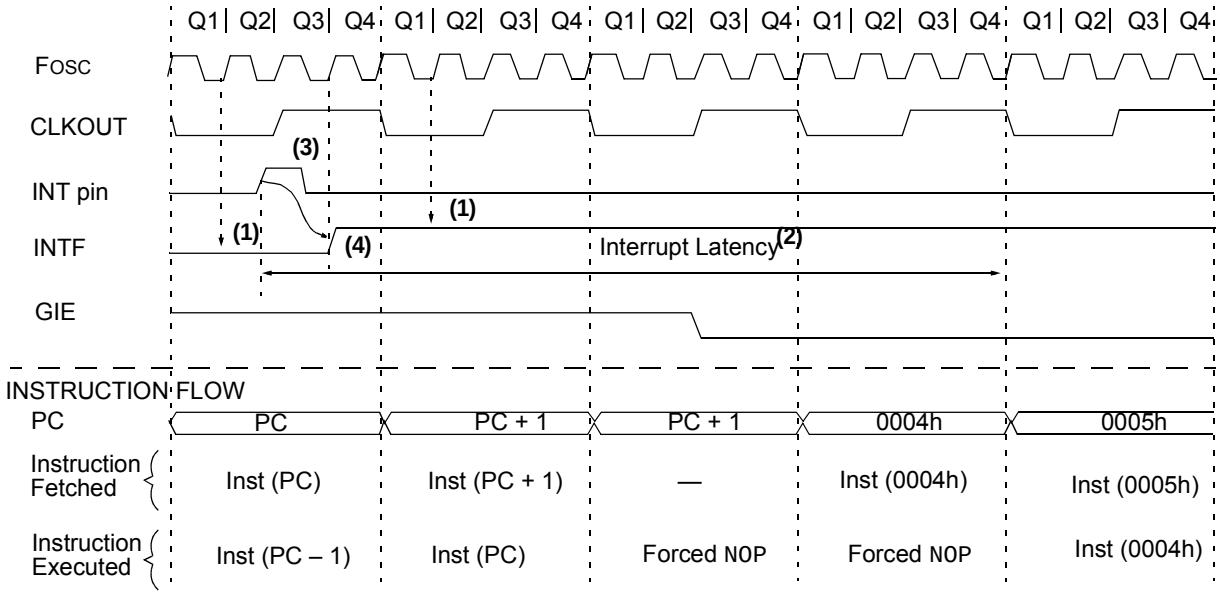


**FIGURE 7-3: INT PIN INTERRUPT TIMING**




**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:** For minimum width of INT pulse, refer to AC specifications in **Section 25.0 "Electrical Specifications"**.

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

 © 2015-2016 Microchip Technology Inc.

## 7.3 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 8.0 "Power-Down Mode (Sleep)"** for more details.

## 7.4 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.

## 7.5 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.

## 7.6 Register Definitions: Interrupt Control

**REGISTER 7-1: 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(1) | PEIE(2) | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF(3) |
| 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)
1 = Enables all active interrupts
0 = Disables all interrupts

bit 6 **PEIE:** Peripheral Interrupt Enable bit(2)
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(3)
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:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

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

**3:** The IOCIF flag bit is read-only and cleared when all the interrupt-on-change flags in the IOCxF registers have been cleared by software.

 © 2015-2016 Microchip Technology Inc.

**REGISTER 7-2: 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-1/1 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | 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 **AD1IE:** Analog-to-Digital Converter (ADC1) 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 **SSP1IE:** Synchronous Serial Port (MSSP1) Interrupt Enable bit
1 = Enables the MSSP1 interrupt
0 = Disables the MSSP1 interrupt

bit 2 **SSP2IE:** Synchronous Serial Port (MSSP1) Interrupt Enable bit
1 = Enables the MSSP2 interrupt
0 = Disables the MSSP2 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.

**REGISTER 7-3: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2**

| U-0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | AD2IE | — | — | BCL1IE | BCL2IE | TMR4IE | — |
| 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 **AD2IE:** Analog-to-Digital Converter (ADC) 2 Interrupt Enable bit
1 = Enables the ADC interrupt
0 = Disables the ADC interrupt

bit 5-4 **Unimplemented:** Read as '0'

bit 3 **BCL1IE:** MSSP1 Bus Collision Interrupt Enable bit
1 = Enables the MSSP Bus Collision Interrupt
0 = Disables the MSSP Bus Collision Interrupt

bit 2 **BCL2IE:** MSSP2 Bus Collision Interrupt Enable bit
1 = Enables the MSSP Bus Collision Interrupt
0 = Disables the MSSP Bus Collision Interrupt

bit 1 **TMR4IE:** TMR4 to PR4 Match Interrupt Enable bit
1 = Enables the TMR4 to PR4 Match Interrupt
0 = Disables the TMR4 to PR4 Match Interrupt

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

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

 © 2015-2016 Microchip Technology Inc.

**REGISTER 7-4: PIR1: PERIPHERAL INTERRUPT REQUEST 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 |
|---|---|---|---|---|---|---|---|
| TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | 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 **AD1IF:** ADC 1 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 **SSP1IF:** Synchronous Serial Port (MSSP1) Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 2 **SSP2IF:** Synchronous Serial Port (MSSP2) 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 Interrupt Enable bit, GIE of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

**REGISTER 7-5: PIR2: PERIPHERAL INTERRUPT REQUEST REGISTER 2**

| U-0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | AD2IF | — | — | BCL1IF | BCL2IF | TMR4IF | — |
| 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 **AD2IF:** ADC 2 Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 5-4 **Unimplemented:** Read as '0'

bit 3 **BCL1IF:** MSSP1 Bus Collision Interrupt Flag bit
1 = A Bus Collision was detected (must be cleared in software)
0 = No Bus Collision was detected

bit 2 **BCL2IF:** MSSP2 Bus Collision Interrupt Flag bit
1 = A Bus Collision was detected (must be cleared in software)
0 = No Bus Collision was detected

bit 1 **TMR4IF:** TMR4 to PR4 Match Interrupt Flag bit
1 = TMR4 to PR4 postscaled match occurred
0 = No TMR4 to PR4 match occurred

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

**Note:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

**TABLE 7-1: 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> | | | 181 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIE2 | — | AD2IE | — | — | BCL1IE | BCL2IE | TMR4IE | — | 86 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| PIR2 | — | AD2IF | — | — | BCL1IF | BCL2IF | TMR4IF | — | 88 |

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


© 2015-2016 Microchip Technology Inc.

## 8.0 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. Timer1 and peripherals that operate from Timer1 continue operation in Sleep when the Timer1 clock source selected is:
   - LFINTOSC
   - T1CKI
   - Timer1 oscillator
7. ADC is unaffected, if the dedicated FRC oscillator is selected.
8. I/O ports maintain the status they had before SLEEP was executed (driving high, low or high-impedance).
9. 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

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 the FVR module. See **Section 13.0 "Fixed Voltage Reference (FVR)"** for more information on this module.

### 8.1 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 6.12 "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.

### 8.1.1 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.

**TABLE 8-1: 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 | IOCIF | 84 |
| IOCBF | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 | 133 |
| IOCBN | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 | 133 |
| IOCBP | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 | 133 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIE2 | — | AD2IE | — | — | BCL1IE | BCL2IE | TMR4IE | — | 86 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| PIR2 | — | AD2IF | — | — | BCL1IF | BCL2IF | TMR4IF | — | 88 |
| STATUS | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | 25 |
| WDTCON | — | — | WDTPS<4:0> | | | | | SWDTEN | 93 |

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


© 2015-2016 Microchip Technology Inc.

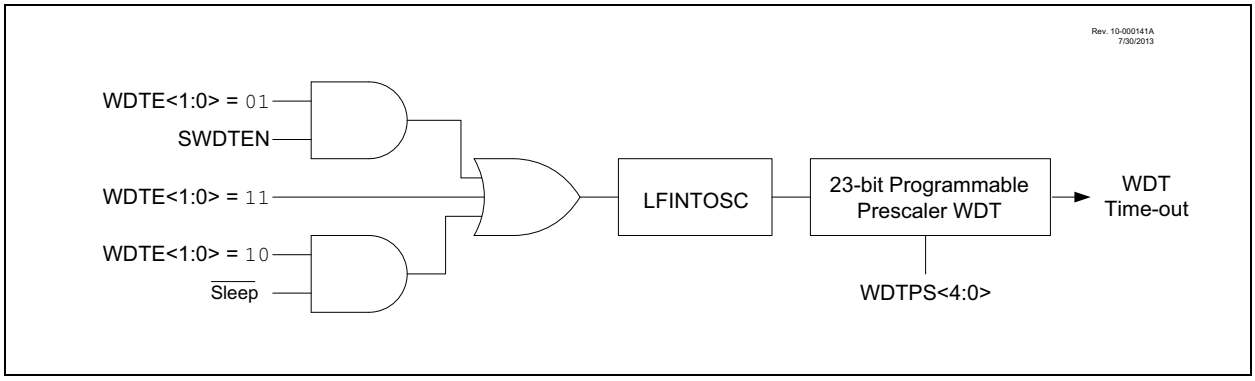
## 9.0 WATCHDOG TIMER (WDT)

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 9-1: WATCHDOG TIMER BLOCK DIAGRAM**



## 9.1 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 1 ms. See **Section 25.0 "Electrical Specifications"** for the LFINTOSC tolerances.

## 9.2 WDT Operating Modes

The Watchdog Timer module has four operating modes controlled by the WDTE<1:0> bits in Configuration Words. See Table 9-1.

### 9.2.1 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.

### 9.2.2 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.

### 9.2.3 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 9-1 for more details.

**TABLE 9-1: 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 | X | Disabled |
| 00 | X | X | Disabled |

## 9.3 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 two seconds.

## 9.4 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 9-2 for more information.

## 9.5 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 5.0 "Oscillator Module"** 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. The $\overline{RWDT}$ bit in the PCON register can also be used. See **Section 3.0 "Memory Organization"** for more information.

**TABLE 9-2: 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 |

 © 2015-2016 Microchip Technology Inc.

## 9.6 Register Definitions: Watchdog Control

**REGISTER 9-1: 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 | -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 5-1 **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.

**TABLE 9-3: 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> | | 69 |
| PCON | STKOVF | STKUNF | — | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 77 |
| STATUS | — | — | — | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | Z | DC | C | 25 |
| WDTCON | — | — | WDTPS<4:0> | | | | | SWDTEN | 93 |

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

**TABLE 9-4: 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 | — | — | — | — | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | — | 59 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | — | FOSC<1:0> | | |

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


© 2015-2016 Microchip Technology Inc.

## 10.0 FLASH PROGRAM MEMORY CONTROL

The Flash program memory is readable and writable during normal operation over the full VDD range. Program memory is indirectly addressed using Special Function Registers (SFRs). The SFRs used to access program memory are:

- PMCON1
- PMCON2
- PMDATL
- PMDATH
- PMADRL
- PMADRH

When accessing the program memory, the PMDATH:PMDATL register pair forms a 2-byte word that holds the 14-bit data for read/write, and the PMADRH:PMADRL register pair forms a 2-byte word that holds the 15-bit address of the program memory location being read.

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 operating voltage range of the device.

The Flash program memory can be protected in two ways; by code protection ($\overline{CP}$ bit in Configuration Words) and write protection (WRT<1:0> bits in Configuration Words).

Code protection ($\overline{CP}$ = 0)**(1)**, disables access, reading and writing, to the Flash program memory via external device programmers. Code protection does not affect the self-write and erase functionality. Code protection can only be reset by a device programmer performing a Bulk Erase to the device, clearing all Flash program memory, Configuration bits and User IDs.

Write protection prohibits self-write and erase to a portion or all of the Flash program memory, as defined by the bits WRT<1:0>. Write protection does not affect a device programmers ability to read, write or erase the device.

| | |
|---|---|
| **Note 1:** | Code protection of the entire Flash program memory array is enabled by clearing the $\overline{CP}$ bit of Configuration Words. |

## 10.1 PMADRL and PMADRH Registers

The PMADRH:PMADRL register pair can address 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 PMADRH register and the LSB is written to the PMADRL register.

### 10.1.1 PMCON1 AND PMCON2 REGISTERS

PMCON1 is the control register for Flash program memory accesses.

Control bits RD and WR initiate read and write, respectively. These bits cannot be cleared, only set, in software. They are cleared by 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.

The PMCON2 register is a write-only register. Attempting to read the PMCON2 register will return all '0's.

To enable writes to the program memory, a specific pattern (the unlock sequence), must be written to the PMCON2 register. The required unlock sequence prevents inadvertent writes to the program memory write latches and Flash program memory.

## 10.2 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 size that can be erased by user software.

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 PMDATH:PMDATL 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. Then, new data and retained data can be written into the write latches to reprogram the row of Flash program memory. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations. |

See Table 10-1 for erase row size and the number of write latches for Flash program memory.

**TABLE 10-1: FLASH MEMORY ORGANIZATION BY DEVICE**

| Device | Row Erase (words) | Write Latches (words) |
|---|---|---|
| PIC16LF1566 | 32 | 32 |
| PIC16LF1567 | | |

### 10.2.1 READING THE FLASH PROGRAM MEMORY

To read a program memory location, the user must:

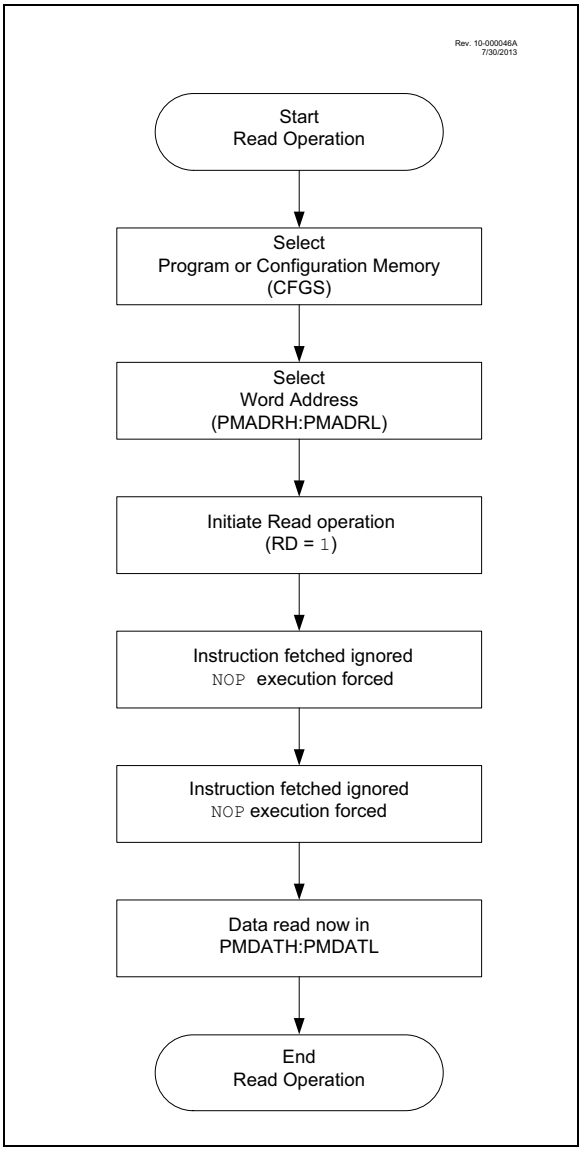
1. Write the desired address to the PMADRH:PMADRL register pair.
2. Clear the CFGS bit of the PMCON1 register.
3. Then, set control bit RD of the PMCON1 register.

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 PMCON1,RD" instruction to be ignored. The data is available in the very next cycle, in the PMDATH:PMDATL register pair; therefore, it can be read as two bytes in the following instructions.

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

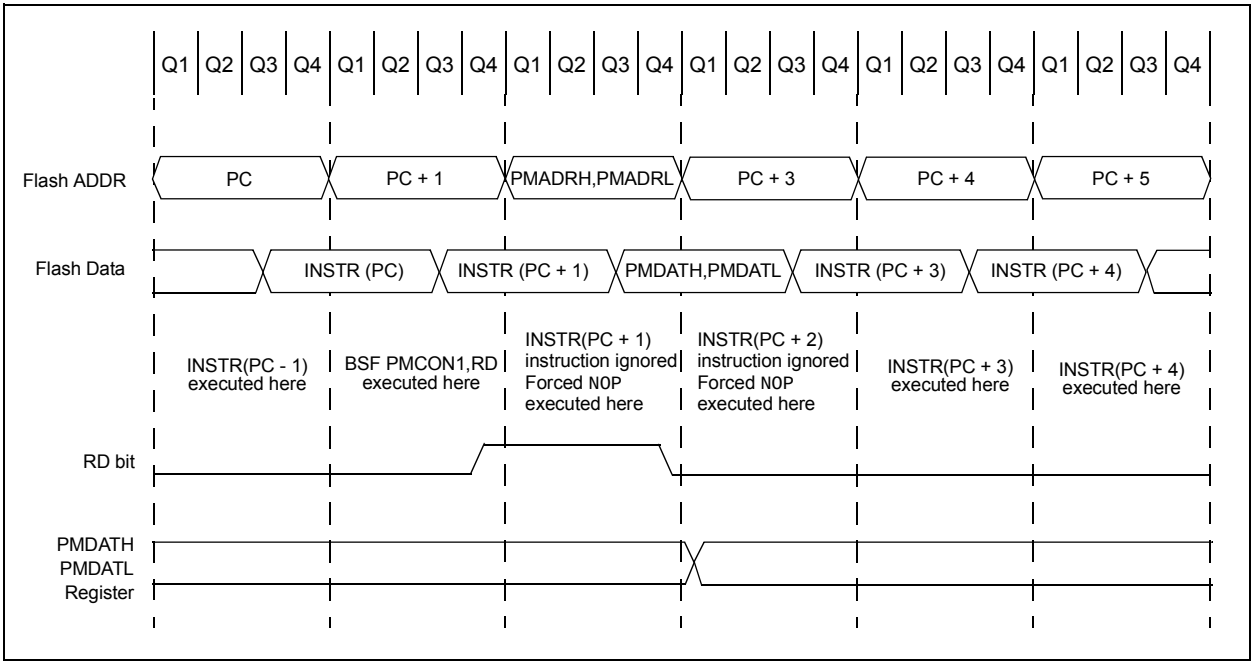
| **Note:** | The two instructions following a program memory read are required to be NOPs. This prevents the user from executing a 2-cycle instruction on the next instruction after the RD bit is set. |
|---|---|

**FIGURE 10-1: FLASH PROGRAM MEMORY READ FLOWCHART**




 © 2015-2016 Microchip Technology Inc.

**FIGURE 10-2: FLASH PROGRAM MEMORY READ CYCLE EXECUTION**




**EXAMPLE 10-1: 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  PMADRL           ; Select Bank for PMCON registers
    MOVLW    PROG_ADDR_LO     ;
    MOVWF    PMADRL           ; Store LSB of address
    MOVLW    PROG_ADDR_HI     ;
    MOVWF    PMADRH           ; Store MSB of address

    BCF      PMCON1,CFGS      ; Do not select Configuration Space
    BSF      PMCON1,RD        ; Initiate read
    NOP                       ; Ignored (Figure 10-2)
    NOP                       ; Ignored (Figure 10-2)

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

### 10.2.2 FLASH MEMORY UNLOCK SEQUENCE

The unlock sequence is a mechanism that protects the Flash program memory from unintended self-write programming or erasing. The sequence must be executed and completed without interruption to successfully complete any of the following operations:

- Row Erase
- Load program memory write latches
- Write of program memory write latches to program memory
- Write of program memory write latches to User IDs

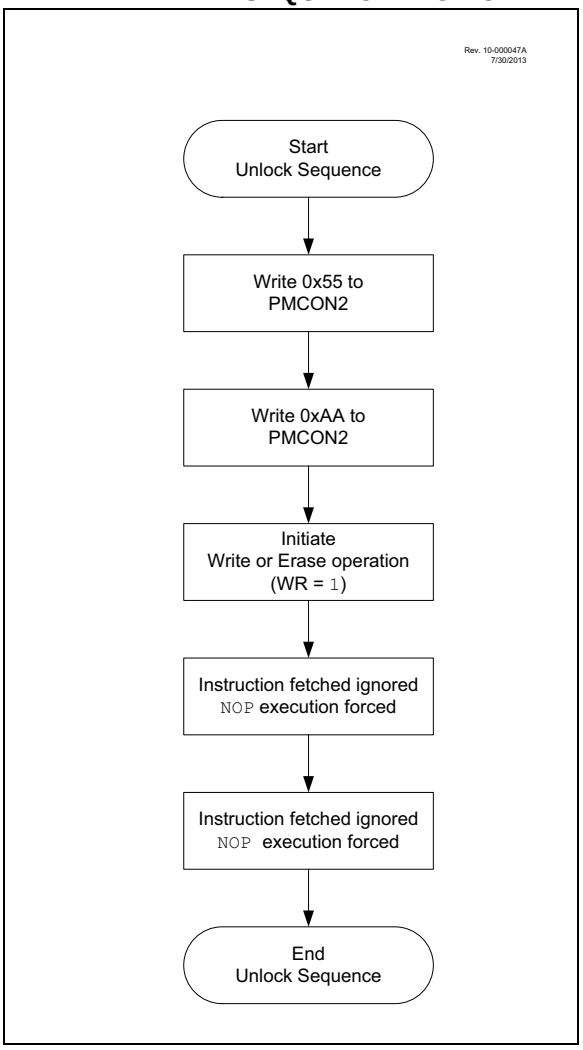
The unlock sequence consists of the following steps:

1. Write 55h to PMCON2
2. Write AAh to PMCON2
3. Set the WR bit in PMCON1
4. NOP instruction
5. NOP instruction

Once the WR bit is set, the processor will always force two NOP instructions. When an Erase Row or Program Row operation is being performed, the processor will stall internal operations (typical 2 ms), until the operation is complete and then resume with the next instruction. When the operation is loading the program memory write latches, the processor will always force the two NOP instructions and continue uninterrupted with the next instruction.

Since the unlock sequence must not be interrupted, global interrupts should be disabled prior to the unlock sequence and re-enabled after the unlock sequence is completed.

**FIGURE 10-3: FLASH PROGRAM MEMORY UNLOCK SEQUENCE FLOWCHART**




 © 2015-2016 Microchip Technology Inc.

### 10.2.3 ERASING FLASH PROGRAM MEMORY

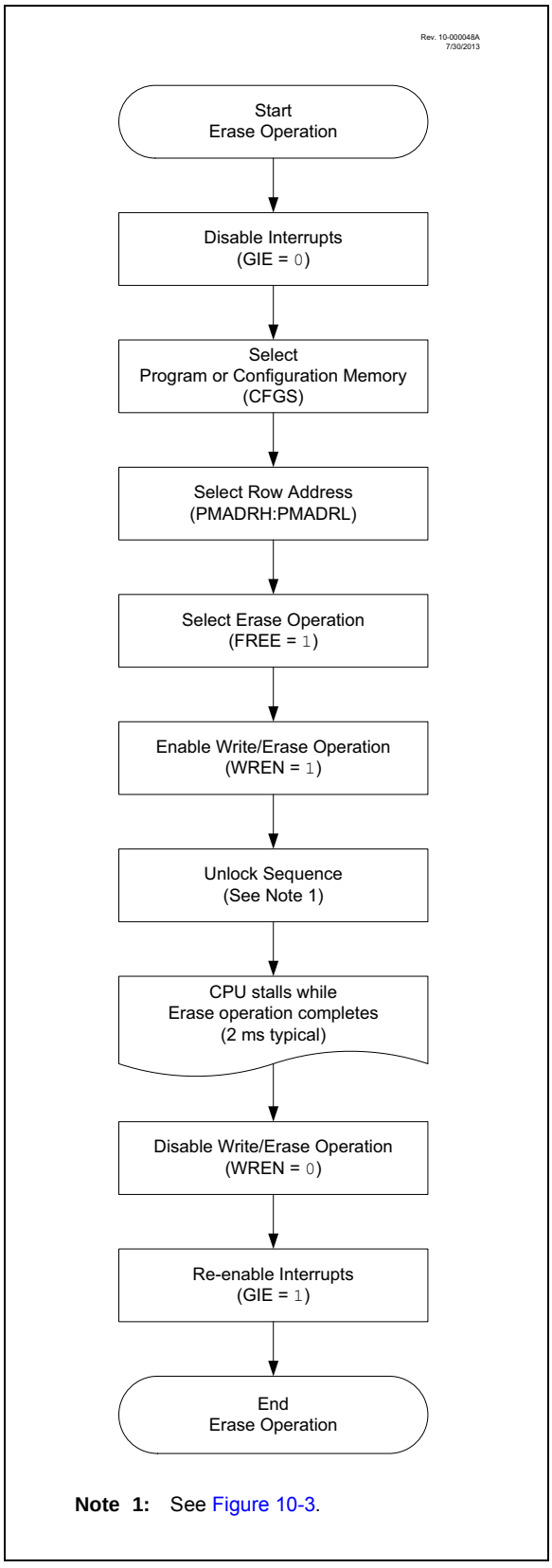
While executing code, program memory can only be erased by rows. To erase a row:

1. Load the PMADRH:PMADRL register pair with any address within the row to be erased.
2. Clear the CFGS bit of the PMCON1 register.
3. Set the FREE and WREN bits of the PMCON1 register.
4. Write 55h, then AAh, to PMCON2 (Flash programming unlock sequence).
5. Set control bit WR of the PMCON1 register to begin the erase operation.

See Example 10-2.

After the "`BSF PMCON1,WR`" instruction, the processor requires two cycles to set up the erase operation. The user must place two `NOP` instructions immediately following the WR bit set instruction. 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 PMCON1 write instruction.

**FIGURE 10-4: FLASH PROGRAM MEMORY ERASE FLOWCHART**




**Note 1:** See Figure 10-3.

**EXAMPLE 10-2: ERASING ONE ROW OF PROGRAM MEMORY**

```
; This row erase routine assumes the following:
; 1. A valid address within the erase row 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     PMADRL
        MOVF        ADDRL,W         ; Load lower 8 bits of erase address boundary
        MOVWF       PMADRL
        MOVF        ADDRH,W         ; Load upper 6 bits of erase address boundary
        MOVWF       PMADRH
        BCF         PMCON1,CFGS     ; Not configuration space
        BSF         PMCON1,FREE     ; Specify an erase operation
        BSF         PMCON1,WREN     ; Enable writes

Required
Sequence
        MOVLW       55h             ; Start of required sequence to initiate erase
        MOVWF       PMCON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       PMCON2          ; Write AAh
        BSF         PMCON1,WR       ; Set WR bit to begin erase
        NOP                         ; NOP instructions are forced as processor starts
        NOP                         ; row erase of program memory.
                                    ;
                                    ; The processor stalls until the erase process is complete
                                    ; after erase processor continues with 3rd instruction

        BCF         PMCON1,WREN     ; Disable writes
        BSF         INTCON,GIE      ; Enable interrupts
```

 © 2015-2016 Microchip Technology Inc.

### 10.2.4 WRITING TO FLASH PROGRAM MEMORY

Program memory is programmed using the following steps:

1. Load the address in PMADRH:PMADRL of the row to be programmed.
2. Load each write latch 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 10-5 (row writes to program memory with 32 write latches) for more details.

The write latches are aligned to the Flash row address boundary defined by the upper ten bits of PMADRH:PMADRL, (PMADRH<6:0>:PMADRL<7:5>) with the lower five bits of PMADRL, (PMADRL<4:0>) determining the write latch being loaded. Write operations do not cross these boundaries. At the completion of a program memory write operation, the data in the write latches is reset to contain 0x3FFF.

The following steps should be completed to load the write latches and program a row of program memory. These steps are divided into two parts. First, each write latch is loaded with data from the PMDATH:PMDATL using the unlock sequence with LWLO = 1. When the last word to be loaded into the write latch is ready, the LWLO bit is cleared and the unlock sequence executed. This initiates the programming operation, writing all the latches into Flash program memory.

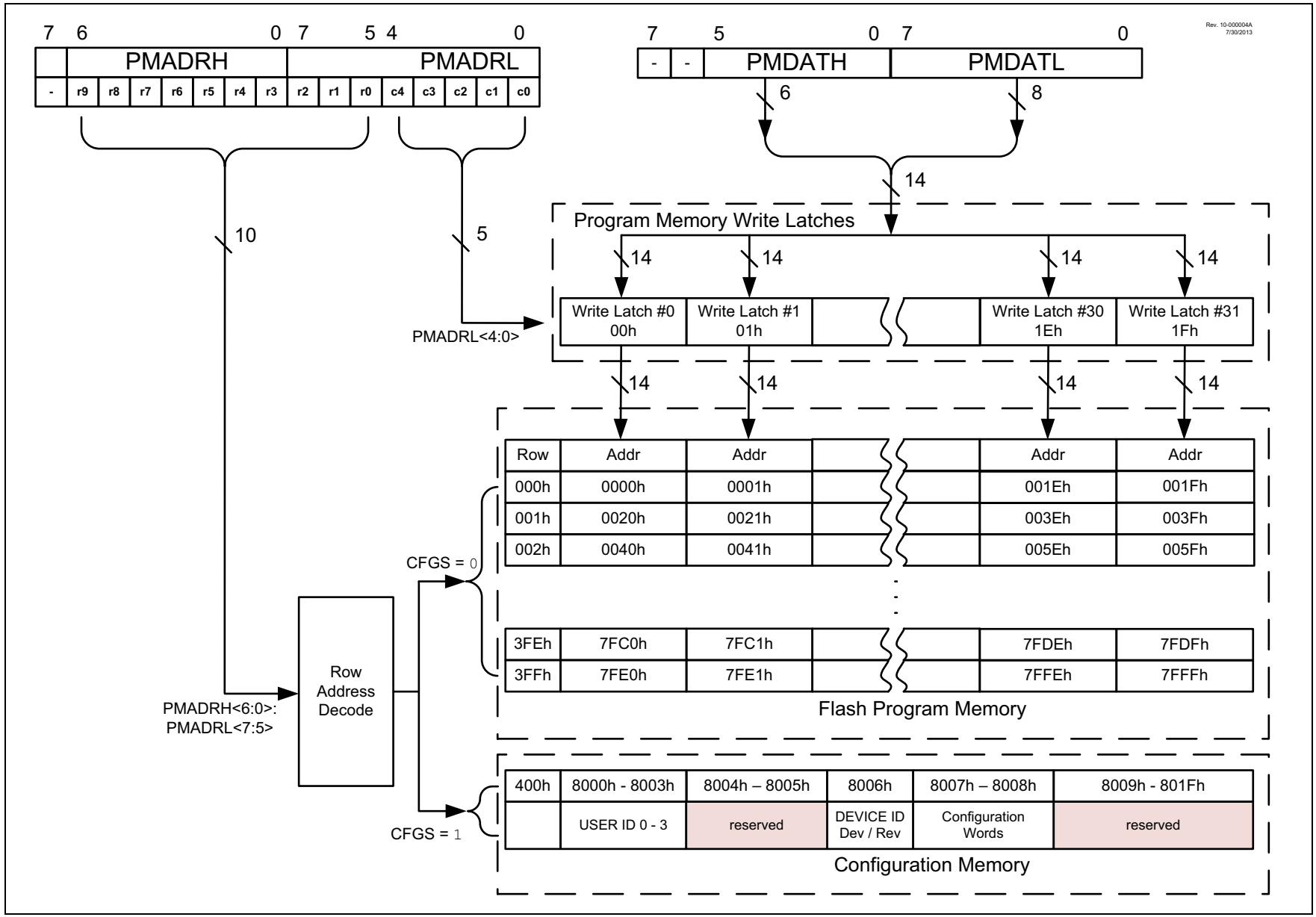
| | |
|---|---|
| **Note:** | The special unlock sequence is required to load a write latch with data or initiate a Flash programming operation. If the unlock sequence is interrupted, writing to the latches or program memory will not be initiated. |

1. Set the WREN bit of the PMCON1 register.
2. Clear the CFGS bit of the PMCON1 register.
3. Set the LWLO bit of the PMCON1 register. When the LWLO bit of the PMCON1 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 PMADRH:PMADRL register pair with the address of the location to be written.
5. Load the PMDATH:PMDATL register pair with the program memory data to be written.
6. Execute the unlock sequence (**Section 10.2.2 "Flash Memory Unlock Sequence"**). The write latch is now loaded.
7. Increment the PMADRH:PMADRL 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 PMCON1 register. When the LWLO bit of the PMCON1 register is '0', the write sequence will initiate the write to Flash program memory.
10. Load the PMDATH:PMDATL register pair with the program memory data to be written.
11. Execute the unlock sequence (**Section 10.2.2 "Flash Memory Unlock Sequence"**). The entire program memory latch content is now written to Flash program memory.

| | |
|---|---|
| **Note:** | The program memory write latches are reset to the blank state (0x3FFF) at the completion of every write or erase operation. As a result, it is not necessary to load all the program memory write latches. Unloaded latches will remain in the blank state. |

An example of the complete write sequence is shown in Example 10-3. The initial address is loaded into the PMADRH:PMADRL register pair; the data is loaded using indirect addressing.

**FIGURE 10-5: BLOCK WRITES TO FLASH PROGRAM MEMORY WITH 32 WRITE LATCHES**

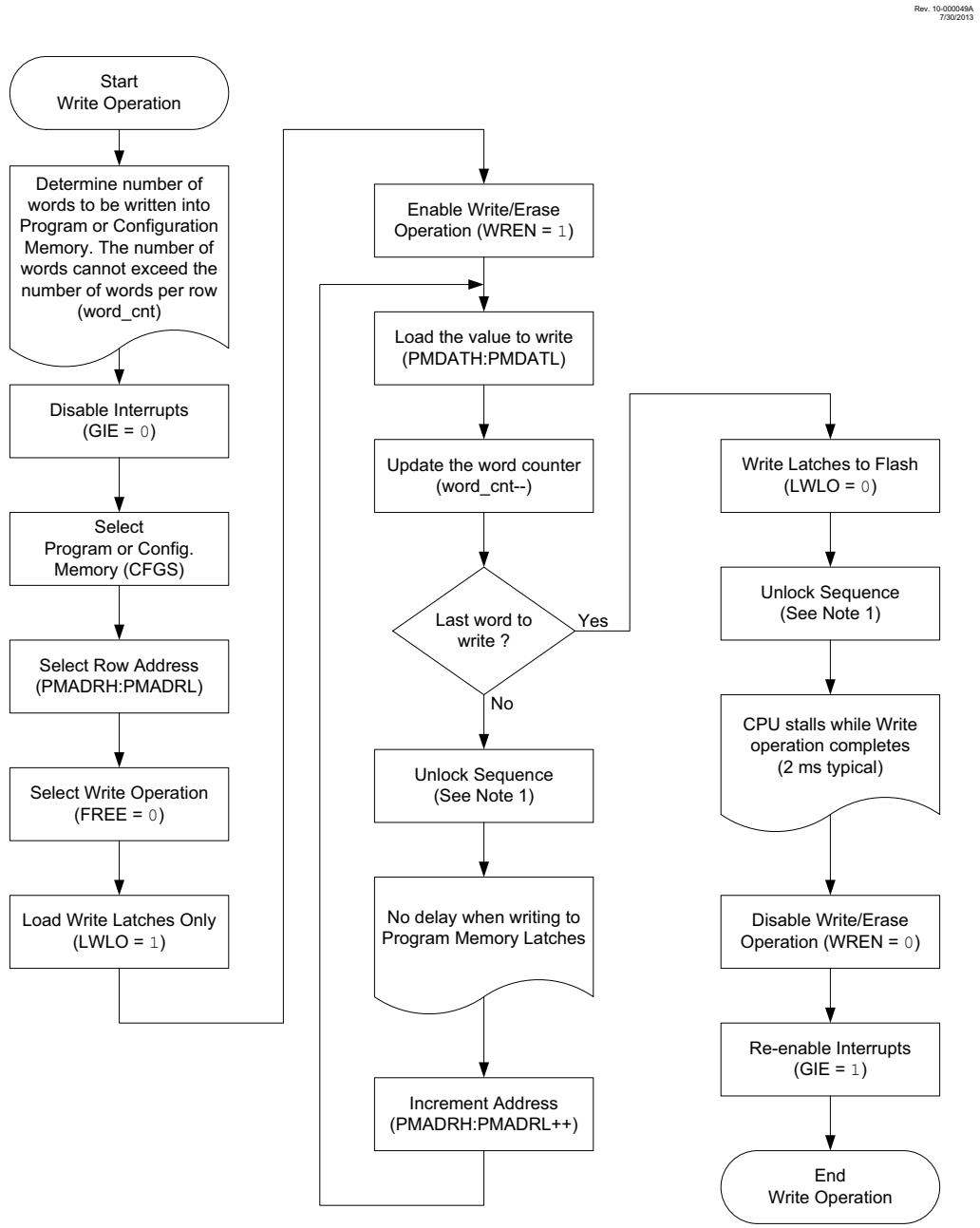





Preliminary

© 2015-2016 Microchip Technology Inc.

**FIGURE 10-6: FLASH PROGRAM MEMORY WRITE FLOWCHART**




**Note 1:** See Figure 10-3.

**EXAMPLE 10-3: WRITING TO FLASH PROGRAM MEMORY (32 WRITE LATCHES)**

```
; This write routine assumes the following:
; 1. 64 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 = 00000) 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     PMADRH          ; Bank 3
        MOVF        ADDRH,W         ; Load initial address
        MOVWF       PMADRH          ;
        MOVF        ADDRL,W         ;
        MOVWF       PMADRL          ;
        MOVLW       LOW DATA_ADDR   ; Load initial data address
        MOVWF       FSR0L           ;
        MOVLW       HIGH DATA_ADDR  ; Load initial data address
        MOVWF       FSR0H           ;
        BCF         PMCON1,CFGS     ; Not configuration space
        BSF         PMCON1,WREN     ; Enable writes
        BSF         PMCON1,LWLO     ; Only Load Write Latches
LOOP
        MOVIW       FSR0++          ; Load first data byte into lower
        MOVWF       PMDATL          ;
        MOVIW       FSR0++          ; Load second data byte into upper
        MOVWF       PMDATH          ;

        MOVF        PMADRL,W        ; Check if lower bits of address are '00000'
        XORLW       0x1F            ; Check if we're on the last of 32 addresses
        ANDLW       0x1F            ;
        BTFSC       STATUS,Z        ; Exit if last of 32 words,
        GOTO        START_WRITE     ;

  Required Sequence
        MOVLW       55h             ; Start of required write sequence:
        MOVWF       PMCON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       PMCON2          ; Write AAh
        BSF         PMCON1,WR       ; Set WR bit to begin write
        NOP                         ; NOP instructions are forced as processor
                                    ; loads program memory write latches
        NOP                         ;

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

START_WRITE
        BCF         PMCON1,LWLO     ; No more loading latches - Actually start Flash program
                                    ; memory write

  Required Sequence
        MOVLW       55h             ; Start of required write sequence:
        MOVWF       PMCON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       PMCON2          ; Write AAh
        BSF         PMCON1,WR       ; Set WR bit to begin write
        NOP                         ; NOP instructions are forced as processor writes
                                    ; all the program memory write latches simultaneously
        NOP                         ; to program memory.
                                    ; After NOPs, the processor
                                    ; stalls until the self-write process in complete
                                    ; after write processor continues with 3rd instruction
        BCF         PMCON1,WREN     ; Disable writes
        BSF         INTCON,GIE      ; Enable interrupts
```

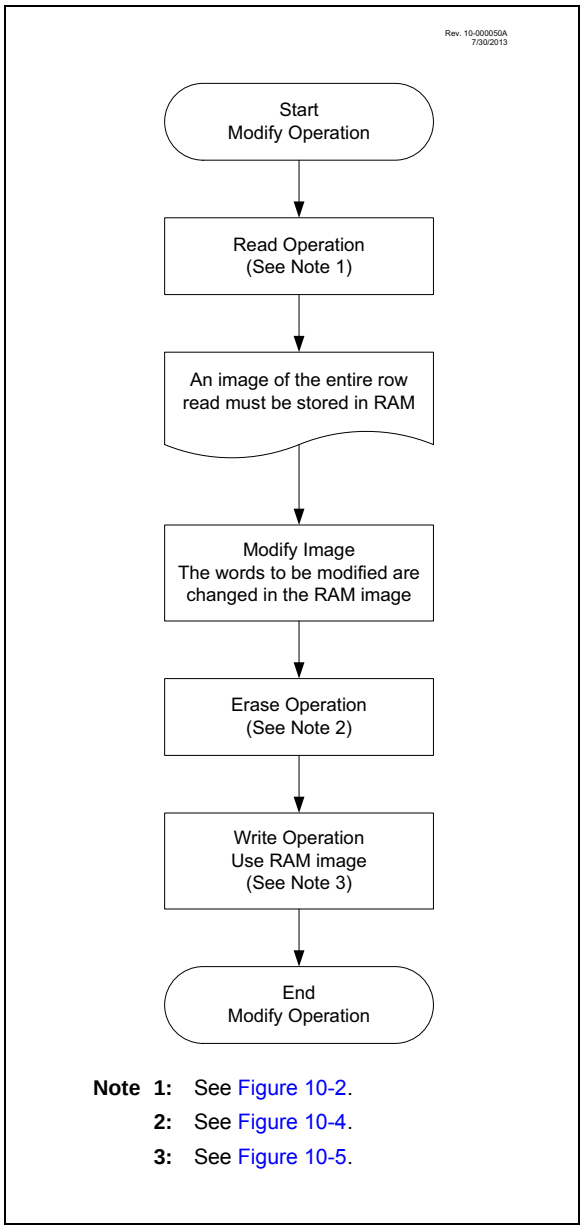
 © 2015-2016 Microchip Technology Inc.

## 10.3 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.

**FIGURE 10-7: FLASH PROGRAM MEMORY MODIFY FLOWCHART**




**Note 1:** See Figure 10-2.
**2:** See Figure 10-4.
**3:** See Figure 10-5.

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

Instead of accessing program memory, the User ID's, Device ID/Revision ID and Configuration Words can be accessed when CFGS = 1 in the PMCON1 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 10-2.

When read access is initiated on an address outside the parameters listed in Table 10-2, the PMDATH:PMDATL register pair is cleared, reading back '0's.

**TABLE 10-2: USER ID, DEVICE ID AND CONFIGURATION WORD ACCESS (CFGS = 1)**

| Address | Function | Read Access | Write Access |
|---|---|---|---|
| 8000h-8003h | User IDs | Yes | Yes |
| 8005h-8006h | Device ID/Revision ID | Yes | No |
| 8007h-8008h | Configuration Words 1 and 2 | Yes | No |

**EXAMPLE 10-4: CONFIGURATION WORD AND DEVICE ID ACCESS**

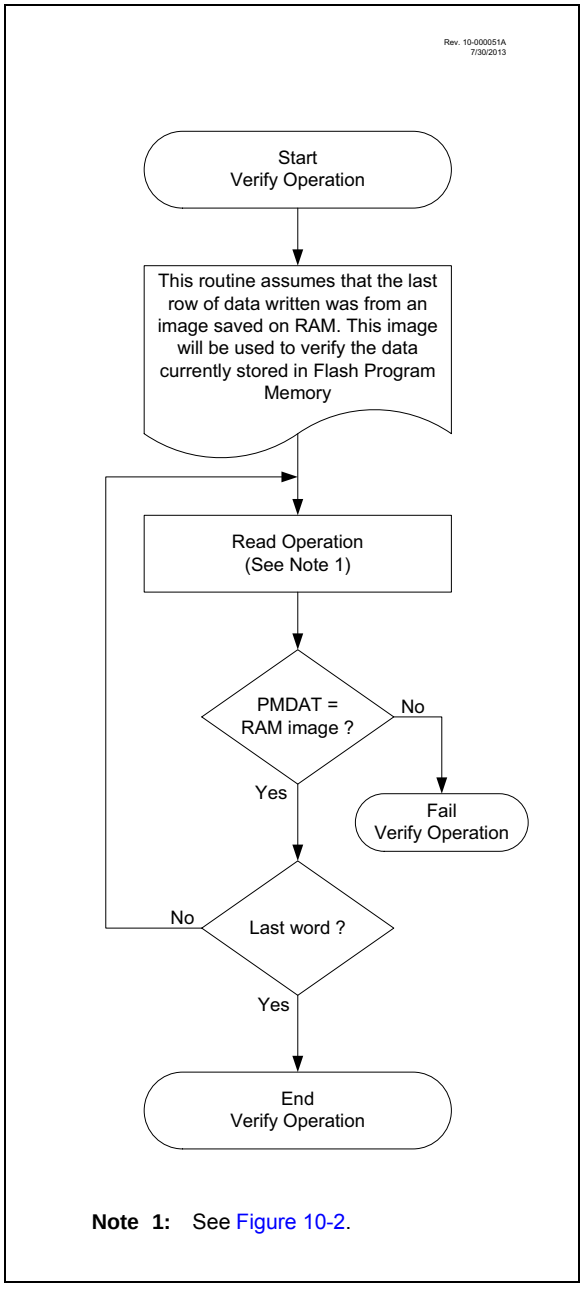
```
* This code block will read 1 word of program memory at the memory address:
*   PROG_ADDR_LO (must be 00h-08h) data will be returned in the variables;
*   PROG_DATA_HI, PROG_DATA_LO

    BANKSEL  PMADRL           ; Select correct Bank
    MOVLW    PROG_ADDR_LO     ;
    MOVWF    PMADRL           ; Store LSB of address
    CLRF     PMADRH           ; Clear MSB of address

    BSF      PMCON1,CFGS      ; Select Configuration Space
    BCF      INTCON,GIE       ; Disable interrupts
    BSF      PMCON1,RD        ; Initiate read
    NOP                       ; Executed (See Figure 10-2)
    NOP                       ; Ignored (See Figure 10-2)
    BSF      INTCON,GIE       ; Restore interrupts

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

 © 2015-2016 Microchip Technology Inc.

## 10.5 Write Verify

It is considered good programming practice to verify that program memory writes agree with the intended value. Since program memory is stored as a full page then the stored program memory contents are compared with the intended data stored in RAM after the last write is complete.

**FIGURE 10-8: FLASH PROGRAM MEMORY VERIFY FLOWCHART**




**Note 1:** See Figure 10-2.

## 10.6 Register Definitions: Flash Program Memory Control

**REGISTER 10-1: PMDATL: PROGRAM MEMORY 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 |
|---|---|---|---|---|---|---|---|
| PMDAT<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 **PMDAT<7:0>**: Read/write value for Least Significant bits of program memory

**REGISTER 10-2: PMDATH: PROGRAM MEMORY 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 |
|---|---|---|---|---|---|---|---|
| — | — | PMDAT<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 **PMDAT<13:8>**: Read/write value for Most Significant bits of program memory

 © 2015-2016 Microchip Technology Inc.

**REGISTER 10-3: PMADRL: PROGRAM MEMORY ADDRESS 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 |
|---|---|---|---|---|---|---|---|
| PMADR<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 **PMADR<7:0>**: Specifies the Least Significant bits for program memory address

**REGISTER 10-4: PMADRH: PROGRAM MEMORY 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 |
|---|---|---|---|---|---|---|---|
| — | PMADR<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 **PMADR<14:8>**: Specifies the Most Significant bits for program memory address

**REGISTER 10-5: PMCON1: PROGRAM MEMORY CONTROL 1 REGISTER**

| U-1 | R/W-0/0 | R/W-0/0 | R/W/HC-0/0 | R/W/HC-x/q | R/W-0/0 | R/S/HC-0/0 | R/S/HC-0/0 |
|---|---|---|---|---|---|---|---|
| — | 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 **Unimplemented:** Read as '1'

bit 6 **CFGS:** Configuration Select bit
- 1 = Access Configuration, User ID and Device ID Registers
- 0 = Access Flash program memory

bit 5 **LWLO:** Load Write Latches Only bit(1)
- 1 = Only the addressed program memory write latch is loaded/updated on the next WR command
- 0 = The addressed program memory write latch is loaded/updated and a write of all program memory write latches will be initiated on the next WR command

bit 4 **FREE:** Program Flash Erase Enable bit
- 1 = Performs an erase operation on the next WR command (hardware cleared upon completion)
- 0 = Performs a write operation on the next WR command

bit 3 **WRERR:** Program/Erase Error Flag bit(2)
- 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

bit 1 **WR:** Write Control bit
- 1 = Initiates a program Flash 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 is complete and inactive.

bit 0 **RD:** Read Control bit
- 1 = Initiates a program Flash 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 read.

**Note 1:** The LWLO bit is ignored during a program memory erase operation (FREE = 1).
**2:** The WRERR bit is automatically set by hardware when a program memory write or erase operation is started (WR = 1).

 © 2015-2016 Microchip Technology Inc.

**REGISTER 10-6: PMCON2: PROGRAM MEMORY 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 |
|---|---|---|---|---|---|---|---|
| Program Memory 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 **Flash Memory Unlock Pattern bits**

To unlock writes, a 55h must be written first, followed by an AAh, before setting the WR bit of the PMCON1 register. The value written to this register is used to unlock the writes. There are specific timing requirements on these writes.

**TABLE 10-3: SUMMARY OF REGISTERS ASSOCIATED WITH FLASH PROGRAM MEMORY**

| 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 |
| PMCON1 | —(1) | CFGS | LWLO | FREE | WRERR | WREN | WR | RD | 110 |
| PMCON2 | Program Memory Control Register 2 | | | | | | | | 111 |
| PMADRL | PMADR<7:0> | | | | | | | | 109 |
| PMADRH | —(1) | PMADR<14:8> | | | | | | | 109 |
| PMDATL | PMDAT<7:0> | | | | | | | | 108 |
| PMDATH | — | — | PMDAT<13:8> | | | | | | 108 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by Flash program memory.
**Note 1:** Unimplemented, read as '1'.

**TABLE 10-4: SUMMARY OF CONFIGURATION WORD WITH FLASH PROGRAM MEMORY**

| 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 | — | — | — | — | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | — | 59 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | — | FOSC<1:0> | | |
| CONFIG2 | 13:8 | — | — | LVP | $\overline{\text{DEBUG}}$ | $\overline{\text{LPBOR}}$ | BORV | STVREN | — | 60 |
| | 7:0 | — | — | — | — | — | — | WRT<1:0> | | |

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

## 11.0 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 11-1: PORT AVAILABILITY PER DEVICE**

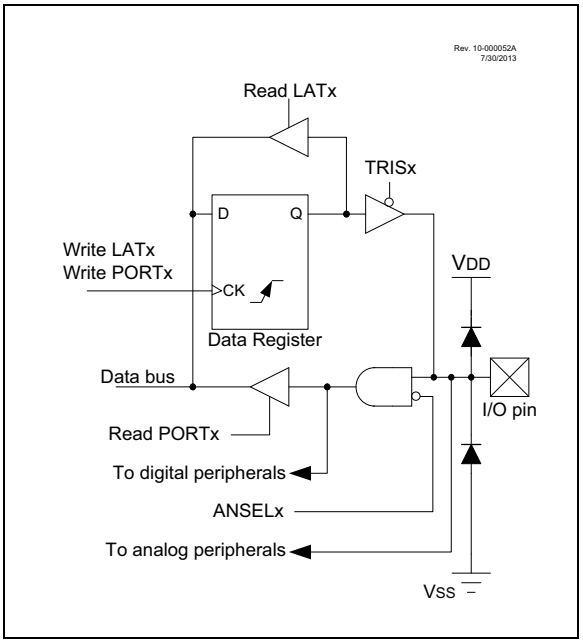
| Device | PORTA | PORTB | PORTC | PORTD | PORTE |
|---|---|---|---|---|---|
| PIC16LF1566 | • | • | • | — | • |
| PIC16LF1567 | • | • | • | • | • |

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 ANSELx 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 11-1.

**FIGURE 11-1: GENERIC I/O PORT OPERATION**




## 11.1 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 11-1. For this device family, the following functions can be moved between different pins.

- SS1
- AD1GRDA
- AD1GRDB
- AD2GRDA
- AD2GRDB

These bits have no effect on the values of any TRISx register. PORTx and TRISx overrides will be routed to the correct pin. The unselected pin will be unaffected.

 © 2015-2016 Microchip Technology Inc.

## 11.2 Register Definitions: Alternate Pin Function Control

**REGISTER 11-1: APFCON: ALTERNATE PIN FUNCTION CONTROL REGISTER**

| U-0 | U-0 | R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| | | SSSEL | — | — | — | GRDBSEL | GRDASEL |
| 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 **Unimplemented:** Read as ‘0’

bit 5 **SSSEL:** Pin Selection
1 = MSSP1 $\overline{SS}$ function is on RA0
0 = MSSP1 $\overline{SS}$ function is on RA5

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **GRDBSEL:** Pin Selection
1 = AD1GRDB function is on RB7, AD2GRDB function is on RB6
0 = AD1GRDB function is on RB6, AD2GRDB function is on RB7

bit 0 **GRDASEL:** Pin Selection bit
1 = AD1GRDA function is on RB5, AD2GRDA function is on RB4
0 = AD1GRDA function is on RB4, AD2GRDA function is on RB5

## 11.3 PORTA Registers

### 11.3.1 DATA REGISTER

PORTA is a 6-bit wide, bidirectional port. The corresponding data direction register is TRISA (Register 11-3). 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 TRISx bit will always read as '1'. Example 11-1 shows how to initialize an I/O port.

Reading the PORTA register (Register 11-2) 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).

### 11.3.2 DIRECTION CONTROL

The TRISA register (Register 11-3) 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 input always read '0'.

### 11.3.3 ANALOG CONTROL

The ANSELA register (Register 11-5) 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 ANSELx bits must be initialized to '0' by user software. |

**EXAMPLE 11-1: INITIALIZING PORTA**

```
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
```

### 11.3.4 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 11-2.

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 below in Table 11-2.

**TABLE 11-2: PORTA OUTPUT PRIORITY**

| Pin Name | Function Priority(1) |
|---|---|
| RA0 | SS1<br>PWM10<br>RA0 |
| RA1 | SS2<br>PWM11<br>RA1 |
| RA2 | PWM12<br>RA2 |
| RA3 | VREF+<br>PWM13<br>RA3 |
| RA4 | T0CKI<br>RA4 |
| RA5 | SS1<br>RA5 |
| RA6 | CLKOUT<br>ADTRIG<br>RA6 |
| RA7 | CLKIN<br>RA7 |

**Note 1:** Priority listed from highest to lowest.


© 2015-2016 Microchip Technology Inc.

## 11.4 Register Definitions: PORTA

**REGISTER 11-2: 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>**: RA7:RA0 PORTA I/O Value bits(1)
1 = Port pin is ≥ VIH
0 = Port pin is ≤ VIL

**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 11-3: TRISA: PORTA 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 |
|---|---|---|---|---|---|---|---|
| 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 7-0 **TRISA<7:0>:** PORTA Tri-State Control bit
1 = PORTA pin configured as an input (tri-stated)
0 = PORTA pin configured as an output

**REGISTER 11-4: 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 7-0 **LATA<7:0>**: RA<7:0> 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 11-5: ANSELA: PORTA ANALOG SELECT 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 |
|---|---|---|---|---|---|---|---|
| ANSA7 | ANSA6 | 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 7-0 **ANSA<7:0>**: Analog Select between Analog or Digital Function on pins RA4<7:0>, respectively
1 = Analog input. Pin is assigned as analog input(1). Digital input buffer disabled.
0 = Digital I/O. Pin is assigned to port or digital special function.

**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.

 © 2015-2016 Microchip Technology Inc.

**TABLE 11-3: 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 | ANSA6 | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 116 |
| APFCON | — | — | SSSEL | — | — | — | GRDBSEL | GRDASEL | 113 |
| LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | 116 |
| OPTION_REG | $\overline{\text{WPUEN}}$ | INTEDG | TMR0CS | TMR0SE | PSA | PS<2:0> | | | 181 |
| PORTA | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 115 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 115 |

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

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

**TABLE 11-4: 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 | — | — | — | — | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | — | 59 |
| | 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.

## 11.5 PORTB Registers (PIC16LF1567 Only)

### 11.5.1 DATA REGISTER

PORTB is a 4-bit wide, bidirectional port. The corresponding data direction register is TRISB (Register 11-7). Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., disable the output driver). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., enables output driver and puts the contents of the output latch on the selected pin). Example 11-1 shows how to initialize an I/O port.

Reading the PORTB register (Register 11-6) 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).

### 11.5.2 DIRECTION CONTROL

The TRISB register (Register 11-7) 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 input always read '0'.

### 11.5.3 ANALOG CONTROL

The ANSELB register (Register 11-9) 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 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 ANSELB bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSELx bits must be initialized to '0' by user software. |

### 11.5.4 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 11-5.

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 below in Table 11-5.

**TABLE 11-5: PORTB OUTPUT PRIORITY**

| Pin Name | Function Priority(1) |
|---|---|
| RB0 | INT<br>PWM20<br>RB0 |
| RB1 | PWM21<br>RB1 |
| RB2 | PWM22<br>RB2 |
| RB3 | PWM23<br>RB3 |
| RB4 | ADxGRDA<br>RB4 |
| RB5 | ADxGRDA<br>RB5 |
| RB6 | ICSPCLK<br>ADxGRDB<br>RB6 |
| RB7 | ICSPDAT<br>ADxGRDB<br>RB7 |

**Note 1:** Priority listed from highest to lowest.


© 2015-2016 Microchip Technology Inc.

## 11.6 Register Definitions: PORTB

**REGISTER 11-6: PORTB: PORTB REGISTER**

| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
|---|---|---|---|---|---|---|---|
| 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 I/O Value bits(1)
1 = Port pin is ≥ VIH
0 = Port pin is ≤ VIL

**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 11-7: 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 11-8: 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>**: RB<7:0> 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 11-9: ANSELB: PORTB ANALOG SELECT 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 |
|---|---|---|---|---|---|---|---|
| ANSB7 | ANSB6 | 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 7-0 **ANSB<7:0>:** Analog Select between Analog or Digital Function on pins RB<5:4>, respectively
1 = Analog input. Pin is assigned as analog input(1). Digital input buffer disabled.
0 = Digital I/O. Pin is assigned to port or digital special function.

**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.

 © 2015-2016 Microchip Technology Inc.

**REGISTER 11-10: WPUB: WEAK PULL-UP PORTB REGISTER(1,2)**

| 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 configured as an output.

**TABLE 11-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB(1)**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| ANSELB | ANSB7 | ANSB6 | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 120 |
| APFCON | — | — | SSSEL | — | — | — | GRDBSEL | GRDASEL | 113 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | 120 |
| OPTION_REG | $\overline{\text{WPUEN}}$ | INTEDG | TMR0CS | TMR0SE | PSA | PS<2:0> | | | 181 |
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | 119 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 | 121 |

**Legend:** x = unknown, u = unchanged, – = unimplemented locations read as '0'. Shaded cells are not used by PORTB.
**Note 1:** PIC16LF1567 only.

**TABLE 11-7: SUMMARY OF CONFIGURATION WORD WITH PORTB**

| 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 | — | — | — | — | $\overline{\text{CLKOUTEN}}$ | BOREN<1:0> | | — | 59 |
| | 7:0 | $\overline{\text{CP}}$ | MCLRE | $\overline{\text{PWRTE}}$ | WDTE<1:0> | | — | FOSC<1:0> | | |

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

## 11.7 PORTC Registers

### 11.7.1 DATA REGISTER

PORTC is a 8-bit wide, bidirectional port. The corresponding data direction register is TRISC (Register 11-12). Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., disable the output driver). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin). Example 11-1 shows how to initialize an I/O port.

Reading the PORTC register (Register 11-11) 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).

### 11.7.2 DIRECTION CONTROL

The TRISC register (Register 11-12) 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 input always read '0'.

### 11.7.3 ANALOG CONTROL

The ANSELC register (Register 11-14) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELC 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 ANSELC 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 ANSELC bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSELx bits must be initialized to '0' by user software. |

### 11.7.4 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 11-8.

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 output priority list. 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 output priority list.

**TABLE 11-8: PORTC OUTPUT PRIORITY**

| Pin Name | Function Priority(1) |
|---|---|
| RC0 | T1CKI<br>SDO2<br>RC0 |
| RC1 | SCK2<br>SCL2<br>PWM2<br>RC1 |
| RC2 | SDA2<br>SDI2<br>PWM1<br>RC2 |
| RC3 | SCK1<br>SCL1<br>RC3 |
| RC4 | SDA1<br>SDI1<br>RC4 |
| RC5 | I2CLVL<br>SDO1<br>RC5 |
| RC6 | TX<br>CK<br>RC6 |
| RC7 | RX<br>DT<br>RC7 |

**Note 1:** Priority listed from highest to lowest.

 © 2015-2016 Microchip Technology Inc.

## 11.8 Register Definitions: PORTC

**REGISTER 11-11: 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 = Port pin is ≥ VIH
0 = Port pin is ≤ VIL

**REGISTER 11-12: 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 11-13: 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 11-14: ANSELC: PORTC ANALOG SELECT 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 |
|---|---|---|---|---|---|---|---|
| ANSC7 | ANSC6 | ANSC5 | ANSC4 | ANSC3 | ANSC2 | ANSC1 | ANSC0 |
| 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 **ANSC<7:0>**: Analog Select between Analog or Digital Function on pins RC<7:0>, respectively
1 = Analog input. Pin is assigned as analog input(1). Digital input buffer disabled.
0 = Digital I/O. Pin is assigned to port or digital special function.

**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.

**TABLE 11-9: 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 |
|---|---|---|---|---|---|---|---|---|---|
| ANSELC | ANSC7 | ANSC6 | ANSC5 | ANSC4 | ANSC3 | ANSC2 | ANSC1 | ANSC0 | 124 |
| LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | 124 |
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | 123 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 123 |

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


© 2015-2016 Microchip Technology Inc.

## 11.9 PORTD Registers

### 11.9.1 DATA REGISTER

PORTD is a 8-bit wide, bidirectional port, for PIC16LF1567 only. The corresponding data direction register is TRISD (Register 11-12). Setting a TRISD bit (= 1) will make the corresponding PORTD pin an input (i.e., disable the output driver). Clearing a TRISD bit (= 0) will make the corresponding PORTD pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin). Example 11-1 shows how to initialize an I/O port.

Reading the PORTD register (Register 11-11) 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 (LATD).

### 11.9.2 DIRECTION CONTROL

The TRISD register (Register 11-12) controls the PORTD pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISD register are maintained set when using them as analog inputs. I/O pins configured as analog input always read '0'.

### 11.9.3 ANALOG CONTROL

The ANSELD register (Register 11-18) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELD 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 ANSELD 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 ANSELD bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSELx bits must be initialized to '0' by user software. |

### 11.9.4 PORTD FUNCTIONS AND OUTPUT PRIORITIES

Each PORTD pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 11-8.

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 output priority list. 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 output priority list.

**TABLE 11-10: PORTD OUTPUT PRIORITY**

| Pin Name | Function Priority(1) |
|---|---|
| RD0 | RD0 |
| RD1 | RD1 |
| RD2 | RD2 |
| RD3 | RD3 |
| RD4 | RD4 |
| RD5 | RD5 |
| RD6 | RD6 |
| RD7 | RD7 |

**Note 1:** Priority listed from highest to lowest.

## 11.10 Register Definitions: PORTD

**REGISTER 11-15: PORTD(1): PORTD 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 |
|---|---|---|---|---|---|---|---|
| RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 |
| 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 **RD<7:0>**: PORTD General Purpose I/O Pin bits
1 = Port pin is $\geq V_{IH}$
0 = Port pin is $\leq V_{IL}$

**Note 1:** Functions not available on PIC16LF1566.

**REGISTER 11-16: TRISD(1): PORTD 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 |
|---|---|---|---|---|---|---|---|
| TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 |
| 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 **TRISD<7:0>:** PORTD Tri-State Control bits
1 = PORTD pin configured as an input (tri-stated)
0 = PORTD pin configured as an output

**Note 1:** Functions not available on PIC16LF1566.

 © 2015-2016 Microchip Technology Inc.

**REGISTER 11-17: LATD(1): PORTD 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 |
|---|---|---|---|---|---|---|---|
| LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 |
| 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 **LATD<7:0>**: PORTD Output Latch Value bits(2)

**Note 1:** Functions not available on PIC16LF1566.

**2:** Writes to PORTD are actually written to corresponding LATD register. Reads from PORTD register is return of actual I/O pin values.

**REGISTER 11-18: ANSELD(1): PORTD ANALOG SELECT 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 |
|---|---|---|---|---|---|---|---|
| ANSD7 | ANSD6 | ANSD5 | ANSD4 | ANSD3 | ANSD2 | ANSD1 | ANSD0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| 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 **ANSD<7:0>**: Analog Select between Analog or Digital Function on pins RD<7:0>, respectively
1 = Analog input. Pin is assigned as analog input(2). Digital input buffer disabled.
0 = Digital I/O. Pin is assigned to port or digital special function.

**Note 1:** Functions not available on PIC16LF1566.

**2:** 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.

**TABLE 11-11: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| ANSELD(1) | ANSD7 | ANSD6 | ANSD5 | ANSD4 | ANSD3 | ANSD2 | ANSD1 | ANSD0 | 127 |
| LATD(1) | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | 127 |
| PORTD(1) | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | 126 |
| TRISD(1) | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 126 |

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

**Note 1:** Functions not available on PIC16LF1566.

## 11.11 PORTE Registers

### 11.11.1 DATA REGISTER

PORTE is a 8-bit wide, bidirectional port, for PIC16LF1566/1567. The corresponding data direction register is TRISE (Register 11-12). Setting a TRISD bit (= 1) will make the corresponding PORTD pin an input (i.e., disable the output driver). Clearing a TRISE bit (= 0) will make the corresponding PORTE pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin). Example 11-1 shows how to initialize an I/O port.

Reading the PORTE register (Register 11-11) 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 (LATE).

### 11.11.2 DIRECTION CONTROL

The TRISE register (Register 11-12) controls the PORTE pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISE register are maintained set when using them as analog inputs. I/O pins configured as analog input always read '0'.

### 11.11.3 ANALOG CONTROL

The ANSELE register (Register 11-22) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELE 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 ANSELE 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 ANSELE bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSELx bits must be initialized to '0' by user software. |

### 11.11.4 PORTE FUNCTIONS AND OUTPUT PRIORITIES

Each PORTE pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 11-8.

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 output priority list. 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 output priority list.

**TABLE 11-12: PORTE OUTPUT PRIORITY**

| Pin Name | Function Priority(1) |
|---|---|
| RE0 | RE0 |
| RE1 | RE1 |
| RE2 | RE2 |
| RE3 | RE3 |

**Note 1:** Priority listed from highest to lowest.

 © 2015-2016 Microchip Technology Inc.

## 11.12 Register Definitions: PORTE

**REGISTER 11-19: PORTE: PORTE REGISTER**

| U-0 | U-0 | U-0 | U-0 | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|---|---|---|---|---|---|---|---|
| — | — | — | — | RE3(2) | RE2(1) | RE1(1) | RE0(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 7-4 **Unimplemented:** Read as '0'

bit 3 **$\overline{\text{MCLR}}$: RE<3:0>**

bit 2-0 **RE<2:0>:** PORTE General Purpose I/O Pin bits
1 = Port pin is ≥ VIH
0 = Port pin is ≤ VIL

**Note 1:** Functions available on PIC16LF1567 only.
**2:** $\overline{\text{MCLR}}$ bit is implemented by both devices.

**REGISTER 11-20: TRISE(1): PORTE TRI-STATE REGISTER**

| U-1 | U-1 | U-1 | U-1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | TRISE2 | TRISE1 | TRISE0 |
| 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 **TRISE<3:0>:** PORTE Tri-State Control bits
1 = PORTE pin configured as an input (tri-stated)
0 = PORTE pin configured as an output

**Note 1:** Functions available on PIC16LF1567 only.

**REGISTER 11-21: LATE(1): PORTE DATA LATCH REGISTER**

| U-1 | U-1 | U-1 | U-1 | U-1 | R/W-x/u | R/W-x/u | R/W-x/u |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | LATE2 | LATE1 | LATE0 |
| bit 7 | | | | | | | bit 0 |

| Legend: | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| 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 2-0 **LATE<2:0>**: PORTE Output Latch Value bits(2)

**Note 1:** Functions available on PIC16LF1567 only.

**REGISTER 11-22: ANSELE(1): PORTE ANALOG SELECT REGISTER**

| U-1 | U-1 | U-1 | U-1 | U-1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | ANSE2 | ANSE1 | ANSE0 |
| 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 2-0 **ANSE<2:0>**: Analog Select between Analog or Digital Function on pins RE<7:0>, respectively
1 = Analog input. Pin is assigned as analog input(2). Digital input buffer disabled.
0 = Digital I/O. Pin is assigned to port or digital special function.

**Note 1:** Functions available on PIC16LF1567 only.
**2:** 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.

**TABLE 11-13: 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 |
|---|---|---|---|---|---|---|---|---|---|
| ANSELE(1) | — | — | — | — | — | ANSE2 | ANSE1 | ANSE0 | 130 |
| LATE(1) | — | — | — | — | — | LATE2 | LATE1 | LATE0 | 130 |
| PORTE | — | — | — | — | RE3 | RE2 | RE1 | RE0 | 129 |
| TRISE(1) | — | — | — | — | —(2) | TRISE2 | TRISE1 | TRISE0 | 129 |

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

**Note 1:** Functions available on PIC16LF1567 only.
**2:** Unimplemented, read as '1'.


© 2015-2016 Microchip Technology Inc.

## 12.0 INTERRUPT-ON-CHANGE

The PORTA and PORTB pins 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 port pin, or combination of port 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 12-1 is a block diagram of the IOC module.

### 12.1 Enabling the Module

To allow individual port 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.

### 12.2 Individual Pin Configuration

For each port 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 both associated bits of the IOCxP and IOCxN registers, respectively.

### 12.3 Interrupt Flags

The IOCBFx bits located in the IOCBF registers, respectively, are status flags that correspond to the interrupt-on-change pins of the associated 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 IOCBFx bits.

### 12.4 Clearing Interrupt Flags

The individual status flags, (IOCBFx 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 12-1: CLEARING INTERRUPT FLAGS (PORTA EXAMPLE)**

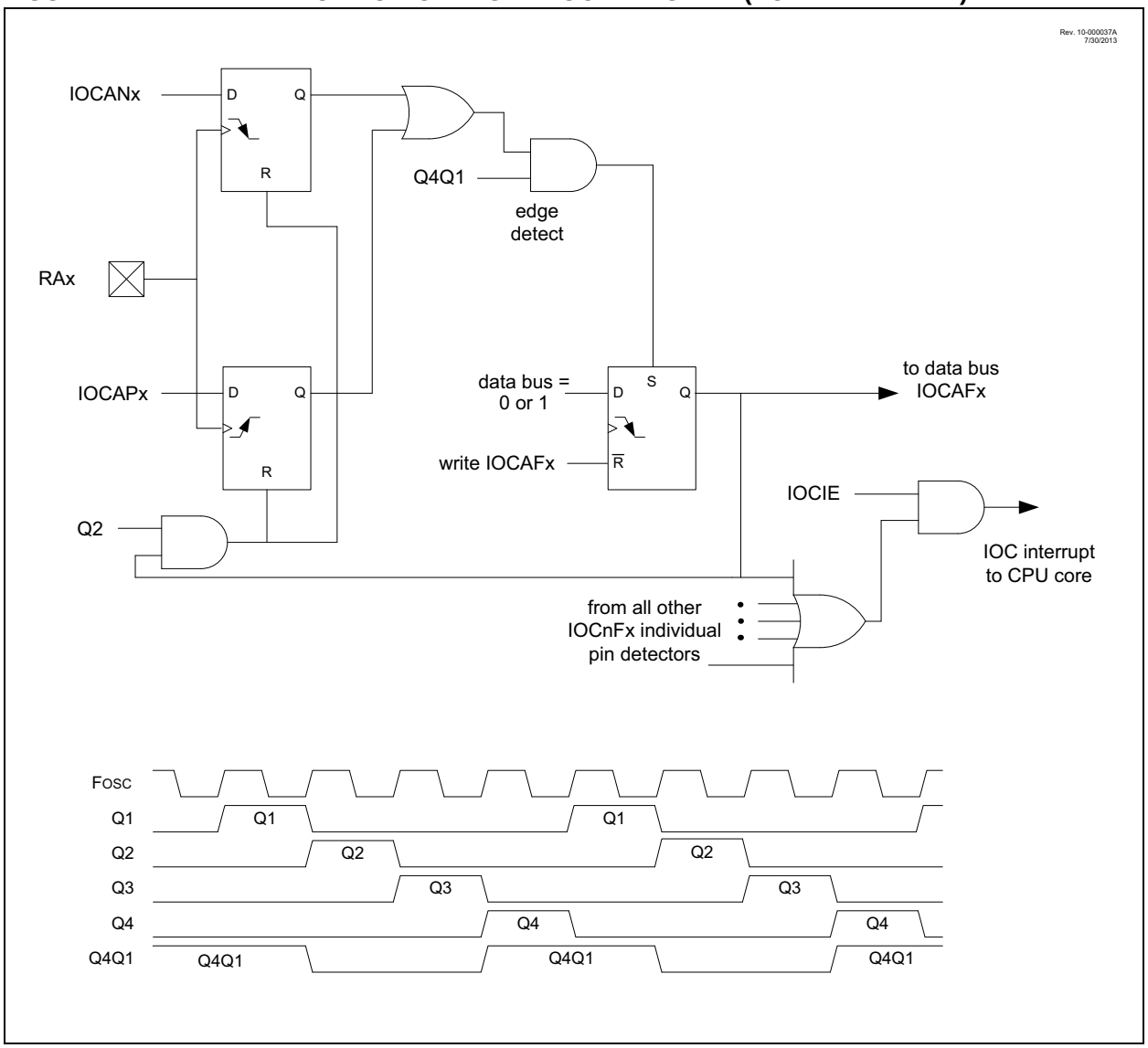
```
MOVLW  0xff
XORWF  IOCAF, W
ANDWF  IOCAF, F
```

### 12.5 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 IOCxF register will be updated prior to the first instruction executed out of Sleep.

**FIGURE 12-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTA EXAMPLE)**




 © 2015-2016 Microchip Technology Inc.

## 12.6 Register Definitions: Interrupt-on-Change Control

### REGISTER 12-1: IOCBP: INTERRUPT-ON-CHANGE PORTB 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 |
|---|---|---|---|---|---|---|---|
| IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 |
| 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 **IOCBP<7:0>:** Interrupt-on-Change PORTB Positive Edge Enable bits
1 = Interrupt-on-Change enabled on the pin for a positive-going edge. IOCBFx bit and IOCIF flag will be set upon detecting an edge.
0 = Interrupt-on-Change disabled for the associated pin.

### REGISTER 12-2: IOCBN: INTERRUPT-ON-CHANGE PORTB 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 |
|---|---|---|---|---|---|---|---|
| IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 |
| 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 **IOCBN<7:0>:** Interrupt-on-Change PORTB Negative Edge Enable bits
1 = Interrupt-on-Change enabled on the pin for a negative-going edge. IOCBFx bit and IOCIF flag will be set upon detecting an edge.
0 = Interrupt-on-Change disabled for the associated pin.

### REGISTER 12-3: IOCBF: INTERRUPT-ON-CHANGE PORTB 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 |
|---|---|---|---|---|---|---|---|
| IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 |
| 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 **IOCBF<7:0>:** Interrupt-on-Change PORTB Flag bits
1 = An enabled change was detected on the associated pin.
Set when IOCBPx = 1 and a rising edge was detected on RBx, or when IOCBNx = 1 and a falling edge was detected on RBx.
0 = No change was detected, or the user cleared the detected change.

**TABLE 12-1: 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 |
|---|---|---|---|---|---|---|---|---|---|
| ANSELA | ANSA7 | ANSA6 | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 116 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| IOCBF | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 | 133 |
| IOCBN | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 | 133 |
| IOCBP | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 | 133 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 115 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.
**Note 1:** Unimplemented, read as '1'.

 © 2015-2016 Microchip Technology Inc.

## 13.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of $V_{DD}$, with 1.024V and 2.048V selectable output levels. The output of the FVR can be configured as the FVR input channel on the ADC.

The FVR can be enabled by setting the FVREN bit of the FVRCON register.

### 13.1 Independent Gain Amplifier

The output of the FVR supplied to the ADC is routed through a programmable gain amplifier. Each amplifier can be programmed for a gain of 1x or 2x, to produce the two 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 15.0 "Analog-to-Digital Converter (ADC) Module"** for additional information.

### 13.2 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 25.0 "Electrical Specifications"** for the minimum delay requirement.

**FIGURE 13-1: VOLTAGE REFERENCE BLOCK DIAGRAM**

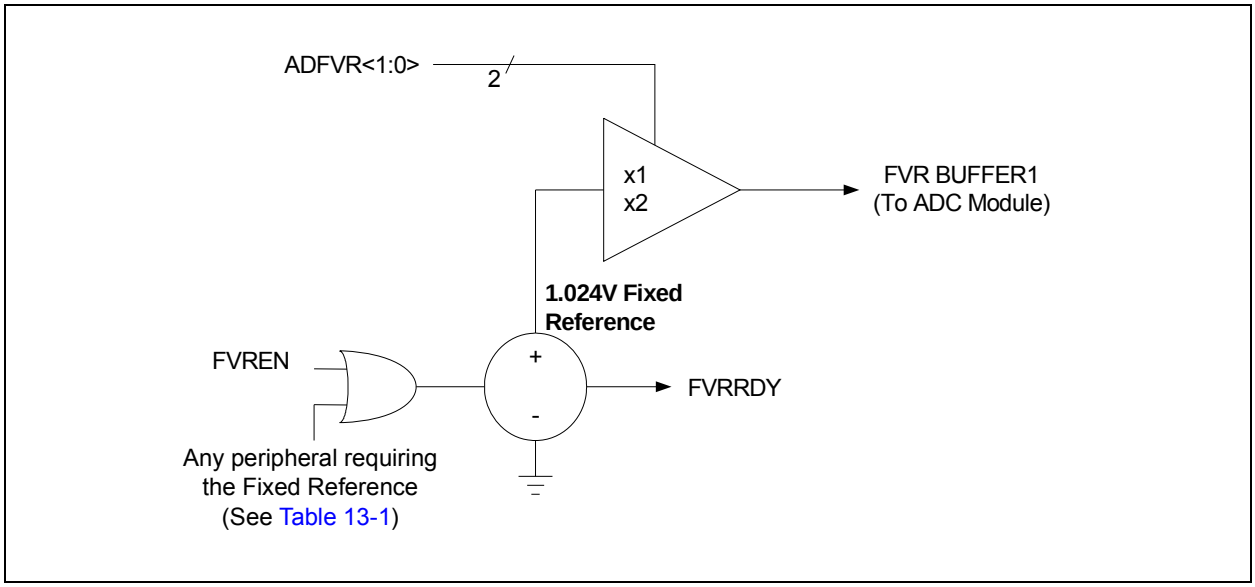



**TABLE 13-1: PERIPHERALS REQUIRING THE FIXED VOLTAGE REFERENCE (FVR)**

| Peripheral | Conditions | Description |
|---|---|---|
| HFINTOSC | FOSC<1:0> = 00 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. |

## 13.3 Register Definitions: FVR Control

**REGISTER 13-1: FVRCON: FIXED VOLTAGE REFERENCE CONTROL REGISTER**

| R/W-0/0 | R-q/q | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| FVREN | FVRRDY | TSEN | TSRNG | — | — | 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 = 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(1)
1 = Temperature Indicator is enabled
0 = Temperature Indicator is disabled

bit 4 **TSRNG:** Temperature Indicator Range Selection bit(1)
1 = $V_{OUT} = V_{DD} - 4V_T$ (High Range)
0 = $V_{OUT} = V_{DD} - 2V_T$ (Low Range)

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

bit 1-0 **ADFVR<1:0>:** ADC Fixed Voltage Reference Selection bit
11 = Reserved
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:** See **Section 14.0 "Temperature Indicator Module"** for additional information.
**2:** Fixed Voltage Reference output cannot exceed $V_{DD}$.

**TABLE 13-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE 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 | — | — | ADFVR<1:0> | | 136 |

**Legend:** Shaded cells are unused by the Fixed Voltage Reference module.


© 2015-2016 Microchip Technology Inc.

## 14.0 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.

### 14.1 Circuit Operation

Figure 14-1 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 14-1 describes the output characteristics of the temperature indicator.

**EQUATION 14-1: VOUT RANGES**

High Range: $V_{OUT} = V_{DD} - 4V_T$

Low Range: $V_{OUT} = V_{DD} - 2V_T$

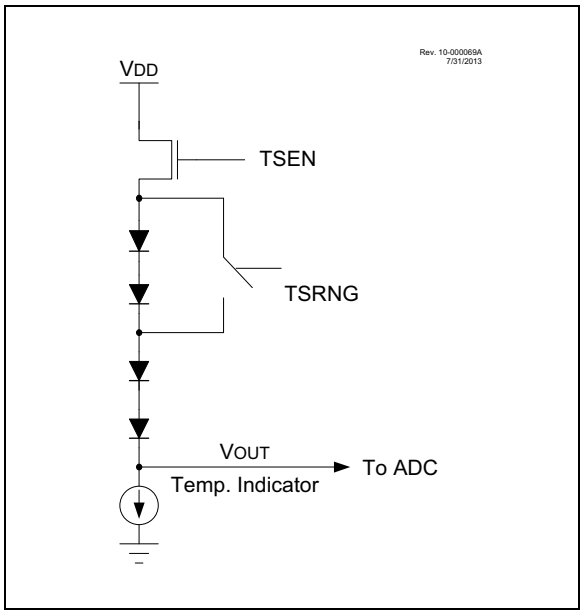
The temperature sense circuit is integrated with the Fixed Voltage Reference (FVR) module. See **Section 13.0 "Fixed Voltage Reference (FVR)"** for more information.

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 VDD 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 14-1: TEMPERATURE CIRCUIT DIAGRAM**




### 14.2 Minimum Operating VDD

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, VDD, must be high enough to ensure that the temperature circuit is correctly biased.

Table 14-1 shows the recommended minimum VDD vs. range setting.

**TABLE 14-1: RECOMMENDED VDD VS. RANGE**

| Min. VDD, TSRNG = 1 | Min. VDD, TSRNG = 0 |
|---|---|
| 3.6V | 1.8V |

### 14.3 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 15.0 "Analog-to-Digital Converter (ADC) Module"** for detailed information.

## 14.4 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.

**TABLE 14-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR**

| 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 | — | — | ADFVR<1:0> | | 136 |

**Legend:** Shaded cells are unused by the temperature indicator module.


© 2015-2016 Microchip Technology Inc.

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

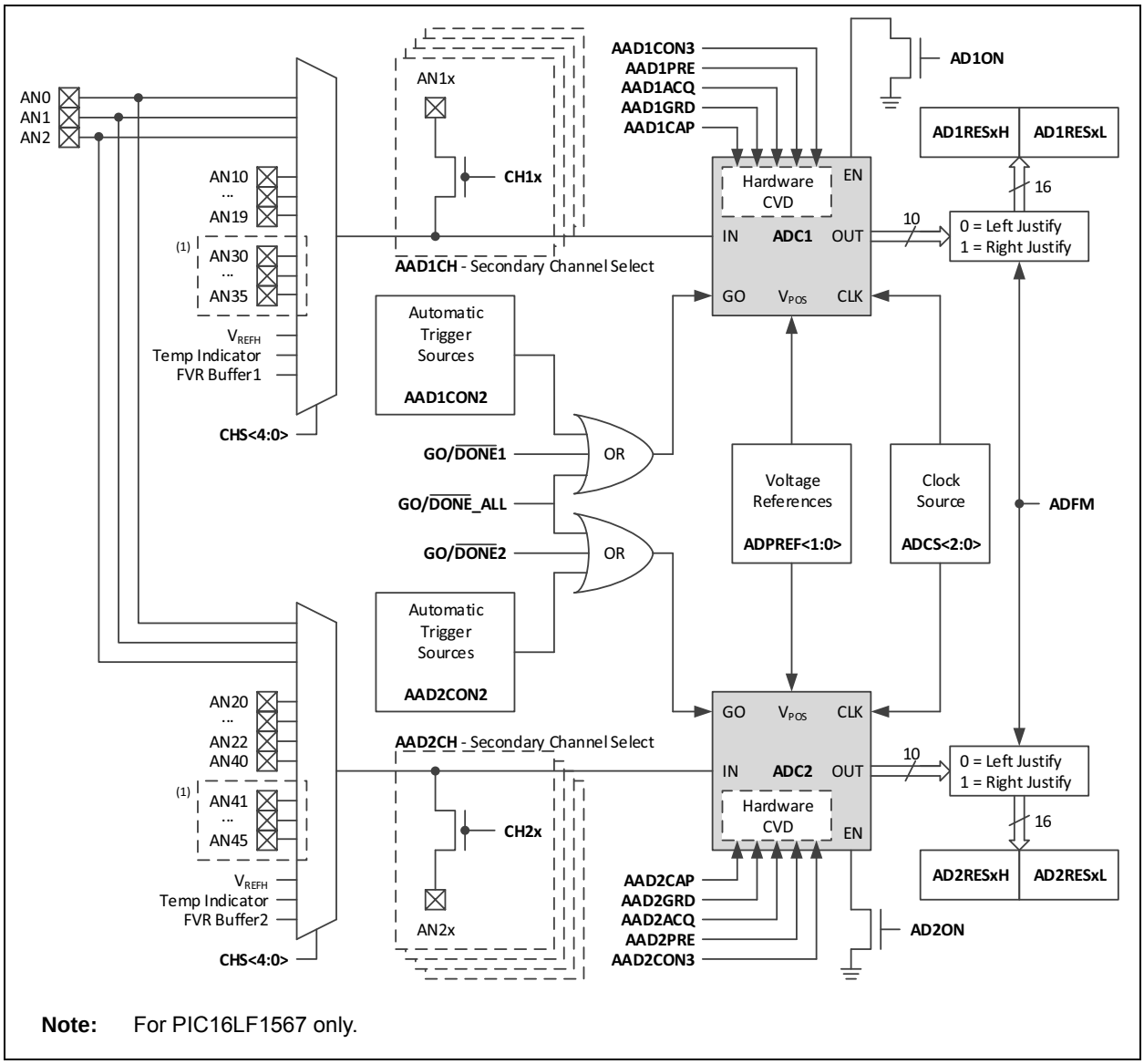
The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADxRESxH:ADxRESxL register pair).

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.

The PIC16LF1566/1567 has two ADCs, which can operate together or separately. Both ADCs can generate an interrupt upon completion of a conversion. This interrupt can be used to wake up the device from Sleep. Figure 15-1 shows the block diagram of the two ADCs.

**FIGURE 15-1: ADC SIMPLIFIED BLOCK DIAGRAM**




**Note:** For PIC16LF1567 only.

## 15.1 ADC Configuration

When configuring and using the ADC the following functions must be considered:

- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Result formatting

### 15.1.1 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 TRISx and ANSELx bits. Refer to **Section 11.0 "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. |

### 15.1.2 CHANNEL SELECTION

There are 24 channel selections available for PIC16LF1566 and 35 for PIC16LF1567. Three channels (AN0, AN1 and AN2) can be selected by both ADC1 and ADC2. The following channels can be selected by either of the ADCs:

- AN<2:0> pins
- Temperature Indicator
- FVR Buffer 1
- VREFH

The CHS bits of the ADxCON0 register determine which channel is connected to the sample and hold circuit of ADCx.

When changing channels, a delay (TACQ) is required before starting the next conversion. Refer to **Section 15.2.6 "Individual ADC Conversion Procedure"** for more information.

### 15.1.3 ADC VOLTAGE REFERENCE

The ADC module uses a positive and a negative voltage reference. The positive reference is labeled VREFH and the negative reference is labeled VREFL.

The positive voltage reference (VREFH) is selected by the ADPREF bits in the ADCON1 register. The positive voltage reference source can be:

- VREF+ pin
- VDD

The negative voltage reference (VREFL) source is:

- VSS

### 15.1.4 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:

- FOSC/2
- FOSC/4
- FOSC/8
- FOSC/16
- FOSC/32
- FOSC/64
- FRC (internal RC oscillator)

The time to complete one bit conversion is defined as TAD. One full 10-bit conversion requires 11.5 TAD periods as shown in Figure 15-2.

For correct conversion, the appropriate TAD specification must be met. Refer to the ADC conversion requirements in **Section 25.0 "Electrical Specifications"** for more information. Table 15-1 gives examples of appropriate ADC clock selections.

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

 © 2015-2016 Microchip Technology Inc.

**TABLE 15-1: ADC CLOCK PERIOD ($T_{AD}$) vs. DEVICE OPERATING FREQUENCIES(1)**

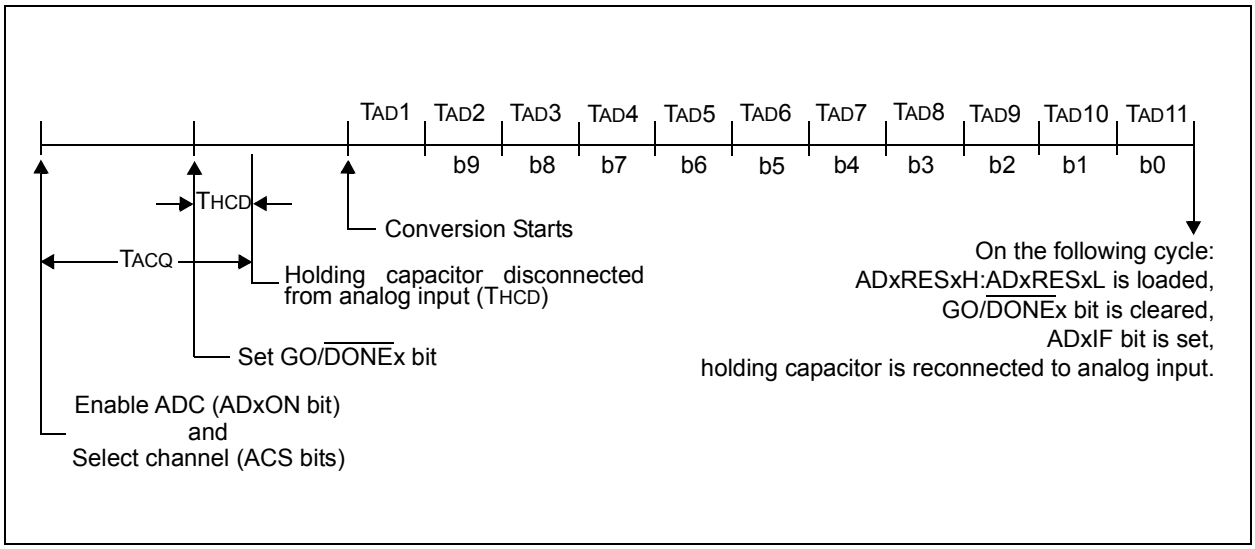
| ADC Clock Period ($T_{AD}$)(2) | | Device Frequency (Fosc) | | | | | |
|---|---|---|---|---|---|---|---|
| ADC Clock Source | ADCS<2:0> | 32 MHz | 20 MHz | 16 MHz | 8 MHz | 4 MHz | 1 MHz |
| Fosc/2 | 000 | 62.5 ns | 100 ns | 125 ns | 250 ns | 500 ns | 2.0 μs |
| Fosc/4 | 100 | 125 ns | 200 ns | 250 ns | 500 ns | 1.0 μs | 4.0 μs |
| Fosc/8 | 001 | 250 ns | 400 ns | 500 ns | 1.0 μs | 2.0 μs | 8.0 μs |
| Fosc/16 | 101 | 500 ns | 800 ns | 1.0 μs | 2.0 μs | 4.0 μs | 16.0 μs |
| Fosc/32 | 010 | 1.0 μs | 1.6 μs | 2.0 μs | 4.0 μs | 8.0 μs | 32.0 μs |
| Fosc/64 | 110 | 2.0 μs | 3.2 μs | 4.0 μs | 8.0 μs | 16.0 μs | 64.0 μs |
| FRC | x11 | 1.0-6.0 μs | 1.0-6.0 μs | 1.0-6.0 μs | 1.0-6.0 μs | 1.0-6.0 μs | 1.0-6.0 μs |

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

**Note 1:** The $T_{AD}$ period when using the FRC clock source can fall within a specified range, (see $T_{AD}$ parameter). The $T_{AD}$ period when using the $F_{OSC}$-based clock source can be configured for a more precise $T_{AD}$ period. However, the FRC clock source must be used when conversions are to be performed with the device in Sleep mode.

**2:** The 250 ns minimum $T_{AD}$ is only true for $V_{DD}$ > 2.4V.

**FIGURE 15-2: ANALOG-TO-DIGITAL CONVERSION $T_{AD}$ CYCLES**



### 15.1.5 INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion. The ADCx interrupt flag is the ADxIF bit in the PIRx register. The ADCx interrupt enable is the ADxIE bit in the PIEx register. The ADxIF bit must be cleared in software.

| | |
|---|---|
| **Note 1:** | The ADxIF bit is set at the completion of every conversion, regardless of whether or not the ADCx interrupt is enabled. |
| **2:** | The ADC operates during Sleep only when the FRC oscillator is selected. |

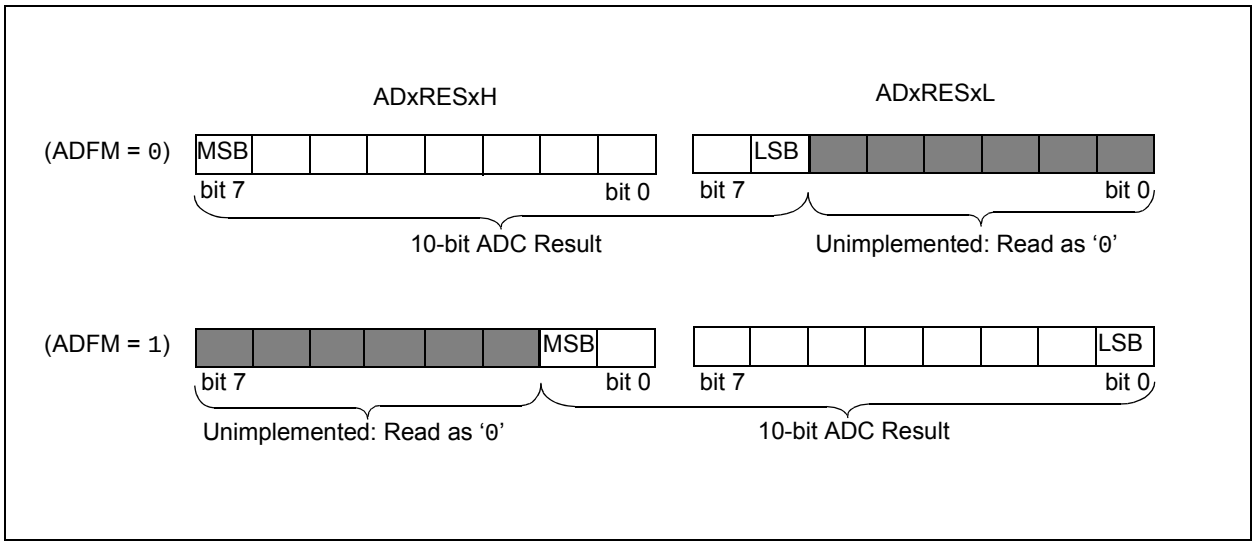
This interrupt can be generated while the device is operating or while in Sleep. If the device is in Sleep, the interrupt will wake-up the device. Upon waking from Sleep, the next instruction following the SLEEP instruction is always executed. If the user is attempting to wake-up from Sleep and resume 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.

### 15.1.6 RESULT FORMATTING

The 10-bit ADC conversion result can be supplied in two formats, left justified or right justified. The ADFM bit of the ADCON1/ADCOMCON register controls the output format.

Figure 15-3 shows the two output formats.

**FIGURE 15-3: 10-BIT ADC CONVERSION RESULT FORMAT**




 © 2015-2016 Microchip Technology Inc.

## 15.2 ADC Operation

### 15.2.1 STARTING A CONVERSION

To enable the ADCx module, the ADxON bit of the ADxCON0 register must be set to a '1'. Setting the GO/DONEx bit of the ADxCON0 register to a '1' will start the Analog-to-Digital Conversion.

Setting the GO/DONE_ALL bit of the ADCON1/ADCOMCON register to a '1' will start the Analog-to-Digital conversion for both ADC1 and ADC2, which is called synchronized conversion.

| | |
|---|---|
| **Note:** | The GO/DONEx bit should not be set in the same instruction that turns on the ADC. Refer to **Section 15.2.6 "Individual ADC Conversion Procedure"**. |

### 15.2.2 COMPLETION OF A CONVERSION

When the conversion is complete, the ADC module will:

- Clear the GO/DONEx bit
- Clear the GO/DONE_ALL bit if a synchronized conversion is done
- Set the ADxIF interrupt flag bit
- Update the ADxRESxH and ADxRESxL registers with new conversion result

| | |
|---|---|
| **Note:** | Only ADxRES0 will be updated after a single sample conversion. The completion of a double sample conversion will update both ADxRES0 and ADxRES1 registers. Refer to **Section 16.1.6 "Double Sample Conversion"** for more information. |

### 15.2.3 TERMINATING A CONVERSION

If a conversion must be terminated before completion, the GO/DONEx bit can be cleared in software. If the GO/DONE_ALL bit is cleared in software, the synchronized conversion will stop. The ADxRESxH and ADxRESxL registers will be updated with the partially complete Analog-to-Digital conversion sample. Incomplete bits will match the last bit converted.

| | |
|---|---|
| **Note:** | A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated. |

### 15.2.4 ADC OPERATION DURING SLEEP

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the FRC option. Performing the ADC conversion during Sleep can reduce system noise. 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 ADxON 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 ADxON bit remains set.

### 15.2.5 AUTO-CONVERSION TRIGGER

The auto-conversion trigger allows periodic ADC measurements without software intervention. When a rising edge of the selected source occurs, the GO/DONEx bit is set by hardware.

The auto-conversion trigger source is selected with the TRIGSEL<2:0> bits of the ADxCON2 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.

See Table 15-2 for auto-conversion sources.

**TABLE 15-2: AUTO-CONVERSION SOURCES**

| Source Peripheral | Trigger Event |
|---|---|
| Timer0 | Timer0 Overflow |
| Timer1 | Timer1 Overflow |
| Timer2 | Timer2 matches PR2 |
| Timer4 | Timer4 matches PR4 |
| ADTRIG pin | ADTRIG Rising Edge |
| ADTRIG pin | ADTRIG Falling Edge |

### 15.2.6 INDIVIDUAL ADC CONVERSION PROCEDURE

This is an example procedure for using the ADCx to perform an Analog-to-Digital conversion:

1. Configure Port:
   - Disable pin output driver (Refer to the TRISx register)
   - Configure pin as analog (Refer to the ANSELx register)
   - Disable weak pull-ups either globally (Refer to the OPTION_REG register) or individually (Refer to the appropriate WPUx register)
2. Configure the ADCx module:
   - Select ADCx conversion clock
   - Configure voltage reference
   - Select ADCx input channel
   - Turn on ADCx module
3. Configure ADCx interrupt (optional):
   - Clear ADCx interrupt flag
   - Enable ADCx interrupt
   - Enable peripheral interrupt
   - Enable global interrupt(1)
4. Wait the required acquisition time(2).
5. Start conversion by setting the GO/$\overline{\text{DONE}}$x bit.
6. Wait for ADCx conversion to complete by one of the following:
   - Polling the GO/$\overline{\text{DONE}}$x bit
   - Waiting for the ADCx interrupt (interrupts enabled)
7. Read ADCx Result.
8. Clear the ADCx 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 15.4 "ADC Acquisition Requirements"**.

**EXAMPLE 15-1: ADC CONVERSION**

```
;This code block configures the ADC1
;for polling, Vdd and Vss references, FRC
;oscillator and AN0 input.
;
;Conversion start and polling for completion
;are included.
;
BANKSEL   ADCON1       ;
MOVLW     B'11110000'  ;Right justify, FRC
                       ;oscillator
MOVWF     ADCON1       ;VDD is VREFH
BANKSEL   TRISA        ;
BSF       TRISA,0      ;Set RA0 to input
BANKSEL   ANSELA       ;
BSF       ANSELA,0     ;Set RA0 to analog
BANKSEL   WPUA
BCF       WPUA,0        ;Disable RA0 weak
                        pull-up
BANKSEL   ADCON0       ;
MOVLW     B'00000001'  ;Select channel AN0
MOVWF     ADCON0       ;Turn ADC On
MOVLW     .5
MOVWF     AAD1ACQ      ;Acquisiton delay
BSF       ADCON0,ADGO  ;Start conversion
BTFSC     ADCON0,ADGO  ;Is conversion done?
GOTO      $-1          ;No, test again
BANKSEL   AD1RES0H     ;
MOVF      AD1RES0H,W   ;Read upper 2 bits
MOVWF     RESULTHI     ;store in GPR space
BANKSEL   AD1RES0L     ;
MOVF      AD1RES0L,W   ;Read lower 8 bits
MOVWF     RESULTLO     ;Store in GPR space
```

 © 2015-2016 Microchip Technology Inc.

## 15.3 Register Definitions: ADC Control

### REGISTER 15-1: ADCON0(1)/AD1CON0(2): ANALOG-TO-DIGITAL (ADC) 1 CONTROL REGISTER 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 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| CHS15 | CHS14 | CHS13 | CHS12 | CHS11 | CHS10 | GO/$\overline{\text{DONE}}$1(4) | AD1ON |
| 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 **CHS15<5:0>:** Analog Channel Select bits for ADC1
- 111111 = Fixed Voltage Reference (FVREF) Buffer 1 Output
- 111110 = Reserved
- 111101 = Temperature Indicator
- 111100 = Reserved
- 111011 = VREFH (ADC Positive Reference)
- 100100 - 111010 = Reserved
- 011110 - 010111 = Channel 30 through 35, (AN30 through AN35)(3)
- 010100 - 011101 = Reserved
- 001010 - 010011 = Channel 10 through 19, (AN10 through AN19)
- 000011 - 001001 = Reserved
- 000010 = Channel 2, (AN2)
- 000001 = Channel 1, (AN1)
- 000000 = Channel 0, (AN0)

bit 1 **GO/$\overline{\text{DONE}}$1:** ADC1 Conversion Status bit (4)

If AD1ON = 1
- 1 = ADC conversion in progress. Setting this bit starts the ADC conversion. When the RC clock source is selected, the ADC module waits one instruction before starting the conversion.
- 0 = ADC conversion not in progress (This bit is automatically cleared by hardware when the ADC conversion is complete.)
  If this bit is cleared while a conversion is in progress, the conversion will stop and the results of the conversion up to this point will be transferred to the result registers, but the AD1IF interrupt flag bit will not be set.

If AD1ON = 0
- 0 = ADC conversion not in progress

bit 0 **AD1ON:** ADC Module 1 Enable bit
- 1 = ADC converter module 1 is operating
- 0 = ADC converter module 1 is shut off and consumes no operating current. All Analog channels are disconnected.

**Note 1:** Bank 1 name is ADCON0.
**2:** Bank 14 name is AD1CON0.
**3:** PIC16LF1567 only. Not implemented on PIC16LF1566.
**4:** When the AD1DSEN bit is set; the GO/$\overline{\text{DONE}}$1 bit will clear after a second conversion has completed.

**REGISTER 15-2: AD2CON0: ANALOG-TO-DIGITAL (ADC) 2 CONTROL REGISTER 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 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| CHS25 | CHS24 | CHS23 | CHS22 | CHS21 | CHS20 | GO/$\overline{\text{DONE}}$2(2) | AD2ON |
| 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 **CHS25<5:0>:** Analog Channel Select bits for ADC2
When AD2ON = 0, all multiplexer inputs are disconnected.
111111 = Fixed Voltage Reference (FVREF)
111101 = Temperature Indicator
111011 = VREFH (ADC Positive Reference)
101110 - 111010 = Reserved
101001 - 101101 = Channel 41 through 45, (AN41 through AN45)(1)
101000 = Channel 40, (AN40)
011110 - 100111 = Reserved
010100 - 011101 = Channel 20 through 29, (AN20 through AN29)
000011 - 010011 = Reserved
000010 = Channel 2, (AN2)
000001 = Channel 1, (AN1)
000000 = Channel 0, (AN0)

bit 1 **GO/$\overline{\text{DONE}}$2:** ADC2 Conversion Status bit(2)
If AD2ON = 1
1 = ADC conversion in progress. Setting this bit starts the ADC conversion. When the RC clock source is selected, the ADC Module waits one instruction before starting the conversion.
0 = ADC conversion not in progress (This bit is automatically cleared by hardware when the ADC conversion is complete.)
If this bit is cleared while a conversion is in progress, the conversion will stop and the results of the conversion up to this point will be transferred to the result registers, but the AD2IF interrupt flag bit will not be set.
If AD2ON = 0
0 = ADC conversion not in progress

bit 0 **AD2ON:** ADC Module 2 Enable bit
1 = ADC converter module 2 is operating
0 = ADC converter module 2 is shut off and consumes no operating current. All Analog channels are disconnected.

**Note 1:** PIC16LF1567 only. Not implemented on PIC16LF1566.
**2:** When the AD2DSEN bit is set; the GO/$\overline{\text{DONE}}$ bit will clear after a second conversion has completed.

 © 2015-2016 Microchip Technology Inc.

**REGISTER 15-3: ADCON1(1)/ADCOMCON(2): ADC 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 |
|---|---|---|---|---|---|---|---|
| ADFM | ADCS<2:0> | | | ADNREF | GO/$\overline{\text{DONE}}$_ALL | 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:** ADC Result Format Select bit
- 1 = Right justified. Six Most Significant bits of ADxRESxH are set to '0' when the conversion result is loaded.
- 0 = Left justified. Six Least Significant bits of ADxRESxL are set to '0' when the conversion result is loaded.

bit 6-4 **ADCS<2:0>:** ADC Conversion Clock Select bits
- 111 = FRC (clock supplied from an internal RC oscillator)
- 110 = FOSC/64
- 101 = FOSC/16
- 100 = FOSC/4
- 011 = FRC (clock supplied from an internal RC oscillator)
- 010 = FOSC/32
- 001 = FOSC/8
- 000 = FOSC/2

bit 3 **ADNREF:** ADC Negative Voltage Reference Configuration bit
- 1 = VREFL is connected to external VREF- pin(4)
- 0 = VREFL is connected to AVSS.

bit 2 **GO/$\overline{\text{DONE}}$_ALL(3):** Synchronized ADC Conversion Status bit
- 1 = Synchronized ADC conversion in progress. Setting this bit starts conversion in any ADC with ADxON = 1.
- 0 = Synchronized ADC conversion completed/ not in progress.

bit 1-0 **ADPREF<1:0>:** ADC Positive Voltage Reference Configuration bits
- 11 = VREFH is connected to internal Fixed Voltage Reference.
- 10 = VREFH is connected to external VREF+ pin(4)
- 01 = Reserved
- 00 = VREFH is connected to VDD

**Note 1:** Bank 1 name is ADCON1.

**2:** Bank 14 name is ADCOMCON.

**3:** Setting this bit triggers the GO/$\overline{\text{DONE}}$x bits in both ADCs. Each ADC will run a conversion according to its control register settings. This bit reads as an OR of the individual GO/$\overline{\text{DONE}}$x bits.

**4:** When selecting the VREF+ or VREF- pin as the source of the positive or negative reference, be aware that a minimum voltage specification exists. See **Section 25.0 "Electrical Specifications"** for details.

**REGISTER 15-4: ADxCON2: ADC CONTROL REGISTER 2**

| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | TRIGSEL<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 **Unimplemented:** Read as ‘0’

bit 6-4 **TRIGSEL<2:0>:** Auto-Conversion Trigger Selection bits
- 111 = ADTRIG Falling Edge
- 110 = ADTRIG Rising Edge
- 101 = TMR2 match to PR2(1)
- 100 = Timer1 Overflow(1)
- 011 = Timer0 Overflow(1)
- 010 = TMR4 match to PR4
- 001 = Reserved
- 000 = No Auto Conversion Trigger selected

bit 3-0 **Unimplemented:** Read as ‘0’

**Note 1:** Signal also sets its corresponding interrupt flag.

**REGISTER 15-5: ADxRESxH: ADC RESULT REGISTER HIGH (ADxRESxH) 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 |
|---|---|---|---|---|---|---|---|
| ADRES<9:2> | | | | | | | |
| 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 **ADRES<9:2>**: ADC Result Register bits
Upper eight bits of 10-bit conversion result

 © 2015-2016 Microchip Technology Inc.

**REGISTER 15-6: ADxRESxL: ADC RESULT REGISTER LOW (ADxRESxL) 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 |
|---|---|---|---|---|---|---|---|
| ADRES<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 **ADRES<1:0>**: ADC Result Register bits
Lower two bits of 10-bit conversion result

bit 5-0 **Reserved**: Do not use.

**REGISTER 15-7: ADxRESxH: ADC RESULT REGISTER HIGH (ADxRESxH) 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 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | ADRES<9:8> | |
| 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 7-2 **Reserved**: Do not use.

bit 1-0 **ADRES<9:8>**: ADC Result Register bits
Upper two bits of 10-bit conversion result

**REGISTER 15-8: ADxRESxL: ADC RESULT REGISTER LOW (ADxRESxL) 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 |
|---|---|---|---|---|---|---|---|
| ADRES<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 **ADRES<7:0>**: ADC Result Register bits
Lower eight bits of 10-bit conversion result

## 15.4 ADC 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 15-4. 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 15-4. **The maximum recommended impedance for analog sources is 10 kΩ.**

As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an ADC acquisition must be done before the conversion can be started. To calculate the minimum acquisition time, Equation 15-1 may be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

**EQUATION 15-1: ACQUISITION TIME EXAMPLE**

*Assumptions: Temperature = 50°C and external impedance of 10 kΩ 3.3V VDD*

$$\begin{aligned} TACQ &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= TAMP + TC + TCOFF \\ &= 2\,\mu s + TC + [(\text{Temperature} - 25°C)(0.05\,\mu s/°C)] \end{aligned}$$

*The value for TC can be approximated with the following equations:*

$$VAPPLIED\left(1 - \frac{1}{(2^{n+1}) - 1}\right) = VCHOLD \quad ;[1]\ VCHOLD \text{ charged to within 1/2 lsb}$$

$$VAPPLIED\left(1 - e^{\frac{-TC}{RC}}\right) = VCHOLD \quad ;[2]\ VCHOLD \text{ charge response to } VAPPLIED$$

$$VAPPLIED\left(1 - e^{\frac{-TC}{RC}}\right) = VAPPLIED\left(1 - \frac{1}{(2^{n+1}) - 1}\right) \quad ;\text{combining [1] and [2]}$$

*Note: Where n = number of bits of the ADC.*

*Solving for TC:*

$$\begin{aligned} TC &= -CHOLD(RIC + RSS + RS)\ ln(1/2047) \\ &= -15\ pF(1k\Omega + 7k\Omega + 10k\Omega)\ ln(0.0004885) \\ &= 2.06\ \mu s \end{aligned}$$

$$\begin{aligned} \text{Therefore: } TACQ &= 2\mu s + 2.06\mu s + [(50°C - 25°C)(0.05\mu s/°C)] \\ &= 5.31\ \mu s \end{aligned}$$

**Note 1:** The reference voltage (VRPOS) has no effect on the equation, since it cancels itself out.

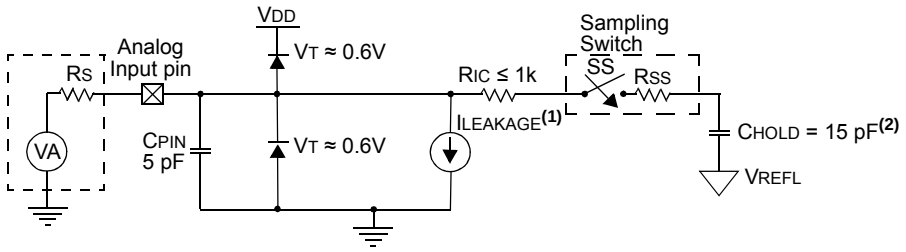
**2:** The charge holding capacitor (CHOLD) is not discharged after each conversion.

**3:** The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

**4:** The calculation above assumed CHOLD = 15pF. This value can be larger than 15pF by setting the AADxCAP register.

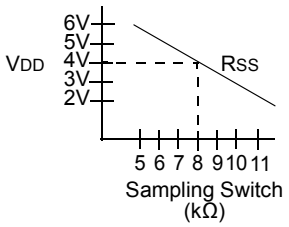
 © 2015-2016 Microchip Technology Inc.

**FIGURE 15-4: ANALOG INPUT MODEL**




**Legend:**

| | | |
|---|---|---|
| CHOLD | = | Sample/Hold Capacitance |
| CPIN | = | Input Capacitance |
| ILEAKAGE | = | Leakage Current at the pin due to varies injunctions |
| RIC | = | Interconnect Resistance |
| RSS | = | Resistance of Sampling switch |
| SS | = | Sampling Switch |
| VT | = | Threshold Voltage |




**Note 1:** Refer to **Section 25.0 "Electrical Specifications"**.

**2:** Additional CHOLD could be added by setting the ADDxCAP register.

**FIGURE 15-5: ADC TRANSFER FUNCTION**

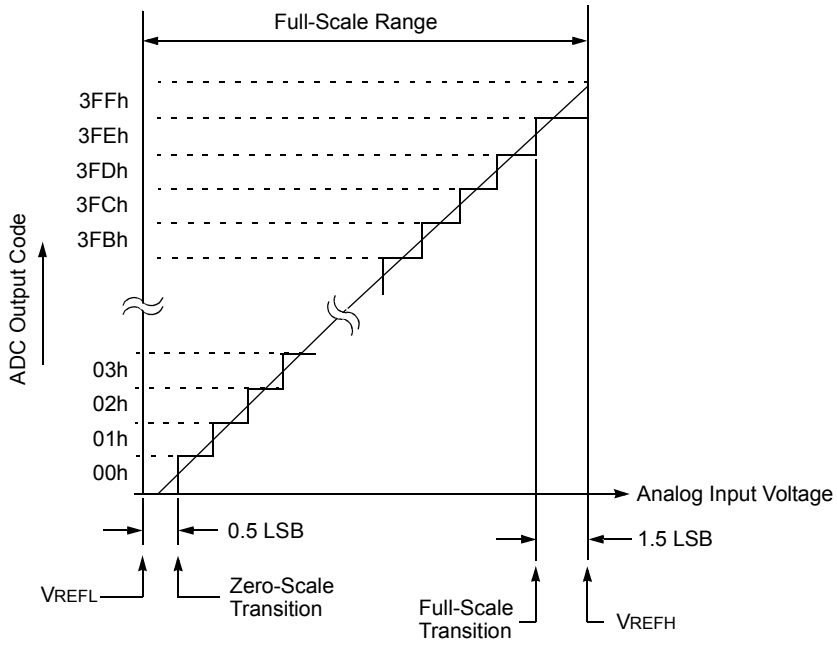


**TABLE 15-3: SUMMARY OF REGISTERS ASSOCIATED WITH ADC**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| ADCON0/ AD1CON0 | CHS15 | CHS14 | CHS13 | CHS12 | CHS11 | CHS10 | GO/$\overline{\text{DONE}}$1 | AD1ON | 145 |
| AD2CON0 | CHS25 | CHS24 | CHS23 | CHS22 | CHS21 | CHS20 | GO/$\overline{\text{DONE}}$2 | AD2ON | 146 |
| ADCON1/ ADCOMCON | ADFM | ADCS<2:0> | | | ADNREF | GO/$\overline{\text{DONE}}$_ALL | ADPREF<1:0> | | 147 |
| ADxCON2 | — | TRIGSEL<2:0> | | | — | — | — | — | 148 |
| ADxRESxH | ADC Result Register High | | | | | | | | 148, 149 |
| ADxRESxL | ADC Result Register Low | | | | | | | | 149, 149 |
| ANSELA | ANSA7 | ANSA6 | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 116 |
| ANSELB | ANSB7 | ANSB6 | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 120 |
| ANSELC | ANSC7 | ANSC6 | ANSC5 | ANSC4 | ANSC3 | ANSC2 | ANSC1 | ANSC0 | 124 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 115 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 123 |
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | — | — | ADFVR<1:0> | | 136 |

**Legend:** x = unknown, u = unchanged, – = unimplemented read as '0', q = value depends on condition. Shaded cells are not used for ADC module.

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


© 2015-2016 Microchip Technology Inc.

## 16.0 HARDWARE CAPACITIVE VOLTAGE DIVIDER (CVD) MODULE

The hardware Capacitive Voltage Divider (CVD) module is a peripheral, which allows the user to perform a relative capacitance measurement on any ADC channel using the internal ADC sample and hold capacitance as a reference. This relative capacitance measurement can be used to implement capacitive touch or proximity sensing applications.

The CVD operation begins with the ADC's internal sample and hold capacitor ($C_{HOLD}$) being disconnected from the path which connects it to the external capacitive sensor node. While disconnected, $C_{HOLD}$ is precharged to $V_{DD}$ or $V_{SS}$, while the path to the sensor node is also discharged to $V_{DD}$ or $V_{SS}$. Typically, this node is discharged to the level opposite that of $C_{HOLD}$. When the precharge phase is complete, the $V_{DD}/V_{SS}$ bias paths for the two nodes are shut off and $C_{HOLD}$ and the path to the external sensor node are re-connected, at which time the acquisition phase of the CVD operation begins. During acquisition, a capacitive voltage divider is formed between the precharged $C_{HOLD}$ the and sensor nodes, which results in a final voltage level settling on $C_{HOLD}$, which is determined by the capacitances and precharge levels of the two nodes involved. After acquisition, the ADC converts the voltage level held on $C_{HOLD}$. This process is then usually repeated with the selected precharge levels for both the $C_{HOLD}$ and the inverted sensor nodes. Figure 16-1 shows the waveform for two inverted CVD measurements, which is also known is differential CVD measurement.

In a typical application, an Analog-to-Digital Converter (ADC) channel is attached to a pad on a Printed Circuit Board (PCB), which is electrically isolated from the end user. A capacitive change is detected on the ADC channel using the CVD conversion method when the end user places a finger over the PCB pad, the developer then can implement software to detect a touch or proximity event. Key features of this module include:

- Automated double sample conversions
- Two sets of result registers
- Inversion of second sample
- 7-bit precharge timer
- 7-bit acquisition timer
- Two guard ring output drives
- Adjustable sample and hold capacitor array
- Simultaneous CVD sampling on two ADCs

**Note:** For more information on capacitive voltage divider sensing method refer to the Application Note AN1478, *"mTouch® Sensing Solution Acquisition Methods Capacitive Voltage Divider"* (DS01478).

**FIGURE 16-1: DIFFERENTIAL CVD MEASUREMENT WAVEFORM**

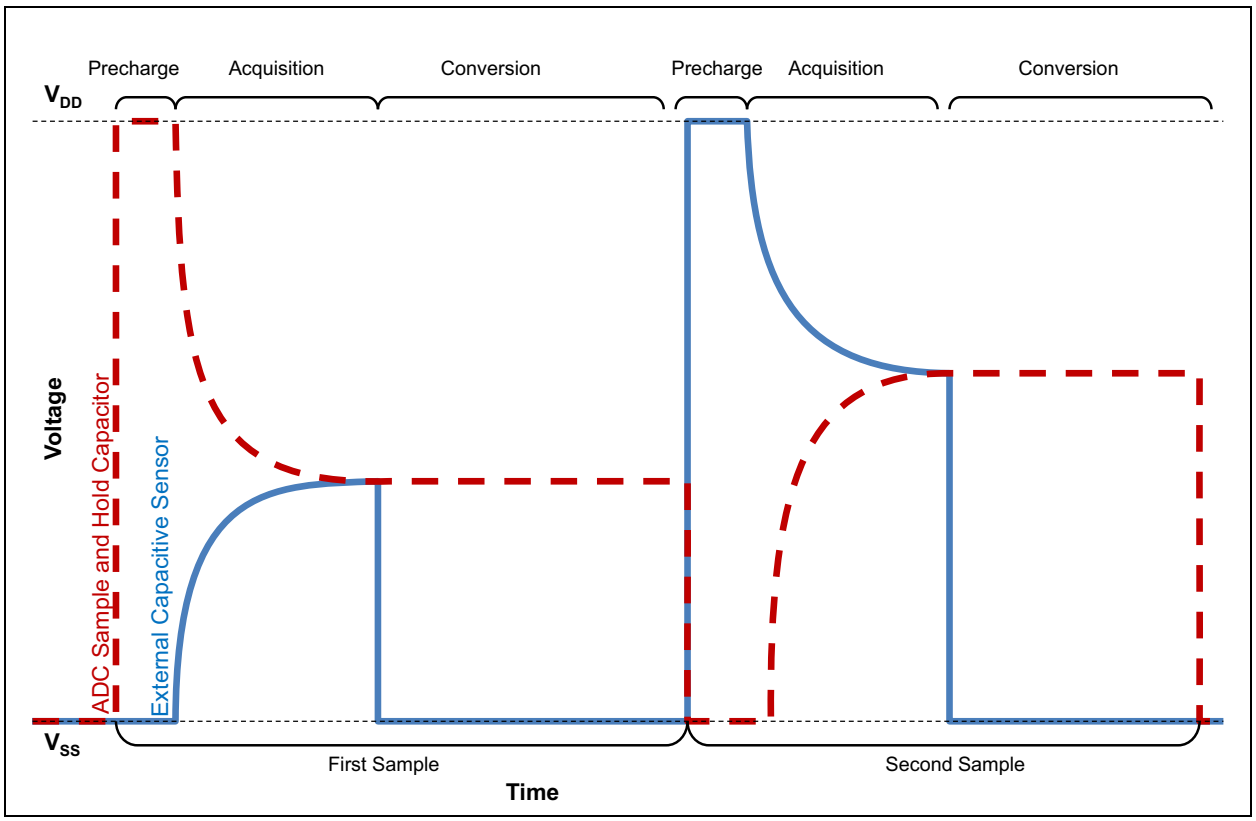
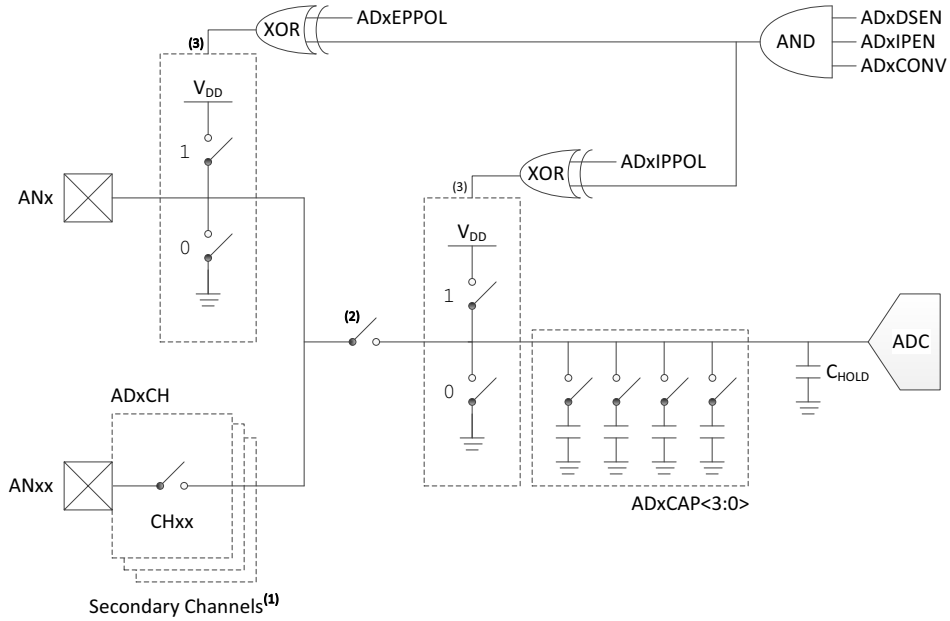



**FIGURE 16-2: HARDWARE CAPACITIVE VOLTAGE DIVIDER BLOCK DIAGRAM**




**Note 1:** Output drivers are disabled on any enabled secondary channel.
**2:** Disconnected for precharge and conversion stages.
**3:** Only enabled during precharge stage.

 © 2015-2016 Microchip Technology Inc.

## 16.1 Hardware CVD Operation

Capacitive Voltage Divider is a charge averaging capacitive sensing method. The hardware CVD module will automate the process of charging, averaging between the external sensor and the internal ADC sample and hold capacitor, and then initiating the ADC conversions. The whole process can be expanded into three stages: precharge, acquisition, and conversion. See Figure 16-5 for basic information on the timing of three stages.

### 16.1.1 PRE-CHARGE TIMER

The precharge stage is an optional 1-127 instruction/TAD cycle time delay used to put the external ADC channel and the internal sample and hold capacitor (CHOLD) into pre-conditioned states. The precharge stage of conversion is enabled by writing a non-zero value to the ADxPRE<6:0> bits of the AADxPRE register. This stage is initiated when a conversion sequence is started by either the GO/$\overline{\text{DONE}}$x, GO/$\overline{\text{DONE}}$_ALL bit or a Special Event Trigger. When initiating an ADC conversion, if the ADxPRE bits are cleared, this stage is skipped.

During the precharge time, CHOLD is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either VDD or VSS, depending on the value of the ADxEPPOL bit of the AADxCON3 register. At the same time, the port pin logic of the selected analog channel is overridden to drive a digital high or low out, in order to precharge the outer portion of the ADC's sample path, which includes the external sensor. The output polarity of this override is determined by the ADxEPPOL bit of the AADxCON3 register.

Even though the analog channel of the pin is selected, the analog multiplexer is forced open during the precharge stage. The ADC multiplex or logic is overridden and disabled only during the precharge time.

### 16.1.2 ACQUISITION TIMER

The acquisition timer controls the time allowed to acquire the signal to be sampled. The acquisition delay time is from 1 to 127 instruction/TAD cycles and is used to allow the voltage on the internal sample and hold capacitor (CHOLD) to settle to a final value through charge averaging. The acquisition time of conversion is enabled by writing a non-zero value to the AADxACQ<6:0> bits of the AADxACQ register. When the acquisition time is enabled, the time starts immediately following the precharge stage. If the ADxPRE<6:0> bits of the AADxPRE register are set to zero, the acquisition time is initiated by either setting the GO/$\overline{\text{DONE}}$x, GO/$\overline{\text{DONE}}$_ALL bit or a Special Event Trigger.

At the start of the acquisition stage, the port pin logic of the selected analog channel is again overridden to turn off the digital high/low output drivers so that they do not affect the final result of charge averaging. Also, the selected ADC channel is connected to CHOLD. This allows charge averaging to proceed between the precharged channel and the CHOLD capacitor.

### 16.1.3 STARTING A CONVERSION

To enable the ADC module, the ADxCON bit of the AADxCON0 register must be set. Setting the GO/$\overline{\text{DONE}}$x, GO/$\overline{\text{DONE}}$_ALL or by the Special Event Trigger inputs will start the Analog-to-Digital conversion.

Once a conversion begins, it proceeds until complete, while the ADxON bit is set. If the ADxON bit is cleared, the conversion is halted. The GO/$\overline{\text{DONE}}$x bit of the AADxCON0 register indicates that a conversion is occurring, regardless of the starting trigger.

| | |
|---|---|
| **Note:** | The GO/$\overline{\text{DONE}}$x bit should not be set in the same instruction that turns on the ADC. Refer to Section **Section 16.1.12 "Hardware CVD Double Conversion Procedure"** |

### 16.1.4 COMPLETION OF A CONVERSION

When the conversion is complete, the ADC module will:

- Clear the GO/$\overline{\text{DONE}}$x bit of the AADxCON0 register or clear the GO/$\overline{\text{DONE}}$_ALL bit of the ADCON1 register if synchronized conversion is used.
- Set the ADxIF interrupt flag bit of the PIRx register.
- Update the AADxRESxH and AADxRESxL registers with new conversion results.

### 16.1.5 TERMINATING A CONVERSION

If a conversion must be terminated before completion, clear the GO/DONEx bit. The AADxRESxH and AADxRESxL registers will be updated with the partially complete Analog-to-Digital conversion sample. Incomplete bits will match the last bit converted.

The ADSTAT register can be used to track the status of the hardware CVD module during a conversion.

| **Note:** | A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated. |
|---|---|

### 16.1.6 DOUBLE SAMPLE CONVERSION

Double sampling can be enabled by setting the AADxSEN bit of the AADxCON3 register. When this bit is set, two conversions are completed each time the GO/DONEx, GO/DONE_ALL bit is set or a Special Event Trigger occurs. The GO/DONEx or GO/DONE_ALL bits remain set for the duration of both conversions and is used to signal the end of the conversion.

Without setting the ADxIPEN bit, the double conversion will have identical charge/discharge on the internal and external capacitor for these two conversions. Setting the ADxIPEN bit prior to a double conversion will allow the user to perform a pseudo-differential CVD measurement by subtracting the results from the double conversion. This is highly recommended for noise immunity purposes.

The result of the first conversion is written to the AADxRES0H and AADxRES0L registers. The second conversion starts two clock cycles after the first has completed, while the GO/DONEx and GO/DONE_ALL bits remain set. When the ADxIPEN bit of AADxCON3 is set, the value used by the ADC for the ADxEPPOL, ADxIPPOL and GRDxPOL bits are inverted. The value stored in those bit locations is unchanged. All other control signals remain unchanged from the first conversion. The result of the second conversion is stored in the AADxRES1H and AADxRES1L registers. See Figure 16-4 and Figure 16-5 for more information.

### 16.1.7 GUARD RING OUTPUTS

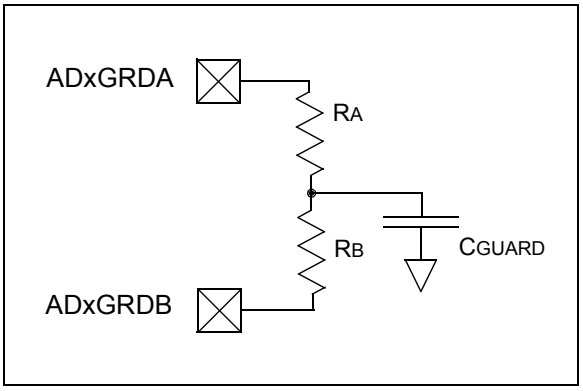
The guard ring outputs consist of a pair of digital outputs from the hardware CVD module. Each ADC has its own pair of guard ring outputs. This function is enabled by the GRDxAOE and GRDxBOE bits of the AADxGRD register. Polarity of the output is controlled by the GRDxPOL bit.

Once enabled and while ADxON = 1, the guard ring outputs of the ADC are active at all times. The outputs are initialized at the start of the precharge stage to match the polarity of the GRDxPOL bit. The guard output signal changes polarity at the start of the acquisition phase. The value stored by the GRDPOL bit does not change. When in Double Sampling mode, the ring output levels are inverted during the second precharge and acquisition phases if ADDxSEN = 1 and ADxIPEN = 1. For more information on the timing of the guard ring output, refer to Figure 16-4 and Figure 16-5.

A typical guard ring circuit is displayed in Figure 16-2. $C_{GUARD}$ represents the capacitance of the guard ring trace placed on a PCB board. The user selects values for RA and RB that will create a voltage profile on $C_{GUARD}$, which will match the selected channel during acquisition.

The purpose of the guard ring is to generate a signal in phase with the CVD sensing signal to minimize the effects of the parasitic capacitance on sensing electrodes. It also can be used as a mutual drive for mutual capacitive sensing. For more information about active guard and mutual drive, see Application Note AN1478, *"mTouch® Sensing Solution Acquisition Methods Capacitive Voltage Divider"* (DS01478).

**FIGURE 16-3: GUARD RING CIRCUIT**




### 16.1.8 MUTUAL TX OUTPUTS

This hardware CVD module has the ability to digitally drive a pulse synchronous to the CVD's waveform. This allows for the measurement of AC coupling between the transmit electrode, TX, and a capacitive sensor, RX, called 'mutual capacitance'. When the mutual capacitance between TX and RX increases, the TX pulse will create a larger voltage change on the RX sensor.

 © 2015-2016 Microchip Technology Inc.

Each ADC can enable the TX output on any or all of its associated analog channels using the ADxTX0 and ADxTX1 registers. The shared analog channels have TX enable bits in the ADCTX register. Once enabled and while ADxON = 1, the TX outputs of the ADC are active at all times except if the ADC is currently selecting the channel for conversion with the CHS bits of ADxCON0.

Polarity of the output is controlled by the TXxPOL bit of the AADxGRD register. The outputs are initialized at the start of the precharge stage to match the polarity of the TXxPOL bit. The TX output signal changes polarity immediately after the start of the acquisition phase. The value stored by TXxPOL does not change. When in double sampling mode (ADxDSEN = 1), the TX output changes polarity during the second precharge and acquisition phases if inversion is enabled (ADxIPEN = 1). For more information about the timing of the TX output, refer to Figure 16-4.

The typical mutual TX trace does not have a series resistor. If radiated emissions are a concern, a series resistor can be used to increase the rise-time at the cost of reduced noise dissipation.

To perform a combined self- and mutual-capacitance measurement, set ADxEPPOL and ADxIPPOL to opposite polarities, and set TXxPOL = ADxEPPOL.

To perform a mutual-only capacitance measurement, set ADxEEPOL and ADxIPPOL to the same polarity, and set TXxPOL = ADxEPPOL.

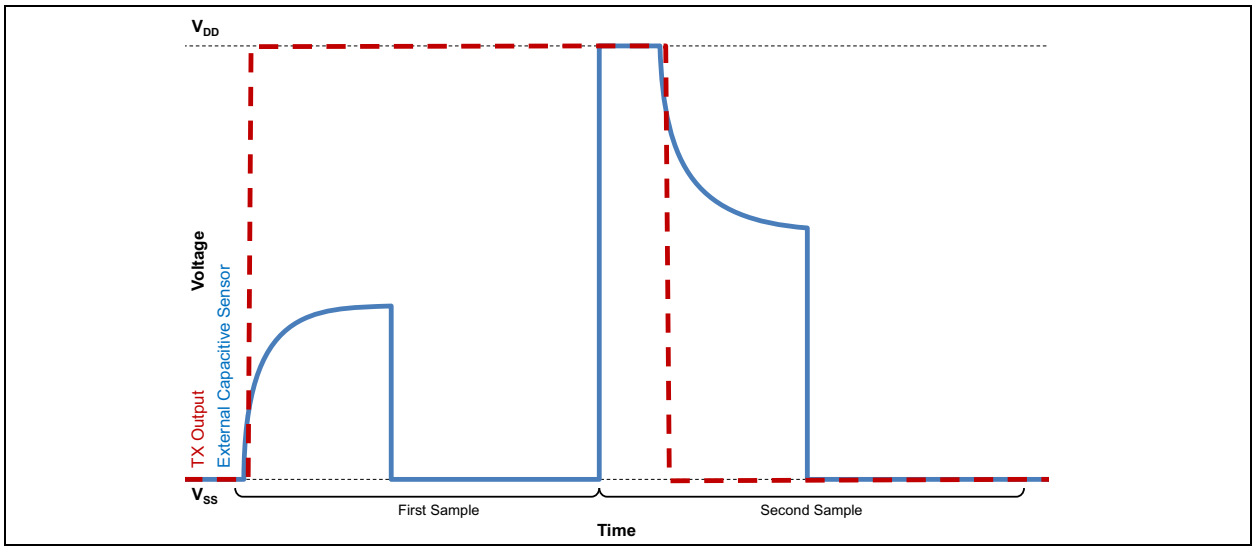
### 16.1.9 COMPARISON OF GUARDING AND MUTUAL CAPACITANCE

The mutual TX drivers are driven the same way as the ADxGRDA output.

Both guard and mutual drivers provide a low impedance path for noise to redirect away from the sensor to improve robustness. Mutual drivers are lower impedance due to the absence of the external voltage divider resistance.

The goal of the guard is to minimize coupling between the sensor (RX) and the environment to improve sensitivity, while the goal of the mutual TX driver is to maximize the change in coupling when the event occurs.

**FIGURE 16-4: DIFFERENTIAL CVD WITH GUARD RING OUTPUT WAVEFORM**




### 16.1.10 ADDITIONAL SAMPLE AND HOLD CAPACITOR

Additional capacitance can be added in parallel with the sample and hold capacitor ($C_{HOLD}$) by setting the ADDxCAP<3:0> bits of the AADxCAP register. This bit connects a digitally programmable capacitance to the ADC conversion bus, increasing the effective internal capacitance of the sample and hold capacitor in the ADC module. Each ADC has its own additional capacitance array. This is used to improve the match between internal and external capacitance for a better sensing performance. The additional capacitance does not affect analog performance of the ADC because it is not connected during conversion. See Figure 16-1.

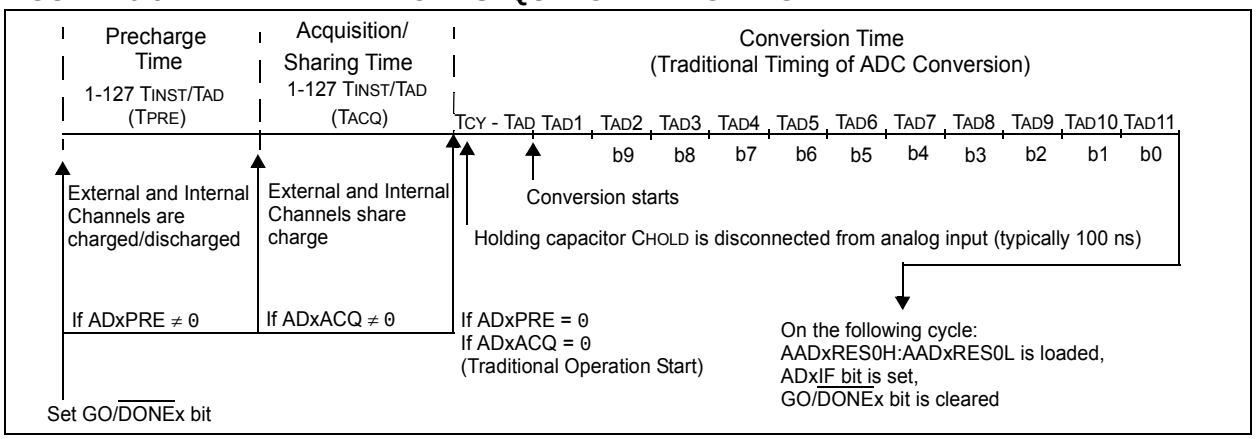
### 16.1.11 SECONDARY CHANNEL

Each ADC has one primary channel selected by CHx<4:0> bits of the AADxCON0 register. Multiple secondary channels can be connected to the ADC conversion bus by setting the bits in the AADxCH register. This allows a combined CVD scan on multiple ADC channels, which is beneficial for low-power and proximity capacitive sensing.

Each secondary channel is forced to input. The ANSELx bit for secondary channel is still under user control. During the precharge stage, the output drivers on each secondary channel will be overridden by the hardware CVD module and do exactly what the output drivers on the ADC's primary channel are configured to do.

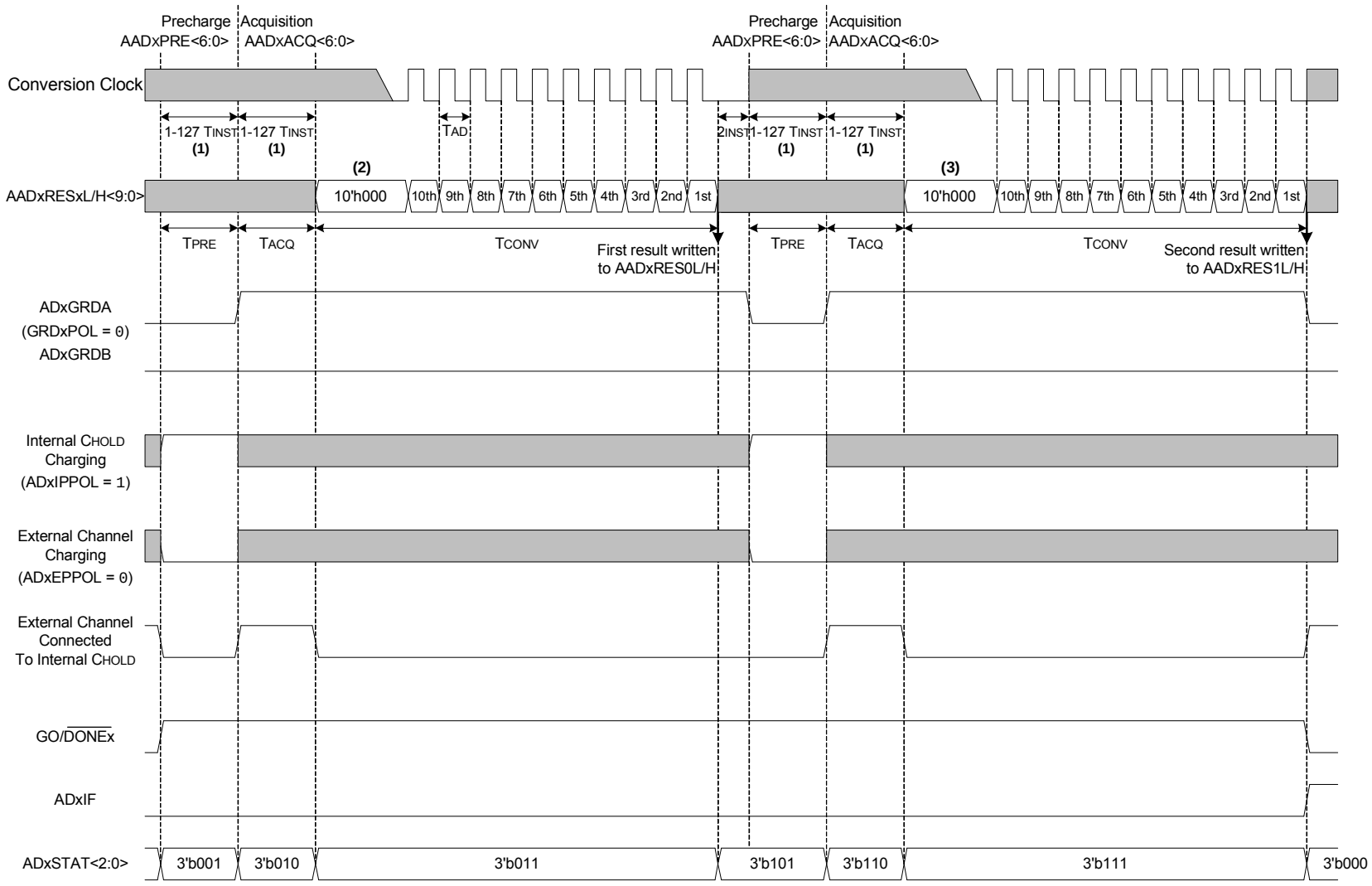
Both the primary and secondary channels are connected to the ADC as soon as the channels are selected by the CHx<4:0> bits of the AADxCON0 register and the bits in the AADxCH register.

**FIGURE 16-5: HARDWARE CVD SEQUENCE TIMING DIAGRAM**




 © 2015-2016 Microchip Technology Inc.

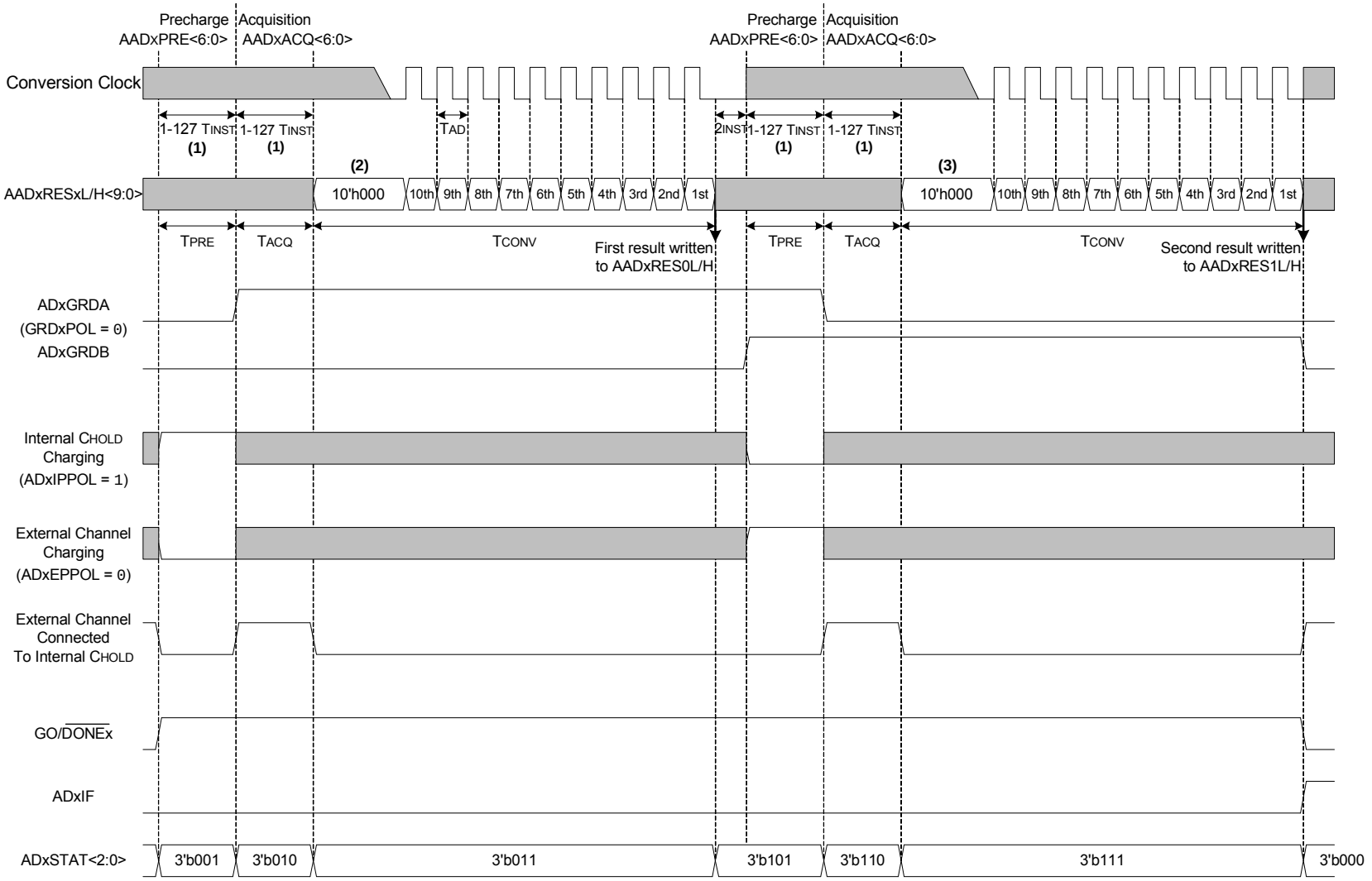
**FIGURE 16-6: DOUBLE SAMPLE CONVERSION SEQUENCE (ADDSEN = 1 AND ADIPEN = 0)**




**Note 1:** When the conversion clock is ADCRC, the precharge and acquisition timers are clocked by ADCRC.
**2:** The AADxRES0L/H registers are set to zero during this period.
**3:** The AADxRES1L/H registers are set to zero during this period.

© 2015-2016 Microchip Technology Inc.

**FIGURE 16-7: DOUBLE SAMPLE CONVERSION SEQUENCE (ADDSEN = 1 AND ADIPEN = 1)**




**Note 1:** When the conversion clock is ADCRC, the precharge and acquisition timers are clocked by ADCRC.
**2:** The AADxRES0L/H registers are set to zero during this period.
**3:** The AADxRES1L/H registers are set to zero during this period.


Preliminary
© 2015-2016 Microchip Technology Inc.

### 16.1.12 HARDWARE CVD DOUBLE CONVERSION PROCEDURE

This is an example procedure for using hardware CVD to perform a double conversion for differential CVD measurement with active guard drive.

1. Configure Port:
   - Enable pin output driver (Refer to the TRISx register).
   - Configure pin output low (Refer to the LATx register).
   - Disable weak pull-up (Refer to the WPUx register).
2. Configure the ADC module:
   - Select an appropriate ADC conversion clock for your oscillator frequency.
   - Configure voltage reference.
   - Select ADC input channel.
   - Turn on the ADC module.
3. Configure the hardware CVD module:
   - Configure charge polarity and double conversion.
   - Configure precharge and acquisition timer.
   - Configure guard ring (optional).
   - Select additional capacitance (optional).
4. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable peripheral interrupt
   - Enable global interrupt**(1)**
5. Start conversion by setting the GO/$\overline{\text{DONE}}$x, GO/$\overline{\text{DONE}}$_ALL bit or by enabling the Special Event Trigger in the ADDxCON2 register.
6. Wait for the ADC conversion to complete by one of the following:
   - Polling the GO/$\overline{\text{DONE}}$x or GO/$\overline{\text{DONE}}$_ALL bit.
   - Waiting for the ADC interrupt (interrupts enabled).
7. Read ADC result:
   - Conversion 1 result in ADDxRES0H and ADDxRES0L
   - Conversion 2 result in ADDxRES1H and ADDxRES1L
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.

**EXAMPLE 16-1: HARDWARE CVD DOUBLE CONVERSION**

```
;This code block configures the ADC
;for polling, VDD and VSS references, Fosc/16
;clock and AN0 input.
;
;The Hardware CVD1 will perform an inverted
;double conversion, Guard A and B drive are
;both enabled.
;Conversion start & polling for completion
 are included.
;
BANKSEL    TRISA
BCF        TRISA,0      ;Set RA0 to output
BANKSEL    LATA
BCF        LATA,0       ;RA0 output low
BANKSEL    ANSELA
BCF        ANSELA,0     ;Set RA0 to digital
BANKSEL    WPUA
BCF        WPUA,0       ;Disable pull-up on
RA0
;Initialize ADC and Hardware CVD

BANKSEL    AAD1CON0
MOVLW      B'00000001'  ;Select channel AN0
MOVWF      AAD1CON0
BANKSEL    AADCON1
MOVLW      B'11010000'  ;VDD and VSS VREF
MOVWF      AADCON1
MOVLW      B'00000000'  ;No secondary channel
MOVWF      AAD1CH

BANKSEL    AAD1CON3
MOVLW      B'01000011'  ;Double and inverted
MOVWF      AAD1CON3     ;
BANKSEL    AAD1PRE
MOVLW      .10
MOVWF      AAD1PRE      ;Pre-charge Timer
BANKSEL    AAD1ACQ
MOVLW      .10
MOVWF      AAD1ACQ      ;Acquisition Timer
BANKSEL    AAD1GRD
MOVLW      B'11000000'  ;Guard on A and B
MOVWF      AAD1GRD
BANKSEL    AAD1CAP
MOVLW      B'00000000'
MOVWF      AAD1CAP      ;No additional
                        ;Capacitance
BANKSEL    AD1CON0
BSF        AD1CON0, GO
BTFSC      AD1CON0, GO
GOTO       $-1          ;No, test again

;RESULTS OF CONVERIONS 1.
BANKSEL    AAD1RES0H    ;
MOVF       AAD1RES0H,W  ;Read upper 2 bits
MOVWF      RESULT0H     ;Store in GPR space
MOVF       AAD1RES0L,W  ;Read lower 8 bits
MOVWF      RESULT0L     ;Store in GPR space

;RESULTS OF CONVERIONS 2.
BANKSEL    AAD1RES1H    ;
MOVF       AAD1RES1H,W  ;Read upper 2 bits
MOVWF      RESULT1H     ;Store in GPR space
MOVF       AAD1RES1L,W  ;Read lower 8 bits
MOVWF      RESULT1L     ;Store in GPR space
```

### 16.1.13 HARDWARE CVD REGISTER MAPPING

The hardware CVD module is an enhanced expansion of the standard ADC module as stated in **Section 15.0 "Analog-to-Digital Converter (ADC) Module"** and is backward compatible with the other devices in this family. Control of the standard ADC1 module uses Bank 1 registers, see Table 16-1. This set of registers is mapped into Bank 14 with the control registers for the hardware CVD module. Although this subset of registers has different names, they are identical. Since the registers for the standard ADC are mapped into the Bank 14 address space, any changes to registers in Bank 1 will be reflected in Bank 14 and vice-versa.

**TABLE 16-1: HARDWARE CVD REGISTER MAPPING**

| [Bank 14 Address] | [Bank 1 Address] |
|---|---|
| **Hardware CVD** | **ADC** |
| [711h] AD1CON0(1) | [09Dh] ADCON0(1) |
| [712h] AD1CON1(1) | [09Eh] ADCON1(1) |
| [713h] AD1CON2(1) | [09Fh] ADCON2(1) |
| [714h] AD1CON3 | |
| [715h] ADSTAT | |
| [716h] AD1PRECON | |
| [717h] AAD1ACQ | |
| [718h] AD1GRD | |
| [719h] AD1CAPCON | |
| [71Ah] AAD1RES0L(1) | [09Bh] AD1RES0L(1) |
| [71Bh] AAD1RES0H(1) | [09Ch] AD1RES0H(1) |
| [71Ch] AAD1RES1L | |
| [71Dh] AAD1RES1H | |
| [71Eh] AD1CH | |

**Note 1:** Register is mapped in Bank 1 and Bank 14, using different names in each bank.

The ADC2 only has one set of registers in Bank 15. However, letter 'A', which stands for advanced, is added to the beginning of each register's name for legacy ADC control in this chapter. For example, AD2CON0 in **Section 15.0 "Analog-to-Digital Converter (ADC) Module"** uses the name of AAD2CON0 in this chapter. Please note that this is just an alias name, they still represent the same SFR register address in memory.

 © 2015-2016 Microchip Technology Inc.

## 16.2 Register Definitions: Hardware CVD Control

**REGISTER 16-1: ADCON0(1)/AD1CON0(2): ANALOG-TO-DIGITAL (ADC) 1 CONTROL REGISTER 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 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| CHS15 | CHS14 | CHS13 | CHS12 | CHS11 | CHS10 | GO/$\overline{\text{DONE}}$1(4) | AD1ON |
| 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 **CHS15<5:0>:** Analog Channel Select bits for ADC1
- 111111 = Fixed Voltage Reference (FVREF) Buffer 1 Output
- 111110 = Reserved
- 111101 = Temperature Indicator
- 111100 = Reserved
- 111011 = VREFH (ADC Positive Reference)
- 100100 - 111010 = Reserved
- 011110 - 010111 = Channel 30 through 35, (AN30 through AN35)(3)
- 010100 - 011101 = Reserved
- 001010 - 010011 = Channel 10 through 19, (AN10 through AN19)
- 000011 - 001001 = Reserved
- 000010 = Channel 2, (AN2)
- 000001 = Channel 1, (AN1)
- 000000 = Channel 0, (AN0)

bit 1 **GO/$\overline{\text{DONE}}$1:** ADC1 Conversion Status bit (4)

If AD1ON = 1
- 1 = ADC conversion in progress. Setting this bit starts the ADC conversion. When the RC clock source is selected, the ADC module waits one instruction before starting the conversion.
- 0 = ADC conversion not in progress (This bit is automatically cleared by hardware when the ADC conversion is complete.)
  If this bit is cleared while a conversion is in progress, the conversion will stop and the results of the conversion up to this point will be transferred to the result registers, but the AD1IF interrupt flag bit will not be set.

If AD1ON = 0
- 0 = ADC conversion not in progress

bit 0 **AD1ON:** ADC Module 1 Enable bit
- 1 = ADC converter module 1 is operating
- 0 = ADC converter module 1 is shut off and consumes no operating current. All Analog channels are disconnected.

**Note 1:** Bank 1 name is ADCON0.
**2:** Bank 14 name is AD1CON0.
**3:** PIC16LF1567 only. Not implemented on PIC16LF1566.
**4:** When the AD1DSEN bit is set; the GO/$\overline{\text{DONE}}$1 bit will clear after a second conversion has completed.

**REGISTER 16-2: AD2CON0: ANALOG-TO-DIGITAL (ADC) 2 CONTROL REGISTER 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 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| CHS25 | CHS24 | CHS23 | CHS22 | CHS21 | CHS20 | GO/$\overline{\text{DONE2}}$(2) | AD2ON |
| 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 **CHS25<5:0>:** Analog Channel Select bits for ADC2
When AD2ON = 0, all multiplexer inputs are disconnected.
111111 = Fixed Voltage Reference (FVREF)
111101 = Temperature Indicator
111011 = VREFH (ADC Positive Reference)
101110 - 111010 = Reserved
101001 - 101101 = Channel 41 through 45, (AN41 through AN45)(1)
101000 = Channel 40, (AN40)
011110 - 100111 = Reserved
010100 - 011101 = Channel 20 through 29, (AN20 through AN29)
000011 - 010011 = Reserved
000010 = Channel 2, (AN2)
000001 = Channel 1, (AN1)
000000 = Channel 0, (AN0)

bit 1 **GO/$\overline{\text{DONE2}}$:** ADC2 Conversion Status bit(2)
If AD2ON = 1
1 = ADC conversion in progress. Setting this bit starts the ADC conversion. When the RC clock source is selected, the ADC Module waits one instruction before starting the conversion.
0 = ADC conversion not in progress (This bit is automatically cleared by hardware when the ADC conversion is complete.)
If this bit is cleared while a conversion is in progress, the conversion will stop and the results of the conversion up to this point will be transferred to the result registers, but the AD2IF interrupt flag bit will not be set.
If AD2ON = 0
0 = ADC conversion not in progress

bit 0 **AD2ON:** ADC Module 2 Enable bit
1 = ADC converter module 2 is operating
0 = ADC converter module 2 is shut off and consumes no operating current. All Analog channels are disconnected.

**Note 1:** PIC16LF1567 only. Not implemented on PIC16LF1566.
**2:** When the AD2DSEN bit is set; the GO/$\overline{\text{DONE}}$ bit will clear after a second conversion has completed.

 © 2015-2016 Microchip Technology Inc.

**REGISTER 16-3: AD1CH1: HARDWARE CVD 1 SECONDARY CHANNEL SELECT REGISTER(1,2,3,4)**

| 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 |
|---|---|---|---|---|---|---|---|
| CH35(5) | CH34(5) | CH33(5) | CH32(5) | CH31(5) | CH30(5) | CH19 | CH18 |
| 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 **CHx:** Channel x to A/S 1 Connection(1, 2, 3, 4)

1 = ANx is connected to A/D 1

0 = ANx is not connected to A/D 1

**Note 1:** This register selects secondary channels which are connected in parallel to the primary channel selected in AD1CON0. Precharge bias is applied to both the primary and secondary channels.

**2:** If the same channel is selected as both primary and secondary then the selection as primary takes precedence.

**3:** Enabling these bits automatically overrides the corresponding TRISx bit to tri-state the selected pin.

**4:** In the same way that the CHS bits in AD1CON0 only close the switch when the ADC is enabled, these connections and the TRISx overrides are only active if the ADC is enabled by setting ADxON.

**5:** PIC16LF1567 only. Unimplemented/ Read as '0' on PIC16LF1566.

**REGISTER 16-4: AD2CH0: HARDWARE CVD 2 SECONDARY CHANNEL SELECT REGISTER(1,2,3,4)**

| 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 |
|---|---|---|---|---|---|---|---|
| CH27 | CH26 | CH25 | CH24 | CH23 | CH22 | CH21 | CH20 |
| 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 **CHx:** Channel x to A/D 2 Connection bit(1,2,3,4,5)
1 = ANx is connected to A/D 2
0 = ANx is not connected to A/D 2

**Note 1:** This register selects secondary channels which are connected in parallel to the primary channel selected in ADxCON1. Precharge bias is applied to both the primary and secondary channels.
**2:** If the same channel is selected as both primary (ADxCON1) and secondary then the selection as primary takes precedence.
**3:** Enabling these bits automatically overrides the corresponding TRISx.x bit to tri-state the selected pin.
**4:** In the same way that the CHSx bits in ADCON0 only close the switch when the A/D is enabled, these connections and the TRIS overrides are only active if the A/D is enabled by setting ADxON.

**REGISTER 16-5: AD2CH1: ANALOG-TO-DIGITAL (A/D) 2 SECONDARY CHANNEL SELECT 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 |
|---|---|---|---|---|---|---|---|
| CH45(5) | CH44(5) | CH43(5) | CH42(5) | CH41(5) | CH40 | CH29 | CH28 |
| 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 **CHx:** Channel x to A/D 2 Connection bit(1,2,3,4)
1 = ANx is connected to A/D 2
0 = ANx is not connected to A/D 2

**Note 1:** This register selects secondary channels which are connected in parallel to the primary channel selected in ADxCON1. Precharge bias is applied to both the primary and secondary channels.
**2:** If the same channel is selected as both primary (ADxCON1) and secondary then the selection as primary takes precedence.
**3:** Enabling these bits automatically overrides the corresponding TRISx.x bit to tri-state the selected pin.
**4:** In the same way that the CHSx bits in ADCON0 only close the switch when the A/D is enabled, these connections and the TRIS overrides are only active if the A/D is enabled by setting ADxON.
**5:** PIC16LF1567 only. Unimplemented / Read as '0' on PIC16LF1566

 © 2015-2016 Microchip Technology Inc.

**REGISTER 16-6: ADCON1(1)/ADCOMCON(2): ADC 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 | GO/$\overline{\text{DONE}}$_ALL | 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:** ADC Result Format Select bit
1 = Right justified. Six Most Significant bits of ADxRESxH are set to '0' when the conversion result is loaded.
0 = Left justified. Six Least Significant bits of ADxRESxL are set to '0' when the conversion result is loaded.

bit 6-4 **ADCS<2:0>:** ADC Conversion Clock Select bits
111 = FRC (clock supplied from an internal RC oscillator)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock supplied from an internal RC oscillator)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

bit 3 **ADNREF:** ADC Negative Voltage Reference Configuration bit
1 = VREFL is connected to external VREF- pin(4)
0 = VREFL is connected to AVSS.

bit 2 **GO/$\overline{\text{DONE}}$_ALL(3):** Synchronized ADC Conversion Status bit
1 = Synchronized ADC conversion in progress. Setting this bit starts conversion in any ADC with ADxON = 1.
0 = Synchronized ADC conversion completed/ not in progress.

bit 1-0 **ADPREF<1:0>:** ADC Positive Voltage Reference Configuration bits
11 = VREFH is connected to internal Fixed Voltage Reference.
10 = VREFH is connected to external VREF+ pin(4)
01 = Reserved
00 = VREFH is connected to VDD

**Note 1:** Bank 1 name is ADCON1.
**2:** Bank 14 name is ADCOMCON.
**3:** Setting this bit triggers the GO/$\overline{\text{DONE}}$x bits in both ADCs. Each ADC will run a conversion according to its control register settings. This bit reads as an OR of the individual GO/$\overline{\text{DONE}}$x bits.
**4:** When selecting the VREF+ or VREF- pin as the source of the positive or negative reference, be aware that a minimum voltage specification exists. See **Section 25.0 "Electrical Specifications"** for details.

**REGISTER 16-7: ADxCON2: ADC CONTROL REGISTER 2(1)**

| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | TRIGSEL<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 **Unimplemented:** Read as '0'

bit 6-4 **TRIGSEL<2:0>:** Auto-Conversion Trigger Selection bits
- 111 = ADTRIG Falling Edge
- 110 = ADTRIG Rising Edge
- 101 = TMR2 match to PR2(1)
- 100 = Timer1 Overflow(1)
- 011 = Timer0 Overflow(1)
- 010 = TMR4 match to PR4
- 001 = Reserved
- 000 = No Auto Conversion Trigger selected

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

**Note 1:** Signal also sets its corresponding interrupt flag.

 © 2015-2016 Microchip Technology Inc.

**REGISTER 16-8: AADxCON3: HARDWARE CVD CONTROL REGISTER 3**

| R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| ADxEPPOL | ADxIPPOL | — | — | — | — | ADxIPEN | ADxDSEN |
| 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 **ADxEPPOL:** External Precharge Polarity bit(1)
- 1 = Selected channel is connected to $V_{DDIO}$ during precharge time
- 0 = Selected channel is connected to $V_{SS}$ during precharge time

bit 6 **ADxIPPOL:** Internal Precharge Polarity bit(1)
- 1 = $C_{HOLD}$ is shorted to $V_{REFH}$ during precharge time
- 0 = $C_{HOLD}$ is shorted to $V_{REFL}$ during precharge time

bit 5-2 **Unimplemented:** Read as '0'

bit 1 **ADxIPEN:** ADC Invert Polarity Enable bit

If ADxDSEN = 1:
- 1 = The output value of the ADxEPPOL, ADxIPPOL, and GRDxPOL bits used by the ADC are inverted for the second conversion
- 0 = The second ADC conversion proceeds like the first

If ADxDSEN = 0:
This bit has no effect.

bit 0 **ADxDSEN:** ADC Double Sample Enable bit
- 1 = The ADC immediately starts a new conversion after completing a conversion. GO/$\overline{DONE}$x bit is not automatically clear at end of conversion.
- 0 = ADC operates in the traditional, single conversion mode

**Note 1:** When the ADxDSEN = 1 and ADxIPEN = 1; the polarity of this output is inverted for the second conversion time. The stored bit value does not change.

**REGISTER 16-9: ADSTAT: HARDWARE CVD STATUS REGISTER**

| U-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 |
|---|---|---|---|---|---|---|---|
| — | AD2CONV | AD2STG<1:0> | | — | AD1CONV | AD1STG<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 **AD2CONV:** ADC2 Conversion Status bit

1 = Indicates ADC2 is in Conversion Sequence for AAD2RES1H:AAD2RES1L

0 = Indicates ADC2 is in Conversion Sequence for AAD2RES0H:AAD2RES0L (Also reads '0' when GO/$\overline{\text{DONE}}$2 = 0)

bit 5-4 **AD2STG<1:0>:** ADC2 Stage Status bit

11 = ADC2 module is in conversion stage
10 = ADC2 module is in acquisition stage
01 = ADC2 module is in precharge stage
00 = ADC2 module is not converting (same as GO/$\overline{\text{DONE}}$2= 0)

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

bit 2 **AD1CONV:** ADC2 Conversion Status bit

1 = Indicates ADC1 is in Conversion Sequence for AAD1RES1H:AAD1RES1L
0 = Indicates ADC1 is in Conversion Sequence for AAD1RES0H:AAD1RES0L (Also reads '0' when GO/$\overline{\text{DONE}}$1 = 0)

bit 1-0 **AD1STG<1:0>:** ADC1 Stage Status bit

11 = ADC1 module is in conversion stage
10 = ADC1 module is in acquisition stage
01 = ADC1 module is in precharge stage
00 = ADC1 module is not converting (same as GO/$\overline{\text{DONE}}$1= 0)

 © 2015-2016 Microchip Technology Inc.

**REGISTER 16-10: AADxPRE: HARDWARE CVD PRECHARGE 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 |
|---|---|---|---|---|---|---|---|
| — | ADxPRE<6:0> | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **Unimplemented:** Read as ‘0’

bit 6-0 **ADxPRE<6:0>:** Precharge Time Select bits(1)

111 1111 = Precharge for 127 instruction cycles

111 1110 = Precharge for 126 instruction cycles

•

•

•

000 0001 = Precharge for 1 instruction cycle (Fosc/4)

000 0000 = ADC precharge time is disabled

**Note 1:** When the FRC clock is selected as the conversion clock source, it is also the clock used for the precharge and acquisition times.

**REGISTER 16-11: AADxACQ: HARDWARE CVD ACQUISITION TIME 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 |
|---|---|---|---|---|---|---|---|
| — | AADxACQ<6:0> | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

bit 7 **Unimplemented:** Read as ‘0’

bit 6-0 **AADxACQ<6:0>:** Acquisition/Charge Share Time Select bits(1)

111 1111 = Acquisition/charge share for 127 instruction cycles

111 1110 = Acquisition/charge share for 126 instruction cycles

•

•

•

000 0001 = Acquisition/charge share for one instruction cycle (Fosc/4)

000 0000 = ADC Acquisition/charge share time is disabled

**Note 1:** When the FRC clock is selected as the conversion clock source, it is also the clock used for the precharge and acquisition times.

**REGISTER 16-12: ADxGRD: HARDWARE CVD GUARD RING 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 |
|---|---|---|---|---|---|---|---|
| GRDxBOE(2) | GRDxAOE(2) | GRDxPOL(1,2) | — | — | — | — | TXxPOL |
| 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 **GRDxBOE:** Guard Ring B Output Enable bit(2,3,5)
- 1 = ADC guard ring output is enabled to ADxGRDB(6) pin. Its corresponding TRISx bit must be clear.
- 0 = No ADC guard ring function to this pin is enabled

bit 6 **GRDxAOE:** Guard Ring A Output Enable bit(1,3,5)
- 1 = ADC Guard Ring Output is enabled to ADxGRDA(6) pin. Its corresponding TRISx, x bit must be clear.
- 0 = No ADC Guard Ring function is enabled

bit 5 **GRDxPOL:** Guard Ring Polarity Selection bit(4)
- 1 = ADCx guard ring outputs start as digital high during precharge stage
- 0 = ADCx guard ring outputs start as digital low during precharge stage

bit 4-1 **Unimplemented:** Read as '0'

bit 0 **TXxPOL:** ADC x TX Polarity Select(3,4,5). ADxTXy registers determine location of TX pins.
- 1 = TX starts as digital high during Precharge stage
- 0 = TX starts as digital low during Precharge stage

**Note 1:** If precharge is enabled (ADxPRE! = '000000'), then Guard A switches polarity at the start of Acquisition / Charge Share. If precharge is disabled, then Guard A switches polarity as soon as the GO/DONEx bit is set.

**2:** Output function "B" is constant throughout all stages of the conversion cycle. In a dual sample setup it will switch polarity at the start of precharge.

**3:** The corresponding TRISx,x bit must be set to '0' to enable output.

**4:** When the ADxDSEN = 1 and ADxIPEN = 1; the polarity of this output is inverted for the second conversion time. The stored bit value does not change.

**5:** Outputs are maintained while ADxON = 1.

**6:** ADxGRD pin locations are selectable in APFCON, Register 3-9.

 © 2015-2016 Microchip Technology Inc.

**REGISTER 16-13: ADCTX: COMMON ADC TX CONTROL REGISTER**

| U-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 |
|---|---|---|---|---|---|---|---|
| — | A2TX2 | A2TX1 | A2TX0 | | A1TX2 | A1TX1 | A1TX0 |
| 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 6-4 **A2TXx:** ADC 2 TX CH x Output Enable. Only valid if A1TXx is not enabled (A1TXx has priority).
1 = TX function on channel x enabled (ANx)
0 = TX function on channel x disabled (ANx)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **A1TXx:** ADC 1 TX CH x Output Enable
1 = TX function on channel x enabled (ANx)
0 = TX function on channel x disabled (ANx)

**REGISTER 16-14: AD1TX0: ADC 1 TX CONTROL REGISTER 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 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| TX17 | TX16 | TX15 | TX14 | TX13 | TX12 | TX11 | TX10 |
| 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 **TXx:** ADC 1 TX CH x Output Enable
1 = TX function on channel x enabled (ANx)
0 = TX function on channel x disabled (ANx)

**REGISTER 16-15: AD1TX1: ADC 1 TX 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 |
|---|---|---|---|---|---|---|---|
| TX35(1) | TX34(1) | TX33(1) | TX32(1) | TX31(1) | TX30(1) | TX19 | TX18 |
| 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 **TXx:** ADC 1 TX CH x Output Enable
1 = TX function on channel x enabled (ANx)
0 = TX function on channel x disabled (ANx)

**Note 1:** PIC16LF1567 only.

**REGISTER 16-16: AD2TX0: ADC 2 TX CONTROL REGISTER 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 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| TX27 | TX26 | TX25 | TX24 | TX23 | TX22 | TX21 | TX20 |
| 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 **TXx:** ADC 2 TX CH x Output Enable
1 = TX function on channel x enabled (ANx)
0 = TX function on channel x disabled (ANx)

**REGISTER 16-17: AD2TX1: ADC 2 TX 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 |
|---|---|---|---|---|---|---|---|
| TX45(1) | TX44(1) | TX43(1) | TX42(1) | TX41(1) | TX40 | TX29 | TX28 |
| 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 **TXx:** ADC 1 TX CH x Output Enable
1 = TX function on channel x enabled (ANx)
0 = TX function on channel x disabled (ANx)

**Note 1:** PIC16LF1567 only.

 © 2015-2016 Microchip Technology Inc.

**REGISTER 16-18: AADxCAP: HARDWARE CVD ADDITIONAL SAMPLE CAPACITOR SELECTION 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 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | ADDxCAP<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 **ADDxCAP<3:0>:** ADC Additional Sample Capacitor Selection bits

1111 = Nominal Additional Sample Capacitor of 30pF
1110 = Nominal Additional Sample Capacitor of 28pF
1101 = Nominal Additional Sample Capacitor of 26pF
1100 = Nominal Additional Sample Capacitor of 24pF
1011 = Nominal Additional Sample Capacitor of 22pF
1010 = Nominal Additional Sample Capacitor of 20pF
1001 = Nominal Additional Sample Capacitor of 18pF
1000 = Nominal Additional Sample Capacitor of 16pF
0111 = Nominal Additional Sample Capacitor of 14pF
0110 = Nominal Additional Sample Capacitor of 12pF
0101 = Nominal Additional Sample Capacitor of 10pF
0100 = Nominal Additional Sample Capacitor of 8pF
0011 = Nominal Additional Sample Capacitor of 6pF
0010 = Nominal Additional Sample Capacitor of 4pF
0001 = Nominal Additional Sample Capacitor of 2pF
0000 = Additional Sample Capacitor is Disabled

**REGISTER 16-19: AADxRESxH: HARDWARE CVD RESULT REGISTER MSB ADFM = 0(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 |
|---|---|---|---|---|---|---|---|
| ADRESx<9:2> | | | | | | | |
| 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 **ADRESx<9:2>:** ADC Result Register bits
Upper eight bits of 10-bit conversion result

**Note 1:** See **Section 16.1.13 "Hardware CVD Register Mapping"** for more information.

**REGISTER 16-20: AADxRESxL: HARDWARE CVD RESULT REGISTER LSL ADFM = 0(1)**

| R/W-x/u | R/W-x/u | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| ADRESx<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 **ADRESx<1:0>**: ADC Result Register bits
Lower two bits of 10-bit conversion result

bit 5-0 **Reserved**: Do not use.

**Note 1:** See **Section 16.1.13 "Hardware CVD Register Mapping"** for more information.

 © 2015-2016 Microchip Technology Inc.

**REGISTER 16-21: AADxRESxH: HARDWARE CVD RESULT REGISTER MSB ADFM = 1(1)**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-x/u | R/W-x/u |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | ADRESx<9:8> | |
| 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 7-2 **Reserved**: Do not use.

bit 1-0 **ADRESx<9:8>:** ADC Result Register bits
Upper two bits of 10-bit conversion result

**Note 1:** See **Section 16.1.13 “Hardware CVD Register Mapping”** for more information.

**REGISTER 16-22: AADxRESxL: HARDWARE CVD RESULT REGISTER LSB ADFM = 1(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 |
|---|---|---|---|---|---|---|---|
| ADRESx<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 **ADRESx<7:0>:** ADC Result Register bits
Lower eight bits of 10-bit conversion result

**Note 1:** See **Section 16.1.13 “Hardware CVD Register Mapping”** for more information.

**TABLE 16-2: SUMMARY OF REGISTERS ASSOCIATED WITH HARDWARE CVD**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| AADxCAP | — | — | — | — | ADDxCAP<3:0> | | | | 175 |
| AD1CON0 | CHS15 | CHS14 | CHS13 | CHS12 | CHS11 | CHS10 | GO/$\overline{\text{DONE}}$1 | AD1ON | 145 |
| AD2CON0 | CHS25 | CHS24 | CHS23 | CHS22 | CHS21 | CHS20 | GO/$\overline{\text{DONE}}$2 | AD2ON | 146 |
| ADCON1/ ADCOMCON | ADFM | ADCS<2:0> | | | ADNREF | GO/$\overline{\text{DONE}}$_ALL | ADPREF<1:0> | | 167 |
| AADxCON2 | — | TRIGSEL<2:0> | | | — | — | — | — | 168 |
| AADxCON3 | ADxEPPOL | ADxIPPOL | — | — | — | — | ADxIPEN | ADxDSEN | 169 |
| AADxGRD | GRDxBOE | GRDxAOE | GRDxPOL | — | — | — | — | — | 172 |
| AADxPRE | — | ADxPRE<6:0> | | | | | | | 171 |
| AADxRES0H | ADC Result 0 Register High | | | | | | | | 176 |
| AADxRES0L | ADC Result 0 Register Low | | | | | | | | 176 |
| AADxRES1H | ADC Result 1 Register High | | | | | | | | 177 |
| AADxRES1L | ADC Result 1 Register Low | | | | | | | | 177 |
| ADSTAT | — | AD2CONV | AD2STG<1:0> | | — | AD1CONV | AD1STG<1:0> | | 170 |
| AADxACQ | — | AADxACQ<6:0> | | | | | | | 171 |
| ANSELA | ANSA7 | ANSA6 | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 116 |
| ANSELB | ANSB7 | ANSB6 | ANSB5 | ANSB4 | ANSB3 | ANSB2 | ANSB1 | ANSB0 | 120 |
| ANSELC | ANSC7 | ANSC6 | ANSC5 | ANSC4 | ANSC3 | ANSC2 | ANSC1 | ANSC0 | 124 |
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | — | — | ADFVR<1:0> | | 136 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIE2 | — | AD2IE | — | — | BCL1IE | BCL2IE | TMR4IE | — | 86 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| PIR2 | — | AD2IF | — | — | BCL1IF | BCL2IF | TMR4IF | — | 88 |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 115 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 123 |

**Legend:** — = unimplemented read as '0'. Shaded cells are not used for hardware CVD module.

 © 2015-2016 Microchip Technology Inc.

## 17.0 TIMER0 MODULE

The Timer0 module is an 8-bit timer/counter with the following features:

- 8-bit timer/counter register (TMR0)
- 3-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 17-1 is a block diagram of the Timer0 module.

## 17.1 Timer0 Operation

The Timer0 module can be used as either an 8-bit timer or an 8-bit counter.

### 17.1.1 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. |
|---|---|

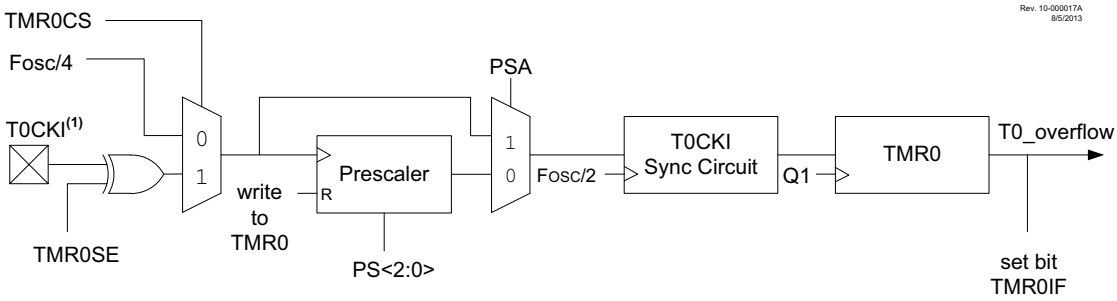
### 17.1.2 8-BIT COUNTER MODE

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 17-1: TIMER0 BLOCK DIAGRAM**




**Note 1:** The T0CKI prescale output frequency should not exceed FOSC/8.

### 17.1.3 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 eight 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.

### 17.1.4 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. |

### 17.1.5 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 25.0 "Electrical Specifications"**.

### 17.1.6 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.

 © 2015-2016 Microchip Technology Inc.

## 17.2 Register Definitions: Option Register

**REGISTER 17-1: 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 **$\overline{\text{WPUEN}}$:** 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 (FOSC/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 |
|---|---|
| 111 | 1 : 256 |
| 110 | 1 : 128 |
| 101 | 1 : 64 |
| 100 | 1 : 32 |
| 011 | 1 : 16 |
| 010 | 1 : 8 |
| 001 | 1 : 4 |
| 000 | 1 : 2 |

**TABLE 17-1: 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 |
|---|---|---|---|---|---|---|---|---|---|
| ADxCON2 | — | TRIGSEL<2:0> | | | — | — | — | — | 148 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| OPTION_REG | $\overline{\text{WPUEN}}$ | INTEDG | TMR0CS | TMR0SE | PSA | PS<2:0> | | | 181 |
| TMR0 | Holding Register for the 8-bit Timer0 Count | | | | | | | | 179* |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 115 |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used by the Timer0 module.
* Page provides register information.

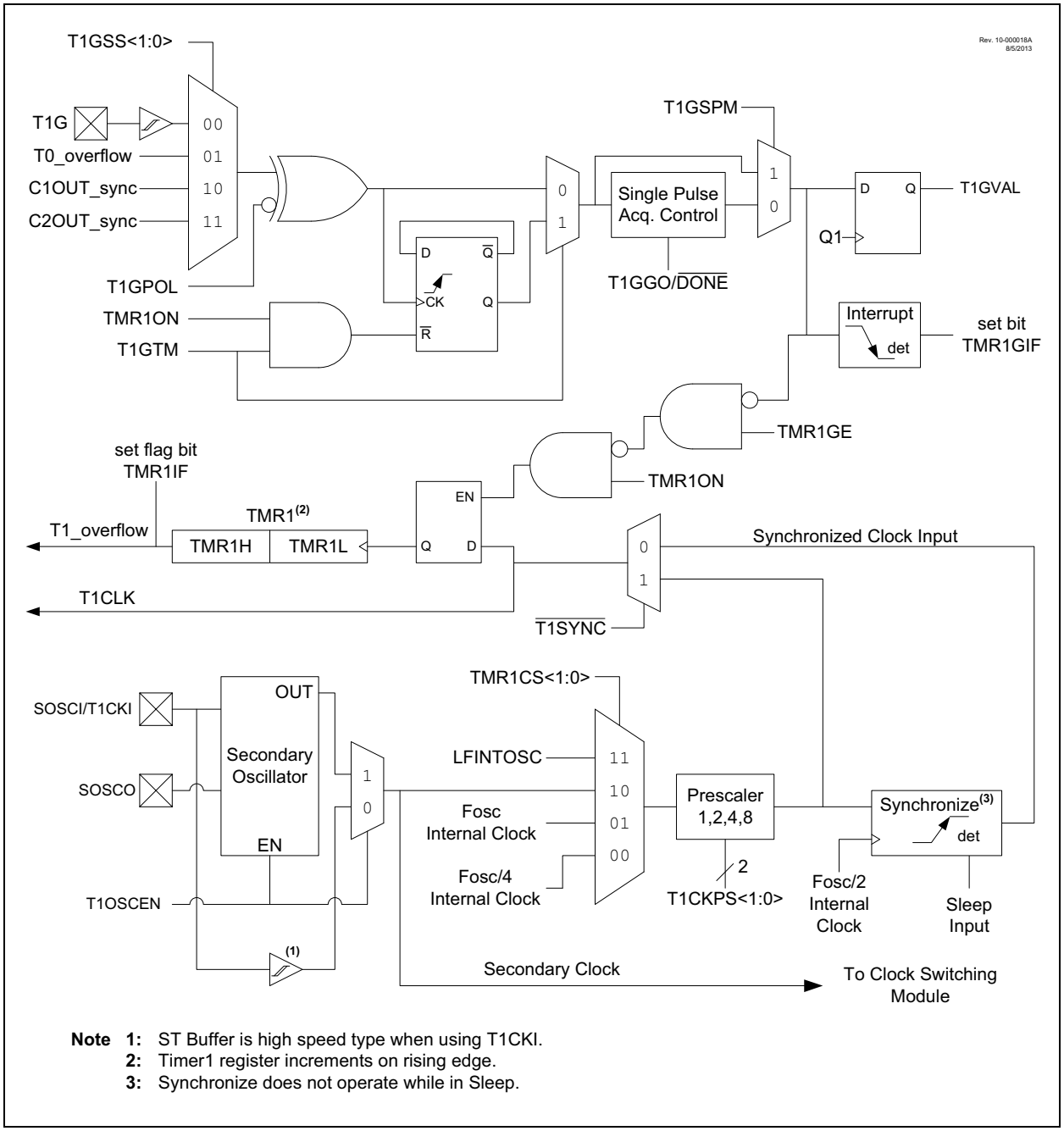
## 18.0 TIMER1 MODULE WITH GATE CONTROL

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
- Optionally synchronized comparator out
- Multiple Timer1 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- ADC Auto-Conversion Trigger(s)
- Selectable Gate Source Polarity
- Gate Toggle mode
- Gate Single-Pulse mode
- Gate Value Status
- Gate Event Interrupt

Figure 18-1 is a block diagram of the Timer1 module.

**FIGURE 18-1: 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.


© 2015-2016 Microchip Technology Inc.

## 18.1 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 18-1 displays the Timer1 enable selections.

**TABLE 18-1: TIMER1 ENABLE SELECTIONS**

| TMR1ON | TMR1GE | Timer1 Operation |
|---|---|---|
| 0 | 0 | Off |
| 0 | 1 | Off |
| 1 | 0 | Always On |
| 1 | 1 | Count Enabled |

## 18.2 Clock Source Selection

The TMR1CS<1:0> bits of the T1CON register are used to select the clock source for Timer1. Table 18-2 displays the clock source selections.

### 18.2.1 INTERNAL CLOCK SOURCE

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

When the FOSC 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

### 18.2.2 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. The external clock source can be synchronized to the microcontroller system clock or it can run asynchronously.

**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 18-2: CLOCK SOURCE SELECTIONS**

| TMR1CS<1:0> | Clock Source |
|---|---|
| 11 | LFINTOSC |
| 10 | External Clocking on T1CKI Pin |
| 01 | System Clock (FOSC) |
| 00 | Instruction Clock (FOSC/4) |

## 18.3 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.

## 18.4 Timer1 Operation in Asynchronous Counter Mode

If 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 18.4.1 "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. |

### 18.4.1 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.

## 18.5 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.

### 18.5.1 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 18-3 for timing details.

**TABLE 18-3: TIMER1 GATE ENABLE SELECTIONS**

| T1CLK | T1GPOL | T1G | Timer1 Operation |
|---|---|---|---|
| ↑ | 0 | 0 | Counts |
| ↑ | 0 | 1 | Holds Count |
| ↑ | 1 | 0 | Holds Count |
| ↑ | 1 | 1 | Counts |

### 18.5.2 TIMER1 GATE SOURCE SELECTION

Timer1 gate source selections are shown in Table 18-4. Source selection is controlled by the T1GSS bit 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 18-4: TIMER1 GATE SOURCES**

| T1GSS | Timer1 Gate Source |
|---|---|
| 0 | Timer1 Gate pin (T1G) |
| 1 | Overflow of Timer0 (T0_overflow) (TMR0 increments from FFh to 00h) |

 © 2015-2016 Microchip Technology Inc.

#### 18.5.2.1 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.

#### 18.5.2.2 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.

### 18.5.3 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 18-4 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. |
|---|---|

### 18.5.4 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/$\overline{\text{DONE}}$ 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/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the T1GGO/$\overline{\text{DONE}}$ bit is once again set in software. See Figure 18-5 for timing details.

If the Single Pulse Gate mode is disabled by clearing the T1GSPM bit in the T1GCON register, the T1GGO/$\overline{\text{DONE}}$ 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 18-6 for timing details.

### 18.5.5 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).

### 18.5.6 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).

## 18.6 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. |
|---|---|

## 18.7 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

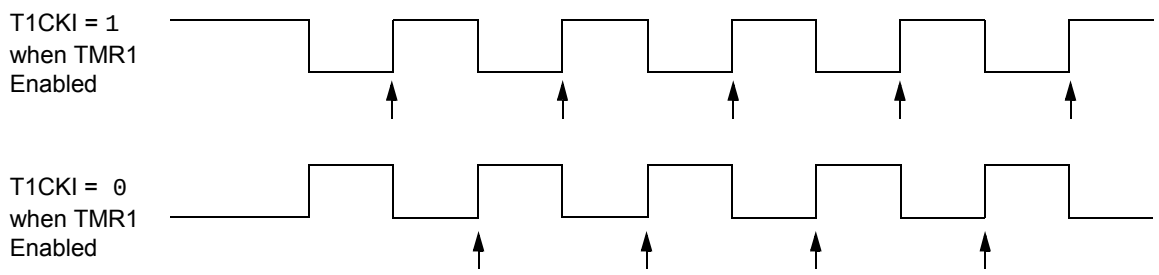
The device will wake-up on an overflow and execute the next instructions. If the GIE bit of the INTCON register is set, the device will call the Interrupt Service Routine.

Timer1 oscillator will continue to operate in Sleep regardless of the $\overline{\text{T1SYNC}}$ bit setting.

### 18.7.1 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 11.1 "Alternate Pin Function"** for more information.

**FIGURE 18-2: TIMER1 INCREMENTING EDGE**




**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.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 18-3: TIMER1 GATE ENABLE MODE**

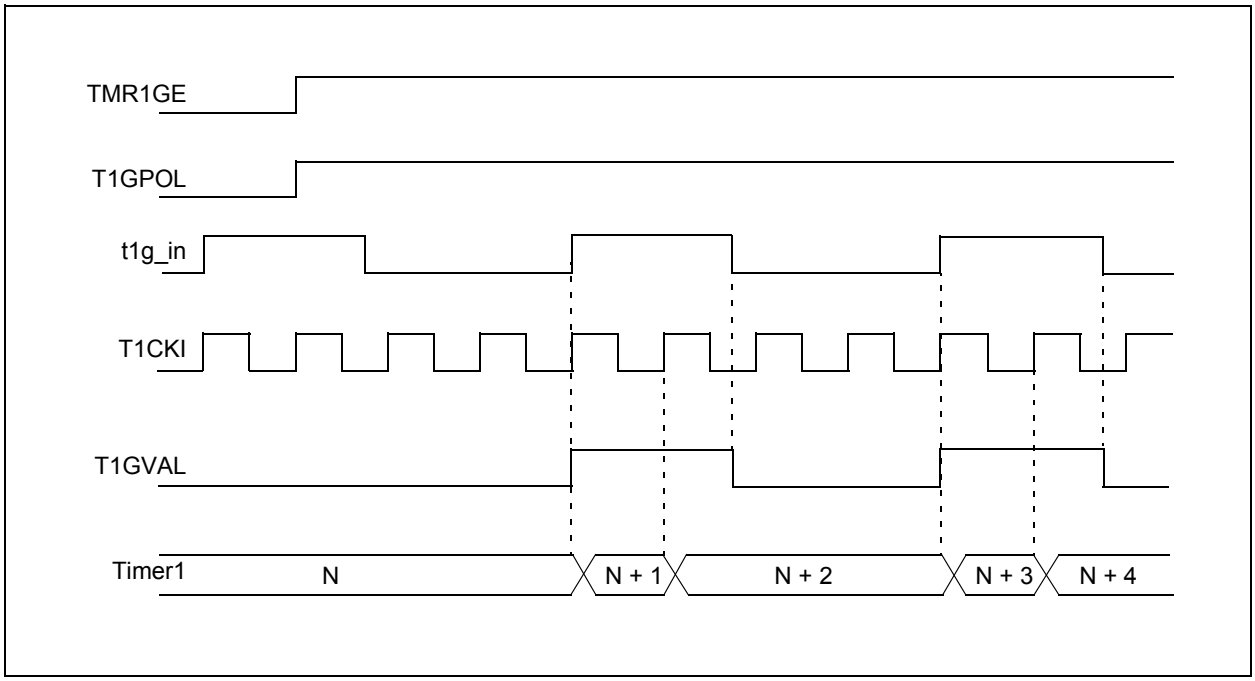



**FIGURE 18-4: TIMER1 GATE TOGGLE MODE**

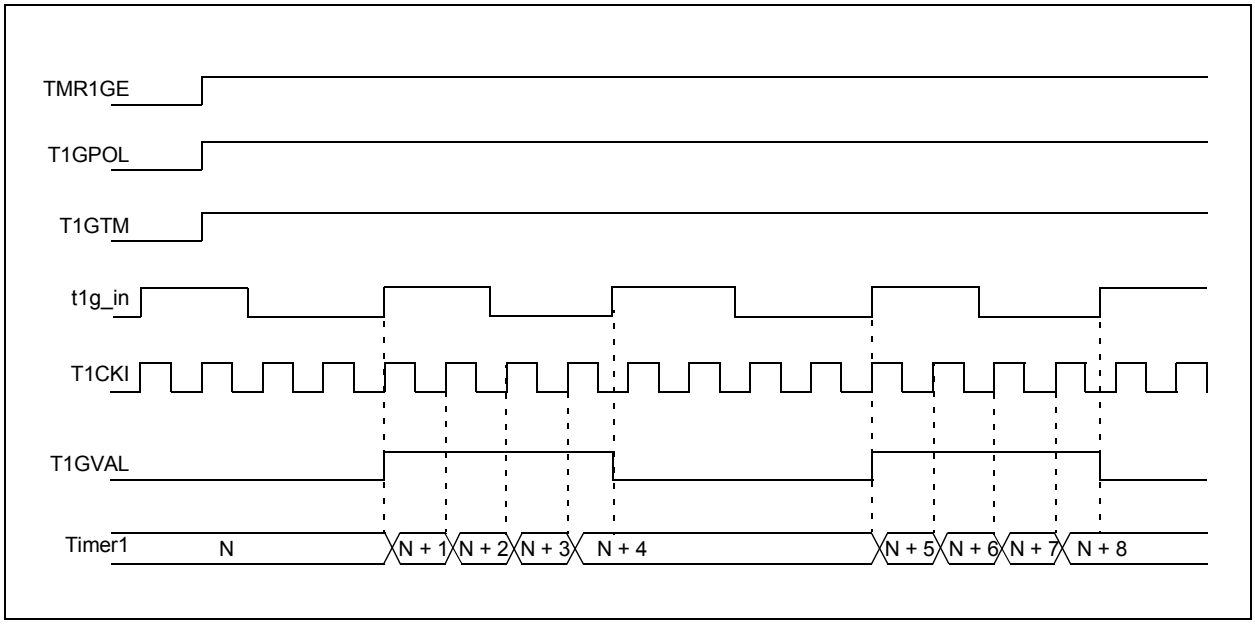
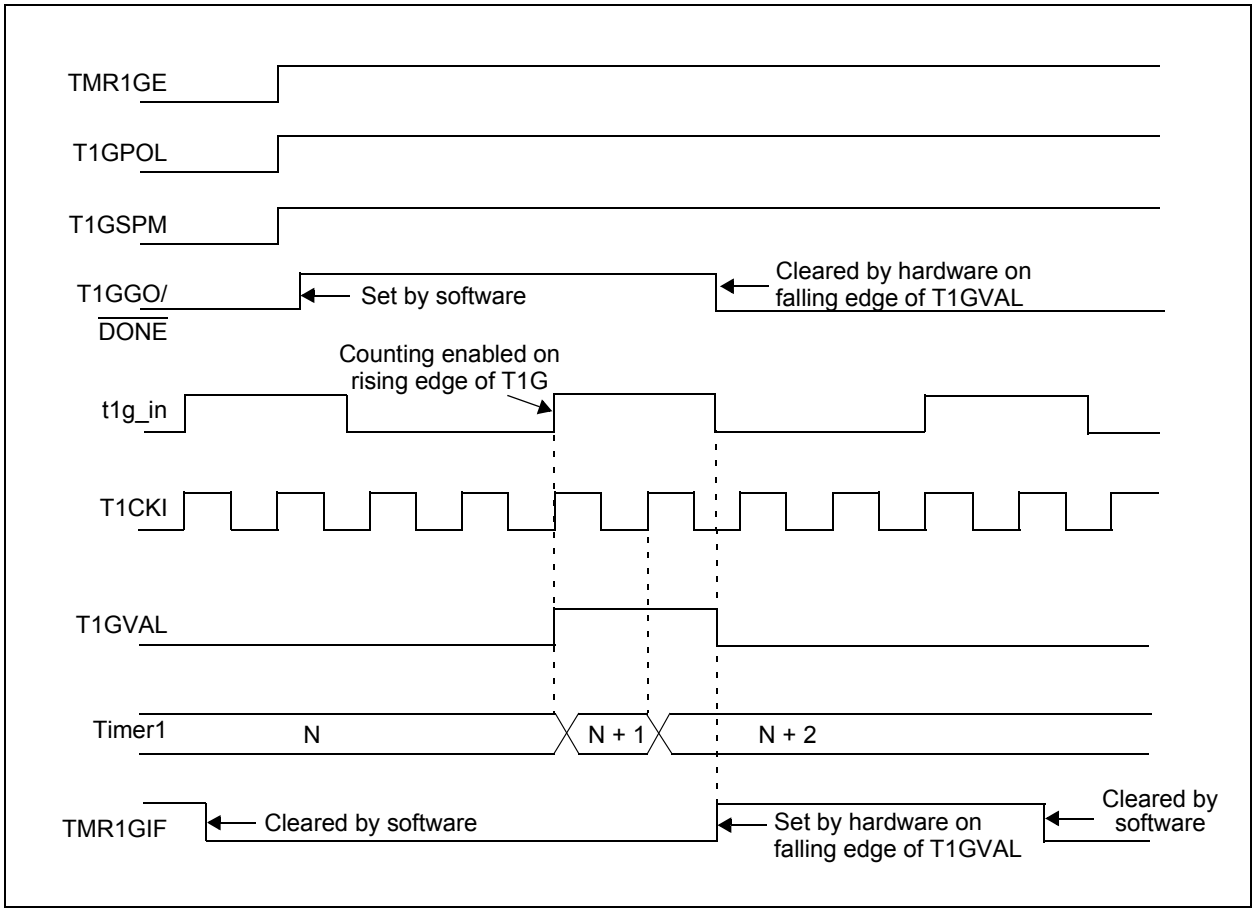
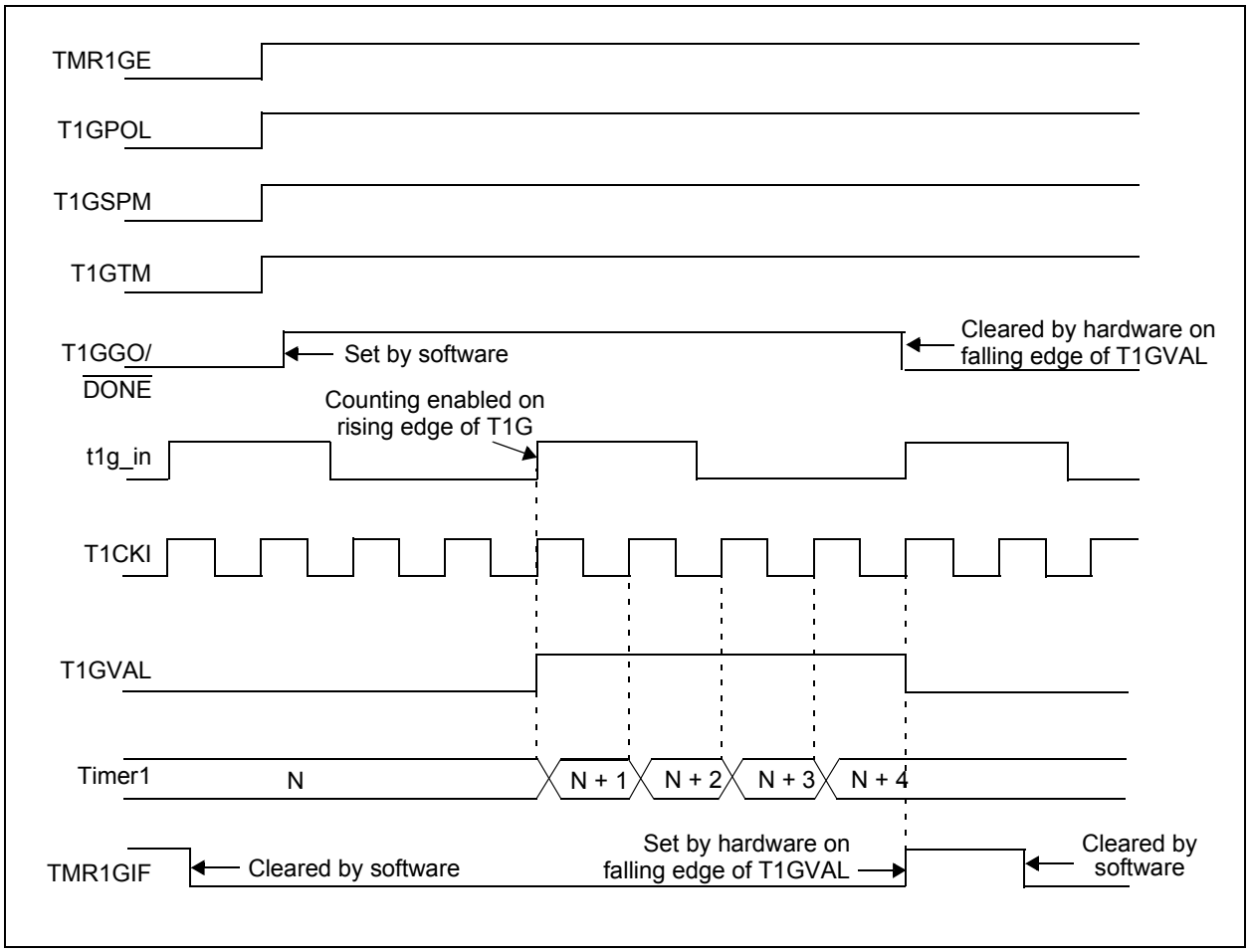


**FIGURE 18-5: TIMER1 GATE SINGLE-PULSE MODE**




 © 2015-2016 Microchip Technology Inc.

**FIGURE 18-6: TIMER1 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE**



## 18.8 Register Definitions: Timer1 Control

**REGISTER 18-1: T1CON: TIMER1 CONTROL REGISTER**

| R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | U-0 | R/W-0/u | U-0 | R/W-0/u |
|---|---|---|---|---|---|---|---|
| TMR1CS<1:0> | | T1CKPS<1:0> | | — | $\overline{\text{T1SYNC}}$ | — | 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 = Timer1 clock source is LFINTOSC
- 10 = Timer1 clock source is external clock from T1CKI pin (on the rising edge)
- 01 = Timer1 clock source is system clock (FOSC)
- 00 = Timer1 clock source is instruction clock (FOSC/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 **Unimplemented:** Read as '0'

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 Synchronization Control bit
- 1 = Do not synchronize asynchronous clock input
- 0 = Synchronize asynchronous clock input with system clock (FOSC)

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

bit 0 **TMR1ON:** Timer1 On bit
- 1 = Enables Timer1
- 0 = Stops Timer1 and clears Timer1 gate flip-flop

 © 2015-2016 Microchip Technology Inc.

**REGISTER 18-2: 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 | U-0 | R/W-0/u |
|---|---|---|---|---|---|---|---|
| TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/$\overline{\text{DONE}}$ | T1GVAL | — | T1GSS |
| 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/$\overline{\text{DONE}}$:** 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 Value Status bit

Indicates the current state of the Timer1 gate that could be provided to TMR1H:TMR1L.
Unaffected by Timer1 Gate Enable (TMR1GE).

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **T1GSS:** Timer1 Gate Source Select bits

01 = Timer0 overflow output (T0_overflow)
00 = Timer1 gate pin (T1G)

**TABLE 18-5: 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 |
|---|---|---|---|---|---|---|---|---|---|
| ANSELA | ANSA7 | ANSA6 | ANSA5 | ANSA4 | ANSA3 | ANSA2 | ANSA1 | ANSA0 | 116 |
| APFCON | — | — | SSSEL | — | — | — | GRDBSEL | GRDASEL | 113 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| OSCSTAT | — | PLLSR | — | HFIOFR | — | — | LFIOFR | HFIOFS | 70 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Count | | | | | | | | 186* |
| TMR1L | Holding Register for the Least Significant Byte of the 16-bit TMR1 Count | | | | | | | | 186* |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 115 |
| T1CON | TMR1CS<1:0> | | T1CKPS<1:0> | | — | $\overline{\text{T1SYNC}}$ | — | TMR1ON | 190 |
| T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/$\overline{\text{DONE}}$ | T1GVAL | — | T1GSS | 191 |

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


© 2015-2016 Microchip Technology Inc.

## 19.0 TIMER2/4 MODULES

There are up to five identical Timer2-type modules available. To maintain pre-existing naming conventions, the Timers are called Timer2 and Timer4 (also Timer2/4).

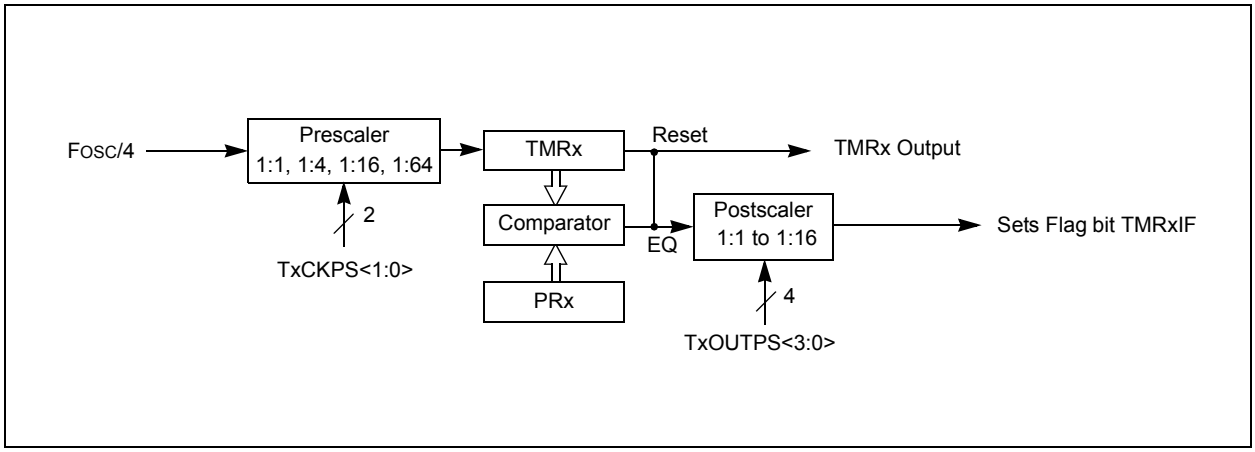
| **Note:** | The 'x' variable used in this section is used to designate Timer2 or Timer4. For example, TxCON references T2CON or T4CON. PRx references PR2 or PR4. |
|---|---|

The Timer2/4 modules incorporate the following features:

- 8-bit Timer and Period registers (TMR2/4 and PR2/4, 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 TMR2/4 match with PR2/4, respectively
- Optional use as the shift clock for the MSSPx modules (Timer2 only)

See Figure 19-1 for a block diagram of Timer2/4.

**FIGURE 19-1: TIMER2/4 BLOCK DIAGRAM**



## 19.1 Timer2/4 Operation

The clock input to the Timer2/4 modules is the system instruction clock (FOSC/4).

TMR2/4 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, TxCKPS<1:0> of the TxCON register. The value of TMR2/4 is compared to that of the Period register, PR2/4, 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 TMR2/4 to 00h on the next cycle and drives the output counter/postscaler (see **Section 19.2 "Timer2/4 Interrupt"**).

The TMR2/4 and PR2/4 registers are both directly readable and writable. The TMR2/4 register is cleared on any device Reset, whereas the PR2/4 register initializes to FFh. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the TMR2/4 register
- a write to the TxCON register
- Power-on Reset (POR)
- Brown-out Reset (BOR)
- $\overline{\text{MCLR}}$ Reset
- Watchdog Timer (WDT) Reset
- Stack Overflow Reset
- Stack Underflow Reset
- RESET Instruction

| **Note:** | TMR2/4 is not cleared when TxCON is written. |
|---|---|

## 19.2 Timer2/4 Interrupt

Timer2/4 can also generate an optional device interrupt. The Timer2/4 output signal (TMRx-to-PRx match) provides the input for the 4-bit counter/postscaler. This counter generates the TMRx match interrupt flag which is latched in TMRxIF of the PIRx register. The interrupt is enabled by setting the TMR2/4 Match Interrupt Enable bit, TMRxIE of the PIEx register.

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

## 19.3 Timer2/4 Output

The unscaled output of TMR2/4 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 MSSPx modules operating in SPI mode. Additional information is provided in **Section 20.1 "Master SSPx (MSSPx) Module Overview"**

## 19.4 Timer2/4 Operation During Sleep

The Timer2/4 timers cannot be operated while the processor is in Sleep mode. The contents of the TMR2/4 and PR2/4 registers will remain unchanged while the processor is in Sleep mode.

 © 2015-2016 Microchip Technology Inc.

## 19.5 Register Definitions: Timer2/4 Control

**REGISTER 19-1: TxCON: TIMER2/TIMER4 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 |
|---|---|---|---|---|---|---|---|
| — | TxOUTPS<3:0> | | | | TMRxON | TxCKPS<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 **TxOUTPS<3:0>:** Timerx 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 **TMRxON:** Timerx On bit
1 = Timer2/4 is on
0 = Timer2/4 is off

bit 1-0 **TxCKPS<1:0>:** Timer2-type Clock Prescale Select bits
11 = Prescaler is 64
10 = Prescaler is 16
01 = Prescaler is 4
00 = Prescaler is 1

**TABLE 19-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2/4**

| 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 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIE2 | — | AD2IE | — | — | BCL1IE | BCL2IE | TMR4IE | — | 86 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| PIR2 | — | AD2IF | — | — | BCL1IF | BCL2IF | TMR4IF | — | 88 |
| PR2 | Timer2 Module Period Register | | | | | | | | 194* |
| PR4 | Timer4 Module Period Register | | | | | | | | 194* |
| T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS1 | T2CKPS0 | 195 |
| T4CON | — | T4OUTPS<3:0> | | | | TMR4ON | T4CKPS1 | T4CKPS0 | 195 |
| TMR2 | Holding Register for the 8-bit TMR2 Register | | | | | | | | 193* |
| TMR4 | Holding Register for the 8-bit TMR4 Register(1) | | | | | | | | 193* |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for Timer2/4 module.

* Page provides register information.

 © 2015-2016 Microchip Technology Inc.

# 20.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP1 AND MSSP2) MODULE

## 20.1 Master SSPx (MSSPx) Module Overview

The Master Synchronous Serial Port (MSSPx) 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 MSSPx module can operate in one of two modes:

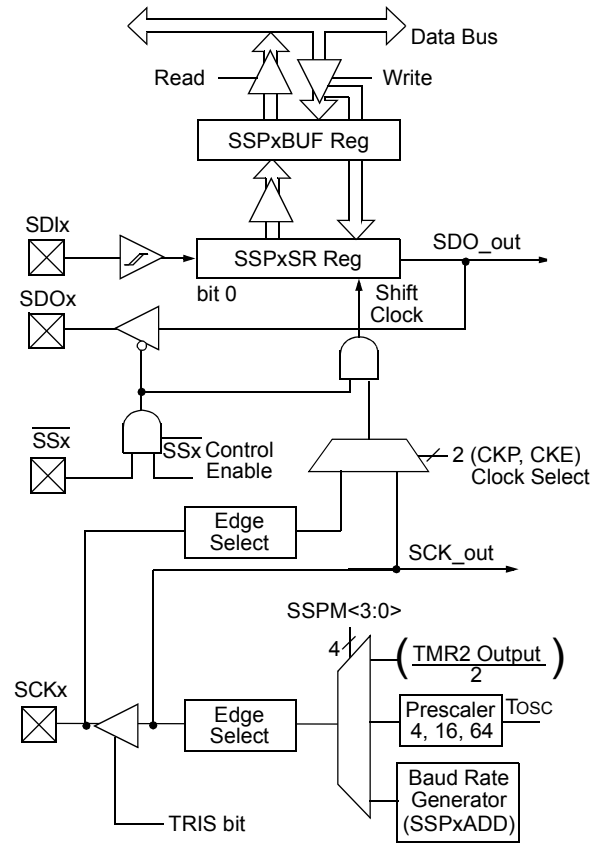
- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit ($I^2C$)

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 20-1 is a block diagram of the SPI interface module.

**FIGURE 20-1: MSSPX BLOCK DIAGRAM (SPI MODE)**



The I²C interface supports the following modes and features:

- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited Multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection
- General call address matching
- Address masking
- Address Hold and Data Hold modes
- Selectable SDAx hold times

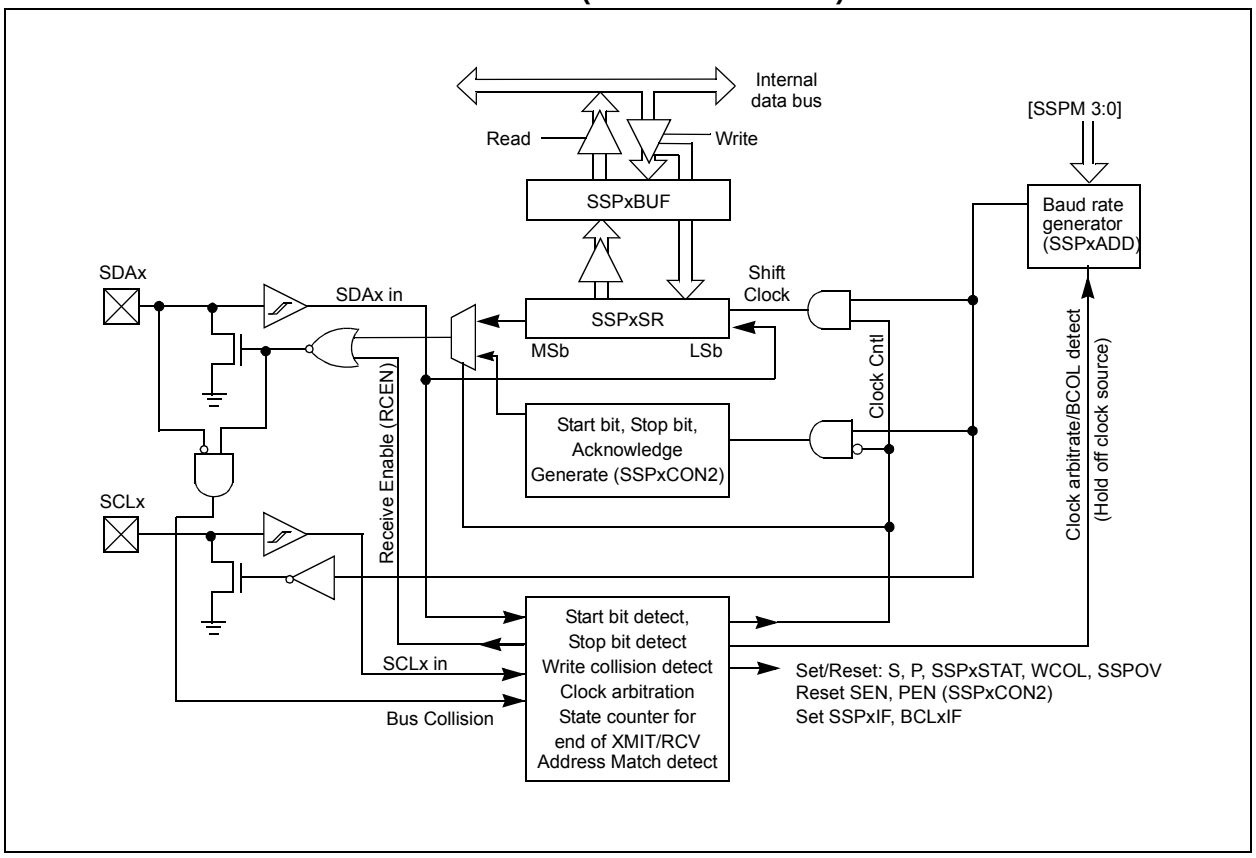
Figure 20-2 is a block diagram of the I²C Interface module in Master mode. Figure 20-3 is a diagram of the I²C interface module in Slave mode.

The PIC12LF1552 has two MSSP modules, MSSP1 and MSSP2, each module operating independently from the other.

**Note 1:** In devices with more than one MSSP module, it is very important to pay close attention to SSPxCONx register names. SSP1CON1 and SSP1CON2 registers control different operational aspects of the same module, while SSP1CON1 and SSP2CON1 control the same features for two different modules.

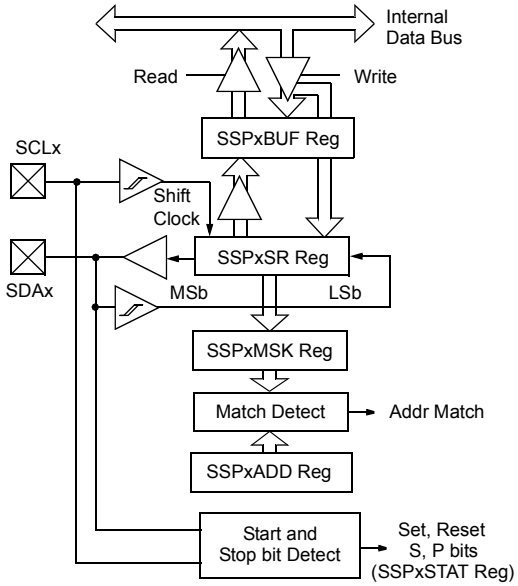
**2:** Throughout this section, generic references to an MSSP module in any of its operating modes may be interpreted as being equally applicable to MSSP1 or MSSP2. Register names, module I/O signals, and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module when required.

**FIGURE 20-2: MSSPX BLOCK DIAGRAM (I²C MASTER MODE)**




 © 2015-2016 Microchip Technology Inc.

**FIGURE 20-3: MSSPX BLOCK DIAGRAM ($I^2C$ SLAVE MODE)**



## 20.2 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 (SCKx)
- Serial Data Out (SDOx)
- Serial Data In (SDIx)
- Slave Select ($\overline{SS}$x)

Figure 20-1 shows the block diagram of the MSSPx 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 20-4 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 20-5 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 SDOx output pin which is connected to, and received by, the slave's SDIx input pin. The slave device transmits information out on its SDOx output pin, which is connected to, and received by, the master's SDIx 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 SDOx 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 SDOx pin) and the master device is reading this bit and saving it as the LSb of its shift register.

After eight bits have been shifted out, the master and slave have exchanged register values.

If there is more data to exchange, the shift registers are loaded with new data and the process repeats itself.

Whether the data is meaningful or not (dummy data), depends on the application software. This leads to three scenarios for data transmission:

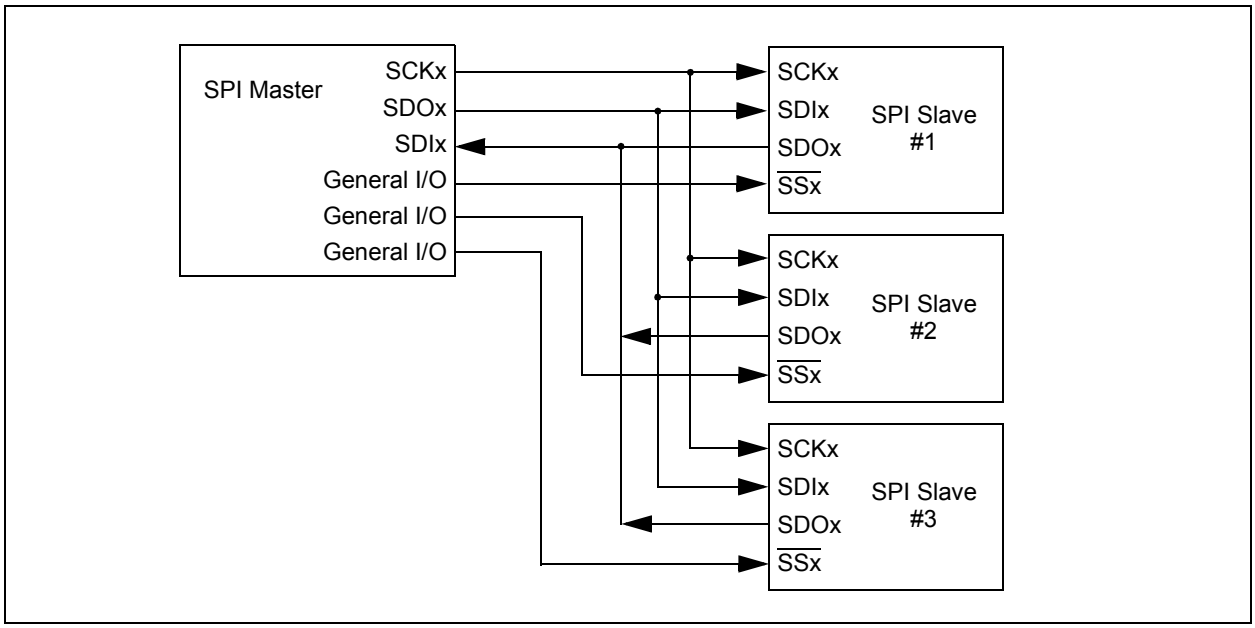
- Master sends useful data and slave sends dummy data.
- Master sends useful data and slave sends useful data.
- Master sends dummy data and slave sends useful data.

Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops sending the clock signal and it deselects the slave.

Every slave device connected to the bus that has not been selected through its slave select line must disregard the clock and transmission signals and must not transmit out any data of its own.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 20-4: SPI MASTER AND MULTIPLE SLAVE CONNECTION**




### 20.2.1 SPI MODE REGISTERS

The MSSPx module has five registers for SPI mode operation. These are:

- MSSPx STATUS register (SSPxSTAT)
- MSSPx Control register 1 (SSPxCON1)
- MSSPx Control register 3 (SSPxCON3)
- MSSPx Data Buffer register (SSPxBUF)
- MSSPx Address register (SSPxADD)
- MSSPx Shift register (SSPxSR) (Not directly accessible)

SSPxCON1 and SSPxSTAT are the control and STATUS registers in SPI mode operation. The SSPxCON1 register is readable and writable. The lower six bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

In one SPI master mode, SSPxADD can be loaded with a value used in the Baud Rate Generator. More information on the Baud Rate Generator is available in **Section 20.7 "Baud Rate Generator"**.

SSPxSR is the shift register used for shifting data in and out. SSPxBUF provides indirect access to the SSPxSR register. SSPxBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPxSR and SSPxBUF together create a buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

### 20.2.2 SPI MODE OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPxCON1<5:0> and SSPxSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCKx is the clock output)
- Slave mode (SCKx is the clock input)
- Clock Polarity (Idle state of SCKx)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCKx)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

To enable the serial port, SSPx Enable bit, SSPEN of the SSPxCON1 register, must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPxCONx registers and then set the SSPEN bit. This configures the SDIx, SDOx, SCKx and $\overline{SSx}$ 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:

- SDIx must have corresponding TRIS bit set
- SDOx must have corresponding TRIS bit cleared
- SCKx (Master mode) must have corresponding TRIS bit cleared
- SCKx (Slave mode) must have corresponding TRIS bit set
- $\overline{SSx}$ must have corresponding TRIS bit set

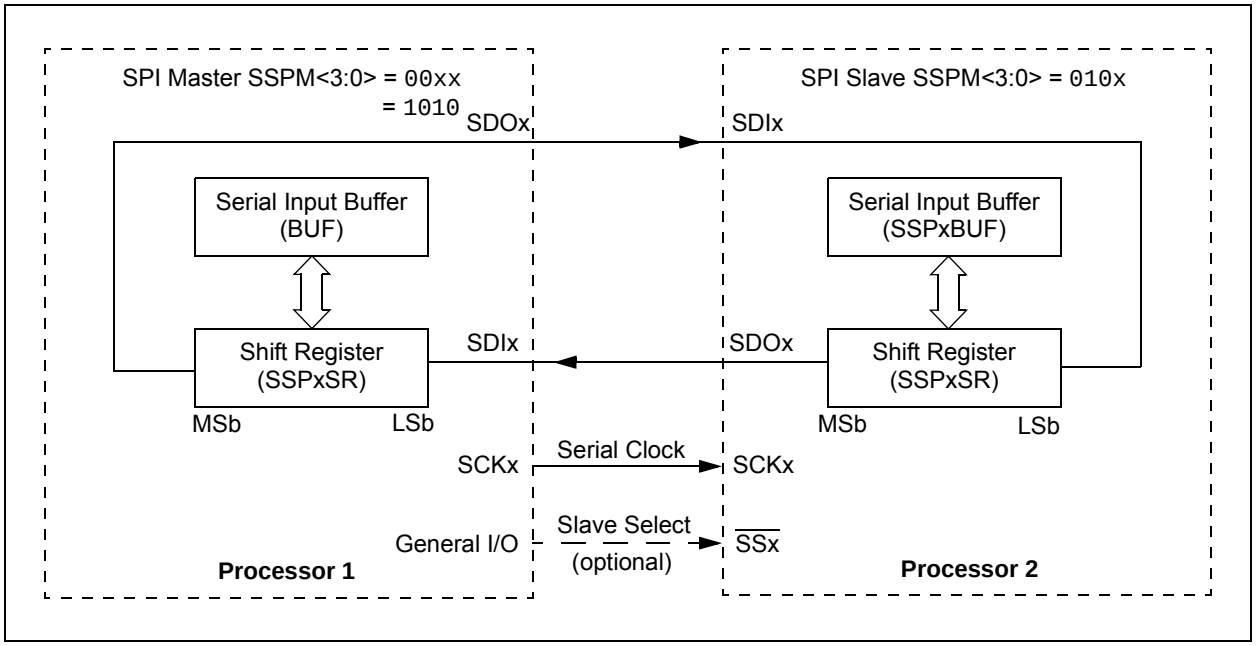
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

The MSSPx consists of a transmit/receive shift register (SSPxSR) and a buffer register (SSPxBUF). The SSPxSR shifts the data in and out of the device, MSb first. The SSPxBUF holds the data that was written to the SSPxSR until the received data is ready. Once the eight bits of data have been received, that byte is moved to the SSPxBUF register. Then, the Buffer Full Detect bit, BF of the SSPxSTAT register, and the interrupt flag bit, SSPxIF, are set. This double-buffering of the received data (SSPxBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPxBUF register during transmission/reception of data will be ignored and the write collision detect bit WCOL of the SSPxCON1 register will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPxBUF register to complete successfully.

When the application software is expecting to receive valid data, the SSPxBUF should be read before the next byte of data to transfer is written to the SSPxBUF. The Buffer Full bit, BF of the SSPxSTAT register, indicates when SSPxBUF has been loaded with the received data (transmission is complete). When the SSPxBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSPx 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 SSPxSR is not directly readable or writable and can only be accessed by addressing the SSPxBUF register. Additionally, the SSPxSTAT register indicates the various Status conditions.

**FIGURE 20-5: SPI MASTER/SLAVE CONNECTION**




 © 2015-2016 Microchip Technology Inc.

### 20.2.3 SPI MASTER MODE

The master can initiate the data transfer at any time because it controls the SCKx line. The master determines when the slave (Processor 2, Figure 20-5) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPxBUF register is written to. If the SPI is only going to receive, the SDOx output could be disabled (programmed as an input). The SSPxSR register will continue to shift in the signal present on the SDIx pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPxBUF register as if a normal received byte (interrupts and Status bits appropriately set).

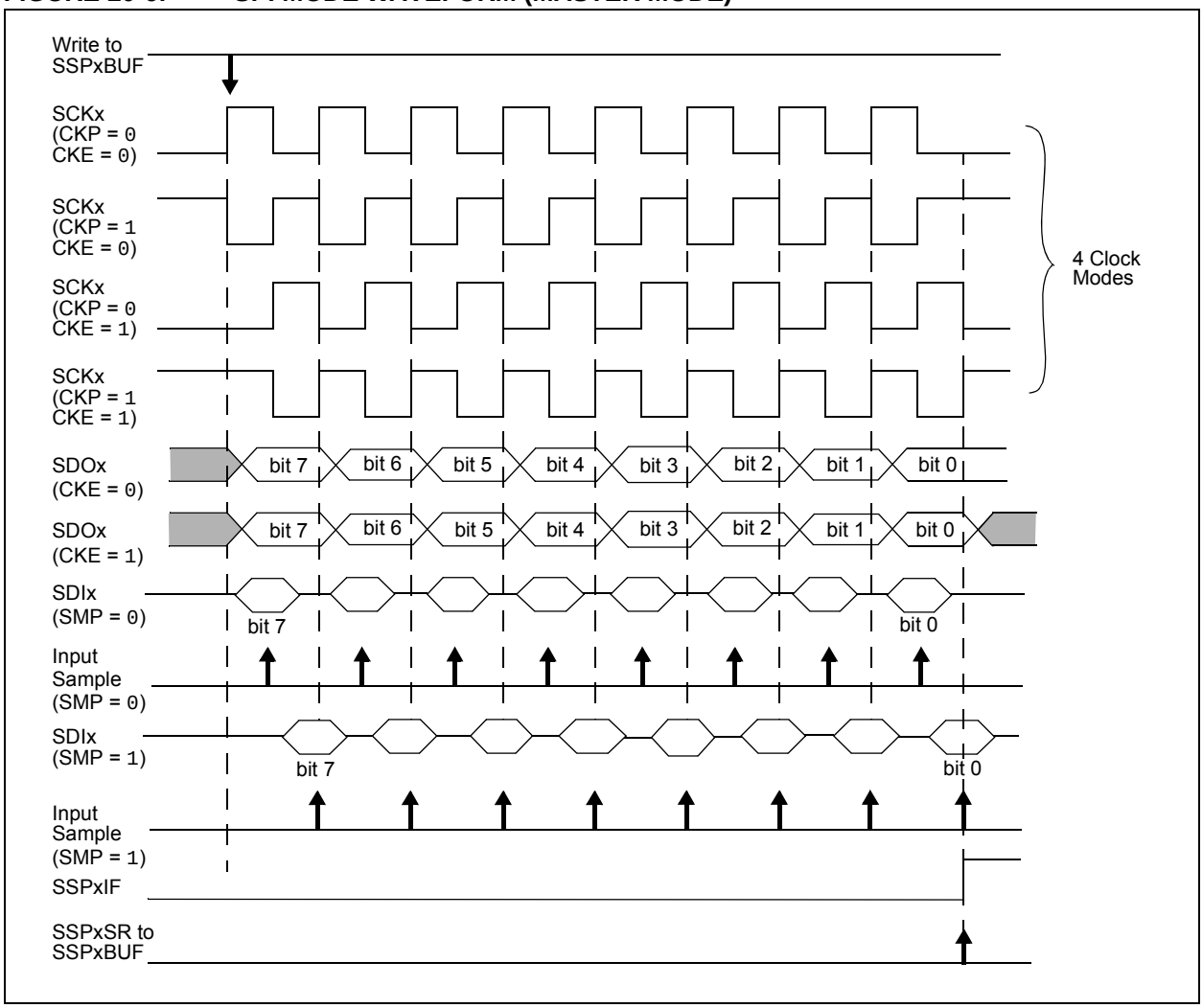
The clock polarity is selected by appropriately programming the CKP bit of the SSPxCON1 register and the CKE bit of the SSPxSTAT register. This then, would give waveforms for SPI communication as shown in Figure 20-6, Figure 20-8, Figure 20-9 and Figure 20-10, where the MSb is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 * TCY)
- FOSC/64 (or 16 * TCY)
- Timer2 output/2
- Fosc/(4 * (SSPxADD + 1))

Figure 20-6 shows the waveforms for Master mode.

When the CKE bit is set, the SDOx data is valid before there is a clock edge on SCKx. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPxBUF is loaded with the received data is shown.

**FIGURE 20-6: SPI MODE WAVEFORM (MASTER MODE)**



### 20.2.4 SPI SLAVE MODE

In Slave mode, the data is transmitted and received as external clock pulses appear on SCKx. When the last bit is latched, the SSPxIF 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 SCKx pin. The Idle state is determined by the CKP bit of the SSPxCON1 register.

While in Slave mode, the external clock is supplied by the external clock source on the SCKx 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 SCKx pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

#### 20.2.4.1 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 20-7 shows the block diagram of a typical daisy-chain connection when operating in SPI mode.

In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the BOEN bit of the SSPxCON3 register will enable writes to the SSPxBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

### 20.2.5 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 $\overline{SSx}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SSx}$ pin control enabled (SSPxCON1<3:0> = 0100).

When the $\overline{SSx}$ pin is low, transmission and reception are enabled and the SDOx pin is driven.

When the $\overline{SSx}$ pin goes high, the SDOx 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 $\overline{SSx}$ pin control enabled (SSPxCON1<3:0> = 0100), the SPI module will reset if the $\overline{SSx}$ pin is set to VDD.

**2:** When the SPI is used in Slave mode with CKE set; the user must enable $\overline{SSx}$ pin control.

**3:** While operated in SPI Slave mode the SMP bit of the SSPxSTAT register must remain clear.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the $\overline{SSx}$ pin to a high level or clearing the SSPEN bit.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 20-7: SPI DAISY-CHAIN CONNECTION**

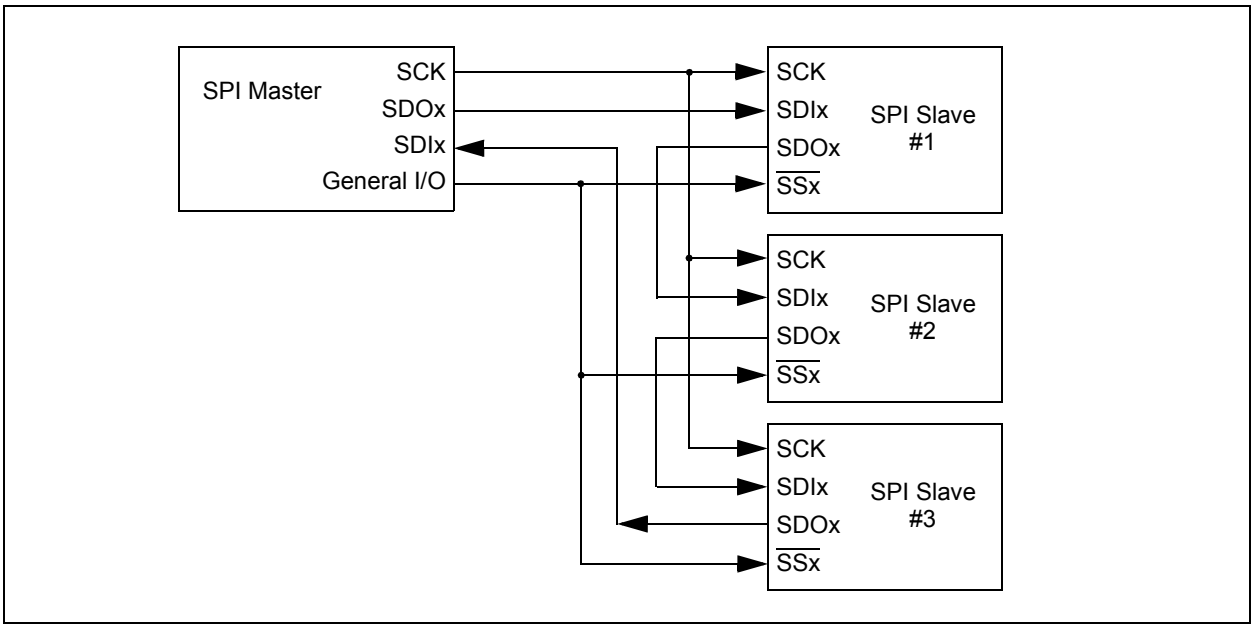



**FIGURE 20-8: SLAVE SELECT SYNCHRONOUS WAVEFORM**

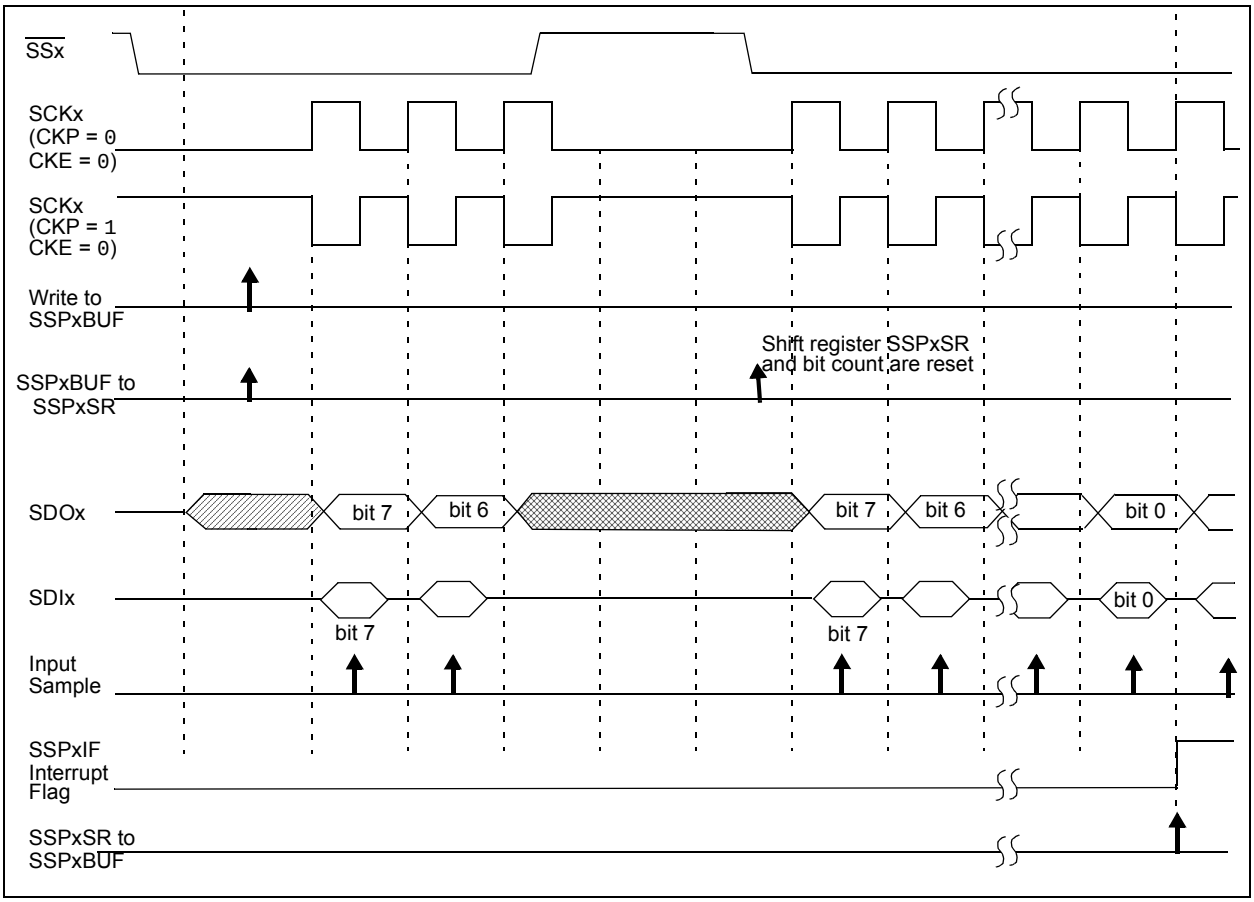


**FIGURE 20-9: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**

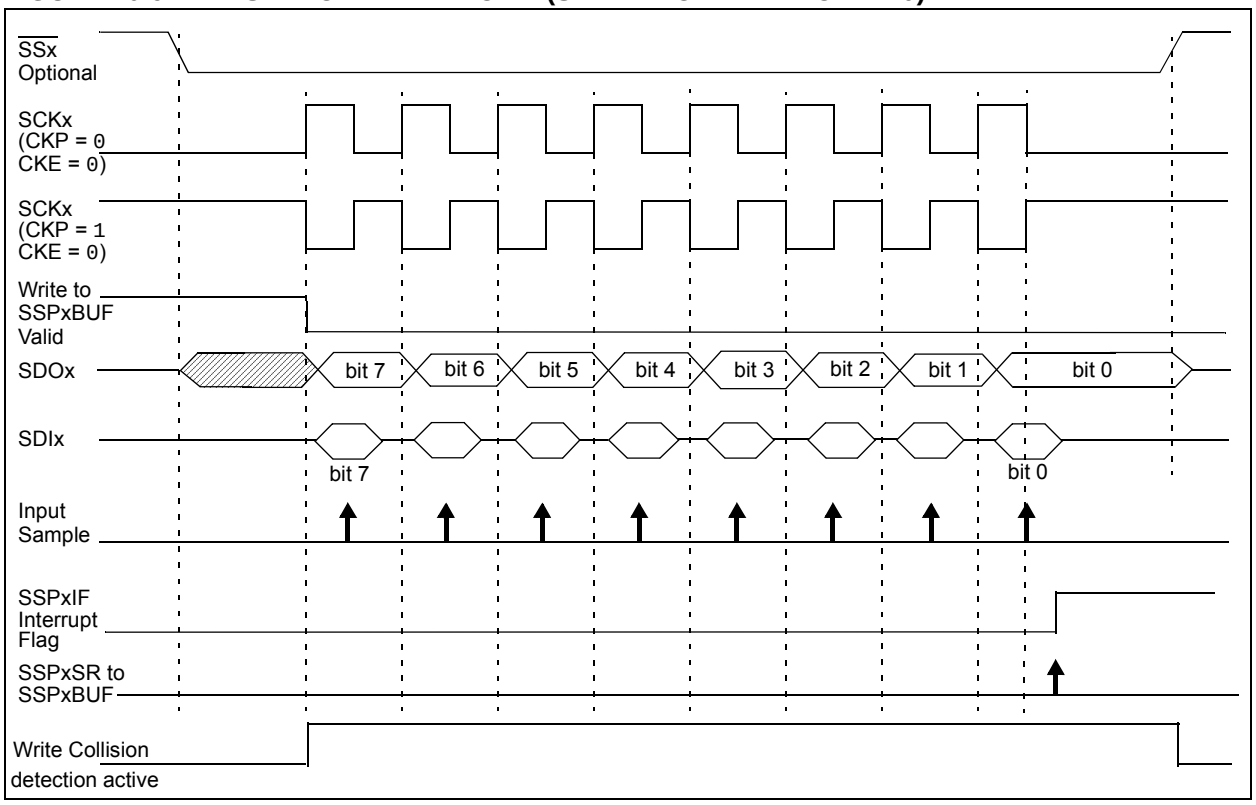
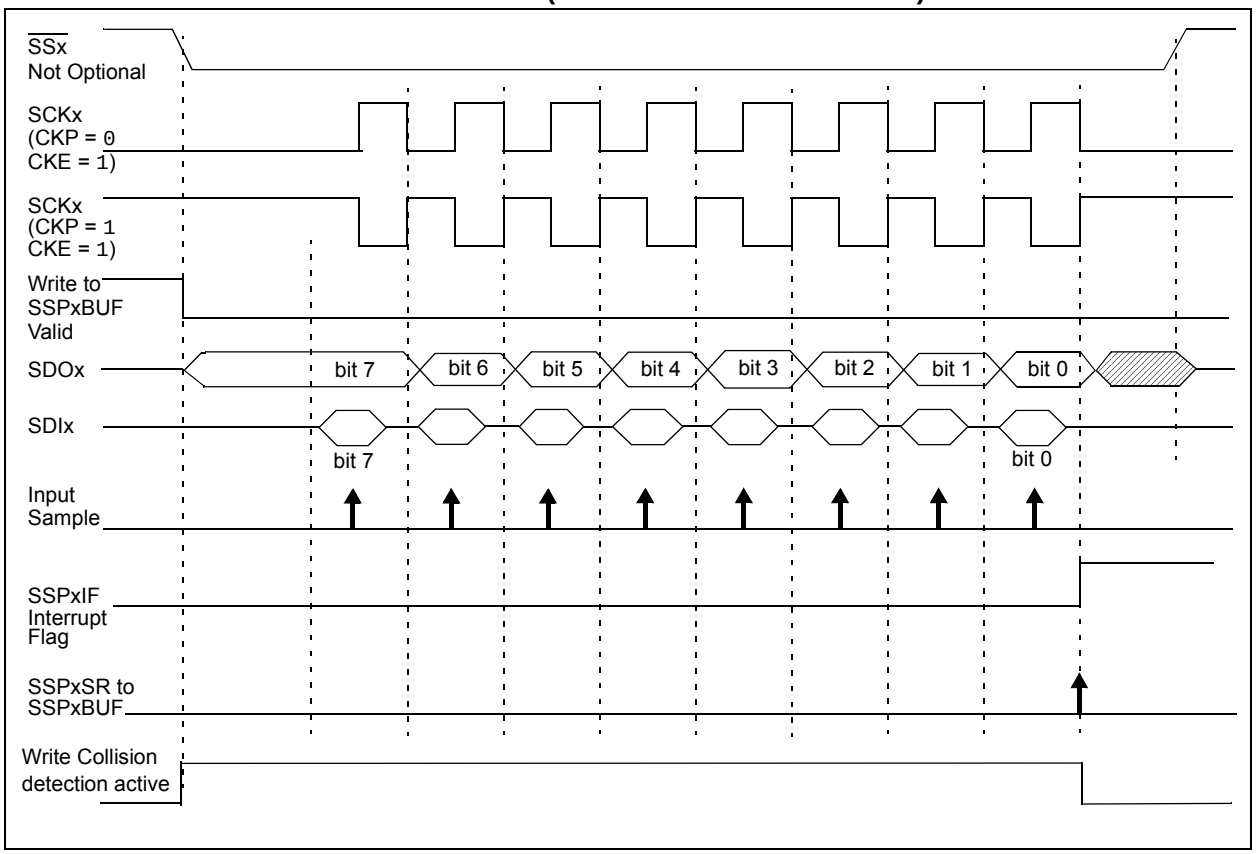



**FIGURE 20-10: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**




 © 2015-2016 Microchip Technology Inc.

### 20.2.6 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 MSSPx clock is much faster than the system clock.

In Slave mode, when MSSPx interrupts are enabled, after the master completes sending data, an MSSPx interrupt will wake the controller from Sleep.

If an exit from Sleep mode is not desired, MSSPx interrupts should be disabled.

In SPI Master mode, when the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all eight bits have been received, the MSSPx interrupt flag bit will be set and if enabled, will wake the device.

**TABLE 20-1: 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 |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| SSP1BUF | MSSPx Receive Buffer/Transmit Register | | | | | | | | 201* |
| SSP2BUF | MSSPx Receive Buffer/Transmit Register | | | | | | | | 201* |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 248 |
| SSP2CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 248 |
| SSP1CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 251 |
| SSP2CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 251 |
| SSP1STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 246 |
| SSP2STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 246 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 123 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 126 |

**Legend:** — = Unimplemented location, read as '0'. Shaded cells are not used by the MSSPx in SPI mode.

* Page provides register information.

## 20.3 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 (SCLx)
- Serial Data (SDAx)

Figure 20-2 and Figure 20-3 shows the block diagram of the MSSPx module when operating in I²C mode.

Both the SCLx and SDAx 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 20-11 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:

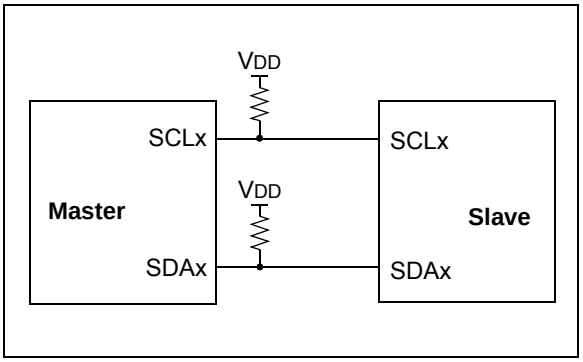
- Master Transmit mode
  (master is transmitting data to a slave)
- Master Receive mode
  (master is receiving data from a slave)
- Slave Transmit mode
  (slave is transmitting data to a master)
- Slave Receive mode
  (slave is receiving data from the master)

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 $\overline{ACK}$. 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 SDAx line while the SCLx 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 20-11: I²C MASTER/ SLAVE CONNECTION**




The Acknowledge bit ($\overline{ACK}$) is an active-low signal, which holds the SDAx 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 SCLx line is held low. Transitions that occur while the SCLx 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 $\overline{ACK}$ 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 $\overline{ACK}$ 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 $\overline{ACK}$ bit. A Stop bit is indicated by a low-to-high transition of the SDAx line while the SCLx 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 $\overline{ACK}$ 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.

 © 2015-2016 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 SCLx 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 SDAx line, it is called arbitration. Arbitration ensures that there is only one master device communicating at any single time.

### 20.3.1 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 SCLx 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 SCLx line in order to transfer the next bit, but will detect that the clock line has not yet been released. Because the SCLx 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.

### 20.3.2 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 SDAx 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 SDAx line.

For example, if one transmitter holds the SDAx 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 SDAx 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 SDAx line. If this transmitter is also a master device, it also must stop driving the SCLx 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 SDAx 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.

## 20.4 I²C MODE OPERATION

All MSSPx I²C communication is byte oriented and shifted out MSb first. Six SFR registers and two interrupt flags interface the module with the PIC® microcontroller and user software. Two pins, SDAx and SCLx, are exercised by the module to communicate with other external I²C devices.

### 20.4.1 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 eighth falling edge of the SCLx line, the device outputting data on the SDAx changes that pin to an input and reads in an acknowledge value on the next clock pulse.

The clock signal, SCLx, is provided by the master. Data is valid to change while the SCLx signal is low, and sampled on the rising edge of the clock. Changes on the SDAx line while the SCLx line is high define special conditions on the bus, explained below.

### 20.4.2 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.

### 20.4.3 SDAX AND SCLX PINS

Selection of any I²C mode with the SSPEN bit set, forces the SCLx and SDAx 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.

### 20.4.4 SDAX HOLD TIME

The hold time of the SDAx pin is selected by the SDAHT bit of the SSPxCON3 register. Hold time is the time SDAx is held valid after the falling edge of SCLx. Setting the SDAHT bit selects a longer 300 ns minimum hold time and may help on buses with large capacitance.

**TABLE 20-2: 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 SDAx and SCLx lines are high. |
| Active | Any time one or more master devices are controlling the bus. |
| Addressed Slave | Slave device that has received a matching address and is actively being clocked by a master. |
| Matching Address | Address byte that is clocked into a slave that matches the value stored in SSPxADD. |
| Write Request | Slave receives a matching address with R/$\overline{W}$ 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 SCLx low to stall communication. |
| Bus Collision | Any time the SDAx line is sampled low by the module while it is outputting and expected high state. |

 © 2015-2016 Microchip Technology Inc.

### 20.4.5 START CONDITION

The $I^2C$ specification defines a Start condition as a transition of SDAx from a high to a low state while SCLx 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 20-12 shows wave forms for Start and Stop conditions.

A bus collision can occur on a Start condition if the module samples the SDAx line low before asserting it low. This does not conform to the $I^2C$ Specification that states no bus collision can occur on a Start.

### 20.4.6 STOP CONDITION

A Stop condition is a transition of the SDAx line from low-to-high state while the SCLx line is high.

**Note:** At least one SCLx low time must appear before a Stop is valid, therefore, if the SDAx line goes low then high again while the SCLx line stays high, only the Start condition is detected.

### 20.4.7 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. Figure 20-13 shows the wave form for a Restart condition.

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/$\overline{W}$ bit set. The slave logic will then hold the clock and prepare to clock out data.

After a full match with R/$\overline{W}$ clear in 10-bit mode, a prior match flag is set and maintained. Until a Stop condition, a high address with R/$\overline{W}$ clear, or high address match fails.

### 20.4.8 START/STOP CONDITION INTERRUPT MASKING

The SCIE and PCIE bits of the SSPxCON3 register can enable the generation of an interrupt in Slave modes that do not typically support this function. Slave modes where interrupt on Start and Stop detect are already enabled, these bits will have no effect.

**FIGURE 20-12: $I^2C$ START AND STOP CONDITIONS**

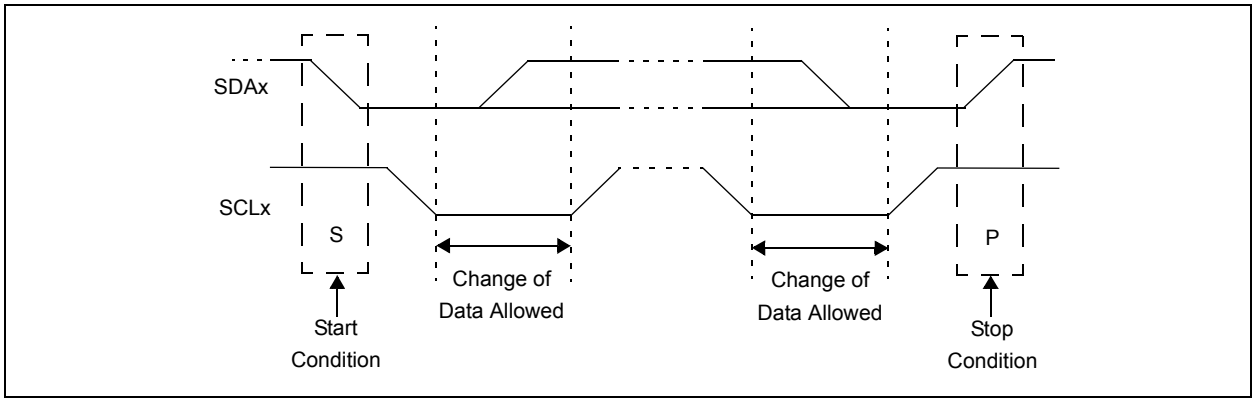
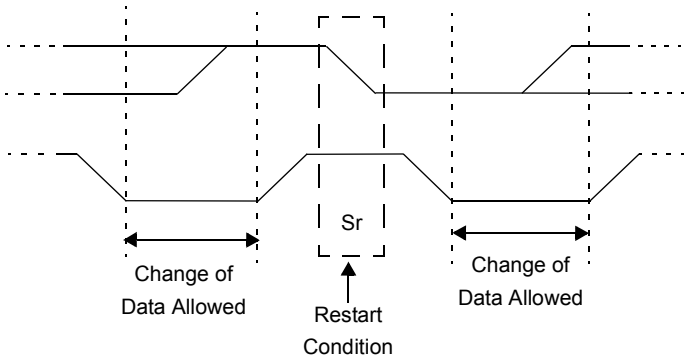


**FIGURE 20-13: $I^2C$ RESTART CONDITION**





© 2015-2016 Microchip Technology Inc.

#### 20.4.9 ACKNOWLEDGE SEQUENCE

The ninth SCLx pulse for any transferred byte in $I^2C$ is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDAx line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge ($\overline{ACK}$) is an active-low signal, pulling the SDAx line low indicated to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an $\overline{ACK}$ is placed in the ACKSTAT bit of the SSPxCON2 register.

Slave software, when the AHEN and DHEN bits are set, allow the user to set the $\overline{ACK}$ value sent back to the transmitter. The ACKDT bit of the SSPxCON2 register is set/cleared to determine the response.

Slave hardware will generate an $\overline{ACK}$ response if the AHEN and DHEN bits of the SSPxCON3 register are clear.

There are certain conditions where an $\overline{ACK}$ will not be sent by the slave. If the BF bit of the SSPxSTAT register or the SSPOV bit of the SSPxCON1 register are set when a byte is received.

When the module is addressed, after the eighth falling edge of SCLx on the bus, the ACKTIM bit of the SSPxCON3 register is set. The ACKTIM bit indicates the acknowledge time of the active bus. The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is enabled.

## 20.5 $I^2C$ SLAVE MODE OPERATION

The MSSPx Slave mode operates in one of four modes selected in the SSPM bits of SSPxCON1 register. The modes can be divided into 7-bit and 10-bit Addressing mode. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

Modes with Start and Stop bit interrupts operated the same as the other modes with SSPxIF additionally getting set upon detection of a Start, Restart, or Stop condition.

#### 20.5.1 SLAVE MODE ADDRESSES

The SSPxADD register (Register 20-6) contains the Slave mode address. The first byte received after a Start or Restart condition is compared against the value stored in this register. If the byte matches, the value is loaded into the SSPxBUF register and an interrupt is generated. If the value does not match, the module goes idle and no indication is given to the software that anything happened.

The SSPx Mask register (Register 20-5) affects the address matching process. See **Section 20.5.9 "SSPx Mask Register"** for more information.

##### 20.5.1.1 $I^2C$ 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.

##### 20.5.1.2 $I^2C$ 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's of the 10-bit address and stored in bits 2 and 1 of the SSPxADD register.

After the acknowledge of the high byte the UA bit is set and SCLx is held low until the user updates SSPxADD with the low address. The low address byte is clocked in and all eight bits are compared to the low address value in SSPxADD. Even if there is not an address match; SSPxIF and UA are set, and SCLx is held low until SSPxADD is updated to receive a high byte again. When SSPxADD is updated the UA bit is cleared. This ensures the module is ready to receive the high address byte on the next communication.

A high and low address match as a write request is required at the start of all 10-bit addressing communication. A transmission can be initiated by issuing a Restart once the slave is addressed, and clocking in the high address with the R/$\overline{W}$ 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.

### 20.5.2 SLAVE RECEPTION

When the R/$\overline{W}$ bit of a matching received address byte is clear, the R/$\overline{W}$ bit of the SSPxSTAT register is cleared. The received address is loaded into the SSPxBUF register and acknowledged.

When the overflow condition exists for a received address, then not Acknowledge is given. An overflow condition is defined as either bit BF of the SSPxSTAT register is set, or bit SSPOV of the SSPxCON1 register is set. The BOEN bit of the SSPxCON3 register modifies this operation. For more information see Register 20-4.

An MSSPx interrupt is generated for each transferred data byte. Flag bit, SSPxIF, must be cleared by software.

When the SEN bit of the SSPxCON2 register is set, SCLx will be held low (clock stretch) following each received byte. The clock must be released by setting the CKP bit of the SSPxCON1 register, except sometimes in 10-bit mode. See **Section 20.2.3 "SPI Master Mode"** for more detail.

#### 20.5.2.1 7-bit Addressing Reception

This section describes a standard sequence of events for the MSSPx module configured as an $I^2C$ slave in 7-bit Addressing mode. Figure 20-14 and Figure 20-15 is used as a visual reference for this description.

This is a step by step process of what typically must be done to accomplish $I^2C$ communication.

1. Start bit detected.
2. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{W}$ bit clear is received.
4. The slave pulls SDAx low sending an $\overline{ACK}$ to the master, and sets SSPxIF bit.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. If SEN = 1; Slave software sets CKP bit to release the SCLx line.
8. The master clocks out a data byte.
9. Slave drives SDAx low sending an $\overline{ACK}$ to the master, and sets SSPxIF bit.
10. Software clears SSPxIF.
11. Software reads the received byte from SSPxBUF clearing BF.
12. Steps 8-12 are repeated for all received bytes from the master.
13. Master sends Stop condition, setting P bit of SSPxSTAT, and the bus goes idle.

#### 20.5.2.2 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 eighth falling edge of SCLx. These additional interrupts allow the slave software to decide whether it wants to $\overline{ACK}$ 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 $I^2C$ communication. Figure 20-16 displays a module using both address and data holding. Figure 20-17 includes the operation with the SEN bit of the SSPxCON2 register set.

1. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
2. Matching address with R/$\overline{W}$ bit clear is clocked in. SSPxIF is set and CKP cleared after the eighth falling edge of SCLx.
3. Slave clears the SSPxIF.
4. Slave can look at the ACKTIM bit of the SSPxCON3 register to determine if the SSPxIF was after or before the $\overline{ACK}$.
5. Slave reads the address value from SSPxBUF, clearing the BF flag.
6. Slave sets $\overline{ACK}$ value clocked out to the master by setting ACKDT.
7. Slave releases the clock by setting CKP.
8. SSPxIF is set after an $\overline{ACK}$, not after a NACK.
9. If SEN = 1 the slave hardware will stretch the clock after the $\overline{ACK}$.
10. Slave clears SSPxIF.

**Note:** SSPxIF is still set after the ninth falling edge of SCLx even if there is no clock stretching and BF has been cleared. Only if NACK is sent to master is SSPxIF not set

11. SSPxIF set and CKP cleared after eighth falling edge of SCLx for a received data byte.
12. Slave looks at ACKTIM bit of SSPxCON3 to determine the source of the interrupt.
13. Slave reads the received data from SSPxBUF clearing BF.
14. Steps 7-14 are the same for each received data byte.
15. Communication is ended by either the slave sending an $\overline{ACK}$ = 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.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 20-14: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 0, DHEN = 0)**

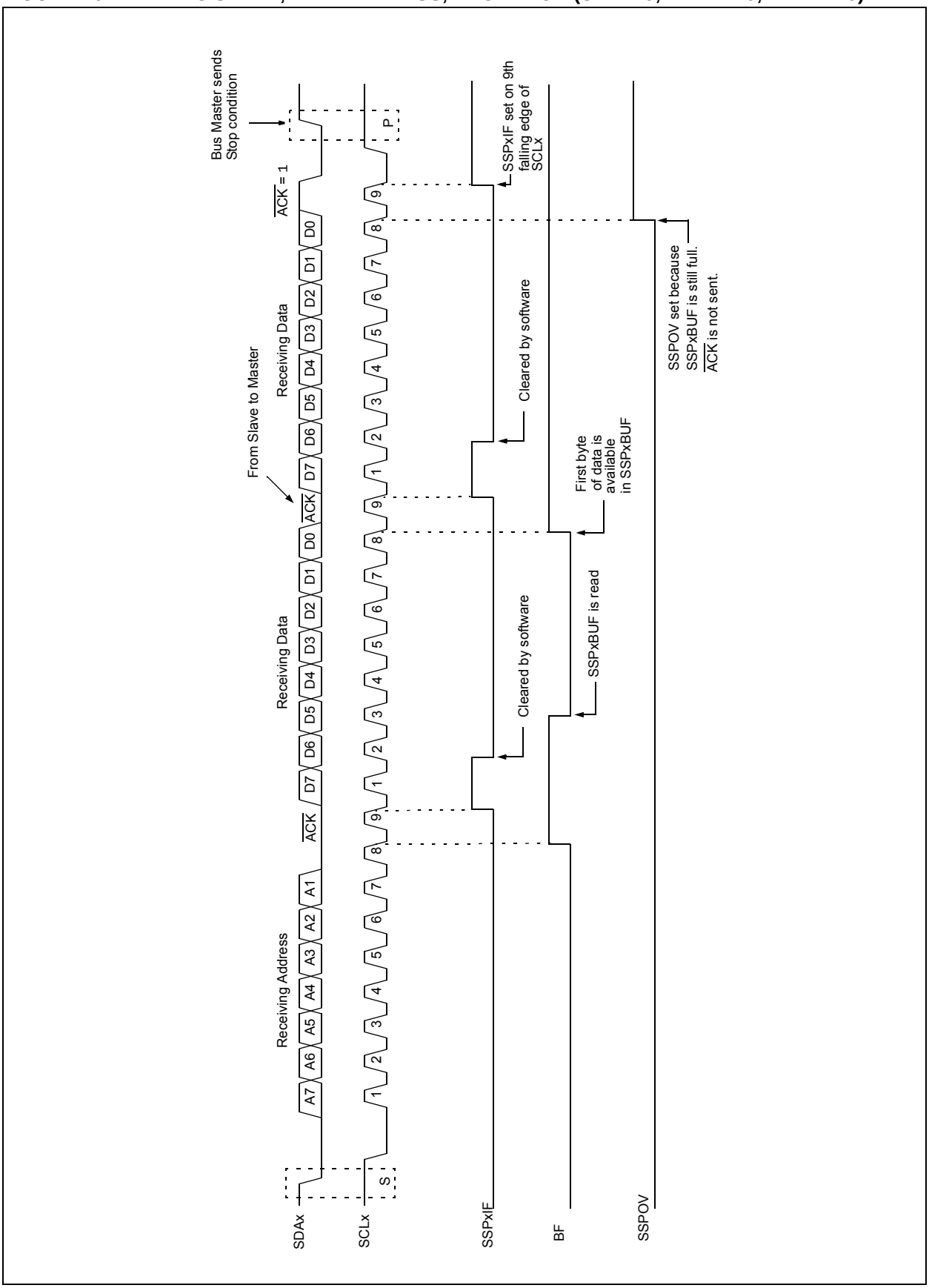
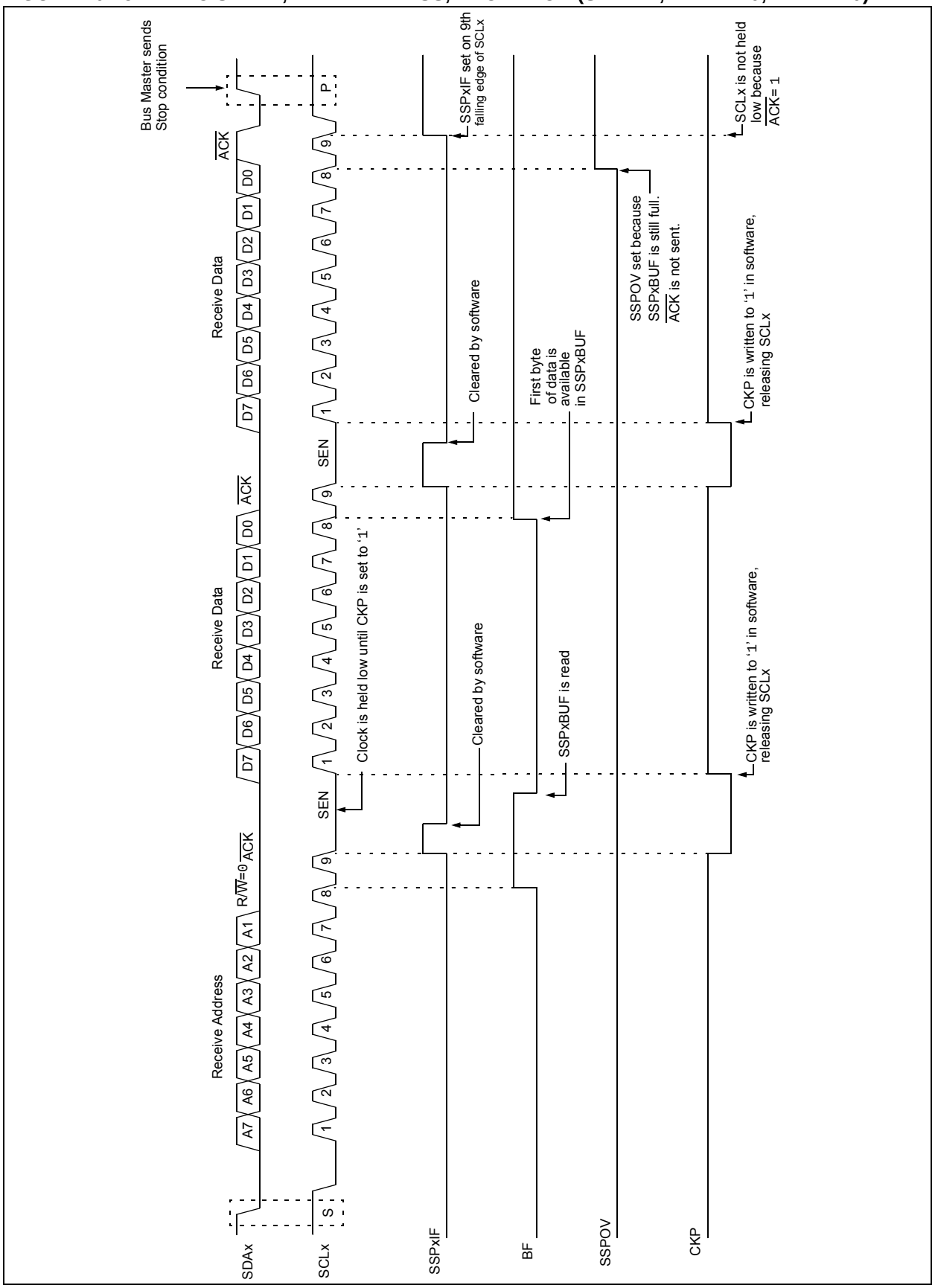


**FIGURE 20-15: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)**




 © 2015-2016 Microchip Technology Inc.

**FIGURE 20-16: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)**

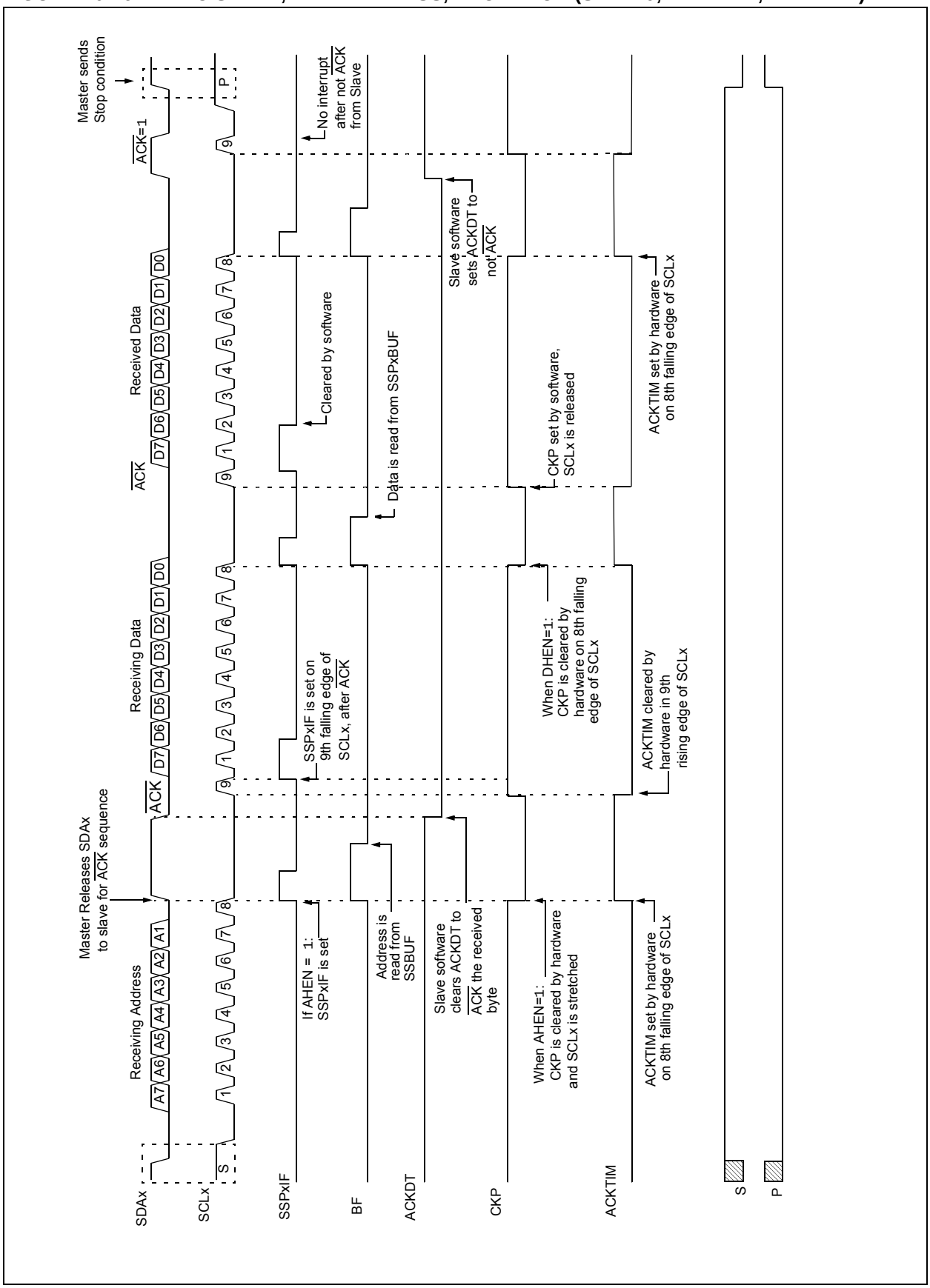
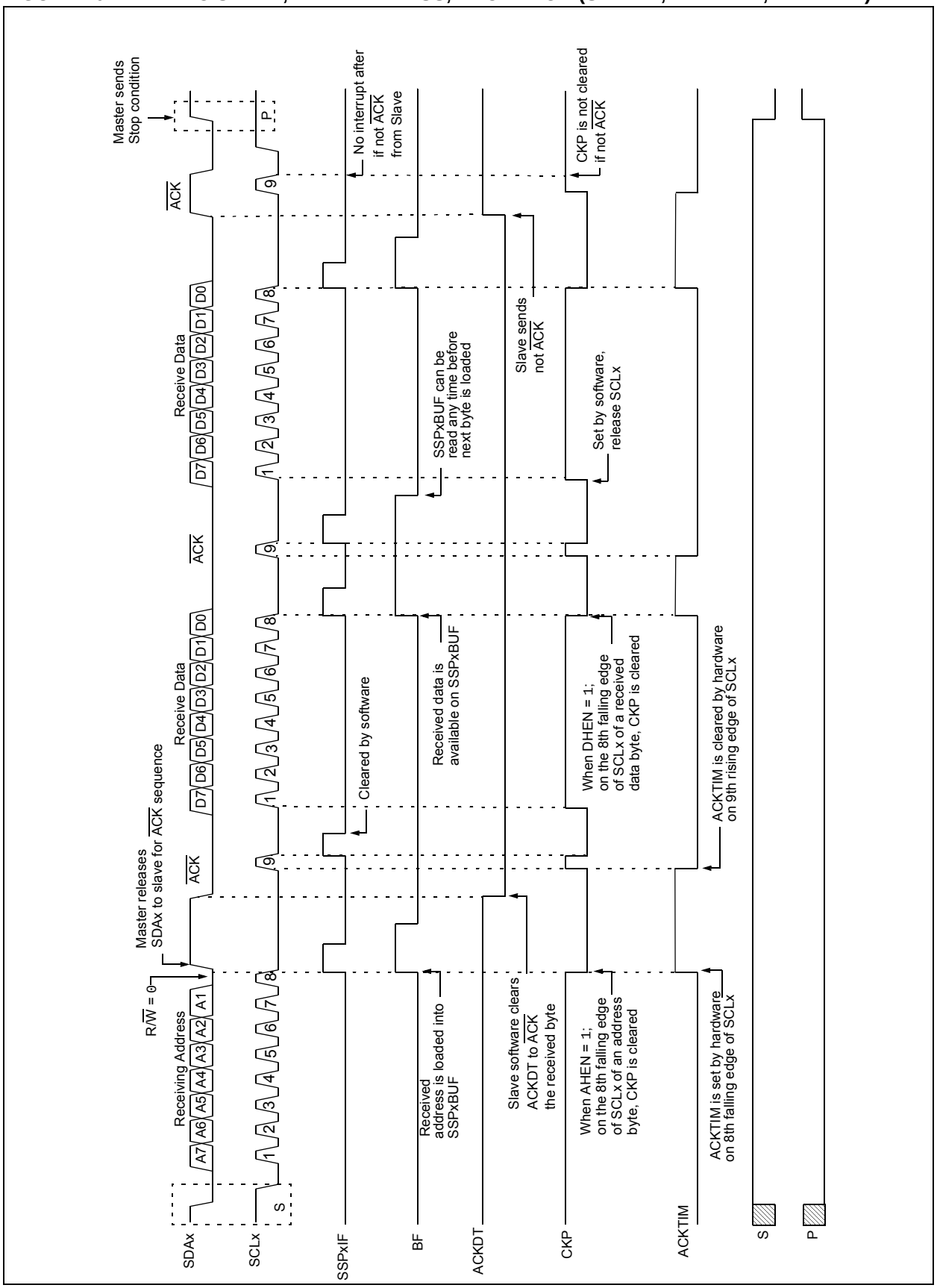


**FIGURE 20-17: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 1, DHEN = 1)**




 © 2015-2016 Microchip Technology Inc.

### 20.5.3 SLAVE TRANSMISSION

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPxSTAT register is set. The received address is loaded into the SSPxBUF register, and an $\overline{ACK}$ pulse is sent by the slave on the ninth bit.

Following the $\overline{ACK}$, slave hardware clears the CKP bit and the SCLx pin is held low (see **Section 20.5.6 "Clock Stretching"** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data.

The transmit data must be loaded into the SSPxBUF register which also loads the SSPxSR register. Then the SCLx pin should be released by setting the CKP bit of the SSPxCON1 register. The eight data bits are shifted out on the falling edge of the SCLx input. This ensures that the SDAx signal is valid during the SCLx high time.

The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCLx input pulse. This $\overline{ACK}$ value is copied to the ACKSTAT bit of the SSPxCON2 register. If ACKSTAT is set (not $\overline{ACK}$), then the data transfer is complete. In this case, when the not $\overline{ACK}$ is latched by the slave, the slave goes idle and waits for another occurrence of the Start bit. If the SDAx line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPxBUF register. Again, the SCLx pin must be released by setting bit CKP.

An MSSPx interrupt is generated for each data transfer byte. The SSPxIF bit must be cleared by software and the SSPxSTAT register is used to determine the status of the byte. The SSPxIF bit is set on the falling edge of the ninth clock pulse.

#### 20.5.3.1 Slave Mode Bus Collision

A slave receives a Read request and begins shifting data out on the SDAx line. If a bus collision is detected and the SBCDE bit of the SSPxCON3 register is set, the BCLxIF bit of the PIRx register is set. Once a bus collision is detected, the slave goes Idle and waits to be addressed again. User software can use the BCLxIF bit to handle a slave bus collision.

#### 20.5.3.2 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 20-18 can be used as a reference to this list.

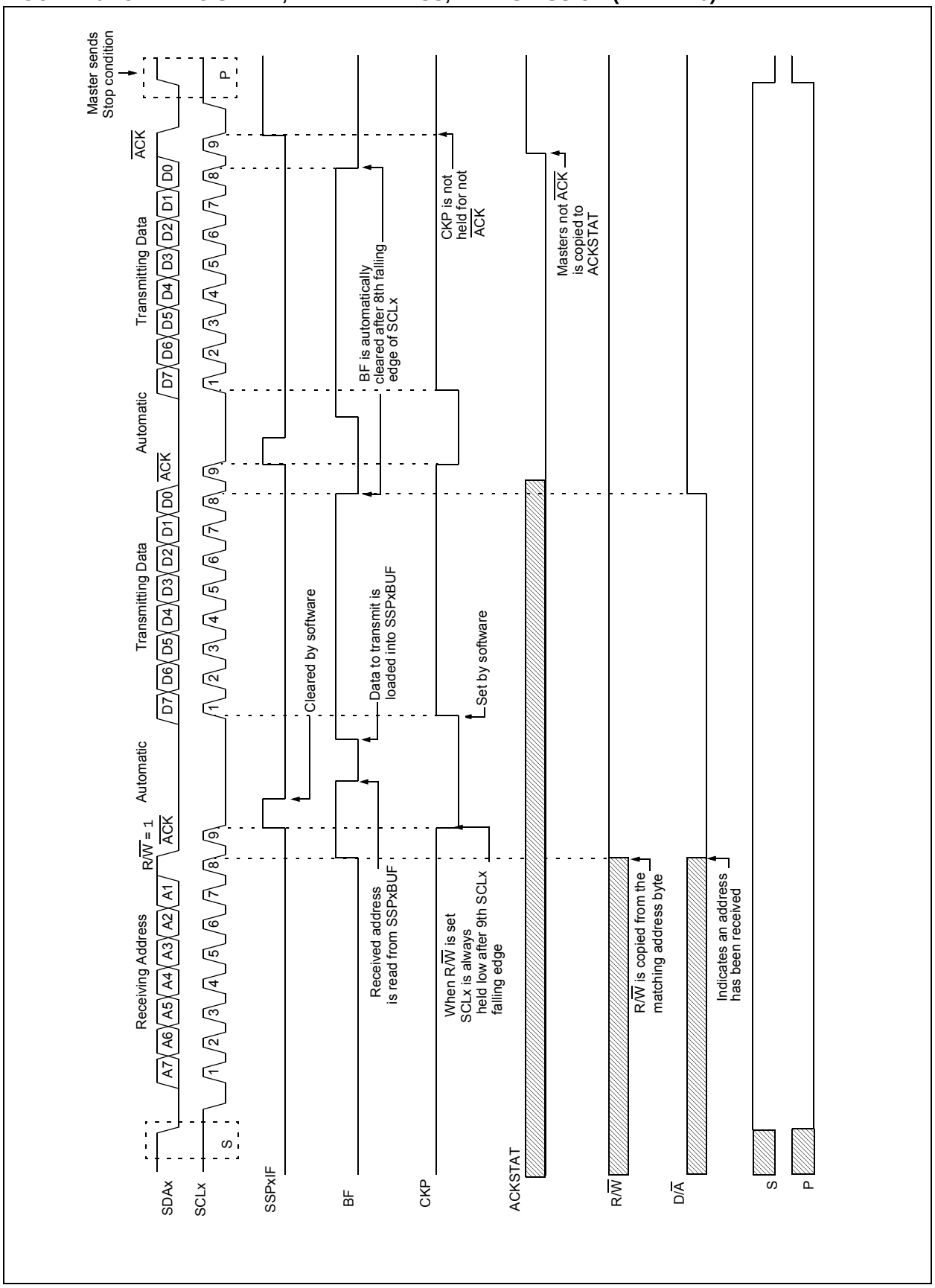
1. Master sends a Start condition on SDAx and SCLx.
2. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{W}$ bit set is received by the slave setting SSPxIF bit.
4. Slave hardware generates an $\overline{ACK}$ and sets SSPxIF.
5. SSPxIF bit is cleared by user.
6. Software reads the received address from SSPxBUF, clearing BF.
7. R/$\overline{W}$ is set so CKP was automatically cleared after the $\overline{ACK}$.
8. The slave software loads the transmit data into SSPxBUF.
9. CKP bit is set releasing SCLx, allowing the master to clock the data out of the slave.
10. SSPxIF is set after the $\overline{ACK}$ response from the master is loaded into the ACKSTAT register.
11. SSPxIF 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 $\overline{ACK}$s the clock will be stretched.

**2:** ACKSTAT is the only bit updated on the rising edge of SCLx (9th) rather than the falling.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not $\overline{ACK}$; the clock is not held, but SSPxIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.

**FIGURE 20-18: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 0)**




 © 2015-2016 Microchip Technology Inc.

#### 20.5.3.3 7-bit Transmission with Address Hold Enabled

Setting the AHEN bit of the SSPxCON3 register enables additional clock stretching and interrupt generation after the eighth falling edge of a received matching address. Once a matching address has been clocked in, CKP is cleared and the SSPxIF interrupt is set.

Figure 20-19 displays a standard waveform of a 7-bit Address Slave Transmission with AHEN enabled.

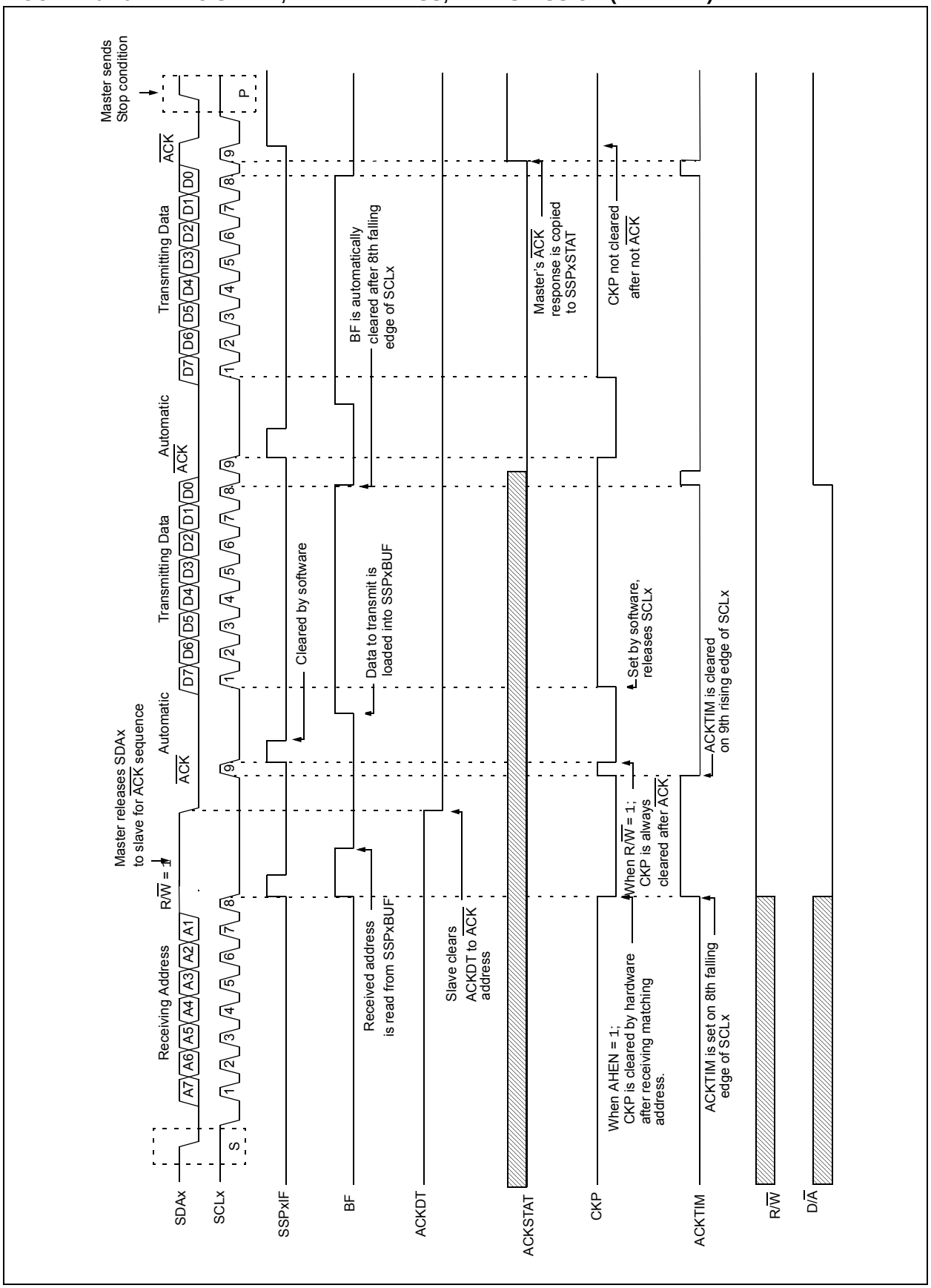
1. Bus starts Idle.
2. Master sends Start condition; the S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Master sends matching address with R/$\overline{W}$ bit set. After the eighth falling edge of the SCLx line the CKP bit is cleared and SSPxIF interrupt is generated.
4. Slave software clears SSPxIF.
5. Slave software reads the ACKTIM bit of SSPxCON3 register, and R/$\overline{W}$ and D/$\overline{A}$ of the SSPxSTAT register to determine the source of the interrupt.
6. Slave reads the address value from the SSPxBUF register clearing the BF bit.
7. Slave software decides from this information if it wishes to $\overline{ACK}$ or not $\overline{ACK}$ and sets the ACKDT bit of the SSPxCON2 register accordingly.
8. Slave sets the CKP bit releasing SCLx.
9. Master clocks in the $\overline{ACK}$ value from the slave.
10. Slave hardware automatically clears the CKP bit and sets SSPxIF after the $\overline{ACK}$ if the R/$\overline{W}$ bit is set.
11. Slave software clears SSPxIF.
12. Slave loads value to transmit to the master into SSPxBUF setting the BF bit.

**Note:** SSPxBUF cannot be loaded until after the $\overline{ACK}$.

13. Slave sets CKP bit releasing the clock.
14. Master clocks out the data from the slave and sends an $\overline{ACK}$ value on the ninth SCLx pulse.
15. Slave hardware copies the $\overline{ACK}$ value into the ACKSTAT bit of the SSPxCON2 register.
16. Steps 10-15 are repeated for each byte transmitted to the master from the slave.
17. If the master sends a not $\overline{ACK}$ the slave releases the bus allowing the master to send a Stop and end the communication.

**Note:** Master must send a not $\overline{ACK}$ on the last byte to ensure that the slave releases the SCLx line to receive a Stop.

**FIGURE 20-19: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)**




 © 2015-2016 Microchip Technology Inc.

### 20.5.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSPx module configured as an $I^2C$ slave in 10-bit Addressing mode.

Figure 20-20 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 $I^2C$ communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit of SSPxSTAT is set; SSPxIF 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 SSPxSTAT register is set.
4. Slave sends $\overline{ACK}$ and SSPxIF is set.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. Slave loads low address into SSPxADD, releasing SCLx.
8. Master sends matching low address byte to the slave; UA bit is set.

| | |
|---|---|
| **Note:** | Updates to the SSPxADD register are not allowed until after the $\overline{ACK}$ sequence. |

9. Slave sends $\overline{ACK}$ and SSPxIF is set.

| | |
|---|---|
| **Note:** | If the low address does not match, SSPxIF and UA are still set so that the slave software can set SSPxADD back to the high address. BF is not set because there is no match. CKP is unaffected. |

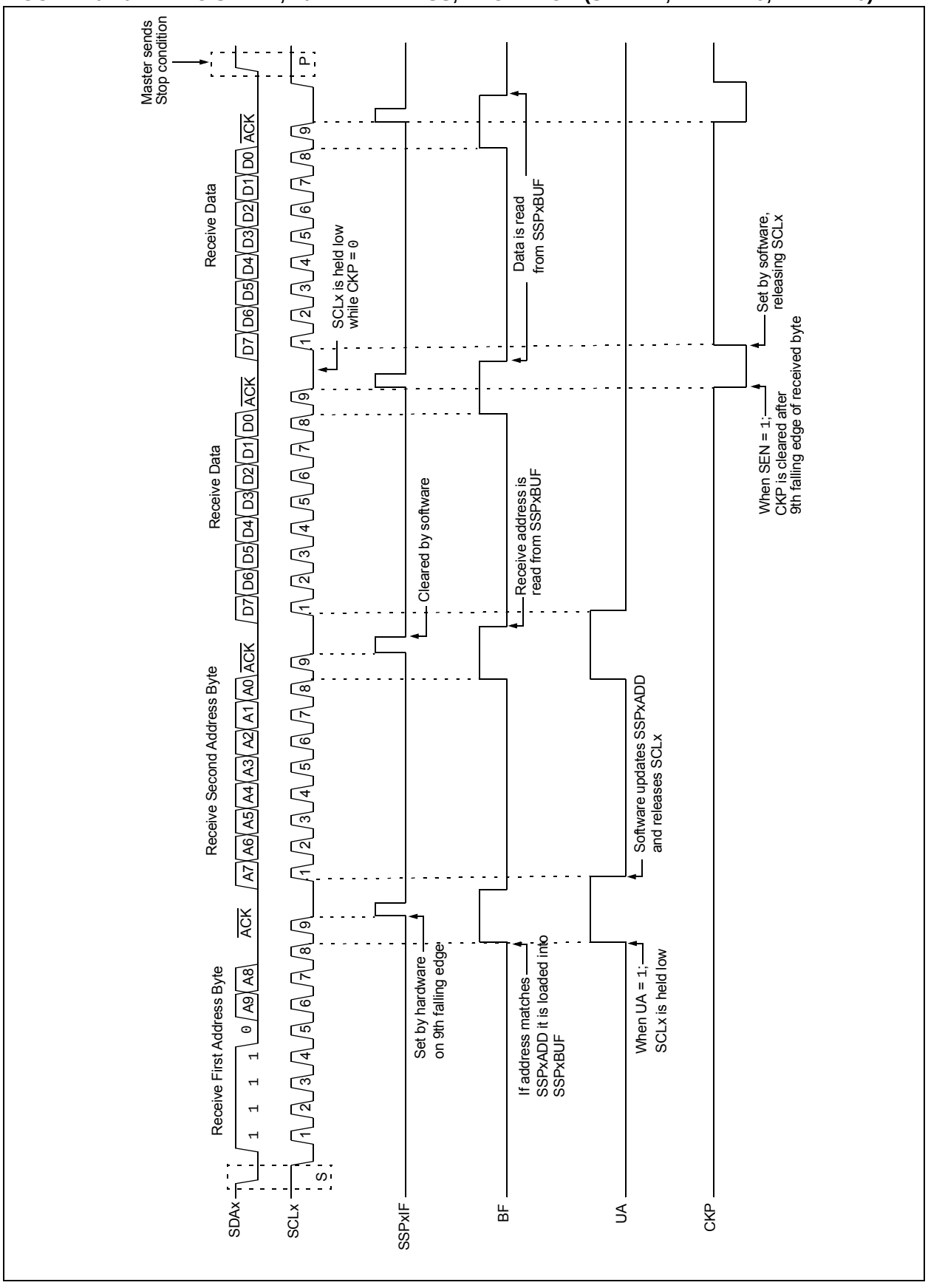
10. Slave clears SSPxIF.
11. Slave reads the received matching address from SSPxBUF clearing BF.
12. Slave loads high address into SSPxADD.
13. Master clocks a data byte to the slave and clocks out the slaves $\overline{ACK}$ on the ninth SCLx pulse; SSPxIF is set.
14. If the SEN bit of SSPxCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPxIF.
16. Slave reads the received byte from SSPxBUF clearing BF.
17. If SEN is set the slave sets CKP to release the SCLx.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

### 20.5.5 10-BIT ADDRESSING WITH ADDRESS OR DATA HOLD

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPxADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCLx line is held low are the same. Figure 20-21 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 20-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

**FIGURE 20-20: I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)**




 © 2015-2016 Microchip Technology Inc.

**FIGURE 20-21: I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 0)**

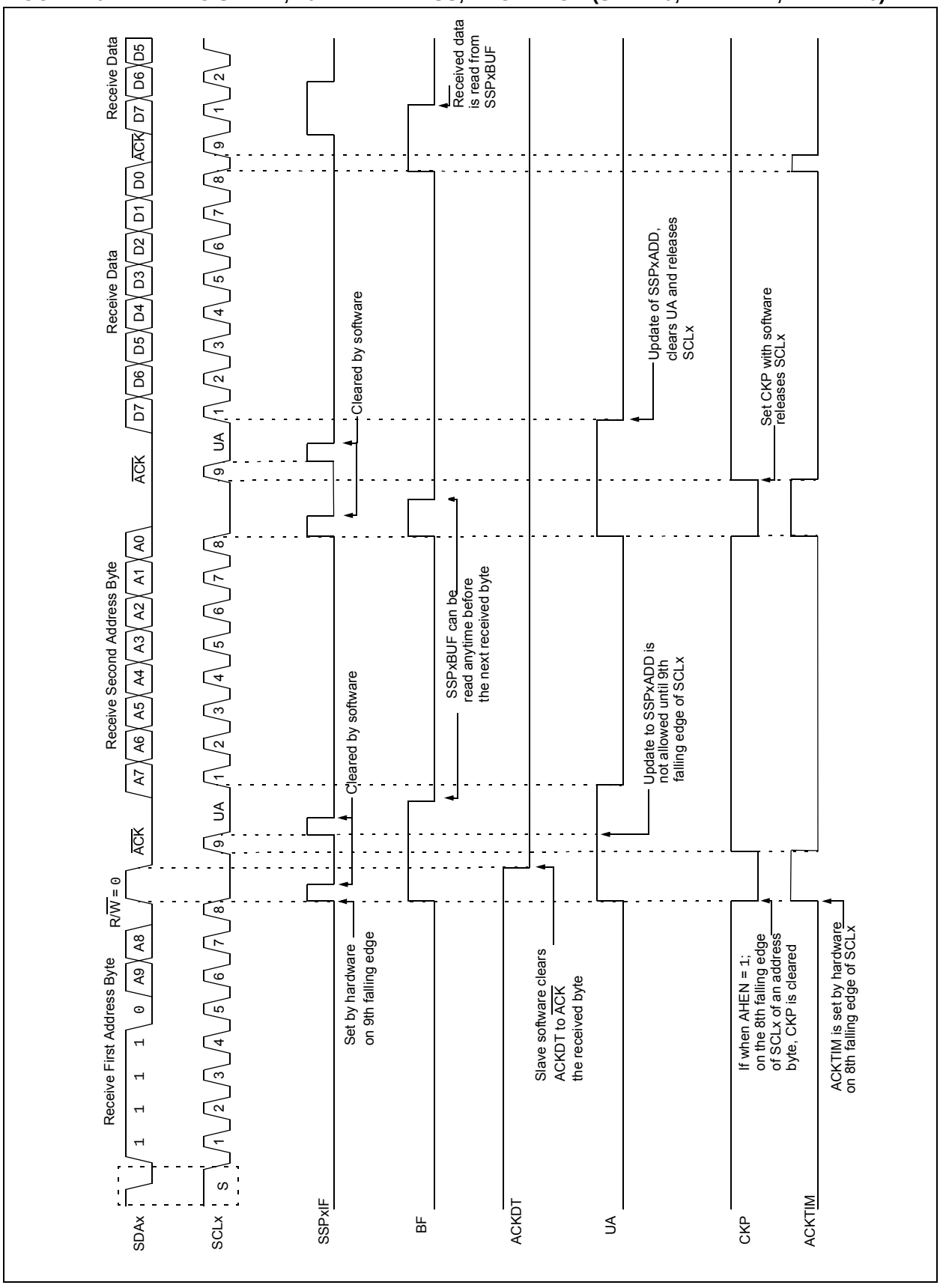
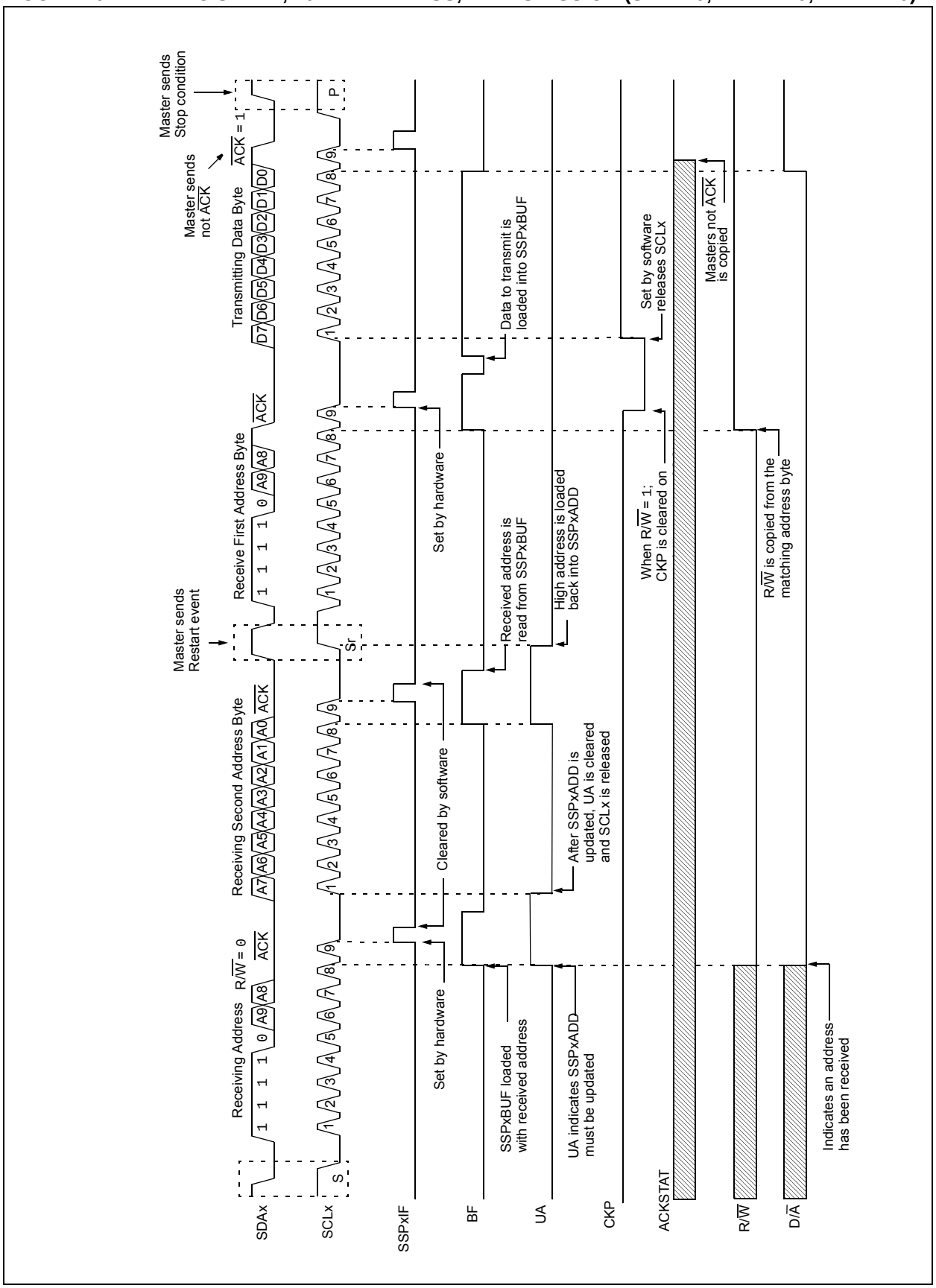


**FIGURE 20-22: I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)**




 © 2015-2016 Microchip Technology Inc.

### 20.5.6 CLOCK STRETCHING

Clock stretching occurs when a device on the bus holds the SCLx 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 SCLx.

The CKP bit of the SSPxCON1 register is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCLx line to go low and then hold it. Setting CKP will release SCLx and allow more communication.

#### 20.5.6.1 Normal Clock Stretching

Following an $\overline{ACK}$ if the R/$\overline{W}$ bit of SSPxSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPxBUF with data to transfer to the master. If the SEN bit of SSPxCON2 is set, the slave hardware will always stretch the clock after the $\overline{ACK}$ 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 SSPxBUF was read before the ninth falling edge of SCLx.

**2:** Previous versions of the module did not stretch the clock for a transmission if SSPxBUF was loaded before the ninth falling edge of SCLx. It is now always cleared for read requests.

#### 20.5.6.2 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 SCLx is stretched without CKP being cleared. SCLx is released immediately after a write to SSPxADD.

**Note:** Previous versions of the module did not stretch the clock if the second address byte did not match.

#### 20.5.6.3 Byte NACKing

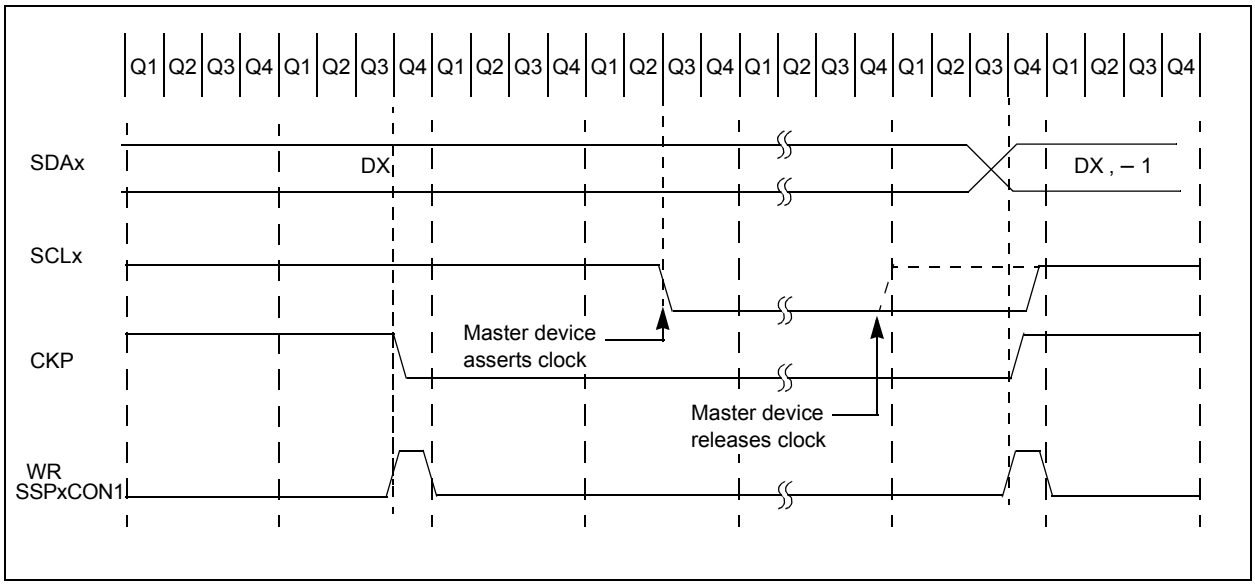
When the AHEN bit of SSPxCON3 is set; CKP is cleared by hardware after the eighth falling edge of SCLx for a received matching address byte. When the DHEN bit of SSPxCON3 is set; CKP is cleared after the eighth falling edge of SCLx for received data.

Stretching after the eighth falling edge of SCLx allows the slave to look at the received address or data and decide if it wants to ACK the received data.

### 20.5.7 CLOCK SYNCHRONIZATION AND THE CKP BIT

Any time the CKP bit is cleared, the module will wait for the SCLx line to go low and then hold it. However, clearing the CKP bit will not assert the SCLx output low until the SCLx output is already sampled low. Therefore, the CKP bit will not assert the SCLx line until an external $I^2C$ master device has already asserted the SCLx line. The SCLx output will remain low until the CKP bit is set and all other devices on the $I^2C$ bus have released SCLx. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCLx (see Figure 20-23).

**FIGURE 20-23: CLOCK SYNCHRONIZATION TIMING**



### 20.5.8 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the $I^2C$ bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master 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 $I^2C$ protocol, defined as address 0x00. When the GCEN bit of the SSPxCON2 register is set, the slave module will automatically $\overline{ACK}$ the reception of this address regardless of the value stored in SSPxADD. 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 SSPxBUF and respond. Figure 20-24 shows a general call reception sequence.

In 10-bit Address mode, the UA bit will not be set on the reception of the general call address. The slave will prepare to receive the second byte as data, just as it would in 7-bit mode.

If the AHEN bit of the SSPxCON3 register is set, just as with any other address reception, the slave hardware will stretch the clock after the eighth falling edge of SCLx. The slave must then set its ACKDT value and release the clock with communication progressing as it would normally.

### 20.5.9 SSPX MASK REGISTER

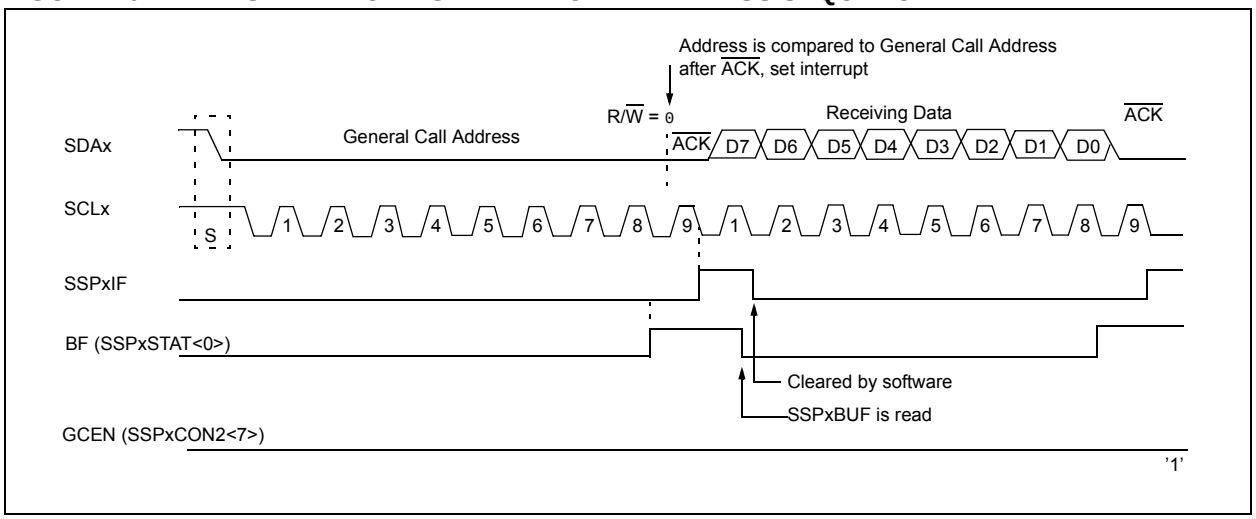
An SSPx Mask (SSPxMSK) register (Register 20-5) is available in $I^2C$ Slave mode as a mask for the value held in the SSPxSR register during an address comparison operation. A zero ('0') bit in the SSPxMSK register has the effect of making the corresponding bit of the received address a "don't care".

This register is reset to all '1's upon any Reset condition and, therefore, has no effect on standard SSPx operation until written with a mask value.

The SSPx Mask register is active during:

- 7-bit Address mode: address compare of A<7:1>.
- 10-bit Address mode: address compare of A<7:0> only. The SSPx mask has no effect during the reception of the first (high) byte of the address.

**FIGURE 20-24: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE**




 © 2015-2016 Microchip Technology Inc.

## 20.6 I²C MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in the SSPxCON1 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 MSSPx 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 SDAx and SCLx lines.

The following events will cause the SSPx Interrupt Flag bit, SSPxIF, to be set (SSPx interrupt, if enabled):

- Start condition detected
- Stop condition detected
- Data transfer byte transmitted/received
- Acknowledge transmitted/received
- Repeated Start generated

**Note 1:** The MSSPx module, when configured in I²C Master mode, does not allow queuing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur

**2:** Master mode suspends Start/Stop detection when sending the Start/Stop condition by means of the SEN/PEN control bits. The SSPIF bit is set at the end of the Start/Stop generation when hardware clears the control bit.

### 20.6.1 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 SDAx, while SCLx 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 eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

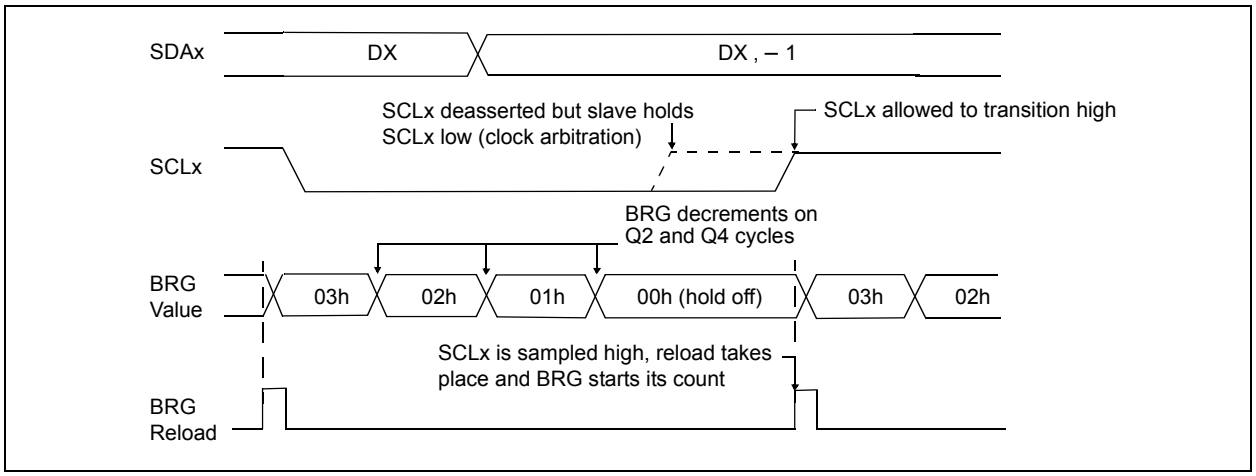
In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive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

A Baud Rate Generator is used to set the clock frequency output on SCLx. See **Section 20.7 "Baud Rate Generator"** for more detail.

### 20.6.2 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, releases the SCLx pin (SCLx allowed to float high). When the SCLx pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCLx pin is actually sampled high. When the SCLx pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and begins counting. This ensures that the SCLx 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 20-25).

**FIGURE 20-25: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**




### 20.6.3 WCOL STATUS FLAG

If the user writes the SSPxBUF when a Start, Restart, Stop, Receive or Transmit sequence is in progress, the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur). Any time the WCOL bit is set it indicates that an action on SSPxBUF was attempted while the module was not Idle.

| **Note:** | Because queuing of events is not allowed, writing to the lower five bits of SSPxCON2 is disabled until the Start condition is complete. |
|---|---|

 © 2015-2016 Microchip Technology Inc.

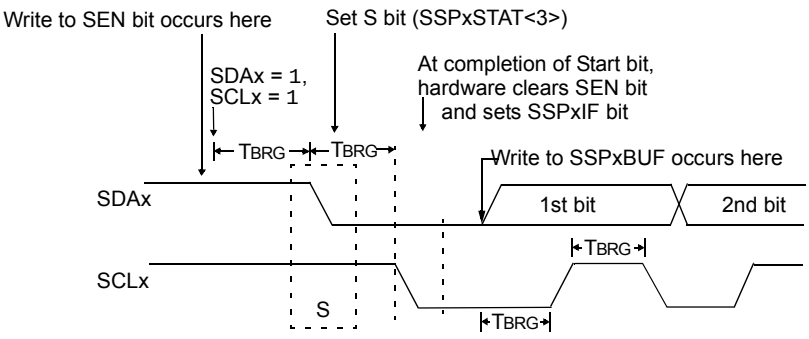
### 20.6.4 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition (Figure 20-26), the user sets the Start Enable bit, SEN bit of the SSPxCON2 register. If the SDAx and SCLx pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and starts its count. If SCLx and SDAx are both sampled high when the Baud Rate Generator times out (TBRG), the SDAx pin is driven low. The action of the SDAx being driven low while SCLx is high is the Start condition and causes the S bit of the SSPxSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit of the SSPxCON2 register will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDAx line held low and the Start condition is complete.

**Note 1:** If at the beginning of the Start condition, the SDAx and SCLx pins are already sampled low, or if during the Start condition, the SCLx line is sampled low before the SDAx line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLxIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state.

**2:** The Philips I²C Specification states that a bus collision cannot occur on a Start.

**FIGURE 20-26: FIRST START BIT TIMING**



### 20.6.5 I²C MASTER MODE REPEATED START CONDITION TIMING

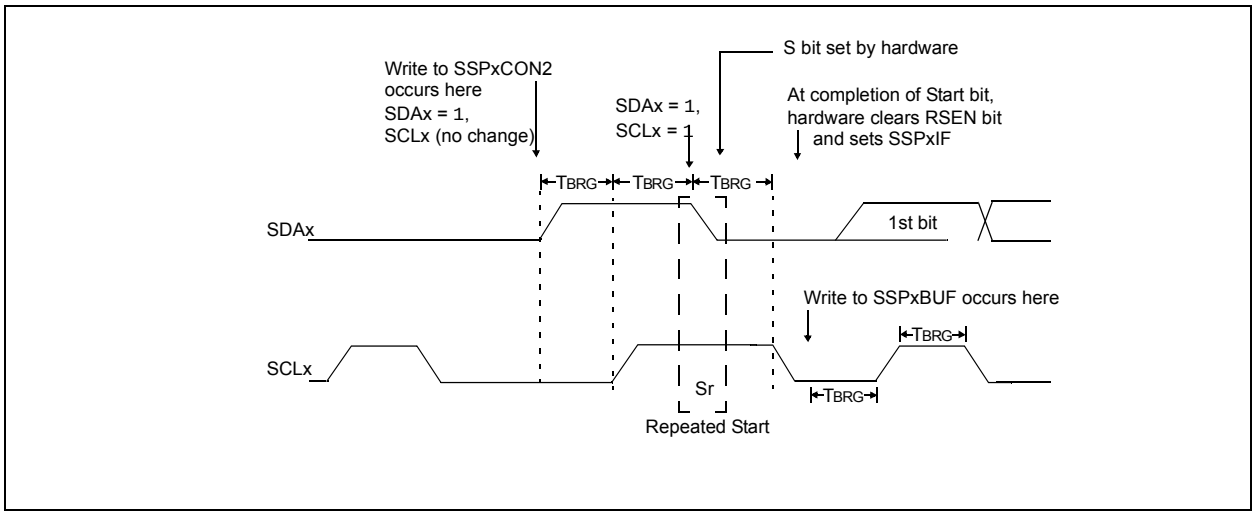
A Repeated Start condition occurs when the RSEN bit of the SSPxCON2 register is programmed high and the master state machine is no longer active. When the RSEN bit is set, the SCLx pin is asserted low. When the SCLx pin is sampled low, the Baud Rate Generator is loaded and begins counting. The SDAx pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDAx is sampled high, the SCLx pin will be deasserted (brought high). When SCLx is sampled high, the Baud Rate Generator is reloaded and begins counting. SDAx and SCLx must be sampled high for one TBRG. This action is then followed by assertion of the SDAx pin (SDAx = 0) for one TBRG while SCLx is high. SCLx is asserted low. Following this, the RSEN bit of the SSPxCON2 register will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDAx pin held low. As soon as a Start condition is detected on the SDAx and SCLx pins, the S bit of the SSPxSTAT register will be set. The SSPxIF 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:

- SDAx is sampled low when SCLx goes from low-to-high.
- SCLx goes low before SDAx is asserted low. This may indicate that another master is attempting to transmit a data '1'.

**FIGURE 20-27: REPEAT START CONDITION WAVEFORM**




 © 2015-2016 Microchip Technology Inc.

### 20.6.6 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPxBUF register. This action will set the Buffer Full flag bit, BF and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDAx pin after the falling edge of SCLx is asserted. SCLx is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCLx is released high. When the SCLx pin is released high, it is held that way for TBRG. The data on the SDAx pin must remain stable for that duration and some hold time after the next falling edge of SCLx. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDAx. 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 SSPxIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPxBUF, leaving SCLx low and SDAx unchanged (Figure 20-28).

After the write to the SSPxBUF, each bit of the address will be shifted out on the falling edge of SCLx 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 SDAx pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDAx 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 SSPxCON2 register. Following the falling edge of the ninth clock transmission of the address, the SSPxIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPxBUF takes place, holding SCLx low and allowing SDAx to float.

#### 20.6.6.1 BF Status Flag

In Transmit mode, the BF bit of the SSPxSTAT register is set when the CPU writes to SSPxBUF and is cleared when all eight bits are shifted out.

#### 20.6.6.2 WCOL Status Flag

If the user writes the SSPxBUF when a transmit is already in progress (i.e., SSPxSR is still shifting out a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

The WCOL bit must be cleared by software before the next transmission.

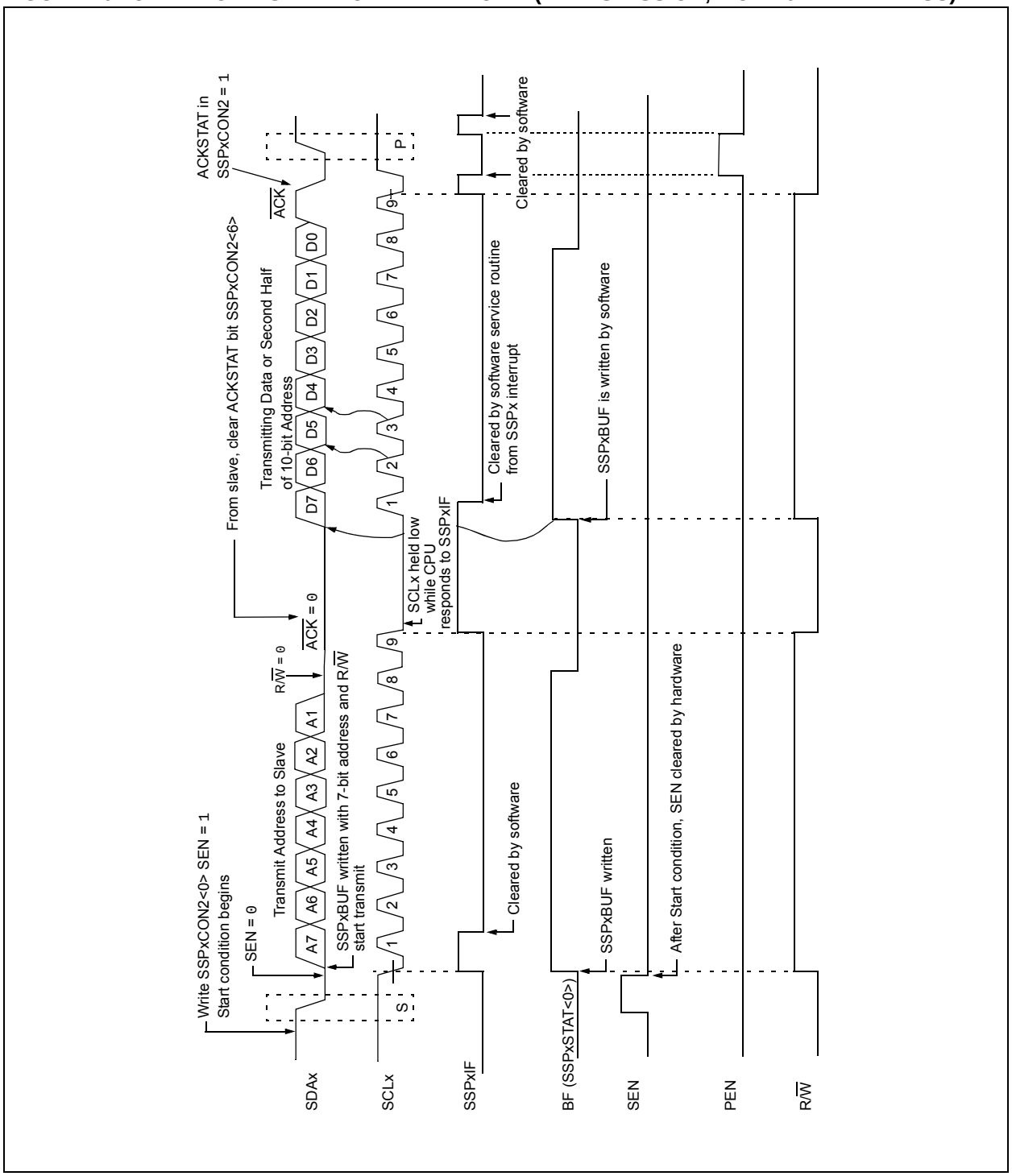
#### 20.6.6.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit of the SSPxCON2 register is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 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.

#### 20.6.6.4 Typical transmit sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPxCON2 register.
2. SSPxIF is set by hardware on completion of the Start.
3. SSPxIF is cleared by software.
4. The MSSPx module will wait the required start time before any other operation takes place.
5. The user loads the SSPxBUF with the slave address to transmit.
6. Address is shifted out the SDAx pin until all eight bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
7. The MSSPx module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
8. The MSSPx module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
9. The user loads the SSPxBUF with eight bits of data.
10. Data is shifted out the SDAx pin until all eight bits are transmitted.
11. The MSSPx module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
12. Steps 8-11 are repeated for all transmitted data bytes.
13. The user generates a Stop or Restart condition by setting the PEN or RSEN bits of the SSPxCON2 register. Interrupt is generated once the Stop/Restart condition is complete.

**FIGURE 20-28: $I^2C$ MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)**




 © 2015-2016 Microchip Technology Inc.

### 20.6.7 I²C MASTER MODE RECEPTION

Master mode reception (Figure 20-29) is enabled by programming the Receive Enable bit, RCEN bit of the SSPxCON2 register.

| | |
|---|---|
| **Note:** | The MSSPx 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 SCLx pin changes (high-to-low/low-to-high) and data is shifted into the SSPxSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPxSR are loaded into the SSPxBUF, the BF flag bit is set, the SSPxIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCLx low. The MSSPx 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 SSPxCON2 register.

#### 20.6.7.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPxBUF from SSPxSR. It is cleared when the SSPxBUF register is read.

#### 20.6.7.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when eight bits are received into the SSPxSR and the BF flag bit is already set from a previous reception.

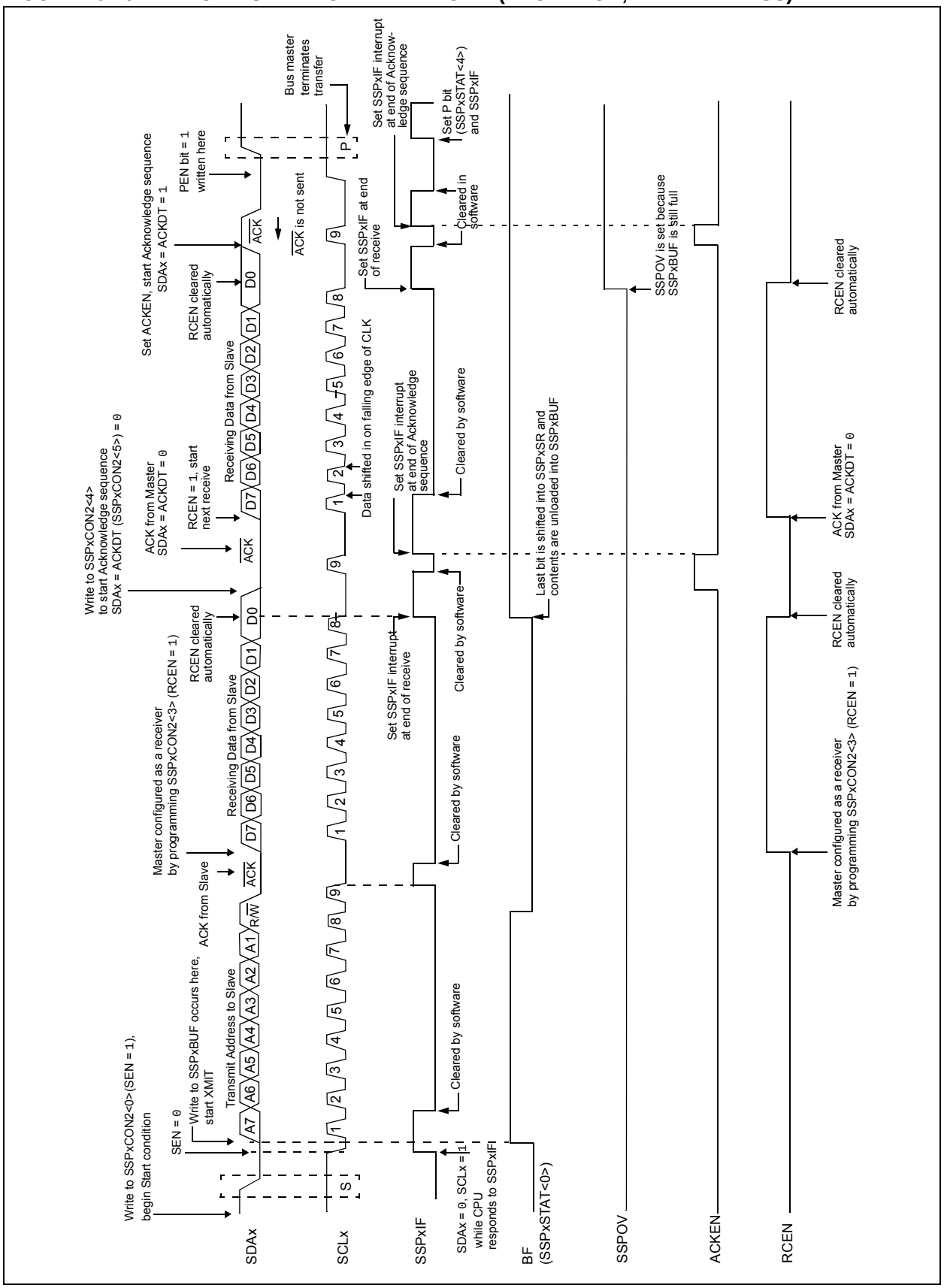
#### 20.6.7.3 WCOL Status Flag

If the user writes the SSPxBUF when a receive is already in progress (i.e., SSPxSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

#### 20.6.7.4 Typical Receive Sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPxCON2 register.
2. SSPxIF is set by hardware on completion of the Start.
3. SSPxIF is cleared by software.
4. User writes SSPxBUF with the slave address to transmit and the R/$\overline{W}$ bit set.
5. Address is shifted out the SDAx pin until all eight bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
6. The MSSPx module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
7. The MSSPx module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
8. User sets the RCEN bit of the SSPxCON2 register and the master clocks in a byte from the slave.
9. After the eighth falling edge of SCLx, SSPxIF and BF are set.
10. Master clears SSPxIF and reads the received byte from SSPxUF, clears BF.
11. Master sets $\overline{ACK}$ value sent to slave in ACKDT bit of the SSPxCON2 register and initiates the $\overline{ACK}$ by setting the ACKEN bit.
12. Masters $\overline{ACK}$ is clocked out to the slave and SSPxIF is set.
13. User clears SSPxIF.
14. Steps 8-13 are repeated for each received byte from the slave.
15. Master sends a not $\overline{ACK}$ or Stop to end communication.

**FIGURE 20-29: I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)**




 © 2015-2016 Microchip Technology Inc.

### 20.6.8 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN bit of the SSPxCON2 register. When this bit is set, the SCLx pin is pulled low and the contents of the Acknowledge data bit are presented on the SDAx pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCLx pin is deasserted (pulled high). When the SCLx pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCLx pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSPx module then goes into Idle mode (Figure 20-30).

#### 20.6.8.1 WCOL Status Flag

If the user writes the SSPxBUF when an Acknowledge sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

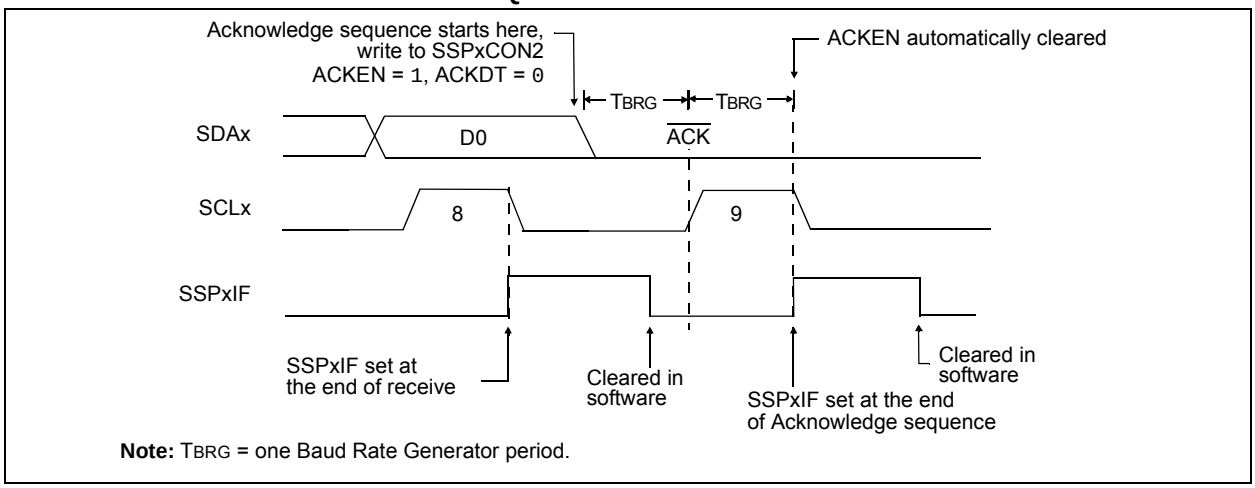
### 20.6.9 STOP CONDITION TIMING

A Stop bit is asserted on the SDAx pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN bit of the SSPxCON2 register. At the end of a receive/transmit, the SCLx line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDAx line low. When the SDAx line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCLx pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDAx pin will be deasserted. When the SDAx pin is sampled high while SCLx is high, the P bit of the SSPxSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPxIF bit is set (Figure 20-31).

#### 20.6.9.1 WCOL Status Flag

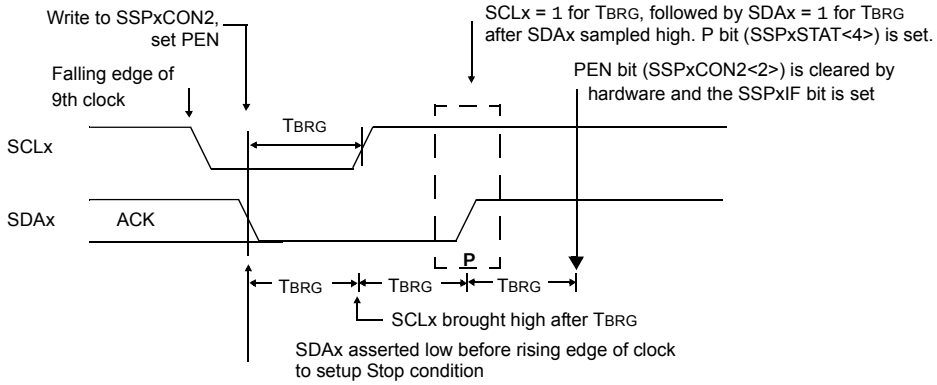
If the user writes the SSPxBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

**FIGURE 20-30: ACKNOWLEDGE SEQUENCE WAVEFORM**




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

**FIGURE 20-31: STOP CONDITION RECEIVE OR TRANSMIT MODE**




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

### 20.6.10 SLEEP OPERATION

While in Sleep mode, the $I^2C$ slave module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSPx interrupt is enabled).

### 20.6.11 EFFECTS OF A RESET

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

### 20.6.12 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 MSSPx module is disabled. Control of the $I^2C$ bus may be taken when the P bit of the SSPxSTAT register is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the SSPx interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDAx 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 BCLxIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

### 20.6.13 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 SDAx pin, arbitration takes place when the master outputs a '1' on SDAx, by letting SDAx float high and another master asserts a '0'. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a '1' and the data sampled on the SDAx pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF and reset the $I^2C$ port to its Idle state (Figure 20-32).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDAx and SCLx lines are deasserted and the SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the $I^2C$ 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 SDAx and SCLx lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the $I^2C$ bus is free, the user can resume communication by asserting a Start condition.

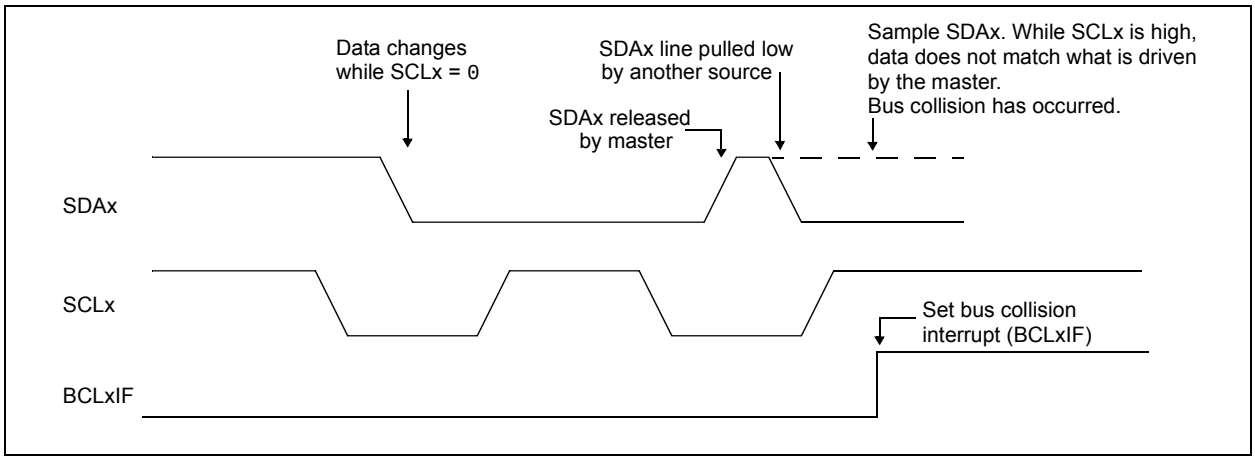
The master will continue to monitor the SDAx and SCLx pins. If a Stop condition occurs, the SSPxIF bit will be set.

A write to the SSPxBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the $I^2C$ bus can be taken when the P bit is set in the SSPxSTAT register, or the bus is Idle and the S and P bits are cleared.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 20-32: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**



#### 20.6.13.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDAx or SCLx are sampled low at the beginning of the Start condition (Figure 20-33).

b) SCLx is sampled low before SDAx is asserted low (Figure 20-34).

During a Start condition, both the SDAx and the SCLx pins are monitored.

If the SDAx pin is already low, or the SCLx pin is already low, then all of the following occur:

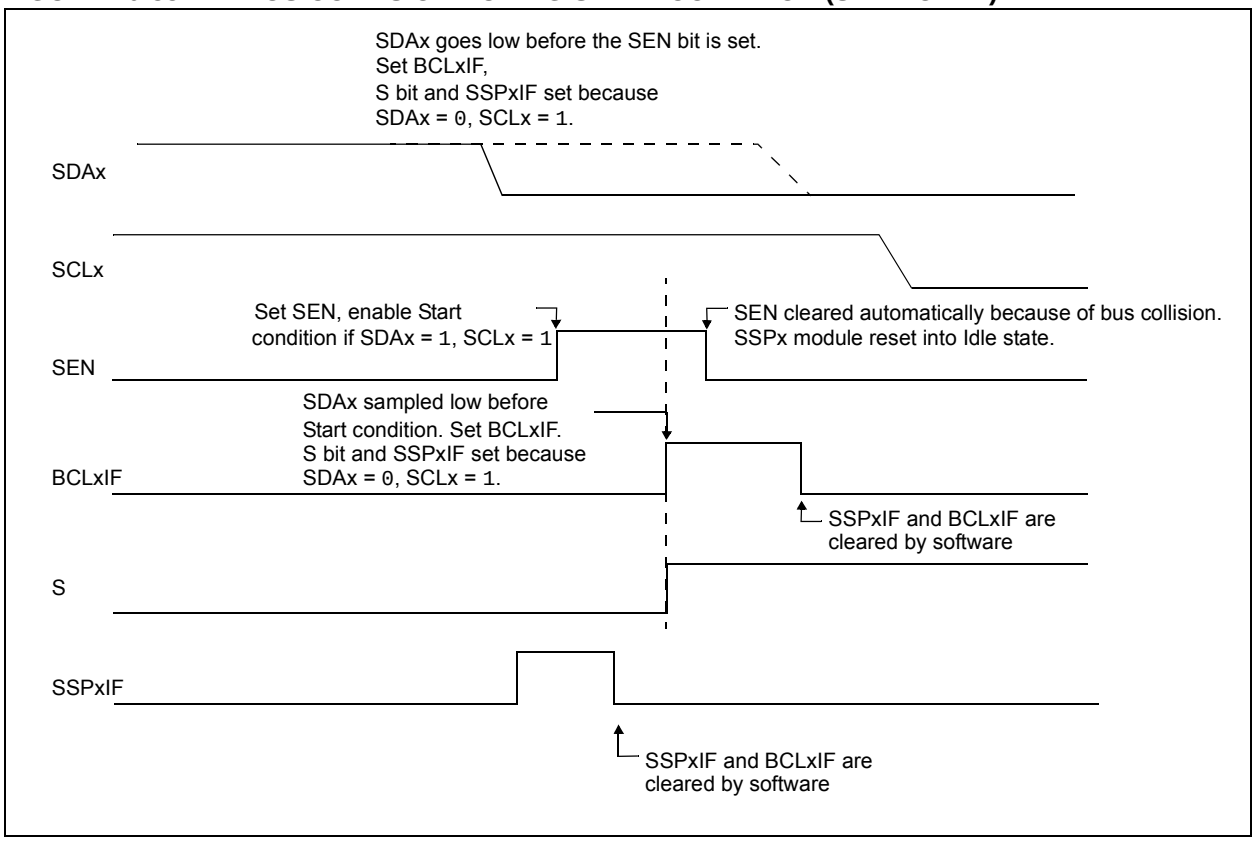
- the Start condition is aborted,
- the BCLxIF flag is set and
- the MSSPx module is reset to its Idle state (Figure 20-33).

The Start condition begins with the SDAx and SCLx pins deasserted. When the SDAx pin is sampled high, the Baud Rate Generator is loaded and counts down. If the SCLx pin is sampled low while SDAx 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 SDAx pin is sampled low during this count, the BRG is reset and the SDAx line is asserted early (Figure 20-35). If, however, a '1' is sampled on the SDAx pin, the SDAx 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 SCLx pin is sampled as '0' during this time, a bus collision does not occur. At the end of the BRG count, the SCLx 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 SDAx 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 20-33: BUS COLLISION DURING START CONDITION (SDAX ONLY)**




 © 2015-2016 Microchip Technology Inc.

**FIGURE 20-34: BUS COLLISION DURING START CONDITION (SCLX = 0)**

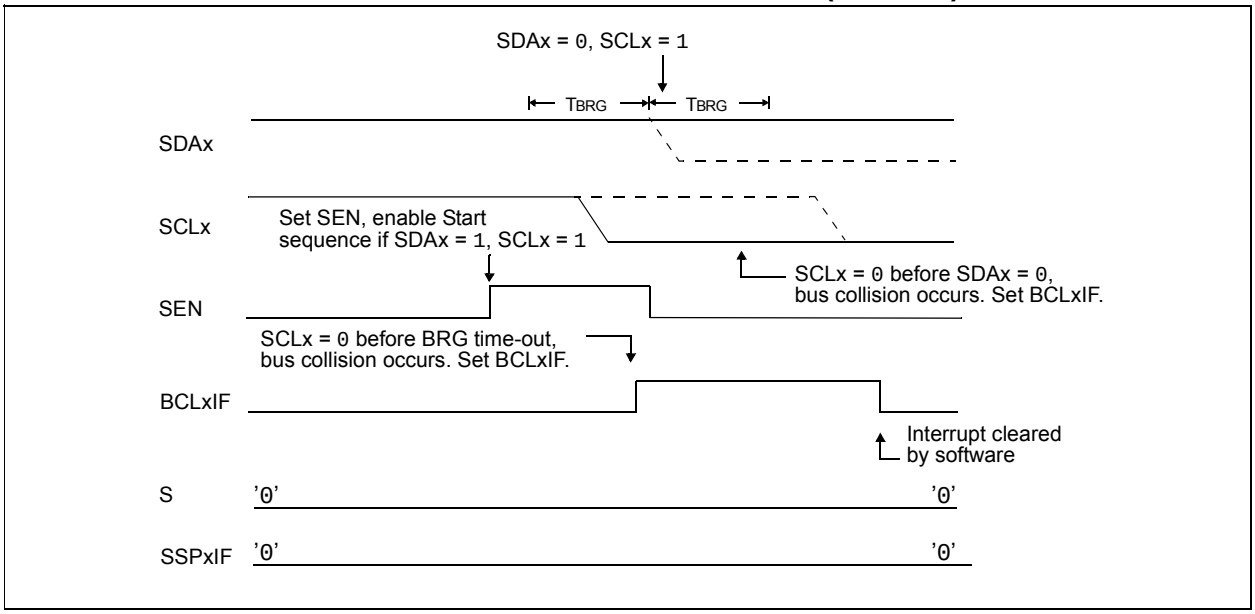
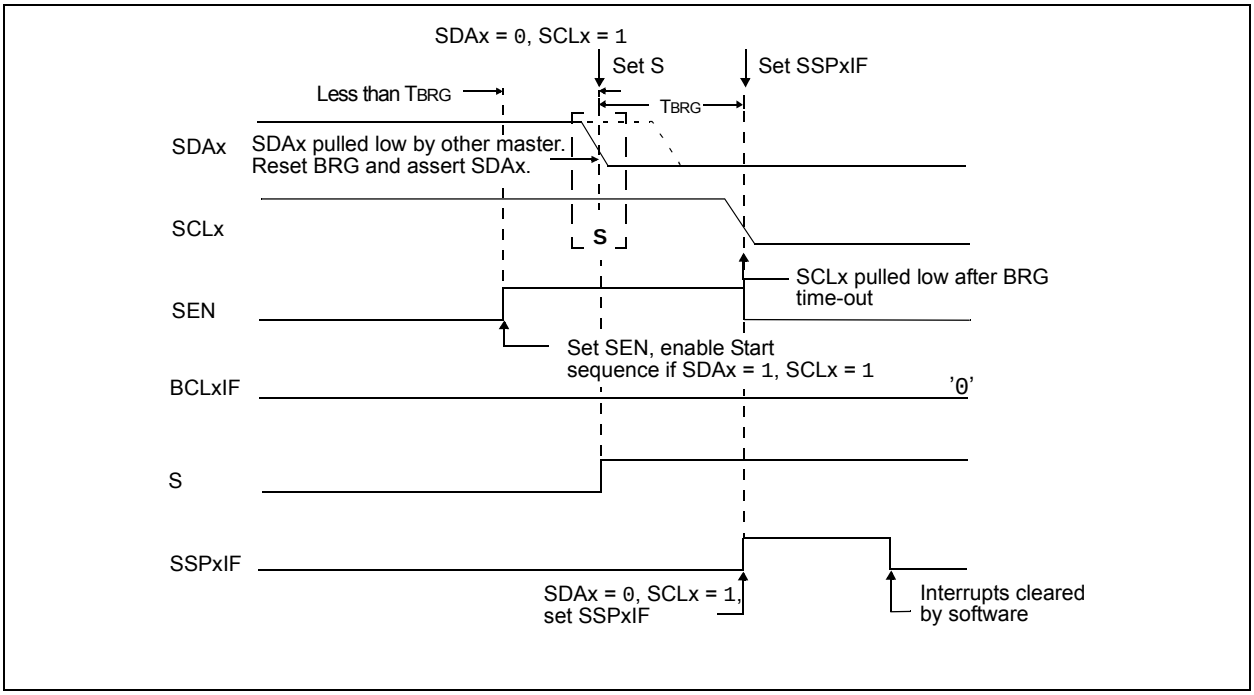



**FIGURE 20-35: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION**



### 20.6.13.2 Bus Collision During a Repeated Start Condition

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

a) A low level is sampled on SDAx when SCLx goes from low level to high level (Case 1).

b) SCLx goes low before SDAx is asserted low, indicating that another master is attempting to transmit a data '1' (Case 2).

When the user releases SDAx and the pin is allowed to float high, the BRG is loaded with SSPxADD and counts down to zero. The SCLx pin is then deasserted and when sampled high, the SDAx pin is sampled.

If SDAx is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', Figure 20-36). If SDAx is sampled high, the BRG is reloaded and begins counting. If SDAx goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDAx at exactly the same time.

If SCLx goes from high-to-low before the BRG times out and SDAx 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 20-37.

If, at the end of the BRG time-out, both SCLx and SDAx are still high, the SDAx pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCLx pin, the SCLx pin is driven low and the Repeated Start condition is complete.

**FIGURE 20-36: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**

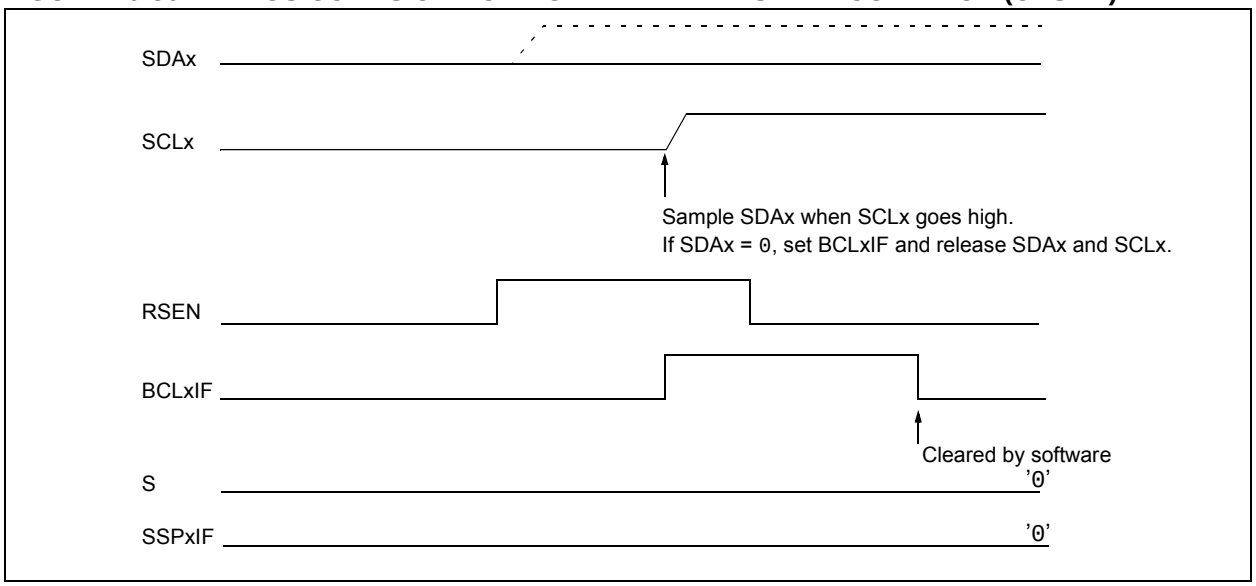
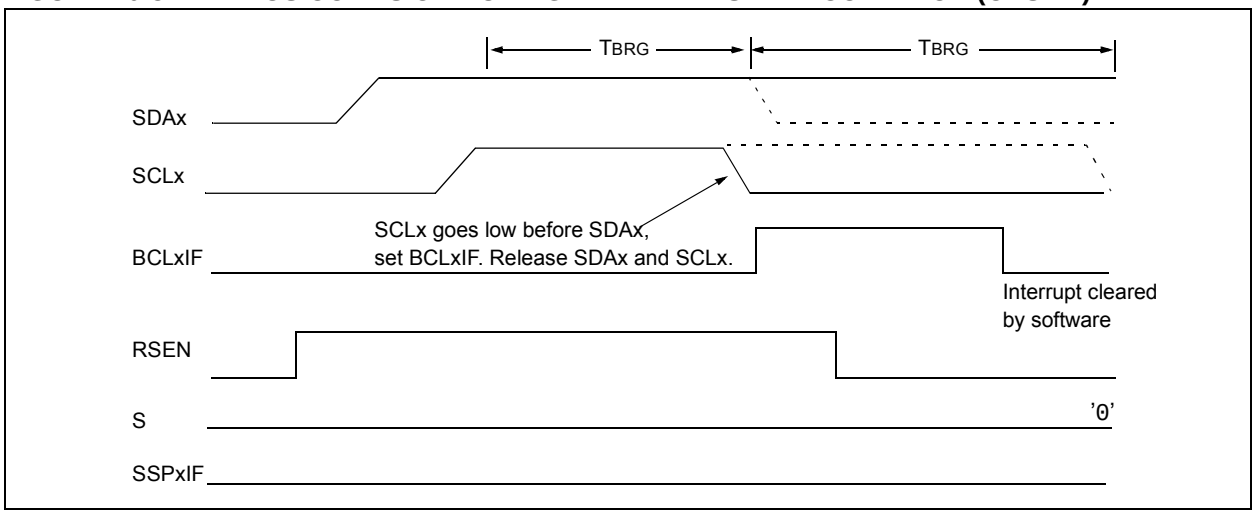



**FIGURE 20-37: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)**




 © 2015-2016 Microchip Technology Inc.

### 20.6.13.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDAx pin has been deasserted and allowed to float high, SDAx is sampled low after the BRG has timed out (Case 1).

b) After the SCLx pin is deasserted, SCLx is sampled low before SDAx goes high (Case 2).

The Stop condition begins with SDAx asserted low. When SDAx is sampled low, the SCLx pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPxADD and counts down to 0. After the BRG times out, SDAx is sampled. If SDAx is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 20-38). If the SCLx pin is sampled low before SDAx is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 20-39).

**FIGURE 20-38: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

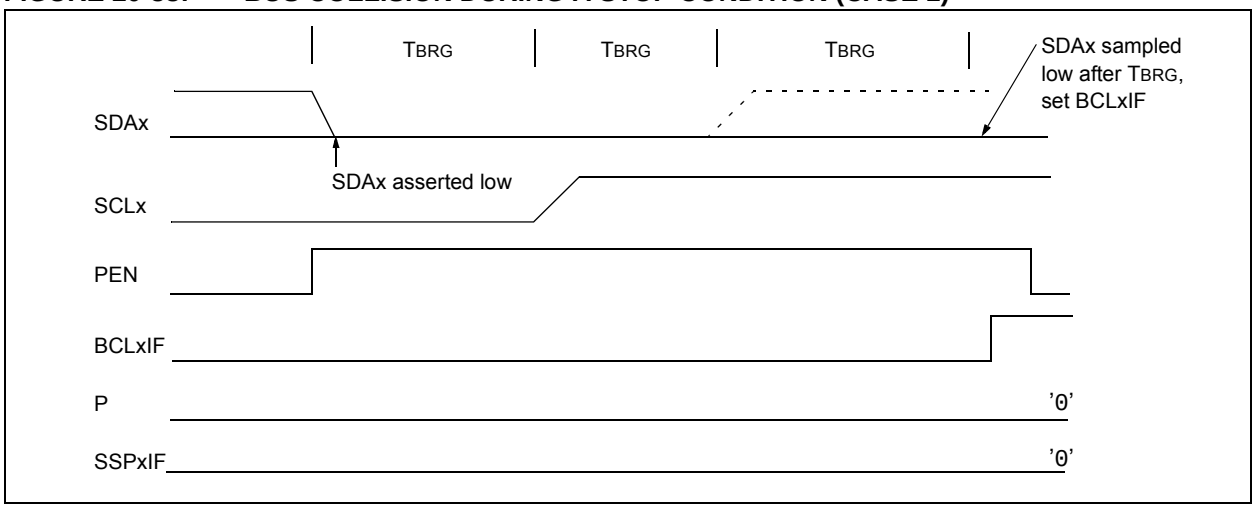



**FIGURE 20-39: BUS COLLISION DURING A STOP CONDITION (CASE 2)**

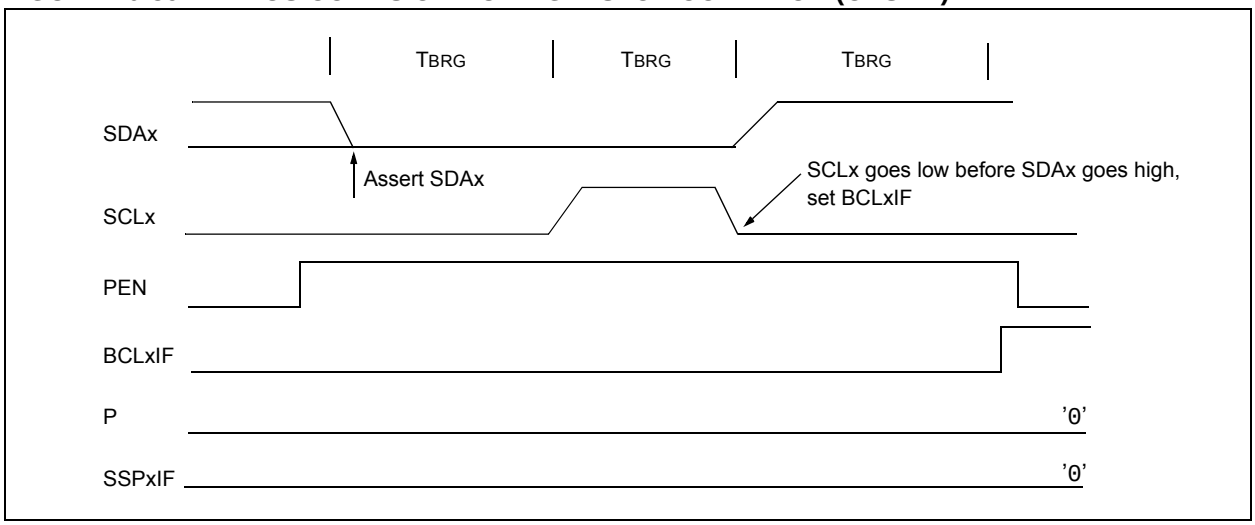


**TABLE 20-3: SUMMARY OF REGISTERS ASSOCIATED WITH $I^2C$ OPERATION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | ADIE | RC1IE | TX1IE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIE2 | — | AD2IE | — | — | BCL1IE | BCL2IE | TMR4IE | — | 86 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| PIR2 | — | AD2IF | — | — | BCL1IF | BCL2IF | TMR4IF | — | 88 |
| SSP1ADD | ADD<7:0> | | | | | | | | 253 |
| SSP2ADD | ADD<7:0> | | | | | | | | 253 |
| SSP1BUF | MSSPx Receive Buffer/Transmit Register | | | | | | | | 201* |
| SSP2BUF | MSSPx Receive Buffer/Transmit Register | | | | | | | | 201* |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 248 |
| SSP2CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 248 |
| SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 250 |
| SSP2CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 250 |
| SSP1CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 251 |
| SSP2CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 251 |
| SSP1MSK | MSK<7:0> | | | | | | | | 252 |
| SSP2MSK | MSK<7:0> | | | | | | | | 252 |
| SSP1STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 246 |
| SSP2STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 246 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 123 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 126 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used by the MSSP module in $I^2C$ mode.

* Page provides register information.

**Note 1:** PIC16F/LF1829 only.


© 2015-2016 Microchip Technology Inc.

## 20.7 BAUD RATE GENERATOR

The MSSPx module has a Baud Rate Generator available for clock generation in both $I^2C$ and SPI Master modes. The Baud Rate Generator (BRG) reload value is placed in the SSPxADD register (Register 20-6). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting down.

Once the given operation is complete, the internal clock will automatically stop counting and the clock pin will remain in its last state.

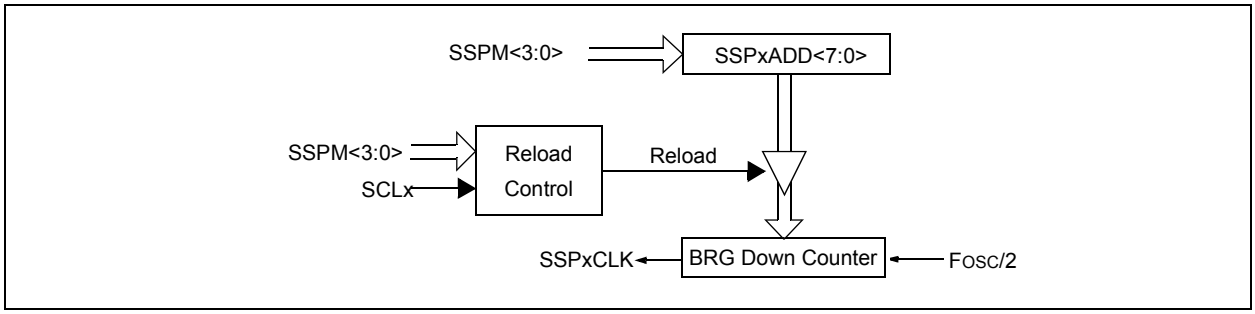
An internal signal “Reload” in Figure 20-40 triggers the value from SSPxADD to be loaded into the BRG counter. This occurs twice for each oscillation of the module clock line. The logic dictating when the reload signal is asserted depends on the mode the MSSPx is being operated in.

Table 20-4 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD.

**EQUATION 20-1:**

$$F_{CLOCK} = \frac{F_{OSC}}{(SSPxADD + 1)(4)}$$

**FIGURE 20-40: BAUD RATE GENERATOR BLOCK DIAGRAM**




**Note:** Values of 0x00, 0x01 and 0x02 are not valid for SSPxADD when used as a Baud Rate Generator for $I^2C$. This is an implementation limitation.

**TABLE 20-4: MSSPX CLOCK RATE W/BRG**

| FOSC | FCY | BRG Value | FCLOCK (2 Rollovers of BRG) |
|---|---|---|---|
| 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:** Refer to I/O port electrical and timing specifications in Table 25-9 and Figure 25-5 to ensure the system is designed to support the I/O timing requirements.

## 20.8 Register Definitions: MSSP Control

**REGISTER 20-1: SSPxSTAT: SSPx 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/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| 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

SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time

SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode

In $I^2C$ Master or Slave mode:
1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for High-Speed mode (400 kHz)

bit 6 **CKE:** SPI Clock Edge Select bit (SPI mode only)

In SPI Master or Slave mode:
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state

In $I^2C$ mode only:
1 = Enable input logic so that thresholds are compliant with SMBus specification
0 = Disable SMBus specific inputs

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit ($I^2C$ 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
($I^2C$ mode only. This bit is cleared when the MSSPx 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
($I^2C$ mode only. This bit is cleared when the MSSPx 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/$\overline{W}$:** Read/$\overline{Write}$ bit information ($I^2C$ mode only)
This bit holds the R/$\overline{W}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit, or not $\overline{ACK}$ bit.

In $I^2C$ Slave mode:
1 = Read
0 = Write

In $I^2C$ Master mode:
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 MSSPx is in Idle mode.

bit 1 **UA:** Update Address bit (10-bit $I^2C$ mode only)
1 = Indicates that the user needs to update the address in the SSPxADD register
0 = Address does not need to be updated

 © 2015-2016 Microchip Technology Inc.

**REGISTER 20-1: SSPxSTAT: SSPx STATUS REGISTER (CONTINUED)**

bit 0 **BF:** Buffer Full Status bit

Receive (SPI and $I^2C$ modes):
1 = Receive complete, SSPxBUF is full
0 = Receive not complete, SSPxBUF is empty

Transmit ($I^2C$ mode only):
1 = Data transmit in progress (does not include the $\overline{ACK}$ and Stop bits), SSPxBUF is full
0 = Data transmit complete (does not include the $\overline{ACK}$ and Stop bits), SSPxBUF is empty

**REGISTER 20-2: SSPxCON1: SSPx 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 | SSPxOV | 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 SSPxBUF register was attempted while the $I^2C$ conditions were not valid for a transmission to be started
0 = No collision
Slave mode:
1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPxOV:** Receive Overflow Indicator bit(1)
In SPI mode:
1 = A new byte is received while the SSPxBUF register is still holding the previous data. In case of overflow, the data in SSPxSR is lost. Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPxBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register (must be cleared in software).
0 = No overflow
In $I^2C$ mode:
1 = A byte is received while the SSPxBUF register is still holding the previous byte. SSPxOV 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 SCKx, SDOx, SDIx and $\overline{SSx}$ as the source of the serial port pins(2)
0 = Disables serial port and configures these pins as I/O port pins
In $I^2C$ mode:
1 = Enables the serial port and configures the SDAx and SCLx 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 $I^2C$ Slave mode:
SCLx release control
1 = Enable clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)
In $I^2C$ Master mode:
Unused in this mode

 © 2015-2016 Microchip Technology Inc.

**REGISTER 20-2: SSPxCON1: SSPx CONTROL REGISTER 1 (CONTINUED)**

bit 3-0 **SSPM<3:0>:** Synchronous Serial Port Mode Select bits
1111 = $I^2C$ Slave mode, 10-bit address with Start and Stop bit interrupts enabled
1110 = $I^2C$ Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1101 = Reserved
1100 = Reserved
1011 = $I^2C$ firmware controlled Master mode (Slave idle)
1010 = SPI Master mode, clock = FOSC/(4 * (SSPxADD+1))**(5)**
1001 = Reserved
1000 = $I^2C$ Master mode, clock = FOSC/(4 * (SSPxADD+1))**(4)**
0111 = $I^2C$ Slave mode, 10-bit address
0110 = $I^2C$ Slave mode, 7-bit address
0101 = SPI Slave mode, clock = SCKx pin, $\overline{SSx}$ pin control disabled, $\overline{SSx}$ can be used as I/O pin
0100 = SPI Slave mode, clock = SCKx pin, $\overline{SSx}$ pin control enabled
0011 = SPI Master mode, clock = TMR2 output/2
0010 = SPI Master mode, clock = FOSC/64
0001 = SPI Master mode, clock = FOSC/16
0000 = SPI Master mode, clock = FOSC/4

**Note 1:** In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register.
**2:** When enabled, these pins must be properly configured as input or output.
**3:** When enabled, the SDAx and SCLx pins must be configured as inputs.
**4:** SSPxADD values of 0, 1 or 2 are not supported for $I^2C$ mode.
**5:** SSPxADD value of '0' is not supported. Use SSPM = 0000 instead.

**REGISTER 20-3: SSPxCON2: SSPx 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 $I^2C$ Slave mode only)
1 = Enable interrupt when a general call address (0x00 or 00h) is received in the SSPxSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (in $I^2C$ mode only)
1 = Acknowledge was not received
0 = Acknowledge was received

bit 5 **ACKDT:** Acknowledge Data bit (in $I^2C$ 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 $I^2C$ Master mode only)
In Master Receive mode:
1 = Initiate Acknowledge sequence on SDAx and SCLx pins, and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence idle

bit 3 **RCEN:** Receive Enable bit (in $I^2C$ Master mode only)
1 = Enables Receive mode for $I^2C$
0 = Receive idle

bit 2 **PEN:** Stop Condition Enable bit (in $I^2C$ Master mode only)
SCKx Release Control:
1 = Initiate Stop condition on SDAx and SCLx pins. Automatically cleared by hardware.
0 = Stop condition idle

bit 1 **RSEN:** Repeated Start Condition Enabled bit (in $I^2C$ Master mode only)
1 = Initiate Repeated Start condition on SDAx and SCLx 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 SDAx and SCLx 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 $I^2C$ module is not in the Idle mode, this bit may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).

 © 2015-2016 Microchip Technology Inc.

**REGISTER 20-4: SSPxCON3: SSPx 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 ($I^2C$ mode only)**(3)**
1 = Indicates the $I^2C$ bus is in an Acknowledge sequence, set on eighth falling edge of SCLx clock
0 = Not an Acknowledge sequence, cleared on $9^{TH}$ rising edge of SCLx clock

bit 6 **PCIE**: Stop Condition Interrupt Enable bit ($I^2C$ Slave mode only)
1 = Enable interrupt on detection of Stop condition
0 = Stop detection interrupts are disabled**(2)**

bit 5 **SCIE**: Start Condition Interrupt Enable bit ($I^2C$ Slave 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 = SSPxBUF updates every time that a new data byte is shifted in ignoring the BF bit
0 = If new byte is received with BF bit of the SSPxSTAT register already set, SSPOV bit of the SSPxCON1 register is set, and the buffer is not updated
In $I^2C$ Master mode and SPI Master mode:
This bit is ignored.
In $I^2C$ Slave mode:
1 = SSPxBUF is updated and $\overline{ACK}$ is generated for a received address/data byte, ignoring the state of the SSPOV bit only if the BF bit = 0.
0 = SSPxBUF is only updated when SSPOV is clear

bit 3 **SDAHT:** SDAx Hold Time Selection bit ($I^2C$ mode only)
1 = Minimum of 300 ns hold time on SDAx after the falling edge of SCLx
0 = Minimum of 100 ns hold time on SDAx after the falling edge of SCLx

bit 2 **SBCDE:** Slave Mode Bus Collision Detect Enable bit ($I^2C$ Slave mode only)
If on the rising edge of SCLx, SDAx is sampled low when the module is outputting a high state, the BCLxIF bit of the PIR2 register is set, and bus goes idle
1 = Enable slave bus collision interrupts
0 = Slave bus collision interrupts are disabled

bit 1 **AHEN:** Address Hold Enable bit ($I^2C$ Slave mode only)
1 = Following the eighth falling edge of SCLx for a matching received address byte; CKP bit of the SSPxCON1 register will be cleared and the SCLx will be held low.
0 = Address holding is disabled

bit 0 **DHEN:** Data Hold Enable bit ($I^2C$ Slave mode only)
1 = Following the eighth falling edge of SCLx for a received data byte; slave hardware clears the CKP bit of the SSPxCON1 register and SCLx is held low.
0 = Data holding is disabled

**Note 1:** For daisy-chained SPI operation; allows the user to ignore all but the last received byte. SSPOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPxBUF.
**2:** This bit has no effect in Slave modes 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.

**REGISTER 20-5: SSPxMSK: SSPx 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 SSPxADD<n> to detect $I^2C$ address match
0 = The received address bit n is not used to detect $I^2C$ address match

bit 0 **MSK<0>:** Mask bit for $I^2C$ Slave mode, 10-bit Address
$I^2C$ Slave mode, 10-bit address (SSPM<3:0> = 0111 or 1111):
1 = The received address bit 0 is compared to SSPxADD<0> to detect $I^2C$ address match
0 = The received address bit 0 is not used to detect $I^2C$ address match
$I^2C$ Slave mode, 7-bit address, the bit is ignored

 © 2015-2016 Microchip Technology Inc.

**REGISTER 20-6: SSPxADD: MSSPx ADDRESS AND BAUD RATE REGISTER ($I^2C$ 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
SCLx pin clock period = ((ADD<7:0> + 1) *4)/FOSC

**10-Bit Slave mode — Most Significant Address byte:**

bit 7-3 **Not used:** Unused for Most Significant Address byte. Bit state of this register is a "don't care". Bit pattern sent by master is fixed by $I^2C$ 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".

**NOTES:**

 © 2015-2016 Microchip Technology Inc.

## 21.0 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 ADC or DAC 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 21-1 and Figure 21-2.

The EUSART transmit output (TX_out) is available to the TX/CK pin and internally to the following peripherals:

- Configurable Logic Cell (CLC)

**FIGURE 21-1: EUSART TRANSMIT BLOCK DIAGRAM**

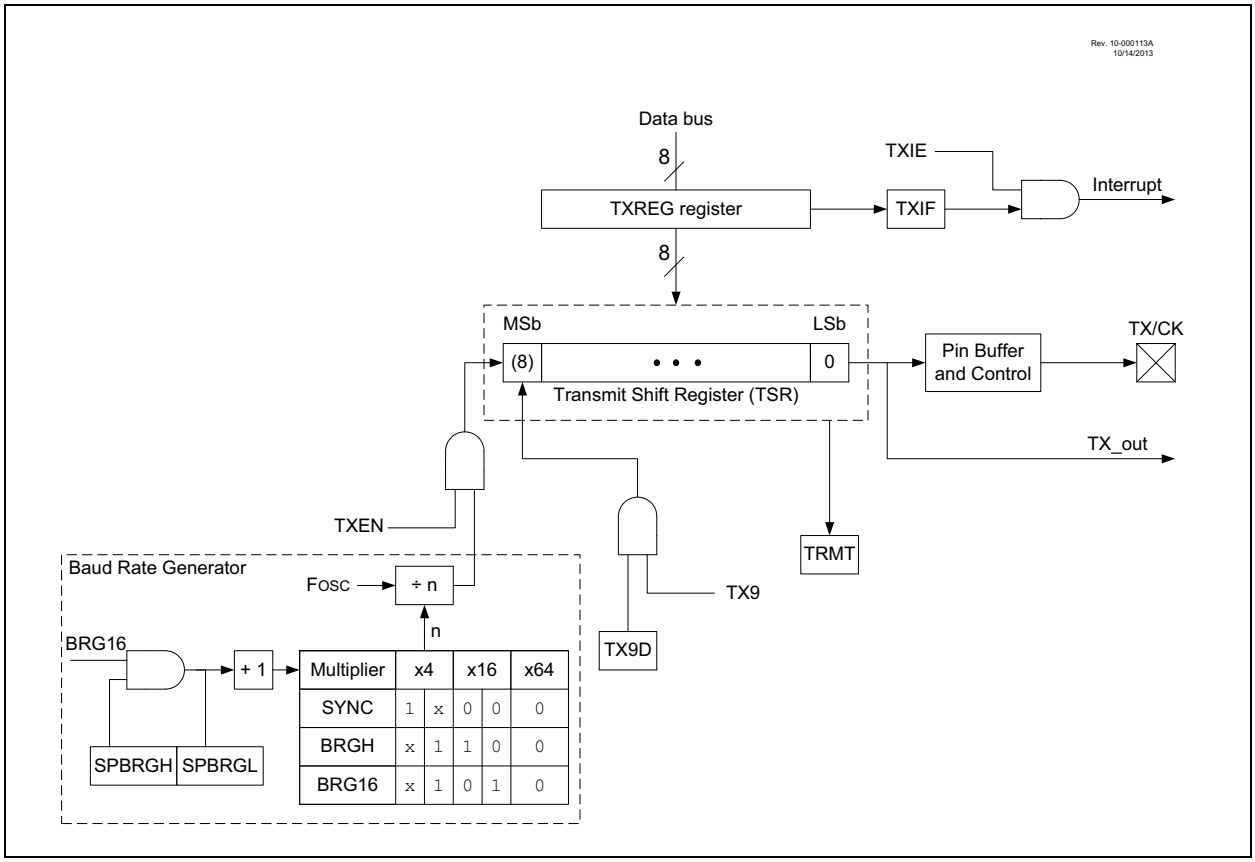
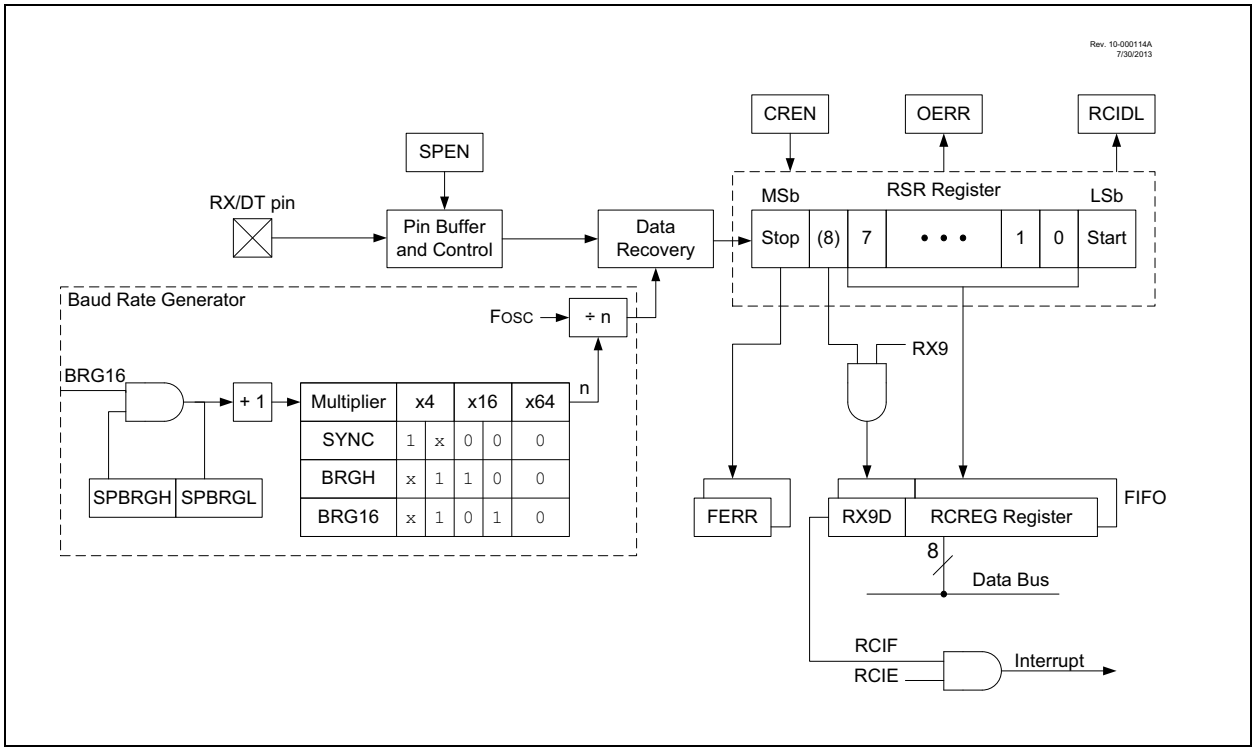


**FIGURE 21-2: EUSART RECEIVE BLOCK DIAGRAM**




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 21-1, Register 21-2 and Register 21-3, 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.

 © 2015-2016 Microchip Technology Inc.

## 21.1 EUSART Asynchronous Mode

The EUSART transmits and receives data using the standard non-return-to-zero (NRZ) format. NRZ is implemented with two levels: a VOH Mark state which represents a '1' data bit, and a VOL Space state which represents a '0' data bit. NRZ refers to the fact that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port idles in the Mark state. Each character transmission consists of one Start bit followed by eight or nine data bits and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is eight 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 21-5 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.

### 21.1.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 21-1. 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.

#### 21.1.1.1 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 ANSELx bit.

| **Note:** | The TXIF Transmitter Interrupt flag is set when the TXEN enable bit is set. |
|---|---|

#### 21.1.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREG is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the TXREG until the Stop bit of the previous character has been transmitted. The pending character in the TXREG is then transferred to the TSR in one TCY immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXREG.

#### 21.1.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the SCKP bit of the BAUDCON 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 21.5.1.2 "Clock Polarity"**.

#### 21.1.1.4 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.

#### 21.1.1.5 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. |
|---|---|

#### 21.1.1.6 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 nine 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 eight 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 21.1.2.7 "Address Detection"** for more information on the address mode.

#### 21.1.1.7 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 21.4 "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 eight 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 21-3: ASYNCHRONOUS TRANSMISSION**

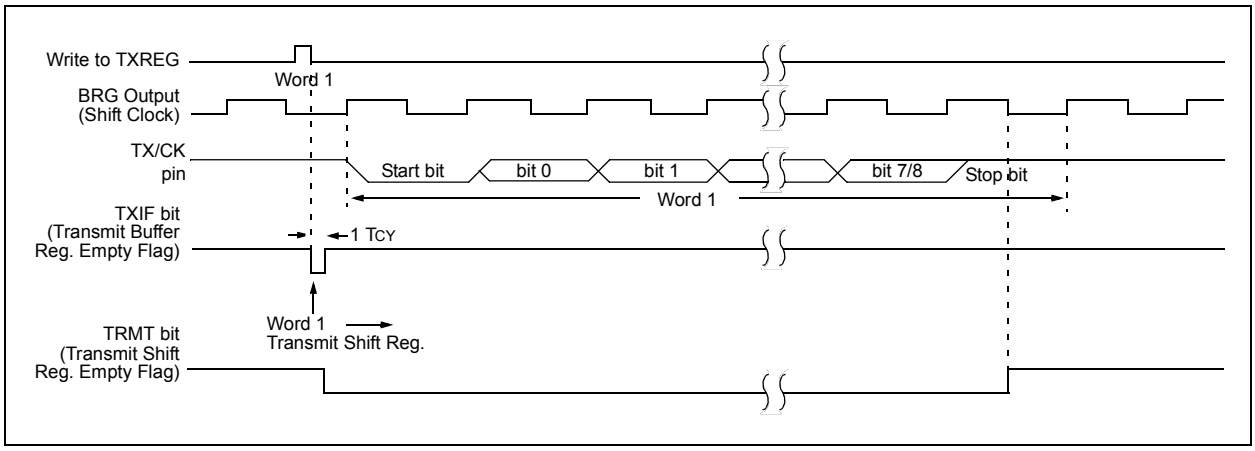
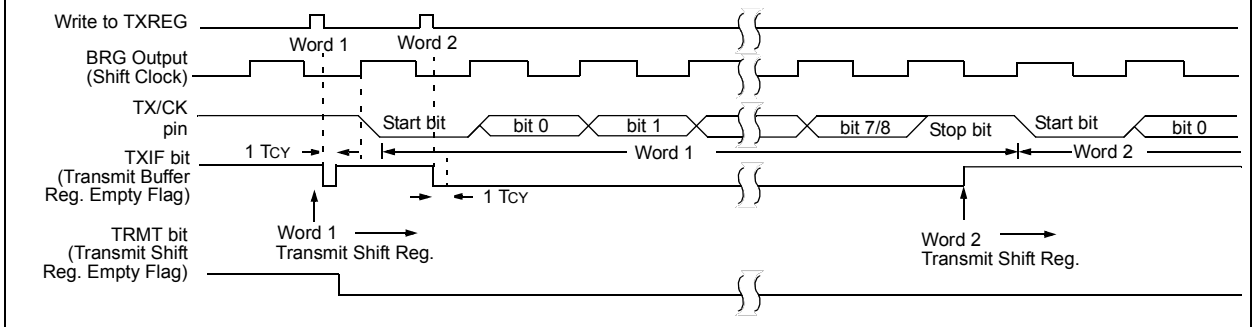



**FIGURE 21-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**




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

 © 2015-2016 Microchip Technology Inc.

**TABLE 21-1: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 266 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 265 |
| SPBRGL | BRG<7:0> | | | | | | | | 267* |
| SPBRGH | BRG<15:8> | | | | | | | | 267* |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| TXREG | EUSART Transmit Data Register | | | | | | | | 257 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 264 |

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

* Page provides register information.

### 21.1.2 EUSART ASYNCHRONOUS RECEIVER

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in Figure 21-2. The data is received on the RX/DT pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all eight or nine bits of the character have been shifted in, they are immediately transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must start servicing the EUSART receiver. The FIFO and RSR registers are not directly accessible by software. Access to the received data is via the RCREG register.

#### 21.1.2.1 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 TRISx 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 ANSELx bit must be cleared for the receiver to function. |
|---|---|

#### 21.1.2.2 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 21.1.2.4 "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 21.1.2.5 "Receive Overrun Error"** for more information on overrun errors. |
|---|---|

#### 21.1.2.3 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.

 © 2015-2016 Microchip Technology Inc.

#### 21.1.2.4 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. |

#### 21.1.2.5 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.

#### 21.1.2.6 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 nine bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth and Most Significant data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the eight Least Significant bits from the RCREG.

#### 21.1.2.7 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.

#### 21.1.2.8 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 21.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSELx 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 eight 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.

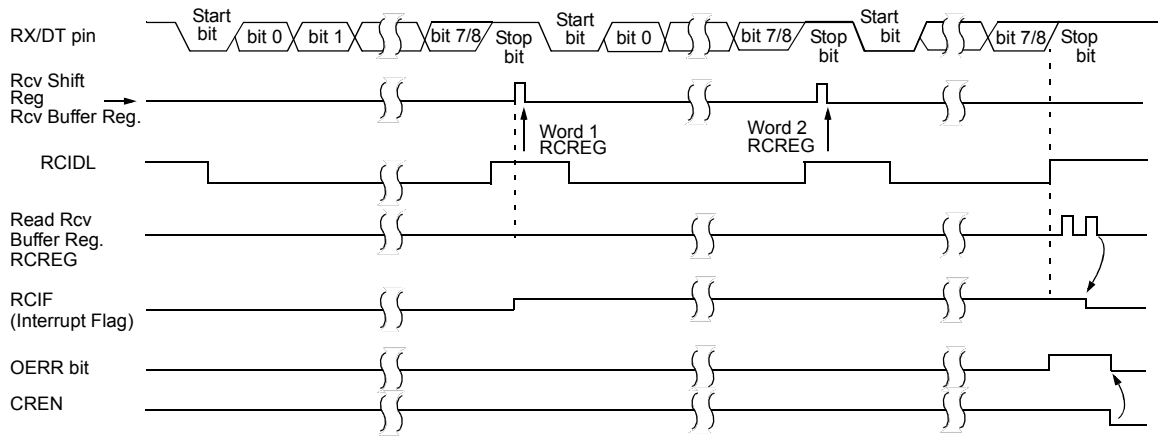
#### 21.1.2.9 9-Bit Address Detection Mode Set-up

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 21.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSELx 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 eight 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.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 21-5: ASYNCHRONOUS RECEPTION**




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

**TABLE 21-2: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 266 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| RCREG | EUSART Receive Data Register | | | | | | | | 260* |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 265 |
| SPBRGL | BRG<7:0> | | | | | | | | 267* |
| SPBRGH | BRG<15:8> | | | | | | | | 267* |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 264 |

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

* Page provides register information.

## 21.2 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.

The Auto-Baud Detect feature (see **Section 21.4.1 "Auto-Baud Detect"**) can be used to compensate for changes in the INTOSC frequency.

There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change in the peripheral clock frequency.

## 21.3 Register Definitions: EUSART Control

**REGISTER 21-1: 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
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

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

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

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

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

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

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

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

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

 © 2015-2016 Microchip Technology Inc.

**REGISTER 21-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER**

| 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.

**REGISTER 21-3: 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, RCIF bit 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

 © 2015-2016 Microchip Technology Inc.

## 21.4 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 21-3 contains the formulas for determining the baud rate. Example 21-1 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 21-3. 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 21-1: CALCULATING BAUD RATE ERROR**

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

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

Solving for SPBRGH:SPBRGL:

$$X = \frac{\frac{FOSC}{Desired\ Baud\ Rate}}{64} - 1$$

$$= \frac{\frac{16000000}{9600}}{64} - 1$$

$$= [25.042] = 25$$

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

$$= 9615$$

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

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

**TABLE 21-3: BAUD RATE FORMULAS**

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|---|---|---|---|---|
| SYNC | BRG16 | BRGH | | |
| 0 | 0 | 0 | 8-bit/Asynchronous | FOSC/[64 (n+1)] |
| 0 | 0 | 1 | 8-bit/Asynchronous | FOSC/[16 (n+1)] |
| 0 | 1 | 0 | 16-bit/Asynchronous | |
| 0 | 1 | 1 | 16-bit/Asynchronous | FOSC/[4 (n+1)] |
| 1 | 0 | x | 8-bit/Synchronous | |
| 1 | 1 | x | 16-bit/Synchronous | |

**Legend:** x = Don't care, n = value of SPBRGH:SPBRGL register pair.

**TABLE 21-4: 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 | 266 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 265 |
| SPBRGL | BRG<7:0> | | | | | | | | 267* |
| SPBRGH | BRG<15:8> | | | | | | | | 267* |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 264 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for the Baud Rate Generator.

* Page provides register information.

 © 2015-2016 Microchip Technology Inc.

**TABLE 21-5: 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 |

**TABLE 21-5: 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 | — | — | — |


© 2015-2016 Microchip Technology Inc.

**TABLE 21-5: 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 | — | — | — |

### 21.4.1 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 21-6). 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 21-6. 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 21-6. 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 21.4.3 "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.

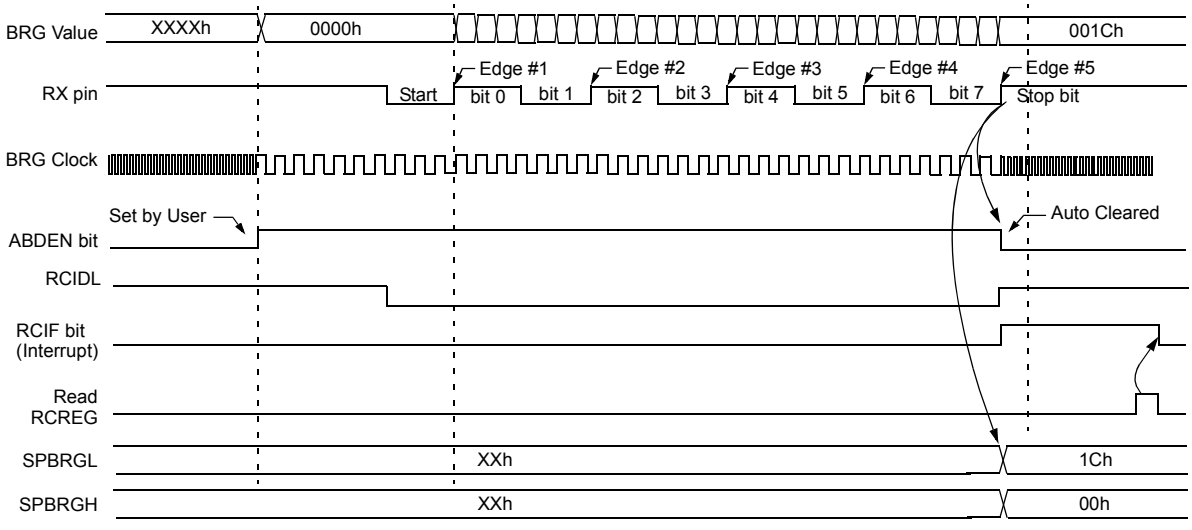
**3:** During the auto-baud process, the auto-baud counter starts counting at 1. Upon completion of the auto-baud sequence, to achieve maximum accuracy, subtract 1 from the SPBRGH:SPBRGL register pair.

**TABLE 21-6: BRG COUNTER CLOCK RATES(1)**

| BRG16 | BRGH | BRG Base Clock | BRG ABD Clock |
|---|---|---|---|
| 0 | 0 | FOSC/64 | FOSC/512 |
| 0 | 1 | FOSC/16 | FOSC/128 |
| 1 | 0 | FOSC/16 | FOSC/128 |
| 1 | 1 | FOSC/4 | FOSC/32 |

**Note 1:** During the ABD sequence, SPBRGL and SPBRGH registers are both used as a 16-bit counter, independent of BRG16 setting.

**FIGURE 21-6: AUTOMATIC BAUD RATE CALIBRATION**




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

 © 2015-2016 Microchip Technology Inc.

### 21.4.2 AUTO-BAUD OVERFLOW

Replace the current paragraphs under the Auto-Baud Overflow section with the following text:

During the course of automatic baud detection, the ABDOVF bit of the BAUDxCON 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 SPxBRGH:SPxBRGL register pair. The overflow condition will set the RCIF flag. The counter continues to count until the fifth rising edge is detected on the RX pin. The RCIDL bit will remain false ('0') until the fifth rising edge at which time the RCIDL bit will be set. If the RCREG is read after the overflow occurs but before the fifth rising edge then the fifth rising edge will set the RCIF again.

Terminating the auto-baud process early to clear an overflow condition will prevent proper detection of the sync character fifth rising edge. If any falling edges of the sync character have not yet occurred when the ABDEN bit is cleared then those will be falsely detected as start bits. The following steps are recommended to clear the overflow condition:

1. Read RCREG to clear RCIF
2. If RCIDL is zero then wait for RCIF and repeat Step 1.
3. Clear the ABDOVF bit.

### 21.4.3 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 21-7), and asynchronously if the device is in Sleep mode (Figure 21-8). 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.

#### 21.4.3.1 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 ten 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

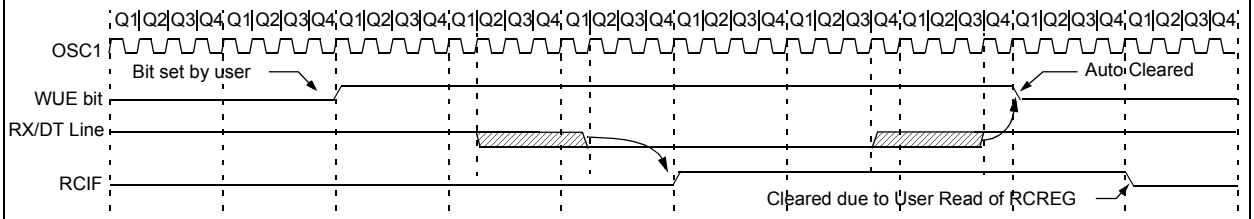
Oscillator start-up time must be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL mode). The Sync Break (or wake-up signal) character must be of sufficient length, and be followed by a sufficient interval, to allow enough time for the selected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

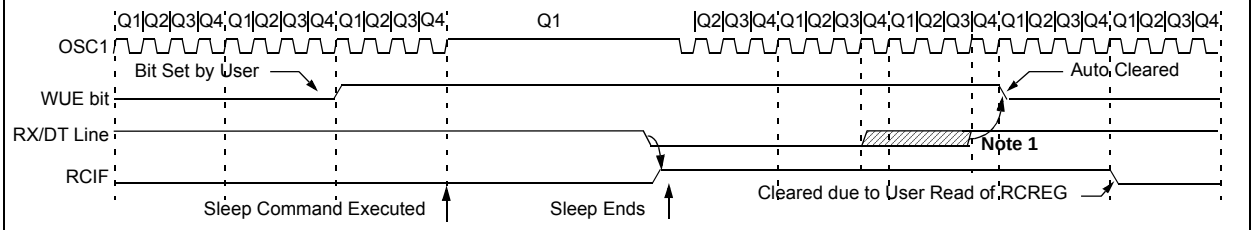
To ensure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process before setting the WUE bit. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**FIGURE 21-7: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION**




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

**FIGURE 21-8: 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.

 © 2015-2016 Microchip Technology Inc.

### 21.4.4 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 21-9 for the timing of the Break character sequence.

#### 21.4.4.1 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.

### 21.4.5 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.

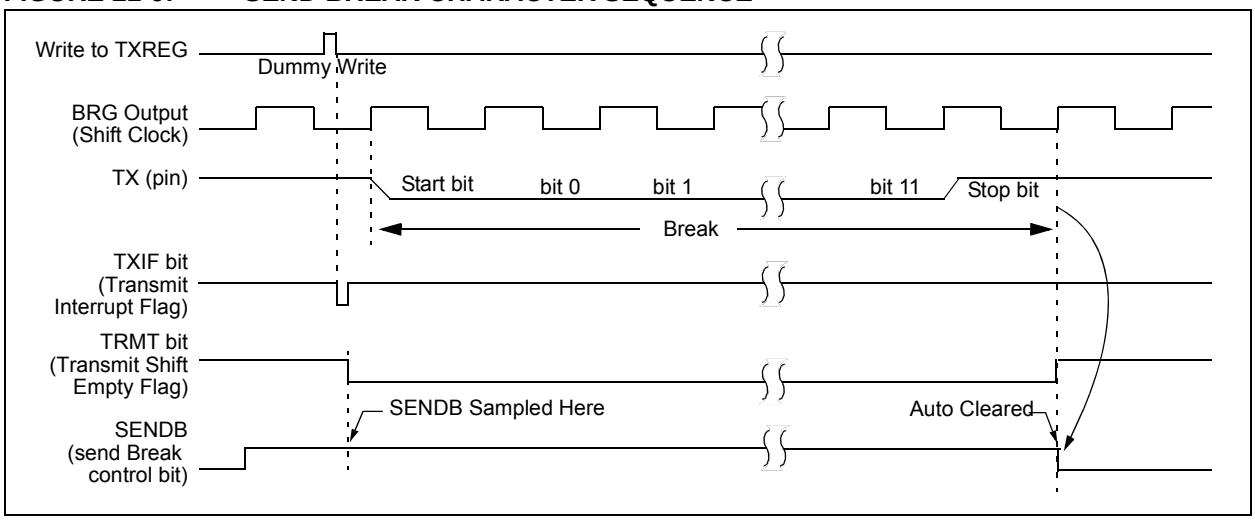
A Break character has been received when;

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

The second method uses the Auto-Wake-up feature described in **Section 21.4.3 "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 21-9: SEND BREAK CHARACTER SEQUENCE**



## 21.5 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.

### 21.5.1 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.

#### 21.5.1.1 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.

#### 21.5.1.2 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.

#### 21.5.1.3 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.

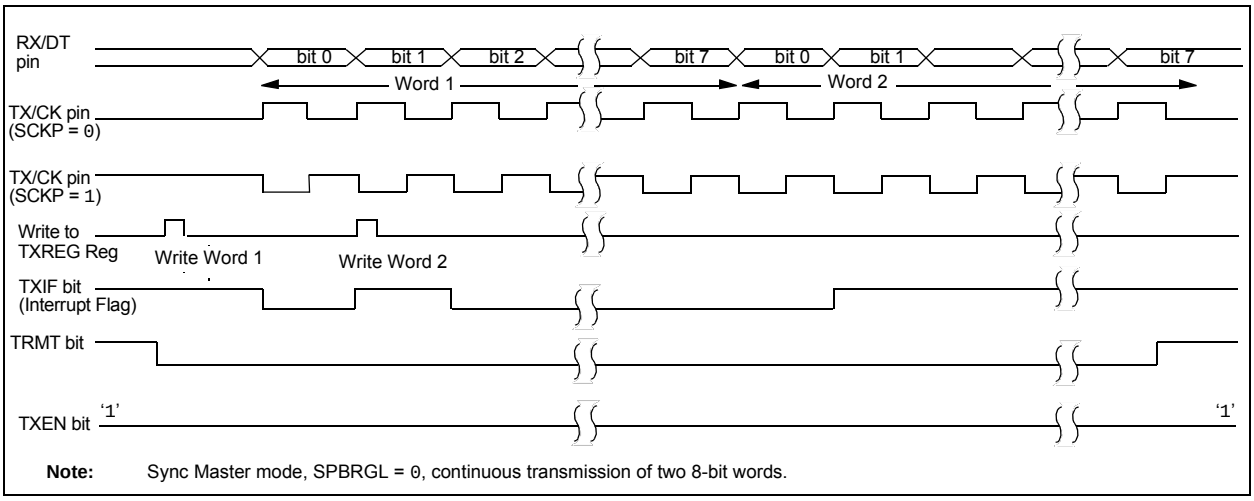
| **Note:** | The TSR register is not mapped in data memory, so it is not available to the user. |
|---|---|

#### 21.5.1.4 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 21.4 "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.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 21-10: SYNCHRONOUS TRANSMISSION**




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

**FIGURE 21-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**

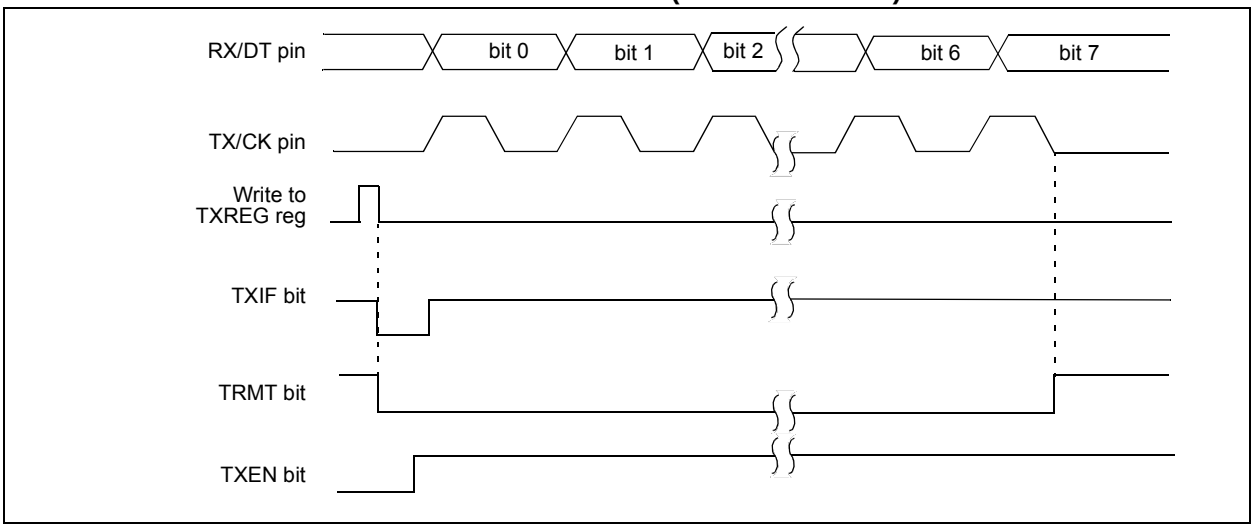



**TABLE 21-7: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 266 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 265 |
| SPBRGL | BRG<7:0> | | | | | | | | 267* |
| SPBRGH | BRG<15:8> | | | | | | | | 267* |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| TXREG | EUSART Transmit Data Register | | | | | | | | 257* |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 264 |

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

#### 21.5.1.5 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 ANSELx bit must be cleared for the receiver to function. |

#### 21.5.1.6 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 ANSELx bit must be cleared. |

#### 21.5.1.7 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.

#### 21.5.1.8 Receiving 9-Bit Characters

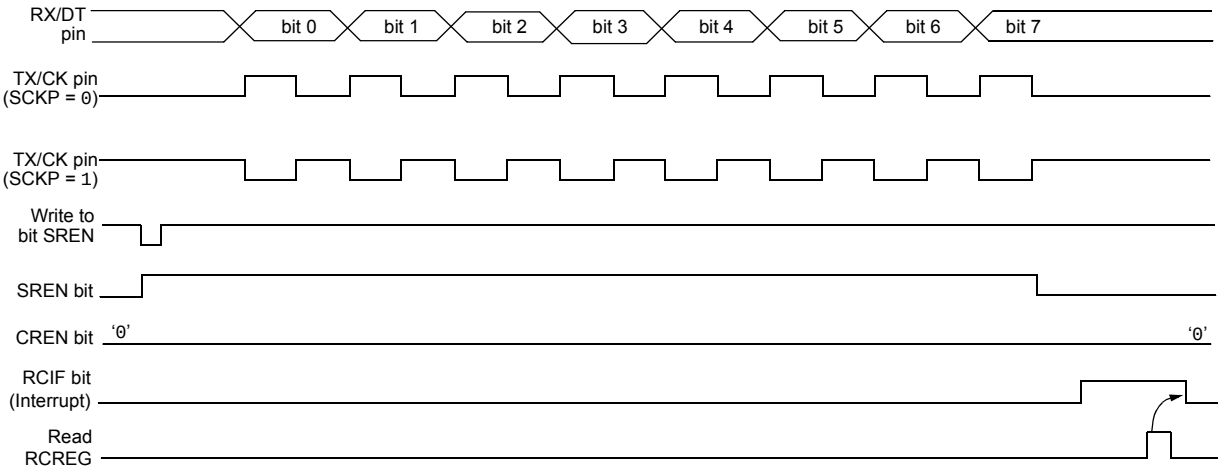
The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTA register is set, the EUSART will shift nine bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth, and Most Significant, data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the eight Least Significant bits from the RCREG.

#### 21.5.1.9 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 ANSELx 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.

 © 2015-2016 Microchip Technology Inc.

**FIGURE 21-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)**




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

**TABLE 21-8: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 266 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| RCREG | EUSART Receive Data Register | | | | | | | | 260* |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 265 |
| SPBRGL | BRG<7:0> | | | | | | | | 267* |
| SPBRGH | BRG<15:8> | | | | | | | | 267* |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 264 |

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

* Page provides register information.

### 21.5.2 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.

#### 21.5.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see **Section 21.5.1.3 "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 the 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.

#### 21.5.2.2 Synchronous Slave Transmission Set-up:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSELx 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 eight bits to the TXREG register.

**TABLE 21-9: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 266 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 265 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| TXREG | EUSART Transmit Data Register | | | | | | | | 257* |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 264 |

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

* Page provides register information.

 © 2015-2016 Microchip Technology Inc.

#### 21.5.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (**Section 21.5.1.5 "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.

#### 21.5.2.4 Synchronous Slave Reception Set-up:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSELx 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 eight 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 21-10: 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 |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 266 |
| INTCON | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 84 |
| PIE1 | TMR1GIE | AD1IE | RCIE | TXIE | SSP1IE | SSP2IE | TMR2IE | TMR1IE | 85 |
| PIR1 | TMR1GIF | AD1IF | RCIF | TXIF | SSP1IF | SSP2IF | TMR2IF | TMR1IF | 87 |
| RCREG | EUSART Receive Data Register | | | | | | | | 260* |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 265 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 119 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 264 |

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

* Page provides register information.

## 22.0 PULSE-WIDTH MODULATION (PWM) MODULE

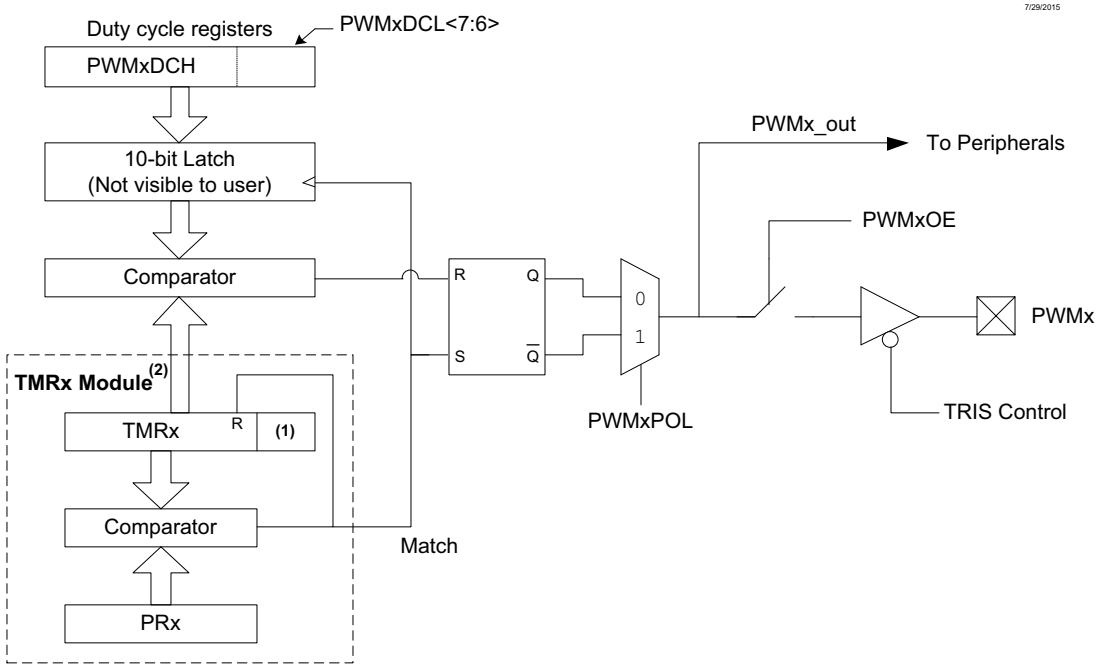
The PWM module generates a pulse-width modulated signal determined by the duty cycle, period, and resolution that are configured by the following registers:

- PRx based on PWMTMRS
- TxCON based on PWMTMRS
- PWMxDCH
- PWMxDCL
- PWMxCON

Figure 22-1 shows a simplified block diagram of PWM operation.

For a step-by-step procedure on how to set up this module for PWM operation, refer to **Section 22.1.9 "Setup for PWM Operation Using PWMx Pins"**.

**FIGURE 22-1: SIMPLIFIED PWM BLOCK DIAGRAM**




**Note 1:** 8-bit timer is concatenated with two bits generated by Fosc or two bits of the internal prescaler to create 10-bit time-base.

**Note 2:** Timer dependent on PWMTMRS register settings.

 © 2015-2016 Microchip Technology Inc.

## 22.1 PWMx Pin Configuration

All PWM outputs are multiplexed with the PORT data latch. The user must configure the pins as outputs by clearing the associated TRISx bits.

| | |
|---|---|
| **Note:** | Clearing the PWMxOE bit will relinquish control of the PWMx pin. |

### 22.1.1 FUNDAMENTAL OPERATION

The PWM module produces a 10-bit resolution output. PWMTMRS selects TMRx and PRx which set the period of the PWM. The PWMxDCL and PWMxDCH registers configure the duty cycle. The period is common to all PWM modules, whereas the duty cycle is independently controlled.

| | |
|---|---|
| **Note:** | The Timer2/4 postscaler is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output. |

All PWM outputs associated with Timer2/4 are set when TMRx is cleared. Each PWMx is cleared when TMRx is equal to the value specified in the corresponding PWMxDCH (8 MSb) and PWMxDCL<7:6> (2 LSb) registers. When the value is greater than or equal to PRx, the PWM output is never cleared (100% duty cycle).

| | |
|---|---|
| **Note:** | The PWMxDCH and PWMxDCL registers are double buffered. The buffers are updated when TMRx matches PRx. Care should be taken to update both registers before the timer match occurs. |

### 22.1.2 PWM OUTPUT POLARITY

The output polarity is inverted by setting the PWMxPOL bit of the PWMxCON register.

### 22.1.3 PWM PERIOD

The PWM period is specified by the PRx register of the timer selected by PWMTMRS. The PWM period can be calculated using the formula of Equation 22-1.

**EQUATION 22-1: PWM PERIOD**

$$PWM\ Period = [(PRx) + 1] \cdot 4 \cdot TOSC \cdot (TMRx\ Prescale\ Value)$$

**Note:** TOSC = 1/FOSC

When TMRx is equal to PRx, the following three events occur on the next increment cycle:

- TMRx is cleared
- The PWM output is active (Exception: When the PWM duty cycle = 0%, the PWM output will remain inactive.)
- The PWMxDCH and PWMxDCL register values are latched into the buffers.

| | |
|---|---|
| **Note:** | The Timer2/4 postscaler has no effect on the PWM operation. |

### 22.1.4 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to the PWMxDCH and PWMxDCL register pair. The PWMxDCH register contains the eight MSbs and the PWMxDCL<7:6>, the two LSbs. The PWMxDCH and PWMxDCL registers can be written to at any time.

Equation 22-2 is used to calculate the PWM pulse width.

Equation 22-3 is used to calculate the PWM duty cycle ratio.

**EQUATION 22-2: PULSE WIDTH**

$$Pulse\ Width = (PWMxDCH:PWMxDCL<7:6>) \cdot TOSC \cdot (TMRx\ Prescale\ Value)$$

**Note:** TOSC = 1/FOSC

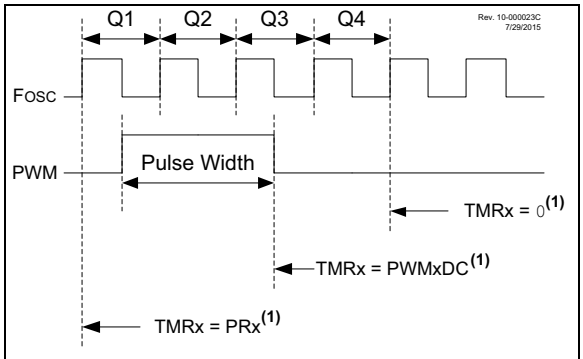
**EQUATION 22-3: DUTY CYCLE RATIO**

$$Duty\ Cycle\ Ratio = \frac{(PWMxDCH:PWMxDCL<7:6>)}{4(PRx + 1)}$$

The 8-bit timer TMR2 register is concatenated with the two Least Significant bits of 1/FOSC, adjusted by the Timer2 prescaler to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

Figure 22-2 shows a waveform of the PWM signal when the duty cycle is set for the smallest possible pulse.

**FIGURE 22-2: PWM OUTPUT**




**Note 1:** Timer dependent on PWMTMRS register settings.

### 22.1.5 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is ten bits when PRx is 255. The resolution is a function of the PRx register value as shown by Equation 22-4.

**EQUATION 22-4: PWM RESOLUTION**

$$Resolution = \frac{log[4(PRxs + 1)]}{log(2)} \ bits$$

**Note:** If the pulse width value is greater than the period the assigned PWM pin(s) will remain unchanged.

**TABLE 22-1: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 32 MHz)**

| PWM Frequency | 1.95 kHz | 7.81 kHz | 31.25 kHz | 125 kHz | 250 kHz | 333.3 kHz |
|---|---|---|---|---|---|---|
| Timer Prescale (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PRx Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.6 |

**TABLE 22-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 20 MHz)**

| PWM Frequency | 0.31 kHz | 4.88 kHz | 19.53 kHz | 78.12 kHz | 156.3 kHz | 208.3 kHz |
|---|---|---|---|---|---|---|
| Timer Prescale | 64 | 4 | 1 | 1 | 1 | 1 |
| PRx Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.6 |

**TABLE 22-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 8 MHz)**

| PWM Frequency | 0.31 kHz | 4.90 kHz | 19.61 kHz | 76.92 kHz | 153.85 kHz | 200.0 kHz |
|---|---|---|---|---|---|---|
| Timer Prescale | 64 | 4 | 1 | 1 | 1 | 1 |
| PRx Value | 0x65 | 0x65 | 0x65 | 0x19 | 0x0C | 0x09 |
| Maximum Resolution (bits) | 8 | 8 | 8 | 6 | 5 | 5 |

### 22.1.6 OPERATION IN SLEEP MODE

In Sleep mode, the TMRx register will not increment and the state of the module will not change. If the PWMx pin is driving a value, it will continue to drive that value. When the device wakes up, TMRx will continue from its previous state.

### 22.1.7 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency (FOSC). Any changes in the system clock frequency will result in changes to the PWM frequency. Refer to **Section 5.0 "Oscillator Module"** for additional details.

### 22.1.8 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the PWM registers to their Reset states.

 © 2015-2016 Microchip Technology Inc.

### 22.1.9 SETUP FOR PWM OPERATION USING PWMx PINS

The following steps should be taken when configuring the module for PWM operation using the PWMx pins:

1. Disable the PWMx pin output driver(s) by setting the associated TRISx bit(s).
2. Clear the PWMxCON register.
3. Load the PRx register with the PWM period value.
4. Clear the PWMxDCH register and bits <7:6> of the PWMxDCL register.

- Configure the PWMTMRS register to select Timer2/4.

5. Configure and start Timer2/4:

- Clear the TMRxIF interrupt flag bit of the PIRx register. See note below.
- Configure the TxCKPS bits of the TxCON register with the Timer2/4 prescale value.
- Enable Timer2/4 by setting the TMRxON bit of the TxCON register.

6. Enable PWM output pin and wait until Timer2/4 overflows, TMRxIF bit of the PIRx register is set. See note below.
7. Enable the PWMx pin output drivers:

- Clear the associated TRISx bit(s).
- Set the PWMxOE bit of the PWMxCON register.
- Select additional output options in the PWMxAOE register.

8. Configure the PWM module by loading the PWMxCON register with the appropriate values.

**Note 1:** In order to send a complete duty cycle and period on the first PWM output, the above steps must be followed in the order given. If it is not critical to start with a complete PWM signal, then move Step 8 to replace Step 4.

**2:** For operation with other peripherals only, disable PWMx pin outputs.

## 22.2 Register Definitions: PWM Control

**REGISTER 22-1: PWMxCON: PWM CONTROL REGISTER**

| R/W-0 | R/W-0 | R-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| PWMxEN | PWMxOE | PWMxOUT | PWMxPOL | — | — | — | — |
| 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 **PWMxEN:** PWM Enable Bit
1 = PWM module is enabled
0 = PWM module is disabled

bit 6 **PWMxOE:** PWM Output on pin PWMx Enable Bit
1 = Output to PWMx pin is enabled
0 = Output to PWMx pin is disabled (PWMxy pins may still be enabled, see Register 22-3)

bit 5 **PWMxOUT:** PWM Output Value Bit
1 = PWM output is high
0 = PWM output is low

bit 4 **PWMxPOL:** PWM Polarity Bit
1 = PWM output is active-low
0 = PWM output is active-high

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

**REGISTER 22-2: PWMTMRS: PWM TIMER SELECT REGISTER(1)**

| U-0 | U-0 | U-0 | R/W-0/0 | U-0 | R/W-0/0 | U-0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| | | | | — | P2TSEL | — | P1TSEL |
| 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 **P2TSEL:** PWM2 Timer Selection bit
1 = PWM is based off Timer 4
0 = PWM is based off Timer 2

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

bit 0 **P1TSEL:** PWM1 Timer Selection bit
1 = PWM is based off Timer 4
0 = PWM is based off Timer 2

 © 2015-2016 Microchip Technology Inc.

**REGISTER 22-3: PWMxAOE: PWM ADDITIONAL OUTPUT ENABLE BITS**

| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| | | | | PWMxOE3 | PWMxOE2 | PWMxOE1 | PWMxOE0 |
| 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 **PWMxOE<3:0>:** PWM Additional Output Channel Enable bits
If bit PWMxOEy is set, PWMxy pin will drive PWM output. Output is independent from PWMxOE bit in PWMxCON

**REGISTER 22-4: PWMxDCH: PWM DUTY CYCLE HIGH BITS**

| 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 |
|---|---|---|---|---|---|---|---|
| PWMxDCH<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 **PWMxDCH<7:0>:** PWM Duty Cycle Most Significant bits
These bits are the MSbs of the PWM duty cycle. The two LSbs are found in the PWMxDCL register.

**REGISTER 22-5: PWMxDCL: PWM DUTY CYCLE LOW BITS**

| R/W-x/u | R/W-x/u | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| PWMxDCL<7:6> | | — | — | — | — | — | — |
| 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 **PWMxDCL<7:6>:** PWM Duty Cycle Least Significant bits
These bits are the LSbs of the PWM duty cycle. The MSbs are found in the PWMxDCH register.

bit 5-0 **Unimplemented:** Read as ‘0’

**TABLE 22-4: SUMMARY OF REGISTERS ASSOCIATED WITH PWM**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| PR2 | Timer2 module Period Register | | | | | | | | 193* |
| PWM1CON | PWM1EN | PWM1OE | PWM1OUT | PWM1POL | — | — | — | — | 286 |
| PWM1DCH | PWM1DCH<7:0> | | | | | | | | 287 |
| PWM1DCL | PWM1DCL<7:6> | | — | — | — | — | — | — | 287 |
| PWM2CON | PWM2EN | PWM2OE | PWM2OUT | PWM2POL | — | — | — | — | 286 |
| PWM2DCH | PWM2DCH<7:0> | | | | | | | | 287 |
| PWM2DCL | PWM2DCL<7:6> | | — | — | — | — | — | — | 287 |
| T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | 195 |
| TMR2 | Timer2 module Register | | | | | | | | 193* |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 115 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 123 |

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

* Page provides register information.


© 2015-2016 Microchip Technology Inc.

## 23.0 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
- $\overline{MCLR}$/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 "*PIC12(L)F1501/PIC16(L)F150X Memory Programming Specification*" (DS41573).

## 23.1 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 $\overline{MCLR}$/VPP to VIHH.

## 23.2 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 ICSP Low-Voltage Programming Entry mode 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. $\overline{MCLR}$ is brought to VIL.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, $\overline{MCLR}$ must be held at VIL for as long as Program/Verify mode is to be maintained.

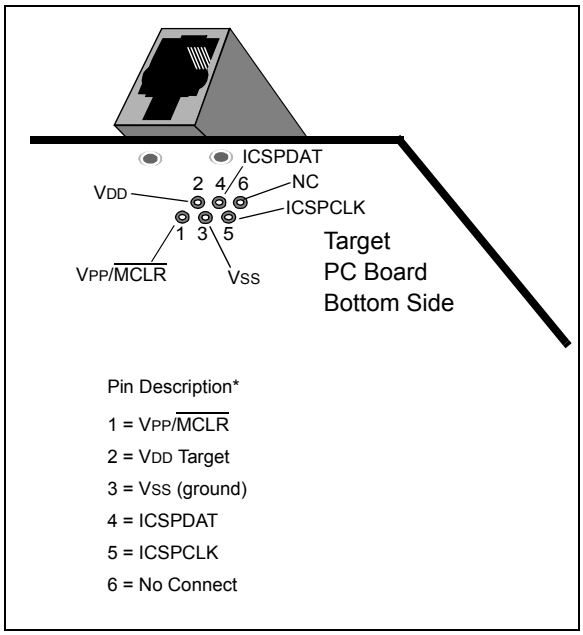
If low-voltage programming is enabled (LVP = 1), the $\overline{MCLR}$ Reset function is automatically enabled and cannot be disabled. See **Section 6.5 "MCLR"** for more information.

The LVP bit can only be reprogrammed to '0' by using the High-Voltage Programming mode.

## 23.3 Common Programming Interfaces

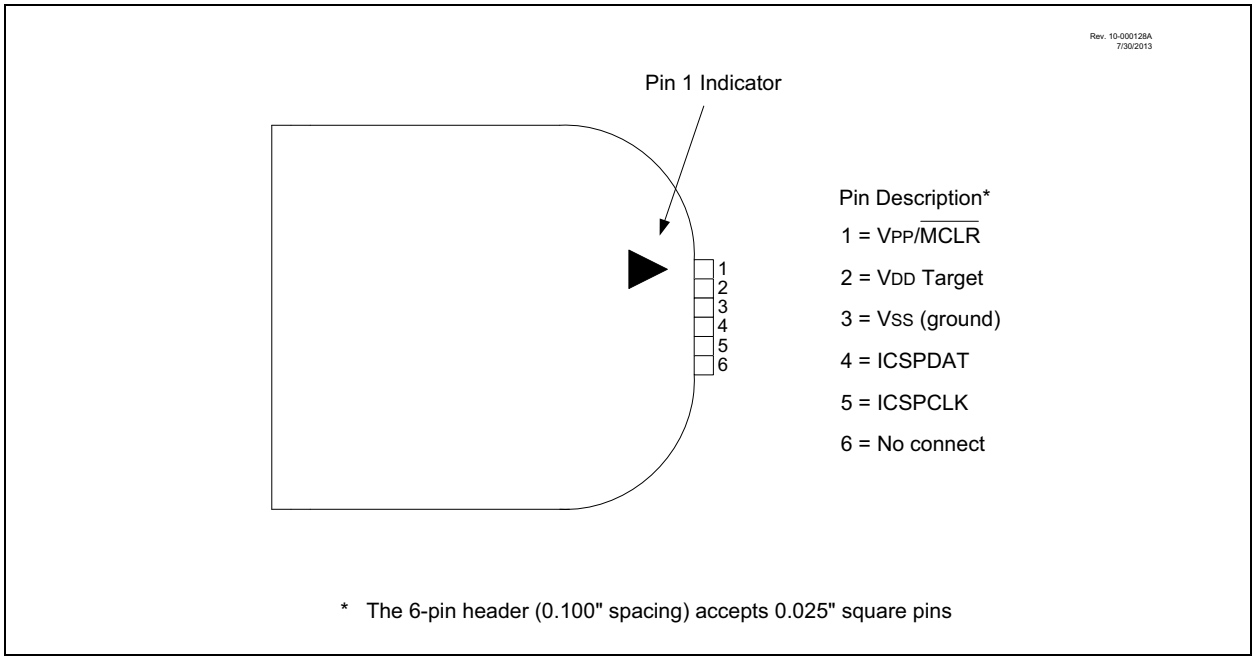
Connection to a target device is typically done through an ICSP™ header. A commonly found connector on development tools is the RJ-11 in the 6P6C (6-pin, 6-connector) configuration. See Figure 23-1.

**FIGURE 23-1: 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 23-2.

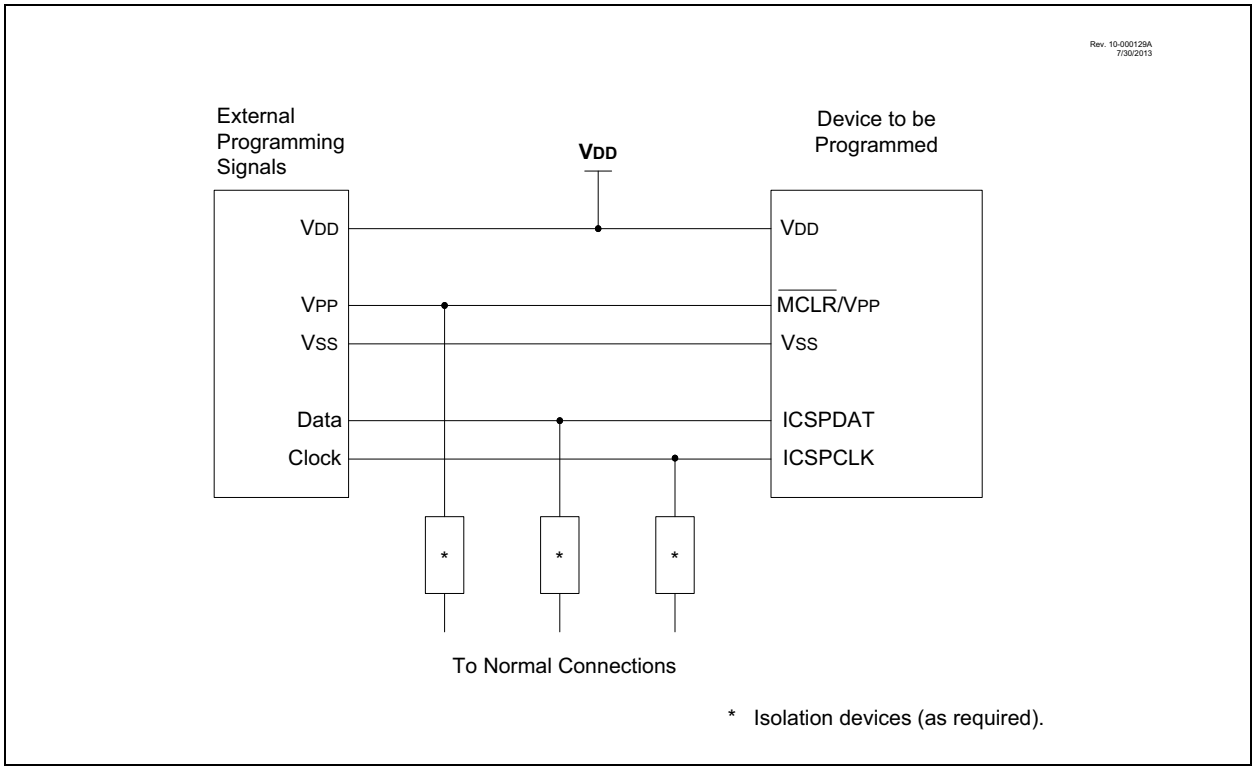
**FIGURE 23-2: PICkit™ PROGRAMMER STYLE CONNECTOR INTERFACE**




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 23-3 for more information.

**FIGURE 23-3: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING**




 © 2015-2016 Microchip Technology Inc.

## 24.0 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 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.

### 24.1 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 24-1: OPCODE FIELD DESCRIPTIONS**

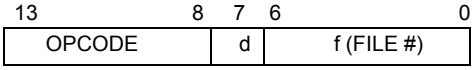
| Field | Description |
|---|---|
| f | Register file address (0x00 to 0x7F) |
| W | Working register (accumulator) |
| b | Bit address within an 8-bit file register |
| k | Literal field, constant data or label |
| x | Don't care location (= 0 or 1).<br>The assembler will generate code with x = 0.<br>It is the recommended form of use for compatibility with all Microchip software tools. |
| d | Destination select; d = 0: store result in W,<br>d = 1: store result in file register f.<br>Default is d = 1. |
| n | FSR or INDF number (0-1) |
| mm | Pre-post increment-decrement mode selection |

**TABLE 24-2: 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 |

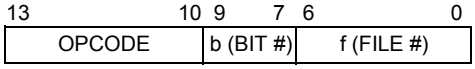
**FIGURE 24-1: GENERAL FORMAT FOR INSTRUCTIONS**

**Byte-oriented file register operations**




d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

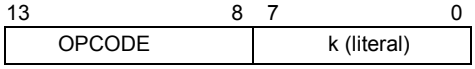
**Bit-oriented file register operations**




b = 3-bit bit address
f = 7-bit file register address

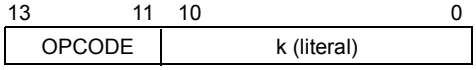
**Literal and control operations**

General



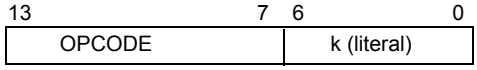

k = 8-bit immediate value

CALL and GOTO instructions only



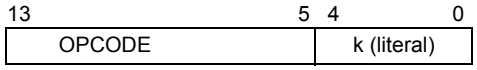

k = 11-bit immediate value

MOVLP instruction only



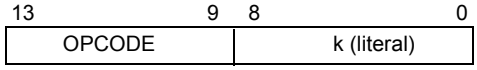

k = 7-bit immediate value

MOVLB instruction only



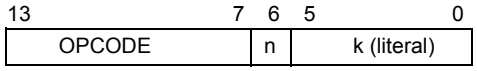

k = 5-bit immediate value

BRA instruction only



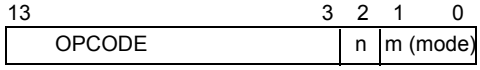

k = 9-bit immediate value

FSR Offset instructions



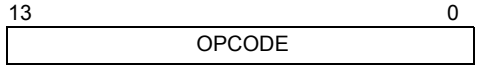

n = appropriate FSR
k = 6-bit immediate value

FSR Increment instructions




n = appropriate FSR
m = 2-bit mode value

OPCODE only




 © 2015-2016 Microchip Technology Inc.

**TABLE 24-3: ENHANCED MID-RANGE 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.

**3:** See Table in the MOVIW and MOVWI instruction descriptions.

**TABLE 24-3: ENHANCED MID-RANGE 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 | – | 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 kkkk | Z | 2, 3 |
| | k[n] | Move INDFn to W, Indexed Indirect. | 1 | 11 | 1111 | 0nkk | 1nmm | Z | 2 |
| MOVWI | n mm | Move W to Indirect FSRn with pre/post inc/dec modifier, mm | 1 | 00 | 0000 | 0001 | kkkk | | 2, 3 |
| | k[n] | Move W to INDFn, Indexed Indirect. | 1 | 11 | 1111 | 1nkk | | | 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.

 © 2015-2016 Microchip Technology Inc.

## 24.2 Instruction Descriptions

**ADDFSR — Add Literal to FSRn**

| | |
|---|---|
| Syntax: | [ *label* ] ADDFSR FSRn, k |
| Operands: | -32 ≤ k ≤ 31<br>n ∈ [ 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 ≤ k ≤ 255 |
| Operation: | (W) + k → (W) |
| Status Affected: | C, DC, Z |
| Description: | The contents of the W register are added to the 8-bit literal ‘k’ and the result is placed in the W register. |

**ADDWF — Add W and f**

| | |
|---|---|
| Syntax: | [ *label* ] ADDWF f,d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [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 ≤ f ≤ 127<br>d ∈ [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**

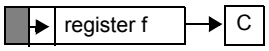
| | |
|---|---|
| Syntax: | [ *label* ] ANDLW k |
| Operands: | 0 ≤ k ≤ 255 |
| Operation: | (W) .AND. (k) → (W) |
| Status Affected: | Z |
| Description: | The contents of W register are AND’ed with the 8-bit literal ‘k’. The result is placed in the W register. |

**ANDWF — AND W with f**

| | |
|---|---|
| Syntax: | [ *label* ] ANDWF f,d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [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 ≤ f ≤ 127<br>d ∈ [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 ≤ f ≤ 127<br>0 ≤ b ≤ 7 |
| Operation: | 0 → (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 ≤ label - PC + 1 ≤ 255<br>-256 ≤ k ≤ 255 |
| Operation: | (PC) + 1 + k → 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 2-cycle instruction. This branch has a limited range. |

**BRW** **Relative Branch with W**

| | |
|---|---|
| Syntax: | [ *label* ] BRW |
| Operands: | None |
| Operation: | (PC) + (W) → 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 2-cycle instruction. |

**BSF** **Bit Set f**

| | |
|---|---|
| Syntax: | [ *label* ] BSF f,b |
| Operands: | 0 ≤ f ≤ 127<br>0 ≤ b ≤ 7 |
| Operation: | 1 → (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 ≤ f ≤ 127<br>0 ≤ b ≤ 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 ≤ f ≤ 127<br>0 ≤ 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. |

 © 2015-2016 Microchip Technology Inc.

**CALL** **Call Subroutine**

| | |
|---|---|
| Syntax: | [ *label* ] CALL k |
| Operands: | $0 \le k \le 2047$ |
| 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 11-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a 2-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 2-cycle instruction. |

**CLRF** **Clear f**

| | |
|---|---|
| Syntax: | [ *label* ] CLRF f |
| Operands: | $0 \le f \le 127$ |
| 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 → $\overline{TO}$<br>1 → $\overline{PD}$ |
| Status Affected: | $\overline{TO}$, $\overline{PD}$ |
| Description: | CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set. |

**COMF** **Complement f**

| | |
|---|---|
| Syntax: | [ *label* ] COMF f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | $(\bar{f}) \rightarrow$ (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: | $0 \le f \le 127$<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: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (f) - 1 → (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. |

**GOTO** **Unconditional Branch**

| | |
|---|---|
| Syntax: | [ *label* ] GOTO k |
| Operands: | $0 \le k \le 2047$ |
| Operation: | k → PC<10:0><br>PCLATH<6:3> → PC<14:11> |
| Status Affected: | None |
| Description: | GOTO is an unconditional branch. The 11-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a 2-cycle instruction. |

**INCF** **Increment f**

| | |
|---|---|
| Syntax: | [ *label* ] INCF f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (f) + 1 → (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'. |

**INCFSZ** **Increment f, Skip if 0**

| | |
|---|---|
| Syntax: | [ *label* ] INCFSZ f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (f) + 1 → (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. |

**IORLW** **Inclusive OR literal with W**

| | |
|---|---|
| Syntax: | [ *label* ] IORLW k |
| Operands: | $0 \le k \le 255$ |
| Operation: | (W) .OR. k → (W) |
| Status Affected: | Z |
| Description: | The contents of the W register are OR'ed with the 8-bit literal 'k'. The result is placed in the W register. |

**IORWF** **Inclusive OR W with f**

| | |
|---|---|
| Syntax: | [ *label* ] IORWF f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (W) .OR. (f) → (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'. |

 © 2015-2016 Microchip Technology Inc.

**LSLF** **Logical Left Shift**

Syntax: [ *label* ] LSLF f {,d}

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f<7>) → C
(f<6:0>) → dest<7:1>
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'.

**LSRF** **Logical Right Shift**

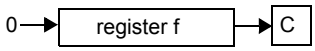
Syntax: [ *label* ] LSRF f {,d}

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: 0 → dest<7>
(f<7:1>) → dest<6:0>,
(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'.




**MOVF** **Move f**

Syntax: [ *label* ] MOVF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

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

**MOVIW** **Move INDFn to W**

Syntax: [ *label* ] MOVIW ++FSRn
[ *label* ] MOVIW --FSRn
[ *label* ] MOVIW FSRn++
[ *label* ] MOVIW FSRn--
[ *label* ] MOVIW k[FSRn]

Operands: n ∈ [0,1]
mm ∈ [00,01, 10, 11]
-32 ≤ k ≤ 31

Operation: INDFn → W
Effective address is determined by
- FSR + 1 (preincrement)
- FSR - 1 (predecrement)
- FSR + k (relative offset)

After the Move, the FSR value will be either:
- FSR + 1 (all increments)
- FSR - 1 (all decrements)
- 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.

**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.

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 ≤ k ≤ 31

Operation: k → BSR

Status Affected: None

Description: The 5-bit literal 'k' is loaded into the Bank Select Register (BSR).

**MOVLP** **Move literal to PCLATH**

Syntax: [ *label* ] MOVLP k

Operands: 0 ≤ k ≤ 127

Operation: k → PCLATH

Status Affected: None

Description: The 7-bit literal 'k' is loaded into the PCLATH register.

**MOVLW** **Move literal to W**

Syntax: [ *label* ] MOVLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (W)

Status Affected: None

Description: The 8-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 ≤ f ≤ 127

Operation: (W) → (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

 © 2015-2016 Microchip Technology Inc.

**MOVWI** **Move W to INDFn**

Syntax: [ *label* ] MOVWI ++FSRn
[ *label* ] MOVWI --FSRn
[ *label* ] MOVWI FSRn++
[ *label* ] MOVWI FSRn--
[ *label* ] MOVWI k[FSRn]

Operands: n ∈ [0,1]
mm ∈ [00,01, 10, 11]
-32 ≤ k ≤ 31

Operation: W → INDFn
Effective address is determined by
- FSR + 1 (preincrement)
- FSR - 1 (predecrement)
- FSR + k (relative offset)

After the Move, the FSR value will be either:
- FSR + 1 (all increments)
- FSR - 1 (all decrements)

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.

**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.

FSRn is limited to the range 0000h - FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wrap-around.

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) → OPTION_REG

Status Affected: None

Description: Move data from W register to OPTION_REG register.

**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,
1 → 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 2-cycle instruction.

Words: 1

Cycles: 2

Example:

```
RETFIE
```

After Interrupt
PC = TOS
GIE = 1

**RETLW** **Return with literal in W**

Syntax: [ *label* ] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (W);
TOS → PC

Status Affected: None

Description: The W register is loaded with the 8-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a 2-cycle instruction.

Words: 1

Cycles: 2

Example:

```
CALL TABLE;W contains table
    ;offset value
•   ;W now has table value
•
•
ADDWF PC  ;W = offset
RETLW k1  ;Begin table
RETLW k2  ;
•
•
•
RETLW kn  ; End of table
```

TABLE

Before Instruction
W = 0x07
After Instruction
W = value of k8

**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 2-cycle instruction.

**RLF** **Rotate Left f through Carry**

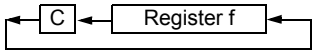
Syntax: [ *label* ] RLF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [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

 © 2015-2016 Microchip Technology Inc.

**RRF** **Rotate Right f through Carry**

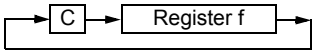
Syntax: [ *label* ] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

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 → WDT,
0 → WDT prescaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

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.
The processor is put into Sleep mode with the oscillator stopped.

**SUBLW** **Subtract W from literal**

Syntax: [ *label* ] SUBLW k

Operands: $0 \le k \le 255$

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Description: The W register is subtracted (2's complement method) from the 8-bit literal 'k'. The result is placed in the W register.

| | |
|---|---|
| C = 0 | W > k |
| C = 1 | W ≤ k |
| DC = 0 | W<3:0> > k<3:0> |
| DC = 1 | W<3:0> ≤ k<3:0> |

**SUBWF** **Subtract W from f**

Syntax: [ *label* ] SUBWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - (W) → (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 ≤ f |
| DC = 0 | W<3:0> > f<3:0> |
| DC = 1 | W<3:0> ≤ f<3:0> |

**SUBWFB** **Subtract W from f with Borrow**

Syntax: SUBWFB f {,d}

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) – (W) – ($\overline{B}$) → 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: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (f<3:0>) → (destination<7:4>),<br>(f<7:4>) → (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: | $5 \le f \le 7$ |
| Operation: | (W) → 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: | $0 \le k \le 255$ |
| Operation: | (W) .XOR. k → (W) |
| Status Affected: | Z |
| Description: | The contents of the W register are XOR'ed with the 8-bit literal 'k'. The result is placed in the W register. |

| XORWF | Exclusive OR W with f |
|---|---|
| Syntax: | [ *label* ] XORWF f,d |
| Operands: | $0 \le f \le 127$<br>$d \in [0,1]$ |
| Operation: | (W) .XOR. (f) → (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'. |

 © 2015-2016 Microchip Technology Inc.

# 25.0 ELECTRICAL SPECIFICATIONS

## 25.1 Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +125°C

Storage temperature ..... -65°C to +150°C

Voltage on pins with respect to $V_{SS}$

- on $V_{DD}$ pin ..... -0.3V to +4.0V
- on $\overline{MCLR}$ pin ..... -0.3V to +9.0V
- on all other pins ..... -0.3V to ($V_{DD}$ + 0.3V)

Total power dissipation(2) ..... 800 mW

Maximum current

- on $V_{SS}$ pin(1), -40°C ≤ $T_A$ ≤ +85°C for industrial ..... 350 mA
- on $V_{SS}$ pin(1), +85°C ≤ $T_A$ ≤ +125°C for extended ..... 120 mA
- on $V_{DD}$ pin(1), -40°C ≤ $T_A$ ≤ +85°C for industrial ..... 250 mA
- on $V_{DD}$ pin(1), +85°C ≤ $T_A$ ≤ +125°C for extended ..... 85 mA
- sunk by any I/O pin ..... 50 mA
- sourced by any I/O pin ..... 50 mA

Clamp current, $I_K$ ($V_{PIN}$ < 0 or $V_{PIN}$ > $V_{DD}$) ..... ± 20 mA

**Note 1:** Maximum current rating requires even load distribution across I/O pins. Maximum current rating may be limited by the device package power dissipation characterizations, see Table 25-6 to calculate device specifications.

**2:** Power dissipation is calculated as follows: $P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$.

† 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.

## 25.2 Standard Operating Conditions

The standard operating conditions for any device are defined as:

Operating Voltage: $V_{DDMIN} \le V_{DD} \le V_{DDMAX}$
Operating Temperature: $T_{A_MIN} \le T_A \le T_{A_MAX}$

**$V_{DD}$ — Operating Supply Voltage(1)**

- $V_{DDMIN}$ (Fosc ≤ 16 MHz) ..... +1.8V
- $V_{DDMIN}$ (16 MHz < Fosc ≤ 32 MHz) ..... +2.5V
- $V_{DDMAX}$ ..... +3.6V

**$T_A$ — Operating Ambient Temperature Range**

Industrial Temperature

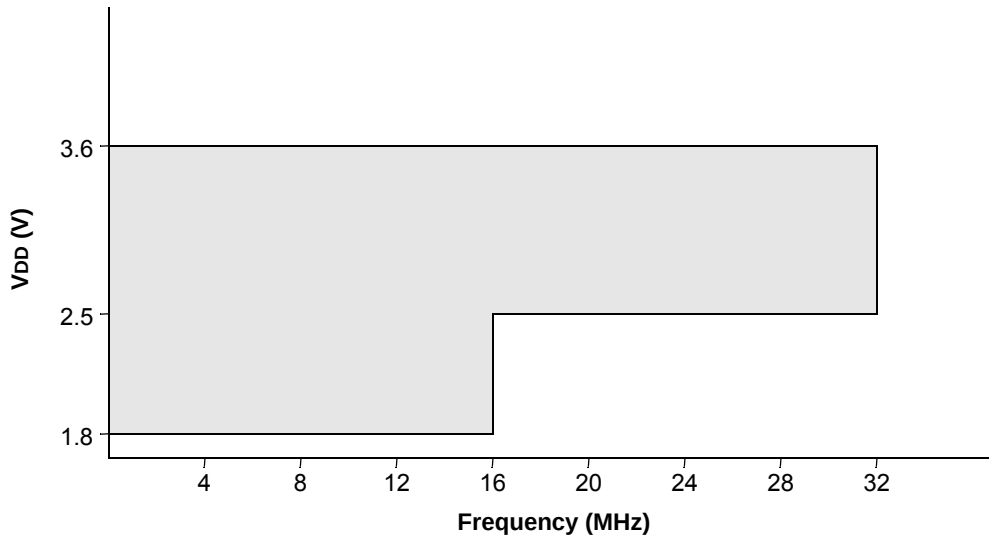
- $T_{A_MIN}$ ..... -40°C
- $T_{A_MAX}$ ..... +85°C

Extended Temperature

- $T_{A_MIN}$ ..... -40°C
- $T_{A_MAX}$ ..... +125°C

**Note 1:** See Parameter D001 in DC Characteristics: Supply Voltage.

**FIGURE 25-1: PIC16LF1566/1567 VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C**




**Note 1:** The shaded region indicates the permissible combinations of voltage and frequency.
**2:** Refer to Table 25-7 for each Oscillator mode's supported frequencies.

 © 2015-2016 Microchip Technology Inc.

## 25.3 DC Characteristics

### TABLE 25-1: SUPPLY VOLTAGE

| PIC16LF1566/1567 | | | 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)** | | | | | |
| | | | 1.8<br>2.5 | —<br>— | 3.6<br>3.6 | V<br>V | FOSC ≤ 16 MHz:<br>FOSC ≤ 32 MHz |
| D002* | VDR | **RAM Data Retention Voltage(1)** | 1.5 | — | — | V | Device in Sleep mode |
| D002A* | VPOR* | **Power-on Reset Release Voltage** | — | 1.6 | — | V | |
| D002B* | VPORR* | **Power-on Reset Rearm Voltage** | — | 0.8 | — | V | |
| D003 | VADFVR | **Fixed Voltage Reference Voltage for ADC, Initial Accuracy** | -7<br>-8<br>-7<br>-8 | —<br>—<br>—<br>— | 6<br>6<br>6<br>6 | % | 1.024V, VDD ≥ 2.5V, 85°C (**Note 2**)<br>1.024V, VDD ≥ 2.5V, 125°C (**Note 2**)<br>2.048V, VDD ≥ 2.5V, 85°C<br>2.048V, VDD ≥ 2.5V, 125°C |
| D003C* | TCVFVR | **Temperature Coefficient, Fixed Voltage Reference** | — | -130 | — | ppm/°C | |
| D003D* | ΔVFVR/ ΔVIN | **Line Regulation, Fixed Voltage Reference** | — | 0.270 | — | %/V | |
| D004* | SVDD | **VDD Rise Rate** to ensure internal Power-on Reset signal | 0.05 | — | — | V/ms | See **Section 6.1 "Power-on Reset (POR)"** for details. |
| D005* | VI2CLVL | I2CLVL Voltage | TBD | — | VDD | V | — |

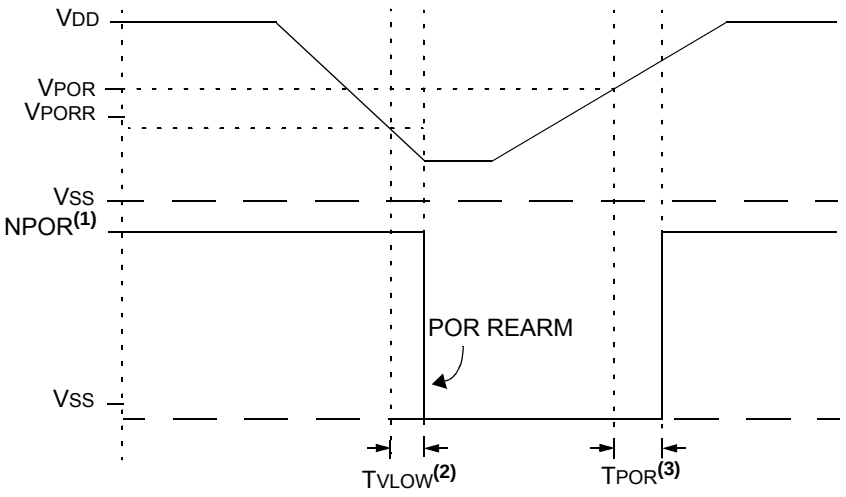
* These parameters are characterized but not tested.

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.

**2:** For proper operation, the minimum value of the ADC positive voltage reference must be 1.8V or greater. When selecting the FVR or the VREF+ pin as the source of the ADC positive voltage reference, be aware that the voltage must be 1.8V or greater.

**FIGURE 25-2: POR AND POR REARM WITH SLOW RISING VDD**




**Note 1:** When NPOR is low, the device is held in Reset.
**2:** TPOR 1 µs typical.
**3:** TVLOW 2.7 µs typical.

**TABLE 25-2: SUPPLY CURRENT (IDD)**

| PIC16LF1566/1567 | | | 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. | Device Characteristics | Min. | Typ.† | Max. | Units | Conditions | |
| | | | | | | VDD | Note |
| | **Supply Current (IDD)(1, 2)** | | | | | | |
| D010 | | — | 2.5 | 18 | µA | 1.8 | FOSC = 31 kHz<br>LFINTOSC mode |
| | | — | 4 | 20 | µA | 3.0 | |
| D011 | | — | 0.35 | 0.70 | mA | 1.8 | FOSC = 8 MHz<br>HFINTOSC mode |
| | | — | 0.55 | 1.10 | mA | 3.0 | |
| D012 | | — | 0.5 | 1.2 | mA | 1.8 | FOSC = 16 MHz<br>HFINTOSC mode |
| | | — | 0.8 | 1.75 | mA | 3.0 | |
| D013 | | — | 1.5 | 3.5 | mA | 3.0 | FOSC = 32 MHz<br>HFINTOSC mode with PLL |
| D014 | | — | 3 | 17 | µA | 1.8 | FOSC = 32 kHz<br>ECL mode |
| | | — | 5 | 20 | µA | 3.0 | |
| D015 | | — | 12 | 40 | µA | 1.8 | FOSC = 500 kHz<br>ECL mode |
| | | — | 18 | 60 | µA | 3.0 | |
| D016 | | — | 25 | 65 | µA | 1.8 | FOSC = 1 MHz<br>ECM mode |
| | | — | 40 | 100 | µA | 3.0 | |

† Data in "Typ." column is at 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The test conditions for all IDD measurements in active operation mode are: CLKIN = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD; $\overline{MCLR}$ = 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.

 © 2015-2016 Microchip Technology Inc.

**TABLE 25-2: SUPPLY CURRENT ($I_{DD}$) (CONTINUED)**

| PIC16LF1566/1567 | | | 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. | Device Characteristics | Min. | Typ.† | Max. | Units | Conditions | |
| | | | | | | VDD | Note |
| D017 | | — | 80 | 250 | μA | 1.8 | FOSC = 4 MHz<br>ECM mode |
| | | — | 135 | 430 | μA | 3.0 | |
| D018 | | — | 0.7 | 1.5 | mA | 3.0 | FOSC = 20 MHz<br>ECH mode |

† Data in "Typ." column is at 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The test conditions for all $I_{DD}$ measurements in active operation mode are: CLKIN = 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.

**TABLE 25-3: POWER-DOWN CURRENTS ($I_{PD}$)**

| PIC16LF1566/1567 | | | 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. | Device Characteristics | Min. | Typ.† | Max. +85°C | Max. +125°C | Units | Conditions | |
| | | | | | | | $V_{DD}$ | Note |
| | **Power-down Base Current ($I_{PD}$)(2)** | | | | | | | |
| D020 | | — | 0.02 | 1.0 | 8 | µA | 1.8 | WDT, BOR, FVR, and T1OSC disabled, all Peripherals Inactive |
| | | — | 0.03 | 2 | 9 | µA | 3.0 | |
| D021 | | — | 0.3 | 2 | 9 | µA | 1.8 | LPWDT Current **(Note 1)** |
| | | — | 0.4 | 3 | 10 | µA | 3.0 | |
| D022 | | — | 13 | 28 | 30 | µA | 1.8 | FVR current **(Note 1)** |
| | | — | 22 | 30 | 33 | µA | 3.0 | |
| D023 | | — | 6.5 | 17 | 20 | µA | 3.0 | BOR Current **(Note 1)** |
| D024 | | — | 0.1 | 4 | 10 | µA | 3.0 | LPBOR Current |
| D025 | | — | 0.03 | 3.5 | 9 | µA | 1.8 | ADC Current **(Note 1, Note 3)**, no conversion in progress |
| | | — | 0.04 | 4.0 | 10 | µA | 3.0 | |
| D026* | | — | 350 | — | — | µA | 1.8 | ADC Current **(Note 1, Note 4)**, conversion in progress |
| | | — | 350 | — | — | µA | 3.0 | |

* These parameters are characterized but not tested.

† Data in "Typ." column is at 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The peripheral current is the sum of the base $I_{DD}$ or $I_{PD}$ and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base $I_{DD}$ or $I_{PD}$ 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 $V_{DD}$.

**3:** ADC oscillator source is FRC.

**4:** Only one of the two ADCs is on.

 © 2015-2016 Microchip Technology Inc.

**TABLE 25-4: I/O PORTS**

| 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** |
| | | **Input Low Voltage** | | | | | |
| | | I/O PORT: | | | | | |
| D030 | | with TTL buffer | — | — | 0.15 VDD | V | 1.8V ≤ VDD ≤ 3.6V |
| D031 | VIL | with Schmitt Trigger buffer | — | — | 0.2 VDD | V | 2.0V ≤ VDD ≤ 3.6V |
| | | with SMBus levels | — | — | 0.8 | V | 3.0V ≤ VDD ≤ 3.6V |
| | | with I2CLVL enabled | — | — | 0.3 VI2CLVL | V | TBD ≤ VI2CLVL ≤ VDD |
| D032 | | $\overline{MCLR}$ | — | — | 0.2 VDD | V | |
| | VIH | **Input High Voltage** | | | | | |
| | | I/O ports: | | — | — | | |
| D040 | | with TTL buffer | 0.25 VDD + 0.8 | — | — | V | 1.8V ≤ VDD ≤ 3.6V |
| D041 | | with Schmitt Trigger buffer | 0.8 VDD | — | — | V | 2.0V ≤ VDD ≤ 3.6V |
| | | with SMBus levels | 2.1 | — | — | V | 3.0V ≤ VDD ≤ 3.6V |
| | | with I2CLVL enabled | 0.7 VI2CLVL | — | — | V | TBD ≤ VI2CLVL ≤ VDD |
| D042 | | $\overline{MCLR}$ | 0.8 VDD | — | — | V | |
| | IIL | **Input Leakage Current(1)** | | | | | |
| D060 | | I/O ports | — | ± 5 | ± 125 | nA | VSS ≤ VPIN ≤ VDD, Pin at high-impedance at 85°C |
| | | | | ± 5 | ± 1000 | nA | 125°C |
| D061 | | $\overline{MCLR}$(2) | — | ± 50 | ± 200 | nA | VSS ≤ VPIN ≤ VDD at 85°C |
| | IPUR | **Weak Pull-up Current** | | | | | |
| D070* | | | 25 | 100 | 200 | µA | VDD = 3.3V, VPIN = VSS |
| | VOL | **Output Low Voltage(3)** | | | | | |
| D080 | | I/O ports | — | — | 0.6 | V | IOL = 6mA, VDD = 3.3V<br>IOL = 1.8mA, VDD = 1.8V |
| | VOH | **Output High Voltage(3)** | | | | | |
| D090 | | I/O ports | VDD - 0.7 | — | — | V | IOH = 3mA, VDD = 3.3V<br>IOH = 1mA, VDD = 1.8V |
| | | **Capacitive Loading Specs on Output Pins** | | | | | |
| D101A* | CIO | All I/O pins | — | — | 50 | pF | |

* These parameters are characterized but not tested.

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Negative current is defined as current sourced by the pin.

**2:** 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.

**3:** Including OSC2 in CLKOUT mode.

**TABLE 25-5: MEMORY PROGRAMMING SPECIFICATIONS**

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +125°C | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| | | **Program Memory Programming Specifications** | | | | | |
| D110 | VIHH | Voltage on $\overline{MCLR}$/VPP pin | 8.0 | — | 9.0 | V | **(Note 2)** |
| D111 | IDDP | Supply Current during Programming | — | — | 10 | mA | |
| D112 | VBE | VDD for Bulk Erase | 2.7 | — | VDDMAX | V | |
| D113 | VPEW | VDD for Write or Row Erase | VDDMIN | — | VDDMAX | V | |
| D114 | IPPPGM | Current on $\overline{MCLR}$/VPP during Erase/Write | — | — | 1.0 | mA | |
| D115 | IDDPGM | Current on VDD during Erase/Write | — | — | 5.0 | mA | |
| | | Program Flash Memory | | | | | |
| D121 | EP | Cell Endurance | 10K | — | — | E/W | -40°C to +85°C **(Note 1)** |
| D122 | VPRW | VDD for Read/Write | 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 |
| D125 | EHEFC | High-Endurance Flash Cell | 100K | — | — | E/W | 0°C to +60°C, Lower byte, Last 128 Addresses in Flash Memory |

† 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:** Required only if single-supply programming is disabled.

 © 2015-2016 Microchip Technology Inc.

**TABLE 25-6: THERMAL CHARACTERISTICS**

| **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +125°C | | | | | |
|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Typ.** | **Units** | **Conditions** |
| TH01 | θJA | Thermal Resistance Junction to Ambient | 60.0 | °C/W | 28-pin SPDIP package |
| | | | 80.3 | °C/W | 28-pin SOIC package |
| | | | 90.0 | °C/W | 28-pin SSOP package |
| | | | 48.0 | °C/W | 28-pin UQFN (4x4 mm) package |
| | | | 47.2 | °C/W | 40-pin PDIP package |
| | | | 46.0 | °C/W | 44-pin TQFP package |
| | | | 41.0 | °C/W | 40-pin UQFN (5x5 mm) package |
| TH02 | θJC | Thermal Resistance Junction to Case | 31.4 | °C/W | 28-pin SPDIP package |
| | | | 24.0 | °C/W | 28-pin SOIC package |
| | | | 24.0 | °C/W | 28-pin SSOP package |
| | | | 12.0 | °C/W | 28-pin UQFN (4x4 mm) package |
| | | | 24.7 | °C/W | 40-pin PDIP package |
| | | | 14.5 | °C/W | 44-pin TQFP package |
| | | | 50.5 | °C/W | 40-pin UQFN (5x5 mm) 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; TJ = Junction Temperature.

## 25.4 AC Characteristics

Timing Parameter Symbology has been created with one of the following formats:

1. TppS2ppS
2. TppS

| T | | | |
|---|---|---|---|
| F | Frequency | T | Time |

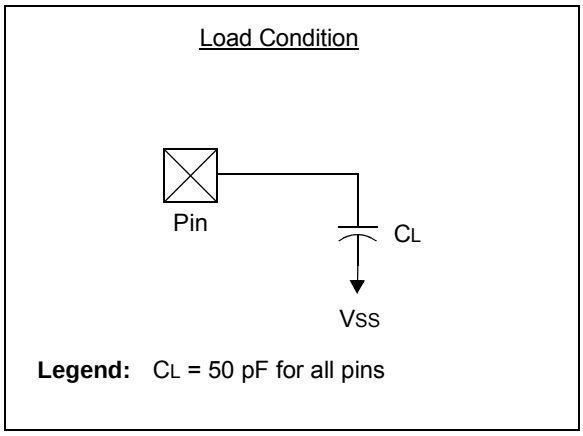
Lowercase letters (pp) and their meanings:

| pp | | | |
|---|---|---|---|
| cc | CCP1 | osc | CLKIN |
| ck | CLKOUT | rd | $\overline{RD}$ |
| cs | $\overline{CS}$ | rw | $\overline{RD}$ or $\overline{WR}$ |
| di | SDIx | sc | SCKx |
| 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:

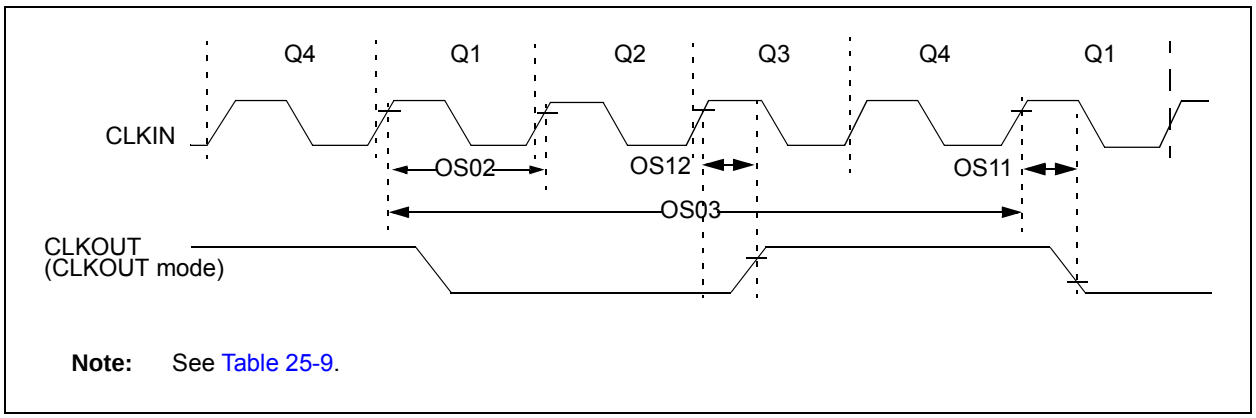
| S | | | |
|---|---|---|---|
| F | Fall | P | Period |
| H | High | R | Rise |
| I | Invalid (High-impedance) | V | Valid |
| L | Low | Z | High-impedance |

**FIGURE 25-3: LOAD CONDITIONS**




**Legend:** CL = 50 pF for all pins

**FIGURE 25-4: CLOCK TIMING**




**Note:** See Table 25-9.

 © 2015-2016 Microchip Technology Inc.

**TABLE 25-7: 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) |
| OS02 | TOSC | External CLKIN Period(1) | 50 | — | ∞ | ns | EC mode |
| OS03 | TCY | Instruction Cycle Time(1) | 200 | — | DC | ns | TCY = FOSC/4 |

* 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 CLKIN pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

**TABLE 25-8: OSCILLATOR PARAMETERS**

| **Standard Operating Conditions (unless otherwise stated)**<br>Operating Temperature -40°C ≤ TA ≤ +125°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| OS08 | HFOSC | Internal Calibrated HFINTOSC Frequency(1) | — | 16.0 | — | MHz | 0°C ≤ TA ≤ +85°C |
| OS08A | HFTOL | Frequency Tolerance | — | ±3 | — | % | 25°C, 16 MHz |
| | | | — | ±6 | — | % | 0°C ≤ TA ≤ +85°C, 16 MHz |
| OS09 | LFOSC | Internal LFINTOSC Frequency | — | 31 | — | kHz | -40°C ≤ TA ≤ +125°C |
| OS10* | TWARM | HFINTOSC<br>Wake-up from Sleep Start-up Time | — | 5 | 15 | µs | |
| | | LFINTOSC<br>Wake-up from Sleep Start-up Time | — | 0.5 | — | ms | |

* These parameters are characterized but not tested.

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** 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.

**FIGURE 25-5: CLKOUT AND I/O TIMING**

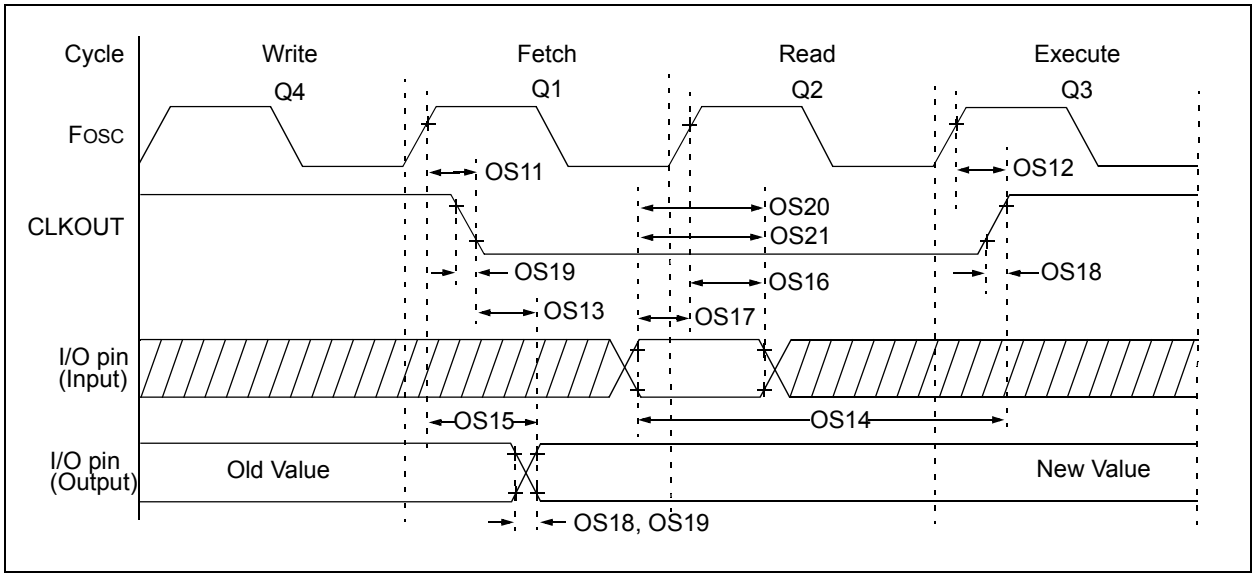



**TABLE 25-9: CLKOUT AND I/O TIMING PARAMETERS**

| **Standard Operating Conditions (unless otherwise stated)**<br>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 = 3.3-3.6V |
| OS12 | TosH2ckH | FOSC↑ to CLKOUT↑ (1) | — | — | 72 | ns | VDD = 3.3-3.6V |
| 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 = 3.3-3.6V |
| OS16 | TosH2ioI | Fosc↑ (Q2 cycle) to Port input invalid (I/O in hold time) | 50 | — | — | ns | VDD = 3.3-3.6V |
| OS17 | TioV2osH | Port input valid to Fosc↑ (Q2 cycle) (I/O in setup time) | 20 | — | — | ns | |
| OS18* | TioR | Port output rise time | — | 15 | 32 | ns | VDD = 2.0V |
| OS19* | TioF | Port output fall time | — | 28 | 55 | ns | VDD = 2.0V |
| OS20* | Tinp | INT pin input high or low time | 25 | — | — | ns | |
| OS21* | Tioc | Interrupt-on-Change new input level time | 25 | — | — | ns | |

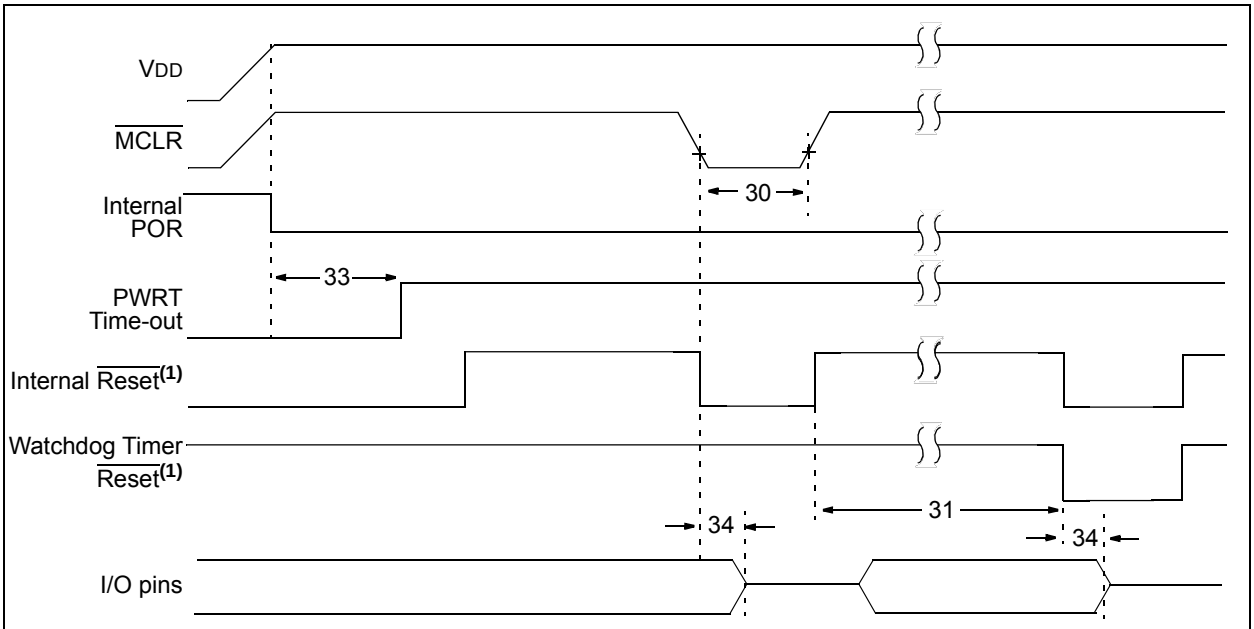
* These parameters are characterized but not tested.

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated.

**Note 1:** Measurements are taken in EC mode where CLKOUT output is 4 x TOSC.

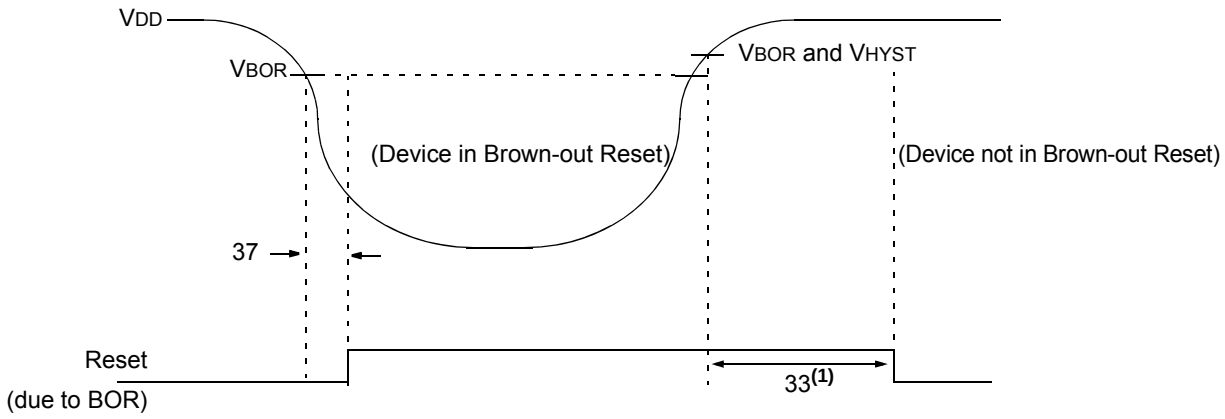
 © 2015-2016 Microchip Technology Inc.

**FIGURE 25-6: RESET, WATCHDOG TIMER AND POWER-UP TIMER TIMING**




**Note 1:** Asserted low.

**FIGURE 25-7: BROWN-OUT RESET TIMING AND CHARACTERISTICS**




**Note 1:** 64 ms delay only if $\overline{\text{PWRTE}}$ bit in the Configuration Words is programmed to '0'.
2 ms delay if $\overline{\text{PWRTE}}$ = 0.

**TABLE 25-10: RESET, WATCHDOG TIMER, POWER-UP TIMER AND BROWN-OUT RESET PARAMETERS**

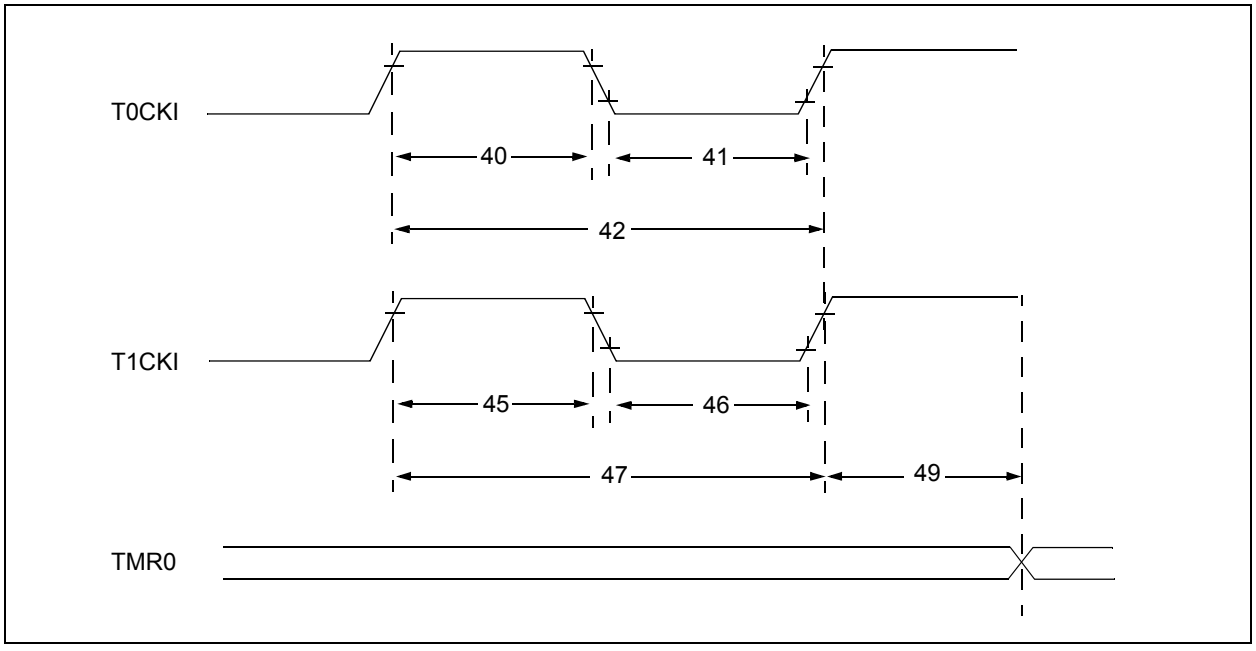
| **Standard Operating Conditions (unless otherwise stated)**<br>Operating Temperature -40°C ≤ TA ≤ +125°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| 30 | TMCL | $\overline{MCLR}$ Pulse Width (low) | 2<br>5 | —<br>— | —<br>— | µs<br>µs | -40°C to +85°C<br>+85°C to +125°C |
| 31 | TWDTLP | Low-Power Watchdog Timer Time-out Period | 10 | 16 | 27 | ms | VDD = 3.3V-3.6V, 1:512 Prescaler used |
| 33* | TPWRT | Power-up Timer Period, $\overline{PWRTE}$ = 0 | 40 | 65 | 140 | ms | |
| 34* | TIOZ | I/O high-impedance from $\overline{MCLR}$ Low or Watchdog Timer Reset | — | — | 2.0 | µs | |
| 35 | VBOR | Brown-out Reset Voltage(1) | 2.55<br>1.80 | 2.70<br>1.90 | 2.85<br>2.05 | V<br>V | BORV = 0<br>BORV = 1 |
| 35A | VLPBOR | Low-Power Brown-out | 1.8 | 2.1 | 2.5 | V | LPBOR = 1 |
| 36* | VHYST | Brown-out Reset Hysteresis | 0 | 25 | 50 | 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:** 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 25-8: TIMER0 EXTERNAL CLOCK TIMINGS**






© 2015-2016 Microchip Technology Inc.

**TABLE 25-11: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| 40* | TT0H | T0CKI High Pulse Width | No Prescaler | 0.5 TCY + 20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 41* | TT0L | T0CKI Low Pulse Width | No Prescaler | 0.5 TCY + 20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 42* | TT0P | T0CKI Period | | Greater of: 20 or TCY + 40 / N | — | — | ns | N = prescale value |
| 45* | TT1H | T1CKI High Time | Synchronous, No Prescaler | 0.5 TCY + 20 | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 46* | TT1L | T1CKI Low Time | Synchronous, No Prescaler | 0.5 TCY + 20 | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 47* | TT1P | T1CKI Input Period | Synchronous | Greater of: 30 or TCY + 40 / N | — | — | ns | N = prescale value |
| | | | Asynchronous | 60 | — | — | ns | |
| 48* | TCKEZTMR1 | Delay from External Clock Edge to Timer Increment | | 2 TOSC | — | 7 TOSC | — | Timers in Sync mode |

* These parameters are characterized but not tested.

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**TABLE 25-12: PIC16LF1566/1567 ANALOG-TO-DIGITAL CONVERTER (ADC) CHARACTERISTICS(1,2,3)**

| Standard Operating Conditions (unless otherwise stated)<br>Operating temperature Tested at 25°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| AD01 | NR | Resolution | — | — | 10 | bit | |
| AD02 | EIL | Integral Error | — | ±0.4 | ±1 | LSb | -40°C to +85°C, VREF ≥ 2.0V |
| AD03 | EDL | Differential Error | — | ±0.3 | ±1 | LSb | -40°C to +85°C, VREF ≥ 2.0V |
| AD04 | EOFF | Offset Error | — | 1.2 | ±3 | LSb | -40°C to +85°C, VREF ≥ 2.0V |
| AD05 | EGN | Gain Error | — | 1.0 | ±3 | LSb | -40°C to +85°C, VREF ≥ 2.0V |
| AD06 | VREF | Reference Voltage Range (VREFH – VREFL) | 1.8<br>2.0 | —<br>— | —<br>— | V<br>V | Absolute Minimum **(Note 4)**<br>Minimum for 1LSb Accuracy |
| AD07 | VAIN | Full-Scale Range | VSS | — | VREF | V | |
| AD08 | ZAIN | Recommended Impedance of Analog Voltage Source | — | — | 3 | 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 ADC conversion result never decreases with an increase in the input voltage and has no missing codes.

**3:** 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.

**4:** ADC VREF is selected by ADPREF<1:0> bits.

**TABLE 25-13: PIC16LF1566/1567 ADC CONVERSION REQUIREMENTS**

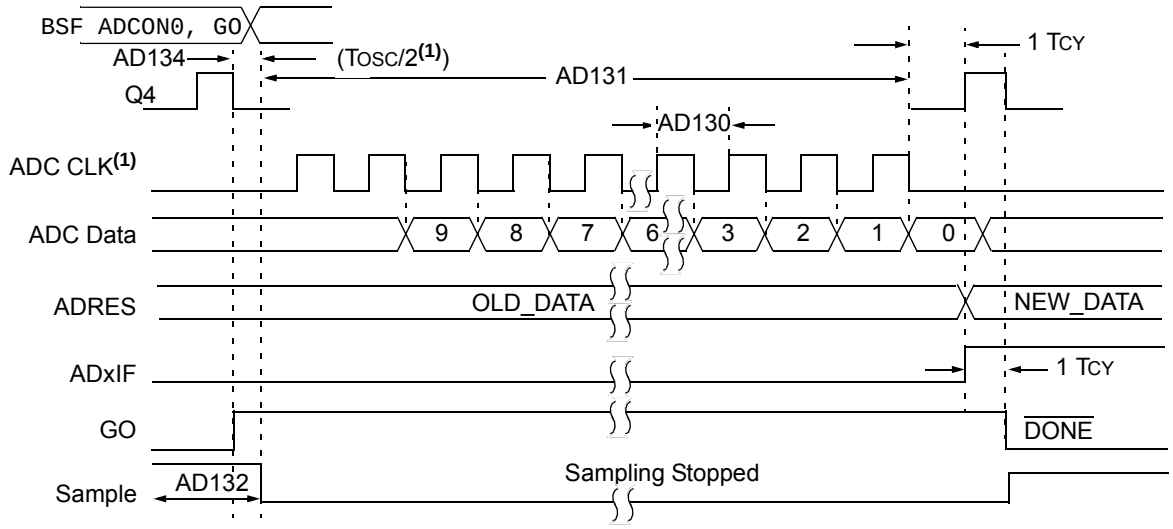
| **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +125°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| AD130* | TAD | ADC Clock Period | 0.25 | — | 25 | µs | TOSC-based, -40°C to +85°C, VREF ≥ 2.4V |
| | | | 0.7 | — | 25 | µs | TOSC-based, -40°C to +85°C, VREF < 2.4V |
| | | | 0.7 | — | 8 | µs | TOSC-based, +86°C to +125°C |
| | | ADC Internal FRC Oscillator Period | 1.0 | 1.6 | 6.0 | µs | ADCS<1:0> = 11 (ADFRC mode) |
| AD131 | TCNV | Conversion Time (not including Acquisition Time)(1) | — | 11 | — | TAD | Set GO/$\overline{\text{DONE}}$x 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.

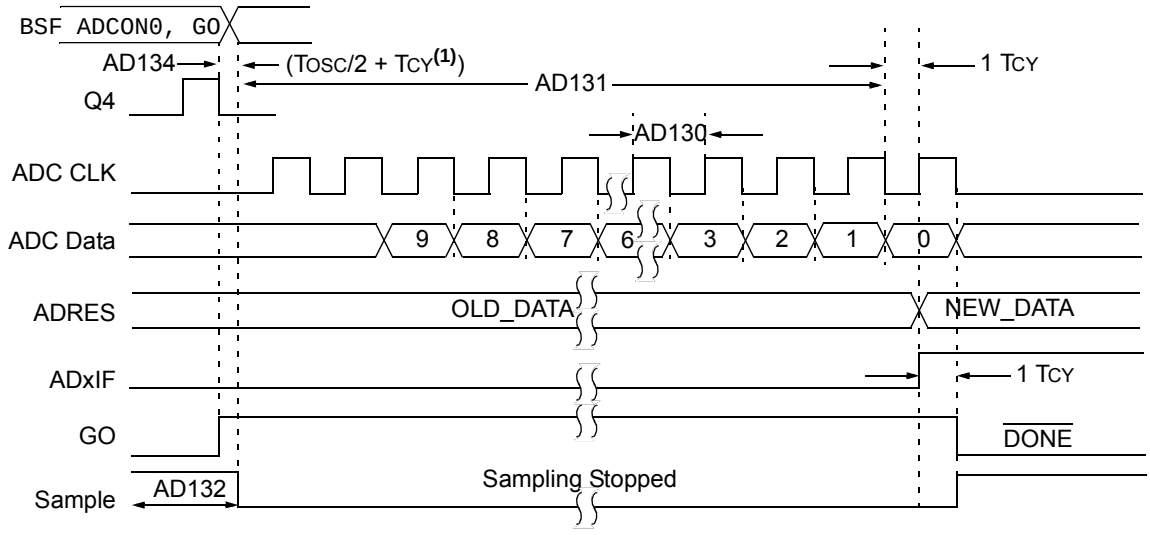
**FIGURE 25-9: PIC16LF1566/1567 ADC CONVERSION TIMING (NORMAL MODE)**




**Note 1:** If the ADC clock source is selected as FRC, a time of TCY is added before the ADC clock starts. This allows the SLEEP instruction to be executed.

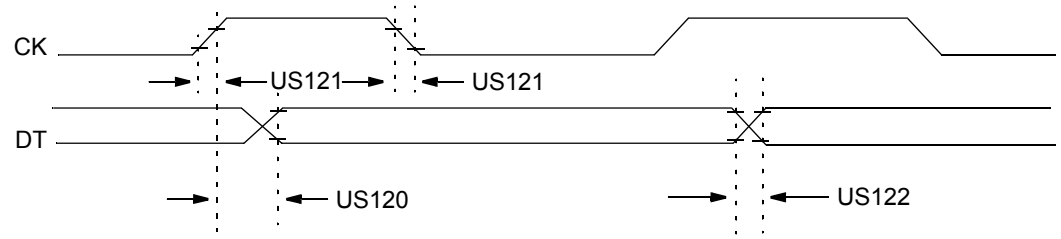
 © 2015-2016 Microchip Technology Inc.

**FIGURE 25-10: PIC16LF1566/1567 ADC CONVERSION TIMING (SLEEP MODE)**




**Note 1:** If the ADC clock source is selected as FRC, a time of TCY is added before the ADC clock starts. This allows the SLEEP instruction to be executed.

**FIGURE 25-11: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

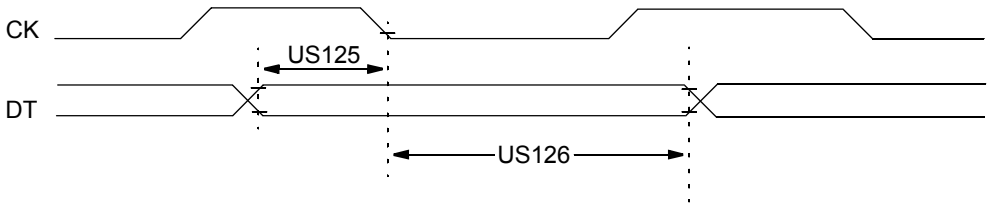



**Note:** Refer to Figure 25-3 for Load conditions.

**TABLE 25-14: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS**

| **Standard Operating Conditions (unless otherwise stated)** | | | | | | |
|---|---|---|---|---|---|---|
| **Param. No.** | **Symbol** | **Characteristic** | **Min.** | **Max.** | **Units** | **Conditions** |
| US120 | TCKH2DTV | SYNC XMIT (Master and Slave) Clock high to data-out valid | — | 80 | ns | 3.0V ≤ VDD ≤ 3.3V |
| | | | — | 100 | ns | 1.8V ≤ VDD ≤ 3.3V |
| US121 | TCKRF | Clock out rise time and fall time (Master mode) | — | 45 | ns | 3.0V ≤ VDD ≤ 3.3V |
| | | | — | 50 | ns | 1.8V ≤ VDD ≤ 3.3V |
| US122 | TDTRF | Data-out rise time and fall time | — | 45 | ns | 3.0V ≤ VDD ≤ 3.3V |
| | | | — | 50 | ns | 1.8V ≤ VDD ≤ 3.3V |

**FIGURE 25-12: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**



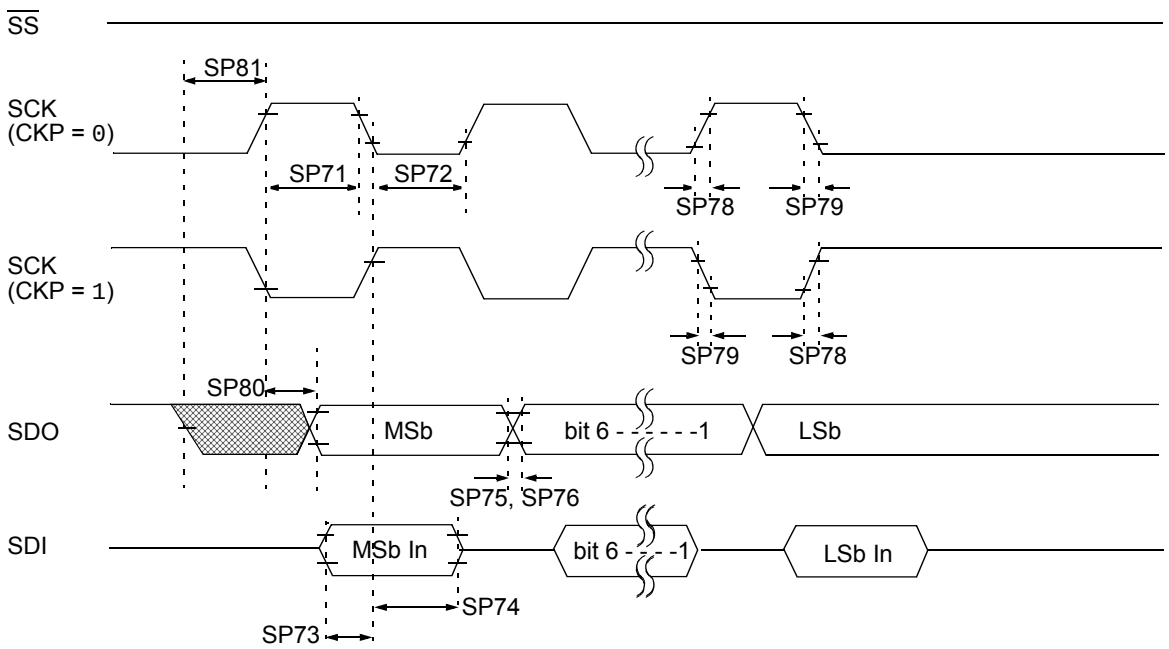

**Note:** Refer to Figure 25-3 for Load conditions.

**TABLE 25-15: USART SYNCHRONOUS RECEIVE REQUIREMENTS**

| **Standard Operating Conditions (unless otherwise stated)** | | | | | | |
|---|---|---|---|---|---|---|
| **Param. No.** | **Symbol** | **Characteristic** | **Min.** | **Max.** | **Units** | **Conditions** |
| US125 | TDTV2CKL | SYNC RCV (Master and Slave) Data-hold before CK ↓ (DT hold time) | 10 | — | ns | |
| US126 | TCKL2DTL | Data-hold after CK ↓ (DT hold time) | 15 | — | ns | |

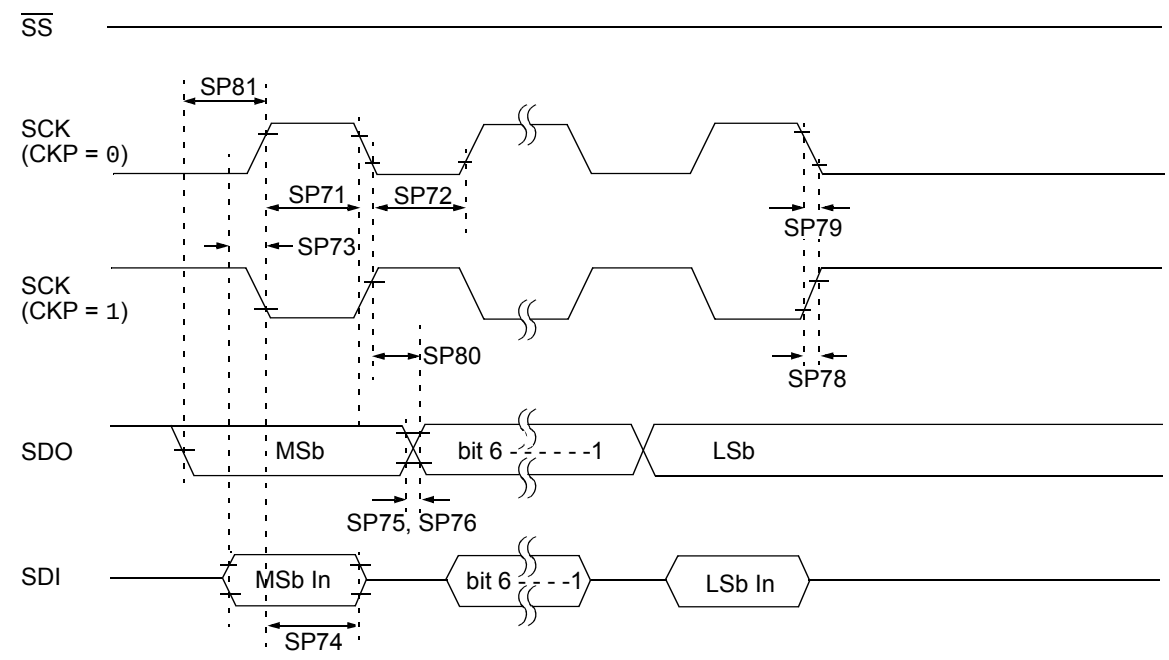
 © 2015-2016 Microchip Technology Inc.

**FIGURE 25-13: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)**



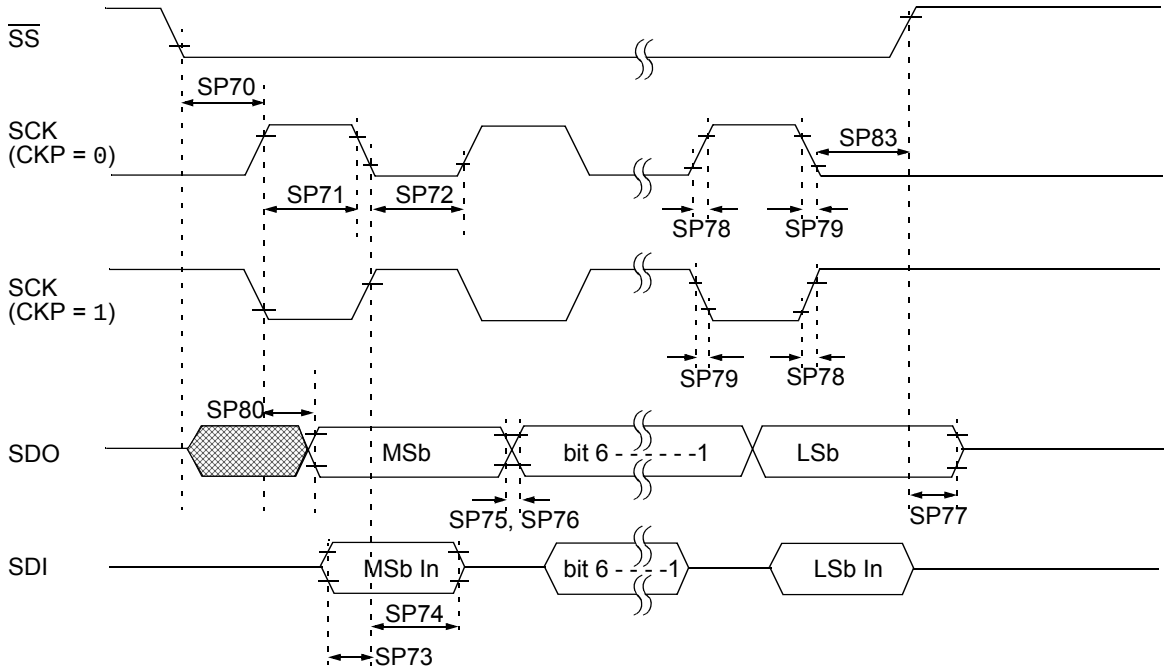

**Note:** Refer to Figure 25-3 for Load conditions.

**FIGURE 25-14: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)**



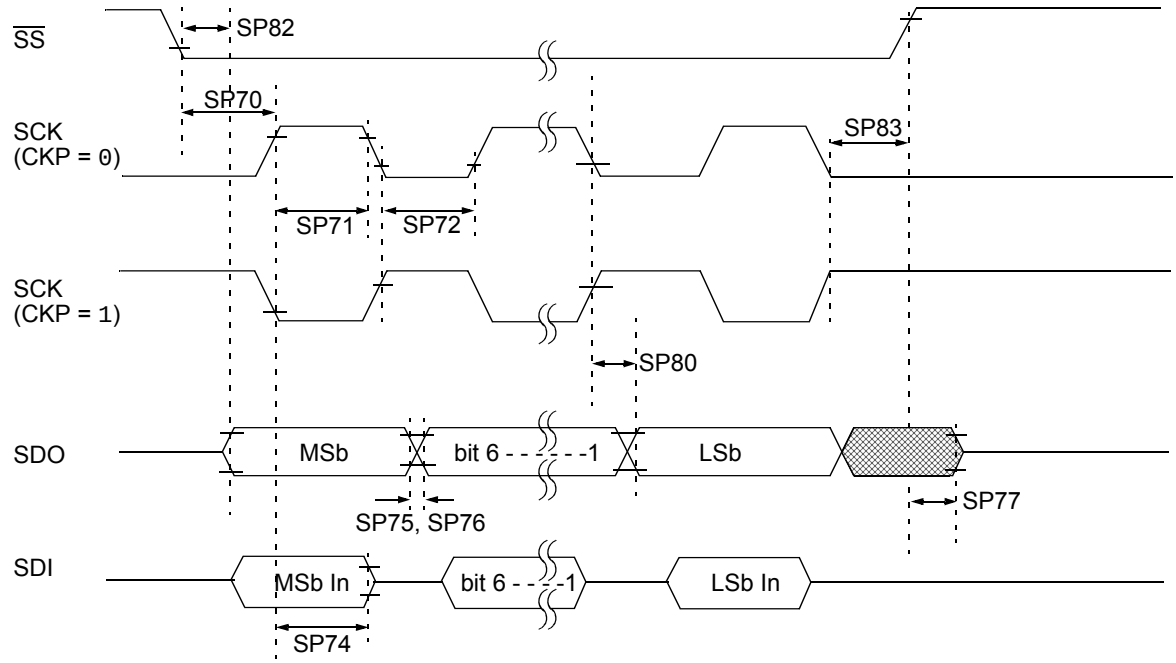

**Note:** Refer to Figure 25-3 for Load conditions.

**FIGURE 25-15: SPI SLAVE MODE TIMING (CKE = 0)**




**Note:** Refer to Figure 25-3 for Load conditions.

**FIGURE 25-16: SPI SLAVE MODE TIMING (CKE = 1)**




**Note:** Refer to Figure 25-3 for Load conditions.

 © 2015-2016 Microchip Technology Inc.

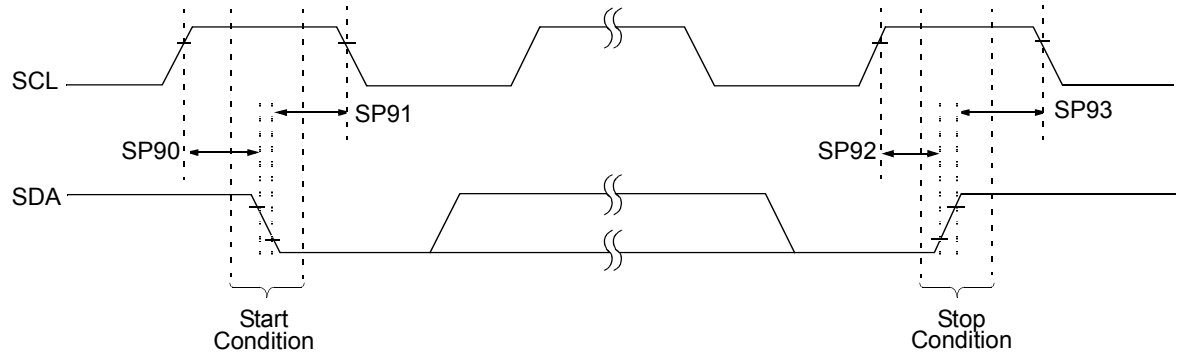
**TABLE 25-16: SPI MODE REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Symbol** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| SP70* | TssL2scH, TssL2scL | $\overline{SS}$↓ to SCK↓ or SCK↑ input | 2.25 Tcy | — | — | ns | |
| SP71* | TscH | SCK input high time (Slave mode) | 1 Tcy + 20 | — | — | ns | |
| SP72* | TscL | SCK input low time (Slave mode) | 1 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 | — | 10 | 25 | ns | 3.0V ≤ VDD ≤ 5.5V |
| | | | — | 25 | 50 | ns | 1.8V ≤ VDD ≤ 5.5V |
| SP76* | TdoF | SDO data output fall time | — | 10 | 25 | ns | |
| SP77* | TssH2doZ | $\overline{SS}$↑ to SDO output high-impedance | 10 | — | 50 | ns | |
| SP78* | TscR | SCK output rise time (Master mode) | — | 10 | 25 | ns | 3.0V ≤ VDD ≤ 5.5V |
| | | | — | 25 | 50 | ns | 1.8V ≤ VDD ≤ 5.5V |
| SP79* | TscF | SCK output fall time (Master mode) | — | 10 | 25 | ns | |
| SP80* | TscH2doV, TscL2doV | SDO data output valid after SCK edge | — | — | 50 | ns | 3.0V ≤ VDD ≤ 5.5V |
| | | | — | — | 145 | ns | 1.8V ≤ VDD ≤ 5.5V |
| SP81* | TdoV2scH, TdoV2scL | SDO data output setup to SCK edge | 1 Tcy | — | — | ns | |
| SP82* | TssL2doV | SDO data output valid after $\overline{SS}$↓ edge | — | — | 50 | ns | |
| SP83* | TscH2ssH, TscL2ssH | $\overline{SS}$ ↑ after SCK edge | 1.5 Tcy + 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 25-17: I²C BUS START/STOP BITS TIMING**



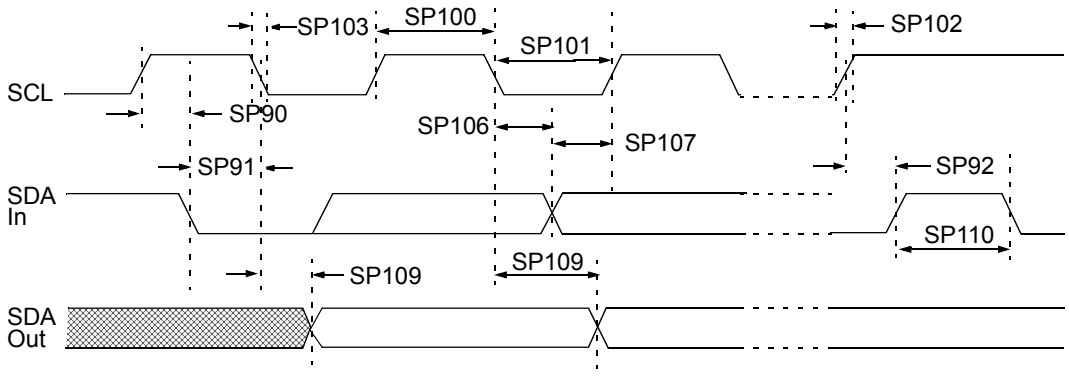

**Note:** Refer to Figure 25-3 for Load conditions.

**TABLE 25-17: I²C BUS START/STOP BITS REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **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 25-18: I²C BUS DATA TIMING**




**Note:** Refer to Figure 25-3 for Load conditions.

 © 2015-2016 Microchip Technology Inc.

**TABLE 25-18: I²C BUS DATA REQUIREMENTS**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Symbol** | **Characteristic** | | **Min.** | **Max.** | **Units** | **Conditions** |
| SP100* | THIGH | Clock high time | 100 kHz mode | 4.0 | — | µs | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 0.6 | — | µs | Device must operate at a minimum of 10 MHz |
| | | | SSP module | 1.5TCY | — | | |
| SP101* | TLOW | Clock low time | 100 kHz mode | 4.7 | — | µs | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 1.3 | — | µs | Device must operate at a minimum of 10 MHz |
| | | | SSP module | 1.5TCY | — | | |
| SP102* | TR | SDA and SCL rise time | 100 kHz mode | — | 1000 | ns | |
| | | | 400 kHz mode | 20 + 0.1CB | 300 | ns | CB is specified to be from 10-400 pF |
| SP103* | TF | SDA and SCL fall time | 100 kHz mode | — | 250 | ns | |
| | | | 400 kHz mode | 20 + 0.1CB | 250 | ns | CB is specified to be from 10-400 pF |
| SP106* | THD:DAT | Data input hold time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | µs | |
| SP107* | TSU:DAT | Data input setup time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| SP109* | TAA | Output valid from clock | 100 kHz mode | — | 3500 | ns | **(Note 1)** |
| | | | 400 kHz mode | — | — | ns | |
| SP110* | TBUF | Bus free time | 100 kHz mode | 4.7 | — | µs | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | µs | |
| SP111 | CB | Bus capacitive loading | | — | 400 | pF | |

* These parameters are characterized but not tested.

**Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.

**2:** A Fast mode (400 kHz) I²C bus device can be used in a Standard mode (100 kHz) I²C bus system, but the requirement TSU:DAT ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I²C bus specification), before the SCL line is released.

## 26.0 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.**

Charts and graphs are not available at this time.

 © 2015-2016 Microchip Technology Inc.

## 27.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICkit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits and Starter Kits
- Third-party development tools

### 27.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

## 27.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

## 27.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

## 27.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

## 27.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

 © 2015-2016 Microchip Technology Inc.

## 27.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 27.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 27.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 27.9 PICkit 3 In-Circuit Debugger/ Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

## 27.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at $V_{DDMIN}$ and $V_{DDMAX}$ for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.

## 27.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

## 27.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

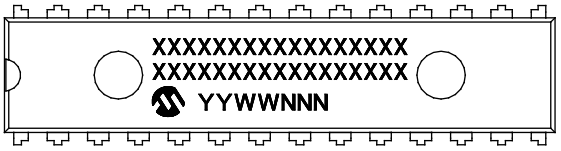
- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

 © 2015-2016 Microchip Technology Inc.

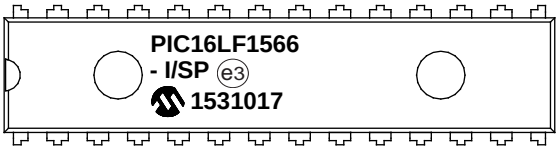
## 28.0 PACKAGING INFORMATION

### 28.1 Package Marking Information

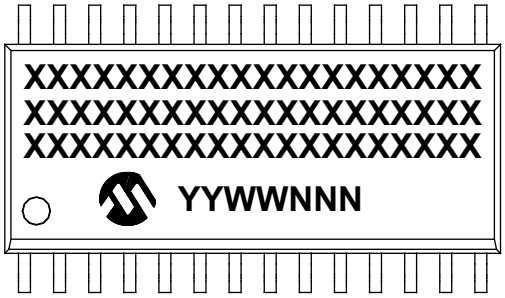
28-Lead SPDIP (.300")



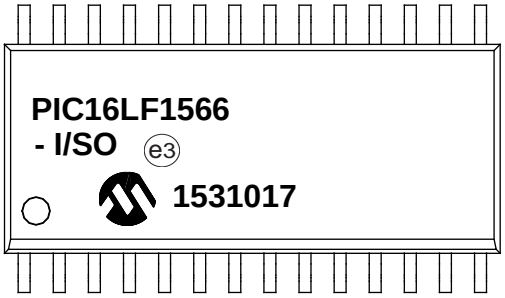

Example



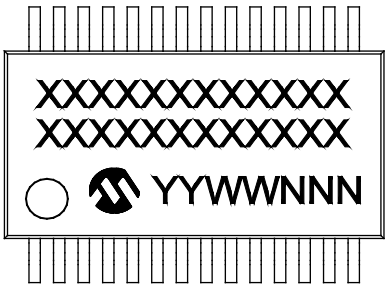

28-Lead SOIC (7.50 mm)



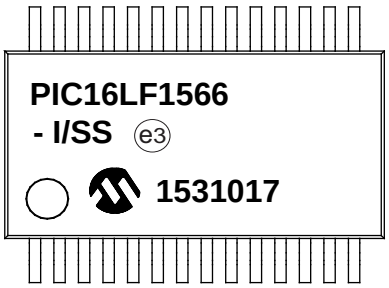

Example




28-Lead SSOP (5.30 mm)




Example

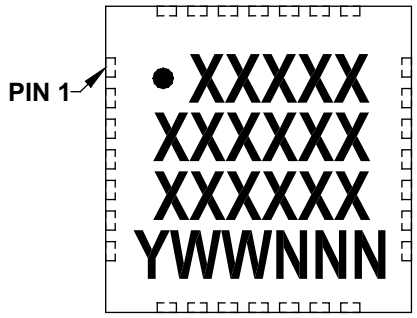



| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week '01') |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC® 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. |

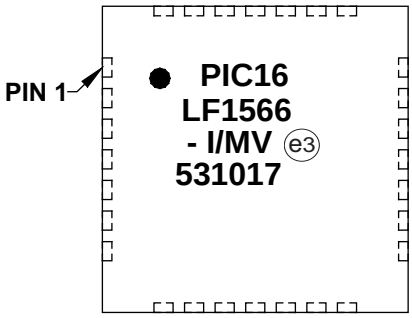
**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

**Package Marking Information (Continued)**

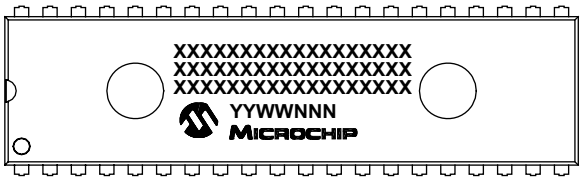
28-Lead UQFN (4x4x0.5 mm)



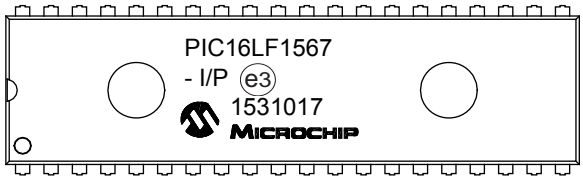

Example



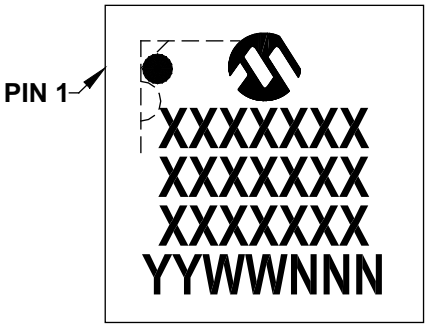

40-Lead PDIP (600 mil)



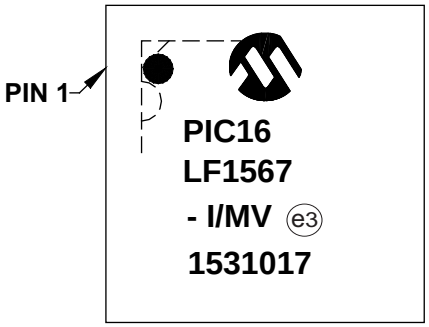

Example




40-Lead UQFN (5x5x0.5 mm)




Example



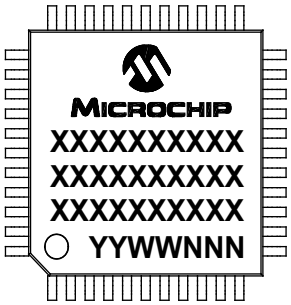

| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week ‘01’) |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC® designator for Matte Tin (Sn) |
| | * | This package is Pb-free. The Pb-free JEDEC® designator ((e3)) can be found on the outer packaging for this package. |

**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

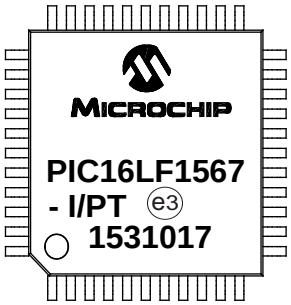
 © 2015-2016 Microchip Technology Inc.

**Package Marking Information (Continued)**

44-Lead TQFP (10x10x1 mm)




Example




| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week ‘01’) |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC® designator for Matte Tin (Sn) |
| | * | This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package. |

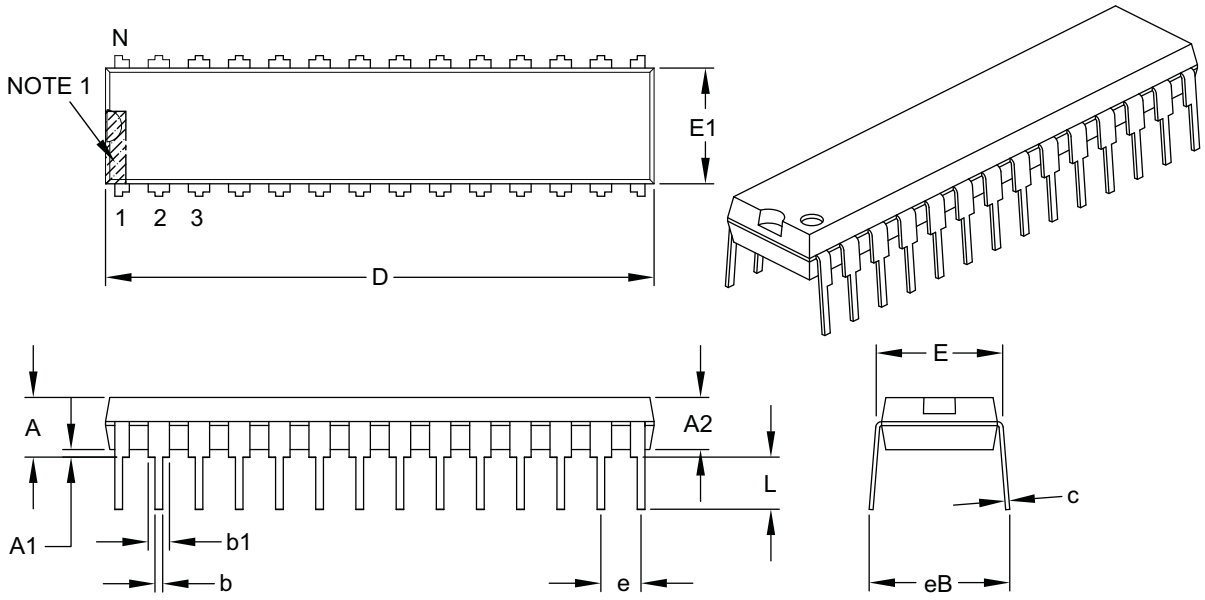
**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

## 28.2 Package Details

The following sections give the technical details of the packages.

### 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.

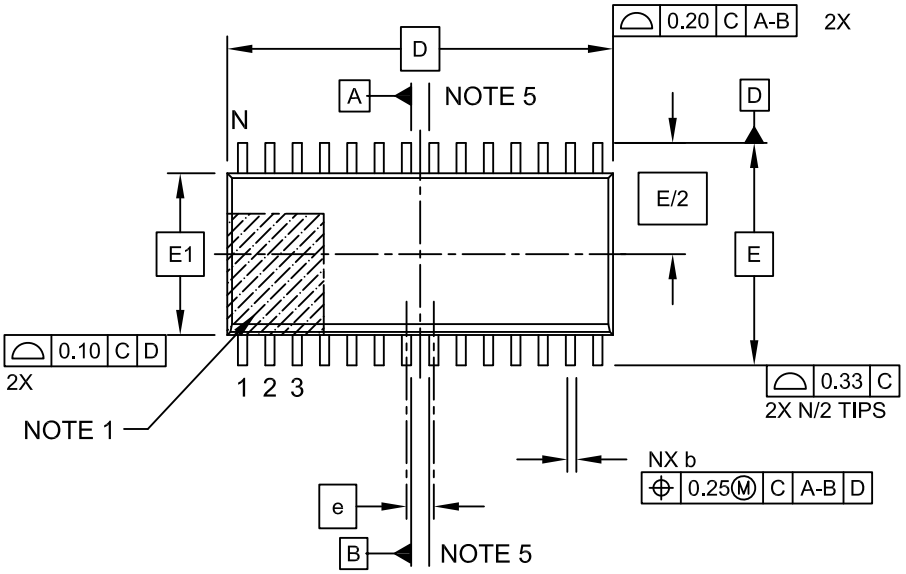
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

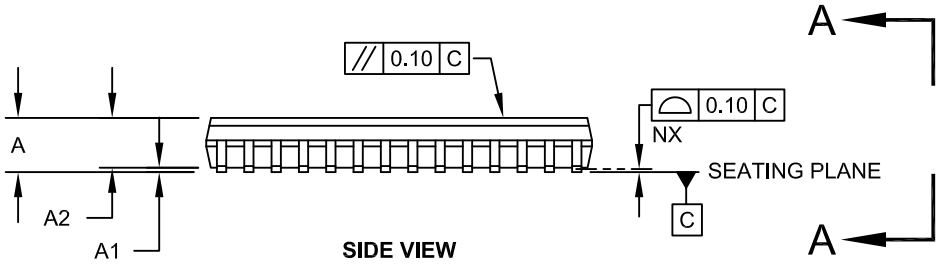

© 2015-2016 Microchip Technology Inc.

### 28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

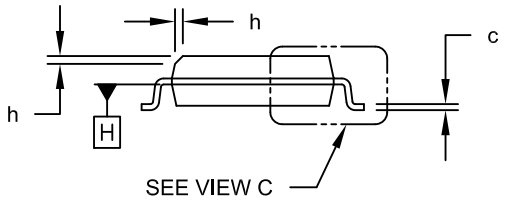
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




**TOP VIEW**




**SIDE VIEW**

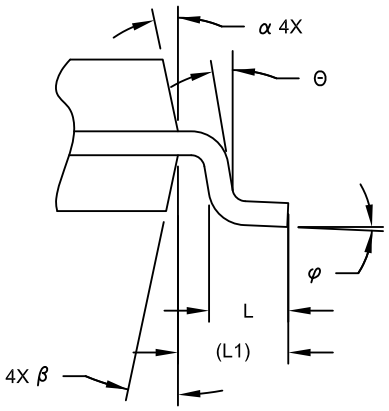



**VIEW A-A**

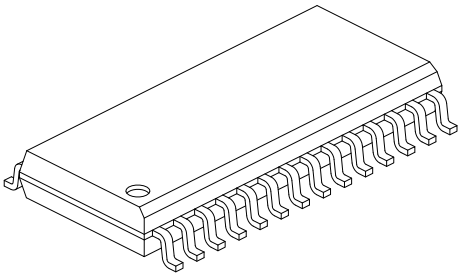
Microchip Technology Drawing C04-052C Sheet 1 of 2

### 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 | Θ | 0° | - | - |
| Foot Angle | φ | 0° | - | 8° |
| Lead Thickness | c | 0.18 | - | 0.33 |
| Lead Width | b | 0.31 | - | 0.51 |
| Mold Draft Angle Top | α | 5° | - | 15° |
| Mold Draft Angle Bottom | β | 5° | - | 15° |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

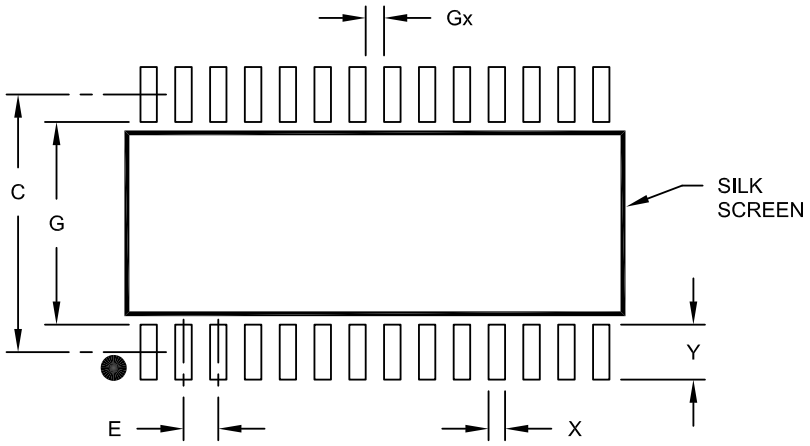
   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing C04-052C Sheet 2 of 2

 © 2015-2016 Microchip Technology Inc.

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 1.27 BSC | | |
| Contact Pad Spacing | C | | 9.40 | |
| Contact Pad Width (X28) | X | | | 0.60 |
| Contact Pad Length (X28) | Y | | | 2.00 |
| Distance Between Pads | Gx | 0.67 | | |
| Distance Between Pads | G | 7.40 | | |

Notes:

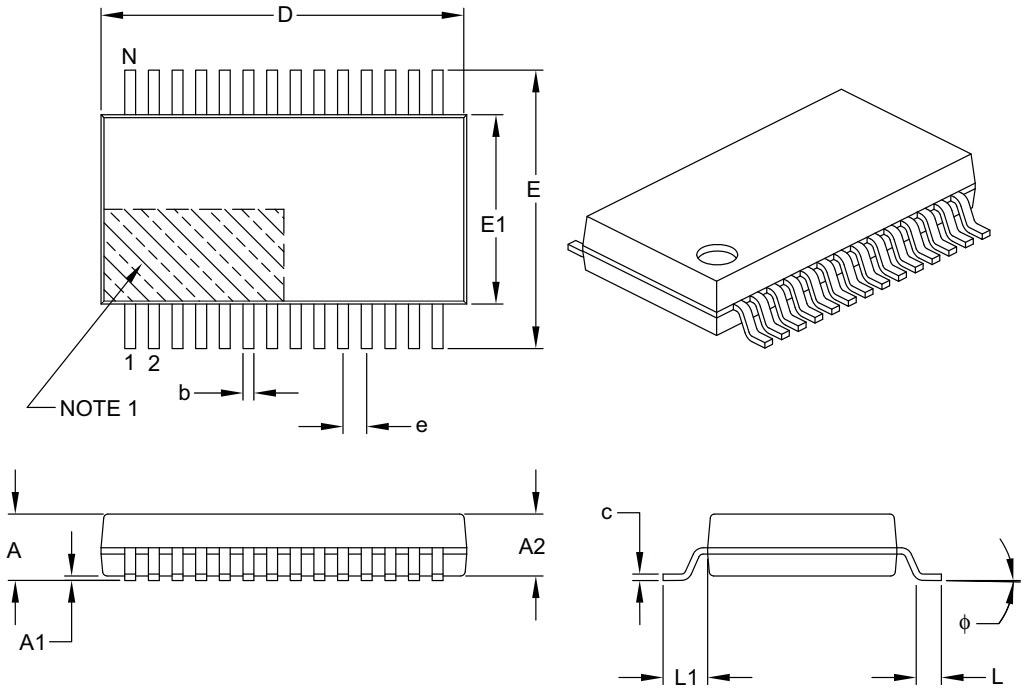
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2052A

### 28-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | – | – | 2.00 |
| Molded Package Thickness | A2 | 1.65 | 1.75 | 1.85 |
| Standoff | A1 | 0.05 | – | – |
| Overall Width | E | 7.40 | 7.80 | 8.20 |
| Molded Package Width | E1 | 5.00 | 5.30 | 5.60 |
| Overall Length | D | 9.90 | 10.20 | 10.50 |
| Foot Length | L | 0.55 | 0.75 | 0.95 |
| Footprint | L1 | | 1.25 REF | |
| Lead Thickness | c | 0.09 | – | 0.25 |
| Foot Angle | ϕ | 0° | 4° | 8° |
| Lead Width | b | 0.22 | – | 0.38 |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

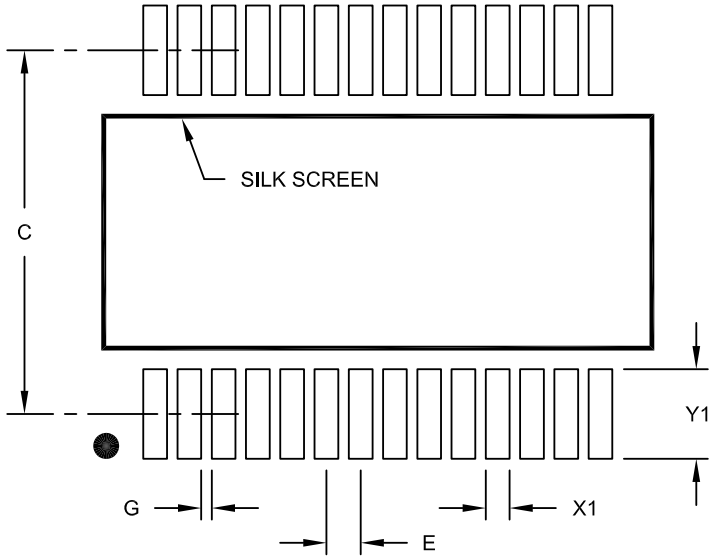
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073B

 © 2015-2016 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




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:

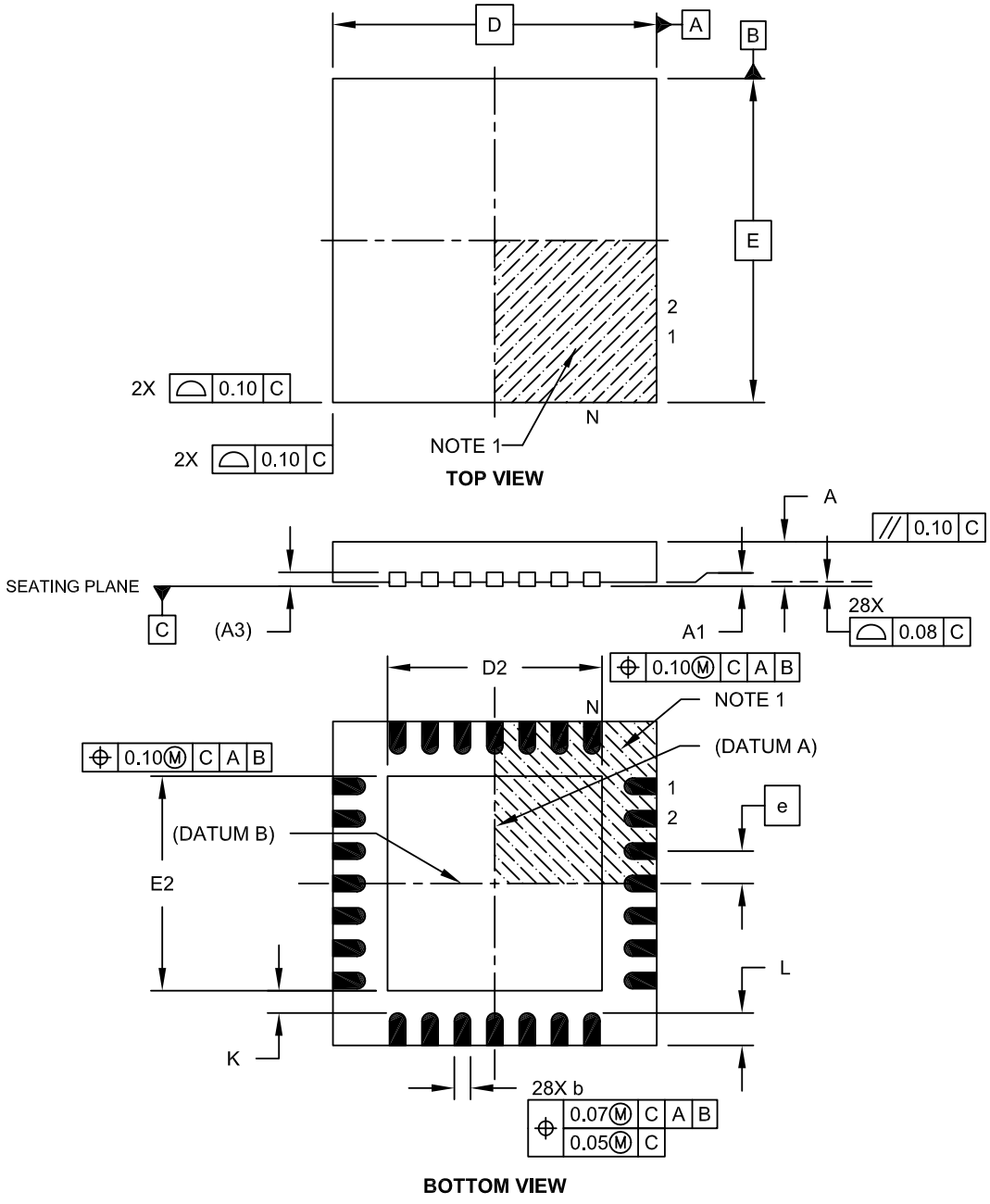
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

### 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

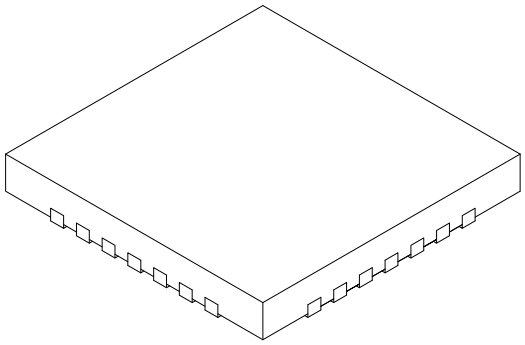



Microchip Technology Drawing C04-152A Sheet 1 of 2

 © 2015-2016 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



| 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.

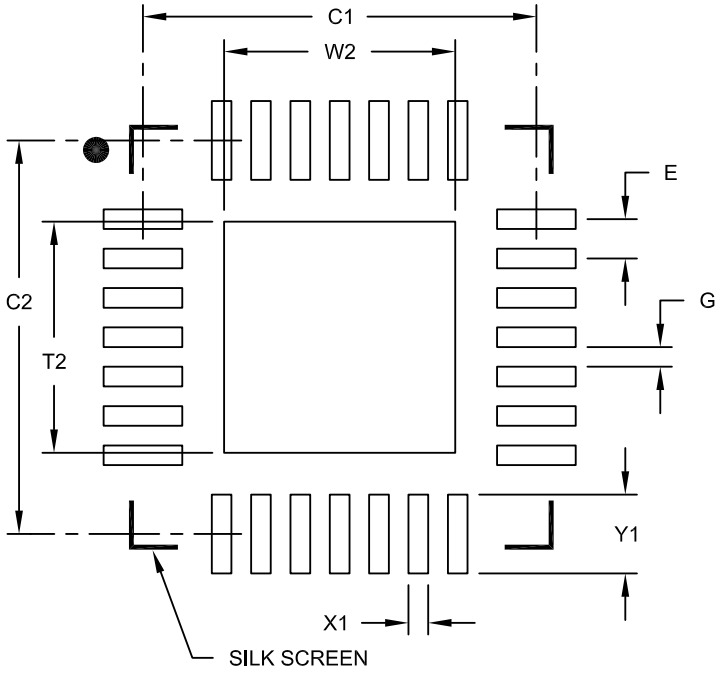
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-152A Sheet 2 of 2

28-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) - 4x4 mm Body [UQFN]
With 0.40 mm Contact Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.40 BSC | |
| Optional Center Pad Width | W2 | | | 2.35 |
| Optional Center Pad Length | T2 | | | 2.35 |
| Contact Pad Spacing | C1 | | 4.00 | |
| Contact Pad Spacing | C2 | | 4.00 | |
| Contact Pad Width (X28) | X1 | | | 0.20 |
| Contact Pad Length (X28) | Y1 | | | 0.80 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

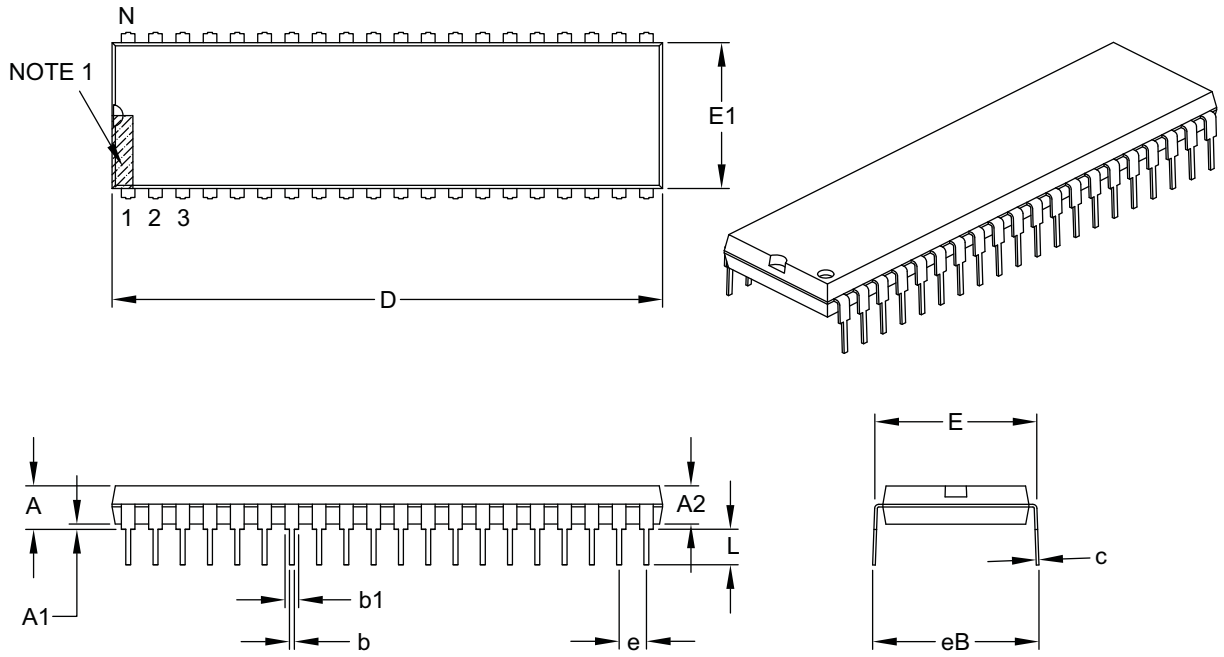
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2152A

 © 2015-2016 Microchip Technology Inc.

### 40-Lead Plastic Dual In-Line (P) – 600 mil Body [PDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | INCHES | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 40 | |
| Pitch | e | | .100 BSC | |
| Top to Seating Plane | A | – | – | .250 |
| Molded Package Thickness | A2 | .125 | – | .195 |
| Base to Seating Plane | A1 | .015 | – | – |
| Shoulder to Shoulder Width | E | .590 | – | .625 |
| Molded Package Width | E1 | .485 | – | .580 |
| Overall Length | D | 1.980 | – | 2.095 |
| Tip to Seating Plane | L | .115 | – | .200 |
| Lead Thickness | c | .008 | – | .015 |
| Upper Lead Width | b1 | .030 | – | .070 |
| Lower Lead Width | b | .014 | – | .023 |
| Overall Row Spacing § | eB | – | – | .700 |

**Notes:**

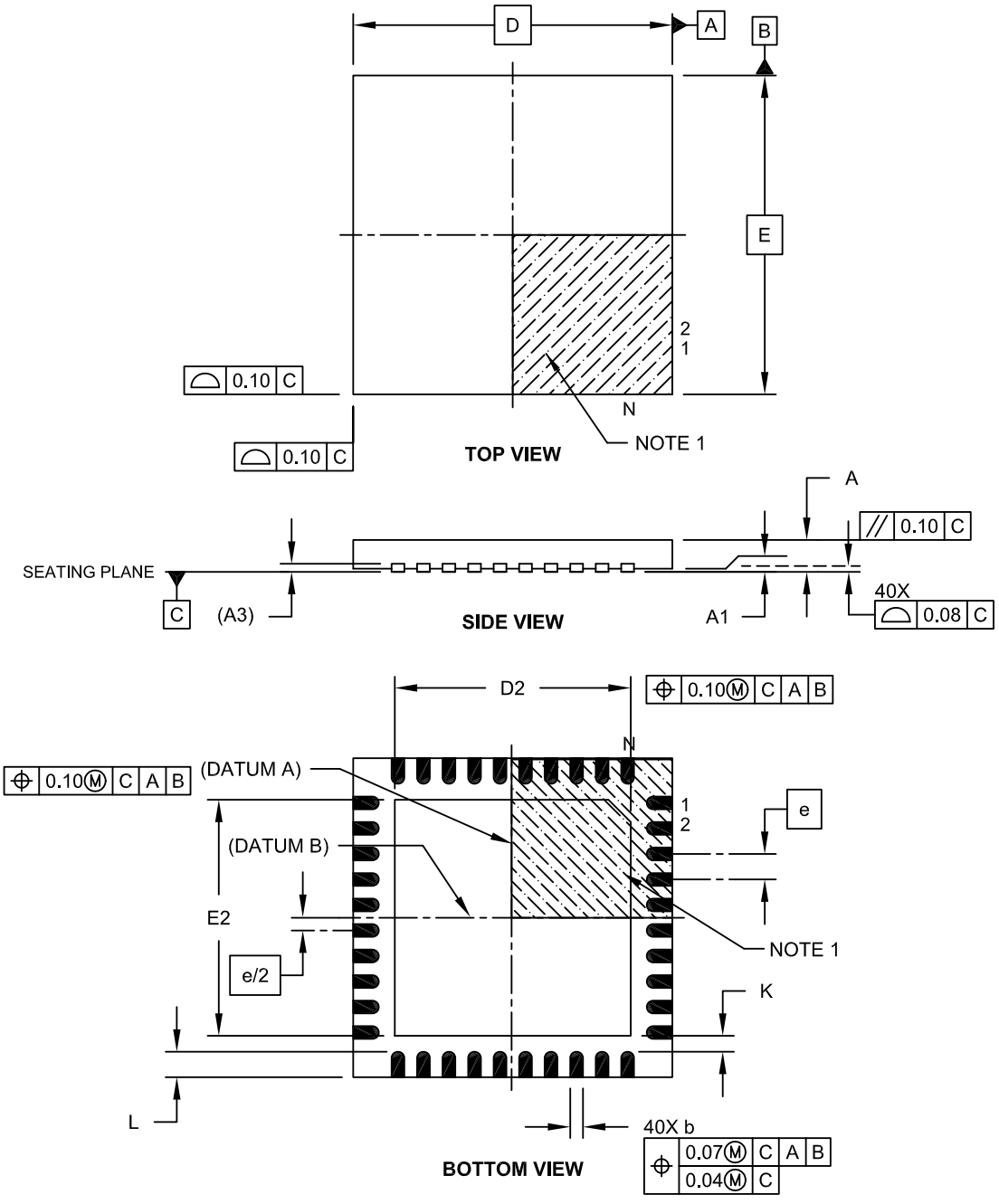
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-016B

**40-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) – 5x5x0.5 mm Body [UQFN]**

| **Note:** | For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging |
|---|---|

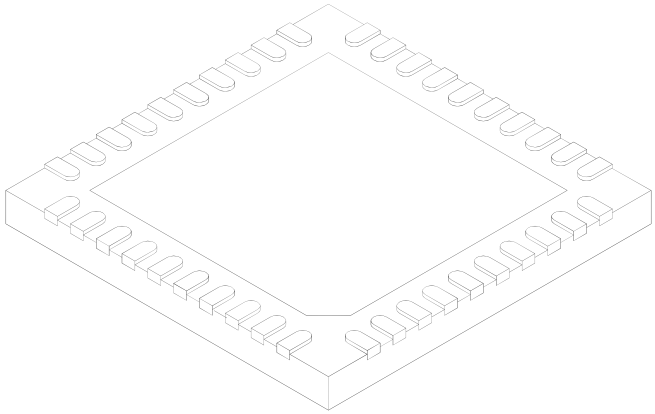



Microchip Technology Drawing C04-156A Sheet 1 of 2

 © 2015-2016 Microchip Technology Inc.

### 40-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) – 5x5x0.5 mm Body [UQFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 40 | |
| Pitch | e | | 0.40 BSC | |
| Overall Height | A | 0.45 | 0.50 | 0.55 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Contact Thickness | A3 | | 0.127 REF | |
| Overall Width | E | | 5.00 BSC | |
| Exposed Pad Width | E2 | 3.60 | 3.70 | 3.80 |
| Overall Length | D | | 5.00 BSC | |
| Exposed Pad Length | D2 | 3.60 | 3.70 | 3.80 |
| Contact Width | b | 0.15 | 0.20 | 0.25 |
| Contact Length | L | 0.30 | 0.40 | 0.50 |
| Contact-to-Exposed Pad | K | 0.20 | - | - |

**Notes:**

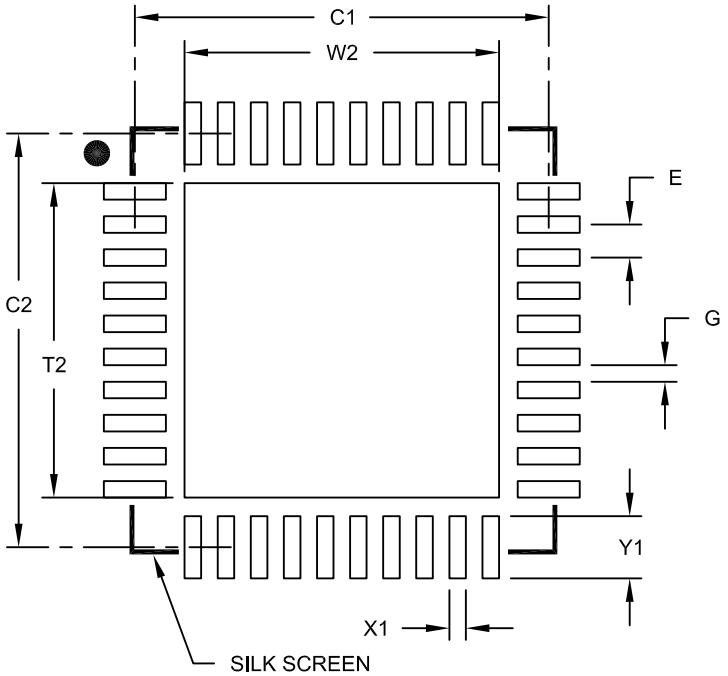
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-156A Sheet 2 of 2

### 40-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) - 5x5 mm Body [UQFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.40 BSC | |
| Optional Center Pad Width | W2 | | | 3.80 |
| Optional Center Pad Length | T2 | | | 3.80 |
| Contact Pad Spacing | C1 | | 5.00 | |
| Contact Pad Spacing | C2 | | 5.00 | |
| Contact Pad Width (X40) | X1 | | | 0.20 |
| Contact Pad Length (X40) | Y1 | | | 0.75 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

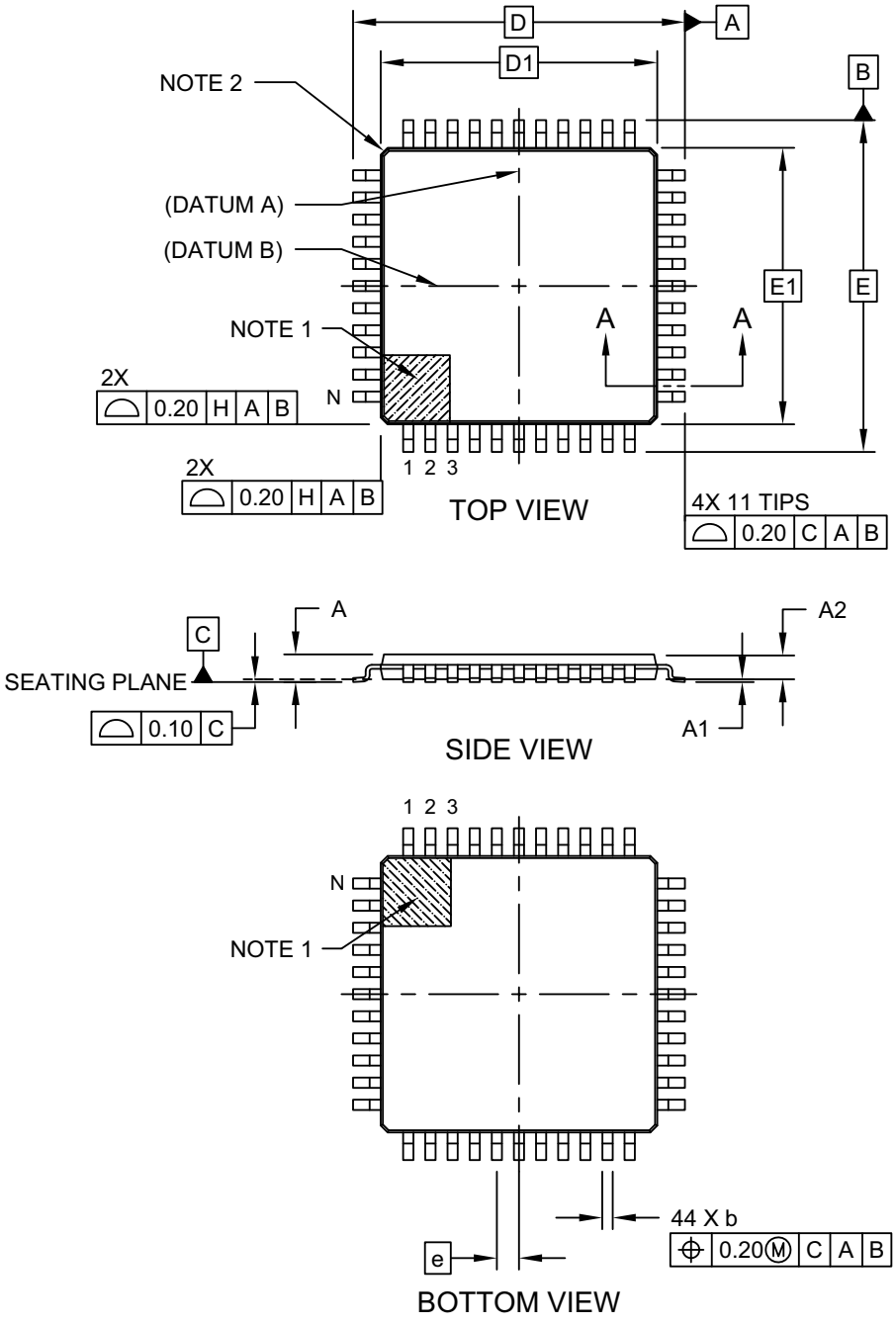
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2156B

 © 2015-2016 Microchip Technology Inc.

### 44-Lead Plastic Thin Quad Flatpack (PT) - 10x10x1.0 mm Body [TQFP]

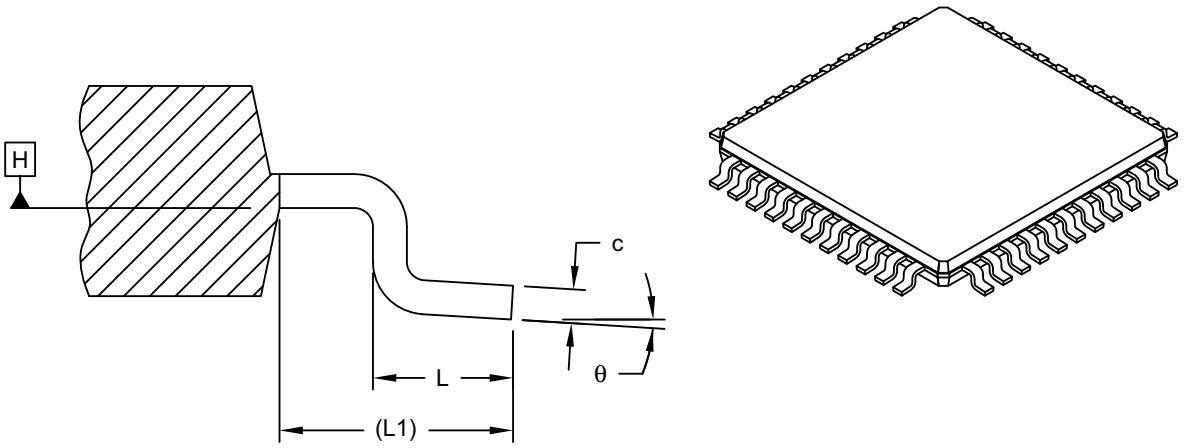
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




Microchip Technology Drawing C04-076C Sheet 1 of 2

### 44-Lead Plastic Thin Quad Flatpack (PT) - 10x10x1.0 mm Body [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




SECTION A-A

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Leads | N | | 44 | |
| Lead Pitch | e | | 0.80 BSC | |
| Overall Height | A | - | - | 1.20 |
| Standoff | A1 | 0.05 | - | 0.15 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Overall Width | E | | 12.00 BSC | |
| Molded Package Width | E1 | | 10.00 BSC | |
| Overall Length | D | | 12.00 BSC | |
| Molded Package Length | D1 | | 10.00 BSC | |
| Lead Width | b | 0.30 | 0.37 | 0.45 |
| Lead Thickness | c | 0.09 | - | 0.20 |
| Lead Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | | 1.00 REF | |
| Foot Angle | θ | 0° | 3.5° | 7° |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Exact shape of each corner is optional.
3. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

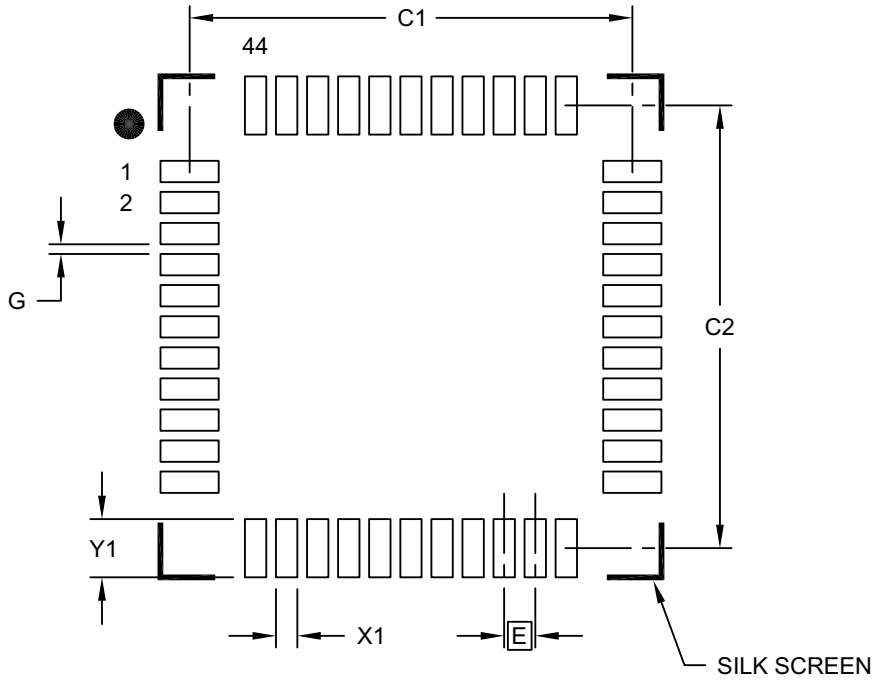
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-076C Sheet 2 of 2

 © 2015-2016 Microchip Technology Inc.

### 44-Lead Plastic Thin Quad Flatpack (PT) - 10X10X1 mm Body, 2.00 mm Footprint [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.80 BSC | |
| Contact Pad Spacing | C1 | | 11.40 | |
| Contact Pad Spacing | C2 | | 11.40 | |
| Contact Pad Width (X44) | X1 | | | 0.55 |
| Contact Pad Length (X44) | Y1 | | | 1.50 |
| Distance Between Pads | G | 0.25 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2076B

## APPENDIX A: DATA SHEET REVISION HISTORY

### Revision A (11/2015)

Initial release of the document.

### Revision B (2/2016)

Updated Example 3-2; Updated Tables 3-3 and 3-4; Updated Table 5-1; Updated Section 15.2.6; Updated Example 15-1; Updated Section 20.6; Updated Table 20-4; Updated Register 20-4; Updated Section 25.1 (Electrical Specifications); Updated Tables 25-3 and 25-8.

 © 2015-2016 Microchip Technology Inc.

## THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the website at: http://www.microchip.com/support**

## 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](1) | X | /XX | XXX |
|---|---|---|---|---|
| Device | Tape and Reel Option | Temperature Range | Package | Pattern |

| | | |
|---|---|---|
| **Device:** | PIC16LF1566; PIC16LF1567 | |
| **Tape and Reel Option:** | Blank = Standard packaging (tube or tray)<br>T = Tape and Reel(1) | |
| **Temperature Range:** | I = -40°C to +85°C (Industrial)<br>E = -40°C to +125°C (Extended) | |
| **Package:** | P = PDIP<br>SP = SPDIP<br>SO = SOIC<br>SS = SSOP<br>MV = UQFN<br>PT = TQFP | |
| **Pattern:** | QTP, SQTP, Code or Special Requirements (blank otherwise) | |

**Examples:**

a) PIC16LF1566T - I/SO, Tape and Reel, Industrial, SOIC package

b) PIC16LF1567 - E/P, Extended, PDIP 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.

 © 2015-2016 Microchip Technology Inc.

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

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

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

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

*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 ═**

**Trademarks**

The Microchip name and logo, the Microchip logo, AnyRate, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, LINK MD, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, PICSTART, PIC32 logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, ETHERSYNCH, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and QUIET-WIRE are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, RightTouch logo, REAL ICE, Ripple Blocker, Serial Quad I/O, SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademarks of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

© 2015-2016, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-0270-1

# Worldwide Sales and Service

## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

**Atlanta**
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

**Austin, TX**
Tel: 512-257-3370

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Cleveland**
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Novi, MI
Tel: 248-848-4000

**Houston, TX**
Tel: 281-894-5983

**Indianapolis**
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

**New York, NY**
Tel: 631-435-6000

**San Jose, CA**
Tel: 408-735-9110

**Canada - Toronto**
Tel: 905-673-0699
Fax: 905-673-6509

## ASIA/PACIFIC

**Asia Pacific Office**
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

**Hong Kong**
Tel: 852-2943-5100
Fax: 852-2401-3431

**Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**China - Beijing**
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

**China - Chengdu**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**China - Chongqing**
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

**China - Dongguan**
Tel: 86-769-8702-9880

**China - Hangzhou**
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

**China - Hong Kong SAR**
Tel: 852-2943-5100
Fax: 852-2401-3431

**China - Nanjing**
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

**China - Qingdao**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**China - Shanghai**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**China - Shenyang**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**China - Shenzhen**
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

**China - Wuhan**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**China - Xian**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

## ASIA/PACIFIC

**China - Xiamen**
Tel: 86-592-2388138
Fax: 86-592-2388130

**China - Zhuhai**
Tel: 86-756-3210040
Fax: 86-756-3210049

**India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

**India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**India - Pune**
Tel: 91-20-3019-1500

**Japan - Osaka**
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

**Japan - Tokyo**
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

**Korea - Daegu**
Tel: 82-53-744-4301
Fax: 82-53-744-4302

**Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

**Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**Taiwan - Hsin Chu**
Tel: 886-3-5778-366
Fax: 886-3-5770-955

**Taiwan - Kaohsiung**
Tel: 886-7-213-7828

**Taiwan - Taipei**
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

**Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

## EUROPE

**Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**Denmark - Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**Germany - Dusseldorf**
Tel: 49-2129-3766400

**Germany - Karlsruhe**
Tel: 49-721-625370

**Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Italy - Venice**
Tel: 39-049-7625286

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Poland - Warsaw**
Tel: 48-22-3325737

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**Sweden - Stockholm**
Tel: 46-8-5090-4654

**UK - Wokingham**
Tel: 44-118-921-5800
Fax: 44-118-921-5820

07/14/15


© 2015-2016 Microchip Technology Inc.

# Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

Microchip:

PIC16LF1566-E/MV PIC16LF1566-E/SP PIC16LF1566-E/SS PIC16LF1566-I/MV PIC16LF1566-I/SP PIC16LF1566-I/SS PIC16LF1566T-I/SO PIC16LF1567-I/P PIC16LF1567-I/PT PIC16LF1567T-I/MV PIC16LF1567T-I/PT PIC16LF1566-E/SO PIC16LF1566-I/SO PIC16LF1566T-I/MV PIC16LF1566T-I/SS PIC16LF1567-E/MV PIC16LF1567-E/P PIC16LF1567-E/PT PIC16LF1567-I/MV

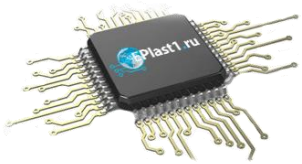
Компания «ЭлектроПласт» предлагает заключение долгосрочных отношений при поставках импортных электронных компонентов на взаимовыгодных условиях!

Наши преимущества:

- Оперативные поставки широкого спектра электронных компонентов отечественного и импортного производства напрямую от производителей и с крупнейших мировых складов;
- Поставка более 17-ти миллионов наименований электронных компонентов;
- Поставка сложных, дефицитных, либо снятых с производства позиций;
- Оперативные сроки поставки под заказ (от 5 рабочих дней);
- Экспресс доставка в любую точку России;
- Техническая поддержка проекта, помощь в подборе аналогов, поставка прототипов;
- Система менеджмента качества сертифицирована по Международному стандарту ISO 9001;
- Лицензия ФСБ на осуществление работ с использованием сведений, составляющих государственную тайну;
- Поставка специализированных компонентов (Xilinx, Altera, Analog Devices, Intersil, Interpoint, Microsemi, Aeroflex, Peregrine, Syfer, Eurofarad, Texas Instrument, Miteq, Cobham, E2V, MA-COM, Hittite, Mini-Circuits,General Dynamics и др.);

Помимо этого, одним из направлений компании «ЭлектроПласт» является направление «Источники питания». Мы предлагаем Вам помощь Конструкторского отдела:

- Подбор оптимального решения, техническое обоснование при выборе компонента;
- Подбор аналогов;
- Консультации по применению компонента;
- Поставка образцов и прототипов;
- Техническая поддержка проекта;
- Защита от снятия компонента с производства.




**Как с нами связаться**

**Телефон:** 8 (812) 309 58 32 (многоканальный)
**Факс:** 8 (812) 320-02-42
**Электронная почта:** org@eplast1.ru
**Адрес:** 198099, г. Санкт-Петербург, ул. Калинина, дом 2, корпус 4, литера А.