# Product Change Notification / SYST-24ZXKC485

## Date:

28-Sep-2020

## Product Category:

8-bit Microcontrollers

## PCN Type:

Document Change

## Notification Subject:

Data Sheet - PIC16F15213/14/23/24/43/44 Data Sheet

## Affected CPNs:

SYST-24ZXKC485_Affected_CPN_09282020.pdf
SYST-24ZXKC485_Affected_CPN_09282020.csv

## Notification Text:

SYST-24ZXKC485

Microchip has released a new Product Documents for the PIC16F15213/14/23/24/43/44 Data Sheet of devices. If you are using one of these devices please read the document located at PIC16F15213/14/23/24/43/44 Data Sheet.

**Notification Status:** Final

**Description of Change:** Updated PWM block diagram; Updated I/O Ports registers; Updated Configuration Word 1 register in cpu chapter; Updated Memory Banks information; Other minor corrections.

**Impacts to Data Sheet:** None

**Reason for Change:** To Improve Productivity

**Change Implementation Status:** Complete

**Date Document Changes Effective:** 28 Sep 2020

**NOTE:** Please be advised that this is a change to the document only the product has not been changed.

**Markings to Distinguish Revised from Unrevised Devices:** N/A

## Attachments:

PIC16F15213/14/23/24/43/44 Data Sheet

Please contact your local Microchip sales office with questions or concerns regarding this notification.

**Terms and Conditions:**

If you wish to receive Microchip PCNs via email please register for our PCN email service at our PCN home page select register then fill in the required fields. You will find instructions about registering for Microchips PCN email service in the PCN FAQ section.

If you wish to change your PCN profile, including opt out, please go to the PCN home page select login and sign into your myMicrochip account. Select a profile option from the left navigation bar and make the applicable selections.

Affected Catalog Part Numbers (CPN)

PIC16F15213-I/MF
PIC16F15213-I/P
PIC16F15213-I/SN
PIC16F15213T-I/MF
PIC16F15213T-I/SN
PIC16F15214-I/MF
PIC16F15214-I/P
PIC16F15214-I/SN
PIC16F15214T-I/MF
PIC16F15214T-I/SN
PIC16F15223-I/MG
PIC16F15223-I/P
PIC16F15223-I/SL
PIC16F15223-I/ST
PIC16F15223T-I/MG
PIC16F15223T-I/SL
PIC16F15223T-I/ST
PIC16F15224-I/MG
PIC16F15224-I/P
PIC16F15224-I/SL
PIC16F15224-I/ST
PIC16F15224T-I/MG
PIC16F15224T-I/SL
PIC16F15224T-I/ST
PIC16F15243-I/P
PIC16F15243-I/REB
PIC16F15243-I/SO
PIC16F15243-I/SS
PIC16F15243T-I/REB
PIC16F15243T-I/SO
PIC16F15243T-I/SS
PIC16F15244-I/P
PIC16F15244-I/REB
PIC16F15244-I/SO
PIC16F15244-I/SS
PIC16F15244T-I/REB
PIC16F15244T-I/SO
PIC16F15244T-I/SS

# PIC16F15213/14/23/24/43/44

## PIC16F15213/14/23/24/43/44 Full-Featured 8/14/20-Pin Microcontrollers

## Introduction

The PIC16F152 microcontroller family is available in 8/14/16/20/28/40/44-pin packages for cost-sensitive sensor and real-time control applications. The PIC16F152 family's simplified feature set includes a 10-bit Analog-to-Digital Converter (ADC), Peripheral Pin Select (PPS), digital communication peripherals, timers and waveform generators. This microcontroller family also provides memory features, such as the Memory Access Partition (MAP) to support users in data protection and bootloader applications, and a Device Information Area (DIA) that stores Fixed Voltage Reference (FVR) offset values to help improve ADC accuracy.

## PIC16F152 Family Types

**Table 1. Devices Included in This Data Sheet**

| Device | Program Flash Memory (bytes) | Data SRAM (bytes) | Memory Access Partition/ Device Information Area | I/O Pins(1)/ Peripheral Pin Select | 8-Bit Timers with HLT/ 16-Bit Timers(2) | 10-Bit PWM/ CCP | 10-Bit ADC Channels (External/Internal) | MSSP | EUSART | SMBus Compatible I/O Pads | External Interrupt Pins | Interrupt-on-Change Pins | Watchdog Timer |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PIC16F15213 | 3.5k | 256 | Y/Y | 6/Y | 1/2 | 2/2 | 5/2 | 1 | 1 | N | 1 | 6 | Y |
| PIC16F15214 | 7k | 512 | Y/Y | 6/Y | 1/2 | 2/2 | 5/2 | 1 | 1 | N | 1 | 6 | Y |
| PIC16F15223 | 3.5k | 256 | Y/Y | 12/Y | 1/2 | 2/2 | 9/2 | 1 | 1 | Y | 1 | 12 | Y |
| PIC16F15224 | 7k | 512 | Y/Y | 12/Y | 1/2 | 2/2 | 9/2 | 1 | 1 | Y | 1 | 12 | Y |
| PIC16F15243 | 3.5k | 256 | Y/Y | 18/Y | 1/2 | 2/2 | 12/2 | 1 | 1 | Y | 1 | 18 | Y |
| PIC16F15244 | 7k | 512 | Y/Y | 18/Y | 1/2 | 2/2 | 12/2 | 1 | 1 | Y | 1 | 18 | Y |

© 2020 Microchip Technology Inc.

**Table 2. Devices not included in this data sheet**

| Device | Program Flash Memory (bytes) | Data SRAM (bytes) | Memory Access Partition/ Device Information Area | I/O Pins(1)/ Peripheral Pin Select | 8-Bit Timers with HLT/ 16-Bit Timers(2) | 10-Bit PWM/ CCP | 10-Bit ADC Channels (External/Internal) | MSSP | EUSART | SMBus Compatible I/O Pads | External Interrupt Pins | Interrupt-on-Change Pins | Watchdog Timer |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PIC16F15225 | 14k | 1024 | Y/Y | 12/Y | 1/2 | 2/2 | 9/2 | 1 | 1 | Y | 1 | 12 | Y |
| PIC16F15245 | 14k | 1024 | Y/Y | 18/Y | 1/2 | 2/2 | 12/2 | 1 | 1 | Y | 1 | 18 | Y |
| PIC16F15254 | 7k | 512 | Y/Y | 25/Y | 1/2 | 2/2 | 17/2 | 1 | 1 | Y | 1 | 25 | Y |
| PIC16F15255 | 14k | 1024 | Y/Y | 25/Y | 1/2 | 2/2 | 17/2 | 1 | 1 | Y | 1 | 25 | Y |
| PIC16F15256 | 28k | 2048 | Y/Y | 25/Y | 1/2 | 2/2 | 17/2 | 1 | 1 | Y | 1 | 25 | Y |
| PIC16F15274 | 7k | 512 | Y/Y | 36/Y | 1/2 | 2/2 | 28/2 | 1 | 1 | Y | 1 | 36 | Y |
| PIC16F15275 | 14k | 1024 | Y/Y | 36/Y | 1/2 | 2/2 | 28/2 | 1 | 1 | Y | 1 | 36 | Y |
| PIC16F15276 | 28k | 2048 | Y/Y | 36/Y | 1/2 | 2/2 | 28/2 | 1 | 1 | Y | 1 | 36 | Y |

**Notes:**

1. Total I/O count includes one pin ($\overline{MCLR}$) that is input-only.
2. Timer0 can be configured as either an 8 or 16-bit timer.

## Core Features

- C Compiler Optimized RISC Architecture
- Operating Speed:
  - DC – 32 MHz clock input
  - 125 ns minimum instruction time
- 16-Level Deep Hardware Stack
- Low-Current Power-on Reset (POR)
- Configurable Power-up Timer (PWRT)
- Brown-out Reset (BOR)
- Watchdog Timer (WDT)

## Memory

- Up to 28 KB of Program Flash Memory
- Up to 2 KB of Data SRAM Memory
- Memory Access Partition (MAP): The Program Flash Memory can be partitioned into:
  - Application Block
  - Boot Block
  - Storage Area Flash (SAF) Block
- Programmable Code Protection and Write Protection
- Device Information Area (DIA) Stores:
  - Fixed Voltage Reference (FVR) measurement data
  - Microchip unique identifier
- Device Characteristics Area (DCI) Stores:

  - Program/erase row sizes
  - Pin count details
- Direct, Indirect, and Relative Addressing Modes

## Operating Characteristics

- Operating Voltage Range:
  - 1.8V to 5.5V
- Temperature Range:
  - Industrial: -40°C to 85°C
  - Extended: -40°C to 125°C

## Power-Saving Functionality

- Sleep:
  - Reduce device power consumption
  - Reduce system electrical noise while performing ADC conversions
- Low-Power Mode Features:
  - Sleep:
    - < 900 nA typical @ 3V/25°C (WDT enabled)
    - < 600 nA typical @ 3V/25°C (WDT disabled)
  - Operating Current:
    - 48 µA typical @ 32 kHz, 3V/25°C
    - < 1 mA typical @ 4 MHz, 5V/25°C

## Digital Peripherals

- Two Capture/Compare/PWM (CCP) modules:
  - 16-bit resolution for Capture/Compare modes
  - 10-bit resolution for PWM mode
- Two Pulse-Width Modulators (PWM):
  - 10-bit resolution
  - Independent pulse outputs
- One Configurable 8/16-Bit Timer (TMR0)
- One 16-Bit Timer (TMR1) with Gate Control
- One 8-Bit Timer (TMR2) with Hardware Limit Timer (HLT)
- One Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART):
  - RS-232, RS-485, LIN compatible
  - Auto-wake-up on Start
- One Master Synchronous Serial Port (MSSP):
  - Serial Peripheral Interface (SPI) mode
    - Slave Select Synchronization
  - Inter-Integrated Circuit ($I^2C$) mode
    - 7/10-bit addressing modes
- Peripheral Pin Select (PPS):
  - Enables pin mapping of digital I/O
- Device I/O Port Features:
  - Up to 35 I/O pins
  - 1 input-only pin

  - Individual I/O direction, open-drain, input threshold, slew rate and weak pull-up control
  - Interrupt-on-Change (IOC) on all pins
  - One External Interrupt pin

## Analog Peripherals

- Analog-to-Digital Converter (ADC):
  - 10-bit resolution
  - Up to 28 external input channels
  - Two internal input channels
  - Internal ADC oscillator (ADCRC)
  - Operates in Sleep
  - Selectable Auto-Conversion Trigger sources
- Fixed Voltage Reference (FVR):
  - Selectable 1.024V, 2.048V, and 4.096V output levels
  - Internally connected to ADC

## Clocking Structure

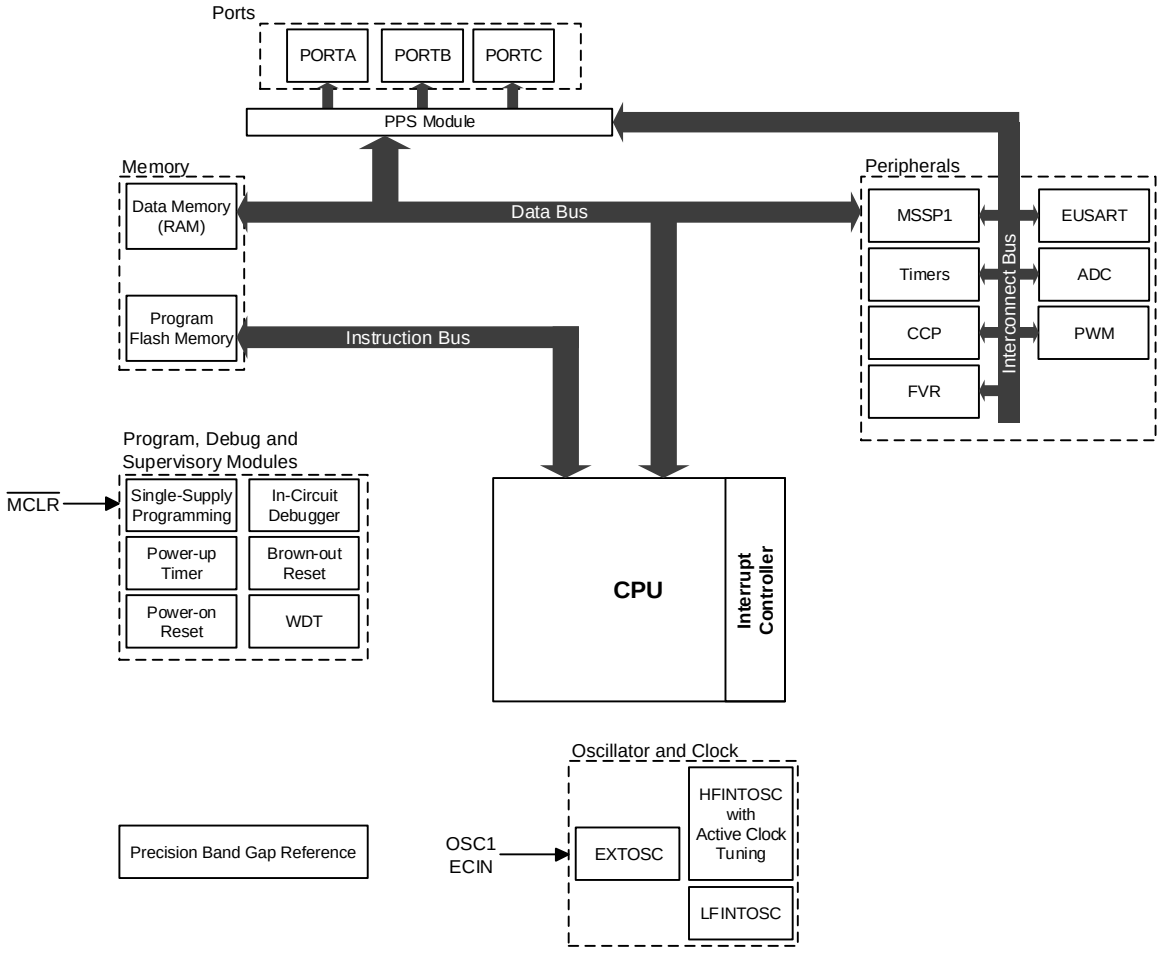
- High-Precision Internal Oscillator Block (HFINTOSC):
  - Selectable frequencies up to 32 MHz
  - ±2% at calibration
- Internal 31 kHz Oscillator (LFINTOSC)
- External High-Frequency Clock Input:
  - Two External Clock (EC) power modes

## Programming/Debug Features

- In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) with One Breakpoint via Two Pins
- Debug Integrated On-Chip

## Block Diagram

**Figure 1. PIC16F15213/14/23/24/43/44 Block Diagram**



# Table of Contents

Introduction....1

PIC16F152 Family Types....1

Core Features....2

1. Packages....8
2. Pin Diagrams....9
3. Pin Allocation Tables....11
4. Guidelines for Getting Started with PIC16F152 Microcontrollers....14
5. Register and Bit Naming Conventions....17
6. Register Legend....19
7. Enhanced Mid-Range CPU....20
8. Device Configuration....22
9. Memory Organization....32
10. Resets....61
11. OSC - Oscillator Module....73
12. Interrupts....83
13. Sleep Mode....96
14. WDT - Watchdog Timer....98
15. NVM - Nonvolatile Memory Control....102
16. I/O Ports....123
17. IOC - Interrupt-on-Change....138
18. PPS - Peripheral Pin Select Module....144
19. TMR0 - Timer0 Module....152
20. TMR1 - Timer1 Module with Gate Control....160
21. TMR2 - Timer2 Module....175
22. CCP - Capture/Compare/PWM Module....198
23. PWM - Pulse-Width Modulation....211
24. EUSART - Enhanced Universal Synchronous Asynchronous Receiver Transmitter....219
25. MSSP - Master Synchronous Serial Port Module....249
26. FVR - Fixed Voltage Reference....309

27. ADC - Analog-to-Digital Converter ..... 312
28. Charge Pump ..... 327
29. Instruction Set Summary ..... 330
30. ICSP™ - In-Circuit Serial Programming™ ..... 350
31. Register Summary ..... 353
32. Electrical Specifications ..... 358
33. DC and AC Characteristics Graphs and Tables ..... 382
34. Packaging Information ..... 383
35. Appendix A: Revision History ..... 411
The Microchip Website ..... 412
Product Change Notification Service ..... 412
Customer Support ..... 412
Product Identification System ..... 413
Microchip Devices Code Protection Feature ..... 413
Legal Notice ..... 414
Trademarks ..... 414
Quality Management System ..... 415
Worldwide Sales and Service ..... 416

# 1. Packages

**Table 1-1. Packages**

| Device | 8-Pin SOIC | 8-Pin DFN | 14-Pin TSSOP | 14-Pin SOIC | 16-Pin VQFN 3x3x0.9 | 20-Pin PDIP | 20-Pin SSOP | 20-Pin SOIC | 20-Pin VQFN 3x3x0.9 | 28-Pin SOIC | 28-Pin SSOP | 28-Pin VQFN 6x6x1 | 40-Pin PDIP | 40-Pin VQFN 5x5x0.9 | 44-Pin TQFP 10x10x1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PIC16F15213 | • | • | | | | | | | | | | | | | |
| PIC16F15214 | • | • | | | | | | | | | | | | | |
| PIC16F15223 | | | • | • | • | | | | | | | | | | |
| PIC16F15224 | | | • | • | • | | | | | | | | | | |
| PIC16F15225 | | | • | • | • | | | | | | | | | | |
| PIC16F15243 | | | | | | • | • | • | • | | | | | | |
| PIC16F15244 | | | | | | • | • | • | • | | | | | | |
| PIC16F15245 | | | | | | • | • | • | • | | | | | | |
| PIC16F15254 | | | | | | | | | | • | • | • | | | |
| PIC16F15255 | | | | | | | | | | • | • | • | | | |
| PIC16F15256 | | | | | | | | | | • | • | • | | | |
| PIC16F15274 | | | | | | | | | | | | | • | • | • |
| PIC16F15275 | | | | | | | | | | | | | • | • | • |
| PIC16F15276 | | | | | | | | | | | | | • | • | • |

© 2020 Microchip Technology Inc.

# 2. Pin Diagrams

**Figure 2-1.**
**8-Pin SOIC**
**8-Pin DFN**

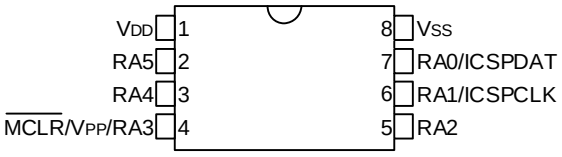



**Figure 2-2.**
**14-Pin SOIC**
**14-Pin TSSOP**

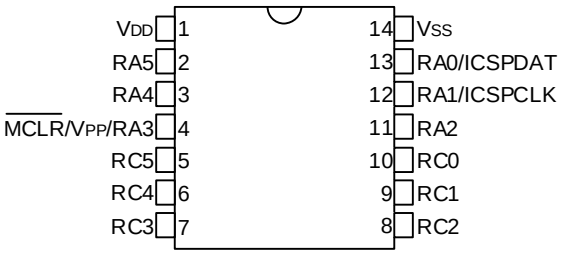
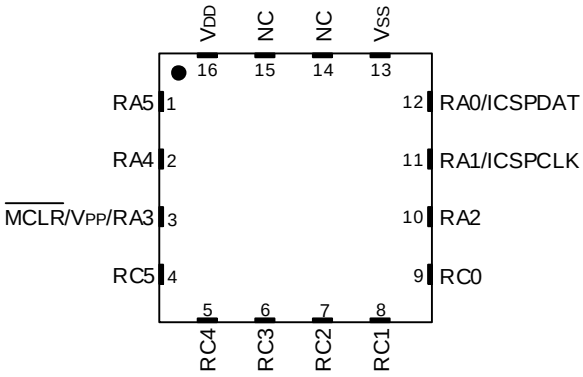



**Figure 2-3.**
**16-Pin VQFN**




**Note:** It is recommended that the exposed bottom pad be connected to $V_{SS}$, however, it must not be the only $V_{SS}$ connection to the device.

**Figure 2-4.**
**20-Pin PDIP**
**20-Pin SOIC**
**20-Pin SSOP**

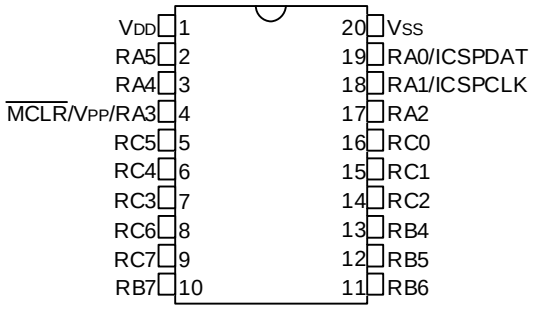
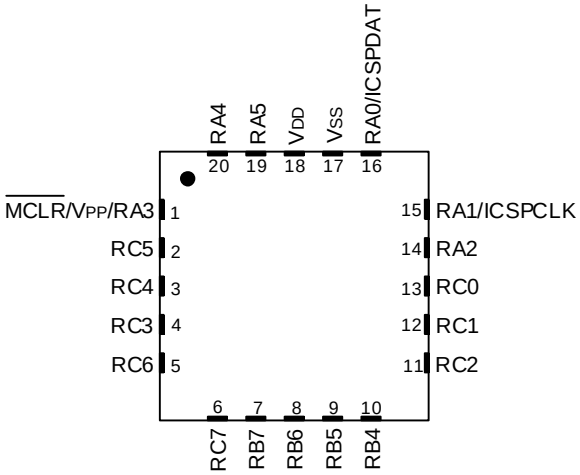



**Figure 2-5.**
**20-Pin VQFN**




**Note:** It is recommended that the exposed bottom pad be connected to $V_{SS}$; however, it must not be the only $V_{SS}$ connection to the device.

# 3. Pin Allocation Tables

### Table 3-1. 8-Pin Allocation Table

| I/O | 8-Pin SOIC DFN | ADC | Reference | Timers | CCP | 10-Bit PWM | MSSP | EUSART | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 7 | ANA0 | — | — | — | — | — | RX1(1) DT1(1,3) | IOCA0 | — | ICSPDAT ICDDAT |
| RA1 | 6 | ANA1 | VREF+ (ADC) | — | — | — | SCL1(1,3) SCK1(1,3) | CK1(1,3) | IOCA1 | — | ICSPCLK ICDCLK |
| RA2 | 5 | ANA2 | — | T0CKI(1) | — | — | SDA1(1,3) SDI1(1,3) | — | IOCA2 | INT(1) | — |
| RA3 | 4 | — | — | — | — | — | $\overline{SS1}$(1) | — | IOCA3 | — | $\overline{MCLR}$ VPP |
| RA4 | 3 | ANA4 | — | T1G(1) | — | — | — | — | IOCA4 | — | CLKOUT |
| RA5 | 2 | ANA5 ADACT(1) | — | T1CKI(1) T2IN(1) | CCP1(1) CCP2(1) | — | — | — | IOCA5 | — | CLKIN |
| VDD | 1 | — | — | — | — | — | — | — | — | — | VDD |
| VSS | 8 | — | — | — | — | — | — | — | — | — | VSS |
| OUT(2) | — | — | — | TMR0 | CCP1 CCP2 | PWM3 PWM4 | SCL1 SCK1 SDA1 SDO1 | TX1 DT1 CK1 | — | — | — |

**Notes:**

1. This is a PPS remappable input signal. The input function may be moved from the default location shown to any PORTx pin.
2. All output signals shown in this row are PPS remappable.
3. This is a bidirectional signal. For normal operation, user software must map this signal to the same pin via the PPS input and PPS output registers.

### Table 3-2. 14/16-Pin Allocation Table

| I/O | 14-Pin SOIC TSSOP | 16-Pin VQFN | ADC | Reference | Timers | CCP | 10-Bit PWM | MSSP | EUSART | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 13 | 12 | ANA0 | — | — | — | — | — | — | IOCA0 | — | ICSPDAT ICDDAT |
| RA1 | 12 | 11 | ANA1 | VREF+ (ADC) | — | — | — | — | — | IOCA1 | — | ICSPCLK ICDCLK |
| RA2 | 11 | 10 | ANA2 | — | T0CKI(1) | — | — | — | — | IOCA2 | INT(1) | — |
| RA3 | 4 | 3 | — | — | — | — | — | — | — | IOCA3 | — | $\overline{MCLR}$ VPP |
| RA4 | 3 | 2 | ANA4 | — | T1G(1) | — | — | — | — | IOCA4 | — | CLKOUT |
| RA5 | 2 | 1 | ANA5 | — | T1CKI(1) T2IN(1) | — | — | — | — | IOCA5 | — | CLKIN |
| RC0 | 10 | 9 | — | — | — | — | — | SCL1(1,3,4) SCK1(1,3,4) | — | IOCC0 | — | — |
| RC1 | 9 | 8 | — | — | — | — | — | SDA1(1,3,4) SDI1(1,3,4) | — | IOCC1 | — | — |
| RC2 | 8 | 7 | ANC2 ADACT(1) | — | — | — | — | — | — | IOCC2 | — | — |
| RC3 | 7 | 6 | ANC3 | — | — | CCP2(1) | — | $\overline{SS1}$(1) | — | IOCC3 | — | — |
| RC4 | 6 | 5 | ANC4 | — | — | — | — | — | CK1(1,3) | IOCC4 | — | — |

.....continued

| I/O | 14-Pin SOIC TSSOP | 16-Pin VQFN | ADC | Reference | Timers | CCP | 10-Bit PWM | MSSP | EUSART | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RC5 | 5 | 4 | ANC5 | — | — | CCP1(1) | — | — | RX1(1) DT1(1, 3) | IOCC5 | — | — |
| VDD | 1 | 16 | — | — | — | — | — | — | — | — | — | VDD |
| VSS | 14 | 13 | — | — | — | — | — | — | — | — | — | VSS |
| OUT(2) | — | — | — | — | TMR0 | CCP1 CCP2 | PWM3 PWM4 | SCL1 SCK1 SDA1 SDO1 | TX1 DT1 CK1 | — | — | — |

**Notes:**

1. This is a PPS remappable input signal. The input function may be moved from the default location shown to any PORTx pin.
2. All output signals shown in this row are PPS remappable.
3. This is a bidirectional signal. For normal operation, user software must map this signal to the same pin via the PPS input and PPS output registers.
4. These pins can be configured for $I^2C$ or SMBus logic levels via the RxyI2C registers. The SCL1/SDA1 signals may be assigned to these pins for expected operation. PPS assignments of these signals to other pins will operate; however, the logic levels will be standard TTL/ST as selected by the INLVL register.

### Table 3-3. 20-Pin Allocation Table

| I/O | 20-Pin PDIP SSOP SOIC | 20-Pin VQFN | ADC | Reference | Timers | CCP | 10-Bit PWM | MSSP | EUSART | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 19 | 16 | ANA0 | — | — | — | — | — | — | IOCA0 | — | ICSPDAT ICDDAT |
| RA1 | 18 | 15 | ANA1 | VREF+ (ADC) | — | — | — | — | — | IOCA1 | — | ICSPCLK ICDCLK |
| RA2 | 17 | 14 | ANA2 | — | T0CKI(1) | — | — | — | — | IOCA2 | INT(1) | — |
| RA3 | 4 | 1 | — | — | — | — | — | — | — | IOCA3 | — | $\overline{MCLR}$ VPP |
| RA4 | 3 | 20 | ANA4 | — | T1G(1) | — | — | — | — | IOCA4 | — | CLKOUT |
| RA5 | 2 | 19 | ANA5 | — | T1CKI(1) T2IN(1) | — | — | — | — | IOCA5 | — | CLKIN |
| RB4 | 13 | 10 | — | — | — | — | — | SCL1(1,3,4) SCK1(1,3,4) | — | IOCB4 | — | — |
| RB5 | 12 | 9 | ANB5 | — | — | — | — | — | RX1(1) DT1(1,3) | IOCB5 | — | — |
| RB6 | 11 | 8 | ANB6 | — | — | — | — | SDA1(1,3,4) SDI1(1,3,4) | — | IOCB6 | — | — |
| RB7 | 10 | 7 | ANB7 | — | — | — | — | — | CK1(1,3) | IOCB7 | — | — |
| RC0 | 16 | 13 | — | — | — | — | — | — | — | IOCC0 | — | — |
| RC1 | 15 | 12 | — | — | — | — | — | — | — | IOCC1 | — | — |
| RC2 | 14 | 11 | ANC2 ADACT(1) | — | — | — | — | — | — | IOCC2 | — | — |
| RC3 | 7 | 4 | ANC3 | — | — | CCP2(1) | — | — | — | IOCC3 | — | — |
| RC4 | 6 | 3 | ANC4 | — | — | — | — | — | — | IOCC4 | — | — |
| RC5 | 5 | 2 | ANC5 | — | — | CCP1(1) | — | — | — | IOCC5 | — | — |
| RC6 | 8 | 5 | — | — | — | — | — | $\overline{SS}$T(1) | — | IOCC6 | — | — |
| RC7 | 9 | 6 | — | — | — | — | — | — | — | IOCC7 | — | — |

.....continued

| I/O | 20-Pin PDIP SSOP SOIC | 20-Pin VQFN | ADC | Reference | Timers | CCP | 10-Bit PWM | MSSP | EUSART | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $V_{DD}$ | 1 | 18 | — | — | — | — | — | — | — | — | — | $V_{DD}$ |
| $V_{SS}$ | 20 | 17 | — | — | — | — | — | — | — | — | — | $V_{SS}$ |
| OUT(2) | — | — | — | — | TMR0 | CCP1<br>CCP2 | PWM3<br>PWM4 | SCL1<br>SCK1<br>SDA1<br>SDO1 | TX1<br>DT1<br>CK1 | — | — | — |

**Notes:**

1. This is a PPS remappable input signal. The input function may be moved from the default location shown to any PORTx pin.
2. All output signals shown in this row are PPS remappable.
3. This is a bidirectional signal. For normal operation, user software must map this signal to the same pin via the PPS input and PPS output registers.
4. These pins can be configured for $I^2C$ or SMBus logic levels via the RxyI2C registers. The SCL1/SDA1 signals may be assigned to these pins for expected operation. PPS assignments of these signals to other pins will operate; however, the logic levels will be standard TTL/ST as selected by the INLVL register.

# 4. Guidelines for Getting Started with PIC16F152 Microcontrollers

## 4.1 Basic Connection Requirements

Getting started with the PIC16F152 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All $V_{DD}$ and $V_{SS}$ pins (see "Power Supply Pins")
- $\overline{MCLR}$ pin (see "Master Clear ($\overline{MCLR}$) Pin")

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

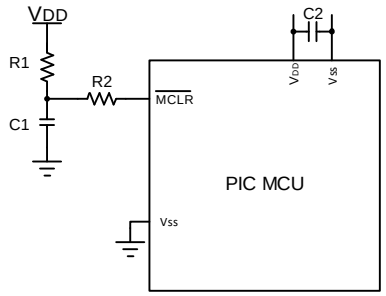
- PGC/PGD pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see "In-Circuit Serial Programming (ICSP) Pins")
- CLKIN pin when an external clock source is used.

Additionally, the following may be required:

- $V_{REF}+/V_{REF}-$ pins are used when external voltage reference for analog modules is implemented

The minimum recommended connections are shown in the figure below.

**Figure 4-1. Minimum Recommended Connections**




**Key (all values are recommendations):**
C1: 10 nF, 16V ceramic
C2: 0.1 µF, 16V ceramic
R1: 10 kΩ
R2: 100Ω to 470Ω

## 4.2 Power Supply Pins

### 4.2.1 Decoupling Capacitors

The use of decoupling capacitors on every pair of power supply pins ($V_{DD}$ and $V_{SS}$) is required.

Consider the following criteria when using decoupling capacitors:

- Value and type of capacitor: A 0.1 µF (100 nF), 10-25V capacitor is recommended. The capacitor should be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- Placement on the printed circuit board: The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).

- Handling high-frequency noise: If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of 0.01 µF to 0.001 µF. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1 µF in parallel with 0.001 µF).
- Maximizing performance: On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

### 4.2.2 Tank Capacitors

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

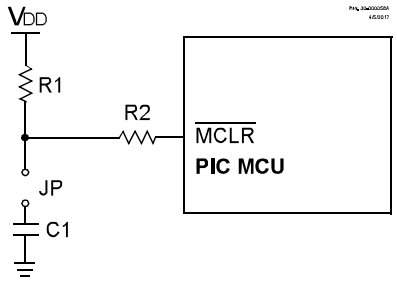
## 4.3 Master Clear ($\overline{MCLR}$) Pin

The $\overline{MCLR}$ pin provides two specific device functions: Device Reset, and Device Programming and Debugging. If programming and debugging are not required in the end application, a direct connection to $V_{DD}$ may be all that is required. The addition of other components, to help increase the application's resistance to spurious Resets from voltage sags, may be beneficial. A typical configuration is shown in Figure 4-1. Other circuit designs may be implemented, depending on the application's requirements.

During programming and debugging, the resistance and capacitance that can be added to the pin must be considered. Device programmers and debuggers drive the $\overline{MCLR}$ pin. Consequently, specific voltage levels ($V_{IH}$ and $V_{IL}$) and fast signal transitions must not be adversely affected. Therefore, specific values of R1 and C1 will need to be adjusted based on the application and PCB requirements. For example, it is recommended that the capacitor, C1, be isolated from the $\overline{MCLR}$ pin during programming and debugging operations by using a jumper (Figure 4-2). The jumper is replaced for normal run-time operations.

Any components associated with the $\overline{MCLR}$ pin should be placed within 0.25 inch (6 mm) of the pin.

**Figure 4-2. Example of $\overline{MCLR}$ Pin Connections**




**Notes:**

1. R1 ≤ 10 kΩ is recommended. A suggested starting value is 10 kΩ. Ensure that the $\overline{MCLR}$ pin $V_{IH}$ and $V_{IL}$ specifications are met.
2. R2 ≤ 470Ω will limit any current flowing into $\overline{MCLR}$ from the extended capacitor, C1, in the event of $\overline{MCLR}$ pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS). Ensure that the $\overline{MCLR}$ pin $V_{IH}$ and $V_{IL}$ specifications are met.

## 4.4 In-Circuit Serial Programming™ (ICSP™) Pins

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

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

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

## 4.5 Unused I/Os

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

# 5. Register and Bit Naming Conventions

## 5.1 Register Names

When there are multiple instances of the same peripheral in a device, the Peripheral Control registers will be depicted as the concatenation of a peripheral identifier, peripheral instance, and control identifier. The control registers section will show just one instance of all the register names with an 'x' in the place of the peripheral instance number. This naming convention may also be applied to peripherals when there is only one instance of that peripheral in the device to maintain compatibility with other devices in the family that contain more than one.

## 5.2 Bit Names

There are two variants for bit names:

- Short name: Bit function abbreviation
- Long name: Peripheral abbreviation + short name

### 5.2.1 Short Bit Names

Short bit names are an abbreviation for the bit function. For example, some peripherals are enabled with the EN bit. The bit names shown in the registers are the short name variant.

Short bit names are useful when accessing bits in C programs. The general format for accessing bits by the short name is RegisterNamebits.ShortName. For example, the enable bit, ON, in the ADCON0 register can be set in C programs with the instruction `ADCON0bits.ON = 1`.

Short names are generally not useful in assembly programs because the same name may be used by different peripherals in different bit positions. When this occurs, during the include file generation, all instances of that short bit name are appended with an underscore plus the name of the register in which the bit resides to avoid naming contentions.

### 5.2.2 Long Bit Names

Long bit names are constructed by adding a peripheral abbreviation prefix to the short name. The prefix is unique to the peripheral, thereby making every long bit name unique. The long bit name for the ADC enable bit is the ADC prefix, AD, appended with the enable bit short name, ON, resulting in the unique bit name ADON.

Long bit names are useful in both C and assembly programs. For example, in C the ADCON0 enable bit can be set with the `ADON = 1` instruction. In assembly, this bit can be set with the `BSF ADCON0,ADON` instruction.

### 5.2.3 Bit Fields

Bit fields are two or more adjacent bits in the same register. Bit fields adhere only to the short bit naming convention. For example, the three Least Significant bit of the ADCON2 register contain the ADC Operating Mode Selection bit. The short name for this field is MD and the long name is ADMD. Bit field access is only possible in C programs. The following example demonstrates a C program instruction for setting the ADC to operate in Accumulate mode:

```
ADCON2bits.MD = 0b001;
```

Individual bits in a bit field can also be accessed with long and short bit names. Each bit is the field name appended with the number of the bit position within the field. For example, the Most Significant mode bit has the short bit name MD2 and the long bit name is ADMD2. The following two examples demonstrate assembly program sequences for setting the ADC to operate in Accumulate mode:

```
MOVLW  ~(1<<MD2 | 1<<MD1)
ANDWF  ADCON2,F
```

```
MOVLW  1<<MD0
IORWF  ADCON2,F
```

```
BCF    ADCON2,ADMD2
BCF    ADCON2,ADMD1
BSF    ADCON2,ADMD0
```

## 5.3 Register and Bit Naming Exceptions

### 5.3.1 Status, Interrupt, and Mirror Bits

Status, Interrupt enables, Interrupt flags, and Mirror bits are contained in registers that span more than one peripheral. In these cases, the bit name shown is unique so there is no prefix or short name variant.

# 6. Register Legend

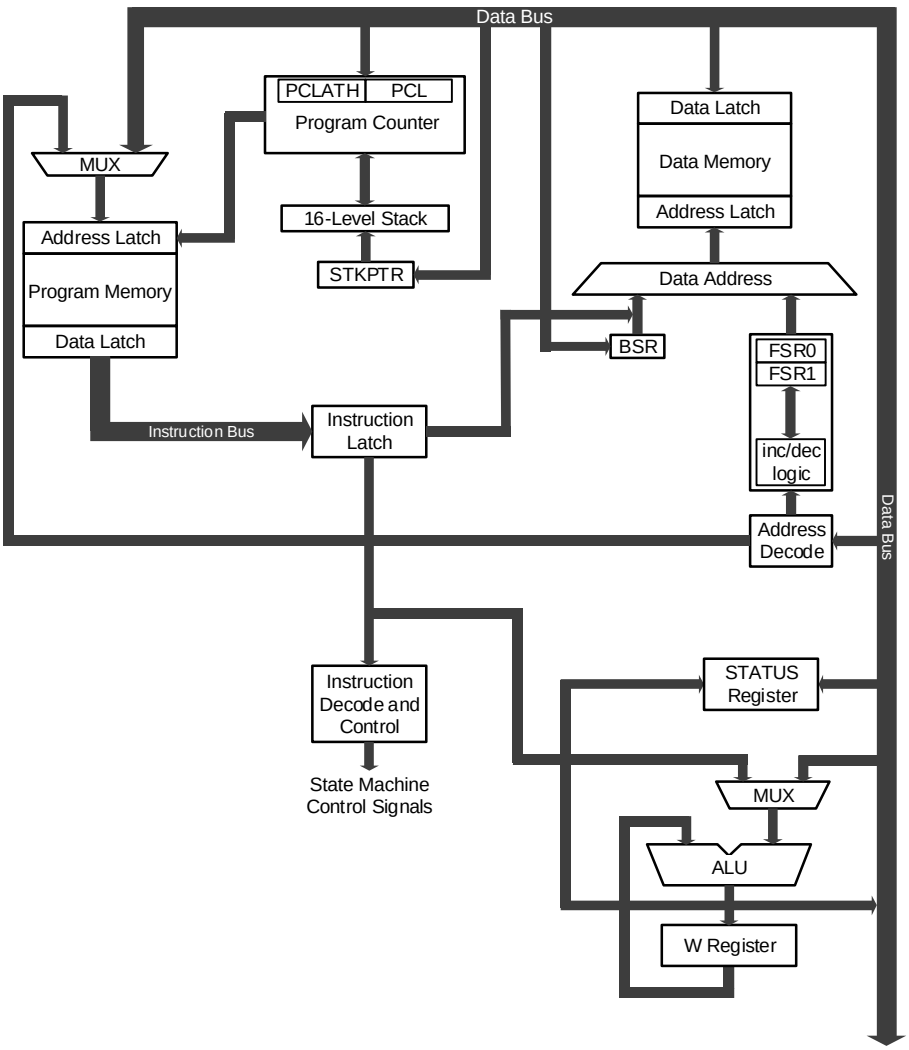
**Table 6-1. Register Legend**

| Symbol | Definition |
|---|---|
| R | Readable bit |
| W | Writable bit |
| HS | Hardware settable bit |
| HC | Hardware clearable bit |
| S | Set only bit |
| C | Clear only bit |
| U | Unimplemented bit, read as '0' |
| '1' | Bit value is set |
| '0' | Bit value is cleared |
| x | Bit value is unknown |
| u | Bit value is unchanged |
| q | Bit value depends on condition |
| m | Bit value is predefined |

# 7. Enhanced Mid-Range CPU

This family of devices contains an enhanced mid-range 8-bit CPU core. The CPU has 50 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.

**Figure 7-1. Core Data Path Diagram**




## 7.1 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.

## 7.2 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), and if enabled, will cause a software Reset.

## 7.3 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 also be addressed linearly, providing the ability to access contiguous data larger than 80 bytes.

## 7.4 Instruction Set

There are 50 instructions for the enhanced mid-range CPU to support the features of the CPU. See the **"Instruction Set Summary"** chapter for more details.

# 8. Device Configuration

Device configuration consists of the Configuration Words, User ID, Device ID, Device Information Area (DIA) and the Device Configuration Information (DCI) regions.

## 8.1 Configuration Words

There are five Configuration Words that allow the user to select the device oscillator, reset and memory protection options. These are implemented at addresses 0x8007 - 0x800B.

**Note:** The DEBUG bit in the 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'.

## 8.2 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.

The entire program memory space is protected from external reads and writes by the $\overline{CP}$ bit. 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. Self-writing the program memory is dependent upon the write protection setting.

## 8.3 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 bits determine which of the program memory blocks are protected.

## 8.4 User ID

Four words in the memory space (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 the **"VMREG Access to DIA, DCI, User ID, DEV/REV ID, and Configuration Words"** section for more information on accessing these memory locations. For more information on checksum calculation, see **"Memory Programming Specification"** section in the **"Electrical Specifications"** chapter for more information on accessing these memory locations. For more information on checksum calculation, see the **"Family Programming Specification"**.

## 8.5 Device ID and Revision ID

The 14-bit Device ID word is located at address 8006h and the 14-bit Revision ID is located at 8005h. These locations are read-only and cannot be erased or modified.

Development tools, such as device programmers and debuggers, may be used to read the Device ID, Revision ID and Configuration Words. Refer to the **"NVM - Nonvolatile Memory Control"** chapter for more information on accessing these locations.

## 8.6 Register Definitions: Configuration Settings

### 8.6.1 CONFIG1

**Name:** CONFIG1
**Address:** 0x8007

Configuration Word 1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | VDDAR | | | | $\overline{\text{CLKOUTEN}}$ |
| Access | | | | R/W | | | | R/W |
| Reset | | | | 1 | | | | 1 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | RSTOSC[1:0] | | | | FEXTOSC[1:0] | |
| Access | | | R/W | R/W | | | R/W | R/W |
| Reset | | | 0 | 1 | | | 0 | 1 |

**Bit 12 – VDDAR** $V_{DD}$ Analog Range Calibration Selection

| Value | Description |
|---|---|
| 1 | Internal analog systems are calibrated for operation between $V_{DD}$ = 2.3V - 5.5V |
| 0 | Internal analog systems are calibrated for operation between $V_{DD}$ = 1.8V - 3.6V |

**Bit 8 – $\overline{\text{CLKOUTEN}}$** Clock Out Enable

| Value | Description |
|---|---|
| 1 | CLKOUT function is disabled; I/O function on CLKOUT pin |
| 0 | CLKOUT function is enabled; $F_{OSC}$/4 clock appears on CLKOUT pin |

**Bits 5:4 – RSTOSC[1:0]** Power-up Default Value for COSC bits
Selects the oscillator source used by user software. Refer to COSC operation.

| Value | Description |
|---|---|
| 11 | EXTOSC operating per the FEXTOSC bits |
| 10 | HFINTOSC with FRQ = 1 MHz |
| 01 | LFINTOSC |
| 00 | HFINTOSC with FRQ = 32 MHz |

**Bits 1:0 – FEXTOSC[1:0]** External Oscillator Mode Selection

| Value | Description |
|---|---|
| 11 | ECH (16 MHz and higher) |
| 10 | ECL (below 16 MHz) |
| 01 | Oscillator not enabled |
| 00 | Reserved |

### 8.6.2 CONFIG2

**Name:** CONFIG2
**Address:** 0x8008

Configuration Word 2

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | $\overline{\text{DEBUG}}$ | STVREN | PPS1WAY | | BORV | |
| Access | | | R/W | R/W | R/W | | R/W | |
| Reset | | | 1 | 1 | 1 | | 1 | |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BOREN[1:0] | | | WDTE[1:0] | | PWRTS[1:0] | | MCLRE |
| Access | R/W | R/W | | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | | 1 | 1 | 1 | 1 | 1 |

**Bit 13 – $\overline{\text{DEBUG}}$** Debugger Enable[1]

| Value | Description |
|---|---|
| 1 | Background debugger disabled |
| 0 | Background debugger enabled |

**Bit 12 – STVREN** Stack Overflow/Underflow Reset Enable

| Value | Description |
|---|---|
| 1 | Stack Overflow or Underflow will cause a Reset |
| 0 | Stack Overflow or Underflow will not cause a Reset |

**Bit 11 – PPS1WAY** PPSLOCKED One-Way Set Enable

| Value | Description |
|---|---|
| 1 | The PPSLOCKED bit can only be set once after an unlocking sequence is executed; once PPSLOCKED is set, all future changes to PPS registers are prevented |
| 0 | The PPSLOCKED bit can be set and cleared as needed (unlocking sequence is required) |

**Bit 9 – BORV** Brown-out Reset (BOR) Voltage Selection[2]

| Value | Description |
|---|---|
| 1 | Brown-out Reset voltage ($V_{BOR}$) set to 1.9V |
| 0 | Brown-out Reset voltage ($V_{BOR}$) set to 2.85V |

**Bits 7:6 – BOREN[1:0]** Brown-out Reset (BOR) Enable[3]

| Value | Description |
|---|---|
| 11 | Brown-out Reset enabled, SBOREN bit is ignored |
| 10 | Brown-out Reset enabled while running, disabled in Sleep; SBOREN bit is ignored |
| 01 | Brown-out Reset enabled according to SBOREN |
| 00 | Brown-out Reset disabled |

**Bits 4:3 – WDTE[1:0]** Watchdog Timer (WDT) Enable

| Value | Description |
|---|---|
| 11 | WDT enabled regardless of Sleep; SEN bit of WDTCON is ignored |
| 10 | WDT enabled while Sleep = 0, suspended when Sleep = 1; SEN bit if WDTCON is ignored |
| 01 | WDT enabled/disabled by the SEN bit of WDTCON |
| 00 | WDT disabled, SEN bit of WDTCON is ignored |

**Bits 2:1 – PWRTS[1:0]** Power-Up Timer (PWRT) Selection

| Value | Description |
|---|---|
| 11 | PWRT disabled |
| 10 | PWRT is set at 64 ms |

| Value | Description |
|---|---|
| 01 | PWRT is set at 16 ms |
| 00 | PWRT is set at 1 ms |

**Bit 0 – MCLRE** Master Clear ($\overline{MCLR}$) Enable

| Value | Condition | Description |
|---|---|---|
| x | If LVP = 1 | $\overline{MCLR}$ pin is $\overline{MCLR}$ |
| 1 | If LVP = 0 | $\overline{MCLR}$ pin is $\overline{MCLR}$ |
| 0 | If LVP = 0 | $\overline{MCLR}$ pin function is port-defined function |

**Notes:**

1. The $\overline{DEBUG}$ bit is managed automatically by device development tools including debuggers and programmers. For normal device operation, this bit should be maintained as a '1'.
2. The higher voltage selection is recommended for operation at or above 16 MHz.
3. When enabled, Brown-out Reset voltage ($V_{BOR}$) is set by the BORV bit.

### 8.6.3 CONFIG3

**Name:** CONFIG3
**Address:** 0x8009

Configuration Word 3

**Note:** This register is reserved.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

### 8.6.4 CONFIG4

**Name:** CONFIG4
**Address:** 0x800A

Configuration Word 4

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | LVP | | $\overline{\text{WRTSAF}}$ | | $\overline{\text{WRTC}}$ | $\overline{\text{WRTB}}$ |
| Access | | | R/W | | R/W | | R/W | R/W |
| Reset | | | 1 | | 1 | | 1 | 1 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $\overline{\text{WRTAPP}}$ | | | $\overline{\text{SAFEN}}$ | $\overline{\text{BBEN}}$ | BBSIZE[2:0] | | |
| Access | R/W | | | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | | | 1 | 1 | 1 | 1 | 1 |

**Bit 13 – LVP** Low-Voltage Programming Enable[1]

| Value | Description |
|---|---|
| 1 | Low-Voltage Programming is enabled. $\overline{\text{MCLR}}/V_{PP}$ pin function is $\overline{\text{MCLR}}$. The MCLRE bit is ignored. |
| 0 | High voltage (HV) on $\overline{\text{MCLR}}/V_{PP}$ must be used for programming. |

**Bit 11 – $\overline{\text{WRTSAF}}$** Storage Area Flash (SAF) Write Protection[2,3]

| Value | Description |
|---|---|
| 1 | SAF is **NOT** write-protected |
| 0 | SAF is write-protected |

**Bit 9 – $\overline{\text{WRTC}}$** Configuration Registers Write-Protection[2]

| Value | Description |
|---|---|
| 1 | Configuration registers are **NOT** write-protected |
| 0 | Configuration registers are write-protected |

**Bit 8 – $\overline{\text{WRTB}}$** Boot Block Write-Protection[2,4]

| Value | Description |
|---|---|
| 1 | Boot Block is **NOT** write-protected |
| 0 | Boot Block is write-protected |

**Bit 7 – $\overline{\text{WRTAPP}}$** Application Block Write-Protection[2]

| Value | Description |
|---|---|
| 1 | Application Block is **NOT** write-protected |
| 0 | Application Block is write-protected |

**Bit 4 – $\overline{\text{SAFEN}}$** Storage Area Flash (SAF) Enable[2]

| Value | Description |
|---|---|
| 1 | SAF is disabled |
| 0 | SAF is enabled |

**Bit 3 – $\overline{\text{BBEN}}$** Boot Block Enable[2]

| Value | Description |
|---|---|
| 1 | Boot Block is disabled |
| 0 | Boot Block is enabled |

**Bits 2:0 – BBSIZE[2:0]** Boot Block Size Selection(5,6)

**Table 8-1. Boot Block Size**

| $\overline{\text{BBEN}}$ | BBSIZE | End Address of Boot Block | Boot Block Size (words) | | | |
|---|---|---|---|---|---|---|
| | | | PIC16F152x3 | PIC16F152x4 | PIC16F152x5 | PIC16F152x6 |
| 1 | xxx | – | – | | | |
| 0 | 111 | 01FFh | 512 | | | |
| 0 | 110 | 03FFh | 1024 | | | |
| 0 | 101 | 07FFh | –(6) | 2048 | | |
| 0 | 100 | 0FFFh | –(6) | | 4096 | |
| 0 | 011 | 1FFFh | –(6) | | | 8192 |
| 0 | 010 | 3FFFh | –(6) | | | |
| 0 | 001 | 3FFFh | –(6) | | | |
| 0 | 000 | 3FFFh | –(6) | | | |

**Notes:**

1. The LVP bit cannot be written (to zero) while operating from the LVP programming interface. The purpose of this rule is to prevent the user from dropping out of LVP mode while programming from LVP mode, or accidentally eliminating LVP mode from the Configuration state.
2. Once protection is enabled through ICSP™ or a self-write, it can only be reset through a Bulk Erase.
3. Applicable only if $\overline{\text{SAFEN}}$ = 0.
4. Applicable only if $\overline{\text{BBEN}}$ = 0.
5. BBSIZE[2:0] bits can only be changed when $\overline{\text{BBEN}}$ = 1. Once $\overline{\text{BBEN}}$ = 0, BBSIZE[2:0] can only be changed through a Bulk Erase.
6. The maximum Boot Block size is half of the user program memory size. Any selection that will exceed the half of a device's program memory will default to a maximum Boot Block size of half PFM. For example, all settings of BBSIZE from '110' to '000' for a PIC16F15213 (Max PFM = 2048 words) will result in a maximum Boot Block size of 1024 words.

### 8.6.5 CONFIG5

**Name:** CONFIG5
**Address:** 0x800B

Configuration Word 5[1]

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| Access | | | | | | | | |
| Reset | | | | | | | | |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | $\overline{CP}$ |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 1 |

**Bit 0 – $\overline{CP}$** User Program Flash Memory (PFM) Code Protection[2]

| Value | Description |
|---|---|
| 1 | User PFM code protection is disabled |
| 0 | User PFM code protection is enabled |

**Notes:**

1. Since device code protection takes effect immediately, this Configuration Word should be written last.
2. Once code protection is enabled it can only be removed through a Bulk Erase.

## 8.7 Register Definitions: Device ID and Revision ID

### 8.7.1 Device ID

**Name:** DEVICEID
**Address:** 8006h

Device ID Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | Reserved | Reserved | DEV[11:8] | | | |
| Access | | | R | R | R | R | R | R |
| Reset | | | 1 | 1 | q | q | q | q |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DEV[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | q | q | q | q | q | q | q | q |

**Bit 13 – Reserved** Reserved - Read as 1

**Bit 12 – Reserved** Reserved - Read as 1

**Bits 11:0 – DEV[11:0]** Device ID

| Device | Device ID |
|---|---|
| PIC16F15213 | 30E3h |
| PIC16F15214 | 30E6h |
| PIC16F15223 | 30E4h |
| PIC16F15224 | 30E7h |
| PIC16F15225 | 30E9h |
| PIC16F15243 | 30E5h |
| PIC16F15244 | 30E8h |
| PIC16F15245 | 30EAh |
| PIC16F15254 | 30F0h |
| PIC16F15255 | 30EFh |
| PIC16F15256 | 30EBh |
| PIC16F15274 | 30EEh |
| PIC16F15275 | 30EDh |
| PIC16F15276 | 30ECh |

### 8.7.2 Revision ID

**Name:** REVISIONID
**Address:** 8005h

Revision ID Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | Reserved | Reserved | MJRREV[5:2] | | | |
| Access | | | R | R | R | R | R | R |
| Reset | | | 1 | 0 | q | q | q | q |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MJRREV[1:0] | | MNRREV[5:0] | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | q | q | q | q | q | q | q | q |

**Bit 13 – Reserved** Reserved - Read as 1

**Bit 12 – Reserved** Reserved - Read as 0

**Bits 11:6 – MJRREV[5:0]** Major Revision ID
These bits are used to identify a major revision. (A0, B0, C0, etc.).

**Bits 5:0 – MNRREV[5:0]** Minor Revision ID
These bits are used to identify a minor revision.

# 9. Memory Organization

These devices contain the following types of memory:

- Program Memory
  - Configuration Words
  - Device ID
  - User ID
  - Program Flash Memory
  - Device Information Area (DIA)
  - Device Configuration Information (DCI)
  - Revision ID
- 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
- NVMREG access

## 9.1 Program Memory Organization

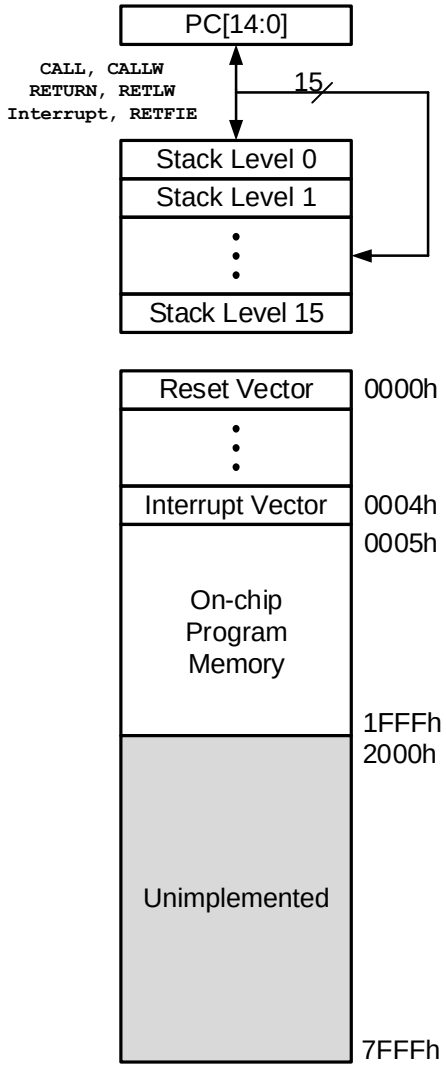
The enhanced mid-range core has a 15-bit Program Counter capable of addressing 32K x 14 program memory space. The table below shows the memory sizes implemented. Accessing a location above these boundaries will cause a wrap-around within the implemented memory space.

The Reset vector is at 0000h and the interrupt vector is at 0004h. Refer to the "**Interrupts**" chapter for more details.

**Table 9-1. Device Sizes And Addresses**

| Device | Program Memory Size (Words) | Last Program Memory Address |
|---|---|---|
| PIC16F15213 | 2048 | 07FFh |
| PIC16F15214 | 4096 | 0FFFh |
| PIC16F15223 | 2048 | 07FFh |
| PIC16F15224 | 4096 | 0FFFh |
| PIC16F15243 | 2048 | 07FFh |
| PIC16F15244 | 4096 | 0FFFh |

**Figure 9-1. Program Memory and Stack (PIC16F152x5)**




### 9.1.1 Reading Program Memory as Data

There are three methods of accessing constants in program memory. The first method is to use tables of `RETLW` instructions, the second, to set an FSR to point to the program memory, and the third is to use the NVMREG interface to access the program memory.

#### 9.1.1.1 `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 the following example.

**Example 9-1. Accessing table of constants using `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 eases the implementation of this type of table.

#### 9.1.1.2 Indirect Read with FSR

The program memory can be accessed as data by setting bit 7 of an 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 INDFx registers. Instructions that read the program memory via the FSR require one extra instruction cycle to complete. The following example demonstrates reading the program memory via an FSR.

The high directive will set bit 7 if a label points to a location in the program memory. This applies to the assembly code shown below.

**Example 9-2. Read of Program Memory using FSR Register**

```
constants
    RETLW DATA0         ;Index0 data
    RETLW DATA1         ;Index1 data
    RETLW DATA2
    RETLW DATA3
my_function
    ;… LOTS OF CODE…
    MOVLW    LOW constants
    MOVWF    FSR1L
    MOVLW    HIGH constants
    MOVWF    FSR1H
    MOVIW    2[FSR1]        ;DATA2 IS IN W
```

### 9.1.2 Memory Access Partition (MAP)

User Flash is partitioned into:

- Application Block
- Boot Block
- Storage Area Flash (SAF) Block

The user can allocate the memory usage by setting the $\overline{\text{BBEN}}$ bit, selecting the size of the partition defined by BBSIZE bits and enabling the Storage Area Flash by the $\overline{\text{SAFEN}}$ bit.

#### 9.1.2.1 Application Block

Default settings of the Configuration bits ($\overline{\text{BBEN}}$ = 1 and $\overline{\text{SAFEN}}$ = 1) assign all memory in the user Flash area to the application block.

#### 9.1.2.2 Boot Block

If $\overline{\text{BBEN}}$ = 1, the Boot Block is enabled and a specific address range is allotted as the Boot Block, based on the value of the BBSIZE bits.

#### 9.1.2.3 Storage Area Flash

Storage Area Flash (SAF) is enabled by clearing the $\overline{\text{SAFEN}}$ bit. If enabled, the SAF block is placed at the end of memory and spans 128 words. If the Storage Area Flash (SAF) is enabled, the SAF area is not available for program execution.

#### 9.1.2.4 Memory Write Protection

All the memory blocks have corresponding write protection bits (WRTAPP, WRTB and WRTC). If write-protected locations are written from NVMCON registers, the memory is not changed and the WRERR bit of the NVMCON1 register is set as explained in the **"WRERR Bit"** section in the **"NVM - Nonvolatile Memory Control"** chapter.

#### 9.1.2.5 Memory Violation

A Memory Execution Violation Reset occurs while executing an instruction that has been fetched from outside a valid execution area, clearing the MEMV bit. Refer to the **"Memory Execution Violation"** section in the **"Resets"** chapter for the available valid program execution areas and the PCON1 register definition for MEMV bit conditions.

**Table 9-2. Memory Access Partition**

| REG | Address | Partition | | | |
|---|---|---|---|---|---|
| | | BBEN = 1<br>SAFEN = 1 | BBEN = 1<br>SAFEN = 0 | BBEN = 0<br>SAFEN = 1 | BBEN = 0<br>SAFEN = 0 |
| PFM | 00 0000h ... Last Block Memory Address | Application Block(4) | Application Block(4) | Boot Block(4) | Boot Block(4) |
| | Last Boot Block Memory Address + 1(1) ... Last Program Memory Address - 80h | | | Application Block(4) | Application Block(4) |
| | Last Program Memory Address - 7Fh(2) ... Last Program Memory Address | | SAF(4) | | SAF(4) |
| CONFIG | Config Memory Address(3) | CONFIG | | | |

**Notes:**

1. Last Boot Block Memory Address is based on BBSIZE.
2. Last Program Memory Address is the Flash size given in the **"Program Memory Organization"** section in the **"NVM - Nonvolatile Memory Control"** chapter.
3. Config Memory Address are the address locations of the Configuration Words given in the **"NVMREG Access to DIA, DCI, User ID, DEV/REV ID, and Configuration Words"** section in the **"NVM - Nonvolatile Memory Control"** chapter.
4. Each memory block has a corresponding write protection fuse defined by the WRTAPP, WRTB, and WRTC bits.

### 9.1.3 Device Information Area (DIA)

The Device Information Area (DIA) is a dedicated region in the Program Flash Memory. The data is mapped from address 8100h to 811 Fh. These locations are read-only and cannot be erased or modified. The DIA contains the Microchip Unique Identifier words, and the Fixed Voltage Reference (FVR) voltage readings in millivolts (mV). Table 9-3 holds the DIA information for the PIC16F152 family of microcontrollers.

**Table 9-3. Device Information Area**

| Address Range | Name of Region | Standard Device Information |
|---|---|---|
| 8100h - 8108h | MUI0 | Microchip Unique Identifier (9 Words) |
| | MUI1 | |
| | MUI2 | |
| | MUI3 | |
| | MUI4 | |
| | MUI5 | |
| | MUI6 | |
| | MUI7 | |
| | MUI8 | |
| 8109h | MUI9 | Reserved (1 Word) |
| 810Ah - 8111h | EUI0 | Optional External Unique Identifier (8 Words) |
| | EUI1 | |
| | EUI2 | |
| | EUI3 | |
| | EUI4 | |
| | EUI5 | |
| | EUI6 | |
| | EUI7 | |
| 8112h - 8117h | Reserved | Reserved (6 Words) |
| 8118h | FVRA1X | ADC FVR1 output voltage for 1x setting (in mV) |
| 8119h | FVRA2X | ADC FVR1 output voltage for 2x setting (in mV) |
| 811Ah | FVRA4X | ADC FVR1 output voltage for 4x setting (in mV) |
| 811Bh - 811Fh | Reserved | Reserved (5 Words) |

### 9.1.3.1 Microchip Unique Identifier (MUI)

The PIC16F152 devices are individually encoded during final manufacturing with a Microchip Unique Identifier (MUI). The MUI cannot be erased by a Bulk Erase command or any other user-accessible means. This feature allows for manufacturing traceability of Microchip Technology devices in applications where this is required. It may also be used by the application manufacturer for a number of functions that require unverified unique identification, such as:

- Tracking the device
- Unique serial number

The MUI consists of nine program words. When taken together, these fields form a unique identifier. The MUI is stored in read-only locations, located between 8100h to 8108h in the DIA space. Table 9-3 lists the addresses of the identifier words.

**Important:** For applications that require verified unique identification, contact your Microchip Technology sales office to create a serialized quick turn programming option.

#### 9.1.3.2 External Unique Identifier (EUI)

The EUI data is stored at locations 810Ah to 8111h in the program memory region. This region is an optional space for placing application specific information. The data is coded per customer requirements during manufacturing. The EUI cannot be erased by a Bulk Erase command.

**Important:** Data is stored in this address range on receiving a request from the customer. The customer may contact the local sales representative or Field Applications Engineer, and provide them the unique identifier information that is required to be stored in this region.

#### 9.1.3.3 Fixed Voltage Reference Data

The Fixed Voltage Reference (FVR) is a stable voltage reference, independent of $V_{DD}$, with 1.024V, 2.048V or 4.096V selectable output levels. The output of the FVR can be configured to supply a reference voltage to the following:

- ADC input channel
- ADC positive reference

For more information on the FVR, refer to the **"FVR - Fixed Voltage Reference"** chapter.

The DIA stores measured FVR voltages for this device in mV for the different buffer settings of 1x, 2x or 4x.

- FVRA1X stores the value of ADC FVR1 Output Voltage for 1x setting (in mV)
- FVRA2X stores the value of ADC FVR1 Output Voltage for 2x setting (in mV)
- FVRA4X stores the value of ADC FVR1 Output Voltage for 4x setting (in mV)

### 9.1.4 Device Configuration Information (DCI)

The Device Configuration Information (DCI) is a dedicated region in the memory that holds information about the device, which is useful for programming and bootloader applications. The data stored in this region is read-only and cannot be modified/erased. Refer to the table below for complete DCI table addresses and description.

**Table 9-4. Device Configuration Information**

| Address | Name | Description | Value | | Units |
|---|---|---|---|---|---|
| | | | PIC16F15213<br>PIC16F15223<br>PIC16F15224 | PIC16F15214<br>PIC16F15224<br>PIC16F15244 | |
| 8200h | ERSIZ | Erase row size | 32 | | Words |
| 8201h | WLSIZ | Number of write latches per row | 32 | | Words |
| 8202h | URSIZ | Number of user erasable rows | 64 | 128 | Rows |
| 8203h | EESIZ | Data EEPROM memory size | 0 | | Bytes |
| 8204h | PCNT | Pin Count | 8/14/20 | 8/14/20 | Pins |

#### 9.1.4.1 DIA and DCI Access

The DIA and DCI data are read-only and cannot be erased or modified. See **"NVMREG Access to DIA, DCI, User ID, DEV/REV ID, and Configuration Words"** section in the **"NVM - Nonvolatile Memory Control"** chapter for more information on accessing these memory locations.

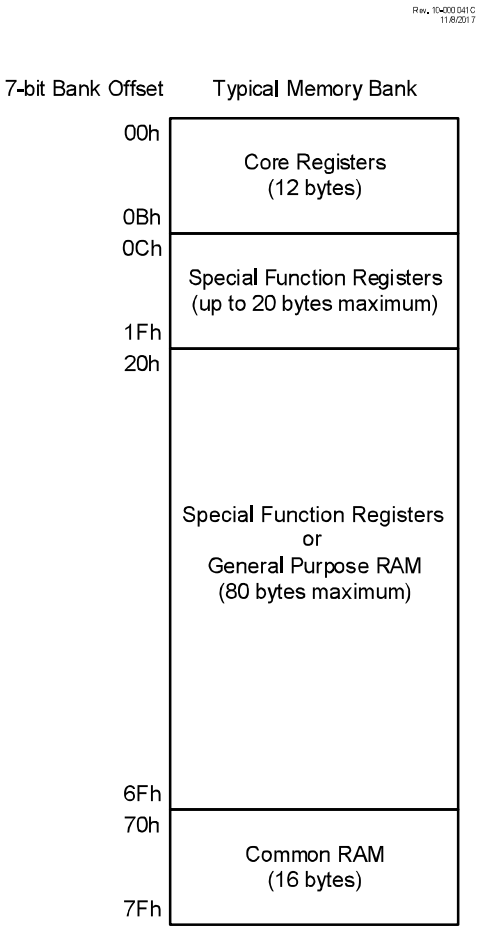
Development tools, such as device programmers and debuggers, may be used to read the DIA and DCI regions, similar to the Device ID and Revision ID.

## 9.2 Data Memory Organization

The data memory is partitioned into up to 64 memory banks with 128 bytes in each bank. Each bank consists of:

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

**Figure 9-2. Banked Memory Partition**




### 9.2.1 Bank Selection

The active bank is selected by writing the bank number into the Bank Select Register (BSR). All data memory can be accessed either directly via instructions that use the file registers, or indirectly via the two File Select Registers (FSRs). Data memory uses a 13-bit address. The upper six bits of the address define the Bank Address and the lower seven bits select the registers/RAM in that bank.

### 9.2.2 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. These registers are listed in Table 9-5.

**Table 9-5. Core Registers**

| Addresses in BANKx | Core Registers |
|---|---|
| x00h or x80h | INDF0 |
| x01h or x81h | INDF1 |

| .....continued | |
|---|---|
| Addresses in BANKx | Core Registers |
| 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 |

### 9.2.3 Special Function Register

The Special Function Registers (SFR) are registers used by the application to control the desired operation of peripheral functions in the device. The SFRs occupy the first 20 bytes of the data banks 0-59 and the first 100 bytes of the data banks 60-63, after the core registers.

The SFRs associated with the operation of the peripherals are described in the appropriate peripheral chapter of this data sheet.

### 9.2.4 General Purpose RAM

There are up to 80 bytes of GPR in each data memory bank. The general purpose RAM can be accessed in a non-banked method via the FSRs. This can simplify access to large memory structures.

Refer to the "Linear Data Memory" section in the "Memory Organization" chapter for details about linear memory accessing.

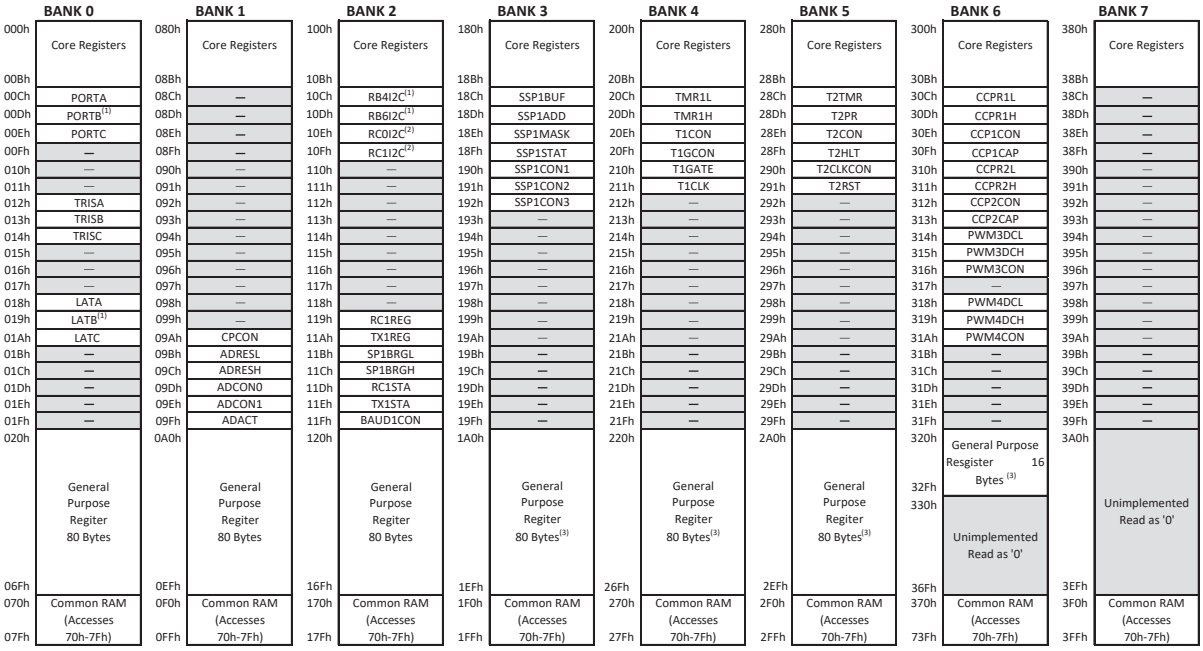
### 9.2.5 Common RAM

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

### 9.2.6 Device Memory Maps

The memory maps for the devices in this data sheet are listed in the following figures.

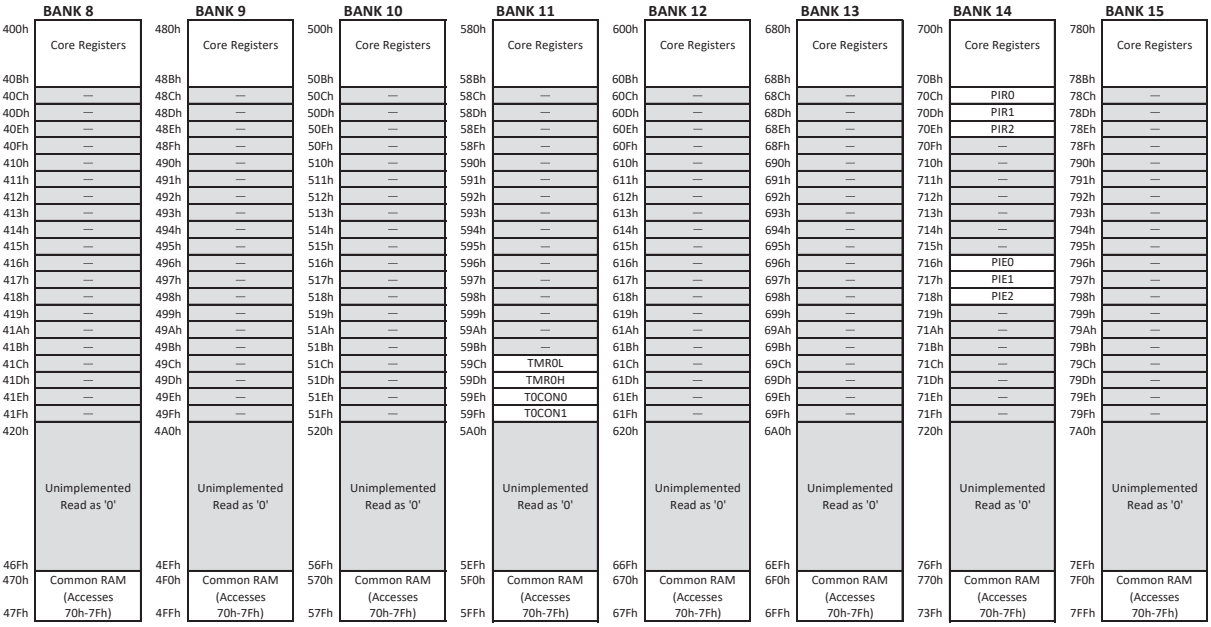
## Figure 9-3. Memory Map - Banks 0 - 7




Note: 1. Available on PIC16F15243/44 devices only
2. Not available on PIC16F15213/14 devices
3. Available on PIC16F15214/24/44 devices only

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

## Figure 9-4. Memory Map - Banks 8 - 15




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

**Figure 9-5. Memory Maps Banks 16 - 23**

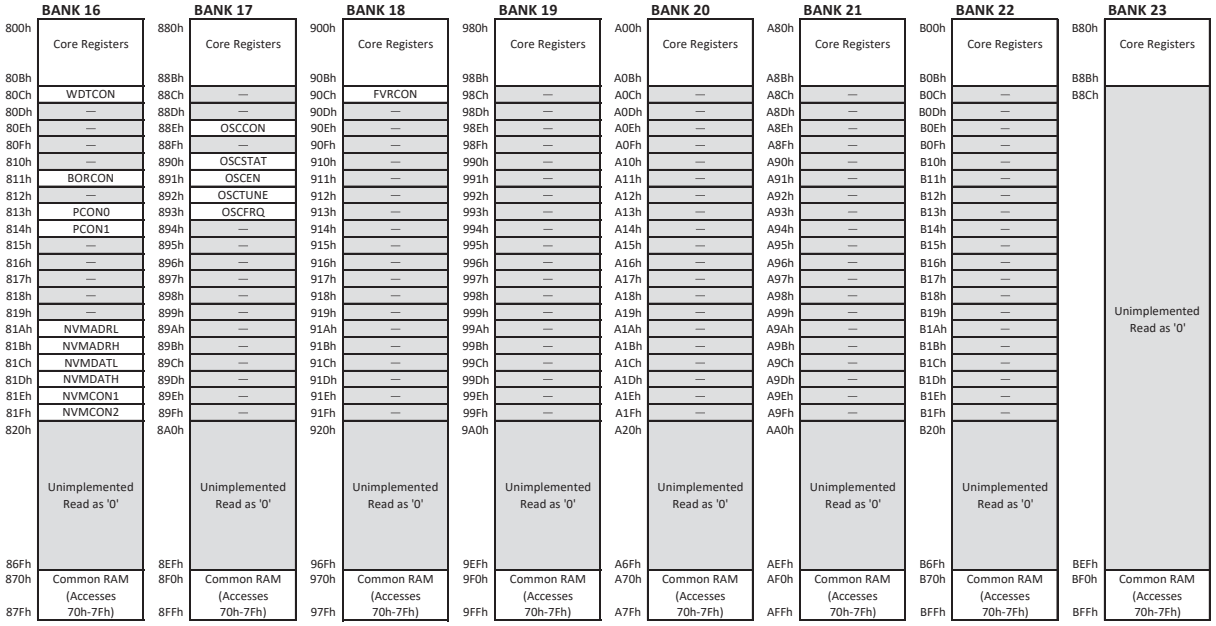



**Figure 9-6. Memory Map Banks 56 - 63**

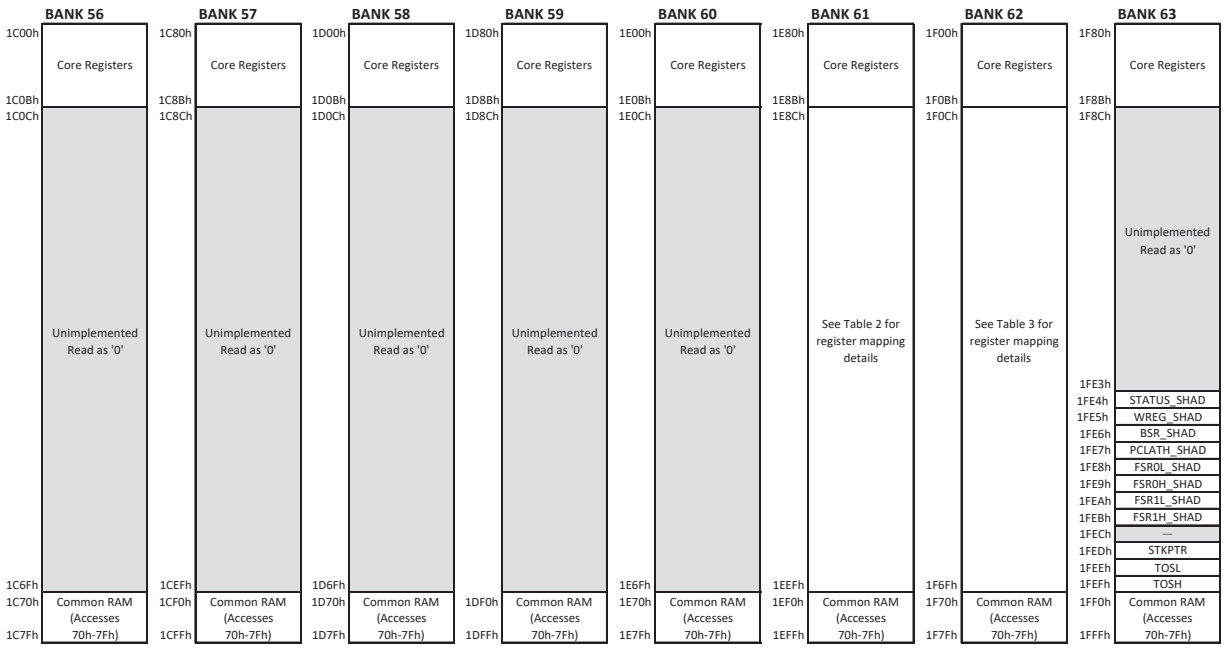


**Figure 9-7. Memory Map - Bank 61**

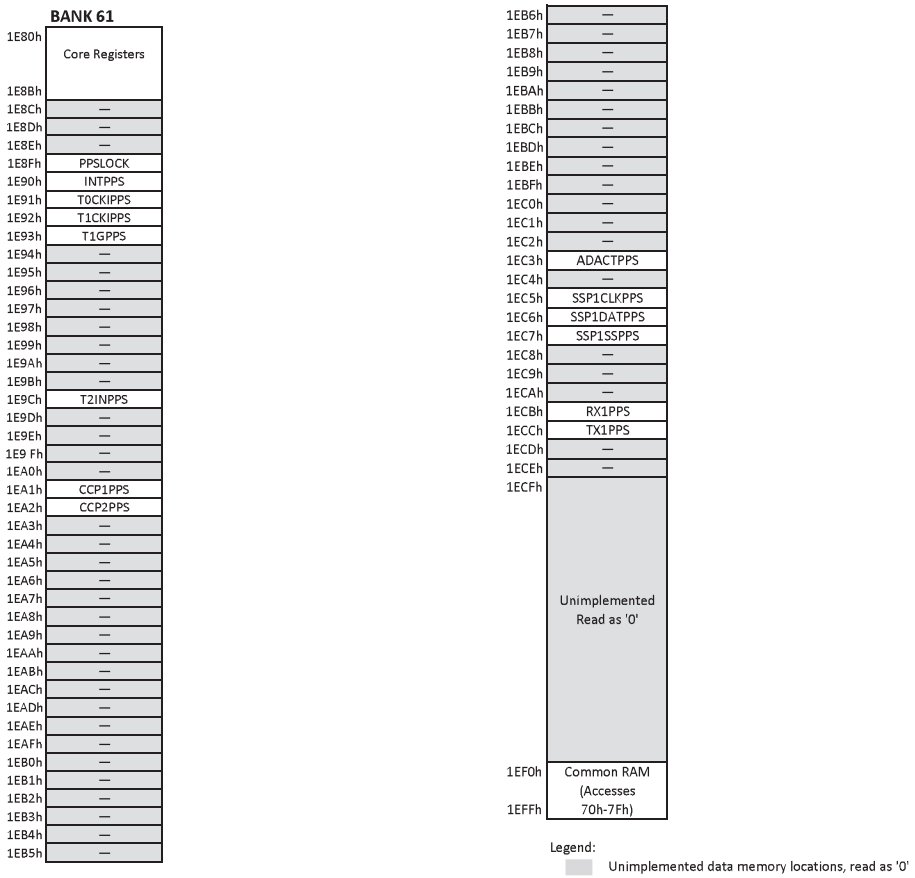
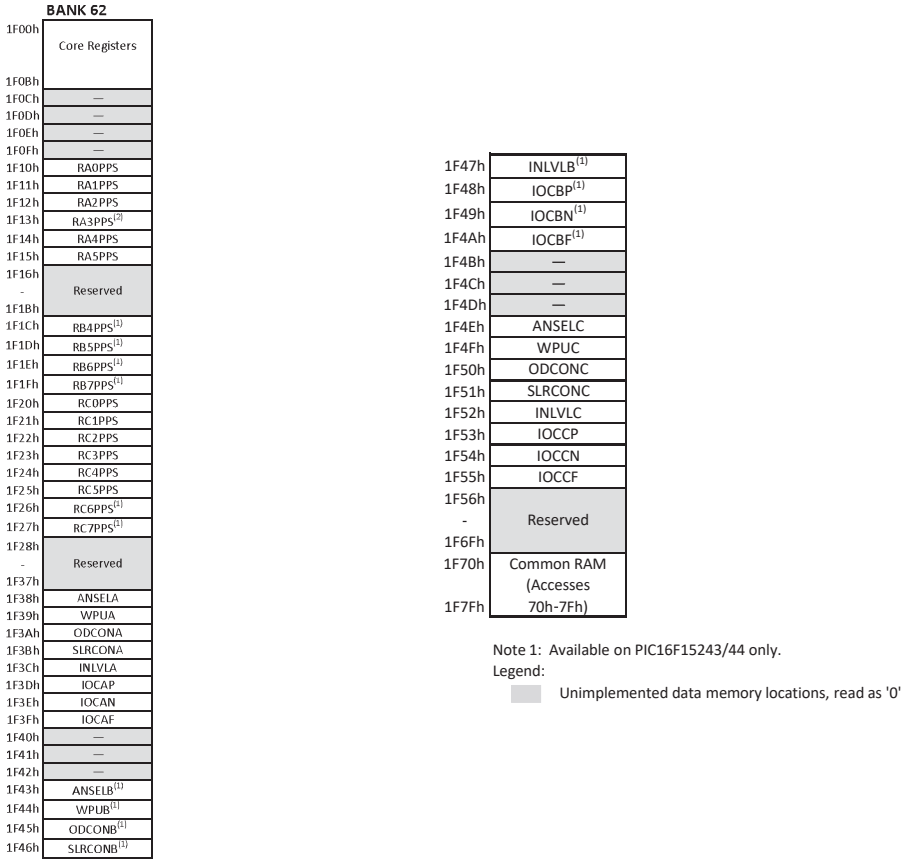


**Figure 9-8. Memory Map - Bank 62**




## 9.3 STATUS Register

The STATUS register 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 writes to these three bits are 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 bits [4:3] and [1:0], and set the Z bit. This leaves the STATUS register as '000u uluu' (where u = unchanged).

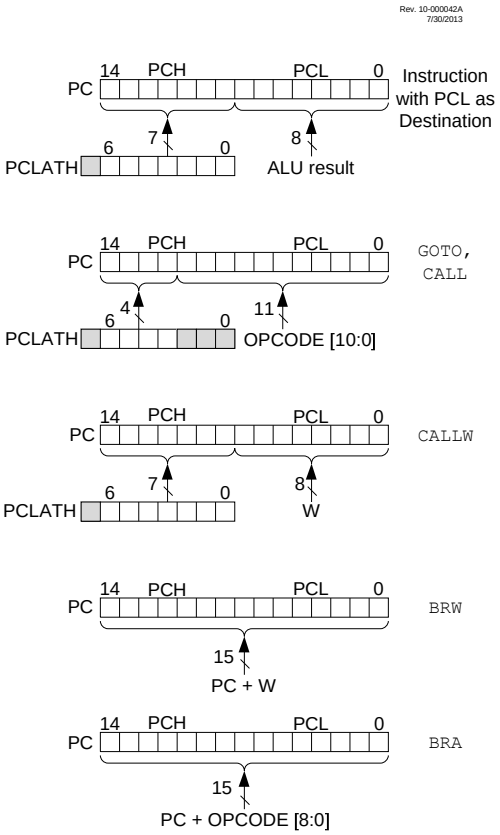
It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect any Status bits. For other instructions not affecting any Status bits, refer to the **"Instruction Set Summary"** chapter.

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

## 9.4 PCL and PCLATH

The Program Counter (PC) is 15 bits wide. The low byte comes from the PCL register, which is a readable and writable register. The high byte (PC[14:8]) is not directly readable or writable and comes from PCLATH. On any Reset the PC is cleared. Figure 9-9 shows the five situations for the loading of the PC.

**Figure 9-9. Loading of PC in Different Situations**




### 9.4.1 Modifying PCL

Executing any instruction with the PCL register as the destination simultaneously causes the Program Counter PC[14:8] bits (PCH) to be replaced by the contents of the PCLATH register. This allows the entire contents of the Program Counter to be changed by writing the desired upper 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.

### 9.4.2 Computed GOTO

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

### 9.4.3 Computed Function Calls

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

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

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

### 9.4.4 Branching

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

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

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

## 9.5 Stack

All devices have a 16-level by 15-bit wide hardware stack. 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 Configuration bit is programmed to '0'. 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.

If the STVREN bit is programmed to '1', the device will be Reset if the stack is PUSHed beyond the sixteenth level or POPed beyond the fist level, setting the appropriate bits (STKOVF or STKUNF, respectively).

**Important:** 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.

### 9.5.1 Accessing the Stack

The stack is accessible 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.

**Important:** 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. STKPTR can be monitored to obtain the value of stack memory left at any given time. 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 value from the stack and then decrement the STKPTR.

Reference the following figures for examples of accessing the stack.

**Figure 9-10. Accessing the Stack Example 1**

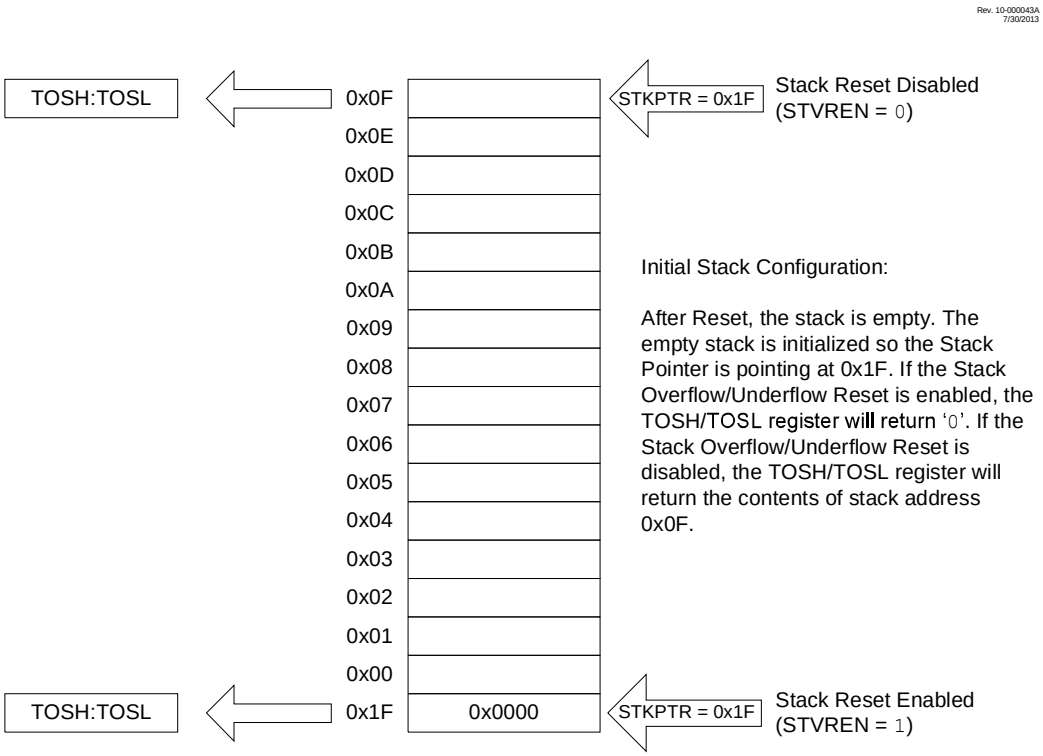



**Figure 9-11. Accessing the Stack Example 2**

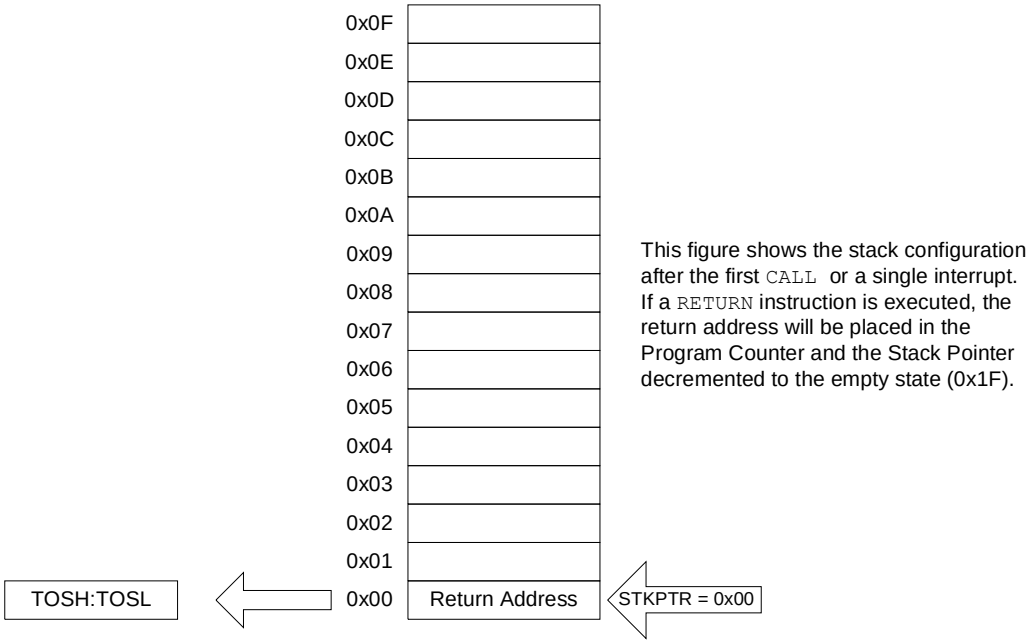


**Figure 9-12. Accessing the Stack Example 3**

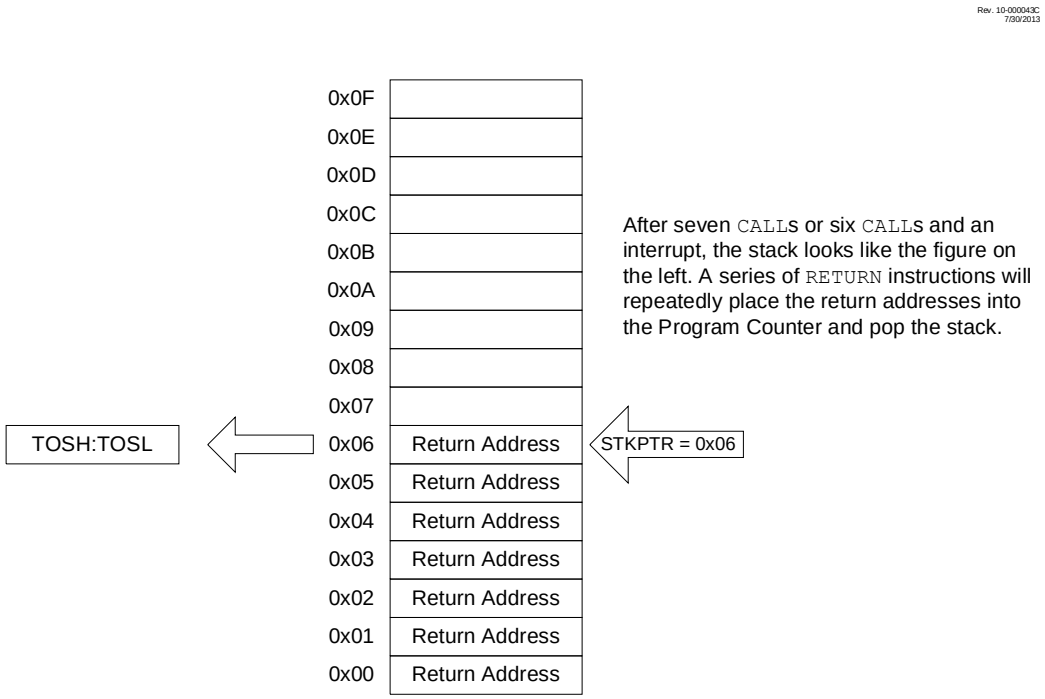
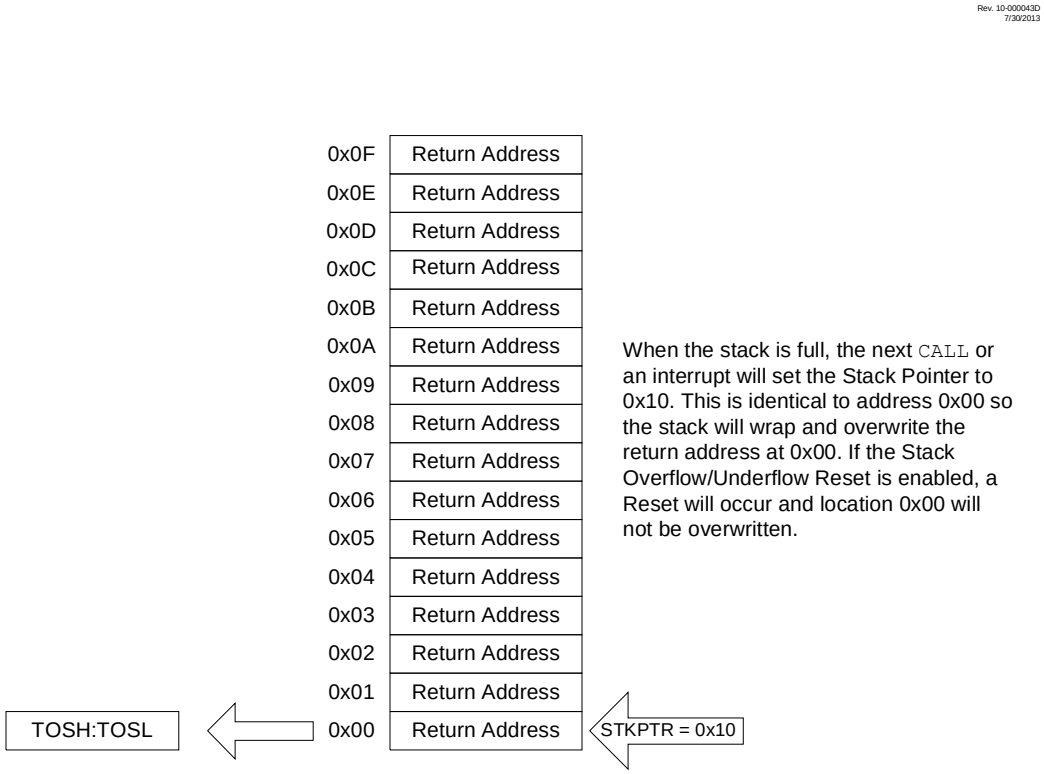



**Figure 9-13. Accessing the Stack Example 4**



### 9.5.2 Overflow/Underflow Reset

If the STVREN bit 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).

## 9.6 Indirect Addressing

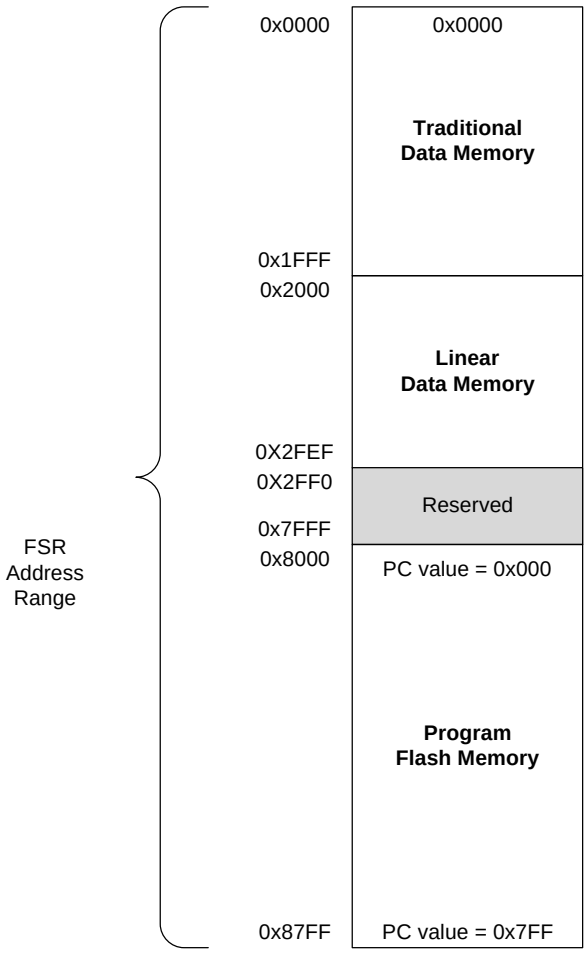
The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the File Select Registers (FSR). If the FSRn address specifies one of the two INDFn registers, the read will return '0' and the write will not occur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

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

- Traditional/Banked Data Memory
- Linear Data Memory
- Program Flash Memory

**Figure 9-14. Indirect Addressing PIC16F152**

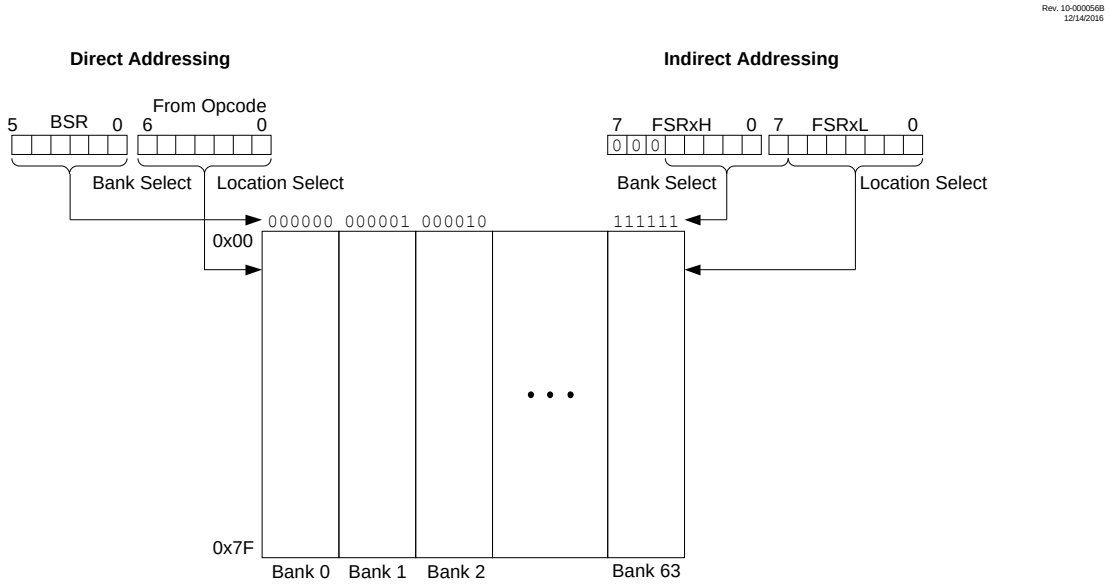
Rev. 10-000044F
1/13/2017



### 9.6.1 Traditional/Banked Data Memory

The traditional or banked data memory is a region from FSR address 0x0000 to FSR address 0x1FFF. The addresses correspond to the absolute addresses of all SFR, GPR and common registers.

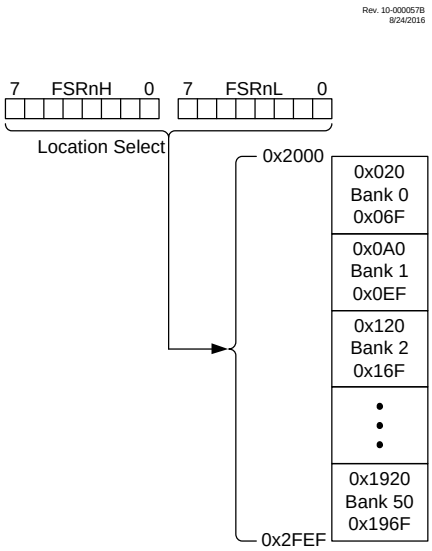
**Figure 9-15. Traditional/Banked Data Memory Map**




### 9.6.2 Linear Data Memory

The linear data memory is the region from FSR address 0x2000 to FSR address 0x2FEF. This region is a virtual region that points back to the 80-byte blocks of GPR memory in all the banks. Refer to Figure 9-16 for the Linear Data Memory Map.

**Figure 9-16. Linear Data Memory Map**



**Important:** The address range 0x2000 to 0x2FEF represents the complete addressable Linear Data Memory for PIC® devices (up to Bank 50). The actual implemented Linear Data Memory will differ from one device to the other in a family.

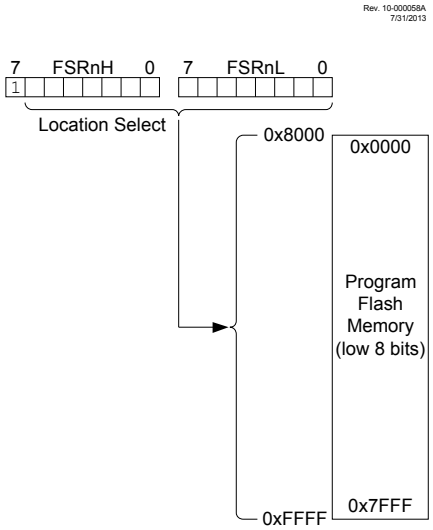
Unimplemented memory reads as 0x00. Use of the linear data memory region allows buffers to be larger than 80 bytes because incrementing the FSR beyond one bank will go directly to the GPR memory of the next bank.

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

### 9.6.3 Program Flash Memory

To make constant data access easier, the entire Program Flash Memory is mapped to the upper half of the FSR address space. When the MSB of FSRnH is set, the lower 15 bits are the address in program memory which will be accessed through INDF. Only the lower eight bits of each memory location are 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 9-17. Program Flash Memory Map**




## 9.7 Register Definitions: Memory Organization

### 9.7.1 INDF0

**Name:** INDF0
**Address:** 0x000

Indirect Data Register. This is a virtual register. The GPR/SFR register addressed by the FSR0 register is the target for all operations involving the INDF0 register.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INDF0[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – INDF0[7:0]**
Indirect data pointed to by the FSR0 register

### 9.7.2 INDF1

**Name:** INDF1
**Address:** 0x001

Indirect Data Register. This is a virtual register. The GPR/SFR register addressed by the FSR1 register is the target for all operations involving the INDF1 register.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INDF1[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – INDF1[7:0]**
Indirect data pointed to by the FSR1 register

### 9.7.3 PCL

**Name:** PCL
**Address:** 0x002

Low byte of the Program Counter

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PCL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – PCL[7:0]**
Provides direct read and write access to the Program Counter

### 9.7.4 STATUS

**Name:** STATUS
**Address:** 0x003

Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C |
| Access | | | | R | R | R/W | R/W | R/W |
| Reset | | | | 1 | 1 | 0 | 0 | 0 |

**Bit 4 – $\overline{TO}$** Time-Out
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | Set at power-up or by execution of CLRWDT or SLEEP instruction |
| 0 | A WDT time-out occurred |

**Bit 3 – $\overline{PD}$** Power-Down
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | Set at power-up or by execution of CLRWDT instruction |
| 0 | Cleared by execution of the SLEEP instruction |

**Bit 2 – Z** Zero
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 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/$\overline{Borrow}$(1)
ADDWF, ADDLW, SUBLW, SUBWF instructions
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | A carry-out from the 4th low-order bit of the result occurred |
| 0 | No carry-out from the 4th low-order bit of the result |

**Bit 0 – C** Carry/$\overline{Borrow}$(1)
ADDWF, ADDLW, SUBLW, SUBWF instructions
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | A carry-out from the Most Significant bit of the result occurred |
| 0 | No carry-out from the Most Significant bit of the result occurred |

**Note:**

1. For $\overline{Borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For Rotate (RRCF, RLCF) instructions, this bit is loaded with either the high or low-order bit of the Source register.

### 9.7.5 FSR0

**Name:** FSR0
**Address:** 0x004

Indirect Address Register

The FSR0 value is the address of the data to which the INDF0 register points.
**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- FSR0H: Accesses the high byte FSR0[15:8]
- FSR0L: Accesses the low byte FSR0[7:0]

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | FSR0[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FSR0[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – FSR0[15:0]** Address of INDF0 data

### 9.7.6 FSR1

**Name:** FSR1
**Address:** 0x006

Indirect Address Register

The FSR1 value is the address of the data to which the INDF1 register points.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | FSR1[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FSR1[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – FSR1[15:0]**
Address of INDF1 data

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- FSR1H: Accesses the high byte FSR1[15:8]
- FSR1L: Accesses the low byte FSR1[7:0]

### 9.7.7 BSR

**Name:** BSR
**Address:** 0x008

Bank Select Register
The BSR indicates the data memory bank by writing the bank number into the register. All data memory can be accessed directly via instructions, or indirectly via FSRs.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | BSR[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – BSR[5:0]**
Six Most Significant bits of the data memory address

### 9.7.8 WREG

**Name:** WREG
**Address:** 0x009

Working Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WREG[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – WREG[7:0]**

### 9.7.9 PCLATH

**Name:** PCLATH
**Address:** 0x00A

Program Counter Latches.

Write Buffer for the upper 7 bits of the Program Counter

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | PCLATH[6:0] | | | | | | |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 6:0 – PCLATH[6:0]** High PC Latch Register
Holding register for Program Counter bits [6:0]

## 9.8 Register Summary: Memory Organization

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | INDF0 | 7:0 | INDF0[7:0] | | | | | | | |
| 0x01 | INDF1 | 7:0 | INDF1[7:0] | | | | | | | |
| 0x02 | PCL | 7:0 | PCL[7:0] | | | | | | | |
| 0x03 | STATUS | 7:0 | | | | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C |
| 0x04 | FSR0 | 7:0 | FSR0[7:0] | | | | | | | |
| | | 15:8 | FSR0[15:8] | | | | | | | |
| 0x06 | FSR1 | 7:0 | FSR1[7:0] | | | | | | | |
| | | 15:8 | FSR1[15:8] | | | | | | | |
| 0x08 | BSR | 7:0 | | | BSR[5:0] | | | | | |
| 0x09 | WREG | 7:0 | WREG[7:0] | | | | | | | |
| 0x0A | PCLATH | 7:0 | | PCLATH[6:0] | | | | | | |

# 10. Resets

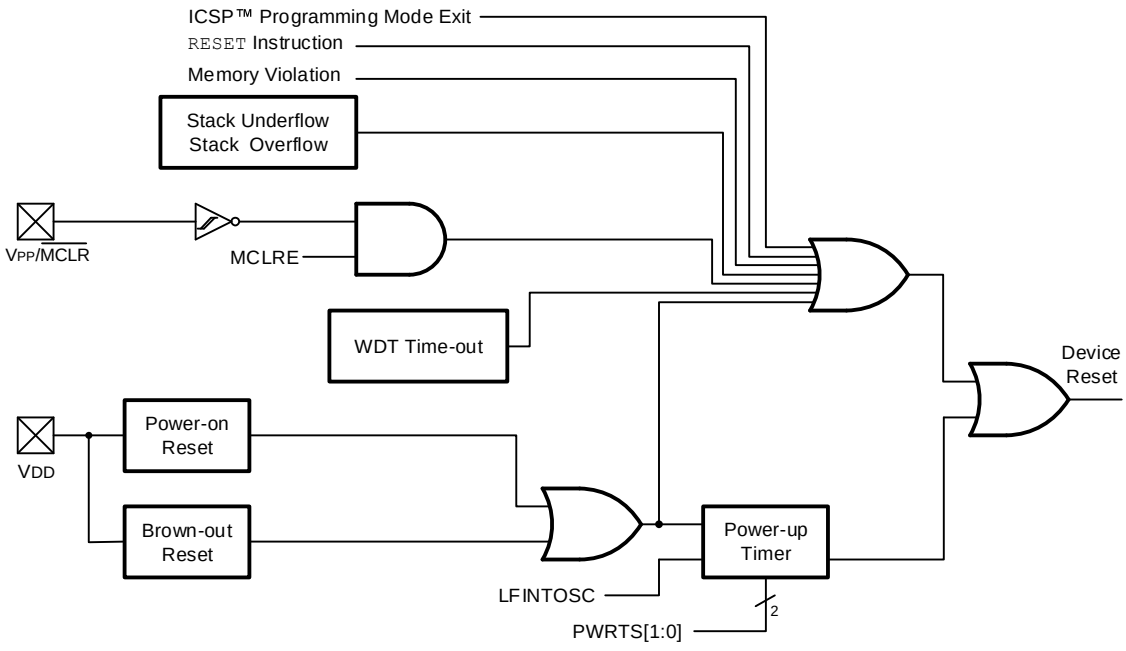
There are multiple ways to reset this device:

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

To allow $V_{DD}$ 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 10-1.

**Figure 10-1. Simplified Block Diagram of On-Chip Reset Circuit**




## 10.1 Power-on Reset (POR)

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

### 10.1.1 Programming Mode Exit

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

## 10.2 Brown-out Reset (BOR)

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

The Brown-out Reset module has four operating modes controlled by the BOREN bits. 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 10-1 for more information.

The Brown-out Reset voltage level is selectable by configuring the BORV bits.

A $V_{DD}$ noise rejection filter prevents the BOR from triggering on small events. If $V_{DD}$ falls below $V_{BOR}$ for a duration greater than parameter $T_{BORDC}$, the device will reset and the $\overline{BOR}$ bit will be cleared, indicating the Brown-out Reset condition occurred. See Figure 10-2.

### 10.2.1 BOR is Always ON

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

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

### 10.2.2 BOR is OFF in Sleep

When the BOREN bits are programmed to '10', the BOR is on, except in Sleep. BOR protection is not active during Sleep, but device wake-up will be delayed until the BOR can determine that $V_{DD}$ is higher than the BOR threshold. The device wake-up will be delayed until the BOR is ready.

### 10.2.3 BOR Controlled by Software

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

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

BOR protection is unchanged by Sleep.

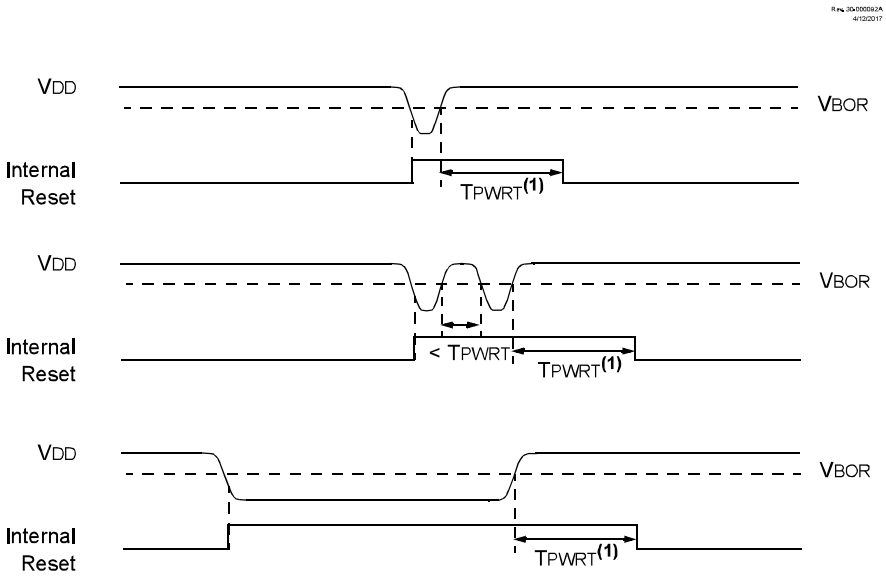
**Table 10-1. BOR Operating Modes**

| BOREN | SBOREN | Device Mode | BOR Mode | Instruction Execution upon: | |
|---|---|---|---|---|---|
| | | | | Release of POR | Wake-up from Sleep |
| 11(1) | X | X | Active | Wait for release of BOR (BORRDY = 1) | Begins immediately |
| 10 | X | Awake | Active | Wait for release of BOR (BORRDY = 1) | N/A |
| | | Sleep | Hibernate | N/A | Wait for release of BOR (BORRDY = 1) |
| 01 | 1 | X | Active | Wait for release of BOR (BORRDY = 1) | Begins immediately |
| | 0 | X | Hibernate | | |
| 00 | X | X | Disabled | Begins immediately | |

**Note:**

1. In this specific case, '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 bits.

**Figure 10-2. Brown-out Situations**




**Note:** $T_{PWRT}$ delay when the PWRTS bits are enabled ($\overline{PWRTS}$ != 00).

### 10.2.4 BOR is Always OFF

When the BOREN bits are programmed to '00', the BOR is always disabled. In the configuration, setting the SBOREN bit will have no affect on BOR operations.

## 10.3 $\overline{MCLR}$ Reset

The $\overline{MCLR}$ is an optional external input that can reset the device. The $\overline{MCLR}$ function is controlled by the MCLRE bit and the LVP bit (see Table 10-2). The $\overline{RMCLR}$ bit will be set to '0' if a $\overline{MCLR}$ has occurred.

**Table 10-2. $\overline{MCLR}$ Configuration**

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

### 10.3.1 $\overline{MCLR}$ 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 $V_{DD}$ 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.

**Important:** An internal Reset event (RESET instruction, BOR, WDT, POR, STKOVF, STKUNF) does not drive the $\overline{MCLR}$ pin low.

### 10.3.2 $\overline{\text{MCLR}}$ Disabled

When $\overline{\text{MCLR}}$ is disabled, the $\overline{\text{MCLR}}$ becomes input-only and pin functions such as internal weak pull-ups are under software control.

## 10.4 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 TO, PD and $\overline{\text{RWDT}}$ bits are changed to indicate a WDT Reset caused by the timer overflowing.

## 10.5 RESET Instruction

A RESET instruction will cause a device Reset. The $\overline{\text{RI}}$ bit will be set to '0'. See Table 10-4 for default conditions after a RESET instruction has occurred.

## 10.6 Stack Overflow/Underflow Reset

The device can reset when the Stack Overflows or Underflows. The STKOVF or STKUNF bits indicate the Reset condition. These Resets are enabled by setting the STVREN bit.

## 10.7 Power-up Timer (PWRT)

The Power-up Timer provides up to a 64 ms time-out period on POR or Brown-out Reset. The device is held in Reset as long as PWRT is active. The PWRT delay allows additional time for the $V_{DD}$ to rise to an acceptable level.

The Power-up Timer is controlled by the PWRTS bits. 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).

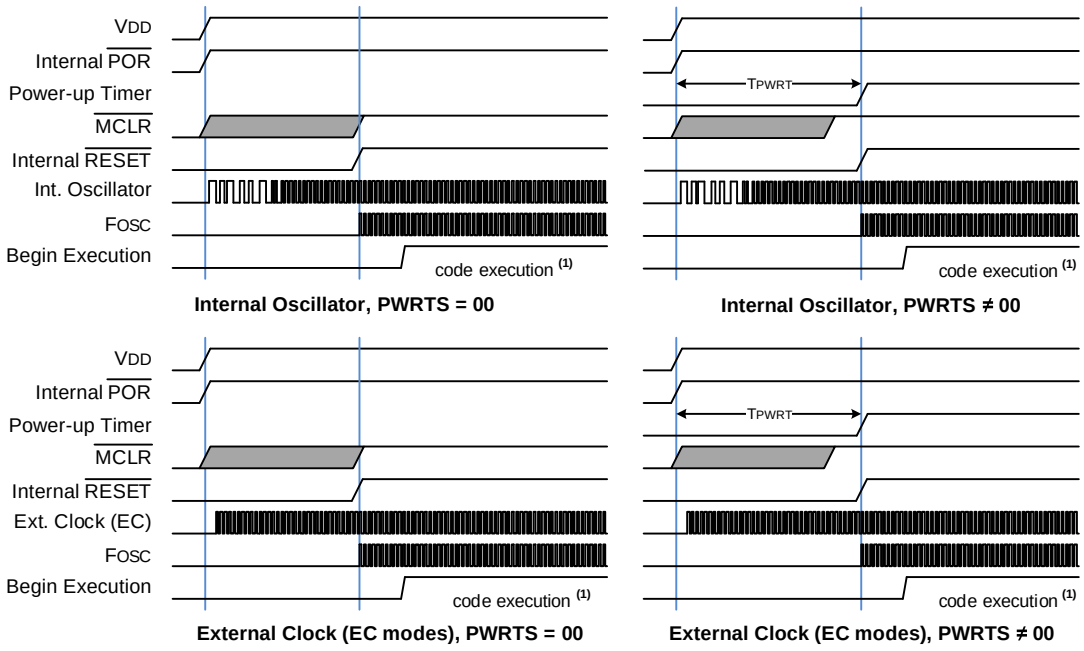
## 10.8 Start-up Sequence

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. $\overline{\text{MCLR}}$ must be released (if enabled).

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

**Figure 10-3. Reset Start-up Sequence**




**Note:**

1. Code execution begins 10 $F_{OSC}$ cycles after the $F_{OSC}$ clock is released.

## 10.9 Memory Execution Violation

A memory execution violation Reset occurs if executing an instruction being fetched from outside the valid execution area. The invalid execution areas are:

1. Addresses outside implemented program memory. Refer to the **"Memory Organization"** chapter for details about available Flash size.
2. Storage Area Flash (SAF) inside program memory, if it is enabled

When a memory execution violation is generated, the device is reset and the $\overline{MEMV}$ bit is cleared to signal the cause of the Reset. The $\overline{MEMV}$ bit must be set in the user code after a memory execution violation Reset has occurred to detect further violation Resets.

## 10.10 Determining the Cause of a Reset

Upon any Reset, multiple bits in the STATUS, PCON0 and PCON1 registers are updated to indicate the cause of the Reset. The following tables show the Reset conditions of these registers.

**Table 10-3. Reset Status Bits and Their Significance**

| STKOVF | STKUNF | $\overline{RWDT}$ | $\overline{RMCLR}$ | $\overline{RI}$ | $\overline{POR}$ | $\overline{BOR}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{MEMV}$ | Condition |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | x | 1 | 1 | 1 | Power-on Reset |
| 0 | 0 | 1 | 1 | 1 | 0 | x | 0 | x | u | Illegal, $\overline{TO}$ is set on POR |
| 0 | 0 | 1 | 1 | 1 | 0 | x | x | 0 | u | Illegal, $\overline{PD}$ is set on POR |

| .....continued | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| STKOVF | STKUNF | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{MEMV}}$ | Condition |
| 0 | 0 | u | 1 | 1 | u | 0 | 1 | 1 | u | Brown-out Reset |
| u | u | 0 | u | u | u | u | 0 | u | u | WDT Reset |
| u | u | u | u | u | u | u | 0 | 0 | u | WDT Wake-up from Sleep |
| u | u | u | u | u | u | u | 1 | 0 | u | Interrupt Wake-up from Sleep |
| u | u | u | 0 | u | u | u | u | u | 1 | $\overline{\text{MCLR}}$ Reset during normal operation |
| u | u | u | 0 | u | u | u | 1 | 0 | u | $\overline{\text{MCLR}}$ Reset during Sleep |
| u | u | u | u | 0 | u | u | u | u | u | RESET Instruction Executed |
| 1 | u | u | u | u | u | u | u | u | u | Stack Overflow Reset (STVREN = 1) |
| u | 1 | u | u | u | u | u | u | u | u | Stack Underflow Reset (STVREN = 1) |
| u | u | u | u | u | u | u | u | u | 0 | Memory Violation Reset |

**Table 10-4. Reset Conditions for Special Registers**

| Condition | Program Counter | STATUS Register | PCON0 Register | PCON1 Register |
|---|---|---|---|---|
| Power-on Reset | 0 | ---1 1000 | 0011 110x | ---- --1- |
| Brown-out Reset | 0 | ---1 1000 | 0011 11u0 | ---- --u- |
| $\overline{\text{MCLR}}$ Reset during normal operation | 0 | -uuu uuuu | uuuu 0uuu | ---- --1- |
| $\overline{\text{MCLR}}$ Reset during Sleep | 0 | ---1 0uuu | uuuu 0uuu | ---- --u- |
| WDT Time-out Reset | 0 | ---0 uuuu | uuu0 uuuu | ---- --u- |
| WDT Wake-up from Sleep | PC + 1 | ---0 0uuu | uuuu uuuu | ---- --u- |
| Interrupt Wake-up from Sleep | PC + 1(1) | ---1 0uuu | uuuu uuuu | ---- --u- |
| RESET Instruction Executed | 0 | ---u uuuu | uuuu u0uu | ---- --u- |
| Stack Overflow Reset (STVREN = 1) | 0 | ---u uuuu | 1uuu uuuu | ---- --u- |
| Stack Underflow Reset (STVREN = 1) | 0 | ---u uuuu | u1uu uuuu | ---- --u- |
| Memory Violation Reset | 0 | -uuu uuuu | uuuu uuuu | ---- --0- |

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

**Note:**

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

## 10.11 Power Control (PCONx) Register

The Power Control (PCONx) registers contain flag bits to differentiate between a:

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

Hardware will change the corresponding register bit during the Reset process; if the Reset was not caused by the condition, the bit remains unchanged.

Software should reset the bit to the Inactive state after restart (hardware will not reset the bit).

Software may also set any PCONx bit to the Active state, so that user code may be tested, but no Reset action will be generated.

## 10.12 Register Definitions: Power Control

### 10.12.1 BORCON

**Name:** BORCON
**Address:** 0x811

Brown-out Reset Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SBOREN | | | | | | | BORRDY |
| Access | R/W | | | | | | | R |
| Reset | 1 | | | | | | | q |

**Bit 7 – SBOREN** Software Brown-out Reset Enable
Reset States: POR/BOR = 1
All Other Resets = u

| Value | Condition | Description |
|---|---|---|
| – | If BOREN ≠ 01 | SBOREN is read/write, but has no effect on the BOR. |
| 1 | If BOREN = 01 | BOR Enabled |
| 0 | If BOREN = 01 | BOR Disabled |

**Bit 0 – BORRDY** Brown-out Reset Circuit Ready Status
Reset States: POR/BOR = q
All Other Resets = u

| Value | Description |
|---|---|
| 1 | The Brown-out Reset circuit is active and armed |
| 0 | The Brown-out Reset circuit is disabled or is warming up |

### 10.12.2 PCON0

**Name:** PCON0
**Address:** 0x813

Power Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | STKOVF | STKUNF | | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |
| Access | R/W/HS | R/W/HS | | R/W/HC | R/W/HC | R/W/HC | R/W/HC | R/W/HC |
| Reset | 0 | 0 | | 1 | 1 | 1 | 0 | q |

**Bit 7 – STKOVF** Stack Overflow Flag
Reset States: POR/BOR = 0
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A Stack Overflow occurred (more CALLs than fit on the stack) |
| 0 | A Stack Overflow has not occurred or set to '0' by firmware |

**Bit 6 – STKUNF** Stack Underflow Flag
Reset States: POR/BOR = 0
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A Stack Underflow occurred (more RETURNs than CALLs) |
| 0 | A Stack Underflow has not occurred or set to '0' by firmware |

**Bit 4 – $\overline{\text{RWDT}}$** WDT Reset Flag
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A WDT overflow/time-out Reset has not occurred or set to '1' by firmware |
| 0 | A WDT overflow/time-out Reset has occurred (set to '0' in hardware when a WDT Reset occurs) |

**Bit 3 – $\overline{\text{RMCLR}}$** $\overline{\text{MCLR}}$ Reset Flag
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A $\overline{\text{MCLR}}$ Reset has not occurred or set to '1' by firmware |
| 0 | A $\overline{\text{MCLR}}$ Reset has occurred (set to '0' in hardware when a $\overline{\text{MCLR}}$ Reset occurs) |

**Bit 2 – $\overline{\text{RI}}$** RESET Instruction Flag
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A RESET instruction has not been executed or set to '1' by firmware |
| 0 | A RESET instruction has been executed (set to '0' in hardware upon executing a RESET instruction) |

**Bit 1 – $\overline{\text{POR}}$** Power-on Reset Status
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | No Power-on Reset occurred or set to '1' by firmware |
| 0 | A Power-on Reset occurred (set to '0' in hardware when a Power-on Reset occurs) |

**Bit 0 – $\overline{\text{BOR}}$** Brown-out Reset Status
Reset States: POR/BOR = q

All Other Resets = u

| Value | Description |
|---|---|
| 1 | No Brown-out Reset occurred or set to '1' by firmware |
| 0 | A Brown-out Reset occurred (set to '0' in hardware when a Brown-out Reset occurs) |

### 10.12.3 PCON1

**Name:** PCON1
**Address:** 0x814

Power Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | $\overline{MEMV}$ | |
| Access | | | | | | | R/W/HC | |
| Reset | | | | | | | 1 | |

**Bit 1 – $\overline{MEMV}$** Memory Violation Flag
Reset States: POR/BOR = 1
All Other Resets = u

| Value | Description |
|---|---|
| 1 | No Memory Violation Reset occurred or set to '1' by firmware. |
| 0 | A Memory Violation Reset occurred (set to '0' in hardware when a Memory Violation occurs) |

## 10.13 Register Summary: Power Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x0810 | Reserved | | | | | | | | | |
| 0x0811 | BORCON | 7:0 | SBOREN | | | | | | | BORRDY |
| 0x0812 | Reserved | | | | | | | | | |
| 0x0813 | PCON0 | 7:0 | STKOVF | STKUNF | | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |
| 0x0814 | PCON1 | 7:0 | | | | | | | $\overline{\text{MEMV}}$ | |

# 11. OSC - Oscillator Module

## 11.1 Oscillator Module Overview

The oscillator module contains multiple clock sources and selection features that allow it to be used in a wide range of applications while maximizing performance and minimizing power consumption.

Clock sources can be supplied either internally or externally. External sources include:

- External Clock (EC) oscillators

Internal sources include:

- High-Frequency Internal Oscillator (HFINTOSC)
- Low-Frequency Internal Oscillator (LFINTOSC)
- Analog-to-Digital Converter RC Oscillator (ADCRC)

Special features of the oscillator module include:

- HFINTOSC Frequency Adjustment: Provides the ability to adjust the HFINTOSC frequency.

The Reset Oscillator (RSTOSC) Configuration bits determine the type of oscillator that will be used when the device runs after a Reset, including when the device is first powered up (see Table 11-1).

**Table 11-1. RSTOSC Selection Table**

| RSTOSC | SFR Reset Values | | Clock Source |
|---|---|---|---|
| | COSC | OSCFRQ | |
| 11 | 11 | 100 (16 MHz) | EXTOSC per FEXTOSC |
| 10 | 10 | | HFINTOSC @ 1 MHz |
| 01 | 01 | | LFINTOSC |
| 00 | 00 | | HFINTOSC @ 32 MHz |

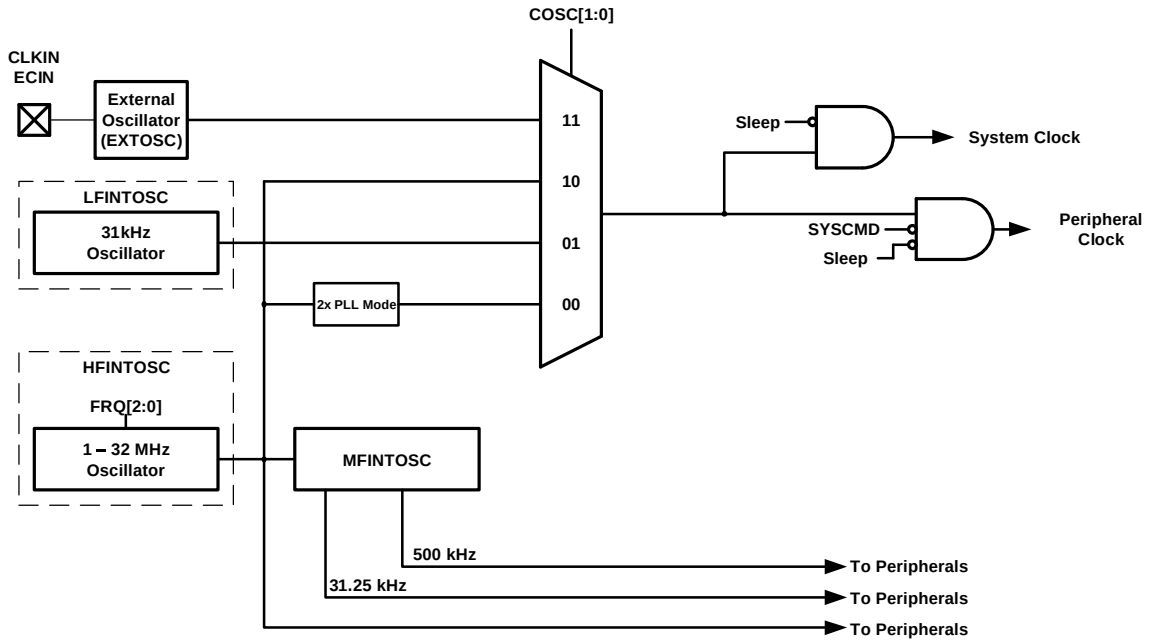
If an external clock source is selected by the RSTOSC bits, the External Oscillator Mode Select (FEXTOSC) Configuration bits must be used to select the external clock mode. These modes include:

- ECL: External Clock Low-Power mode
- ECH: External Clock High-Power mode

The ECH and ECL modes rely on an external logic-level signal as the device clock source. Each mode is optimized for a specific frequency range. The internal oscillator block produces both low-frequency and high-frequency clock signals, designated LFINTOSC and HFINTOSC, respectively. Multiple system operating frequencies may be derived from these clock sources.

Figure 11-1 illustrates a block diagram of the oscillator module.

**Figure 11-1. Clock Source Block Diagram**




## 11.2 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 of external clock sources include:

- Digital oscillator modules

Internal clock sources are contained within the oscillator module. The internal oscillator block features two internal oscillators that are used to generate internal system clock sources. The High-Frequency Internal Oscillator (HFINTOSC) can produce a wide range of frequencies which are determined via the HFINTOSC Frequency Selection (OSCFRQ) register. The Low-Frequency Internal Oscillator (LFINTOSC) generates a fixed nominal 31 kHz clock signal. The internal oscillator block also features an RC oscillator (ADCRC) which is dedicated to the Analog-to-Digital Converter (ADC).

The instruction clock ($F_{OSC}/4$) can be routed to the CLKOUT pin when the pin is not in use. The Clock Out Enable ($\overline{CLKOUTEN}$) Configuration bit controls the functionality of the CLKOUT signal. When $\overline{CLKOUTEN}$ is clear ($\overline{CLKOUTEN}$ = 0), the CLKOUT signal is routed to the CLKOUT pin. When $\overline{CLKOUTEN}$ is set ($\overline{CLKOUTEN}$ = 1), the CLKOUT pin functions as an I/O pin.

### 11.2.1 External Clock Sources

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

- Program the RSTOSC Configuration bits to select an external clock source that will be used as the default system clock upon a device Reset.
- Program the FEXTOSC Configuration bits to select either the ECH (16 MHz and higher) or ECL (below 16 MHz) mode based on the frequency of the external clock.

#### 11.2.1.1 EC Mode

The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in EC mode, an external clock source is connected to the CLKIN input pin. The CLKOUT pin is available as a general purpose I/O pin or as the CLKOUT signal pin.

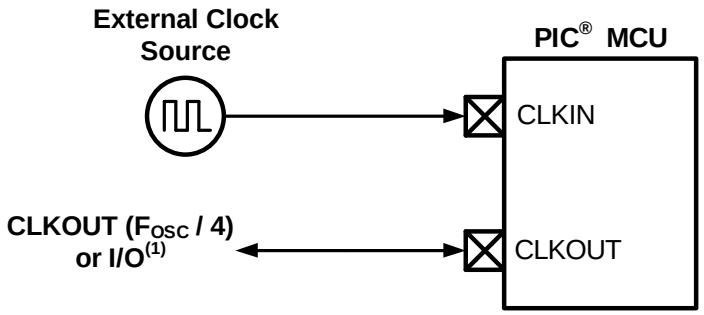
EC mode provides two power mode selections:

- ECH: High-power mode
- ECL: Low-power mode

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 11-2 shows the pin connections for EC mode.

**Figure 11-2. External Clock (EC) Mode Operation**




**Note:**

1. Output depends on the setting of the $\overline{\text{CLKOUTEN}}$ Configuration bit.

### 11.2.2 Internal Clock Sources

The internal oscillator block contains two independent oscillators that can produce two internal system clock sources:

- High-Frequency Internal Oscillator (HFINTOSC)
- Low-Frequency Internal Oscillator (LFINTOSC)

An internal oscillator source can be used as the device system clock by programming the RSTOSC Configuration bits to select one of the INTOSC sources.

In INTOSC mode, the CLKIN and CLKOUT pins are available for use as general purpose I/Os, provided that no external oscillator is connected. The function of the CLKOUT pin is determined by the $\overline{\text{CLKOUTEN}}$ Configuration bit. When $\overline{\text{CLKOUTEN}}$ is set ($\overline{\text{CLKOUTEN}}$ = 1), the pin functions as a general purpose I/O. When $\overline{\text{CLKOUTEN}}$ is clear ($\overline{\text{CLKOUTEN}}$ = 0), the system instruction clock ($F_{OSC}$/4) is available as an output signal on the pin.

#### 11.2.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a factory-calibrated, precision digitally-controlled internal clock source that produces a wide range of stable clock frequencies. The HFINTOSC can be enabled by programming the RSTOSC Configuration bits to select the one of two HFINTOSC options upon device Reset or power-up.

The HFINTOSC frequency is selected via the HFINTOSC Frequency Selection (FRQ) bits. Fine-tuning of the HFINTOSC is done via the HFINTOSC Frequency Tuning (TUN) bits.

##### 11.2.2.1.1 HFINTOSC Frequency Tuning

The HFINTOSC frequency can be fine-tuned via the HFINTOSC Tuning Register (OSCTUNE). The OSCTUNE register provides small adjustments to the HFINTOSC nominal frequency.

The OSCTUNE register contains the HFINTOSC Frequency Tuning (TUN) bits. The TUN bits default to a 6-bit, two's compliment value of 0x00, which indicates that the oscillator is operating at the selected frequency. When a value between 0x01 and 0x1F is written to the TUN bits, the HFINTOSC frequency is increased. When a value between 0x3F and 0x20 is written to the TUN bits, the HFINTOSC frequency is decreased.

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

**Important:** OSCTUNE tuning does not affect the LFINTOSC frequency.

#### 11.2.2.1.2 MFINTOSC

The Medium-Frequency Internal Oscillator (MFINTOSC) generates two constant clock outputs (500 kHz and 31.25 kHz). The MFINTOSC clock signals are created from the HFINTOSC using dynamic divider logic, which provides constant MFINTOSC clock rates regardless of selected HFINTOSC frequency.

The MFINTOSC cannot be used as the system clock, but can be used as a clock source for certain peripherals, such as a Timer.

### 11.2.2.2 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a factory-calibrated 31 kHz internal clock source.

The LFINTOSC can be used as a system clock source, and may be used by certain peripheral modules as a clock source. Additionally, the LFINTOSC provides a time base for the following:

- Power-up Timer (PWRT)
- Watchdog Timer (WDT)

The LFINTOSC is enabled by programming the RSTOSC Configuration bits to select LFINTOSC.

## 11.3 Register Definitions: Oscillator Control

### 11.3.1 OSCCON

**Name:** OSCCON
**Address:** 0x88E

Oscillator Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | COSC[1:0] | | | | | |
| Access | | | R | R | | | | |
| Reset | | | q | q | | | | |

**Bits 5:4 – COSC[1:0]** Current Oscillator Source Select
Indicates the current oscillator source per the RSTOSC Configuration bits

### 11.3.2 OSCSTAT

**Name:** OSCSTAT
**Address:** 0x890

Oscillator Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | HFOR | MFOR | LFOR | | ADOR | SFOR | |
| Access | | R | R | R | | R | R | |
| Reset | | 0 | 0 | 0 | | 0 | 0 | |

**Bit 6 – HFOR** HFINTOSC Ready

| Value | Description |
|---|---|
| 1 | The HFINTOSC is ready for use |
| 0 | The HFINTOSC is not enabled, or it is not ready for use |

**Bit 5 – MFOR** MFINTOSC Ready

| Value | Description |
|---|---|
| 1 | The MFINTOSC is ready for use |
| 0 | The MFINTOSC is not enabled, or it is not ready for use |

**Bit 4 – LFOR** LFINTOSC Ready

| Value | Description |
|---|---|
| 1 | The LFINTOSC is ready for use |
| 0 | The LFINTOSC is not enabled, or is not ready for use |

**Bit 2 – ADOR** ADCRC Oscillator Ready

| Value | Description |
|---|---|
| 1 | The ADCRC oscillator is ready for use |
| 0 | The ADCRC oscillator is not enabled, or is not ready for use |

**Bit 1 – SFOR** Special Frequency Oscillator is Ready

| Value | Description |
|---|---|
| 1 | The SFINTOSC is ready for use |
| 0 | The SFINTOSC is not ready for use |

### 11.3.3 OSCEN

**Name:** OSCEN
**Address:** 0x891

Oscillator Enable Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | HFOEN | MFOEN | LFOEN | | ADOEN | | |
| Access | | R/W | R/W | R/W | | R/W | | |
| Reset | | 0 | 0 | 0 | | 0 | | |

**Bit 6 – HFOEN** HFINTOSC Enable

| Value | Description |
|---|---|
| 1 | HFINTOSC is explicitly enabled, operating as specified by the OSCFRQ register |
| 0 | HFINTOSC can be enabled by a peripheral request |

**Bit 5 – MFOEN** MFINTOSC Enable

| Value | Description |
|---|---|
| 1 | MFINTOSC is explicitly enabled |
| 0 | MFINTOSC can be enabled by a peripheral request |

**Bit 4 – LFOEN** LFINTOSC Enable

| Value | Description |
|---|---|
| 1 | LFINTOSC is explicitly enabled |
| 0 | LFINTOSC can be enabled by a peripheral request |

**Bit 2 – ADOEN** ADCRC Oscillator Enable

| Value | Description |
|---|---|
| 1 | ADCRC is explicitly enabled |
| 0 | ADCRC may be enabled by a peripheral request |

### 11.3.4 OSCFRQ

**Name:** OSCFRQ
**Address:** 0x893

HFINTOSC Frequency Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | FRQ[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – FRQ[2:0]** HFINTOSC Frequency Selection

| FRQ | Nominal Freq (MHz) |
|---|---|
| 111–110 | Reserved |
| 101 | 32 |
| 100 | 16 |
| 011 | 8 |
| 010 | 4 |
| 001 | 2 |
| 000 | 1 |

### 11.3.5 OSCTUNE

**Name:** OSCTUNE
**Address:** 0x892

HFINTOSC Frequency Tuning Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | TUN[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – TUN[5:0]** HFINTOSC Frequency Tuning

| TUN | Condition |
|---|---|
| 01 1111 | Maximum frequency |
| • | • |
| • | • |
| • | • |
| 00 0000 | Center frequency. Oscillator is operating at the selected nominal frequency. (Default value) |
| • | • |
| • | • |
| • | • |
| 10 0000 | Minimum frequency |

## 11.4 Register Summary: Oscillator Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x088D | Reserved | | | | | | | | | |
| 0x088E | OSCCON | 7:0 | | | COSC[1:0] | | | | | |
| 0x088F | Reserved | | | | | | | | | |
| 0x0890 | OSCSTAT | 7:0 | | HFOR | MFOR | LFOR | | ADOR | SFOR | |
| 0x0891 | OSCEN | 7:0 | | HFOEN | MFOEN | LFOEN | | ADOEN | | |
| 0x0892 | OSCTUNE | 7:0 | | | TUN[5:0] | | | | | |
| 0x0893 | OSCFRQ | 7:0 | | | | | | FRQ[2:0] | | |

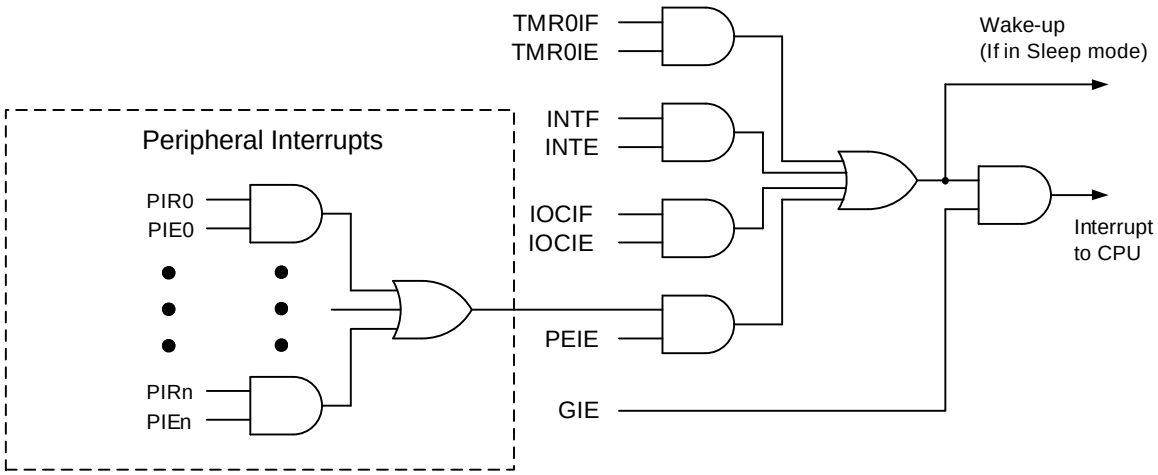
# 12. Interrupts

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.

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

A block diagram of the interrupt logic is shown in Figure 12-1.

**Figure 12-1. Interrupt Logic**




## 12.1 INTCON Register

The Interrupt Control (INTCON) register is readable and writable, and contains the Global Interrupt Enable (GIE), Peripheral Interrupt Enable (PEIE), and External Interrupt Edge Select (INTEDG) bits.

## 12.2 PIE Registers

The Peripheral Interrupt Enable (PIE) registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three PIE registers in the PIC16F152 family.

## 12.3 PIR Registers

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

## 12.4 Operation

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

- GIE bit
- PEIE bit (if the Interrupt Enable bit of the interrupt event is contained in the PIE registers)
- Interrupt Enable bit(s) for the specific interrupt event(s)

The PIR 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 "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 interrupts operation, refer to its peripheral chapter.

**Important:**

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.

## 12.5 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 interrupt is sampled during Q1 of the instruction cycle. The actual interrupt latency then depends on the instruction that is executing at the time the interrupt is detected. See the following figures for more details.

**Figure 12-2. Interrupt Latency**

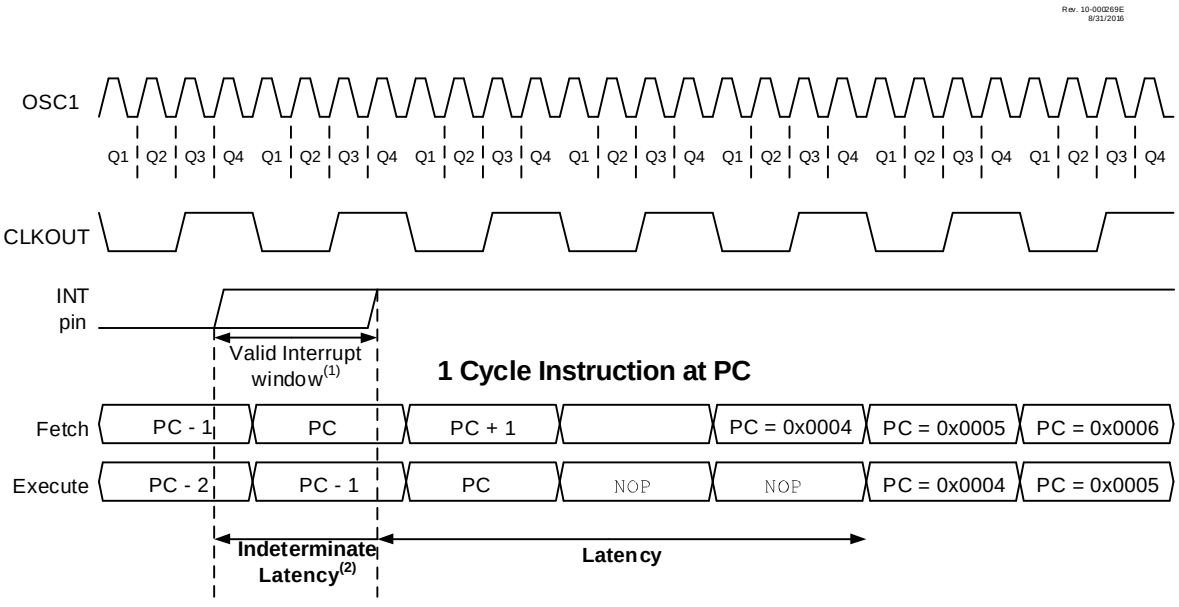
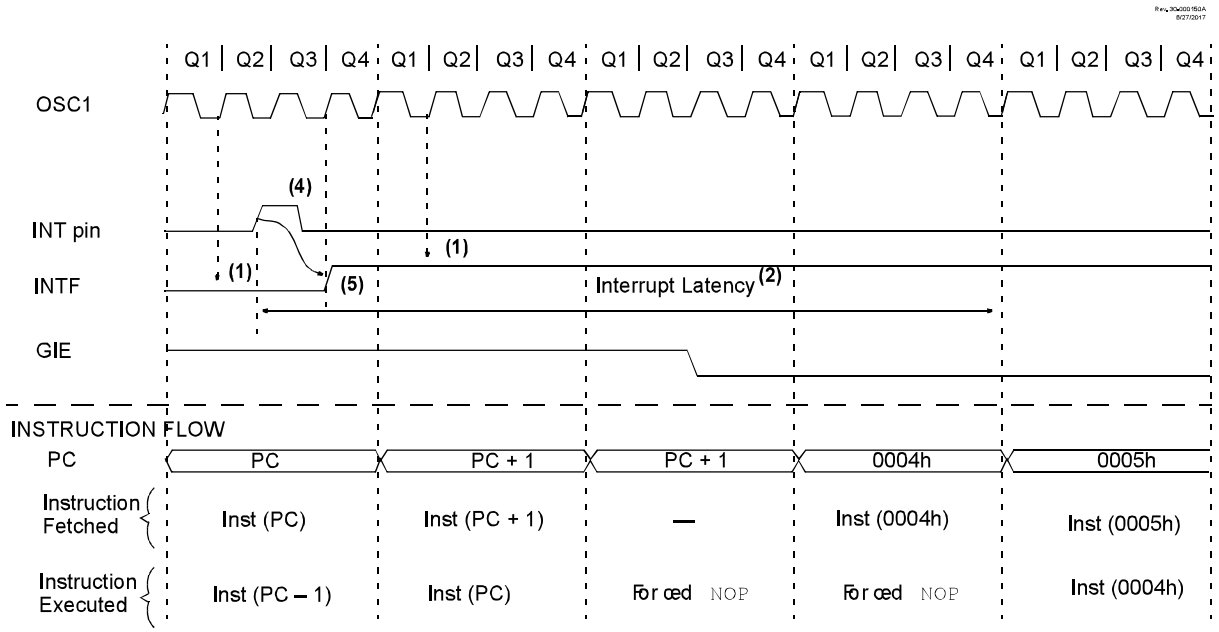


**Figure 12-3. INT Pin Interrupt Timing**




**Notes:**

1. INTF flag is sampled here (every Q1).
2. Asynchronous interrupt latency = 3-5 $T_{CY}$. Synchronous latency = 3-4 $T_{CY}$, where $T_{CY}$ = 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 the **"Electrical Specifications"** chapter.
4. INTF may be set any time during the Q4-Q1 cycles.

## 12.6 Interrupts During Sleep

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.

## 12.7 INT Pin

The INT pin can be used to generate an asynchronous edge-triggered interrupt. This interrupt is enabled by setting the External Interrupt Enable (INTE) bit. The External Interrupt Edge Select (INTEDG) bit 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 External Interrupt Flag (INTF) bit 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.

## 12.8 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:

- WREG 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 63 and are readable and writable. Depending on the user's application, other registers may also need to be saved.

## 12.9 Register Definitions: Interrupt Control

### 12.9.1 INTCON

**Name:** INTCON
**Address:** 0x00B

Interrupt Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | GIE | PEIE | | | | | | INTEDG |
| Access | R/W | R/W | | | | | | R/W |
| Reset | 0 | 0 | | | | | | 1 |

**Bit 7 – GIE** Global Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enables all active interrupts |
| 0 | Disables all interrupts |

**Bit 6 – PEIE** Peripheral Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enables all active peripheral interrupts |
| 0 | Disables all peripheral interrupts |

**Bit 0 – INTEDG** External Interrupt Edge Select

| Value | Description |
|---|---|
| 1 | Interrupt on rising edge of INT pin |
| 0 | Interrupt on falling edge of INT pin |

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

### 12.9.2 PIE0

**Name:** PIE0
**Address:** 0x716

Peripheral Interrupt Enable Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | TMR0IE | IOCIE | | | | INTE |
| Access | | | R/W | R/W | | | | R/W |
| Reset | | | 0 | 0 | | | | 0 |

**Bit 5 – TMR0IE** Timer0 Interrupt Enable

| Value | Description |
|---|---|
| 1 | TMR0 interrupts are enabled |
| 0 | TMR0 interrupts are disabled |

**Bit 4 – IOCIE** Interrupt-on-Change Enable

| Value | Description |
|---|---|
| 1 | IOC interrupts are enabled |
| 0 | IOC interrupts are disabled |

**Bit 0 – INTE** External Interrupt Enable(1)

| Value | Description |
|---|---|
| 1 | External interrupts are enabled |
| 0 | External interrupts are disabled |

**Notes:**

1. The External Interrupt INT pin is selected by INTPPS.
2. Bit PEIE in the INTCON register must be set to enable any peripheral interrupt controlled by the PIE1 and PIE2 registers. Interrupt sources controlled by the PIE0 register do not require the PEIE bit to be set in order to allow interrupt vectoring (when the GIE bit in the INTCON register is set).

### 12.9.3 PIE1

**Name:** PIE1
**Address:** 0x717

Peripheral Interrupt Enable Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CCP1IE | TMR2IE | TMR1IE | RC1IE | TX1IE | BCL1IE | SSP1IE | ADIE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – CCP1IE** CCP1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | CCP1 interrupts are enabled |
| 0 | CCP1 interrupts are disabled |

**Bit 6 – TMR2IE** TMR2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | TMR2 interrupts are enabled |
| 0 | TMR2 interrupts are disabled |

**Bit 5 – TMR1IE** TMR1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | TMR1 interrupts are enabled |
| 0 | TMR1 interrupts are disabled |

**Bit 4 – RC1IE** EUSART1 Receive Interrupt Enable

| Value | Description |
|---|---|
| 1 | EUSART1 receive interrupts are enabled |
| 0 | EUSART1 receive interrupts are disabled |

**Bit 3 – TX1IE** EUSART1 Transmit Interrupt Enable

| Value | Description |
|---|---|
| 1 | EUSART1 transmit interrupts are enabled |
| 0 | EUSART1 transmit interrupts are disabled |

**Bit 2 – BCL1IE** MSSP1 Bus Collision Interrupt Enable

| Value | Description |
|---|---|
| 1 | MSSP1 bus collision interrupts are enabled |
| 0 | MSSP1 bus collision interrupts are disabled |

**Bit 1 – SSP1IE** MSSP1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | MSSP1 interrupts are enabled |
| 0 | MSSP1 interrupts are disabled |

**Bit 0 – ADIE** ADC Interrupt Enable

| Value | Description |
|---|---|
| 1 | ADC interrupts are enabled |
| 0 | ADC interrupts are disabled |

**Note:** Bit PEIE of the INTCON register must be set to enable any peripheral interrupt controlled by registers PIE1 and PIE2.

### 12.9.4 PIE2

**Name:** PIE2
**Address:** 0x718

Peripheral Interrupt Enable Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CCP2IE | NVMIE | TMR1GIE | | | | | |
| Access | R/W | R/W | R/W | | | | | |
| Reset | 0 | 0 | 0 | | | | | |

**Bit 7 – CCP2IE** CCP2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | CCP2 interrupts are enabled |
| 0 | CCP2 interrupts are disabled |

**Bit 6 – NVMIE** NVM Interrupt Enable

| Value | Description |
|---|---|
| 1 | NVM interrupts are enabled |
| 0 | NVM interrupts are disabled |

**Bit 5 – TMR1GIE** TMR1 Gate Interrupt Enable

| Value | Description |
|---|---|
| 1 | TMR1 Gate interrupts are enabled |
| 0 | TMR1 Gate interrupts are disabled |

**Note:** Bit PEIE of the INTCON register must be set to enable any peripheral interrupt controlled by registers PIE1 and PIE2.

### 12.9.5 PIR0

**Name:** PIR0
**Address:** 0x70C

Peripheral Interrupt Request Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | TMR0IF | IOCIF | | | | INTF |
| Access | | | R/W/HS | R | | | | R/W/HS |
| Reset | | | 0 | 0 | | | | 0 |

**Bit 5 – TMR0IF** Timer0 Interrupt Flag

| Value | Description |
|---|---|
| 1 | TMR0 register has overflowed (must be cleared by software) |
| 0 | TMR0 register has not overflowed |

**Bit 4 – IOCIF** Interrupt-on-Change Flag(2)

| Value | Description |
|---|---|
| 1 | One or more of the IOCAF-IOCCF register bits are currently set, indicating an enabled edge was detected by the IOC module. |
| 0 | None of the IOCAF-IOCCF register bits are currently set |

**Bit 0 – INTF** External Interrupt Flag(1)

| Value | Description |
|---|---|
| 1 | External Interrupt has occurred |
| 0 | External Interrupt has not occurred |

**Notes:**

1. The External Interrupt INT pin is selected by INTPPS.
2. The IOCIF bit is the logical OR of all the IOCAF-IOCCF flags. Therefore, to clear the IOCIF flag, application firmware must clear all of the lower level IOCAF-IOCCF register bits.
3. Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Enable (GIE) bit. User software should ensure the appropriate interrupt flag bits are cleared before enabling an interrupt.

### 12.9.6 PIR1

**Name:** PIR1
**Address:** 0x70D

Peripheral Interrupt Request Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CCP1IF | TMR2IF | TMR1IF | RC1IF | TX1IF | BCL1IF | SSP1IF | ADIF |
| Access | R/W/HS | R/W/HS | R/W/HS | R | R | R/W/HS | R/W/HS | R/W/HS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – CCP1IF** CCP1 Interrupt Flag

| Value | CCP Mode | | |
|---|---|---|---|
| | Capture | Compare | PWM |
| 1 | Capture occurred (must be cleared in software) | Compare match occurred (must be cleared in software) | Output trailing edge occurred (must be cleared in software) |
| 0 | Capture did not occur | Compare match did not occur | Output trailing edge did not occur |

**Bit 6 – TMR2IF** TMR2 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared in software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – TMR1IF** TMR1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared in software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – RC1IF** EUSART1 Receive Interrupt Flag(1)

| Value | Description |
|---|---|
| 1 | The EUSART1 receive buffer (RC1REG) is not empty (contains at least one byte) |
| 0 | The EUSART1 receive buffer is empty |

**Bit 3 – TX1IF** EUSART1 Transmit Interrupt Flag(2)

| Value | Description |
|---|---|
| 1 | The EUSART1 transmit buffer (TX1REG) is empty |
| 0 | The EUSART1 transmit buffer is not empty |

**Bit 2 – BCL1IF** MSSP1 Bus Collision Interrupt Flag

| Value | Description |
|---|---|
| 1 | A bus collision was detected (must be cleared in software) |
| 0 | No bus collision was detected |

**Bit 1 – SSP1IF** MSSP1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared in software) |
| 0 | Interrupt event has not occurred |

**Bit 0 – ADIF** ADC Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared in software) |
| 0 | Interrupt event has not occurred |

**Notes:**

1. RC1IF is read-only. User software must read RC1REG to clear RC1IF.
2. TX1IF is read-only. User software must load TX1REG to clear TX1IF. TX1IF does not indicate a completed transmission (use TMRT for this purpose instead).
3. Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Enable (GIE) bit. User software should ensure the appropriate interrupt flag bits are cleared before enabling an interrupt.

### 12.9.7 PIR2

**Name:** PIR2
**Address:** 0x70E

Peripheral Interrupt Request Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CCP2IF | NVMIF | TMR1GIF | | | | | |
| Access | R/W/HS | R/W/HS | R/W/HS | | | | | |
| Reset | 0 | 0 | 0 | | | | | |

**Bit 7 – CCP2IF** CCP2 Interrupt Flag

| Value | CCP Mode | | |
|---|---|---|---|
| | Capture | Compare | PWM |
| 1 | Capture occurred (must be cleared in software) | Compare match occurred (must be cleared in software) | Output trailing edge occurred (must be cleared in software) |
| 0 | Capture did not occur | Compare match did not occur | Output trailing edge did not occur |

**Bit 6 – NVMIF** Nonvolatile Memory (NVM) Interrupt Flag

| Value | Description |
|---|---|
| 1 | The requested NVM operation has completed (must be cleared in software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – TMR1GIF** TMR1 Gate Interrupt Flag

| Value | Description |
|---|---|
| 1 | The TMR1 Gate has gone inactive (must be cleared in software) |
| 0 | TMR1 Gate is active |

**Note:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Enable (GIE) bit. User software should ensure the appropriate interrupt flag bits are cleared before enabling an interrupt.

## 12.10 Register Summary: Interrupt Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x070B | Reserved | | | | | | | | | |
| 0x070C | PIR0 | 7:0 | | | TMR0IF | IOCIF | | | | INTF |
| 0x070D | PIR1 | 7:0 | CCP1IF | TMR2IF | TMR1IF | RC1IF | TX1IF | BCL1IF | SSP1IF | ADIF |
| 0x070E | PIR2 | 7:0 | CCP2IF | NVMIF | TMR1GIF | | | | | |
| 0x070F<br>...<br>0x0715 | Reserved | | | | | | | | | |
| 0x0716 | PIE0 | 7:0 | | | TMR0IE | IOCIE | | | | INTE |
| 0x0717 | PIE1 | 7:0 | CCP1IE | TMR2IE | TMR1IE | RC1IE | TX1IE | BCL1IE | SSP1IE | ADIE |
| 0x0718 | PIE2 | 7:0 | CCP2IE | NVMIE | TMR1GIE | | | | | |

# 13. Sleep Mode

## 13.1 Sleep Mode Operation

Sleep mode is entered by executing the SLEEP instruction.

Upon entering Sleep mode, the following conditions exist:

1. Resets other than WDT are not affected by Sleep mode; WDT will be cleared but keeps running if enabled for operation during Sleep.
2. The $\overline{PD}$ bit is cleared.
3. The $\overline{TO}$ bit is set.
4. The CPU and the System clocks are disabled.
5. LFINTOSC and/or HFINTOSC will remain enabled if any peripheral has requested them as a clock source or if the HFOEN, MFOEN or LFOEN bits are set.
6. ADC is unaffected if the ADCRC oscillator is selected. When the ADC clock is something other than ADCRC, a SLEEP instruction causes the present conversion to be aborted and the ADC module is turned off, although the ADON bit remains active.
7. I/O ports maintain the status they had before SLEEP was executed (driving high, low, or high-impedance) only if no peripheral connected to the I/O port is active.

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 any oscillator

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

### 13.1.1 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 the **"Determining the Cause of a Reset"** section in the **"Resets"** chapter.

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.

### 13.1.2 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 will not be set
  - $\overline{PD}$ bit 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 will be set
  - $\overline{PD}$ bit 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.

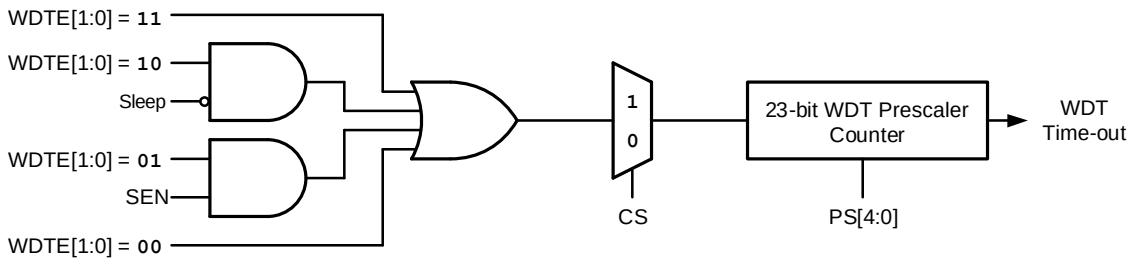
# 14. WDT - Watchdog Timer

The Watchdog Timer (WDT) is a system timer that generates a Reset event if the firmware does not issue a CLRWDT instruction within the time-out period. The Watchdog Timer is typically used to reset the processor in the event of a software malfunction, but can also be used to wake the device when in Sleep mode.

The WDT has the following features:

- Selectable clock sources
- 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 from 1 ms to 256 seconds (nominal)
- Multiple Reset conditions
- Operation during Sleep

**Figure 14-1. Watchdog Timer (WDT) Block Diagram**




## 14.1 Selectable Clock Sources

The WDT can derive its time base from either the 31.25 kHz MFINTOSC or the 31 kHz LFINTOSC as selected by the WDT Clock Source Select (CS) bit.

**Important:** Time intervals detailed in this chapter are based on a minimum nominal interval of 1 ms generated from the LFINTOSC clock source.

## 14.2 WDT Operating Modes

The WDT module has four operating modes controlled by the Watchdog Timer Enable (WDTE) bits. See Table 14-1.

**Table 14-1. WDT Operating Modes**

| WDTE[1:0] | SEN | Device Mode | WDT Mode |
|---|---|---|---|
| 11 | x | X | Active |

.....continued

| WDTE[1:0] | SEN | Device Mode | WDT Mode |
|---|---|---|---|
| 10 | x | Awake | Active |
| | | Sleep | Disabled |
| 01 | 1 | X | Active |
| | 0 | X | Disabled |
| 00 | x | X | Disabled |

### 14.2.1 WDT is Always On

When the WDTE bits are set to '11', the WDT is always on. WDT protection is active during Sleep mode.

### 14.2.2 WDT is Off During Sleep

When the WDTE bits are set to '10', the WDT is on except during Sleep mode. During Sleep mode, WDT protection is disabled.

### 14.2.3 WDT Controlled By Software

When the WDTE bits are set to '01', the WDT is controlled by the Software Watchdog Timer Enable (SEN) bit. When SEN is set (SEN = 1), WDT protection is active. When SEN is clear (SEN = 0), WDT protection is disabled.

### 14.2.4 WDT is Off

When the WDTE bits are set to '00', the WDT is disabled. In this mode, the SEN bit is ignored.

## 14.3 WDT Time-out Period

The Watchdog Timer Prescale Select (PS) bits set the time-out period from 1 ms to 256 seconds (nominal). After a Reset, the default time-out period is two seconds.

## 14.4 Clearing the WDT

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

- Any Reset
- Valid CLRWDT instruction is executed
- Device enters Sleep
- Devices wakes up from Sleep
- Any write to the WDTCON register

## 14.5 WDT Operation During Sleep

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

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 Time-out ($\overline{TO}$) and Power-down ($\overline{PD}$) bits are cleared to indicate the event. Additionally, the Watchdog Timer Reset Flag ($\overline{RWDT}$) bit is cleared, indicating a WDT Reset event occurred.

## 14.6 Register Definitions: WDT Control

### 14.6.1 WDTCON

**Name:** WDTCON
**Address:** 0x80C

Watchdog Timer Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CS | | PS[4:0] | | | | | SEN |
| Access | R/W | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – CS** Watchdog Timer Clock Source Selection

| Value | Description |
|---|---|
| 1 | MFINTOSC (31.25 kHz) |
| 0 | LFINTOSC (31 kHz) |

**Bits 5:1 – PS[4:0]** Watchdog Timer Prescale Selection(1)

| Value | Description |
|---|---|
| 11111 – 10011 | Reserved. Results in minimum interval (1:32) |
| 10010 | 1:8388608 (Interval 256s nominal) |
| 10001 | 1:4194304 (Interval 128s nominal) |
| 10000 | 1:2097152 (Interval 64s nominal) |
| 01111 | 1:1048576 (Interval 32s nominal) |
| 01110 | 1:524288 (Interval 16s nominal) |
| 01101 | 1:262144 (Interval 8s nominal) |
| 01100 | 1:131072 (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 – SEN** Software WDT Enable/Disable

| Value | Condition | Description |
|---|---|---|
| x | If WDTE[1:0] ≠ 01 | This bit is ignored |
| 1 | If WDTE[1:0] = 01 | WDT is enabled |
| 0 | If WDTE[1:0] = 01 | WDT is disabled |

**Note:**

1. Times are approximate and based on the 31 kHz LFINTOSC clock source.

## 14.7 Register Summary - WDT Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x080B | Reserved | | | | | | | | | |
| 0x080C | WDTCON | 7:0 | CS | | PS[4:0] | | | | | SEN |

# 15. NVM - Nonvolatile Memory Control

Nonvolatile Memory (NVM) consists of the Program Flash Memory (PFM).

NVM is accessible by using both the FSR and INDF registers, or through the NVMREG register interface.

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.

NVM can be protected in two ways, by either code protection or write protection.

Code protection ($\overline{CP}$ bit in the Configuration Words) disables access, both reading and writing, to the PFM 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 nonvolatile memory, Configuration bits and User IDs.

Write protection prohibits self-write and erase to a portion or all of the NVM, as defined by the $\overline{WRTSAF}$, $\overline{WRTC}$, $\overline{WRTB}$ and $\overline{WRTAPP}$ bits of the Configuration Words. Write protection does not affect a device programmer's ability to read, write or erase the device.

## 15.1 Program Flash Memory

Program Flash memory consists of an array of 14-bit words as user memory, with additional words for User ID information, Configuration words and interrupt vectors. Program memory provides storage locations for:

- User program instructions
- User defined data

Program memory data can be read and/or written to through:

- CPU instruction fetch (read-only)
- FSR/INDF indirect access (read-only)
- NVMREG access
- In-Circuit Serial Programming™ (ICSP™)

Read operations return a single word of memory. When write and erase operations are done on a row basis, the row size is defined. Program memory will erase to a logic '1' and program to a logic '0'.

It is important to understand the program memory structure for erase and programming operations. Program memory is arranged in rows. A row consists of 32 14-bit program memory words. A row is the minimum size that can be erased by user software.

All or a portion of a row can be programmed. Data to be written into the program memory row is written to 14-bit wide data write latches. These latches are not directly accessible, but may be loaded via sequential writes to the NVMDATH:NVMDATL register pair.

**Important:** To modify only a portion of a previously programmed row, the contents of the entire row must be read. Then, the new data and retained data can be written into the write latches to reprogram the row of 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.

### 15.1.1 Program Memory Voltages

The program memory is readable and writable during normal operation over the full $V_{DD}$ range.

#### 15.1.1.1 Programming Externally

The program memory cell and control logic support write and Bulk Erase operations down to the minimum device operating voltage. Special BOR operation is enabled during Bulk Erase.

#### 15.1.1.2 Self-Programming

The program memory cell and control logic will support write and row erase operations across the entire $V_{DD}$ range. Bulk Erase is not available when self-programming.

## 15.2 FSR and INDF Access

The FSR and INDF registers allow indirect access to the Program Flash Memory.

### 15.2.1 FSR Read

With the intended address loaded into an FSR register a `MOVIW` instruction or read of INDF will read data from the program memory.

Reading from NVM requires one instruction cycle. The CPU operation is suspended during the read, and resumes immediately after. Read operations return a single byte of memory.

### 15.2.2 FSR Write

Writing/erasing the NVM through the FSR registers (ex. `MOVWI` instruction) is not supported in the PIC16F152 devices.

## 15.3 NVMREG Access

The NVMREG interface allows read/write access to all the locations accessible by FSRs, read/write access to the User ID locations, and read-only access to the device identification, revision, and configuration data.

Writing or erasing of NVM via the NVMREG interface is prevented when the device is write-protected.

### 15.3.1 NVMREG Read Operation

To read a NVM location using the NVMREG interface, the user must:

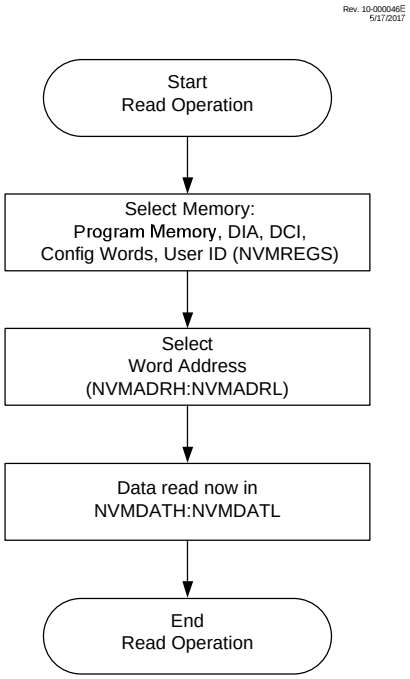
1. Clear the NVMREGS bit if the user intends to access program memory locations, or set NMVREGS if the user intends to access User ID or configuration locations.
2. Write the desired address into the NVMADRH:NVMADRL register pair.
3. Set the RD bit to initiate the read.

Once the read control bit is set, the CPU operation is suspended during the read, and resumes immediately after. The data is available in the very next cycle, in the NVMDATH:NVMDATL register pair; therefore, it can be read as two bytes in the following instructions.

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

Upon completion, the RD bit is cleared by hardware.

**Figure 15-1. Program Flash Memory Read Sequence**




**Example 15-1. 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    NVMADRL             ; Select Bank for NVMCON registers
   MOVLW      PROG_ADDR_LO
   MOVWF      NVMADRL             ; Store LSB of address
   MOVLW      PROG_ADDR_HI
   MOVWF      NVMADRH             ; Store MSB of address

   BCF        NVMCON1,NVMREGS     ; Do not select Configuration Space
   BSF        NVMCON1,RD          ; Initiate read

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

### 15.3.2 NVM Unlock Sequence

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

- PFM Row Erase
- Write of PFM write latches to PFM memory
- Write of PFM write latches to User IDs
- Write to Configuration Words

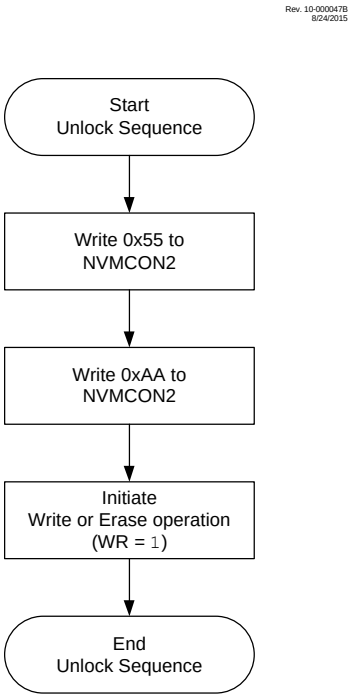
The unlock sequence consists of the following steps and must be completed in order:

- Write 55h to NVMCON2
- Write AAh to NMVCON2
- Set the WR bit

Once the WR bit is set, the processor will stall internal operations until the operation is complete and then resume 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 15-2. NVM Unlock Sequence**




**Example 15-2. NVM Unlock Sequence**

```
BCF         INTCON,GIE      ; Recommended so sequence is not interrupted
BANKSEL     NVMCON1         ; Bank to NVMCON1 register
BSF         NVMCON1,WREN    ; Enable write/erase
MOVLW       55h             ; Load 55h
MOVWF       NVMCON2         ; Step 1: Load 55h into NVMCON2
MOVLW       AAh             ; Step 2: Load W with AAh
MOVWF       NVMCON2         ; Step 3: Load AAh into NVMCON2
BSF         NVMCON1,WR      ; Step 4: Set WR bit to begin write/erase
BSF         INTCON,GIE      ; Re-enable interrupts
```

**Notes:**

1. Sequence begins when NVMCON2 is written; the following four steps must occur in the cycle-accurate order shown. If the timing of the four steps is corrupted by an interrupt or a debugger Halt, the action will not take place.
2. Opcodes shown are illustrative; any instruction that has the indicated effect may be used.

### 15.3.3 NVMREG Erase of Program Memory

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 to program memory. To erase a program memory row:

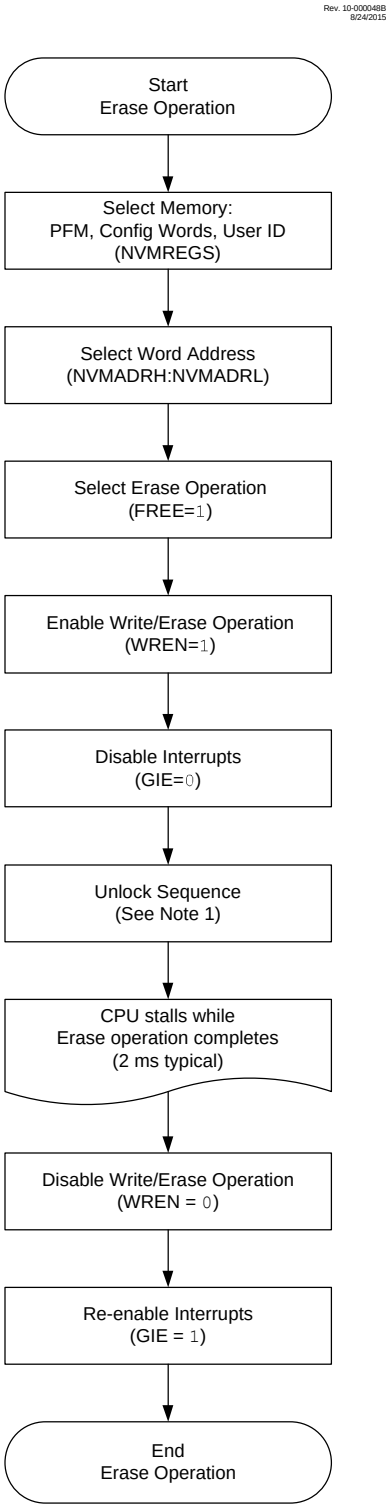
1. Clear the NVMREGS bit to erase program memory locations, or set the NMVREGS bit to erase User ID locations.
2. Write the desired address into the NVMADRH:NVMADRL register pair.
3. Set the FREE and WREN bits.
4. Perform the unlock sequence as described in the "NVM Unlock Sequence" section.

If the program memory address is write-protected, the WR bit will be cleared and the erase operation will not take place.

While erasing program memory, CPU operation is suspended, and resumes when the operation is complete. Upon completion, the NVMIF bit is set, and an interrupt will occur if the NVMIE bit is also set.

Write latch data is not affected by erase operations, and WREN will remain unchanged.

**Figure 15-3. NVM Erase Sequence**




**Note:**

1. See Figure 15-2.

**Example 15-3. Erasing One Row of Program Flash Memory**

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

   BANKSEL    NVMADRL
   MOVF       ADDRL,W
   MOVWF      NVMADRL            ; Load lower 8 bits of erase address boundary
   MOVF       ADDRH,W
   MOVWF      NVMADRH            ; Load upper 6 bits of erase address boundary
   BCF        NVMCON1,NVMREGS    ; Choose PFM memory area
   BSF        NVMCON1,FREE       ; Specify an erase operation
   BSF        NVMCON1,WREN       ; Enable writes
   BCF        INTCON,GIE         ; Disable interrupts during unlock sequence

    ; ----------------------REQUIRED UNLOCK SEQUENCE:---------------------

   MOVLW      0x55               ; Load 0x55 to get ready for unlock sequence
   MOVWF      NVMCON2            ; First step is to load 0x55 into NVMCON2
   MOVLW      0xAA               ; Second step is to load 0xAA into W
   MOVWF      NVMCON2            ; Third step is to load 0xAA into NVMCON2
   BSF        NVMCON1,WR         ; Final step is to set WR bit

   ; ---------------------------------------------------------------------

   BSF        INTCON,GIE         ; Re-enable interrupts, erase is complete
   BCF        NVMCON1,WREN       ; Disable writes
```

**Table 15-1. NVM Organization and Access Information**

| Master Values | | | NVMREG Access | | | FSR Access | |
|---|---|---|---|---|---|---|---|
| Memory Function | Memory Type | Program Counter (PC), ICSP™ Address | NVMREGS bit (NVMCON1) | NVMADR[14:0] | Allowed Operations | FSR Address | FSR Programming Access |
| Reset Vector | Program Flash Memory | 0x0000 | 0 | 0x0000 | Read/Write | 0x8000 | Read-Only |
| User Memory | | 0x0001 | 0 | 0x0001 | | 0x8001 | |
| | | 0x0003 | | 0x0003 | | 0x8003 | |
| INT Vector | | 0x0004 | 0 | 0x0004 | | 0x8004 | |
| User Memory | | 0x0005 | 0 | 0x0005 | | 0x8005 | |
| | | 0x3FFF(1) | | 0x3FFF(1) | | 0xFFFF | |
| User ID | Program Flash Memory | 0x8000 | 1 | 0x0000 | Read/Write | No Access | |
| | | 0x8003 | | 0x0003 | | | |
| Reserved | — | — | — | 0x0004 | — | | |
| Revision ID | Hard Coded in Program Flash Memory | 0x8005 | 1 | 0x0005 | Read | | |
| Device ID | | 0x8006 | 1 | 0x0006 | | | |
| CONFIG1 | Program Flash Memory | 0x8007 | 1 | 0x0007 | Read/Write | | |
| CONFIG2 | | 0x8008 | 1 | 0x0008 | | | |
| CONFIG3 | | 0x8009 | 1 | 0x0009 | | | |
| CONFIG4 | | 0x800A | 1 | 0x000A | | | |
| CONFIG5 | | 0x800B | 1 | 0x000B | | | |
| DIA and DCI | Hard Coded in Program Flash Memory | 0x8100 | 1 | 0x0100 | Read | | |
| | | 0x82FF | 1 | 0x02FF | | | |
| Reserved | — | 0xF000 | — | 0x7000 | — | 0x7000 | — |
| | | 0xF0FF | | 0x7FFF | | 0x7FFF | |

**Note:**

1. The maximum Program Flash Memory address for the 0x3FFF family is 0x07FF.

### 15.3.4 NVMREG Write to Program Memory

Program memory is programmed using the following steps:

1. Load the address of the row to be programmed into NVMADRH:NVMADRL.
2. Load each write latch with data via the NMVDATH:NVMDATL registers.
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 15-4 for more details.

The write latches are aligned to the Flash row address boundary defined by the upper ten bits of NVMADRH:NVMADRL, (NVMADRH[6:0]:NVMADRL[7:5]) with the lower five bits of NVMADRL, (NVMADRL[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 NVMDATH:NVMDATL 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.



**Important:** 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.
2. Clear the NVMREGS bit.
3. Set the LWLO bit. When the LWLO bit is set (LWLO = 1), the write sequence will only load the write latches and will not initiate the write to Program Flash Memory.
4. Load the NVMADRH:NVMADRL register pair with the address of the location to be written.
5. Load the NVMDATH:NVMDATL register pair with the program memory data to be written.
6. Execute the unlock sequence. The write latch is now loaded.
7. Increment the NVMADRH:NVMADRL 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. When the LWLO bit is clear (LWLO = 0), the write sequence will initiate the write to Program Flash Memory.
10. Load the NVMDATH:NVMDATL register pair with the program memory data to be written.
11. Execute the unlock sequence. The entire program memory latch content is now written to Flash program memory.



**Important:** 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 15-4. The initial address is loaded into the NVMADRH:NVMADRL register pair; the data is loaded using indirect addressing.

**Figure 15-4. NVMREG Writes to Program Flash Memory with 32 Write Latches**

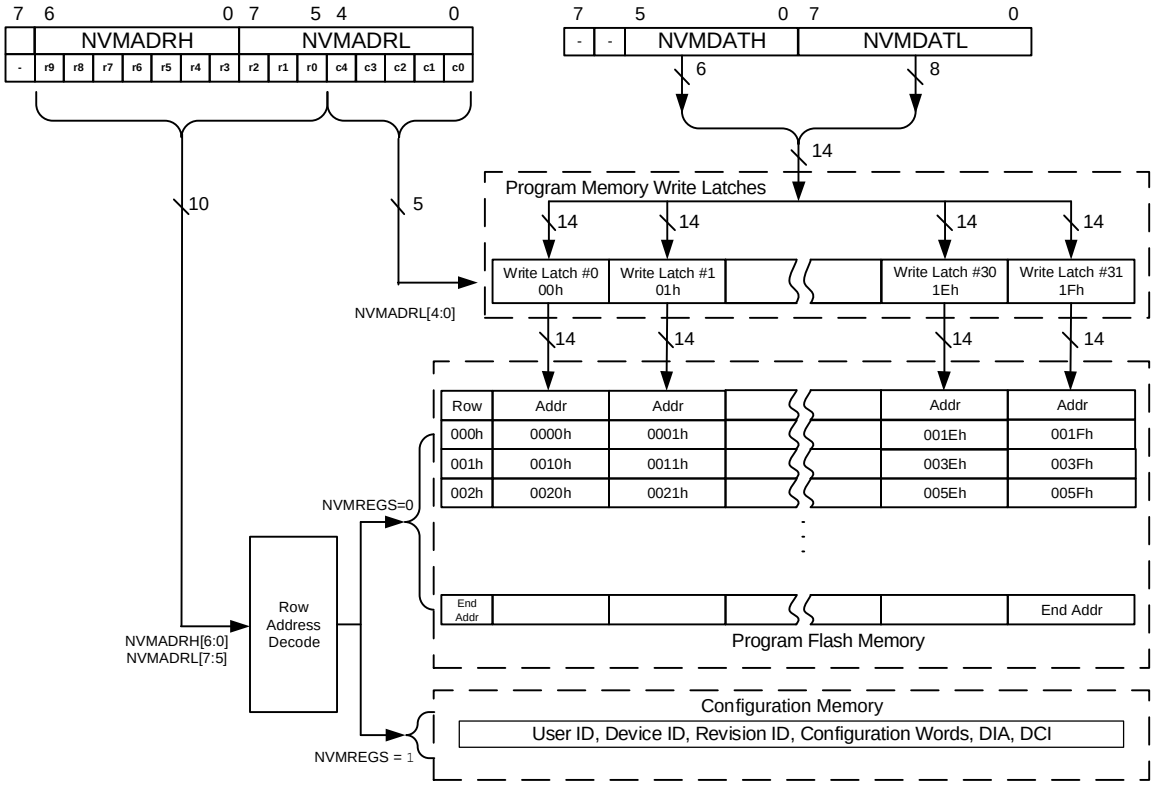
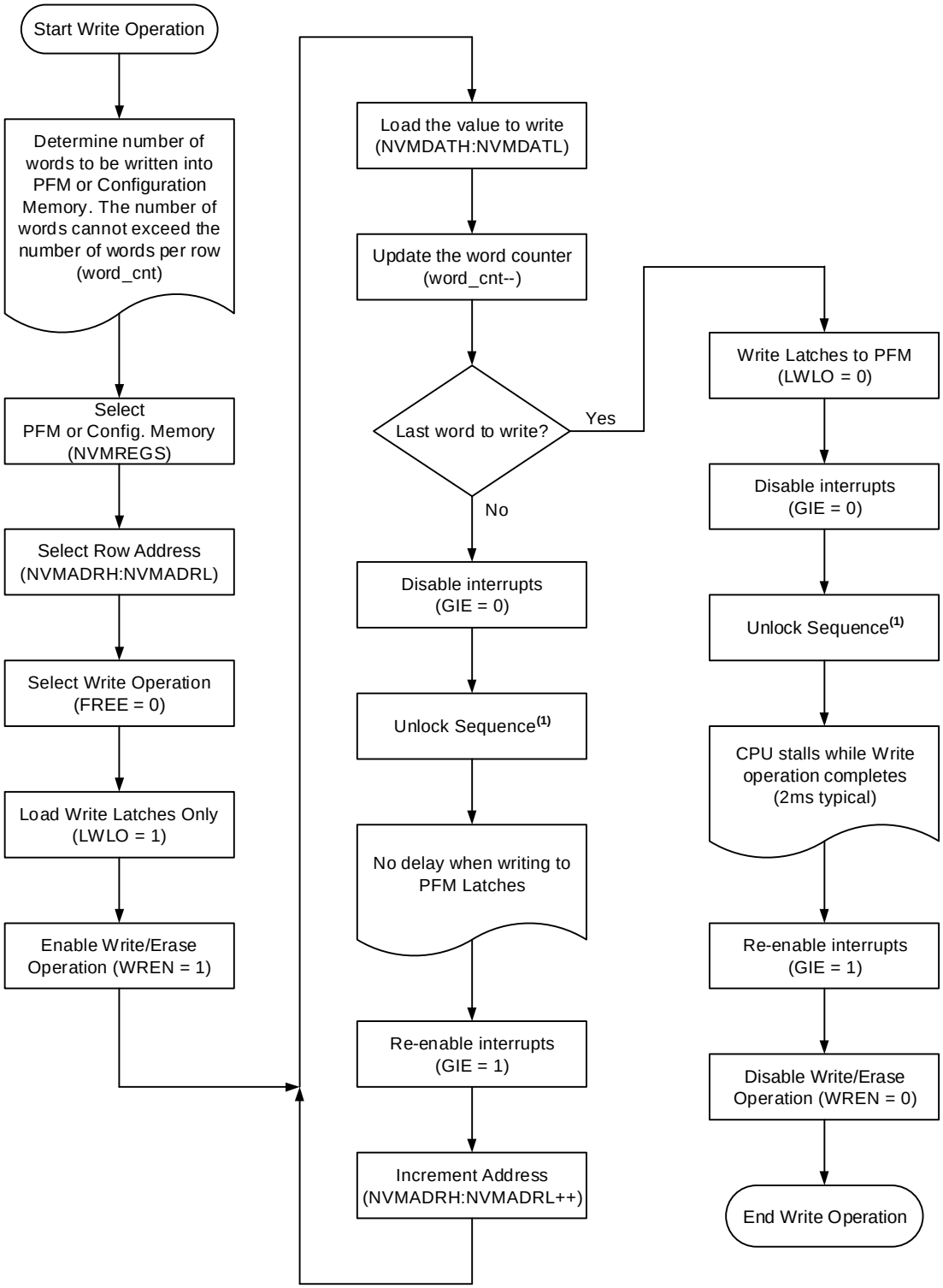


**Figure 15-5. Program Flash Memory Write Sequence**



**Note:**

1. See Figure 15-2.

**Example 15-4. Writing to Program Flash Memory**

```
; 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 is made up of two adjacent bytes in DATA_ADDR,
   ; stored in little endian format
; 3. A valid starting address (LSb's = 00000) is loaded in ADDRH:ADDRL
; 4. ADDRH and ADDRL are located in common RAM (locations 0x70 - 0x7F)
; 5. NVM interrupts are not taken into account

   BANKSEL    NVMADRH
   MOVF       ADDRH,W
   MOVWF      NVMADRH          ; Load initial address
   MOVF       ADDRL,W
   MOVWF      NVMADRL
   MOVLW      LOW DATA_ADDR    ; Load initial data address
   MOVWF      FSR0L
   MOVLW      HIGH DATA_ADDR
   MOVWF      FSR0H
   BCF        NVMCON1,NVMREGS  ; Set Program Flash Memory as write location
   BSF        NVMCON1,WREN     ; Enable writes
   BSF        NVMCON1,LWLO     ; Load only write latches

LOOP
   MOVIW      FSR0++
   MOVWF      NVMDATL          ; Load first data byte
   MOVIW      FSR0++
   MOVWF      NVMDATH          ; Load second data byte
   MOVF       NVMADRL,W
   XORLW      0x1F             ; Check if lower bits of address are 00000
   ANDLW      0x1F             ; and if on last of 32 addresses
   BTFSC      STATUS,Z         ; Last of 32 words?
   GOTO       START_WRITE      ; If so, go write latches into memory
   CALL       UNLOCK_SEQ       ; If not, go load latch
   INCF       NVMADRL,F        ; Increment address
   GOTO       LOOP

START_WRITE
   BCF        NVMCON1,LWLO     ; Latch writes complete, now write memory
   CALL       UNLOCK_SEQ       ; Perform required unlock sequence
   BCF        NVMCON1,WREN     ; Disable writes

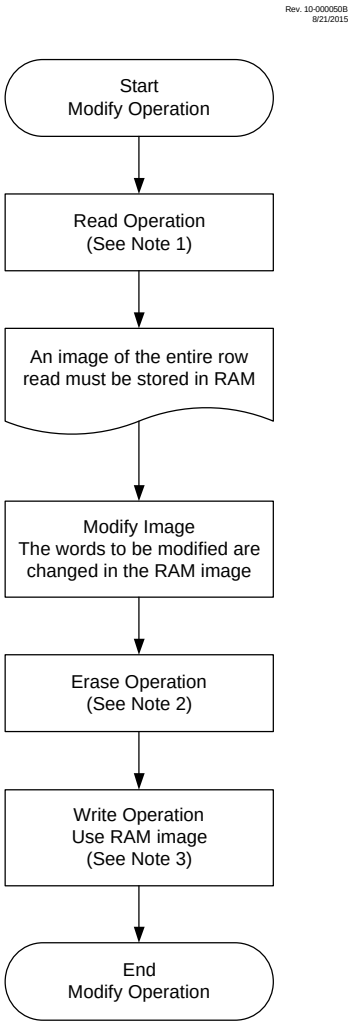
UNLOCK_SEQ
   MOVLW      55h
   BCF        INTCON,GIE       ; Disable interrupts
   MOVWF      NVMCON2          ; Begin unlock sequence
   MOVLW      AAh
   MOVWF      NVMCON2
   BSF        NVMCON1,WR
   BSF        INTCON,GIE       ; Unlock sequence complete, re-enable interrupts
   return
```

### 15.3.5 Modifying Flash Program Memory

When modifying existing data in a program memory, data within the memory row must 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 15-6. Program Flash Memory Modify Sequence**




**Notes:**

1. See Figure 15-1.
2. See Figure 15-3.
3. See Figure 15-5.

### 15.3.6 NVMREG Access to DIA, DCI, User ID, Device ID, Revision ID and Configuration Words

NVMREGS can be used to access the following memory regions:

- Device Information Area (DIA)
- Device Configuration Information (DCI)
- User ID region
- Device ID and Revision ID
- Configuration Words

The value of NVMREGS is set to '1' to access these regions. The memory regions listed above would be pointed to by PC[15] = 1, but not all addresses reference valid data. Different access may exist for reads and writes. Refer to the table below. When read access is initiated on an address outside the parameters listed in the following table, the NVMDATH: NVMDATL register pair is cleared, reading back '0's.

**Table 15-2. NVMREG Access to DIA, DCI, User ID, Device ID, Revision ID and Configuration Words (NVMREGS = 1)**

| Address | Function | Read Access | Write Access |
|---|---|---|---|
| 0x8000 - 0x8003 | User IDs | Yes | Yes |
| 0x8005 - 0x8006 | Device ID/Revision ID | Yes | No |
| 0x8007 - 0x800B | Configuration Words 1-5 | Yes | Yes |
| 0x8100 - 0x82FF | DIA and DCI | Yes | No |

**Example 15-5. Device ID Access**

```
; This write routine assumes the following:
 ; 1. A full row 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 LSb's = 00000) is loaded in ADDRH:ADDRL
 ; 4. ADDRH and ADDRL are located in common RAM (locations 0x70 - 0x7F)
 ; 5. NVM interrupts are not taken into account

   BANKSEL    NVMADRH
   MOVF       ADDRH,W
   MOVWF      NVMADRH             ; Load initial address
   MOVF       ADDRL,W
   MOVWF      NVMADRL
   MOVLW      LOW DATA_ADDR       ; Load initial data address
   MOVWF      FSR0L
   MOVLW      HIGH DATA_ADDR
   MOVWF      FSR0H
   BCF        NVMCON1,NVMREGS     ; Set PFM as write location
   BSF        NVMCON1,WREN        ; Enable writes
   BSF        NVMCON1,LWLO        ; Load only write latches

LOOP
   MOVIW      FSR0++
   MOVWF      NVMDATL             ; Load first data byte
   MOVIW      FSR0++
   MOVWF      NVMDATH             ; Load second data byte
   CALL       UNLOCK_SEQ          ; If not, go load latch
   INCF       NVMADRL,F           ; Increment address
   MOVF       NVMADRL,W
   XORLW      0x1F                ; Check if lower bits of address are 00000
   ANDLW      0x1F                ; and if on last of 32 addresses
   BTFSC      STATUS,Z            ; Last of 32 words?
   GOTO       START_WRITE         ; If so, go write latches into memory
   GOTO       LOOP

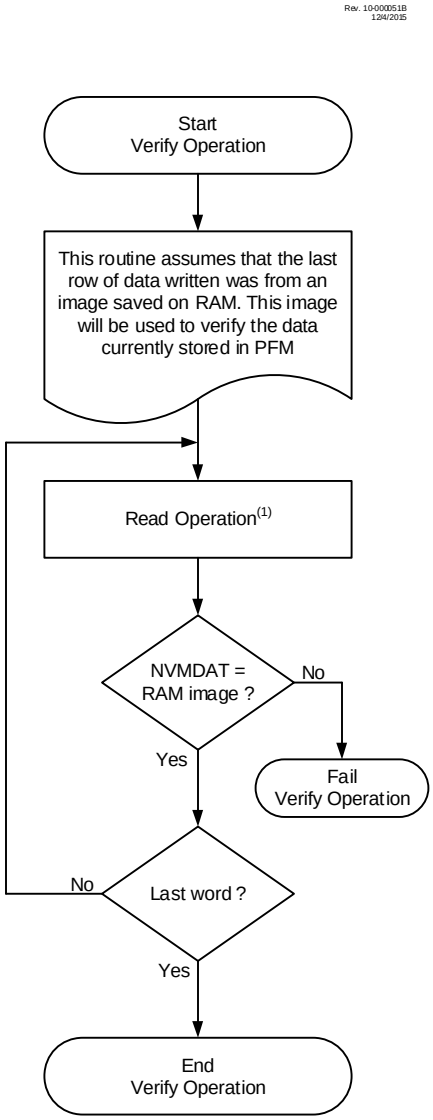
START_WRITE
   BCF        NVMCON1,LWLO        ; Latch writes complete, now write memory
   CALL       UNLOCK_SEQ          ; Perform required unlock sequence
   BCF        NVMCON1,LWLO        ; Disable writes

UNLOCK_SEQ
   MOVLW      0x55
   BCF        INTCON,GIE          ; Disable interrupts
   MOVWF      NVMCON2             ; Begin unlock sequence
   MOVLW      0xAA
   MOVWF      NVMCON2
   BSF        NVMCON1,WR
   BSF        INTCON,GIE          ; Unlock sequence complete, re-enable
interrupts
   RETURN
```

### 15.3.7 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 row then the stored program memory contents are compared with the intended data stored in RAM after the last write is complete.

**Figure 15-7. Program Flash Memory Write Verify Sequence**




**Note:**

1. See Figure 15-1.

### 15.3.8 WRERR Bit

The WRERR bit can be used to determine if a write error occurred. WRERR will be set if one of the following conditions occurs:

- If WR is set while the NVMADRH:NMVADRL points to a write-protected address
- A Reset occurs while a self-write operation was in progress
- An unlock sequence was interrupted

The WRERR bit is normally set by hardware, but can be set by the user for test purposes. Once set, WRERR must be cleared in software.

**Table 15-3. Actions for PFM When WR = 1**

| Free | LWLO | Actions for PFM when WR = 1 | Comments |
|---|---|---|---|
| 1 | x | Erase the 32-word row of NVMADRH:NVMADRL location. | • If WP is enabled, WR is cleared and WRERR is set<br>• All 32 words are erased<br>• NVMDATH:NVMDATL is ignored |
| 0 | 1 | Copy NVMDATH:NVMDATL to the write latch corresponding to NVMADR LSBs. | • Write protection is ignored<br>• No memory access occurs |
| 0 | 0 | Write the write-latch data to PFM row. | • If WP is enabled, WR is cleared and WRERR is set<br>• Write latches are reset to 0x3FFF<br>• NVMDATH:NVMDATL is ignored |

## 15.4 Register Definitions: Nonvolatile Memory Control

### 15.4.1 NVMADR

**Name:** NVMADR
**Address:** 0x81A

Nonvolatile Memory Address Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | NVMADR[14:8] | | | | | | |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NVMADR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 14:0 – NVMADR[14:0]** NVM Address Bits

**Notes:**

1. The individual bytes in this multi-byte register can be accessed with the following register names:
   - NVMADRH: Accesses the high byte NVMADR[15:8]
   - NVMADRL: Accesses the low byte NVMADR[7:0]
2. Bit [15] is undefined while WR = 1.

### 15.4.2 NVMDAT

**Name:** NVMDAT
**Address:** 0x81C

Nonvolatile Memory Data Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | NVMDAT[13:8] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NVMDAT[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 13:0 – NVMDAT[13:0]** NVM Data bits
Reset States: POR/BOR = xxxxxxxxxxxxxx
All Other Resets = uuuuuuuuuuuuuu

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- NVMDATH: Accesses the high byte NVMDAT[13:8]
- NVMDATL: Accesses the low byte NVMDAT[7:0]

### 15.4.3 NVMCON1

**Name:** NVMCON1
**Address:** 0x81E

Nonvolatile Memory Control 1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | NVMREGS | LWLO | FREE | WRERR | WREN | WR | RD |
| Access | | R/W | R/W | R/S/HC | R/W/HS | R/W | R/S/HC | R/S/HC |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 6 – NVMREGS** NVM Region Selection bit

| Value | Description |
|---|---|
| 1 | Access DIA, DCI, Configuration, User ID, Revision ID, and Device ID Registers |
| 0 | Access Program Flash Memory |

**Bit 5 – LWLO** Load Write Latches Only bit

| Value | Condition | Description |
|---|---|---|
| 1 | When FREE = 0 | The next WR command updates the write latch for this word within the row; no memory operation is initiated. |
| 0 | When FREE = 0 | The next WR command writes data or erases |
| – | Otherwise: | This bit is ignored. |

**Bit 4 – FREE** Program Flash Memory Erase Enable bit

| Value | Description |
|---|---|
| 1 | Performs an erase operation with the next WR command; the 32-word pseudo-row containing the indicated address is erased (to all 1s) to prepare for writing. |
| 0 | The next WR command writes without erasing. |

**Bit 3 – WRERR**
Write-Reset Error Flag bit(1,2,3)

| Value | Description |
|---|---|
| 1 | A write operation was interrupted by a Reset, interrupted unlock sequence, or WR was written to one while NVMADR points to a write-protected address. |
| 0 | All write operations have completed normally. |

**Bit 2 – WREN** Program/Erase Enable bit

| Value | Description |
|---|---|
| 1 | Allows program/erase cycles |
| 0 | Inhibits programming/erasing of program Flash |

**Bit 1 – WR** Write Control bit(4,5,6)

| Value | Description |
|---|---|
| 1 | Initiates the operation indicated by table in "WRERR Bit" section. |
| 0 | NVM program/erase operation is complete and inactive. |

**Bit 0 – RD** Read Control bit

| Value | Description |
|---|---|
| 1 | Initiates a read at address = NVMADR, and loads data to NVMDAT Read takes one instruction cycle and the bit is cleared when the operation is complete. The bit can only be set (not cleared) in software. |
| 0 | NVM read operation is complete and inactive |

**Notes:**

1. Bit is undefined while WR = 1.
2. Bit must be cleared by software; hardware will not clear this bit.
3. Bit may be written to '1' by the user in order to implement test sequences.
4. This bit can only be set by following the sequence described in the "NVM Unlock Sequence" section.
5. Operations are self-timed and the WR bit is cleared by hardware when complete.
6. Once a write operation is initiated, setting this bit to zero will have no effect.

### 15.4.4 NVMCON2

**Name:** NVMCON2
**Address:** 0x81F

Nonvolatile Memory Control 2 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NVMCON2[7:0] | | | | | | | |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – NVMCON2[7:0]** Flash Memory Unlock Pattern bits

**Note:** To unlock writes, a 0x55 must be written first followed by an 0xAA before setting the WR bit of the NVMCON1 register. The value written to this register is used to unlock the writes.

## 15.5 Register Summary: NVM Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 ... 0x0819 | Reserved | | | | | | | | | |
| 0x081A | NVMADR | 7:0 | NVMADR[7:0] | | | | | | | |
| | | 15:8 | | NVMADR[14:8] | | | | | | |
| 0x081C | NVMDAT | 7:0 | NVMDAT[7:0] | | | | | | | |
| | | 15:8 | | | NVMDAT[13:8] | | | | | |
| 0x081E | NVMCON1 | 7:0 | | NVMREGS | LWLO | FREE | WRERR | WREN | WR | RD |
| 0x081F | NVMCON2 | 7:0 | NVMCON2[7:0] | | | | | | | |

# 16. I/O Ports

## 16.1 Overview

**Table 16-1. Port Availability per Device**

| Device | PORTA | PORTB | PORTC |
|---|---|---|---|
| 8-pin devices | • | | |
| 14/16-pin devices | • | | • |
| 20-pin devices | • | • | • |

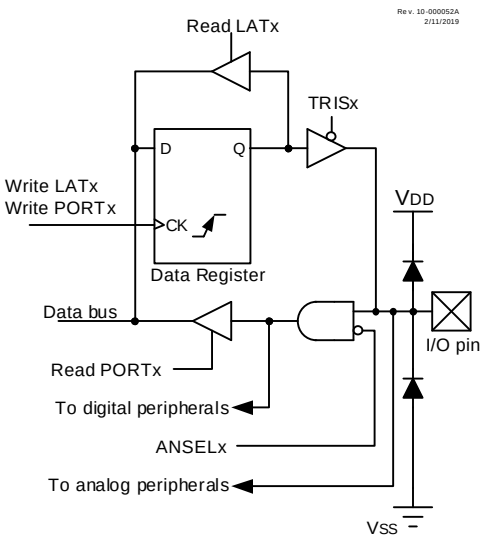
Each port has eight registers to control the operation. These registers are:

- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)
- TRISx registers (data direction)
- ANSELx registers (analog select)
- WPUx registers (weak pull-up)
- INLVLx (input level control)
- SLRCONx registers (slew rate control)
- ODCONx registers (open-drain control)

In this chapter the generic names such as PORTx, LATx, TRISx, etc. can be associated with PORTA, PORTB, PORTC, etc., depending on availability per device.

A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in the following figure:

**Figure 16-1. Generic I/O Port Operation**




## 16.2 PORTx - Data Register

PORTx is a bidirectional port, and its corresponding data direction register is TRISx.

Reading the PORTx register 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, and this value is modified, then written to the PORT data latch (LATx). The PORT data latch LATx holds the output port data and contains the latest value of a LATx or PORTx write. The example below shows how to initialize PORTA.

**Example 16-1. Initializing PORTA in assembly**

```
; This code example illustrates initializing the PORTA register.
; The other ports are initialized in the same manner.

    BANKSEL    PORTA        ;
    CLRF       PORTA        ;Clear PORTA
    BANKSEL    LATA         ;
    CLRF       LATA         ;Clear Data Latch
    BANKSEL    ANSELA       ;
    CLRF       ANSELA       ;Enable digital drivers
    BANKSEL    TRISA        ;
    MOVLW      B'00111000'  ;Set RA[5:3] as inputs
    MOVWF      TRISA        ;and set others as outputs
```

**Example 16-2. Initializing PORTA in C**

```
// This code example illustrates initializing the PORTA register.
// The other ports are initialized in the same manner.

    PORTA  = 0x00;       // Clear PORTA
    LATA   = 0x00;       // Clear Data Latch
    ANSELA = 0x00;       // Enable digital drivers
    TRISA  = 0x38;       // Set RA[5:3] as inputs and set others as outputs
```

**Important:** Most PORT pins share functions with device peripherals, both analog and digital. 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.

## 16.3 LATx - Output Latch

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.

**Important:** As a general rule, output operations to a port should use the LAT register to avoid read-modify-write issues. For example, a bit set or clear operation reads the port, modifies the bit, and writes the result back to the port. When two bit operations are executed in succession, output loading on the changed bit may delay the change at the output in which case the bit will be misread in the second bit operation and written to an unexpected level. The LAT registers are isolated from the port loading and therefore changes are not delayed.

## 16.4 TRISx - Direction Control

The TRISx register controls the PORTx pin output drivers, even when the pins are being used as analog inputs. The user should ensure the bits in the TRISx register are set when using the pins as analog inputs. I/O pins configured as analog inputs always read '0'.

Setting a TRISx bit (TRISx = 1) will make the corresponding PORTx pin an input (i.e., disable the output driver). Clearing a TRISx bit (TRISx = 0) will make the corresponding PORTx pin an output (i.e., it enables output driver and puts the contents of the output latch on the selected pin).

## 16.5 ANSELx - Analog Control

Ports that support analog inputs have an associated ANSELx register. The ANSELx register is used to configure the input mode of an I/O pin to analog. Setting an ANSELx bit high will disable the digital input buffer associated with that bit and cause the corresponding input value to always read '0', whether the value is read in PORTx register or selected by PPS as a peripheral input.

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.

The state of the ANSELx bits has no effect on digital or analog 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 PORTx register.

**Important:** The ANSELx bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be changed to '0' by user.

## 16.6 WPUx - Weak Pull-Up Control

The WPUx register controls the individual weak pull-ups for each PORT pin. When a WPUx bit is set (WPUx = 1), the weak pull-up will be enabled for the corresponding pin. When a WPUx bit is cleared (WPUx = 0), the weak pull-up will be disabled for the corresponding pin.

## 16.7 INLVLx - Input Threshold Control

The INLVLx register controls the input voltage threshold for each of the available PORTx input pins. A selection between the Schmitt Trigger CMOS or the TTL compatible thresholds is available. If that feature is enabled, the input threshold is important in determining the value of a read of the PORTx register and also all other peripherals which are connected to the input. Refer to the I/O Ports table in the **"Electrical Specifications"** chapter for more details on threshold levels.

**Important:** Changing the input threshold selection should be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin.

## 16.8 SLRCONx - Slew Rate Control

The SLRCONx register controls the slew rate option for each PORT pin. Slew rate for each PORT pin can be controlled independently. When a SLRCONx bit is set (SLRCONx = 1), the corresponding PORT pin drive is slew

rate limited. When a SLRCONx bit is cleared (SLRCONx = 0), The corresponding PORT pin drive slews at the maximum rate possible.

## 16.9 ODCONx - Open-Drain Control

The ODCONx register controls the open-drain feature of the port. Open-drain operation is independently selected for each pin. When a ODCONx bit is set (ODCONx = 1), the corresponding port output becomes an open-drain driver capable of sinking current only. When a ODCONx bit is cleared (ODCONx = 0), the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

**Important:** It is necessary to set open-drain control when using the pin for $I^2C$.

## 16.10 Edge Selectable Interrupt-on-Change

An interrupt can be generated by detecting a signal at the PORT pin that has either a rising edge or a falling edge. Individual pins can be independently configured to generate an interrupt. Refer to the **"IOC - Interrupt-on-Change"** chapter for more details.

## 16.11 $I^2C$ Pad Control

For this family of devices, the $I^2C$ specific pads are available on RB4, RB6, RC0 and RC1 pins. The $I^2C$ characteristics of each of these pins is controlled by the RxyI2C registers. These characteristics include enabling $I^2C$ specific slew rate (over standard GPIO slew rate), selecting internal pull-ups for $I^2C$ pins, and selecting appropriate input threshold as per SMBus specifications.

**Important:** Any peripheral using the $I^2C$ pins reads the $I^2C$ input levels when enabled via RxyI2C.

## 16.12 I/O Priorities

Each pin defaults to the data latch after Reset. Other functions are selected with the peripheral pin select logic. Refer to the **"PPS - Peripheral Pin Select Module"** chapter for more details.

Analog input functions, such as ADC and comparator inputs, are not shown in the peripheral pin select lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx register. Digital output functions may continue to control the pin when it is in Analog mode.

Analog outputs, when enabled, take priority over digital outputs and force the digital output driver into a high-impedance state.

The pin function priorities are as follows:

1. Port functions determined by the Configuration bits
2. Analog outputs (input buffers should be disabled)
3. Analog inputs
4. Port inputs and outputs from PPS

## 16.13 $\overline{MCLR}/V_{PP}$/RA3 Pin

The $\overline{MCLR}/V_{PP}$ pin is an input-only pin. Its operation is controlled by the MCLRE Configuration bit. When selected as a PORT pin (MCLRE = 0), it functions as a digital input-only pin; as such, it does not have TRISx and LATx bits associated with its operation. Otherwise, it functions as the device's Master Clear input. In either configuration, the $\overline{MCLR}/V_{PP}$ pin also functions as the programming voltage input pin during high voltage programming.

The $\overline{MCLR}/V_{PP}$ pin is a read-only bit and will read '1' when MCLRE = 1 (i.e., Master Clear enabled).

**Important:** On a Power-on Reset, the $\overline{MCLR}/V_{PP}$ pin is enabled as a digital input-only if Master Clear functionality is disabled.

The $\overline{MCLR}/V_{PP}$ pin has an individually controlled internal weak pull-up. When set, the corresponding WPU bit enables the pull-up. When the $\overline{MCLR}/V_{PP}$ pin is configured as $\overline{MCLR}$, (MCLRE = 1 and, LVP = 0), or configured for Low-Voltage Programming, (MCLRE = x and LVP = 1), the pull-up is always enabled and the WPU bit has no effect.

## 16.14 Register Definitions: Port Control

### 16.14.1 PORTx

**Name:** PORTx

PORTx Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | Rx7 | Rx6 | Rx5 | Rx4 | Rx3 | Rx2 | Rx1 | Rx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – Rxn** Port I/O Value
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

| Value | Description |
|---|---|
| 1 | PORT pin is ≥ $V_{IH}$ |
| 0 | PORT pin is ≤ $V_{IL}$ |

**Important:**

- Writes to PORTx are actually written to the corresponding LATx register. Reads from PORTx register return actual I/O pin values.
- The PORT bit associated with the $\overline{MCLR}$ pin is Read-Only and will read '1' when $\overline{MCLR}$ function is enabled (LVP = 1 or (LVP = 0 and MCLRE = 1)).
- Refer to the **"Pin Allocation Table"** for details about $\overline{MCLR}$ pin and pin availability per port.

### 16.14.2 LATx

**Name:** LATx

Output Latch Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LATx7 | LATx6 | LATx5 | LATx4 | LATx3 | LATx2 | LATx1 | LATx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – LATxn** Output Latch Value
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu



**Important:**

- Writes to LATx are equivalent with writes to the corresponding PORTx register. Reads from LATx register return register values, not I/O pin values.
- Refer to the **"Pin Allocation Table"** for details about pin availability per port.

### 16.14.3 TRISx

**Name:** TRISx

Tri-State Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TRISx7 | TRISx6 | TRISx5 | TRISx4 | TRISx3 | TRISx2 | TRISx1 | TRISx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – TRISxn** Port I/O Tri-state Control

| Value | Description |
|---|---|
| 1 | PORTx output driver is disabled. PORTx pin configured as an input (tri-stated) |
| 0 | PORTx output driver is enabled. PORTx pin configured as an output |

**Important:**

- The TRIS bit associated with the $\overline{MCLR}$ pin is Read-Only and the value is 1.
- Refer to the **"Pin Allocation Table"** for details about $\overline{MCLR}$ pin and pin availability per port.

### 16.14.4 ANSELx

**Name:** ANSELx

Analog Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ANSELx7 | ANSELx6 | ANSELx5 | ANSELx4 | ANSELx3 | ANSELx2 | ANSELx1 | ANSELx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – ANSELxn** Analog Select on Rx Pin

| Value | Description |
|---|---|
| 1 | Analog input. Pin is assigned as analog input. Digital input buffer disabled. |
| 0 | Digital I/O. Pin is assigned to port or digital special function. |

**Important:**

- When setting a pin as 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.
- Refer to the **"Pin Allocation Table"** for details about pin availability per port.

### 16.14.5 WPUx

**Name:** WPUx

Weak pull-up Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WPUx7 | WPUx6 | WPUx5 | WPUx4 | WPUx3 | WPUx2 | WPUx1 | WPUx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – WPUxn** Weak Pull-up PORTx Control

| Value | Description |
|---|---|
| 1 | Weak pull-up enabled |
| 0 | Weak pull-up disabled |

**Important:**

- The weak pull-up device is automatically disabled if the pin is configured as an output but this register remains unchanged.
- If MCLRE = 1, the weak pull-up on $\overline{MCLR}$ pin is always enabled and the corresponding WPU bit is not affected.
- Refer to the **"Pin Allocation Table"** for details about pin availability per port.

### 16.14.6 INLVLx

**Name:** INLVLx

Input Level Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INLVLx7 | INLVLx6 | INLVLx5 | INLVLx4 | INLVLx3 | INLVLx2 | INLVLx1 | INLVLx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – INLVLxn** Input Level Select on Rx Pin

| Value | Description |
|---|---|
| 1 | ST input used for port reads and interrupt-on-change |
| 0 | TTL input used for port reads and interrupt-on-change |

**Important:** Refer to the "**Pin Allocation Table**" for details about pin availability per port.

### 16.14.7 SLRCONx

**Name:** SLRCONx

Slew Rate Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SLRx7 | SLRx6 | SLRx5 | SLRx4 | SLRx3 | SLRx2 | SLRx1 | SLRx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – SLRxn** Slew Rate Control on Rx Pin

| Value | Description |
|---|---|
| 1 | PORT pin slew rate is limited |
| 0 | PORT pin slews at maximum rate |



**Important:** Refer to the "**Pin Allocation Table**" for details about pin availability per port.

### 16.14.8 ODCONx

**Name:** ODCONx

Open-Drain Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ODCx7 | ODCx6 | ODCx5 | ODCx4 | ODCx3 | ODCx2 | ODCx1 | ODCx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – ODCxn** Open-Drain Configuration on Rx Pin

| Value | Description |
|---|---|
| 1 | PORT pin operates as open-drain drive (sink current only) |
| 0 | PORT pin operates as standard push-pull drive (source and sink current) |



**Important:** Refer to the "**Pin Allocation Table**" for details about pin availability per port.

### 16.14.9 RxyI2C

**Name:** RxyI2C

I²C Pad Rxy Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| Access | R/W | R/W | R/W | R/W | | | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | | 0 | 0 |

**Bits 7:6 – SLEW[1:0]** I²C Specific Slew Rate Limiting Control

| Value | Description |
|---|---|
| 11 | I²C fast-mode-plus (1 MHz) slew rate enabled. The SLRxy bit is ignored. |
| 10 | Reserved |
| 01 | I²C fast-mode (400 kHz) slew rate enabled. The SLRxy bit is ignored. |
| 00 | Standard GPIO Slew Rate; enabled/disabled via SLRxy bit |

**Bits 5:4 – PU[1:0]** I²C Pull-up Selection

| Value | Description |
|---|---|
| 11 | Reserved |
| 10 | 10x current of standard weak pull-up |
| 01 | 2x current of standard weak pull-up |
| 00 | Standard GPIO weak pull-up, enabled via WPUxy bit |

**Bits 1:0 – TH[1:0]** I²C Input Threshold Selection

| Value | Description |
|---|---|
| 11 | Reserved |
| 10 | SMBus 2.0 (2.1 V) input threshold |
| 01 | I²C-specific input thresholds |
| 00 | Standard GPIO Input pull-up, enabled via INLVLxy registers |



**Important:** Refer to the "**Pin Allocation Table**" for details about I²C compatible pins.

## 16.15 Register Summary - IO Ports

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 ... 0x0B | Reserved | | | | | | | | | |
| 0x0C | PORTA | 7:0 | | | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 |
| 0x0D | PORTB | 7:0 | RB7 | RB6 | RB5 | RB4 | | | | |
| 0x0E | PORTC | 7:0 | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 |
| 0x0F ... 0x11 | Reserved | | | | | | | | | |
| 0x12 | TRISA | 7:0 | | | TRISA5 | TRISA4 | Reserved | TRISA2 | TRISA1 | TRISA0 |
| 0x13 | TRISB | 7:0 | TRISB7 | TRISB6 | TRISB5 | TRISB4 | | | | |
| 0x14 | TRISC | 7:0 | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| 0x15 ... 0x17 | Reserved | | | | | | | | | |
| 0x18 | LATA | 7:0 | | | LATA5 | LATA4 | | LATA2 | LATA1 | LATA0 |
| 0x19 | LATB | 7:0 | LATB7 | LATB6 | LATB5 | LATB4 | | | | |
| 0x1A | LATC | 7:0 | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 |
| 0x1B ... 0x010B | Reserved | | | | | | | | | |
| 0x010C | RB4I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x010D | RB6I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x010E | RC0I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x010F | RC1I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x0110 ... 0x1F37 | Reserved | | | | | | | | | |
| 0x1F38 | ANSELA | 7:0 | | | ANSELA5 | ANSELA4 | | ANSELA2 | ANSELA1 | ANSELA0 |
| 0x1F39 | WPUA | 7:0 | | | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 |
| 0x1F3A | ODCONA | 7:0 | | | ODCA5 | ODCA4 | | ODCA2 | ODCA1 | ODCA0 |
| 0x1F3B | SLRCONA | 7:0 | | | SLRA5 | SLRA4 | | SLRA2 | SLRA1 | SLRA0 |
| 0x1F3C | INLVLA | 7:0 | | | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 |
| 0x1F3D ... 0x1F42 | Reserved | | | | | | | | | |
| 0x1F43 | ANSELB | 7:0 | ANSELB7 | ANSELB6 | ANSELB5 | ANSELB4 | | | | |
| 0x1F44 | WPUB | 7:0 | WPUB7 | WPUB6 | WPUB5 | WPUB4 | | | | |
| 0x1F45 | ODCONB | 7:0 | ODCB7 | ODCB6 | ODCB5 | ODCB4 | | | | |
| 0x1F46 | SLRCONB | 7:0 | SLRB7 | SLRB6 | SLRB5 | SLRB4 | | | | |
| 0x1F47 | INLVLB | 7:0 | INLVLB7 | INLVLB6 | INLVLB5 | INLVLB4 | | | | |
| 0x1F48 ... 0x1F4D | Reserved | | | | | | | | | |
| 0x1F4E | ANSELC | 7:0 | ANSELC7 | ANSELC6 | ANSELC5 | ANSELC4 | ANSELC3 | ANSELC2 | ANSELC1 | ANSELC0 |
| 0x1F4F | WPUC | 7:0 | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 |
| 0x1F50 | ODCONC | 7:0 | ODCC7 | ODCC6 | ODCC5 | ODCC4 | ODCC3 | ODCC2 | ODCC1 | ODCC0 |
| 0x1F51 | SLRCONC | 7:0 | SLRC7 | SLRC6 | SLRC5 | SLRC4 | SLRC3 | SLRC2 | SLRC1 | SLRC0 |
| 0x1F52 | INLVLC | 7:0 | INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 |

# 17. IOC - Interrupt-on-Change

## 17.1 Overview

The pins denoted in the table below can be configured to operate as Interrupt-on-Change (IOC) pins for this device. 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.

**Table 17-1. IOC Pin Availability per Device**

| Device | PORTA | PORTB | PORTC | PORTD | PORTE | PORTF |
|---|---|---|---|---|---|---|
| 28-pin devices | • | • | • | | •(1) | |
| 40/44-pin devices | • | • | • | | •(1) | |
| 48-pin devices | • | • | • | | •(1) | |

**Note:**
1. Pin RE3 only

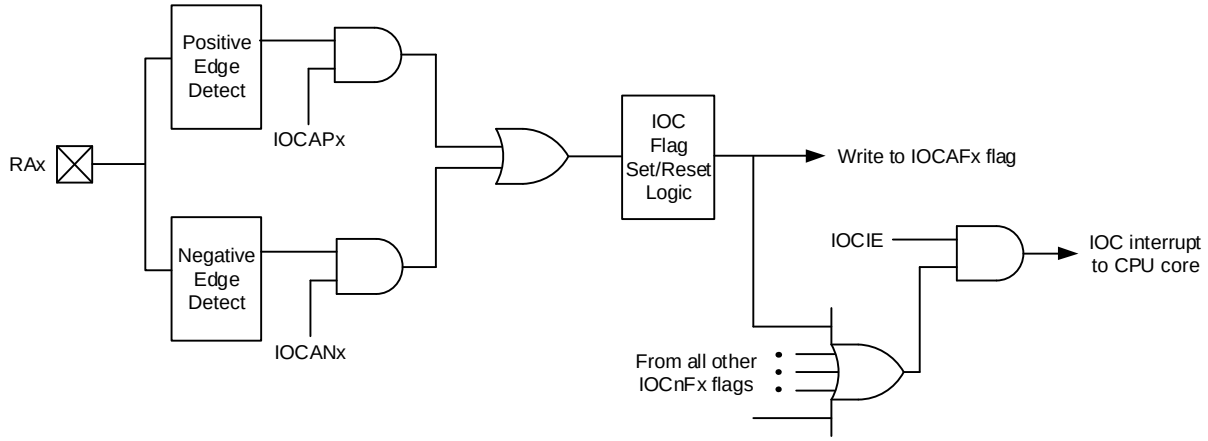
**Important:** If MCLRE = 1 or LVP = 1, the $\overline{MCLR}$ pin port functionality is disabled and IOC on that pin is not available

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

The following figure is a block diagram of the IOC module.

**Figure 17-1. Interrupt-on-Change Block Diagram (PORTA Example)**




## 17.2 Enabling the Module

In order for individual PORT pins to generate an interrupt, the IOC Interrupt Enable bit (IOCIE) of the Peripheral Interrupt Enable register (PIEx) must be set. If the IOC Interrupt Enable bit is disabled, the edge detection on the pin will still occur, but an interrupt will not be generated.

## 17.3 Individual Pin Configuration

A rising edge detector and a falling edge detector are present for each PORT pin. To enable a pin to detect a rising edge, the associated bit of the IOCxP register must be set. To enable a pin to detect a falling edge, the associated bit of the IOCxN register must be set. A PORT pin can be configured to detect rising and falling edges simultaneously by setting both associated bits of the IOCxP and IOCxN registers, respectively.

## 17.4 Interrupt Flags

The bits located in the IOCxF registers are status flags that correspond to the interrupt-on-change pins of each port. If an expected edge is detected on an appropriately enabled pin, then the status flag for that pin will be set, and an interrupt will be generated if the IOCIE bit is set. The IOCIF bit located in the corresponding Peripheral Interrupt Request register (PIRx), is all the IOCxF bits ORd together. The IOCIF bit is read-only. All of the IOCxF Status bits must be cleared to clear the IOCIF bit.

## 17.5 Clearing Interrupt Flags

The individual status flags, (IOCxF register bits), will 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 clearing an IOC interrupt flag using this method.

**Example 17-1. Clearing Interrupt Flags (PORTA Example)**

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

## 17.6 Operation in Sleep

An interrupt-on-change interrupt event 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.

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

### 17.7.1 IOCxF

**Name:** IOCxF

Interrupt-on-Change Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IOCxF7 | IOCxF6 | IOCxF5 | IOCxF4 | IOCxF3 | IOCxF2 | IOCxF1 | IOCxF0 |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCxFn** Interrupt-on-Change Flag

| Value | Condition | Description |
|---|---|---|
| 1 | IOCxP[n] = 1 | A positive edge was detected on the Rx[n] pin |
| 1 | IOCxN[n] = 1 | A negative edge was detected on the Rx[n] pin |
| 0 | IOCxP[n] = x and IOCxN[n] = x | No change was detected, or the user cleared the detected change |

**Important:**

- If MCLRE = 1 or LVP = 1, the $\overline{\text{MCLR}}$ pin port functionality is disabled and IOC on that pin is not available.
- Refer to the **"Pin Allocation Table"** for details about pins with configurable IOC per port.

### 17.7.2 IOCxN

**Name:** IOCxN

Interrupt-on-Change Negative Edge Register Example

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IOCxN7 | IOCxN6 | IOCxN5 | IOCxN4 | IOCxN3 | IOCxN2 | IOCxN1 | IOCxN0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCxNn** Interrupt-on-Change Negative Edge Enable

| Value | Description |
|---|---|
| 1 | Interrupt-on-Change enabled on the IOCx pin for a negative-going edge. Associated Status bit and interrupt flag will be set upon detecting an edge. |
| 0 | Falling edge Interrupt-on-Change disabled for the associated pin |

**Important:**

- If MCLRE = 1 or LVP = 1, the $\overline{\text{MCLR}}$ pin port functionality is disabled and IOC on that pin is not available.
- Refer to the **"Pin Allocation Table"** for details about pins with configurable IOC per port.

### 17.7.3 IOCxP

**Name:** IOCxP

Interrupt-on-Change Positive Edge Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IOCxP7 | IOCxP6 | IOCxP5 | IOCxP4 | IOCxP3 | IOCxP2 | IOCxP1 | IOCxP0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCxPn** Interrupt-on-Change Positive Edge Enable

| Value | Description |
|---|---|
| 1 | Interrupt-on-Change enabled on the IOCx pin for a positive-going edge. Associated Status bit and interrupt flag will be set upon detecting an edge. |
| 0 | Rising edge Interrupt-on-Change disabled for the associated pin. |

**Important:**

- If MCLRE = 1 or LVP = 1, the $\overline{\text{MCLR}}$ pin port functionality is disabled and IOC on that pin is not available.
- Refer to the **"Pin Allocation Table"** for details about pins with configurable IOC per port.

## 17.8 Register Summary: Interrupt-on-Change

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 ... 0x1F3C | Reserved | | | | | | | | | |
| 0x1F3D | IOCAP | 7:0 | | | IOCAP5 | IOCAP4 | IOCAP3 | IOCAP2 | IOCAP1 | IOCAP0 |
| 0x1F3E | IOCAN | 7:0 | | | IOCAN5 | IOCAN4 | IOCAN3 | IOCAN2 | IOCAN1 | IOCAN0 |
| 0x1F3F | IOCAF | 7:0 | | | IOCAF5 | IOCAF4 | IOCAF3 | IOCAF2 | IOCAF1 | IOCAF0 |
| 0x1F40 ... 0x1F47 | Reserved | | | | | | | | | |
| 0x1F48 | IOCBP | 7:0 | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | | | | |
| 0x1F49 | IOCBN | 7:0 | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | | | | |
| 0x1F4A | IOCBF | 7:0 | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | | | | |
| 0x1F4B ... 0x1F52 | Reserved | | | | | | | | | |
| 0x1F53 | IOCCP | 7:0 | IOCCP7 | IOCCP6 | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 |
| 0x1F54 | IOCCN | 7:0 | IOCCN7 | IOCCN6 | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 |
| 0x1F55 | IOCCF | 7:0 | IOCCF7 | IOCCF6 | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 |

# 18. PPS - Peripheral Pin Select Module

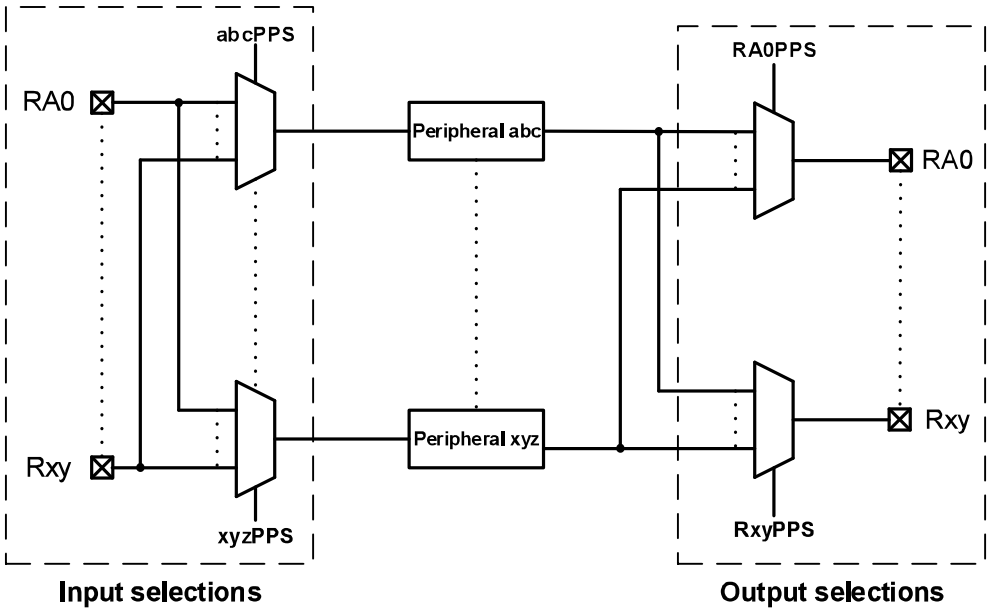
## 18.1 Overview

The Peripheral Pin Select (PPS) module connects peripheral inputs and outputs to the device I/O pins. Only digital signals are included in the selections.

**Important:** All analog inputs and outputs remain fixed to their assigned pins and cannot be changed through PPS.

Input and output selections are independent as shown in the figure below.

**Figure 18-1. PPS Block Diagram**




## 18.2 PPS Inputs

Each digital peripheral has a dedicated PPS Peripheral Input Selection (xxxPPS) register with which the input pin to the peripheral is selected. Devices that have 20 leads or less (8/14/16/20) allow PPS routing to any I/O pin, while devices with 28 leads or more allow PPS routing to I/Os contained within two ports (see the PPS Input Selection Table below).

**Important:** The notation "xxx" in the generic register name is a place holder for the peripheral identifier. For example, xxx = T0CKI for the T0CKIPPS register.

Multiple peripherals can operate from the same source simultaneously. Port reads always return the pin level regardless of peripheral PPS selection. If a pin also has analog functions associated, the ANSEL bit for that pin must be cleared to enable the digital input buffer.

**Table 18-1. PPS Input Selection Table**

| Input Signal | PPS Input Selection Register | Default Pin Location | | |
|---|---|---|---|---|
| | | PIC16F15213/14 | PIC16F15223/24 | PIC16F15243/44 |
| INT | INTPPS | RA2 | RA2 | RA2 |
| T0CKI | T0CKIPPS | RA2 | RA2 | RA2 |
| T1CKI | T1CKIPPS | RA5 | RA5 | RA5 |
| T1G | T1GPPS | RA4 | RA4 | RA4 |
| T2IN | T2INPPS | RA5 | RA5 | RA5 |
| CCP1 | CCP1PPS | RA5 | RC5 | RC5 |
| CCP2 | CCP2PPS | RA5 | RC3 | RC3 |
| SCL1/SCK1 | SSP1CLKPPS(1) | RA1 | RC0 | RB4 |
| SDA1/SDI1 | SSP1DATPPS(1) | RA2 | RC1 | RB6 |
| SS1 | SS1PPS | RA3 | RC3 | RC6 |
| RX1/DT1 | RX1PPS(1) | RA1 | RC5 | RB5 |
| TX1/CK1 | TX1PPS(1) | RA0 | RC4 | RB7 |
| ADACT | ADACTPPS | RA5 | RC2 | RC2 |

**Note:**

1. Bidirectional pin. The corresponding output must select the same pin.

## 18.3 PPS Outputs

Each digital peripheral has a dedicated Pin Rxy Output Source Selection (RxyPPS) register with which the pin output source is selected. With few exceptions, the port TRIS control associated with that pin retains control over the pin output driver. Peripherals that control the pin output driver as part of the peripheral operation will override the TRIS control as needed. The $I^2C$ module is an example of such a peripheral.



**Important:** The notation 'Rxy' is a place holder for the pin identifier. The 'x' holds the place of the PORT letter and the 'y' holds the place of the bit number. For example, Rxy = RA0 for the RA0PPS register.

The PPS Output Selection Table below shows the output codes for each peripheral, as well as the available Port selections.

**Table 18-2. PPS Output Selection Table**

| RxyPPS | Output Source |
|---|---|
| 0x09 | TMR0 |
| 0x08 | SDA1/SDO1(1) |
| 0x07 | SCL1/SCK1(1) |
| 0x06 | DT1(1) |
| 0x05 | TX1/CK1(1) |
| 0x04 | PWM4 |
| 0x03 | PWM3 |
| 0x02 | CCP2 |
| 0x01 | CCP1 |
| 0x00 | LATxy |

**Note:**

1. Bidirectional pin. The corresponding input must select the same pin.

## 18.4 Bidirectional Pins

PPS selections for peripherals with bidirectional signals on a single pin must be made so that the PPS input and PPS output select the same pin. The $I^2C$ Serial Clock (SCL) and Serial Data (SDA) are examples of such pins.

**Important:** The $I^2C$ default pins, and a limited number of other alternate pins, are $I^2C$ and SMBus compatible. SDA and SCL signals can be routed to any pin; however, pins without $I^2C$ compatibility will operate at standard TTL/ST logic levels as selected by the port's INLVL register.

## 18.5 PPS Lock

The PPS module provides an extra layer of protection to prevent inadvertent changes to the PPS selection registers. The PPSLOCKED bit is used in combination with specific code execution blocks to lock/unlock the PPS selection registers.

**Important:** The PPSLOCKED bit is clear by default (PPSLOCKED = 0), which allows the PPS selection registers to be modified without an unlock sequence.

PPS selection registers are locked when the PPSLOCKED bit is set (PPSLOCKED = 1). Setting the PPSLOCKED bit requires a specific lock sequence as shown in the examples below in both C and assembly languages.

PPS selection registers are unlocked when the PPSLOCKED bit is clear (PPSLOCKED = 0). Clearing the PPSLOCKED bit requires a specific unlock sequence as shown in the examples below in both C and assembly languages.

**Important:** All interrupts should be disabled before starting the lock/unlock sequence to ensure proper execution.

**Example 18-1.** ***PPS Lock Sequence (Assembly language)***

```
; suspend interrupts
   BCF      INTCON0,GIE
   BANKSEL  PPSLOCK
; required sequence, next 5 instructions
   MOVLW    0x55
   MOVWF    PPSLOCK
   MOVLW    0xAA
   MOVWF    PPSLOCK
; Set PPSLOCKED bit
   BSF      PPSLOCK,PPSLOCKED
; restore interrupts
   BSF      INTCON0,GIE
```

**Example 18-2.** ***PPS Lock Sequence (C language)***

```
INTCON0bits.GIE = 0;         //Suspend interrupts
PPSLOCK = 0x55;              //Required sequence
```

```
PPSLOCK = 0xAA;              //Required sequence
PPSLOCKbits.PPSLOCKED = 1;   //Set PPSLOCKED bit
INTCON0bits.GIE = 1;         //Restore interrupts
```

**Example 18-3.** ***PPS Unlock Sequence (Assembly language)***

```
; suspend interrupts
   BCF      INTCON0,GIE
   BANKSEL  PPSLOCK
; required sequence, next 5 instructions
   MOVLW    0x55
   MOVWF    PPSLOCK
   MOVLW    0xAA
   MOVWF    PPSLOCK
; Clear PPSLOCKED bit
   BCF      PPSLOCK,PPSLOCKED
; restore interrupts
   BSF      INTCON0,GIE
```

**Example 18-4.** ***PPS Unlock Sequence (C language)***

```
INTCON0bits.GIE = 0;         //Suspend interrupts
PPSLOCK = 0x55;              //Required sequence
PPSLOCK = 0xAA;              //Required sequence
PPSLOCKbits.PPSLOCKED = 0;   //Clear PPSLOCKED bit
INTCON0bits.GIE = 1;         //Restore interrupts
```

### 18.5.1 PPS One-Way Lock

The PPS1WAY Configuration bit can also be used to prevent inadvertent modification to the PPS selection registers.

When the PPS1WAY bit is set (PPS1WAY = 1), the PPSLOCKED bit can only be set one time after a device Reset. Once the PPSLOCKED bit has been set, it cannot be cleared again unless a device Reset is executed.

When the PPS1WAY bit is clear (PPS1WAY = 0), the PPSLOCKED bit can be set or cleared as needed; however, the PPS lock/unlock sequences must be executed.

## 18.6 Operation During Sleep

PPS input and output selections are unaffected by Sleep.

## 18.7 Effects of a Reset

A device Power-on Reset (POR) or Brown-out Reset (BOR) returns all PPS input selection registers to their default values, and clears all PPS output selection registers. All other Resets leave the selections unchanged. Default input selections are shown in the PPS input register details table. The PPSLOCKED bit is cleared in all Reset conditions.

## 18.8 Register Definitions: Peripheral Pin Select (PPS)

### 18.8.1 xxxPPS

**Name:** xxxPPS

Peripheral Input Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | PORT[1:0] | | PIN[2:0] | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | m | m | m | m | m |

**Bits 4:3 – PORT[1:0]** Peripheral Input PORT Selection(1)
See the PPS Input Selection Table for the list of available Ports and default pin locations.
Reset States: POR = mm
All other Resets = uu

| Value | Description |
|---|---|
| 010 | PORTC |
| 001 | PORTB |
| 000 | PORTA |

**Bits 2:0 – PIN[2:0]** Peripheral Input PORT Pin Selection(2)
Reset States: POR = mmm
All other Resets = uuu

| Value | Description |
|---|---|
| 111 | Peripheral input is from PORTx Pin 7 (Rx7) |
| 110 | Peripheral input is from PORTx Pin 6 (Rx6) |
| 101 | Peripheral input is from PORTx Pin 5 (Rx5) |
| 100 | Peripheral input is from PORTx Pin 4 (Rx4) |
| 011 | Peripheral input is from PORTx Pin 3 (Rx3) |
| 010 | Peripheral input is from PORTx Pin 2 (Rx2) |
| 001 | Peripheral input is from PORTx Pin 1 (Rx1) |
| 000 | Peripheral input is from PORTx Pin 0 (Rx0) |

**Notes:**

1. The Reset value 'm' is determined by device default locations for that input.
2. Refer to the Pin Allocation Table for details about available pins per port.

### 18.8.2 RxyPPS

**Name:** RxyPPS

Pin Rxy Output Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | RxyPPS[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – RxyPPS[5:0]** Pin Rxy Output Source Selection
See the PPS Output Selection Table for the list of RxyPPS Output Source codes
Reset States: POR = 000000
All other Resets = uuuuuu

### 18.8.3 PPSLOCK

**Name:** PPSLOCK

PPS Lock Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | PPSLOCKED |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – PPSLOCKED** PPS Locked
Reset States: POR = 0
All other Resets = 0

| Value | Description |
|---|---|
| 1 | PPS is locked. PPS selections cannot be changed. Writes to any PPS register are ignored. |
| 0 | PPS is not locked. PPS selections can be changed, but may require the PPS lock/unlock sequence. |

## 18.9 Register Summary: Peripheral Pin Select Module

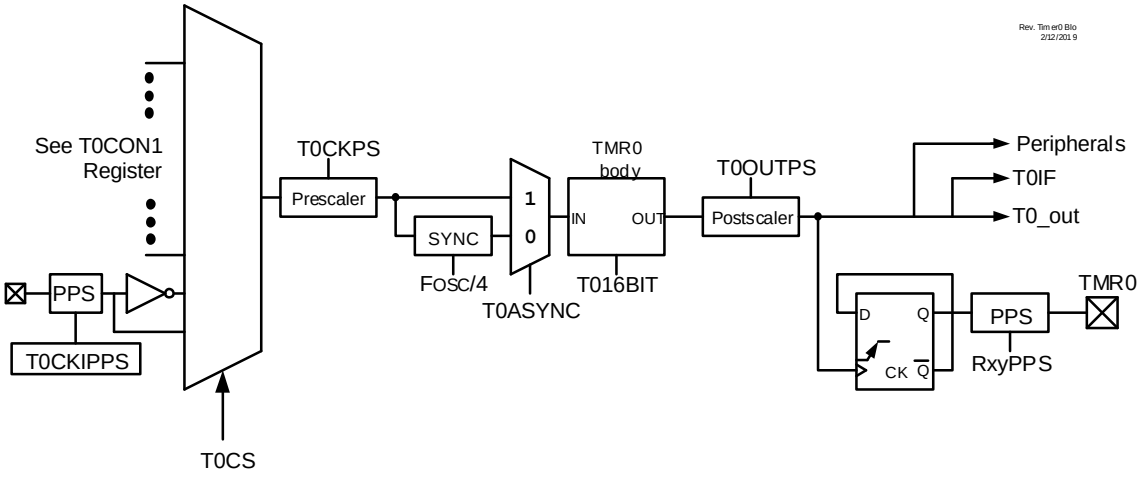
| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 ... 0x1E8F | Reserved | | | | | | | | | |
| 0x1E90 | INTPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1E91 | T0CKIPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1E92 | T1CKIPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1E93 | T1GPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1E94 ... 0x1E9B | Reserved | | | | | | | | | |
| 0x1E9C | T2INPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1E9D ... 0x1EA0 | Reserved | | | | | | | | | |
| 0x1EA1 | CCP1PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1EA2 | CCP2PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1EA3 ... 0x1EC2 | Reserved | | | | | | | | | |
| 0x1EC3 | ADACTPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1EC4 | Reserved | | | | | | | | | |
| 0x1EC5 | SSP1CLKPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1EC6 | SSP1DATPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1EC7 | SSP1SSPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1EC8 ... 0x1ECA | Reserved | | | | | | | | | |
| 0x1ECB | RX1PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1ECC | TX1PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1ECD ... 0x1F0F | Reserved | | | | | | | | | |
| 0x1F10 | RA0PPS | 7:0 | | | RA0PPS[5:0] | | | | | |
| 0x1F11 | RA1PPS | 7:0 | | | RA1PPS[5:0] | | | | | |
| 0x1F12 | RA2PPS | 7:0 | | | RA2PPS[5:0] | | | | | |
| 0x1F13 | Reserved | | | | | | | | | |
| 0x1F14 | RA4PPS | 7:0 | | | RA4PPS[5:0] | | | | | |
| 0x1F15 | RA5PPS | 7:0 | | | RA5PPS[5:0] | | | | | |
| 0x1F16 ... 0x1F1B | Reserved | | | | | | | | | |
| 0x1F1C | RB4PPS | 7:0 | | | RB4PPS[5:0] | | | | | |
| 0x1F1D | RB5PPS | 7:0 | | | RB5PPS[5:0] | | | | | |
| 0x1F1E | RB6PPS | 7:0 | | | RB6PPS[5:0] | | | | | |
| 0x1F1F | RB7PPS | 7:0 | | | RB7PPS[5:0] | | | | | |
| 0x1F20 | RC0PPS | 7:0 | | | RC0PPS[5:0] | | | | | |
| 0x1F21 | RC1PPS | 7:0 | | | RC1PPS[5:0] | | | | | |
| 0x1F22 | RC2PPS | 7:0 | | | RC2PPS[5:0] | | | | | |
| 0x1F23 | RC3PPS | 7:0 | | | RC3PPS[5:0] | | | | | |
| 0x1F24 | RC4PPS | 7:0 | | | RC4PPS[5:0] | | | | | |
| 0x1F25 | RC5PPS | 7:0 | | | RC5PPS[5:0] | | | | | |
| 0x1F26 | RC6PPS | 7:0 | | | RC6PPS[5:0] | | | | | |
| 0x1F27 | RC7PPS | 7:0 | | | RC7PPS[5:0] | | | | | |

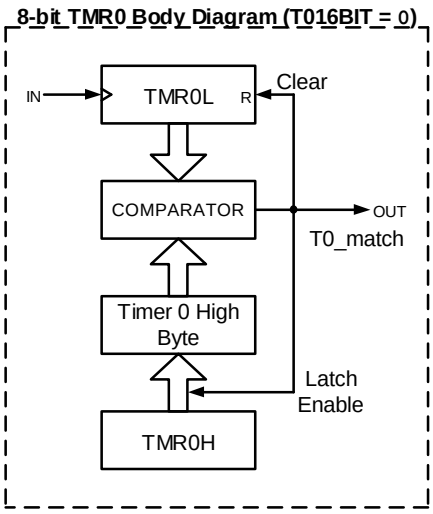
# 19. TMR0 - Timer0 Module

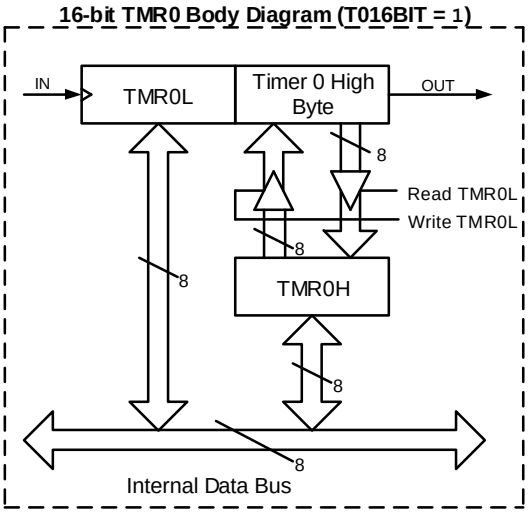
The Timer0 module has the following features:

- 8-bit timer with programmable period
- 16-bit timer
- Selectable clock sources
- Synchronous and asynchronous operation
- Programmable prescaler (Independent of Watchdog Timer)
- Programmable postscaler
- Interrupt on match or overflow
- Output on I/O pin (via PPS) or to other peripherals
- Operation during Sleep

**Figure 19-1. Timer0 Block Diagram**




## 19.1 Timer0 Operation

Timer0 can operate as either an 8-bit or 16-bit timer. The mode is selected with the MD16 bit.

### 19.1.1 8-Bit Mode

In this mode Timer0 increments on the rising edge of the selected clock source. A prescaler on the clock input gives several prescale options (see prescaler control bits, CKPS). In this mode, as shown in Figure 19-1, a buffered version of TMR0H is maintained.

This is compared with the value of TMR0L on each cycle of the selected clock source. When the two values match, the following events occur:

- TMR0L is reset
- The contents of TMR0H are copied to the TMR0H buffer for next comparison

### 19.1.2 16-Bit Mode

In this mode Timer0 increments on the rising edge of the selected clock source. A prescaler on the clock input gives several prescale options (see prescaler control bits, CKPS). In this mode TMR0H:TMR0L form the 16-bit timer value. As shown in Figure 19-1, reads and writes of the TMR0H register are buffered. The TMR0H register is updated with the contents of the high byte of Timer0 when the TMR0L register is read. Similarly, writing the TMR0L register causes a transfer of the TMR0H register value to the Timer0 high byte.

This buffering allows all 16 bits of Timer0 to be read and written at the same time. Timer0 rolls over to 0x0000 on incrementing past 0xFFFF. This makes the timer free-running. While actively operating in 16-bit mode, the Timer0 value can be read but not written.

## 19.2 Clock Selection

Timer0 has several options for clock source selections, the option to operate synchronously/asynchronously and an available programmable prescaler. The CS bits are used to select the clock source for Timer0.

### 19.2.1 Synchronous Mode

When the ASYNC bit is clear, Timer0 clock is synchronized to the system clock ($F_{OSC}/4$). When operating in Synchronous mode, Timer0 clock frequency cannot exceed $F_{OSC}/4$. During Sleep mode the system clock is not available and Timer0 cannot operate.

### 19.2.2 Asynchronous Mode

When the ASYNC bit is set, Timer0 increments with each rising edge of the input source (or output of the prescaler, if used). Asynchronous mode allows Timer0 to continue operation during Sleep mode provided the selected clock source operates during Sleep.

### 19.2.3 Programmable Prescaler

Timer0 has 16 programmable input prescaler options ranging from 1:1 to 1:32768. The prescaler values are selected using the CKPS bits. The prescaler counter is not directly readable or writable. The prescaler counter is cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset

### 19.2.4 Programmable Postscaler

Timer0 has 16 programmable output postscaler options ranging from 1:1 to 1:16. The postscaler values are selected using the OUTPS bits. The postscaler divides the output of Timer0 by the selected ratio. The postscaler counter is not directly readable or writable. The postscaler counter is cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset

## 19.3 Timer0 Output and Interrupt

### 19.3.1 Timer0 Output

TMR0_out toggles on every match between TMR0L and TMR0H in 8-bit mode, or when TMR0H:TMR0L rolls over in 16-bit mode. If the output postscaler is used, the output is scaled by the ratio selected. The Timer0 output can be routed to an I/O pin via the RxyPPS output selection register, or internally to a number of Core Independent Peripherals. The Timer0 output can be monitored through software via the OUT output bit.

### 19.3.2 Timer0 Interrupt

The Timer0 Interrupt Flag bit (TMR0IF) is set when the TMR0_out toggles. If the Timer0 interrupt is enabled (TMR0IE), the CPU will be interrupted when the TMR0IF bit is set. When the postscaler bits (T0OUTPS) are set to 1:1 operation (no division), the T0IF flag bit will be set with every TMR0 match or rollover. In general, the TMR0IF flag bit will be set every T0OUTPS +1 matches or rollovers.

### 19.3.3 Timer0 Example

Timer0 Configuration:

- Timer0 mode = 16-bit
- Clock Source = $F_{OSC}/4$ (250 kHz)
- Synchronous operation
- Prescaler = 1:1
- Postscaler = 1:2 (T0OUTPS = 1)

In this case the TMR0_out toggles every two rollovers of TMR0H:TMR0L. i.e., (0xFFFF)*2*(1/250kHz) = 524.28 ms

## 19.4 Operation During Sleep

When operating synchronously, Timer0 will halt when the device enters Sleep mode. When operating asynchronously and the selected clock source is active, Timer0 will continue to increment and wake the device from Sleep mode if the Timer0 interrupt is enabled.

## 19.5 Register Definitions: Timer0 Control

### 19.5.1 T0CON0

**Name:** T0CON0
**Address:** 0x59E

Timer0 Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OUT | MD16 | OUTPS[3:0] | | | |
| Access | R/W | | R | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** TMR0 Enable

| Value | Description |
|---|---|
| 1 | The module is enabled and operating |
| 0 | The module is disabled |

**Bit 5 – OUT** TMR0 Output

**Bit 4 – MD16** 16-Bit Timer Operation Select

| Value | Description |
|---|---|
| 1 | TMR0 is a 16-bit timer |
| 0 | TMR0 is an 8-bit timer |

**Bits 3:0 – OUTPS[3:0]** TMR0 Output Postscaler (Divider) Select

| Value | Description |
|---|---|
| 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 |

### 19.5.2 T0CON1

**Name:** T0CON1
**Address:** 0x59F

Timer0 Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CS[2:0] | | | ASYNC | CKPS[3:0] | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:5 – CS[2:0]** Timer0 Clock Source Select

**Table 19-1. Timer0 Clock Source Selections**

| T0CS | Clock Source |
|---|---|
| 111-110 | Reserved |
| 101 | MFINTOSC(500 kHz) |
| 100 | LFINTOSC |
| 011 | HFINTOSC |
| 010 | $F_{OSC}/4$ |
| 001 | Pin selected by T0CKIPPS (Inverted) |
| 000 | Pin selected by T0CKIPPS (Non-inverted) |

**Bit 4 – ASYNC** TMR0 Input Asynchronization Enable

| Value | Description |
|---|---|
| 1 | The input to the TMR0 counter is not synchronized to system clocks |
| 0 | The input to the TMR0 counter is synchronized to Fosc/4 |

**Bits 3:0 – CKPS[3:0]** Prescaler Rate Select

| Value | Description |
|---|---|
| 1111 | 1:32768 |
| 1110 | 1:16384 |
| 1101 | 1:8192 |
| 1100 | 1:4096 |
| 1011 | 1:2048 |
| 1010 | 1:1024 |
| 1001 | 1:512 |
| 1000 | 1:256 |
| 0111 | 1:128 |
| 0110 | 1:64 |
| 0101 | 1:32 |
| 0100 | 1:16 |
| 0011 | 1:8 |
| 0010 | 1:4 |
| 0001 | 1:2 |
| 0000 | 1:1 |

### 19.5.3 TMR0H

**Name:** TMR0H
**Address:** 0x59D

Timer0 Period/Count High Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMR0H[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:0 – TMR0H[7:0]** TMR0 Most Significant Counter

| Value | Condition | Description |
|---|---|---|
| 0 to 255 | MD16 = 0 | 8-bit Timer0 Period Value. TMR0L continues counting from 0 when this value is reached. |
| 0 to 255 | MD16 = 1 | 16-bit Timer0 Most Significant Byte |

### 19.5.4 TMR0L

**Name:** TMR0L
**Address:** 0x59C

Timer0 Period/Count Low Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMR0L[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TMR0L[7:0]** TMR0 Least Significant Counter

| Value | Condition | Description |
|---|---|---|
| 0 to 255 | MD16 = 0 | 8-bit Timer0 Counter bits |
| 0 to 255 | MD16 = 1 | 16-bit Timer0 Least Significant Byte |

## 19.6 Register Summary: Timer0

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x059B | Reserved | | | | | | | | | |
| 0x059C | TMR0L | 7:0 | TMR0L[7:0] | | | | | | | |
| 0x059D | TMR0H | 7:0 | TMR0H[7:0] | | | | | | | |
| 0x059E | T0CON0 | 7:0 | EN | | OUT | MD16 | OUTPS[3:0] | | | |
| 0x059F | T0CON1 | 7:0 | CS[2:0] | | | ASYNC | CKPS[3:0] | | | |

# 20. TMR1 - Timer1 Module with Gate Control

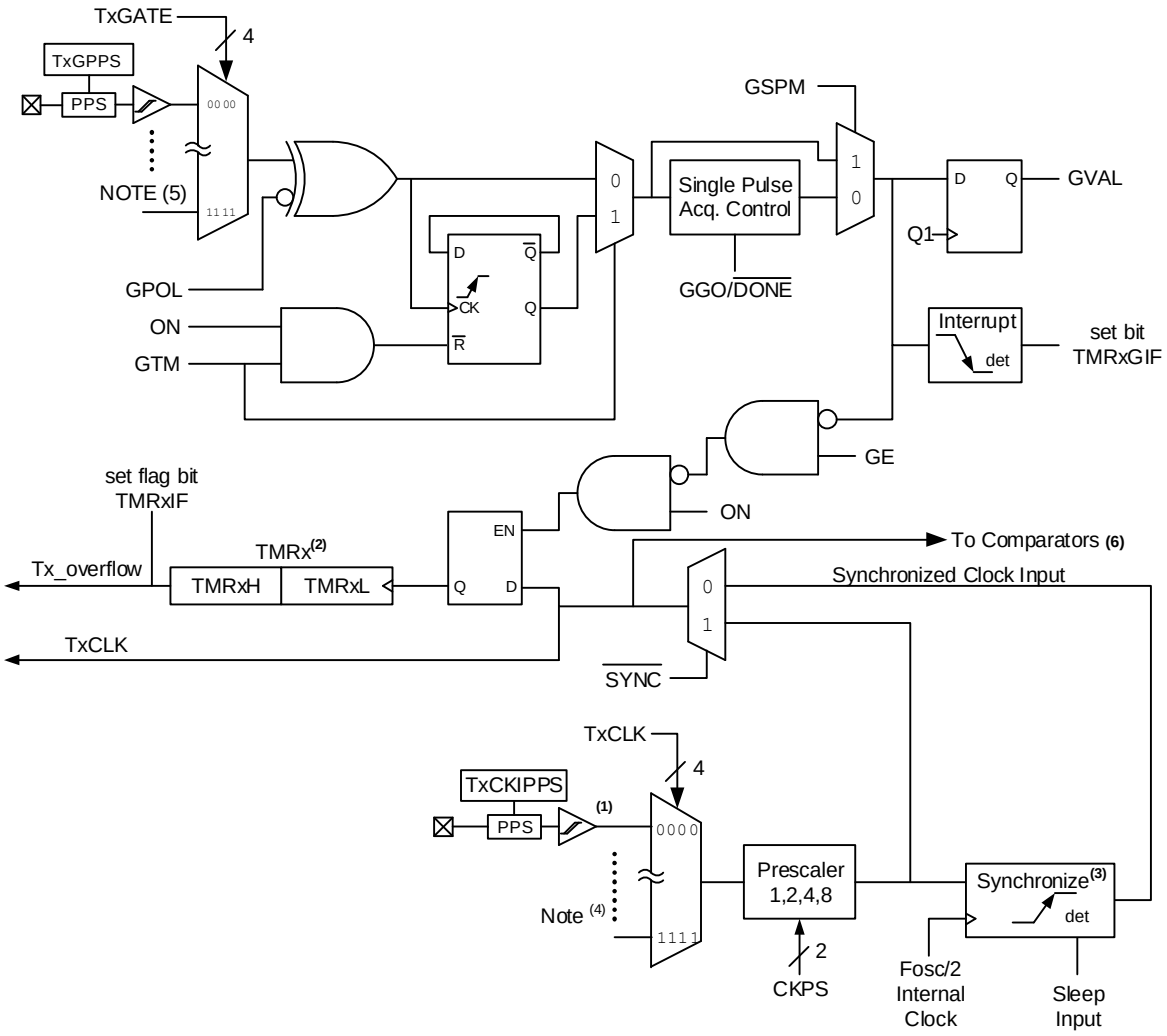
The Timer1 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMRxH:TMRxL)
- Programmable internal or external clock source
- 2-bit prescaler
- Clock source for optional comparator synchronization
- Multiple Timer1 gate (count enable) sources
- Interrupt-on-overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- 16-bit read/write operation
- Time base for the capture/compare function with the CCP modules
- Special event trigger (with CCP)
- Selectable gate source polarity
- Gate Toggle mode
- Gate Single-Pulse mode
- Gate value status
- Gate event interrupt



**Important:** References to the module Timer1 apply to all the odd numbered timers on this device.

**Figure 20-1. Timer1 Block Diagram**




**Notes:**

1. This signal comes from the pin selected by Timer1 PPS register.
2. TMRx register increments on rising edge.
3. Synchronize does not operate while in Sleep.
4. See TxCLK for clock source selections.
5. See TxGATE for gate source selections.
6. Synchronized comparator output should not be used in conjunction with synchronized input clock.

## 20.1 Timer1 Operation

The Timer1 module is a 16-bit incrementing counter that is accessed through the TMRx register. Writes to TMRx directly update the counter. When used with an internal clock source, the module is a timer that 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 ON and GE bits. Table 20-1 displays the possible Timer1 enable selections.

**Table 20-1. Timer1 Enable Selections**

| ON | GE | Timer1 Operation |
|---|---|---|
| 1 | 1 | Count Enabled |
| 1 | 0 | Always On |
| 0 | 1 | Off |
| 0 | 0 | Off |

## 20.2 Clock Source Selection

The CS bits select the clock source for Timer1. These bits allow the selection of several possible synchronous and asynchronous clock sources.

### 20.2.1 Internal Clock Source

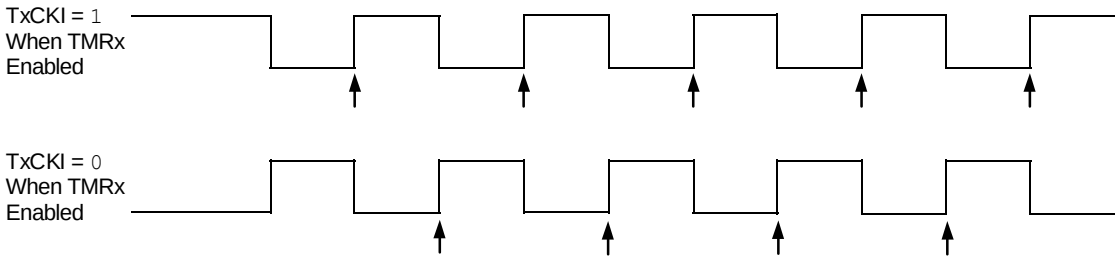
When the internal clock source is selected the TMRx register will increment on multiples of $F_{OSC}$ as determined by the Timer1 prescaler.

When the $F_{OSC}$ internal clock source is selected, the TMRx 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 TMRx value. To utilize the full resolution of Timer1, an asynchronous input signal must be used to gate the Timer1 clock input.

**Important:** 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 TMRxH or TMRxL
- Timer1 is disabled
- Timer1 is disabled (ON = 0) when TxCKI is high then Timer1 is enabled (ON = 1) when TxCKI is low. Refer to the figure below.

**Figure 20-2. Timer1 Incrementing Edge**




**Notes:**

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.

### 20.2.2 External Clock Source

When the external clock source is selected, the TMRx 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 of the TxCKIPPS pin. This external clock source can be synchronized to the system clock or it can run asynchronously.

## 20.3 Timer1 Prescaler

Timer1 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The CKPS bits control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMRx.

## 20.4 Timer1 Operation in Asynchronous Counter Mode

When the SYNC control bit 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.

**Important:** 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.

### 20.4.1 Reading and Writing TMRx in Asynchronous Counter Mode

Reading TMRxH or TMRxL while the timer is running from an external asynchronous clock will ensure a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since there may be a carry out of TMRxL to TMRxH between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers, while the register is incrementing. This may produce an unpredictable value in the TMRxH:TMRxL register pair.

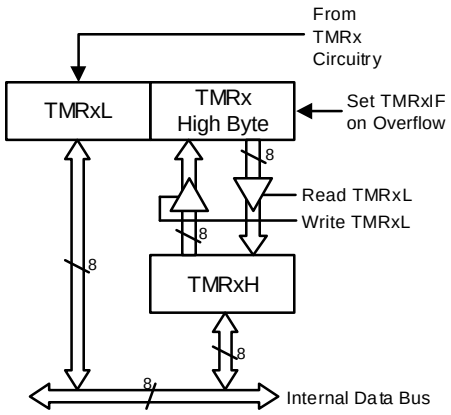
## 20.5 Timer1 16-Bit Read/Write Mode

Timer1 can be configured to read and write all 16 bits of data to and from the 8-bit TMRxL and TMRxH registers, simultaneously. The 16-bit read and write operations are enabled by setting the RD16 bit. To accomplish this function, the TMRxH register value is mapped to a buffer register called the TMRxH buffer register. While in 16-Bit mode, the TMRxH register is not directly readable or writable and all read and write operations take place through the use of this TMRxH buffer register.

When a read from the TMRxL register is requested, the value of the TMRxH register is simultaneously loaded into the TMRxH buffer register. When a read from the TMRxH register is requested, the value is provided from the TMRxH buffer register instead. This provides the user with the ability to accurately read all 16 bits of the Timer1 value from a single instance in time. Refer to the figure below for more details. In contrast, when not in 16-Bit mode, the user must read each register separately and determine if the values have become invalid due to a rollover that may have occurred between the read operations.

When a write request of the TMRxL register is requested, the TMRxH buffer register is simultaneously updated with the contents of the TMRxH register. The value of TMRxH must be preloaded into the TMRxH buffer register prior to the write request for the TMRxL register. This provides the user with the ability to write all 16 bits to the TMRx register at the same time. Any requests to write to TMRxH directly does not clear the Timer1 prescaler value. The prescaler value is only cleared through write requests to the TMRxL register.

**Figure 20-3. Timer1 16-Bit Read/Write Mode Block Diagram**




## 20.6 Timer1 Gate

Timer1 can be configured to count freely or the count can be enabled and disabled using Timer1 gate circuitry. This is also referred to as Timer1 gate enable. Timer1 gate can also be driven by multiple selectable sources.

### 20.6.1 Timer1 Gate Enable

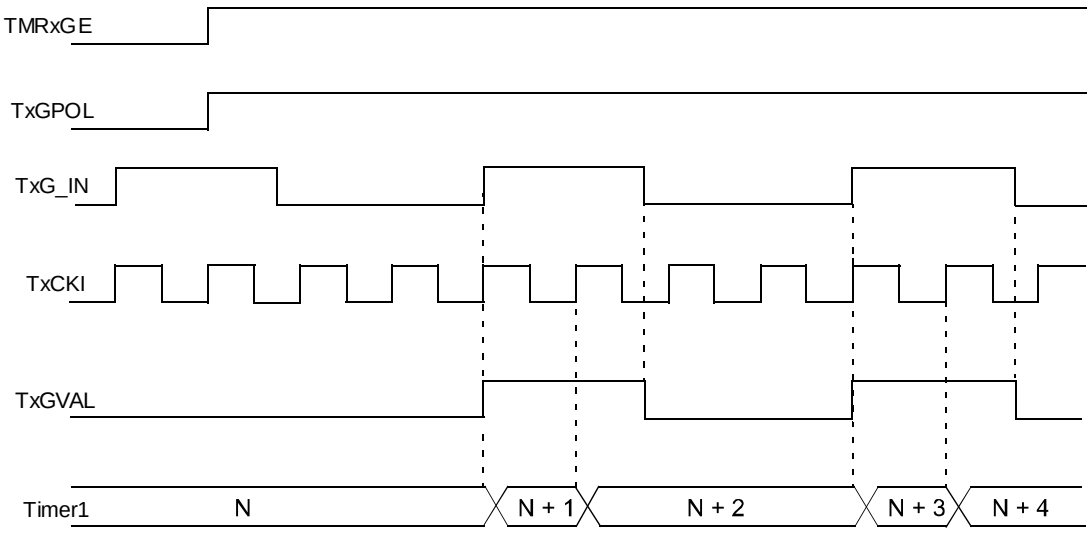
The Timer1 Gate Enable mode is enabled by setting the GE bit. The polarity of the Timer1 Gate Enable mode is configured using the GPOL bit.

When Timer1 Gate Enable mode is enabled, Timer1 will increment on the rising edge of the Timer1 clock source. When Timer1 Gate signal is inactive, the timer will not increment and hold the current count. Enable mode is disabled, no incrementing will occur and Timer1 will hold the current count. See figure below for timing details.

**Table 20-2. Timer1 Gate Enable Selections**

| TMRxCLK | GPOL | TxG | Timer1 Operation |
|---|---|---|---|
| ↑ | 1 | 1 | Counts |
| ↑ | 1 | 0 | Holds Count |
| ↑ | 0 | 1 | Holds Count |
| ↑ | 0 | 0 | Counts |

**Figure 20-4. Timer1 Gate Enable Mode**




### 20.6.2 Timer1 Gate Source Selection

The gate source for Timer1 is selected using the GSS bits. The polarity selection for the gate source is controlled by the GPOL bit.

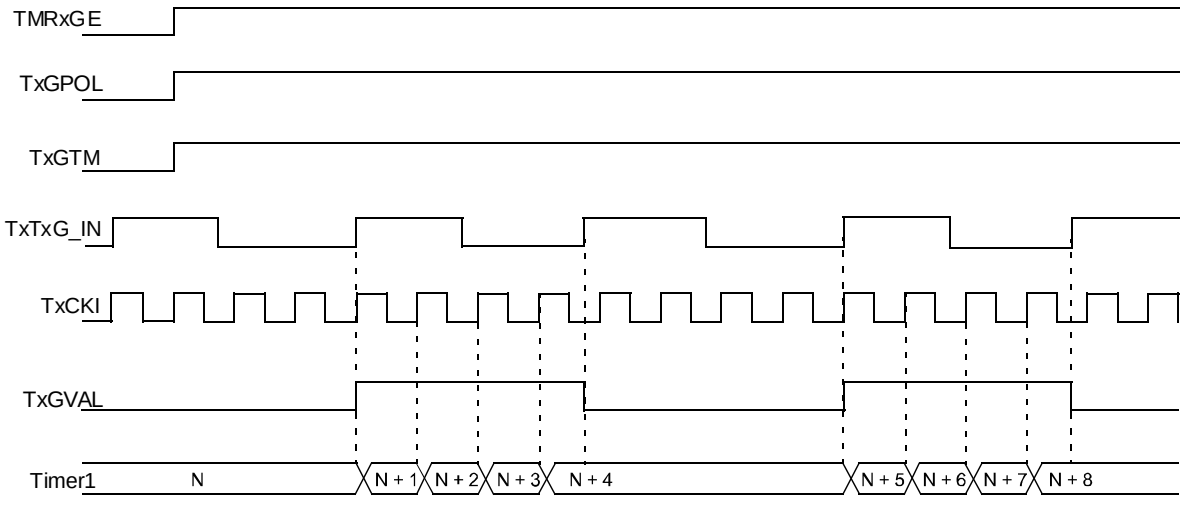
### 20.6.3 Timer1 Gate Toggle Mode

When Timer1 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1 Gate signal, as opposed to the duration of a single level pulse. The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See figure below for timing details.

Timer1 Gate Toggle mode is enabled by setting the GTM bit. When the GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

**Important:** Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

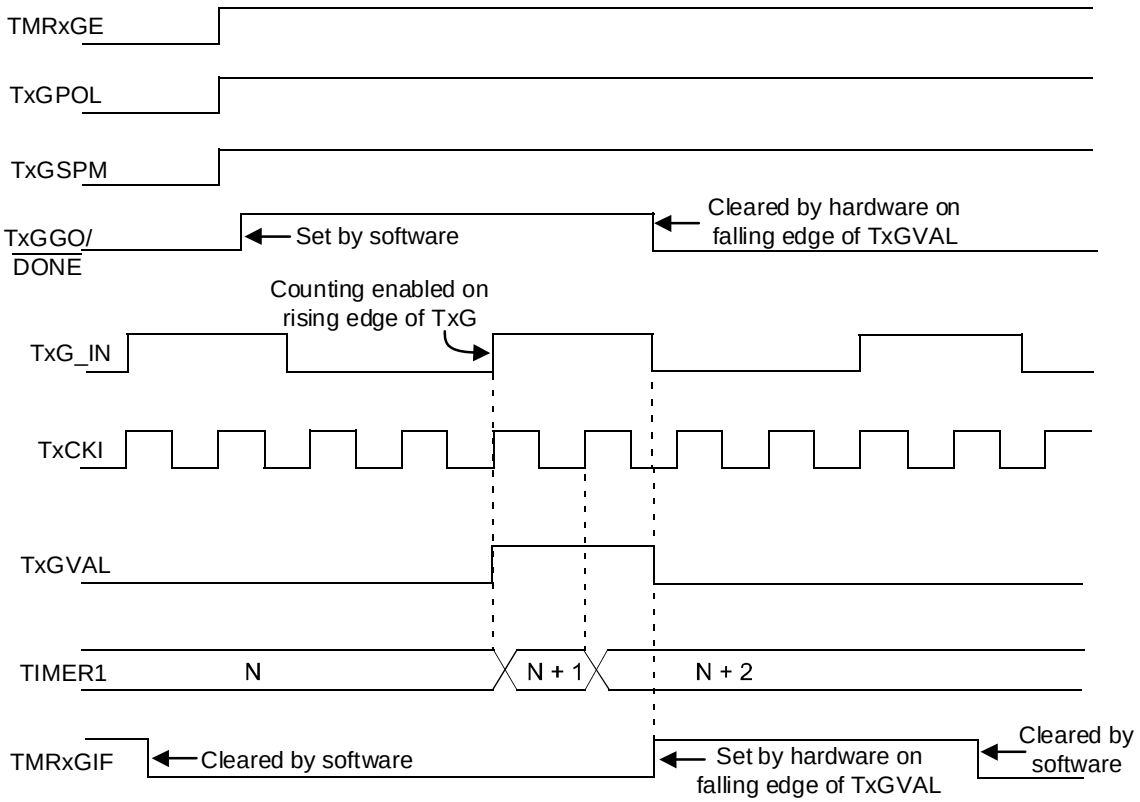
**Figure 20-5. Timer1 Gate Toggle Mode**



### 20.6.4 Timer1 Gate Single Pulse Mode

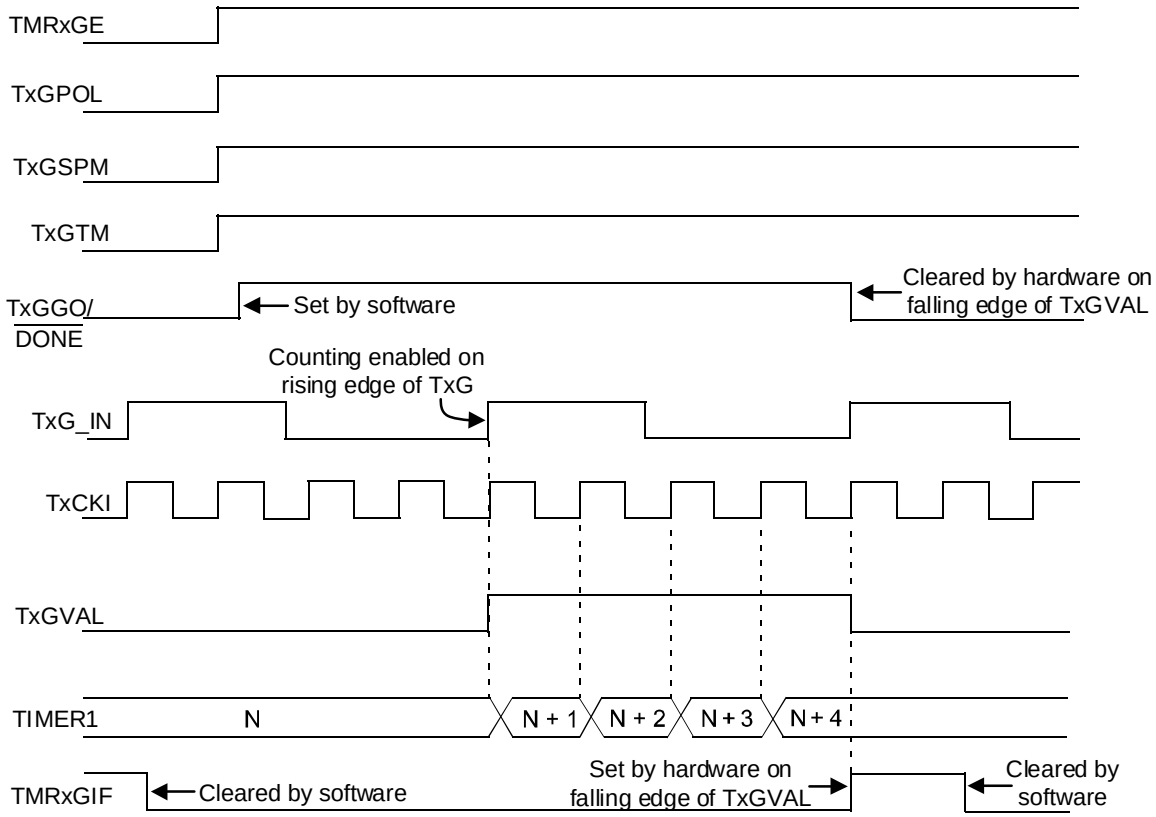
When Timer1 Gate Single Pulse mode is enabled, it is possible to capture a single pulse gate event. Timer1 Gate Single Pulse mode is first enabled by setting the GSPM bit. Next, the GGO/DONE must be set. The Timer1 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the GGO/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the GGO/$\overline{\text{DONE}}$ bit is once again set in software.

**Figure 20-6. Timer1 Gate Single Pulse Mode**




Clearing the GSPM bit will also clear the GGO/$\overline{\text{DONE}}$ bit. See the figure below for timing details. Enabling the Toggle mode and the Single Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1 gate source to be measured. See figure below for timing details.

**Figure 20-7. Timer1 Gate Single Pulse and Toggle Combined Mode**




### 20.6.5 Timer1 Gate Value Status

When Timer1 gate value status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the GVAL bit in the TxGCON register. The GVAL bit is valid even when the Timer1 gate is not enabled (GE bit is cleared).

### 20.6.6 Timer1 Gate Event Interrupt

When Timer1 gate event interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of GVAL occurs, the TMRxGIF flag bit will be set. If the TMRxGIE bit in the corresponding PIE register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer1 gate is not enabled (GE bit is cleared).

## 20.7 Timer1 Interrupt

The TMRx register increments to FFFFh and rolls over to 0000h. When TMRx rolls over, the TMRx interrupt flag (TMRxIF) bit of the PIRx register is set. To enable the interrupt-on-rollover, the following bits must be set:

- ON bit of the TxCON register
- TMRxIE bits of the PIEx register
- Global interrupts must be enabled

The interrupt is cleared by clearing the TMRxIF bit as a task in the Interrupt Service Routine.

**Important:** The TMRx register and the TMRxIF bit should be cleared before enabling interrupts.

## 20.8 Timer1 Operation During Sleep

Timer1 can only operate during Sleep when configured as an asynchronous counter. In this mode, many clock sources can be used to increment the counter. To set up the timer to wake the device:

- ON bit must be set
- TMRxIE bit of the PIEx register must be set
- Global interrupts must be enabled
- SYNC bit must be set
- Configure the TxCLK register for using any clock source other than $F_{OSC}$ and $F_{OSC}/4$

The device will wake-up on an overflow and execute the next instruction. If global interrupts are enabled, the device will call the Interrupt Service Routine. The secondary oscillator will continue to operate in Sleep regardless of the SYNC bit setting.

## 20.9 CCP Capture/Compare Time Base

The CCP modules use TMRx as the time base when operating in Capture or Compare mode. In Capture mode, the value in TMRx is copied into the CCPRx register on a capture event. In Compare mode, an event is triggered when the value in the CCPRx register matches the value in TMRx. This event can be a Special Event Trigger. For more information, see the **"CCP - Capture/Compare/PWM Module"** chapter.

## 20.10 CCP Special Event Trigger

When any of the CCPs are configured to trigger a special event, the trigger will clear the TMRx register. This special event does not cause a Timer1 interrupt. The CCP module may still be configured to generate a CCP interrupt. In this mode of operation, the CCPRx register becomes the period register for Timer1. Timer1 should be synchronized and $F_{OSC}/4$ should be selected as the clock source in order to utilize the Special Event Trigger. Asynchronous operation of Timer1 can cause a Special Event Trigger to be missed. In the event that a write to TMRxH or TMRxL coincides with a Special Event Trigger from the CCP, the write will take precedence.

## 20.11 Register Definitions: Timer1 Control

Long bit name prefixes for the System Arbiter Priority Registers are shown in the table below where "x" refers to the Priority Register instance number. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 20-3. Timer1 Register Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| Timer1 | T1 |

### 20.11.1 TxCON

**Name:** TxCON
**Address:** 0x20E

Timer Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | CKPS[1:0] | | | $\overline{SYNC}$ | RD16 | ON |
| Access | | | R/W | R/W | | R/W | R/W | R/W |
| Reset | | | 0 | 0 | | 0 | 0 | 0 |

**Bits 5:4 – CKPS[1:0]** Timer Input Clock Prescaler Select
Reset States: POR/BOR = 00
All Other Resets = uu

| Value | Description |
|---|---|
| 11 | 1:8 Prescaler value |
| 10 | 1:4 Prescaler value |
| 01 | 1:2 Prescaler value |
| 00 | 1:1 Prescaler value |

**Bit 2 – $\overline{SYNC}$** Timer External Clock Input Synchronization Control
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Condition | Description |
|---|---|---|
| x | CS = $F_{OSC}$/4 or $F_{OSC}$ | This bit is ignored. Timer uses the incoming clock as is. |
| 1 | All other clock sources | Do not synchronize external clock input |
| 0 | All other clock sources | Synchronize external clock input with system clock |

**Bit 1 – RD16** 16-Bit Read/Write Mode Enable
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Enables register read/write of Timer in one 16-bit operation |
| 0 | Enables register read/write of Timer in two 8-bit operations |

**Bit 0 – ON** Timer On
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Enables Timer |
| 0 | Disables Timer |

### 20.11.2 TxGCON

**Name:** TxGCON
**Address:** 0x20F

Timer Gate Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | GE | GPOL | GTM | GSPM | GGO/$\overline{\text{DONE}}$ | GVAL | | |
| Access | R/W | R/W | R/W | R/W | R/W | R | | |
| Reset | 0 | 0 | 0 | 0 | 0 | x | | |

**Bit 7 – GE** Timer Gate Enable
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Condition | Description |
|---|---|---|
| 1 | ON = 1 | Timer counting is controlled by the Timer gate function |
| 0 | ON = 1 | Timer is always counting |
| X | ON = 0 | This bit is ignored |

**Bit 6 – GPOL** Timer Gate Polarity
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Timer gate is active-high (Timer counts when gate is high) |
| 0 | Timer gate is active-low (Timer counts when gate is low) |

**Bit 5 – GTM** Timer Gate Toggle Mode
Timer Gate Flip-Flop Toggles on every rising edge when Toggle mode is enabled.
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Timer Gate Toggle mode is enabled |
| 0 | Timer Gate Toggle mode is disabled and Toggle flip-flop is cleared |

**Bit 4 – GSPM** Timer Gate Single Pulse Mode
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Timer Gate Single Pulse mode is enabled and is controlling Timer gate |
| 0 | Timer Gate Single Pulse mode is disabled |

**Bit 3 – GGO/$\overline{\text{DONE}}$** Timer Gate Single Pulse Acquisition Status
This bit is automatically cleared when TxGSPM is cleared.
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Timer Gate Single Pulse Acquisition is ready, waiting for an edge |
| 0 | Timer Gate Single Pulse Acquisition has completed or has not been started |

**Bit 2 – GVAL** Timer Gate Current State
Indicates the current state of the timer gate that could be provided to TMRxH:TMRxL
Unaffected by Timer Gate Enable (GE bit)

### 20.11.3 TxCLK

**Name:** TxCLK
**Address:** 0x211

Timer Clock Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CS[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CS[4:0]** Timer Clock Source Selection

**Table 20-4. Timer Clock Sources**

| CS | Clock Source |
|---|---|
| 11111-01001 | Reserved |
| 01000 | TMR0_OUT |
| 00111 | MFINTOSC (32 kHz) |
| 00110 | MFINTOSC (500 kHz) |
| 00101 | SFINTOSC (1 MHz) |
| 00100 | LFINTOSC |
| 00011 | HFINTOSC |
| 00010 | $F_{OSC}$ |
| 00001 | $F_{OSC}/4$ |
| 00000 | Pin selected by T1CKIPPS |

Reset States: POR/BOR = 00000
All Other Resets = uuuuu

### 20.11.4 TxGATE

**Name:** TxGATE
**Address:** 0x210

Timer Gate Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | GSS[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – GSS[4:0]** Timer Gate Source Selection

**Table 20-5. Timer Gate Sources**

| GSS | Gate Source |
|---|---|
| 11111-00111 | Reserved |
| 00110 | PWM4_OUT |
| 00101 | PWM3_OUT |
| 00100 | CCP2_OUT |
| 00011 | CCP1_OUT |
| 00010 | TMR2_Postscaler_OUT |
| 00001 | TMR0_OUT |
| 00000 | Pin selected by T1GPPS |

### 20.11.5 TMRx

**Name:** TMRx
**Address:** 0x20C

Timer Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | TMRx[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMRx[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – TMRx[15:0]** Timer Register Value
Reset States: POR/BOR = 0000000000000000
All Other Resets = uuuuuuuuuuuuuuuu

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- TMRxH: Accesses the high byte TMRx[15:8]
- TMRxL: Accesses the low byte TMRx[7:0]

## 20.12 Register Summary Timer 1

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x020B | Reserved | | | | | | | | | |
| 0x020C | TMRx | 7:0 | TMR1[7:0] | | | | | | | |
| | | 15:8 | TMR1[15:8] | | | | | | | |
| 0x020E | TxCON | 7:0 | | | CKPS[1:0] | | | SYNC | RD16 | ON |
| 0x020F | TxGCON | 7:0 | GE | GPOL | GTM | GSPM | GGO/DONE | GVAL | | |
| 0x0210 | TxGATE | 7:0 | | | | GSS[4:0] | | | | |
| 0x0211 | TxCLK | 7:0 | | | | CS[4:0] | | | | |

# 21. TMR2 - Timer2 Module

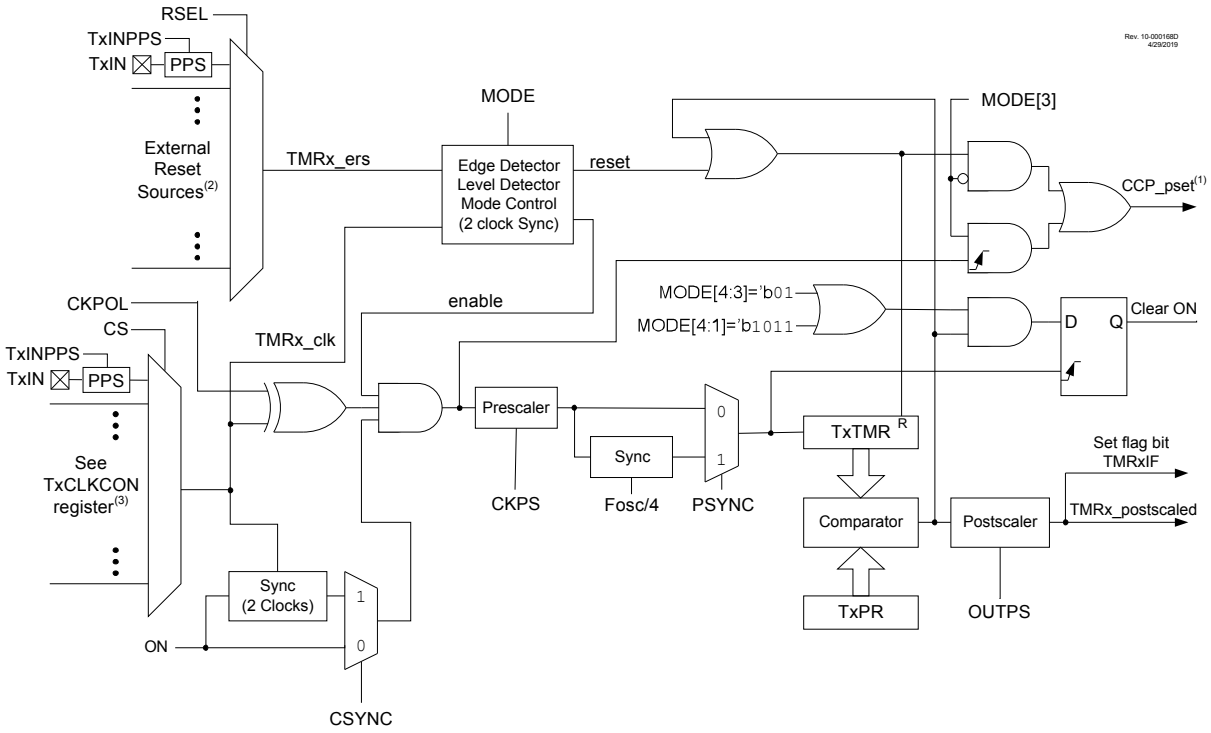
The Timer2 module is a 8-bit timer that incorporates the following features:

- 8-bit timer and period registers
- Readable and writable
- Software programmable prescaler (1:1 to 1:128)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on T2TMR match with T2PR
- One-shot operation
- Full asynchronous operation
- Includes Hardware Limit Timer (HLT)
- Alternate clock sources
- External timer Reset signal sources
- Configurable timer Reset operation

See Figure 21-1 for a block diagram of Timer2.

**Important:** References to module Timer2 apply to all the even numbered timers on this device. (Timer2, Timer4, etc.)

**Figure 21-1. Timer2 with Hardware Limit Timer (HLT) Block Diagram**




**Notes:**

1. Signal to the CCP peripheral for PWM pulse trigger in PWM mode.
2. See RSEL for external Reset sources.
3. See CS for clock source selections.

## 21.1 Timer2 Operation

Timer2 operates in three major modes:

- Free-Running Period
- One-shot
- Monostable

Within each operating mode there are several options for starting, stopping, and Reset. Table 21-1 lists the options.

In all modes, the T2TMR count register increments on the rising edge of the clock signal from the programmable prescaler. When T2TMR equals T2PR, a high level output to the postscaler counter is generated. T2TMR is cleared on the next clock input.

An external signal from hardware can also be configured to gate the timer operation or force a T2TMR count Reset. In Gate modes the counter stops when the gate is disabled and resumes when the gate is enabled. In Reset modes the T2TMR count is reset on either the level or edge from the external source.

The T2TMR and T2PR registers are both directly readable and writable. The T2TMR register is cleared and the T2PR register initializes to 0xFF on any device Reset. Both the prescaler and postscaler counters are cleared on the following events:

- A write to the T2TMR register
- A write to the T2CON register
- Any device Reset
- External Reset source event that resets the timer.



**Important:** T2TMR is not cleared when T2CON is written.

### 21.1.1 Free-Running Period Mode

The value of T2TMR is compared to that of the Period register, T2PR, on each clock cycle. When the two values match, the comparator resets the value of T2TMR to 0x00 on the next cycle and increments the output postscaler counter. When the postscaler count equals the value in the OUTPS bits of the T2CON register then a one clock period wide pulse occurs on the TMR2_postscaled output, and the postscaler count is cleared.

### 21.1.2 One-Shot Mode

The One-Shot mode is identical to the Free-Running Period mode except that the ON bit is cleared and the timer is stopped when T2TMR matches T2PR and will not restart until the ON bit is cycled off and on. Postscaler (OUTPS) values other than zero are ignored in this mode because the timer is stopped at the first period event and the postscaler is reset when the timer is restarted.

### 21.1.3 Monostable Mode

Monostable modes are similar to One-Shot modes except that the ON bit is not cleared and the timer can be restarted by an external Reset event.

## 21.2 Timer2 Output

The Timer2 module's primary output is TMR2_postscaled, which pulses for a single TMR2_clk period upon each match of the postscaler counter and the OUTPS bits of the T2CON register. The postscaler is incremented each time the T2TMR value matches the T2PR value. This signal can also be selected as an input to other Core Independent Peripherals:

In addition, the Timer2 is also used by the CCP module for pulse generation in PWM mode. See "**PWM Overview**" and "**PWM Period**" sections in the **"CCP - Capture/Compare/PWM Module"** chapter for more details on setting up Timer2 for use with the CCP and PWM modules.

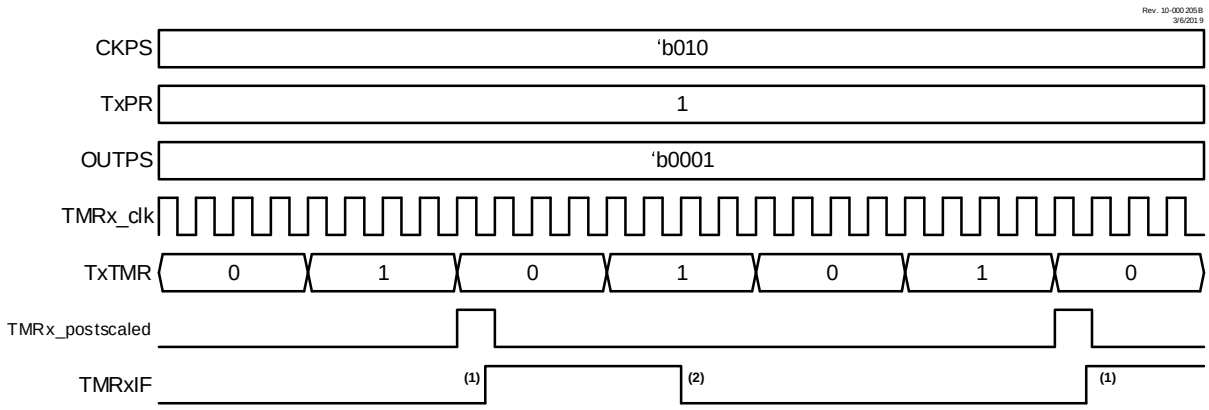
## 21.3 External Reset Sources

In addition to the clock source, the Timer2 can also be driven by an external Reset source input. This external Reset input is selected for each timer with the corresponding TxRST register. The external Reset input can control starting and stopping of the timer, as well as resetting the timer, depending on the mode used.

## 21.4 Timer2 Interrupt

Timer2 can also generate a device interrupt. The interrupt is generated when the postscaler counter matches the selected postscaler value (OUTPS bits of T2CON register). The interrupt is enabled by setting the TMR2IE interrupt enable bit. Interrupt timing is illustrated in the figure below.

**Figure 21-2. Timer2 Prescaler, Postscaler, and Interrupt Timing Diagram**




Note 1: Setting the interrupt flag is synchronized with the instruction clock. Synchronization may take as many as 2 instruction cycles
2: Cleared by software.

## 21.5 PSYNC bit

Setting the PSYNC bit synchronizes the prescaler output to $F_{OSC}/4$. Setting this bit is required for reading the Timer2 counter register while the selected Timer clock is asynchronous to $F_{OSC}/4$.

**Note:** Setting PSYNC requires that the output of the prescaler is slower than $F_{OSC}/4$. Setting PSYNC when the output of the prescaler is greater than or equal to $F_{OSC}/4$ may cause unexpected results.

## 21.6 CSYNC bit

All bits in the Timer2 SFRs are synchronized to $F_{OSC}/4$ by default, not the Timer2 input clock. As such, if the Timer2 input clock is not synchronized to $F_{OSC}/4$, it is possible for the Timer2 input clock to transition at the same time as the ON bit is set in software, which may cause undesirable behavior and glitches in the counter. Setting the CSYNC bit remedies this problem by synchronizing the ON bit to the Timer2 input clock instead of $F_{OSC}/4$. However, as this synchronization uses an edge of the TMR2 input clock, up to one input clock cycle will be consumed and not counted by the Timer2 when CSYNC is set. Conversely, clearing the CSYNC bit synchronizes the ON bit to $F_{OSC}/4$, which does not consume any clock edges, but has the previously stated risk of glitches.

## 21.7 Operating Modes

The mode of the timer is controlled by the MODE bits. Edge-Triggered modes require six Timer clock periods between external triggers. Level-Triggered modes require the triggering level to be at least three Timer clock periods long. External triggers are ignored while in Debug mode.

**Table 21-1. Operating Modes Table**

| Mode | MODE | | Output Operation | Operation | Timer Control | | |
|---|---|---|---|---|---|---|---|
| | [4:3] | [2:0] | | | Start | Reset | Stop |
| Free-Running Period | 00 | 000 | Period Pulse | Software gate (Figure 21-3) | ON = 1 | — | ON = 0 |
| | | 001 | | Hardware gate, active-high (Figure 21-4) | ON = 1 and TMRx_ers = 1 | — | ON = 0 or TMRx_ers = 0 |
| | | 010 | | Hardware gate, active-low | ON = 1 and TMRx_ers = 0 | — | ON = 0 or TMRx_ers = 1 |
| | | 011 | Period Pulse with Hardware Reset | Rising or falling edge Reset | ON = 1 | TMRx_ers ↕ | ON = 0 |
| | | 100 | | Rising edge Reset (Figure 21-5) | | TMRx_ers ↑ | |
| | | 101 | | Falling edge Reset | | TMRx_ers ↓ | |
| | | 110 | | Low level Reset | | TMRx_ers = 0 | ON = 0 or TMRx_ers = 0 |
| | | 111 | | High level Reset (Figure 21-6) | | TMRx_ers = 1 | ON = 0 or TMRx_ers = 1 |
| One-shot | 01 | 000 | One-shot | Software start (Figure 21-7) | ON = 1 | — | ON = 0 or Next clock after TxTMR = TxPR (Note 2) |
| | | 001 | Edge-Triggered Start (Note 1) | Rising edge start (Figure 21-8) | ON = 1 and TMRx_ers ↑ | — | |
| | | 010 | | Falling edge start | ON = 1 and TMRx_ers ↓ | — | |
| | | 011 | | Any edge start | ON = 1 and TMRx_ers ↕ | — | |
| | | 100 | Edge-Triggered Start and Hardware Reset (Note 1) | Rising edge start and Rising edge Reset (Figure 21-9) | ON = 1 and TMRx_ers ↑ | TMRx_ers ↑ | |
| | | 101 | | Falling edge start and Falling edge Reset | ON = 1 and TMRx_ers ↓ | TMRx_ers ↓ | |
| | | 110 | | Rising edge start and Low level Reset (Figure 21-10) | ON = 1 and TMRx_ers ↑ | TMRx_ers = 0 | |
| | | 111 | | Falling edge start and High level Reset | ON = 1 and TMRx_ers ↓ | TMRx_ers = 1 | |

.....continued

| Mode | MODE | | Output Operation | Operation | Timer Control | | |
|---|---|---|---|---|---|---|---|
| | [4:3] | [2:0] | | | Start | Reset | Stop |
| Monostable | 10 | 000 | Reserved | | | | |
| | | 001 | Edge-Triggered Start (Note 1) | Rising edge start (Figure 21-11) | ON = 1 and TMRx_ers ↑ | — | ON = 0 or Next clock after TxTMR = TxPR (Note 3) |
| | | 010 | | Falling edge start | ON = 1 and TMRx_ers ↓ | — | |
| | | 011 | | Any edge start | ON = 1 and TMRx_ers ↕ | — | |
| Reserved | | 100 | Reserved | | | | |
| Reserved | | 101 | Reserved | | | | |
| One-shot | | 110 | Level Triggered Start and Hardware Reset | High level start and Low level Reset (Figure 21-12) | ON = 1 and TMRx_ers = 1 | TMRx_ers = 0 | ON = 0 or Held in Reset (Note 2) |
| | | 111 | | Low level start and High level Reset | ON = 1 and TMRx_ers = 0 | TMRx_ers = 1 | |
| Reserved | 11 | xxx | Reserved | | | | |

**Notes:**

1. If ON = 0 then an edge is required to restart the timer after ON = 1.
2. When T2TMR = T2PR then the next clock clears ON and stops T2TMR at 00h.
3. When T2TMR = T2PR then the next clock stops T2TMR at 00h but does not clear ON.

## 21.8 Operation Examples

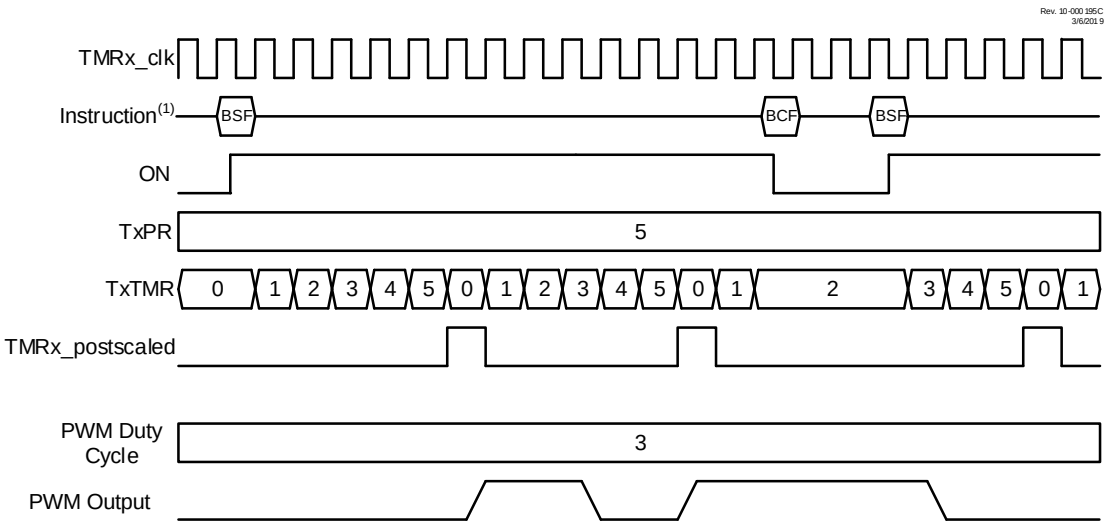
Unless otherwise specified, the following notes apply to the following timing diagrams:

- Both the prescaler and postscaler are set to 1:1 (both the CKPS and OUTPS bits.)
- The diagrams illustrate any clock except $F_{OSC}/4$ and show clock-sync delays of at least two full cycles for both ON and TMRx_ers. When using $F_{OSC}/4$, the clock-sync delay is at least one instruction period for TMRx_ers; ON applies in the next instruction period.
- ON and TMRx_ers are somewhat generalized, and clock-sync delays may produce results that are slightly different than illustrated.
- The PWM Duty Cycle and PWM output are illustrated assuming that the timer is used for the PWM function of the CCP module as described in the "**PWM Overview**" section. The signals are not a part of the Timer2 module.

### 21.8.1 Software Gate Mode

This mode corresponds to legacy Timer2 operation. The timer increments with each clock input when ON = 1 and does not increment when ON = 0. When the TxTMR count equals the TxPR period count the timer resets on the next clock and continues counting from 0. Operation with the ON bit software controlled is illustrated in Figure 21-3. With TxPR = 5, the counter advances until TxTMR = 5, and goes to zero with the next clock.

**Figure 21-3. Software Gate Mode Timing Diagram (MODE = 'b00000)**




Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

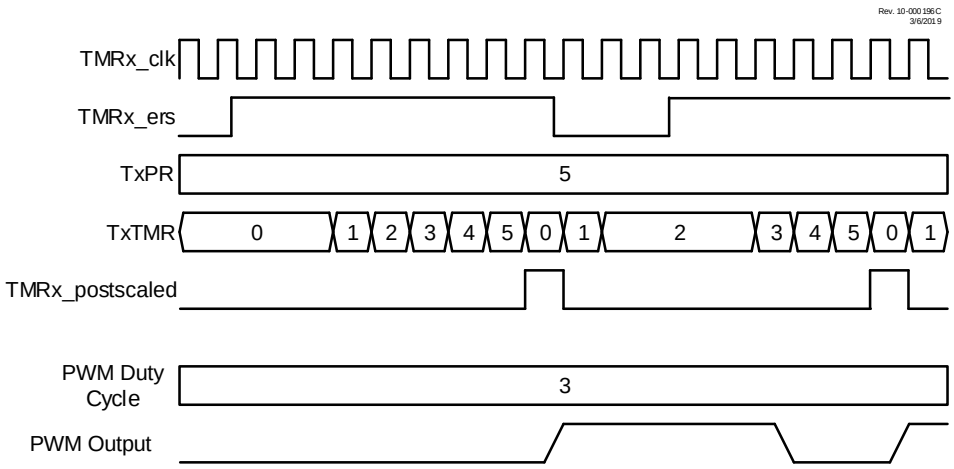
### 21.8.2 Hardware Gate Mode

The Hardware Gate modes operate the same as the Software Gate mode except the TMRx_ers external signal can also gate the timer. When used with the CCP, the gating extends the PWM period. If the timer is stopped when the PWM output is high, then the duty cycle is also extended.

When MODE = `'b00001` then the timer is stopped when the external signal is high. When MODE = `'b00010`, then the timer is stopped when the external signal is low.

Figure 21-4 illustrates the Hardware Gating mode for MODE = `'b00001` in which a high input level starts the counter.

**Figure 21-4. Hardware Gate Mode Timing Diagram (MODE = `'b00001`)**



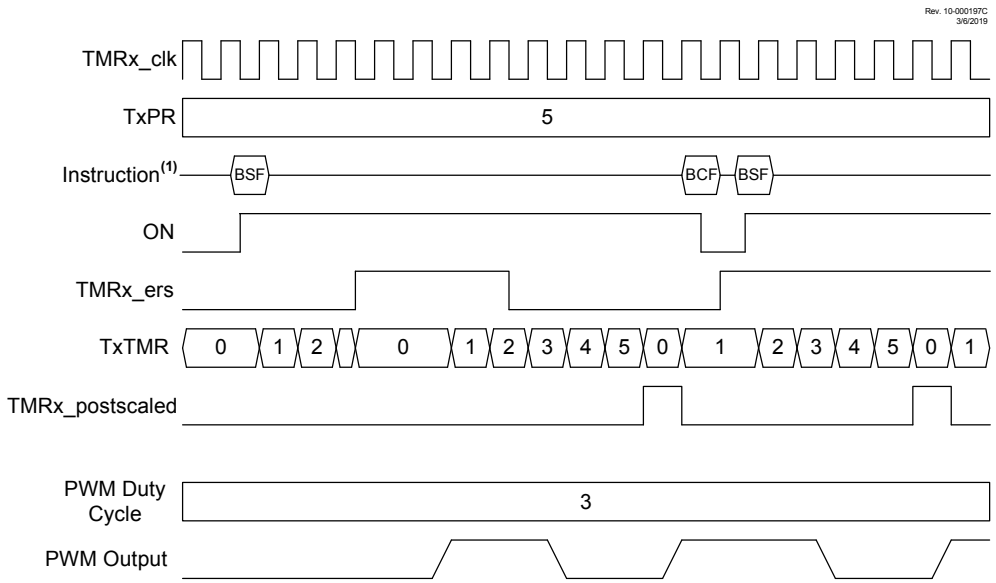
### 21.8.3 Edge-Triggered Hardware Limit Mode

In Hardware Limit mode, the timer can be reset by the TMRx_ers external signal before the timer reaches the period count. Three types of Resets are possible:

- Reset on rising or falling edge (MODE= 'b00011)
- Reset on rising edge (MODE = 'b00100)
- Reset on falling edge (MODE = 'b00101)

When the timer is used in conjunction with the CCP in PWM mode then an early Reset shortens the period and restarts the PWM pulse after a two clock delay. Refer to Figure 21-5.

**Figure 21-5. Edge-Triggered Hardware Limit Mode Timing Diagram (MODE = 'b00100)**




Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

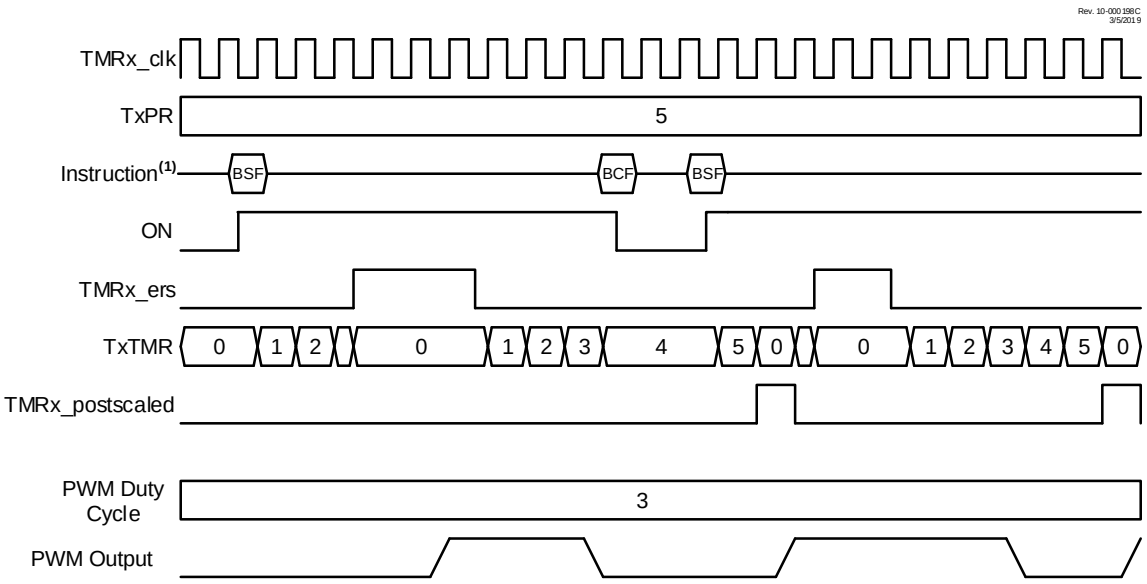
### 21.8.4 Level-Triggered Hardware Limit Mode

In the Level-Triggered Hardware Limit Timer modes the counter is reset by high or low levels of the external signal TMRx_ers, as shown in Figure 21-6. Selecting MODE = `'b00110` will cause the timer to reset on a low level external signal. Selecting MODE = `'b00111` will cause the timer to reset on a high level external signal. In the example, the counter is reset while TMRx_ers = `1`. ON is controlled by `BSF` and `BCF` instructions. When ON = `0` the external signal is ignored.

When the CCP uses the timer as the PWM time base then the PWM output will be set high when the timer starts counting and then set low only when the timer count matches the CCPRx value. The timer is reset when either the timer count matches the TxPR value or two clock periods after the external Reset signal goes true and stays true.

The timer starts counting, and the PWM output is set high, on either the clock following the TxPR match or two clocks after the external Reset signal relinquishes the Reset. The PWM output will remain high until the timer counts up to match the CCPRx pulse width value. If the external Reset signal goes true while the PWM output is high then the PWM output will remain high until the Reset signal is released allowing the timer to count up to match the CCPRx value.

**Figure 21-6. Level-Triggered Hardware Limit Mode Timing Diagram (MODE = `'b00111`)**



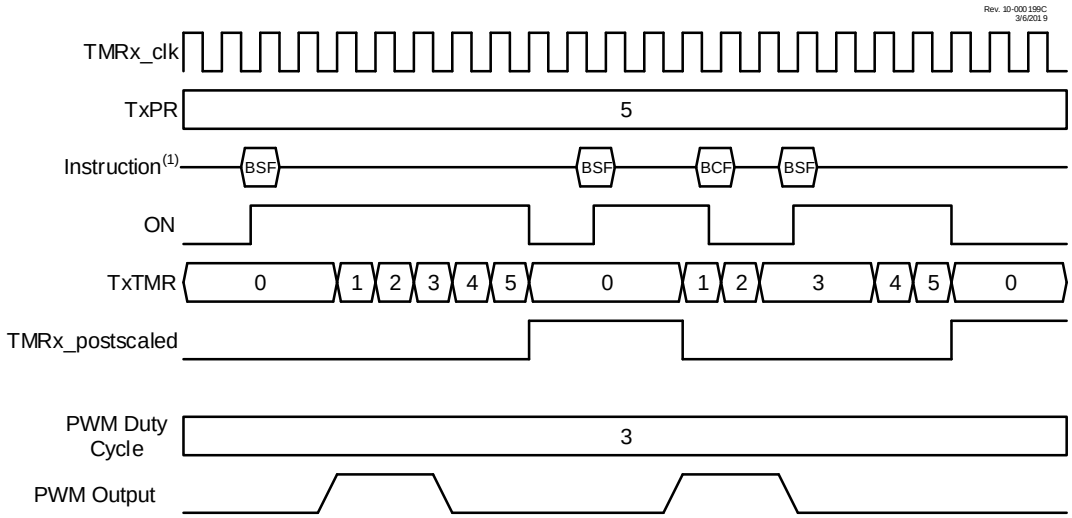

Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 21.8.5 Software Start One-Shot Mode

In One-Shot mode the timer resets and the ON bit is cleared when the timer value matches the TxPR period value. The ON bit must be set by software to start another timer cycle. Setting MODE = `'b01000` selects One-Shot mode which is illustrated in Figure 21-7. In the example, ON is controlled by `BSF` and `BCF` instructions. In the first case, a `BSF` instruction sets ON and the counter runs to completion and clears ON. In the second case, a `BSF` instruction starts the cycle, `BCF`/`BSF` instructions turn the counter off and on during the cycle, and then it runs to completion.

When One-Shot mode is used in conjunction with the CCP PWM operation the PWM pulse drive starts concurrent with setting the ON bit. Clearing the ON bit while the PWM drive is active will extend the PWM drive. The PWM drive will terminate when the timer value matches the CCPRx pulse width value. The PWM drive will remain off until software sets the ON bit to start another cycle. If software clears the ON bit after the CCPRx match but before the TxPR match then the PWM drive will be extended by the length of time the ON bit remains cleared. Another timing cycle can only be initiated by setting the ON bit after it has been cleared by a TxPR period count match.

**Figure 21-7. Software Start One-Shot Mode Timing Diagram (MODE = `'b01000`)**




Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 21.8.6 Edge-Triggered One-Shot Mode

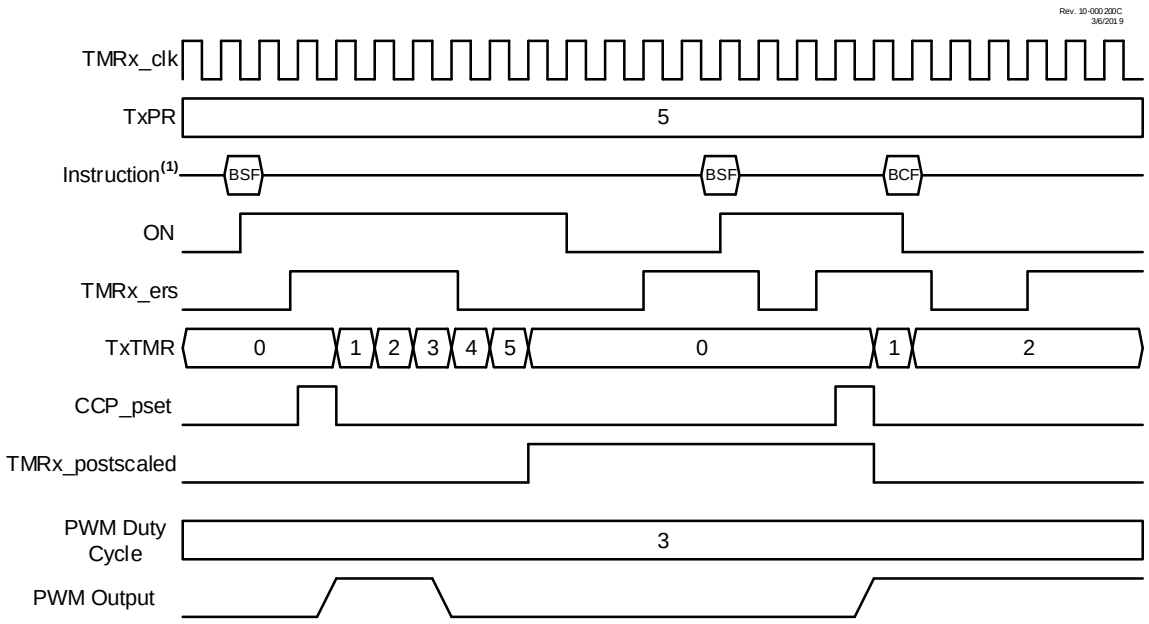
The Edge-Triggered One-Shot modes start the timer on an edge from the external signal input, after the ON bit is set, and clear the ON bit when the timer matches the TxPR period value. The following edges will start the timer:

- Rising edge (MODE = 'b01001)
- Falling edge (MODE = 'b01010)
- Rising or Falling edge (MODE = 'b01011)

If the timer is halted by clearing the ON bit then another TMRx_ers edge is required after the ON bit is set to resume counting. Figure 21-8 illustrates operation in the rising edge One-Shot mode.

When Edge-Triggered One-Shot mode is used in conjunction with the CCP then the edge-trigger will activate the PWM drive and the PWM drive will deactivate when the timer matches the CCPRx pulse width value and stay deactivated when the timer halts at the TxPR period count match.

**Figure 21-8. Edge-Triggered One-Shot Mode Timing Diagram (MODE = 'b01001)**




Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 21.8.7 Edge-Triggered Hardware Limit One-Shot Mode

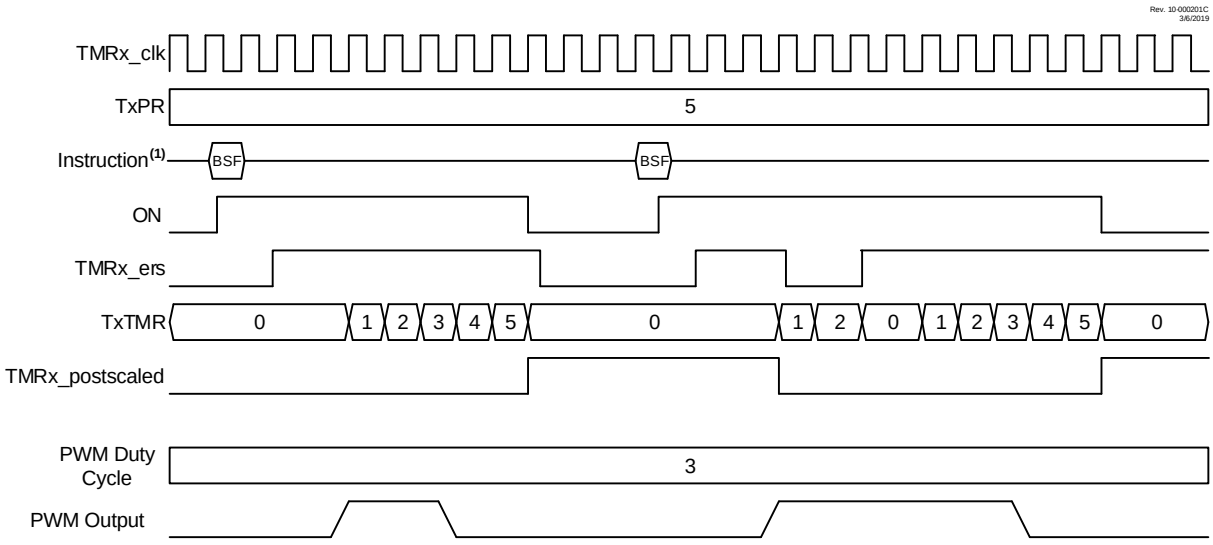
In Edge-Triggered Hardware Limit One-Shot modes the timer starts on the first external signal edge after the ON bit is set and resets on all subsequent edges. Only the first edge after the ON bit is set is needed to start the timer. The counter will resume counting automatically two clocks after all subsequent external Reset edges. Edge triggers are as follows:

- Rising edge start and Reset (MODE = `'b01100`)
- Falling edge start and Reset (MODE = `'b01101`)

The timer resets and clears the ON bit when the timer value matches the TxPR period value. External signal edges will have no effect until after software sets the ON bit. Figure 21-9 illustrates the rising edge hardware limit one-shot operation.

When this mode is used in conjunction with the CCP then the first starting edge trigger, and all subsequent Reset edges, will activate the PWM drive. The PWM drive will deactivate when the timer matches the CCPRx pulse width value and stay deactivated until the timer halts at the TxPR period match unless an external signal edge resets the timer before the match occurs.

**Figure 21-9. Edge-Triggered Hardware Limit One-Shot Mode Timing Diagram (MODE = `'b01100`)**




Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 21.8.8 Level Reset, Edge-Triggered Hardware Limit One-Shot Modes

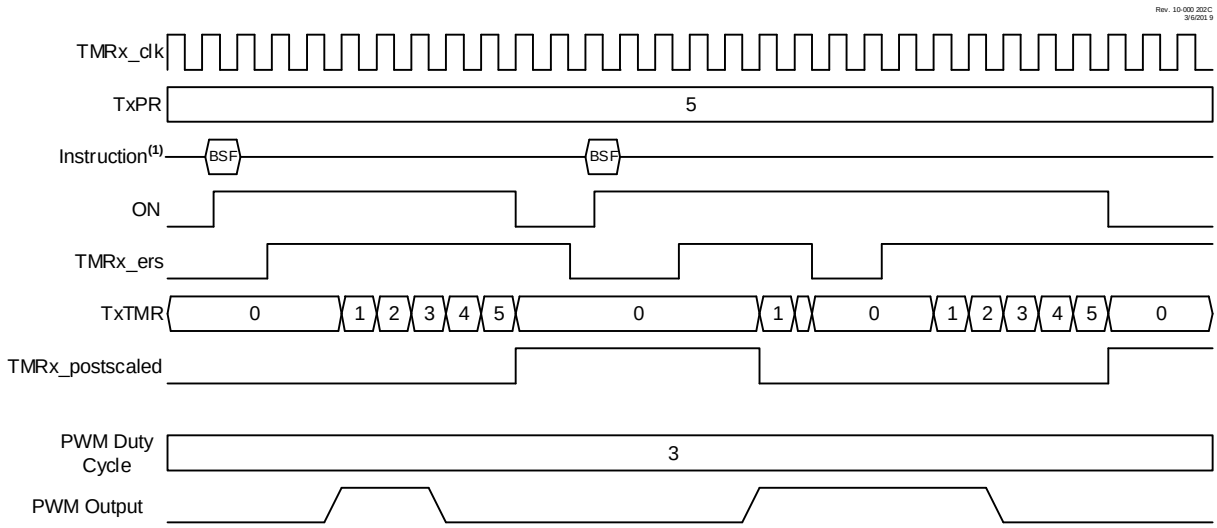
In Level-Triggered One-Shot mode the timer count is reset on the external signal level and starts counting on the rising/falling edge of the transition from Reset level to the active level while the ON bit is set. Reset levels are selected as follows:

- Low Reset level (MODE = 'b01110)
- High Reset level (MODE = 'b01111)

When the timer count matches the TxPR period count, the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a TxPR match or by software control, a new external signal edge is required after the ON bit is set to start the counter.

When Level-Triggered Reset One-Shot mode is used in conjunction with the CCP PWM operation, the PWM drive goes active with the external signal edge that starts the timer. The PWM drive goes inactive when the timer count equals the CCPRx pulse width count. The PWM drive does not go active when the timer count clears at the TxPR period count match.

**Figure 21-10. Low Level Reset, Edge-Triggered Hardware Limit One-Shot Mode Timing Diagram (MODE = 'b01110)**




Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

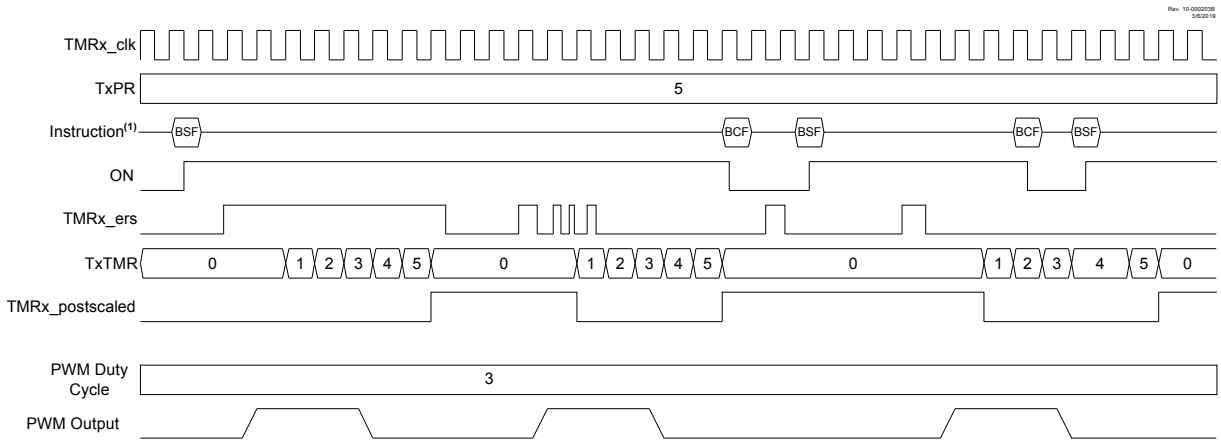
### 21.8.9 Edge-Triggered Monostable Modes

The Edge-Triggered Monostable modes start the timer on an edge from the external Reset signal input, after the ON bit is set, and stop incrementing the timer when the timer matches the TxPR period value. The following edges will start the timer:

- Rising edge (MODE = 'b10001)
- Falling edge (MODE = 'b10010)
- Rising or Falling edge (MODE = 'b10011)

When an Edge-Triggered Monostable mode is used in conjunction with the CCP PWM operation, the PWM drive goes active with the external Reset signal edge that starts the timer, but will not go active when the timer matches the TxPR value. While the timer is incrementing, additional edges on the external Reset signal will not affect the CCP PWM.

**Figure 21-11. Rising Edge-Triggered Monostable Mode Timing Diagram (MODE = 'b10001)**




Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 21.8.10 Level-Triggered Hardware Limit One-Shot Modes

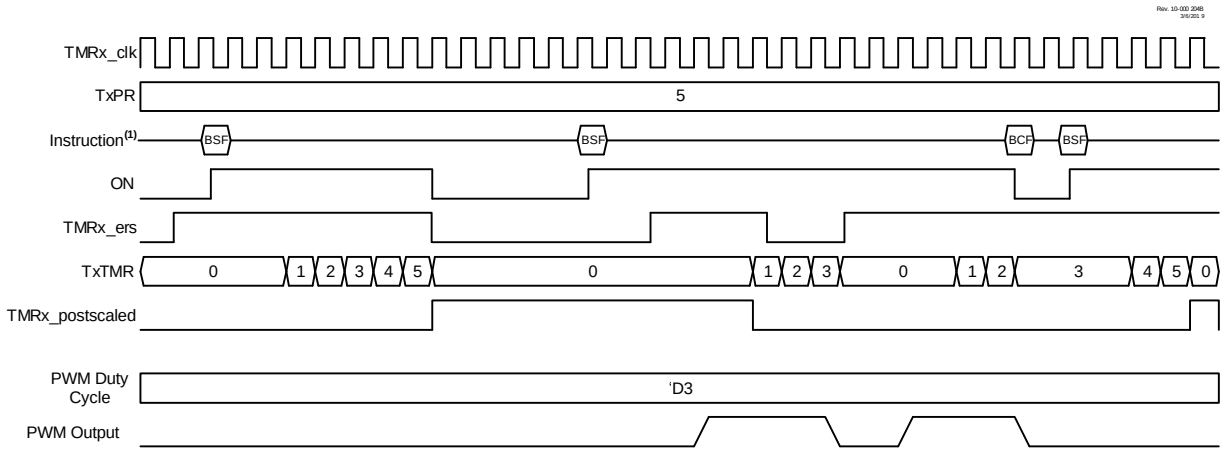
The Level-Triggered Hardware Limit One-Shot modes hold the timer in Reset on an external Reset level and start counting when both the ON bit is set and the external signal is not at the Reset level. If one of either the external signal is not in Reset or the ON bit is set, then the other signal being set/made active will start the timer. Reset levels are selected as follows:

- Low Reset level (MODE = 'b10110)
- High Reset level (MODE = 'b10111)

When the timer count matches the TxPR period count, the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a TxPR match or by software control, the timer will stay in Reset until both the ON bit is set and the external signal is not at the Reset level.

When Level-Triggered Hardware Limit One-Shot modes are used in conjunction with the CCP PWM operation, the PWM drive goes active with either the external signal edge or the setting of the ON bit, whichever of the two starts the timer.

**Figure 21-12. Level-Triggered hardware Limit One-Shot Mode Timing Diagram (MODE = 'b10110)**




Note 1: BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

## 21.9 Timer2 Operation During Sleep

When PSYNC = 1, Timer2 cannot be operated while the processor is in Sleep mode. The contents of the T2TMR and T2PR registers will remain unchanged while processor is in Sleep mode.

When PSYNC = 0, Timer2 will operate in Sleep as long as the clock source selected is also still running. If any internal oscillator is selected as the clock source, it will stay active during Sleep mode.

## 21.10 Register Definitions: Timer2 Control

Long bit name prefixes for the Timer2 peripherals are shown in table below. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 21-2. Timer2 Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| Timer2 | T2 |



**Notice:** References to module Timer2 apply to all the even numbered timers on this device. (Timer2, Timer4, etc.)

### 21.10.1 TxTMR

**Name:** TxTMR
**Address:** 0x28C

Timer Counter Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TxTMR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TxTMR[7:0]** Timerx Counter

### 21.10.2 TxPR

**Name:** TxPR
**Address:** 0x28D

Timer Period Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TxPR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:0 – TxPR[7:0]** Timer Period Register

| Value | Description |
|---|---|
| 0 – 255 | The timer restarts at '0' when TxTMR reaches TxPR value |

### 21.10.3 TxCON

**Name:** TxCON
**Address:** 0x28E

Timerx Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ON | CKPS[2:0] | | | OUTPS[3:0] | | | |
| Access | R/W/HC | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – ON** Timer On(1)

| Value | Description |
|---|---|
| 1 | Timer is on |
| 0 | Timer is off: all counters and state machines are reset |

**Bits 6:4 – CKPS[2:0]** Timer Clock Prescale Select

| Value | Description |
|---|---|
| 111 | 1:128 Prescaler |
| 110 | 1:64 Prescaler |
| 101 | 1:32 Prescaler |
| 100 | 1:16 Prescaler |
| 011 | 1:8 Prescaler |
| 010 | 1:4 Prescaler |
| 001 | 1:2 Prescaler |
| 000 | 1:1 Prescaler |

**Bits 3:0 – OUTPS[3:0]** Timer Output Postscaler Select

| Value | Description |
|---|---|
| 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 |

**Note:**

1. In certain modes, the ON bit will be auto-cleared by hardware. See Table 21-1.

### 21.10.4 TxHLT

**Name:** TxHLT
**Address:** 0x28F

Timer Hardware Limit Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – PSYNC** Timer Prescaler Synchronization Enable(1, 2)

| Value | Description |
|---|---|
| 1 | Timer Prescaler Output is synchronized to $F_{OSC}/4$ |
| 0 | Timer Prescaler Output is not synchronized to $F_{OSC}/4$ |

**Bit 6 – CPOL** Timer Clock Polarity Selection(3)

| Value | Description |
|---|---|
| 1 | Falling edge of input clock clocks timer/prescaler |
| 0 | Rising edge of input clock clocks timer/prescaler |

**Bit 5 – CSYNC** Timer Clock Synchronization Enable(4, 5)

| Value | Description |
|---|---|
| 1 | ON bit is synchronized to timer clock input |
| 0 | ON bit is not synchronized to timer clock input |

**Bits 4:0 – MODE[4:0]** Timer Control Mode Selection(6, 7)

| Value | Description |
|---|---|
| 00000 to 11111 | See Table 21-1 |

**Notes:**

1. Setting this bit ensures that reading TxTMR will return a valid data value.
2. When this bit is '1', Timer cannot operate in Sleep mode.
3. CKPOL should not be changed while ON = 1.
4. Setting this bit ensures glitch-free operation when the ON is enabled or disabled.
5. When this bit is set then the timer operation will be delayed by two input clocks after the ON bit is set.
6. Unless otherwise indicated, all modes start upon ON = 1 and stop upon ON = 0 (stops occur without affecting the value of TxTMR).
7. When TxTMR = TxPR, the next clock clears TxTMR, regardless of the operating mode.

### 21.10.5 TxCLKCON

**Name:** TxCLKCON
**Address:** 0x290

Timer Clock Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | CS[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – CS[2:0]** Timer Clock Source Selection

**Table 21-3. Clock Source Selection**

| CS | Clock Source |
|---|---|
| 111 | Reserved |
| 110 | MFINTOSC (32 kHz) |
| 101 | MFINTOSC (500 kHz) |
| 100 | LFINTOSC |
| 011 | HFINTOSC |
| 010 | $F_{OSC}$ |
| 001 | $F_{OSC}/4$ |
| 000 | Pin selected by T2INPPS |

### 21.10.6 TxRST

**Name:** TxRST
**Address:** 0x291

Timer External Reset Signal Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | RSEL[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – RSEL[3:0]** External Reset Source Selection

**Table 21-4. External Reset Sources**

| RSEL | Reset Source |
|---|---|
| 1111-0101 | Reserved |
| 0100 | PWM4_OUT |
| 0011 | PWM3_OUT |
| 0010 | CCP2_OUT |
| 0001 | CCP1_OUT |
| 0000 | Pin selected by T2INPPS |

## 21.11 Register Summary - Timer2

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x028B | Reserved | | | | | | | | | |
| 0x028C | TxTMR | 7:0 | T2TMR[7:0] | | | | | | | |
| 0x028D | TxPR | 7:0 | T2PR[7:0] | | | | | | | |
| 0x028E | TxCON | 7:0 | ON | CKPS[2:0] | | | OUTPS[3:0] | | | |
| 0x028F | TxHLT | 7:0 | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| 0x0290 | TxCLKCON | 7:0 | | | | | | CS[2:0] | | |
| 0x0291 | TxRST | 7:0 | | | | | RSEL[3:0] | | | |

# 22. CCP - Capture/Compare/PWM Module

The Capture/Compare/PWM module is a peripheral that allows the user to time and control different events, and to generate Pulse-Width Modulation (PWM) signals. In Capture mode, the peripheral allows the timing of the duration of an event. The Compare mode allows the user to trigger an external event when a predetermined amount of time has expired. The PWM mode can generate Pulse-Width Modulated signals of varying frequency and duty cycle.

The Capture and Compare functions are identical for all CCP modules.

**Important:** In devices with more than one CCP module, it is very important to pay close attention to the register names used. Throughout this section, the prefix 'CCPx' is used as a generic replacement for specific numbering. A number placed where the 'x' is in the prefix is used to distinguish between separate modules. For example, CCP1CON and CCP2CON control the same operational aspects of two completely different CCP modules.

## 22.1 CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (CCPxCON), a capture input selection register (CCPxCAP) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte).

### 22.1.1 CCP Modules and Timer Resources

The CCP modules utilize Timers 1 and 2 that vary with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in the table below.

**Table 22-1. CCP Mode - Timer Resources**

| CCP Mode | Timer Resource |
|---|---|
| Capture | Timer1 |
| Compare | |
| PWM | Timer2 |

All of the modules may be active at once and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time.

### 22.1.2 Open-Drain Output Option

When operating in Output mode (the Compare or PWM modes), the drivers for the CCPx pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor and allows the output to communicate with external circuits without the need for additional level shifters.

## 22.2 Capture Mode

Capture mode makes use of the 16-bit odd numbered timer resources (Timer1) . When an event occurs on the capture source, the 16-bit CCPRx register captures and stores the 16-bit value of the TMRx register. An event is defined as one of the following and is configured by the MODE bits:

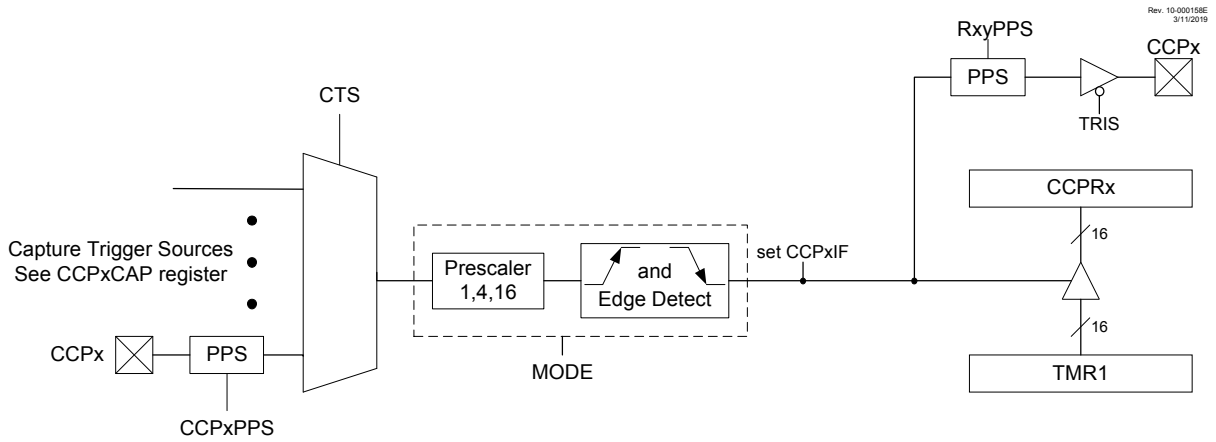
- Every falling edge of CCPx input
- Every rising edge of CCPx input
- Every 4$^{th}$ rising edge of CCPx input
- Every 16$^{th}$ rising edge of CCPx input

- Every edge of CCPx input (rising or falling)

When a capture is made, the Interrupt Request Flag bit CCPxIF of the PIRx register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRx register is read, the old captured value is overwritten by the new captured value. The following figure shows a simplified diagram of the capture operation.

**Important:** If an event occurs during a 2-byte read, the high and low-byte data will be from different events. It is recommended while reading the CCPRx register pair to either disable the module or read the register pair twice for data integrity.

**Figure 22-1. Capture Mode Operation Block Diagram**




### 22.2.1 Capture Sources

The capture source is selected with the CTS bits.

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

**Important:** If the CCPx pin is configured as an output, a write to the port can cause a capture condition.

### 22.2.2 Timer1 Mode for Capture

Timer1 must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

See the **"TMR1 - Timer1 Module with Gate Control"** chapter for more information on configuring Timer1.

### 22.2.3 Software Interrupt Mode

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE Interrupt Priority bit of the PIEx register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF Interrupt Flag bit of the PIRx register following any change in Operating mode.

**Important:** Clocking Timer1 from the system clock ($F_{OSC}$) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock ($F_{OSC}/4$) or from an external clock source.

### 22.2.4 CCP Prescaler

There are four prescaler settings specified by the MODE bits. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another does not clear the prescaler and may generate a false interrupt. To avoid this unexpected operation, turn the module off by clearing the CCPxCON register before changing the prescaler. The example below demonstrates the code to perform this function.

**Example 22-1. Changing Between Capture Prescalers**

```
BANKSEL CCP1CON
CLRF    CCP1CON         ;Turn CCP module off
MOVLW   NEW_CAPT_PS     ;CCP ON and Prescaler select → W
MOVWF   CCP1CON         ;Load CCP1CON with this value
```

### 22.2.5 Capture During Sleep

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock ($F_{OSC}/4$), or by an external clock source.

When Timer1 is clocked by $F_{OSC}/4$, Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep when Timer1 is clocked by an external clock source.

## 22.3 Compare Mode

The Compare mode function described in this section is available and identical for all CCP modules.

Compare mode makes use of the 16-bit odd numbered Timer resources (Timer1). The 16-bit value of the CCPRx register is constantly compared against the 16-bit value of the TMR1 register. When a match occurs, one of the following events can occur:

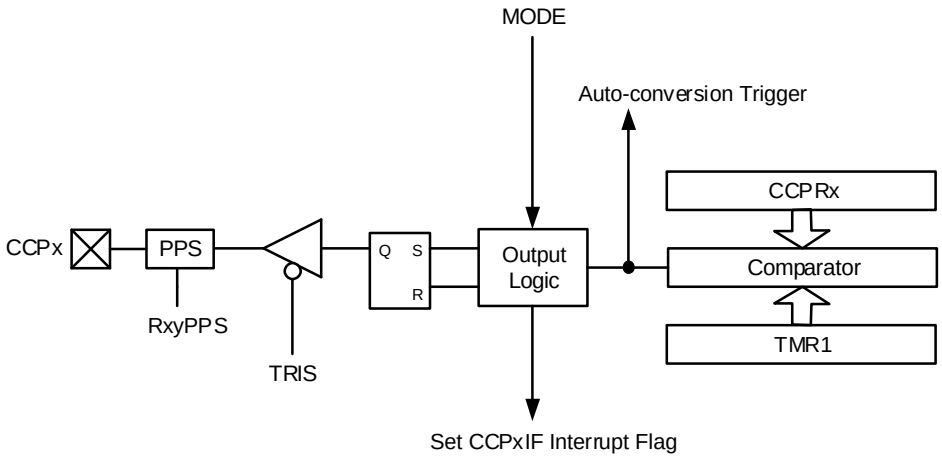
- Toggle the CCPx output and clear TMR1
- Toggle the CCPx output without clearing TMR1
- Set the CCPx output
- Clear the CCPx output
- Generate a Pulse output
- Generate a Pulse output and clear TMR1

The action on the pin is based on the value of the MODE control bits.

All Compare modes can generate an interrupt. When MODE = 'b0001 or 'b1011, the CCP resets the TMR1 register.

The following figure shows a simplified diagram of the compare operation.

**Figure 22-2. Compare Mode Operation Block Diagram**




### 22.3.1 CCPx Pin Configuration

The CCPx pin must be configured as an output in software by clearing the associated TRIS bit and defining the appropriate output pin through the RxyPPS registers. See the **"PPS - Peripheral Pin Select Module"** chapter for more details.

The CCP output can also be used as an input for other peripherals.

**Important:** Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

### 22.3.2 Timer1 Mode for Compare

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See the **"TMR1 - Timer1 Module with Gate Control"** chapter for more information on configuring Timer1.

**Important:** Clocking Timer1 from the system clock ($F_{OSC}$) should not be used in Compare mode. In order for Compare mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock ($F_{OSC}/4$) or from an external clock source.

### 22.3.3 Compare During Sleep

Since $F_{OSC}$ is shut down during Sleep mode, the Compare mode will not function properly during Sleep, unless the timer is running. The device will wake on interrupt (if enabled).

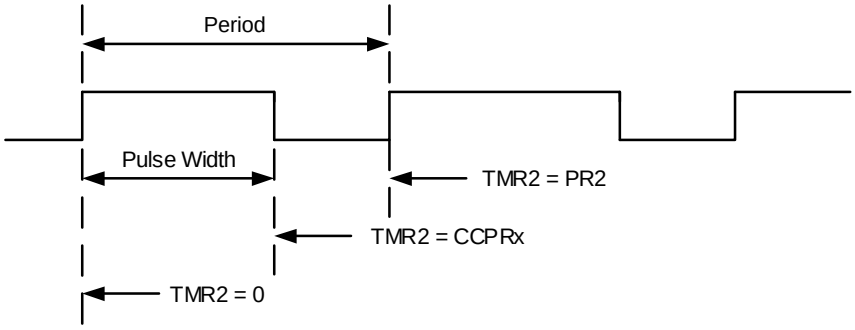
## 22.4 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that controls power to a load by switching quickly between fully ON and fully OFF states. The PWM signal resembles a square wave where the high portion of the signal is considered the ON state and the low portion of the signal is considered the OFF state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of ON and OFF time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse-width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the ON time to the OFF time and is expressed in percentages, where 0% is fully OFF and 100% is fully ON. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied. The figure below shows a typical waveform of the PWM signal.

**Figure 22-3. CCP PWM Output Signal**



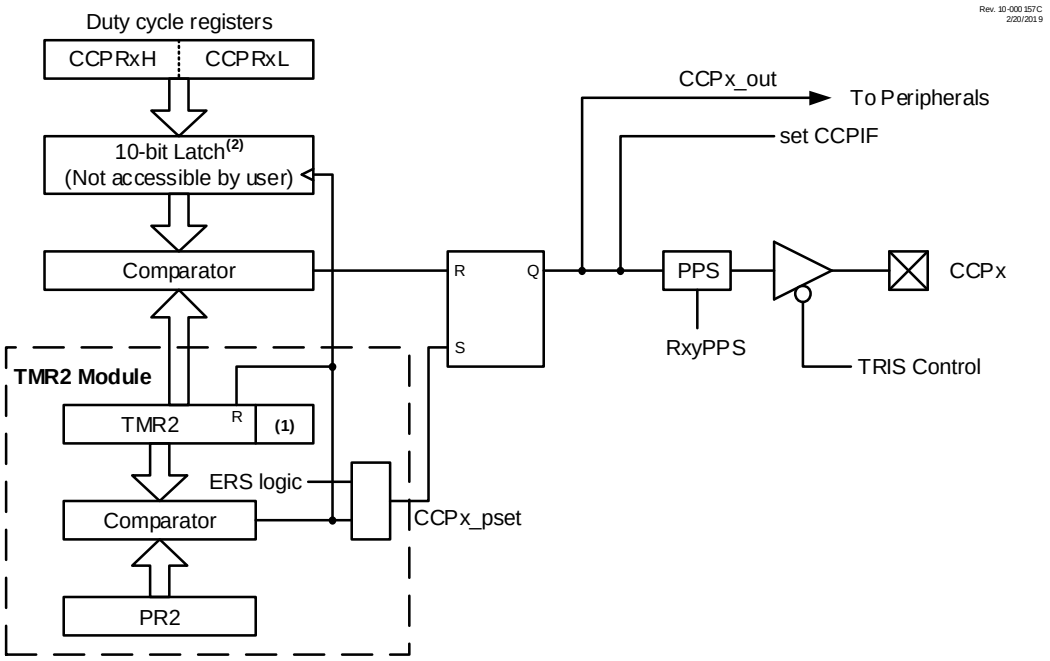

### 22.4.1 Standard PWM Operation

The standard PWM function described in this section is available and identical for all CCP modules. It generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to ten bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- Even numbered TxPR registers (T2PR)
- Even numbered TxCON registers (T2CON)
- 16-bit CCPRx registers
- CCPxCON registers

It is required to have $F_{OSC}/4$ as the clock input to T2TMR for correct PWM operation. The following figure shows a simplified block diagram of PWM operation.

**Figure 22-4. Simplified PWM Block Diagram**




Notes: 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.
2. The alignment of the 10 bits from the CCPR register is determined by the CCPxFMT bit.

**Important:** The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

### 22.4.2 Timer2 Timer Resource

The PWM standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

### 22.4.3 PWM Period

The PWM period is specified by the T2PR register of Timer2. The PWM period can be calculated using the formula in the equation below.

**Equation 22-1. PWM Period**

$$PWMPeriod = [(T2PR + 1)] \bullet 4 \bullet T_{OSC} \bullet (TMR2\text{Pr}escaleValue)$$

where $T_{OSC} = 1/F_{OSC}$

When T2TMR is equal to T2PR, the following three events occur on the next increment event:

- T2TMR is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is transferred from the CCPRx register into a 10-bit buffer.

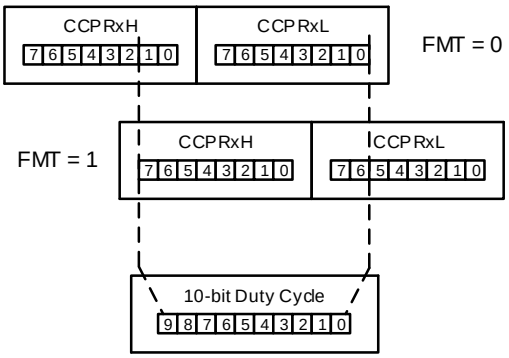
**Important:** The Timer postscaler (see **"Timer2 Interrupt"**) is not used in the determination of the PWM frequency.

### 22.4.4 PWM Duty Cycle

The PWM duty cycle is specified by writing a 10-bit value to the CCPRx register. The alignment of the 10-bit value is determined by the FMT bit (see Figure 22-5). The CCPRx register can be written to at any time. However, the duty cycle value is not latched into the 10-bit buffer until after a match between T2PR and T2TMR.

The equations below are used to calculate the PWM pulse width and the PWM duty cycle ratio.

**Figure 22-5. PWM 10-Bit Alignment**




**Equation 22-2. Pulse Width**

$$Pulse\ Width = (CCPRxH{:}CCPRxL\ register\ value) \bullet T_{OSC} \bullet (TMR2\ \mathrm{Prescale}\ Value)$$

**Equation 22-3. Duty Cycle**

$$DutyCycleRatio = \frac{(CCPRxH{:}CCPRxL\ register\ value)}{4(T2PR + 1)}$$

The CCPRx register is used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

The 8-bit timer T2TMR register is concatenated with either the 2-bit internal system clock ($F_{OSC}$), or two bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

When the 10-bit time base matches the CCPRx register, then the CCPx pin is cleared (see Figure 22-4).

### 22.4.5 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 T2PR is 0xFF. The resolution is a function of the T2PR register value, as shown below.

**Equation 22-4. PWM Resolution**

$$\mathrm{Resolution} = \frac{\log[4(T2PR + 1)]}{\log(2)} bits$$

**Important:** If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

**Table 22-2. Example PWM Frequencies and Resolutions ($F_{OSC}$ = 20 MHz)**

| PWM Frequency | 1.22 kHz | 4.88 kHz | 19.53 kHz | 78.12 kHz | 156.3 kHz | 208.3 kHz |
|---|---|---|---|---|---|---|
| Timer Prescale | 16 | 4 | 1 | 1 | 1 | 1 |
| T2PR 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 ($F_{OSC}$ = 8 MHz)**

| PWM Frequency | 1.22 kHz | 4.90 kHz | 19.61 kHz | 76.92 kHz | 153.85 kHz | 200.0 kHz |
|---|---|---|---|---|---|---|
| Timer Prescale | 16 | 4 | 1 | 1 | 1 | 1 |
| T2PR Value | 0x65 | 0x65 | 0x65 | 0x19 | 0x0C | 0x09 |
| Maximum Resolution (bits) | 8 | 8 | 8 | 6 | 5 | 5 |

### 22.4.6 Operation in Sleep Mode

In Sleep mode, the T2TMR register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, T2TMR will continue from the previous state.

### 22.4.7 Changes in System Clock Frequency

The PWM frequency is derived from the system clock frequency. Any changes in the system clock frequency will result in changes to the PWM frequency. See the **"OSC - Oscillator Module"** chapter for additional details.

### 22.4.8 Effects of Reset

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

### 22.4.9 Setup for PWM Operation

The following steps illustrate how to configure the CCP module for standard PWM operation:

1. Select the desired output pin with the RxyPPS control to select CCPx as the source. Disable the selected pin output driver by setting the associated TRIS bit. The output will be enabled later at the end of the PWM setup.
2. Load the timer period register T2PR with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRx register with the PWM duty cycle value and configure the FMT bit to set the proper register alignment.
5. Configure and start the Timer:
   - Clear the TMR2IF Interrupt Flag bit of the PIRx register. See Note below.
   - Select the timer clock source to be as $F_{OSC}/4$. This is required for correct operation of the PWM module.
   - Configure the CKPS bits of the T2CON register with the desired Timer prescale value.
   - Enable the Timer by setting the ON bit of the T2CON register.
6. Enable the PWM output:
   - Wait until the Timer overflows and the TMR2IF bit of the PIRx register is set. See Note below.
   - Enable the CCPx pin output driver by clearing the associated TRIS bit.



**Important:** In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.

## 22.5 Register Definitions: CCP Control

Long bit name prefixes for the CCP peripherals are shown in the following table. Refer to the "**Long Bit Names**" section in the "**Register and Bit Naming Conventions**" chapter for more information.

**Table 22-4. CCP Long bit name prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| CCP1 | CCP1 |
| CCP2 | CCP2 |

### 22.5.1 CCPxCON

**Name:** CCPxCON
**Address:** 0x30E,0x312

CCP Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OUT | FMT | MODE[3:0] | | | |
| Access | R/W | | R | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | x | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** CCP Module Enable

| Value | Description |
|---|---|
| 1 | CCP is enabled |
| 0 | CCP is disabled |

**Bit 5 – OUT** CCP Output Data (read-only)

**Bit 4 – FMT** CCPW (pulse-width) Value Alignment

| Value | Condition | Description |
|---|---|---|
| x | Capture mode | Not used |
| x | Compare mode | Not used |
| 1 | PWM mode | Left-aligned format |
| 0 | PWM mode | Right-aligned format |

**Bits 3:0 – MODE[3:0]** CCP Mode Select

**Table 22-5. CCPx Mode Select**

| MODE Value | Operating Mode | Operation | Set CCPxIF |
|---|---|---|---|
| 11xx | PWM | PWM operation | Yes |
| 1011 | Compare | Pulse output; clear TMR1(2) | Yes |
| 1010 | | Pulse output | Yes |
| 1001 | | Clear output(1) | Yes |
| 1000 | | Set output(1) | Yes |
| 0111 | Capture | Every 16th rising edge of CCPx input | Yes |
| 0110 | | Every 4th rising edge of CCPx input | Yes |
| 0101 | | Every rising edge of CCPx input | Yes |
| 0100 | | Every falling edge of CCPx input | Yes |
| 0011 | | Every edge of CCPx input | Yes |
| 0010 | Compare | Toggle output | Yes |
| 0001 | | Toggle output; clear TMR1(2) | Yes |
| 0000 | Disabled | | — |

**Notes:**

1. The set and clear operations of the Compare mode are reset by setting MODE = 'b0000 or EN = 0.
2. When MODE = 'b0001 or 'b1011, then the timer associated with the CCP module is cleared. TMR1 is the default selection for the CCP module, so it is used for indication purposes only.

### 22.5.2 CCPxCAP

**Name:** CCPxCAP
**Address:** 0x30F,0x313

Capture Trigger Input Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | CTS[1:0] | |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bits 1:0 – CTS[1:0]** Capture Trigger Input Selection

**Table 22-6. Capture Trigger Sources**

| CTS Value | Source |
|---|---|
| 11–10 | Reserved |
| 01 | IOC Interrupt |
| 00 | Pin selected by CCPxPPS |

### 22.5.3 CCPRx

**Name:** CCPRx
**Address:** 0x30C,0x310

Capture/Compare/Pulse Width Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | CCPR[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CCPR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 15:0 – CCPR[15:0]** Capture/Compare/Pulse Width
Reset States: POR/BOR = xxxxxxxxxxxxxxxx
All other Resets = uuuuuuuuuuuuuuuu

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- When MODE = Capture or Compare
  - CCPRxH: Accesses the high byte CCPR[15:8]
  - CCPRxL: Accesses the low byte CCPR[7:0]
- When MODE = PWM and FMT = 0
  - CCPRx[15:10]: Not used
  - CCPRxH[1:0]: Accesses the two Most Significant bits CCPR[9:8]
  - CCPRxL: Accesses the eight Least Significant bits CCPR[7:0]
- When MODE = PWM and FMT = 1
  - CCPRxH: Accesses the eight Most Significant bits CCPR[9:2]
  - CCPRxL[7:6]: Accesses the two Least Significant bits CCPR[1:0]
  - CCPRx[5:0]: Not used

## 22.6 Register Summary - CCP Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x030B | Reserved | | | | | | | | | |
| 0x030C | CCPR1 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x030E | CCP1CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x030F | CCP1CAP | 7:0 | | | | | | | CTS[1:0] | |
| 0x0310 | CCPR2 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x0312 | CCP2CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x0313 | CCP2CAP | 7:0 | | | | | | | CTS[1:0] | |

# 23. PWM - Pulse-Width Modulation

The PWM module generates a Pulse-Width Modulated signal determined by the duty cycle, period, and resolution that are configured by the following registers:

- T2PR
- T2CON
- PWMxDC
- PWMxCON

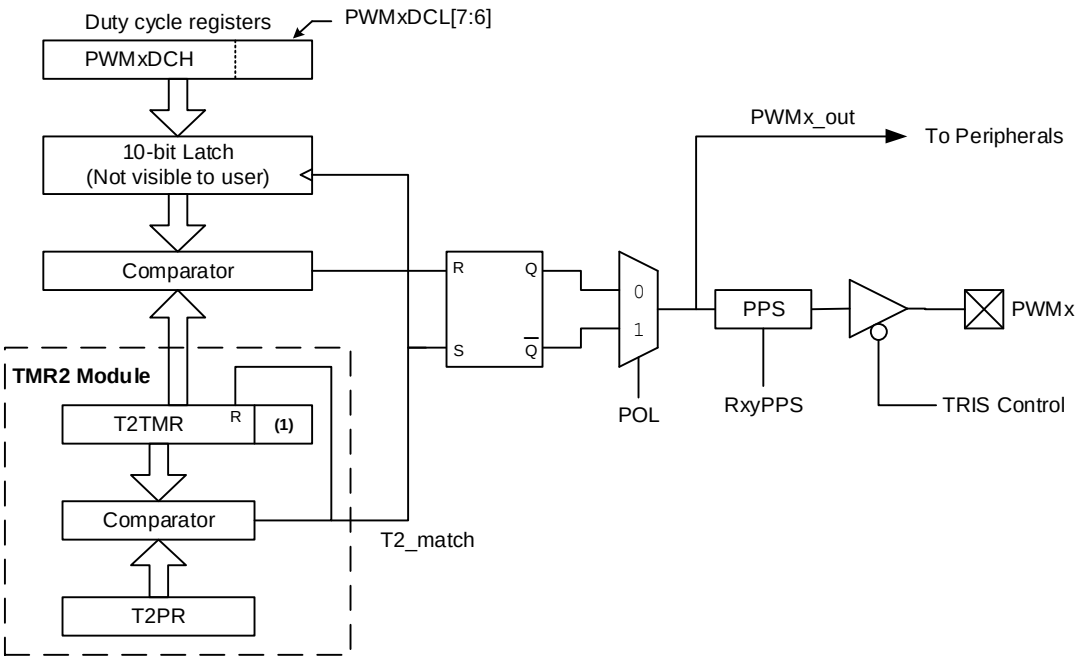
**Important:** The corresponding TRIS bit must be cleared to enable the PWM output on the PWMx pin.

Each PWM module uses the same timer source, Timer2, to control each module.

Figure 23-1 shows a simplified block diagram of PWM operation.

Figure 23-2 shows a typical waveform of the PWM signal.

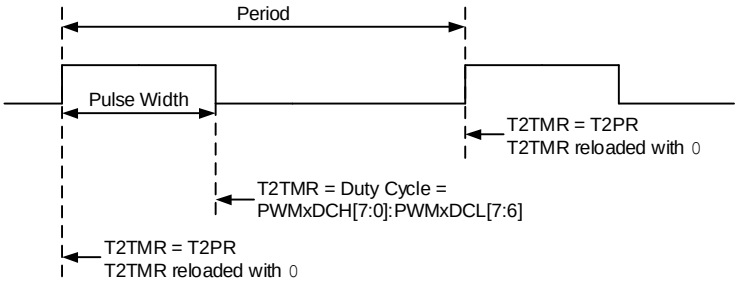
**Figure 23-1. Simplified PWM Block Diagram**




**Note:**

1. 8-bit timer is concatenated with two bits generated by $F_{OSC}$ or two bits of the internal prescaler to create 10-bit time base.

**Figure 23-2. PWM Output**




For a step-by-step procedure on how to set up this module for PWM operation, refer to "Setup for PWM Operation using PWMx Output Pins".

## 23.1 Fundamental Operation

The PWM module produces a 10-bit resolution output. The timer selection for PWMx is TMR2. T2TMR and T2PR 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.

**Important:** The Timer2 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 are set when T2TMR is cleared. Each PWMx is cleared when T2TMR 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 T2PR, the PWM output is never cleared (100% duty cycle).

**Important:** The PWMxDCH and PWMxDCL registers are double-buffered. The buffers are updated when T2TMR matches T2PR. Care should be taken to update both registers before the timer match occurs.

## 23.2 PWM Output Polarity

The output polarity is inverted by setting the POL bit.

## 23.3 PWM Period

The PWM period is specified by the T2PR register The PWM period can be calculated using the formula of Equation 23-1. It is required to have $F_{OSC}/4$ as the selected clock input to the timer for correct PWM operation.

**Equation 23-1. PWM Period**

$$PWMPeriod = [(T2PR) + 1] \bullet 4 \bullet Tosc \bullet (TMR2\ PrescaleValue)$$

**Note:** $T_{OSC} = 1/F_{OSC}$

When T2TMR is equal to T2PR, the following three events occur on the next increment cycle:

- T2TMR 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.

**Important:** The Timer2 postscaler has no effect on the PWM operation.

## 23.4 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.

The equations below are used to calculate the PWM pulse width and the PWM duty cycle ratio.

**Equation 23-2. Pulse Width**

$$PulseWidth = (PWMxDCH{:}PWMxDCL[7{:}6]) \bullet Tosc \bullet (TMR2\mathrm{P}rescaleValue)$$

**Note:** $T_{OSC} = 1/F_{OSC}$

**Equation 23-3. Duty Cycle Ratio**

$$DutyCycleRatio = \frac{(PWMxDCH{:}PWMxDCL[7{:}6])}{4(T2PR+1)}$$

The 8-bit timer T2TMR register is concatenated with the two Least Significant bits of $1/F_{OSC}$, 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.

## 23.5 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 T2PR is 255. The resolution is a function of the T2PR register value as shown below.

**Equation 23-4. PWM Resolution**

$$Resolution = \frac{\log[4(T2PR+1)]}{\log(2)} bits$$

**Important:** If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

**Table 23-1. Example PWM Frequencies and Resolutions ($F_{OSC}$ = 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 |
| T2PR Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.6 |

**Table 23-2. Example PWM Frequencies and Resolutions ($F_{OSC}$ = 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 |
| T2PR Value | 0x65 | 0x65 | 0x65 | 0x19 | 0x0C | 0x09 |
| Maximum Resolution (bits) | 8 | 8 | 8 | 6 | 5 | 5 |

## 23.6 Operation in Sleep Mode

In Sleep mode, the T2TMR 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, T2TMR will continue from its previous state.

## 23.7 Changes in System Clock Frequency

The PWM frequency is derived from the system clock frequency ($F_{OSC}$). Any changes in the system clock frequency will result in changes to the PWM frequency.

## 23.8 Effects of Reset

Any Reset will force all ports to Input mode and the PWM registers to their Reset states.

## 23.9 Setup for PWM Operation using PWMx Output 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 TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the T2PR register with the PWM period value.
4. Load the PWMxDCH register and bits [7:6] of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
   - Clear the TMR2IF Interrupt Flag bit of the PIRx register.(1)
   - Select the timer clock source to be as $F_{OSC}$/4 using the T2CLKCON register. This is required for correct operation of the PWM module.
   - Configure the CKPS bits of the T2CON register with the Timer2 prescale value.
   - Enable Timer2 by setting the ON bit of the T2CON register.
6. Enable PWM output pin and wait until Timer2 overflows, TMR2IF bit of the PIRx register is set.(2)
7. Enable the PWMx pin output driver(s) by clearing the associated TRIS bit(s) and setting the desired pin PPS control bits.
8. Configure the PWM module by loading the PWMxCON register with the appropriate values.

**Notes:**

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.

### 23.9.1 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 TRIS bits.

## 23.10 Setup for PWM Operation to Other Device Peripherals

The following steps should be taken when configuring the module for PWM operation to be used by other device peripherals:

1. Disable the PWMx pin output driver(s) by setting the associated TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the T2PR register with the PWM period value.
4. Load the PWMxDCH register and bits [7:6] of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
   - Clear the TMR2IF Interrupt Flag bit of the PIRx register.(1)
   - Select the timer clock source to be as $F_{OSC}/4$ using the T2CLKCON register. This is required for correct operation of the PWM module.
   - Configure the CKPS bits of the T2CON register with the Timer2 prescale value.
   - Enable Timer2 by setting the ON bit of the T2CON register.
6. Wait until Timer2 overflows, TMR2IF bit of the PIRx register is set.(1)
7. 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 included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then Step 6 may be ignored.

## 23.11 Register Definitions: PWM Control

Long bit name prefixes for the PWM peripherals are shown in the table below. Refer to the **"Long Bit Names Section"** for more information.

**Table 23-3. PWM Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| PWM3 | PWM3 |
| PWM4 | PWM4 |

### 23.11.1 PWMxCON

**Name:** PWMxCON
**Address:** 0x316,0x31A

PWM Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OUT | POL | | | | |
| Access | R/W | | R | R/W | | | | |
| Reset | 0 | | 0 | 0 | | | | |

**Bit 7 – EN** PWM Module Enable bit

| Value | Description |
|---|---|
| 1 | PWM module is enabled |
| 0 | PWM module is disabled |

**Bit 5 – OUT** PWM Module Output Level
Indicates PWM module output level when bit is read

**Bit 4 – POL** PWM Output Polarity Select bit

| Value | Description |
|---|---|
| 1 | PWM output is inverted |
| 0 | PWM output is normal |

### 23.11.2 PWMxDC

**Name:** PWMxDC
**Address:** 0x314,0x318

PWM Duty Cycle Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | DCH[7:0] | | | | | | | |
| Access | | | | | | | | |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DCL[1:0] | | | | | | | |
| Access | | | | | | | | |
| Reset | x | x | | | | | | |

**Bits 15:8 – DCH[7:0]** PWM Duty Cycle Most Significant bits
These bits are the MSbs of the PWM duty cycle.
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

**Bits 7:6 – DCL[1:0]** PWM Duty Cycle Least Significant bits
These bits are the LSbs of the PWM duty cycle.
Reset States: POR/BOR = xx
All Other Resets = uu

## 23.12 Register Summary - PWM

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x0313 | Reserved | | | | | | | | | |
| 0x0314 | PWM3DC | 7:0 | DCL[1:0] | | | | | | | |
| | | 15:8 | DCH[7:0] | | | | | | | |
| 0x0316 | PWM3CON | 7:0 | EN | | OUT | POL | | | | |
| 0x0317 | Reserved | | | | | | | | | |
| 0x0318 | PWM4DC | 7:0 | DCL[1:0] | | | | | | | |
| | | 15:8 | DCH[7:0] | | | | | | | |
| 0x031A | PWM4CON | 7:0 | EN | | OUT | POL | | | | |

# 24. EUSART - Enhanced Universal Synchronous Asynchronous Receiver Transmitter

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer independent of device program execution. The EUSART, also known as a Serial Communications Interface (SCI), can be configured as a full-duplex asynchronous system or half-duplex synchronous system. Full-Duplex mode is useful for communications with peripheral systems, such as CRT terminals and personal computers. Half-Duplex Synchronous mode is intended for communications with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs or other microcontrollers. These devices typically do not have internal clocks for baud rate generation and require the external clock signal provided by a master synchronous device.

The EUSART module includes the following capabilities:

- Full-Duplex Asynchronous Transmit and Receive
- Two-Character Input Buffer
- One-Character Output Buffer
- Programmable 8-Bit or 9-Bit Character Length
- Address Detection in 9-bit Mode
- Input Buffer Overrun Error Detection
- Received Character Framing Error Detection
- Half-Duplex Synchronous Master
- Half-Duplex Synchronous Slave
- Programmable Clock Polarity in Synchronous Modes
- Sleep Operation

The EUSART module implements the following additional features, making it ideally suited for use in Local Interconnect Network (LIN) bus systems:

- Automatic detection and calibration of the baud rate
- Wake-up on Break reception
- 13-bit Break character transmit

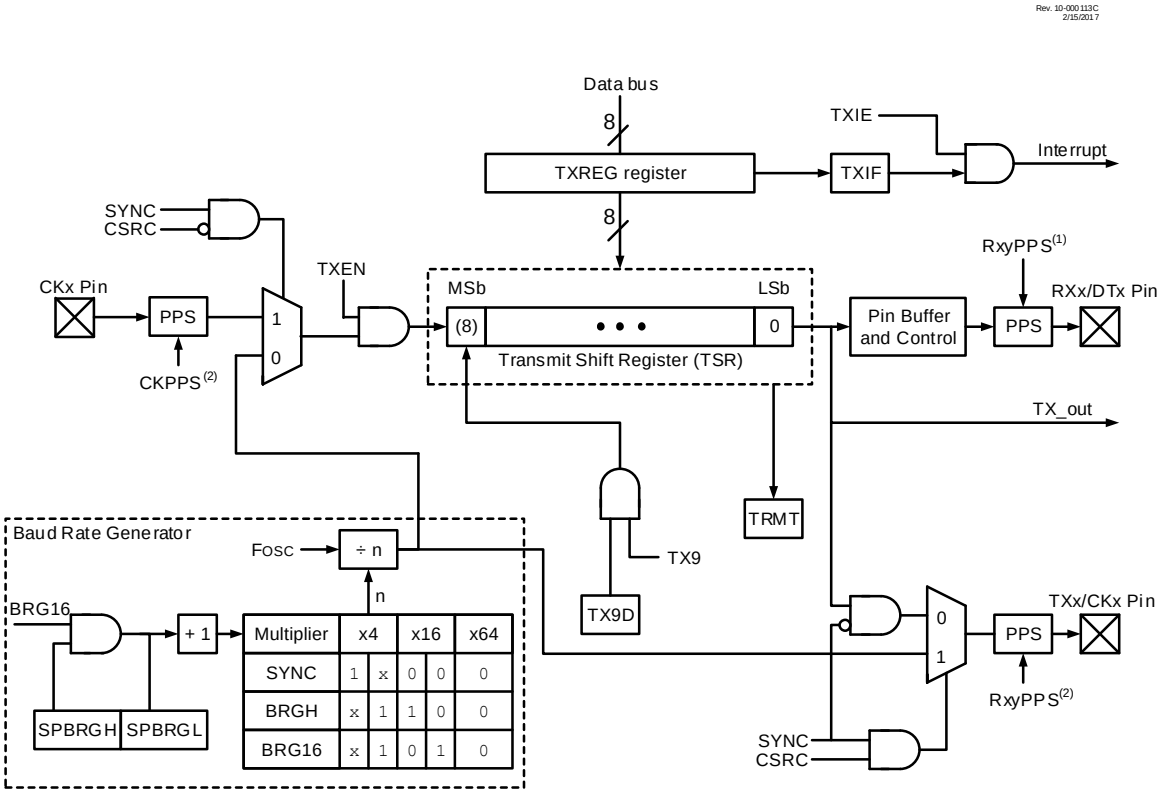
Block diagrams of the EUSART transmitter and receiver are shown in Figure 24-1 and Figure 24-2.

The operation of the EUSART module consists of six registers:

- Transmit Status and Control (TXxSTA)
- Receive Status and Control (RCxSTA)
- Baud Rate Control (BAUDxCON)
- Baud Rate Value (SPxBRG)
- Receive Data Register (RCxREG)
- Transmit Data Register (TXxREG)

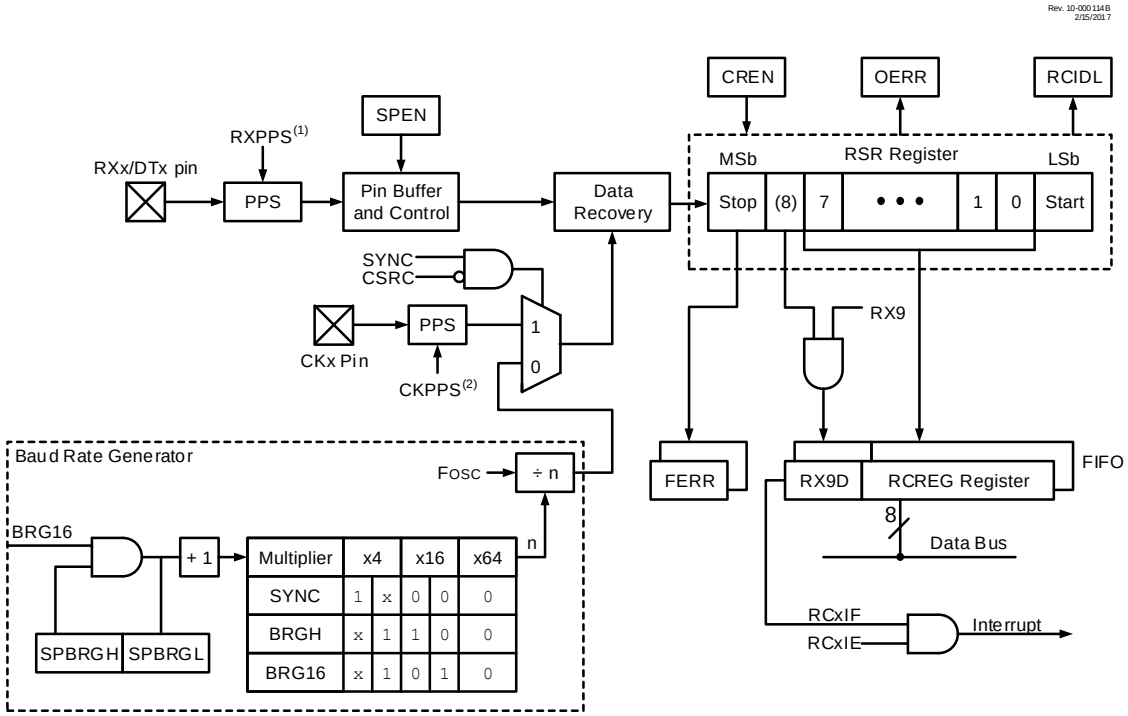
The RXx/DTx and TXx/CKx input pins are selected with the RXxPPS and TXxPPS registers, respectively. TXx, CKx, and DTx output pins are selected with each pin's RxyPPS register. Since the RX input is coupled with the DT output in Synchronous mode, it is the user's responsibility to select the same pin for both of these functions when operating in Synchronous mode. The EUSART control logic will control the data direction drivers automatically.

**Figure 24-1. EUSART Transmit Block Diagram**




**Note 1:** In Synchronous mode, the DT output and RX input PPS selections should enable the same pin.
**2:** In Master Synchronous mode the TX output and CK input PPS selections should enable the same pin.

**Figure 24-2. EUSART Receive Block Diagram**




**Note 1:** In Synchronous mode, the DT output and RX input PPS selections should enable the same pin.
**2:** In Master Synchronous mode the TX output and CK input PPS selections should enable the same pin.

## 24.1 EUSART Asynchronous Mode

The EUSART transmits and receives data using the standard non-return-to-zero (NRZ) format. NRZ is implemented with two levels: a $V_{OH}$ Mark state which represents a '1' data bit, and a $V_{OL}$ Space state which represents a '0' data bit. NRZ refers to the fact that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port idles in the Mark state. Each character transmission consists of one Start bit followed by eight or nine data bits and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is 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 24-2 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.

### 24.1.1 EUSART Asynchronous Transmitter

Figure 24-1 is a simplified representation of the transmitter. 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 TXxREG register.

#### 24.1.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- The Transmit Enable (TXEN) bit is set to '1' to enable the transmitter circuitry of the EUSART
- The EUSART Mode Select (SYNC) bit is set to '0' to configure the EUSART for asynchronous operation

- The Serial Port Enable (SPEN) bit is set to '1' to enable the EUSART interface and to enable automatically the output drivers for the RxyPPS selected as the TXx/CKx output

All other EUSART control bits are assumed to be in their default state.

If the TXx/CKx pin is shared with an analog peripheral, the analog I/O function must be disabled by clearing the corresponding ANSEL bit.

**Important:** The TXxIF Transmitter Interrupt Flag in the PIRx register is set when the TXEN enable bit is set and the Transmit Shift register (TSR) is Idle.

#### 24.1.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXxREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXxREG 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 TXxREG until the Stop bit of the previous character has been transmitted. The pending character in the TXxREG is then transferred to the TSR in one $T_{CY}$ immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXxREG.

#### 24.1.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the Clock/Transmit Polarity Select (SCKP) bit. 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 the "Clock Polarity" section for more details.

#### 24.1.1.4 Transmit Interrupt Flag

The EUSART Transmit Interrupt Flag (TXxIF) bit of the PIRx register is set whenever the EUSART transmitter is enabled and no character is being held for transmission in the TXxREG. In other words, the TXxIF bit is only cleared when the TSR is busy with a character and a new character has been queued for transmission in the TXxREG. The TXxIF flag bit is not cleared immediately upon writing TXxREG. TXxIF becomes valid in the second instruction cycle following the write execution. Polling TXxIF immediately following the TXxREG write will return invalid results. The TXxIF bit is read-only, it cannot be set or cleared by software.

The TXxIF interrupt can be enabled by setting the EUSART Transmit Interrupt Enable (TXxIE) bit of the PIEx register. However, the TXxIF flag bit will be set whenever the TXxREG is empty, regardless of the state of TXxIE enable bit.

To use interrupts when transmitting data, set the TXxIE bit only when there is more data to send. Clear the TXxIE interrupt enable bit upon writing the last character of the transmission to the TXxREG.

#### 24.1.1.5 TSR Status

The Transmit Shift Register Status (TRMT) bit 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 TXxREG. The TRMT bit remains clear until all bits have been shifted out of the TSR register. No interrupt logic is tied to this bit, so the user needs to poll this bit to determine the TSR status.

**Important:** The TSR register is not mapped in data memory, so it is not available to the user.

#### 24.1.1.6 Transmitting 9-Bit Characters

The EUSART supports 9-bit character transmissions. When the 9-Bit Transmit Enable (TX9) bit is set, the EUSART will shift nine bits out for each character transmitted. The TX9D bit 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 TXxREG. All nine bits of data will be transferred to the TSR shift register immediately after the TXxREG is written.

A special 9-bit Address mode is available for use with multiple receivers. See the "Address Detection" section for more information on the Address mode.

### 24.1.1.7 Asynchronous Transmission Setup

1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see section "EUSART Baud Rate Generator (BRG)").
2. Select the transmit output pin by writing the appropriate value to the RxyPPS register.
3. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
4. If 9-bit transmission is desired, set the TX9 control bit. That will indicate that the eight Least Significant data bits are an address when the receiver is set for address detection.
5. Set SCKP bit if inverted transmit is desired.
6. Enable the transmission by setting the TXEN control bit. This will cause the TXxIF interrupt bit to be set.
7. If interrupts are desired, set the TXxIE interrupt enable bit of the PIEx register.
8. An interrupt will occur immediately provided that the GIE and PEIE bits of the INTCON register are also set.
9. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
10. Load 8-bit data into the TXxREG register. This will start the transmission.

**Figure 24-3. Asynchronous Transmission**

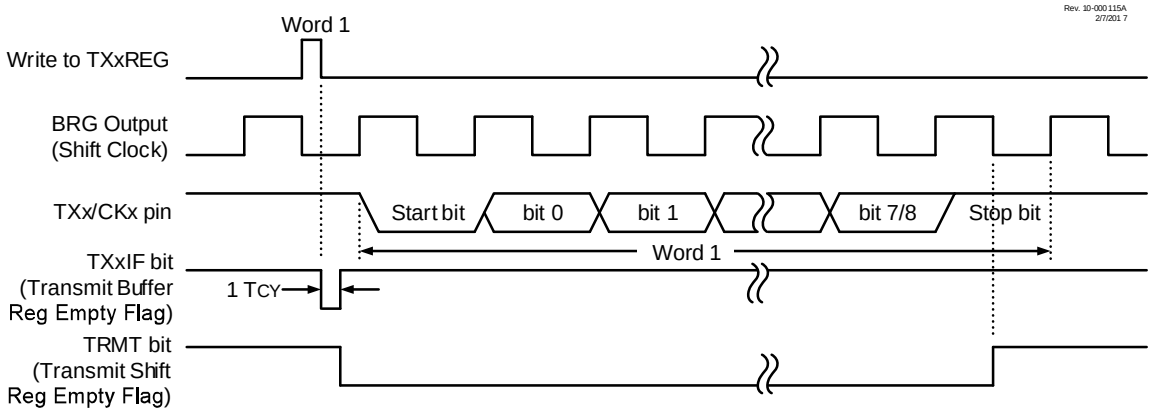
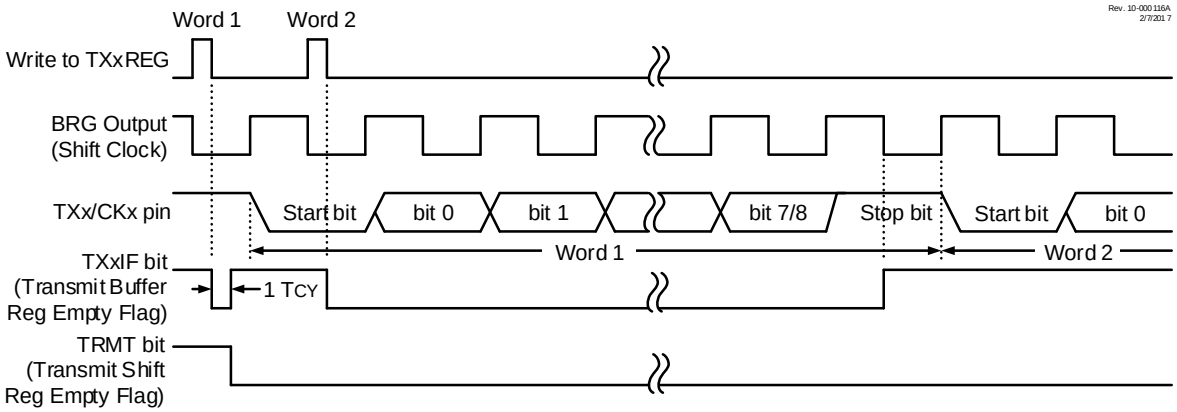



**Figure 24-4. Asynchronous Transmission (Back-to-Back)**




## 24.1.2 EUSART Asynchronous Receiver

The Asynchronous mode is typically used in RS-232 systems. A simplified representation of the receiver is shown in Figure 24-2. The data is received on the RXx/DTx 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 RCxREG register.

#### 24.1.2.1 Enabling the Receiver

The EUSART receiver is enabled for asynchronous operation by configuring the following three control bits:

- The Continuous Receive Enable (CREN) bit is set to '1' to enables the receiver circuitry of the EUSART
- The EUSART Mode Select (SYNC) bit is set to '0' to configure the EUSART for asynchronous operation
- The Serial Port Enable (SPEN) bit is set to '1' to enable the EUSART interface

All other EUSART control bits are assumed to be in their default state.

The user must set the RXxPPS register to select the RXx/DTx I/O pin and set the corresponding TRIS bit to configure the pin as an input.

**Important:** If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

#### 24.1.2.2 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 the "Receive Framing Error" section 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 EUSART Receive Interrupt Flag (RCxIF) bit of the PIRx register is set. The top character in the FIFO is transferred out of the FIFO by reading the RCxREG register.

**Important:** If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See the "Receive Overrrun Error" section for more information.

#### 24.1.2.3 Receive Interrupts

The EUSART Receive Interrupt Flag (RCxIF) bit of the PIRx register is set whenever the EUSART receiver is enabled and there is an unread character in the receive FIFO. The RCxIF Interrupt Flag bit is read-only, it cannot be set or cleared by software.

RCxIF interrupts are enabled by setting all of the following bits:

- RCxIE, Interrupt Enable bit of the PIEx register
- PEIE, Peripheral Interrupt Enable bit of the INTCON register
- GIE, Global Interrupt Enable bit of the INTCON register

The RCxIF Interrupt Flag bit will be set when there is an unread character in the FIFO, regardless of the state of interrupt enable bits.

#### 24.1.2.4 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 Framing Error (FERR) bit. 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 RCxREG.

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, which resets the EUSART. Clearing the CREN bit does not affect the FERR bit. A framing error by itself does not generate an interrupt.

**Important:** If all receive characters in the receive FIFO have framing errors, repeated reads of the RCxREG will not clear the FERR bit.

#### 24.1.2.5 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 Overrun Error (OERR) bit 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 or by resetting the EUSART by clearing the SPEN bit.

#### 24.1.2.6 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the 9-Bit Receive Enable (RX9) bit is set, the EUSART will shift nine bits into the RSR for each character received. The RX9D bit 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 RCxREG.

#### 24.1.2.7 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 Address Detect Enable (ADDEN) bit.

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 RCxIF 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.

#### 24.1.2.8 Asynchronous Reception Setup

1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see the “EUSART Baud Rate Generator (BRG)” section).
2. Set the RXxPPS register to select the RXx/DTx input pin.
3. Clear the ANSEL bit for the RXx pin (if applicable).
4. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
5. If interrupts are desired, set the RCxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
6. If 9-bit reception is desired, set the RX9 bit.
7. Enable reception by setting the CREN bit.
8. The RCxIF 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 RCxIE interrupt enable bit was also set.
9. Read the RCxSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
10. Get the received eight Least Significant data bits from the receive buffer by reading the RCxREG register.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

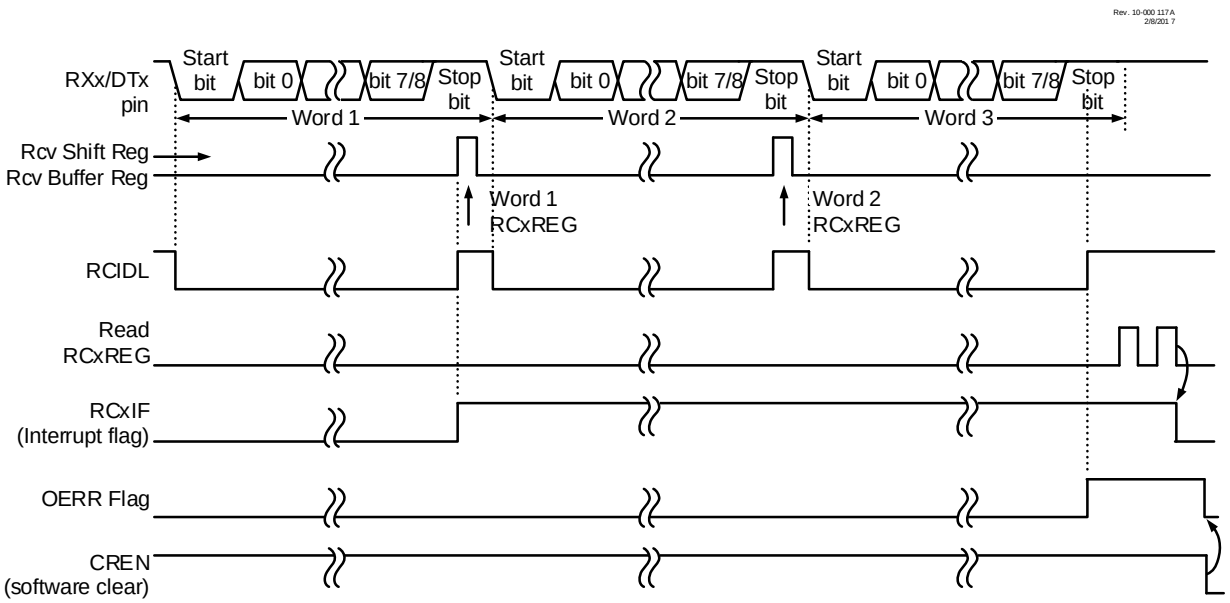
#### 24.1.2.9 9-Bit Address Detection Mode Setup

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable follow these steps:

1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see the “EUSART Baud Rate Generator (BRG)” section).

2. Set the RXxPPS register to select the RXx input pin.
3. Clear the ANSEL bit for the RXx pin (if applicable).
4. Enable the serial port by setting the SPEN bit. The SYNC bit must be cleared for asynchronous operation.
5. If interrupts are desired, set the RCxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
6. Enable 9-bit reception by setting the RX9 bit.
7. Enable address detection by setting the ADDEN bit.
8. Enable reception by setting the CREN bit.
9. The RCxIF 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 RCxIE Interrupt Enable bit is also set.
10. Read the RCxSTA register to get the error flags. The ninth data bit will always be set.
11. Get the received eight Least Significant data bits from the receive buffer by reading the RCxREG register. Software determines if this is the device's address.
12. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
13. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

**Figure 24-5. Asynchronous Reception**




**Note:** This timing diagram shows three bytes appearing on the RXx input. The OERR flag is set because the RCxREG is not read before the third word is received.

## 24.2 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (INTOSC). However, the INTOSC frequency may drift as $V_{DD}$ or temperature changes, and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the INTOSC output. Adjusting the value in the OSCTUNE register allows for fine resolution changes to the system clock source.

The other method adjusts the value in the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see "Auto-Baud Detect"). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change in the peripheral clock frequency.

## 24.3 EUSART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is an 8-bit or 16-bit timer that is dedicated to the support of both the asynchronous and synchronous EUSART operation. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit selects 16-bit mode.

The SPxBRGH:SPxBRGL 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 and the BRG16 bits. In Synchronous mode, the BRGH bit is ignored.

The Baud Rate Formulas table below contains the formulas for determining the baud rate. The Calculating Baud Rate Error equation below 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 and are shown in the table below. 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. The BRGH bit is used to achieve very high baud rates.

Writing a new value to the SPxBRGH:SPxBRGL 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 Receive Idle Flag (RCIDL) bit to make sure that the receive operation is idle before changing the system clock.

**Equation 24-1. Calculating Baud Rate Error**

For a device with $F_{OSC}$ of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$DesiredBaudrate = \frac{F_{OSC}}{64 \times (SPxBRG + 1)}$$

Solving for SPxBRG:

$$SPxBRG = \frac{F_{OSC}}{64 \times DesiredBaudrate} - 1$$

$$SPxBRG = \frac{16000000}{64 \times 9600} - 1$$

$$SPxBRG = 25.042 \simeq 25$$

$$CalculatedBaudrate = \frac{16000000}{64 \times (25 + 1)}$$

$$CalculatedBaudrate = 9615$$

$$Error = \frac{CalculatedBaudrate - DesiredBaudrate}{DesiredBaudrate}$$

$$Error = \frac{9615 - 9600}{9600}$$

$$Error = 0.16\,\%$$

**Table 24-1. Baud Rate Formulas**

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|---|---|---|---|---|
| SYNC | BRG16 | BRGH | | |
| 0 | 0 | 0 | 8-bit/Asynchronous | $F_{OSC}/[64\ (n+1)]$ |
| 0 | 0 | 1 | 8-bit/Asynchronous | $F_{OSC}/[16\ (n+1)]$ |
| 0 | 1 | 0 | 16-bit/Asynchronous | |

.....continued

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|---|---|---|---|---|
| SYNC | BRG16 | BRGH | | |
| 0 | 1 | 1 | 16-bit/Asynchronous | $F_{OSC}/[4\ (n+1)]$ |
| 1 | 0 | x | 8-bit/Synchronous | |
| 1 | 1 | x | 16-bit/Synchronous | |

**Note:** x = Don't care, n = value of SPxBRGH:SPxBRGL register pair.

**Table 24-2. Sample Baud Rates for Asynchronous Modes**

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_{OSC}$ = 32.000 MHz | | | $F_{OSC}$ = 20.000 MHz | | | $F_{OSC}$ = 18.432 MHz | | | $F_{OSC}$ = 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 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_{OSC}$ = 8.000 MHz | | | $F_{OSC}$ = 4.000 MHz | | | $F_{OSC}$ = 3.6864 MHz | | | $F_{OSC}$ = 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 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_{OSC}$ = 32.000 MHz | | | $F_{OSC}$ = 20.000 MHz | | | $F_{OSC}$ = 18.432 MHz | | | $F_{OSC}$ = 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 |

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_{OSC}$ = 8.000 MHz | | | $F_{OSC}$ = 4.000 MHz | | | $F_{OSC}$ = 3.6864 MHz | | | $F_{OSC}$ = 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 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_{OSC}$ = 32.000 MHz | | | $F_{OSC}$ = 20.000 MHz | | | $F_{OSC}$ = 18.432 MHz | | | $F_{OSC}$ = 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 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_{OSC}$ = 8.000 MHz | | | $F_{OSC}$ = 4.000 MHz | | | $F_{OSC}$ = 3.6864 MHz | | | $F_{OSC}$ = 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 | — | — | — |

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_{OSC}$ = 32.000 MHz | | | $F_{OSC}$ = 20.000 MHz | | | $F_{OSC}$ = 18.432 MHz | | | $F_{OSC}$ = 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 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $F_{OSC}$ = 8.000 MHz | | | $F_{OSC}$ = 4.000 MHz | | | $F_{OSC}$ = 3.6864 MHz | | | $F_{OSC}$ = 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 | — | — | — |

### 24.3.1 Auto-Baud Detect

The EUSART module supports automatic detection and calibration of the baud rate.

In the Auto-Baud Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. The Baud Rate Generator is used to time the period of a received 55h (ASCII “U”) which is the Sync character for the LIN bus. The unique feature of this character is that it has five rising edges including the Stop bit edge.

Setting the Auto-Baud Detect Enable (ABDEN) bit starts the auto-baud calibration sequence. 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 SPxBRG begins counting up using the BRG counter clock as shown in Figure 24-6. The fifth rising edge will occur on the RXx pin at the end of the eighth bit period. At that time, an accumulated value totaling the proper BRG period is left in the SPxBRGH:SPxBRGL register pair, the ABDEN bit is automatically cleared and the RCxIF interrupt flag is set. The value in the RCxREG needs to be read to clear the RCxIF interrupt. RCxREG content should be discarded. When calibrating for modes that do not use the SPxBRGH register the user can verify that the SPxBRGL register did not overflow by checking for 00h in the SPxBRGH register.

The BRG auto-baud clock is determined by the BRG16 and BRGH bits as shown in Table 24-3. During ABD, both the SPxBRGH and SPxBRGL registers are used as a 16-bit counter, independent of the BRG16 bit setting. While calibrating the baud rate period, the SPxBRGH and SPxBRGL registers are clocked at 1/8$^{th}$ the BRG base clock rate. The resulting byte measurement is the average bit time when clocked at full speed.

**Notes:**

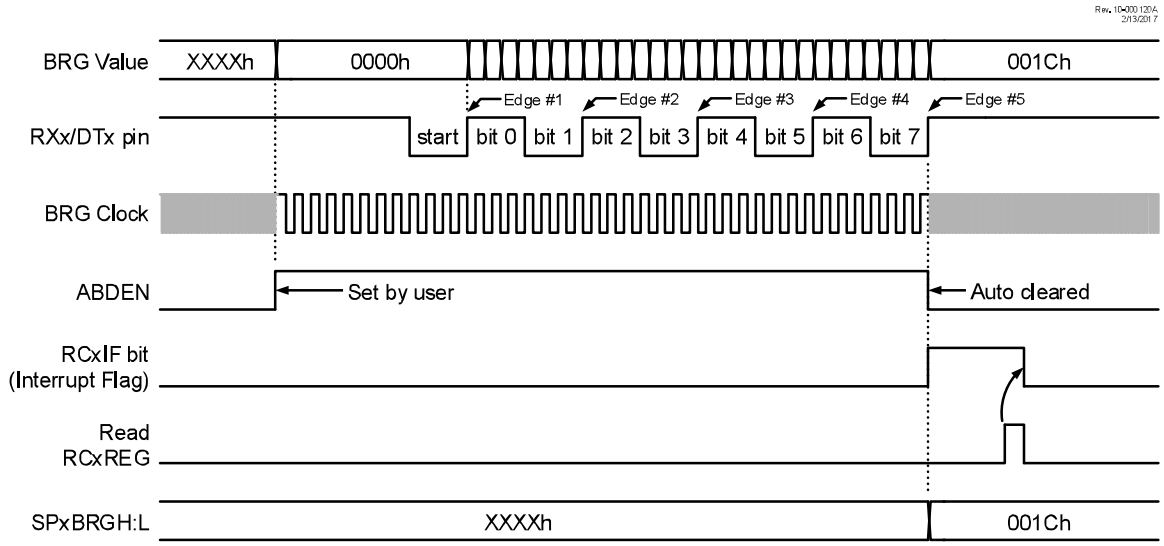
1. If the Wake-up Enable (WUE) bit is set with the ABDEN bit, auto-baud detection will occur on the byte following the Break character (see “Auto-Wake-up on Break”).
2. It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible.
3. During the auto-baud process, the auto-baud counter starts counting at one. Upon completion of the auto-baud sequence, to achieve maximum accuracy, subtract 1 from the SPxBRGH:SPxBRGL register pair.

**Table 24-3. BRG Counter Clock Rates**

| BRG16 | BRGH | BRG Base Clock | BRG ABD Clock |
|---|---|---|---|
| 1 | 1 | $F_{OSC}/4$ | $F_{OSC}/32$ |
| 1 | 0 | $F_{OSC}/16$ | $F_{OSC}/128$ |
| 0 | 1 | $F_{OSC}/16$ | $F_{OSC}/128$ |
| 0 | 0 | $F_{OSC}/64$ | $F_{OSC}/512$ |

**Note:** During the ABD sequence, SPxBRGL and SPxBRGH registers are both used as a 16-bit counter, independent of the BRG16 setting.

**Figure 24-6. Automatic Baud Rate Calibration**




### 24.3.2 Auto-Baud Overflow

During the course of automatic baud detection, the Auto-Baud Detect Overflow (ABDOVF) bit will be set if the baud rate counter overflows before the fifth rising edge is detected on the RXx 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. After the ABDOVF bit has been set, the counter continues to count until the fifth rising edge is detected on the RXx pin. Upon detecting the fifth RX edge, the hardware will set the RCxIF interrupt flag and clear the ABDEN bit. The RCxIF flag can be subsequently cleared by reading the RCxREG register. The ABDOVF bit can be cleared by software directly.

To terminate the auto-baud process before the RCxIF flag is set, clear the ABDEN bit then clear the ABDOVF bit. The ABDOVF bit will remain set if the ABDEN bit is not cleared first.

### 24.3.3 Auto-Wake-up on Break

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the Baud Rate Generator is inactive and a proper character reception cannot be performed. The Auto-Wake-up feature allows the controller to wake-up due to activity on the RX/DT line. This feature is available only in Asynchronous mode.

The Auto-Wake-up feature is enabled by setting the WUE bit. 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 RCxIF interrupt coincident with the wake-up event. The interrupt is generated synchronously to the Q clocks in normal CPU operating modes as shown in Figure 24-7, and asynchronously if the device is in Sleep mode, as shown in Figure 24-8. The interrupt condition is cleared by reading the RCxREG 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.

#### 24.3.3.1 Special Considerations

Break Character

To avoid character errors or character fragments during a wake-up event, the wake-up character must be all zeros.

When the wake-up is enabled the function works independent of the low time on the data stream. If the WUE bit is set and a valid non-zero character is received, the low time from the Start bit to the first rising edge will be interpreted as the wake-up event. The remaining bits in the character will be received as a fragmented character and subsequent characters can result in framing or overrun errors.

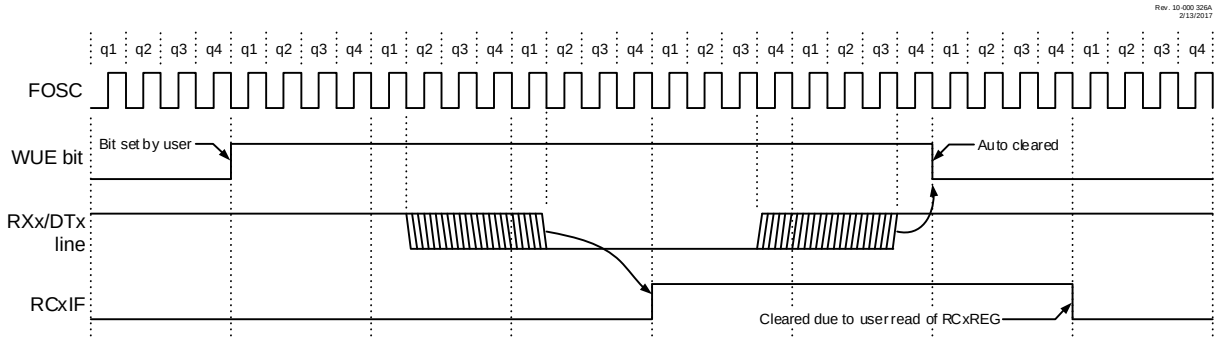
Therefore, the initial character in the transmission must be all '0's. This must be ten or more bit times, 13-bit times recommended for LIN bus, or any number of bit times for standard RS-232 devices.

WUE Bit

The wake-up event causes a receive interrupt by setting the RCxIF 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 RCxREG register and discarding its contents.

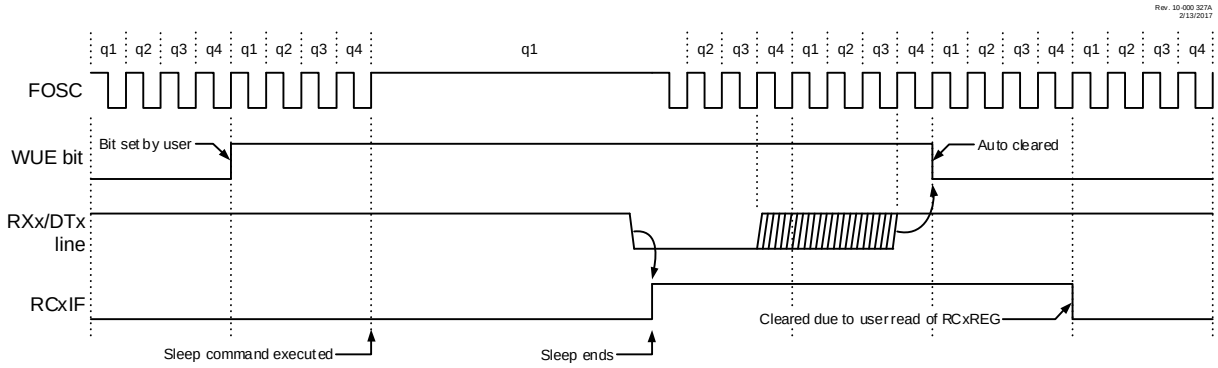
To ensure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process before setting the WUE bit. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**Figure 24-7. Auto-Wake-up Bit (WUE) Timing During Normal Operation**




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

**Figure 24-8. Auto-Wake-up Bit (WUE) Timings During Sleep**




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

### 24.3.4 Break Character Sequence

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. A Break character consists of a Start bit, followed by 12 '0' bits and a Stop bit.

To send a Break character, set the Send Break Character (SENDB) and Transmit Enable (TXEN) bits. The Break character transmission is then initiated by a write to the TXxREG. The value of data written to TXxREG 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 Transmit Shift Register Status (TRMT) bit indicates when the transmit operation is active or idle, just as it does during normal transmission. See Figure 24-9 for more details.

#### 24.3.4.1 Break and Sync Transmit Sequence

The following sequence will start a message frame header made up of a Break, followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to enable the Break sequence.
3. Load the TXxREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXxREG 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 TXxREG becomes empty, as indicated by the TXxIF, the next data byte can be written to TXxREG.

### 24.3.5 Receiving a Break Character

The EUSART module can receive a Break character in two ways.

The first method to detect a Break character uses the Framing Error (FERR) bit and the received data as indicated by RCxREG. The Baud Rate Generator is assumed to have been initialized to the expected baud rate.

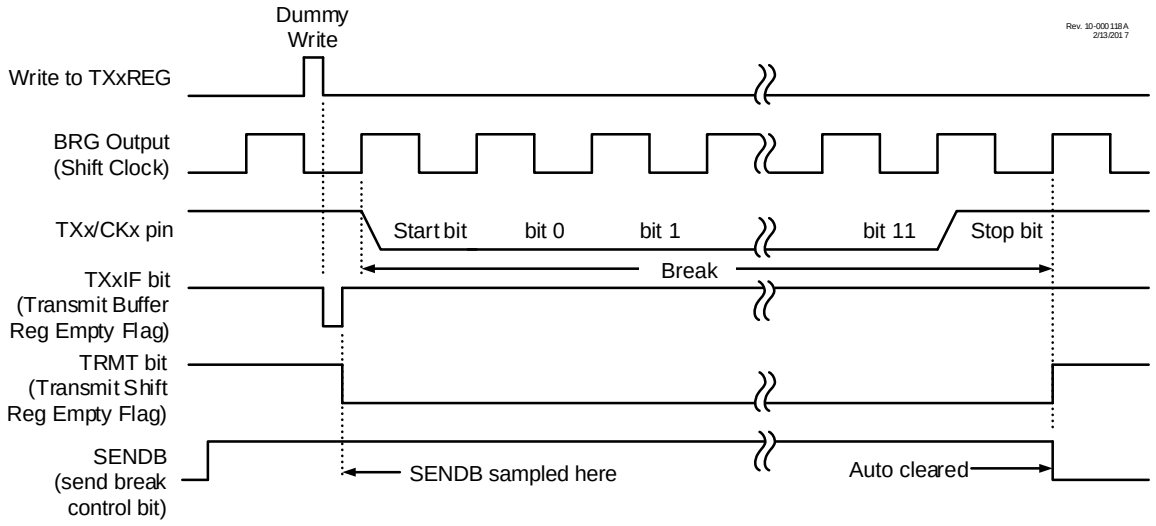
A Break character has been received when all three of the following conditions are true:

- RCxIF bit is set
- FERR bit is set
- RCxREG = 00h

The second method uses the Auto-Wake-up feature described in "Auto-Wake-up on Break". By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCxIF 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 before placing the EUSART in Sleep mode.

**Figure 24-9. Send Break Character Sequence**




## 24.4 EUSART Synchronous Mode

Synchronous serial communications are typically used in systems with a single master and one or more slaves. The master device contains the necessary circuitry for baud rate generation and supplies the clock for all devices in the system. Slave devices can take advantage of the master clock by eliminating the internal clock generation circuitry.

There are two signal lines in Synchronous mode: a bidirectional data line (DT) and a clock line (CK). The 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.

### 24.4.1 Synchronous Master Mode

The following bits are used to configure the EUSART for synchronous master operation:

- The SYNC bit is set to '1' to configure the EUSART for synchronous operation
- The Clock Source Select (CSRC) bit is set to '1' to configure the EUSART as the master
- The Single Receive Enable (SREN) bit is set to '0' for transmit; SREN = 1 for receive (recommended setting to receive 1 byte)
- The Continuous Receive Enable (CREN) bit is set to '0' for transmit; CREN = 1 to receive continuously
- The SPEN bit is set to '1' to enable the EUSART interface



**Important:** Clearing the SREN and CREN bits ensure that the device is in the Transmit mode, otherwise the device will be configured to receive.

#### 24.4.1.1 Master Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a master transmits the clock on the TX/CK line. The TXx/CKx 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.

#### 24.4.1.2 Clock Polarity

A clock polarity option is provided for Microwire compatibility. Clock polarity is selected with the Clock/Transmit Polarity Select (SCKP) bit. 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.

#### 24.4.1.3 Synchronous Master Transmission

Data is transferred out of the device on the RXx/DTx pin. The RXx/DTx and TXx/CKx 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 TXxREG register. If the TSR still contains all or part of a previous character the new character data is held in the TXxREG 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 TXxREG is immediately transferred to the TSR. The transmission of the character commences immediately following the transfer of the data to the TSR from the TXxREG.

Each data bit changes on the leading edge of the master clock and remains valid until the subsequent leading clock edge.

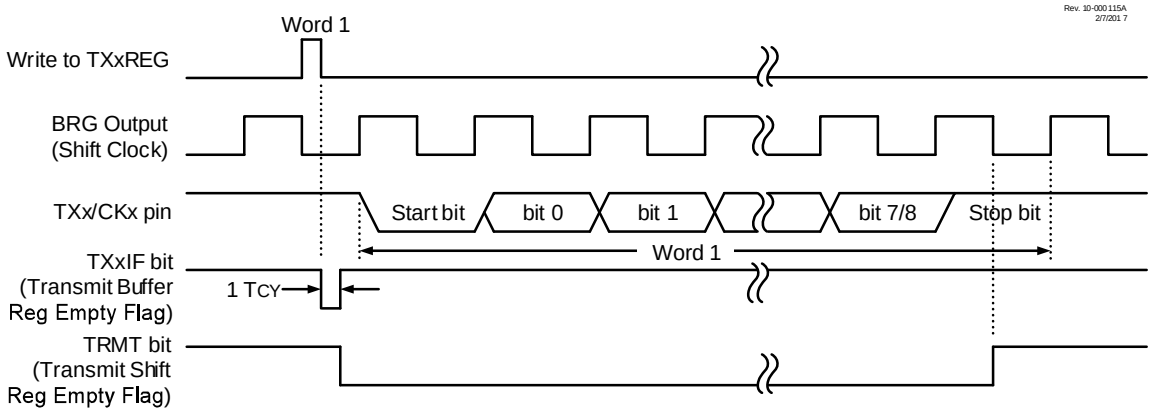
**Note:** The TSR register is not mapped in data memory, so it is not available to the user.

#### 24.4.1.4 Synchronous Master Transmission Setup

1. Initialize the SPxBRGH;SPxBRGL register pair and the BRG16 bit to achieve the desired baud rate (see section "EUSART Baud Rate Generator (BRG)").
2. Select the transmit output pin by writing the appropriate values to the RxyPPS register and RXxPPS register. Both selections should enable the same pin.
3. Select the clock output pin by writing the appropriate values to the RxyPPS register and TXxPPS register. Both selections should enable the same pin.
4. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
5. Disable Receive mode by clearing the SREN and CREN bits.

6. Enable Transmit mode by setting the TXEN bit.
7. If 9-bit transmission is desired, set the TX9 bit.
8. If interrupts are desired, set the TXxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
9. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
10. Start transmission by loading data to the TXxREG register.

**Figure 24-10. Synchronous Transmission**




### 24.4.1.5 Synchronous Master Reception

Data is received at the RXx/DTx pin. The RXx/DTx 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 (SREN) bit or the Continuous Receive Enable (CREN) bit.

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 RXx/DTx 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 RCxIF 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 RCxREG. The RCxIF bit remains set as long as there are unread characters in the receive FIFO.

**Note:** If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

### 24.4.1.6 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 TXx/CKx 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.

**Important:** If the device is configured as a slave and the TX/CK function is on an analog pin, the corresponding ANSEL bit must be cleared.

#### 24.4.1.7 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 the FIFO is accessed. When this happens the Overrun Error (OERR) bit 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 or by resetting the EUSART by clearing the SPEN bit.

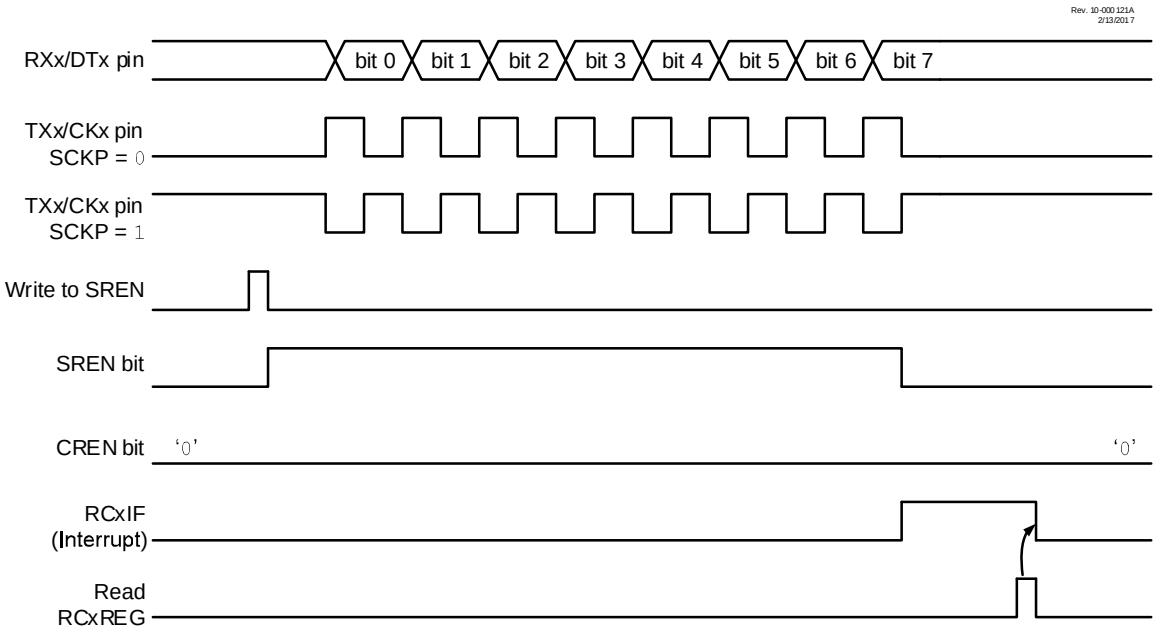
#### 24.4.1.8 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the 9-Bit Receive Enable (RX9) bit is set, the EUSART will shift nine bits into the RSR for each character received. The RX9D bit 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 RCxREG.

#### 24.4.1.9 Synchronous Master Reception Setup

1. Initialize the SPxBRGH:SPxBRGL register pair and set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Select the receive input pin by writing the appropriate values to the RxyPPS register and RXxPPS register. Both selections should enable the same pin.
3. Select the clock output pin by writing the appropriate values to the RxyPPS register and TXxPPS register. Both selections should enable the same pin.
4. Clear the ANSEL bit for the RXx pin (if applicable).
5. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
6. Ensure that CREN and SREN bits are cleared.
7. If interrupts are desired, set the RCxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
8. If 9-bit reception is desired, set bit RX9.
9. Start reception by setting the SREN bit or for continuous reception, set the CREN bit.
10. Interrupt flag bit RCxIF will be set when reception of a character is complete. An interrupt will be generated if the enable bit RCxIE was set.
11. Read the RCxSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
12. Read the 8-bit received data by reading the RCxREG register.
13. If an overrun error occurs, clear the error by either clearing the CREN bit or by clearing the SPEN bit which resets the EUSART.

**Figure 24-11. Synchronous Reception (Master Mode, SREN)**



### 24.4.2 Synchronous Slave Mode

The following bits are used to configure the EUSART for synchronous slave operation:

- SYNC = 1 (configures the EUSART for synchronous operation.)
- CSRC = 0 (configures the EUSART as a slave)
- SREN = 0 (for transmit); SREN = 1 (for single byte receive)
- CREN = 0 (for transmit); CREN = 1 (recommended setting for continuous receive)
- SPEN = 1 (enables the EUSART)

**Important:** Clearing the SREN and CREN bits ensure that the device is in Transmit mode, otherwise the device will be configured to receive.

#### 24.4.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see "Synchronous Master Transmission"), except in the case of the Sleep mode.

If two words are written to the TXxREG 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 TXxREG register.
3. The TXxIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXxREG register will transfer the second character to the TSR and the TXxIF bit will now be set.
5. If the PEIE and TXxIE 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.

#### 24.4.2.2 Synchronous Slave Transmission Setup

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Select the transmit output pin by writing the appropriate values to the RxyPPS register and RXxPPS register. Both selections should enable the same pin.
3. Select the clock input pin by writing the appropriate value to the TXxPPS register.
4. Clear the ANSEL bit for the CKx pin (if applicable).
5. Clear the CREN and SREN bits.
6. If interrupts are desired, set the TXxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
7. If 9-bit transmission is desired, set the TX9 bit.
8. Enable transmission by setting the TXEN bit.
9. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
10. Prepare for transmission by writing the Least Significant eight bits to the TXxREG register. The word will be transmitted in response to the Master clocks at the CKx pin.

#### 24.4.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (see "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 RCxREG register. If the RCxIE 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.

#### 24.4.2.4 Synchronous Slave Reception Setup:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Select the receive input pin by writing the appropriate value to the RXxPPS register.
3. Select the clock input pin by writing the appropriate values to the TXxPPS register.
4. Clear the ANSEL bit for both the TXx/CKx and RXx/DTx pins (if applicable).
5. If interrupts are desired, set the RCxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
6. If 9-bit reception is desired, set the RX9 bit.
7. Set the CREN bit to enable reception.
8. The RCxIF bit will be set when reception is complete. An interrupt will be generated if the RCxIE bit was set.
9. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit.
10. Retrieve the eight Least Significant bits from the receive FIFO by reading the RCxREG register.
11. If an overrun error occurs, clear the error by either clearing the CREN bit or by clearing the SPEN bit which resets the EUSART.

## 24.5 EUSART Operation During Sleep

The EUSART will remain active during Sleep only in the Synchronous Slave mode. All other modes require the system clock and therefore cannot generate the necessary signals to run the Transmit or Receive Shift registers during Sleep.

Synchronous Slave mode uses an externally generated clock to run the Transmit and Receive Shift registers.

### 24.5.1 Synchronous Receive During Sleep

To receive during Sleep, all the following conditions must be met before entering Sleep mode:

- RCxSTA and TXxSTA Control registers must be configured for Synchronous Slave Reception (see "Synchronous Slave Reception Setup").
- If interrupts are desired, set the RCxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
- The RCxIF interrupt flag must be cleared by reading RCxREG to unload any pending characters in the receive buffer.

Upon entering Sleep mode, the device will be ready to accept data and clocks on the RXx/DTx and TXx/CKx pins, respectively. When the data word has been completely clocked in by the external device, the RCxIF Interrupt Flag bit of the PIRx register will be set. Thereby, waking the processor from Sleep.

Upon waking from Sleep, the instruction following the `SLEEP` instruction will be executed. If the Global Interrupt Enable (GIE) bit of the INTCON register is also set, then the Interrupt Service Routine (ISR) will be called.

### 24.5.2 Synchronous Transmit During Sleep

To transmit during Sleep, all the following conditions must be met before entering Sleep mode:

- The RCxSTA and TXxSTA Control registers must be configured for synchronous slave transmission (see "Synchronous Slave Transmission Setup").
- The TXxIF interrupt flag must be cleared by writing the output data to the TXxREG, thereby filling the TSR and transmit buffer.
- The TXxIE interrupt enable bits of the PIEx register and PEIE of the INTCON register must be written to '1'.
- If interrupts are desired, set the GIE bit of the INTCON register.

Upon entering Sleep mode, the device will be ready to accept clocks on the TXx/CKx pin and transmit data on the RXx/DTx pin. When the data word in the TSR register has been completely clocked out by the external device, the pending byte in the TXxREG will transfer to the TSR and the TXxIF flag will be set. Thereby, waking the processor from Sleep. At this point, the TXxREG is available to accept another character for transmission. Writing TXxREG will clear the TXxIF flag.

Upon waking from Sleep, the instruction following the `SLEEP` instruction will be executed. If the Global Interrupt Enable (GIE) bit is also set then the Interrupt Service Routine (ISR) will be called.

## 24.6 Register Definitions: EUSART Control

### 24.6.1 TXxSTA

**Name:** TXxSTA
**Address:** 0x011E

Transmit Status and Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

**Bit 7 – CSRC** Clock Source Select

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 1 | Master mode (clock generated internally from BRG) |
| 0 | SYNC = 1 | Slave mode (clock from external source) |
| X | SYNC = 0 | Don't care |

**Bit 6 – TX9** 9-bit Transmit Enable

| Value | Description |
|---|---|
| 1 | Selects 9-bit transmission |
| 0 | Selects 8-bit transmission |

**Bit 5 – TXEN** Transmit Enable
Enables transmitter(1)

| Value | Description |
|---|---|
| 1 | Transmit enabled |
| 0 | Transmit disabled |

**Bit 4 – SYNC** EUSART Mode Select

| Value | Description |
|---|---|
| 1 | Synchronous mode |
| 0 | Asynchronous mode |

**Bit 3 – SENDB** Send Break Character

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 0 | Send Sync Break on next transmission (cleared by hardware upon completion) |
| 0 | SYNC = 0 | Sync Break transmission disabled or completed |
| X | SYNC = 1 | Don't care |

**Bit 2 – BRGH** High Baud Rate Select

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 0 | High speed, if BRG16 = 1, baud rate is baudclk/4; else baudclk/16 |
| 0 | SYNC = 0 | Low speed |
| X | SYNC = 1 | Don't care |

**Bit 1 – TRMT** Transmit Shift Register (TSR) Status

| Value | Description |
|---|---|
| 1 | TSR is empty |
| 0 | TSR is not empty |

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

**Note:**

1. SREN and CREN bits override TXEN in Sync mode.

### 24.6.2 RCxSTA

**Name:** RCxSTA
**Address:** 0x011D

Receive Status and Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| Access | R/W | R/W | R/W/HC | R/W | R/W | R | R/HC | R/HC |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – SPEN** Serial Port Enable

| Value | Description |
|---|---|
| 1 | Serial port enabled |
| 0 | Serial port disabled (held in Reset) |

**Bit 6 – RX9** 9-Bit Receive Enable

| Value | Description |
|---|---|
| 1 | Selects 9-bit reception |
| 0 | Selects 8-bit reception |

**Bit 5 – SREN** Single Receive Enable
Controls reception. This bit is cleared by hardware when reception is complete

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 1 AND CSRC = 1 | Start single receive |
| 0 | SYNC = 1 AND CSRC = 1 | Single receive is complete |
| X | SYNC = 0 OR CSRC = 0 | Don't care |

**Bit 4 – CREN** Continuous Receive Enable

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 1 | Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN) |
| 0 | SYNC = 1 | Disables continuous receive |
| 1 | SYNC = 0 | Enables receiver |
| 0 | SYNC = 0 | Disables receiver |

**Bit 3 – ADDEN** Address Detect Enable

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 0 AND RX9 = 1 | The receive buffer is loaded and the interrupt occurs only when the ninth received bit is set |
| 0 | SYNC = 0 AND RX9 = 1 | All bytes are received and interrupt always occurs. Ninth bit can be used as parity bit |
| X | RX9 = 0 OR SYNC = 1 | Don't care |

**Bit 2 – FERR** Framing Error

| Value | Description |
|---|---|
| 1 | Unread byte in RCxREG has a framing error |
| 0 | Unread byte in RCxREG does not have a framing error |

**Bit 1 – OERR** Overrun Error

| Value | Description |
|---|---|
| 1 | Overrun error (can be cleared by clearing either SPEN or CREN bit) |
| 0 | No overrun error |

**Bit 0 – RX9D** Ninth bit of Received Data
This can be address/data bit or a parity bit which is determined by user firmware.

### 24.6.3 BAUDxCON

**Name:** BAUDxCON
**Address:** 0x011F

Baud Rate Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ABDOVF | RCIDL | | SCKP | BRG16 | | WUE | ABDEN |
| Access | R | R | | R/W | R/W | | R/W | R/W |
| Reset | 0 | 0 | | 0 | 0 | | 0 | 0 |

**Bit 7 – ABDOVF** Auto-Baud Detect Overflow

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 0 | Auto-baud timer overflowed |
| 0 | SYNC = 0 | Auto-baud timer did not overflow |
| X | SYNC = 1 | Don't care |

**Bit 6 – RCIDL** Receive Idle Flag

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 0 | Receiver is Idle |
| 0 | SYNC = 0 | Start bit has been received and the receiver is receiving |
| X | SYNC = 1 | Don't care |

**Bit 4 – SCKP** Clock/Transmit Polarity Select

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 0 | Idle state for transmit (TX) is a low level (transmit data inverted) |
| 0 | SYNC = 0 | Idle state for transmit (TX) is a high level (transmit data is non-inverted) |
| 1 | SYNC = 1 | Data is clocked on rising edge of the clock |
| 0 | SYNC = 1 | Data is clocked on falling edge of the clock |

**Bit 3 – BRG16** 16-bit Baud Rate Generator Select

| Value | Description |
|---|---|
| 1 | 16-bit Baud Rate Generator is used |
| 0 | 8-bit Baud Rate Generator is used |

**Bit 1 – WUE** Wake-up Enable

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 0 | Receiver is waiting for a falling edge. Upon falling edge no character will be received and flag RCxIF will be set. WUE will automatically clear after RCxIF is set. |
| 0 | SYNC = 0 | Receiver is operating normally |
| X | SYNC = 1 | Don't care |

**Bit 0 – ABDEN** Auto-Baud Detect Enable

| Value | Condition | Description |
|---|---|---|
| 1 | SYNC = 0 | Auto-Baud Detect mode is enabled (clears when auto-baud is complete) |
| 0 | SYNC = 0 | Auto-Baud Detect is complete or mode is disabled |
| X | SYNC = 1 | Don't care |

### 24.6.4 RCxREG

**Name:** RCxREG
**Address:** 0x0119

Receive Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RCREG[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – RCREG[7:0]** Receive data

### 24.6.5 TXxREG

**Name:** TXxREG
**Address:** 0x011A

Transmit Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXREG[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TXREG[7:0]** Transmit Data

### 24.6.6 SPxBRG

**Name:** SPxBRG
**Address:** 0x011B

EUSART Baud Rate Generator

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | SPBRG[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SPBRG[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – SPBRG[15:0]** Baud Rate Register

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- SPxBRGH: Accesses the high byte SPBRG[15:8]
- SPxBRGL: Accesses the low byte SPBRG[7:0]

## 24.7 Register Summary - EUSART

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 ... 0x0118 | Reserved | | | | | | | | | |
| 0x0119 | RC1REG | 7:0 | RCREG[7:0] | | | | | | | |
| 0x011A | TX1REG | 7:0 | TXREG[7:0] | | | | | | | |
| 0x011B | SP1BRG | 7:0 | SPBRG[7:0] | | | | | | | |
| | | 15:8 | SPBRG[15:8] | | | | | | | |
| 0x011D | RC1STA | 7:0 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| 0x011E | TX1STA | 7:0 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| 0x011F | BAUD1CON | 7:0 | ABDOVF | RCIDL | | SCKP | BRG16 | | WUE | ABDEN |

# 25. MSSP - Master Synchronous Serial Port Module

The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, Shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit ($I^2C$)

The SPI interface can operate in Master or Slave mode and supports the following features:

- Selectable Clock Parity
- Slave Select Synchronization (Slave Mode Only)
- Daisy-Chain Connection of Slave Devices

The $I^2C$ interface can operate in Master or Slave mode and supports the following modes and features:

- Byte NACKing (Slave Mode)
- Limited Multi-Master Support
- 7-Bit and 10-Bit Addressing
- Start and Stop Interrupts
- Interrupt Masking
- Clock Stretching
- Bus Collision Detection
- General Call Address Matching
- Address Masking
- Address Hold and Data Hold Modes
- Selectable SDA Hold Times

## 25.1 SPI Mode Overview

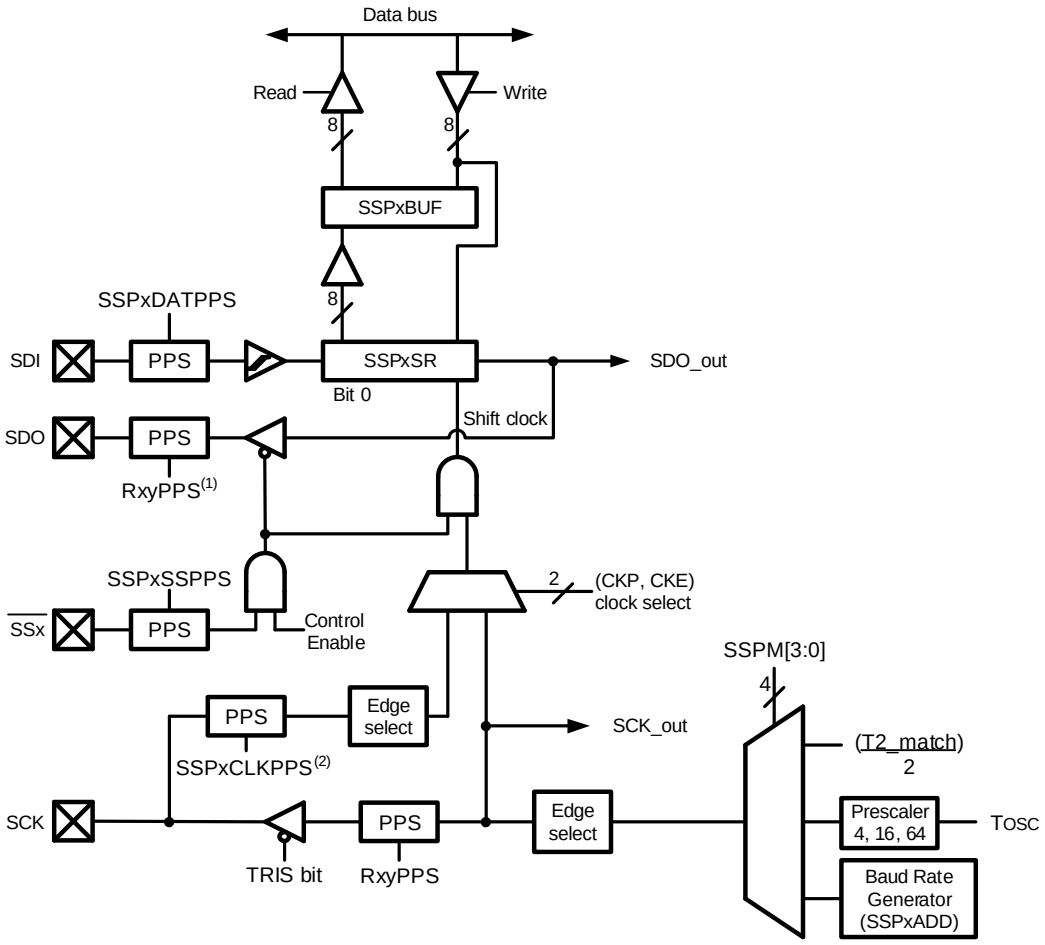
The Serial Peripheral Interface (SPI) is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is selected for communication using the Slave Select feature.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Slave Select ($\overline{SS}$)

Figure 25-1 shows the block diagram of the MSSP module when operating in SPI mode.

**Figure 25-1. MSSP Block Diagram (SPI mode)**

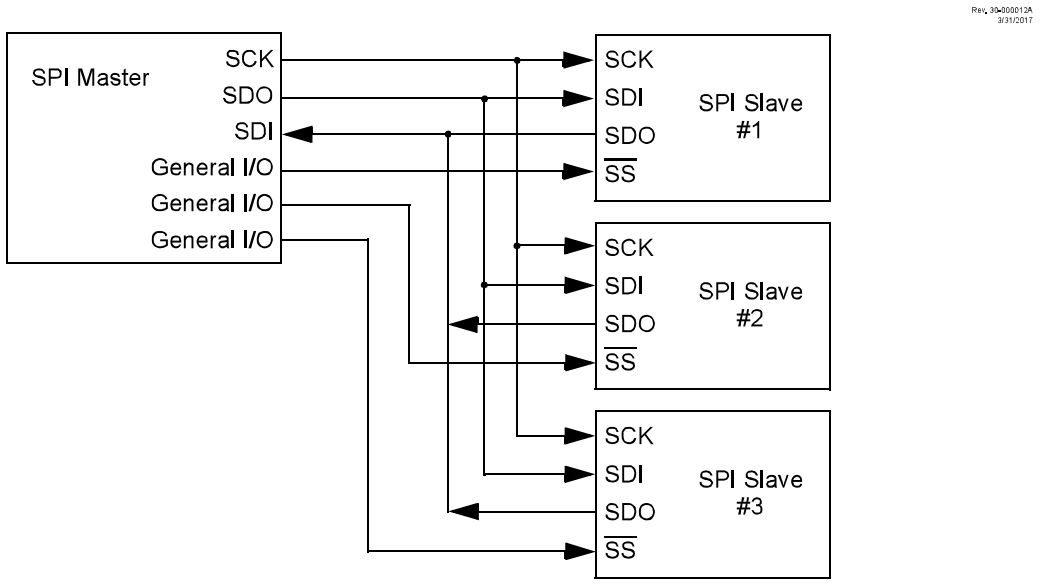



**Note 1:** Output selection for Master mode
**2:** Input selection for Slave and Master modes

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. 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.

Figure 25-2 shows a typical connection between a master device and multiple slave devices.

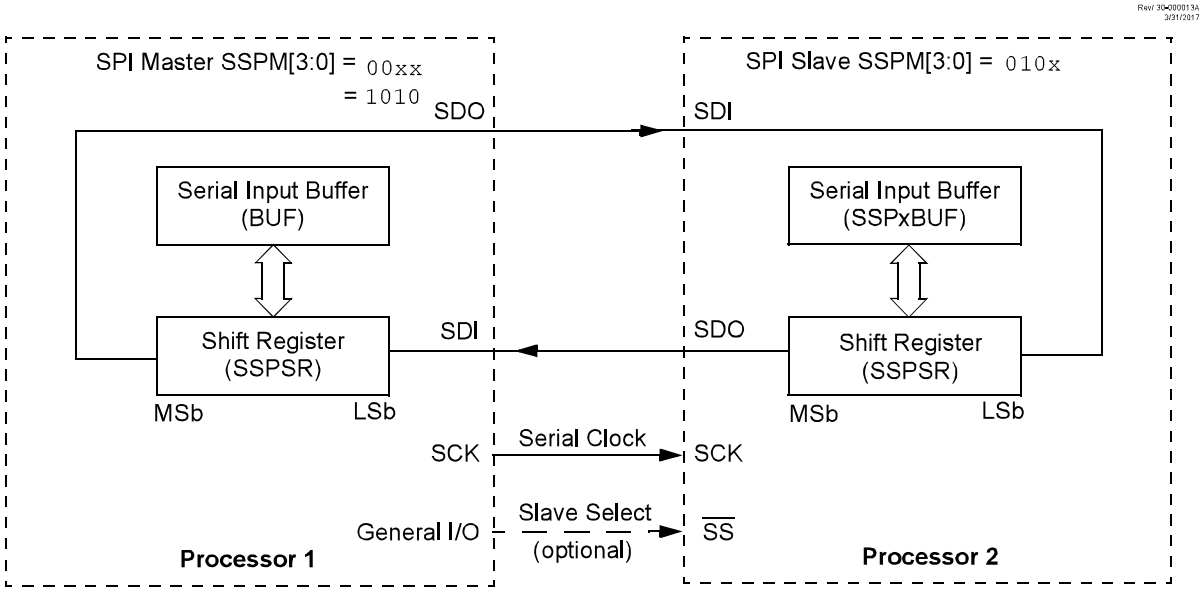
**Figure 25-2. SPI Master and Multiple Slave Connection**




Transmissions involve two Shift registers, eight bits in size, one in the master and one in the slave. 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 25-3 shows a typical connection between two processors configured as master and slave devices.

**Figure 25-3. SPI Master/Slave Connection**




Data is shifted out of both Shift registers on the programmed clock edge and latched on the opposite edge of the clock.

The master device transmits information out on its SDO output pin, which is connected to and received by the slave's SDI input pin. The slave device transmits information out on its SDO output pin, which is connected to and received by the master's SDI input pin.

To begin communication, the master device transmits both the MSb from its Shift register the clock signal. Both the master and the slave devices should be configured for the same clock polarity. During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the master device is sending out the MSb from its Shift register (on its SDO pin) and the slave device is reading this bit and saving it as the LSb of its Shift register, the slave device is also sending out the MSb from its Shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its Shift register.

After 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 must be performed in multiples of eight 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.

### 25.1.1 SPI Mode Registers

The MSSP module has six registers for SPI mode operation. These are:

- MSSP STATUS register (SSPxSTAT)
- MSSP Control register 1 (SSPxCON1)
- MSSP Control register 3 (SSPxCON3)
- MSSP Data Buffer register (SSPxBUF)
- MSSP Address register (SSPxADD)
- MSSP Shift register (SSPSR)
  (Not directly accessible)

SSPxCON1 and SSPxSTAT are the control and status registers for 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.

One of the five SPI Master modes uses the SSPxADD value to determine the Baud Rate Generator clock frequency. More information on the Baud Rate Generator is available in the “Baud Rate Generator” section.

SSPSR is the Shift register used for shifting data in and out. SSPxBUF provides indirect access to the SSPSR register. SSPxBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPSR and SSPxBUF together create a buffered receiver. When SSPSR 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 SSPSR.

### 25.1.2 SPI Mode Operation

When initializing the SPI several options need to be specified. This is done by programming the appropriate control bits (SSPxCON1[5:0] and SSPxSTAT[7:6]). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

To enable the serial port, the SSP Enable bit (SSPEN) must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPxCONy registers and then set the SSPEN bit. The SDI, SDO, SCK and $\overline{SS}$ serial port pins are selected with the PPS controls. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI must have corresponding TRIS bit set.
- SDO must have corresponding TRIS bit cleared.
- SCK (Master mode) must have corresponding TRIS bit cleared.
- SCK (Slave mode) must have corresponding TRIS bit set.
- The RxyPPS and SSPxCLKPPS controls must select the same pin.
- $\overline{SS}$ must have corresponding TRIS bit set.

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

The MSSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPxBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPxBUF holds the data that was written to the SSPSR 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 Status (BF) bit and the MSSP Interrupt Flag (SSPxIF) bit are set. This double-buffering of the received data 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 (WCOL) bit 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 BF bit, indicates when SSPxBUF has been loaded with the received data (transmission is complete). When the SSPxBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur.



**Important:** The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPxBUF register.

#### 25.1.2.1 SPI Master Mode

The master can initiate the data transfer at any time because it controls the SCK line. The master determines when the slave (Processor 2, Figure 25-3) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPxBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPxBUF register as if a normal received byte (interrupts and Status bits appropriately set).

The clock polarity is selected by appropriately programming the Clock Polarity Select (CKP) and SPI Clock Edge Select (CKE) bits. Figure 25-4 shows the four clocking configurations. When the CKE bit is set, the SDO data is valid one half of a clock cycle before a clock edge appears on SCK, and transmission occurs on the transition from the Active to Idle clock state. When CKE is clear, the SDO data is valid at the same time as the clock edge appears on SCK, and transmission occurs on the transition from the Idle to Active clock states.

The SPI Data Input Sample (SMP) bit determines when the SDI input is sampled. When SMP is set, input data is sampled at the end of the data output time. When SMP is clear, input data is sampled at the middle of the data output time.

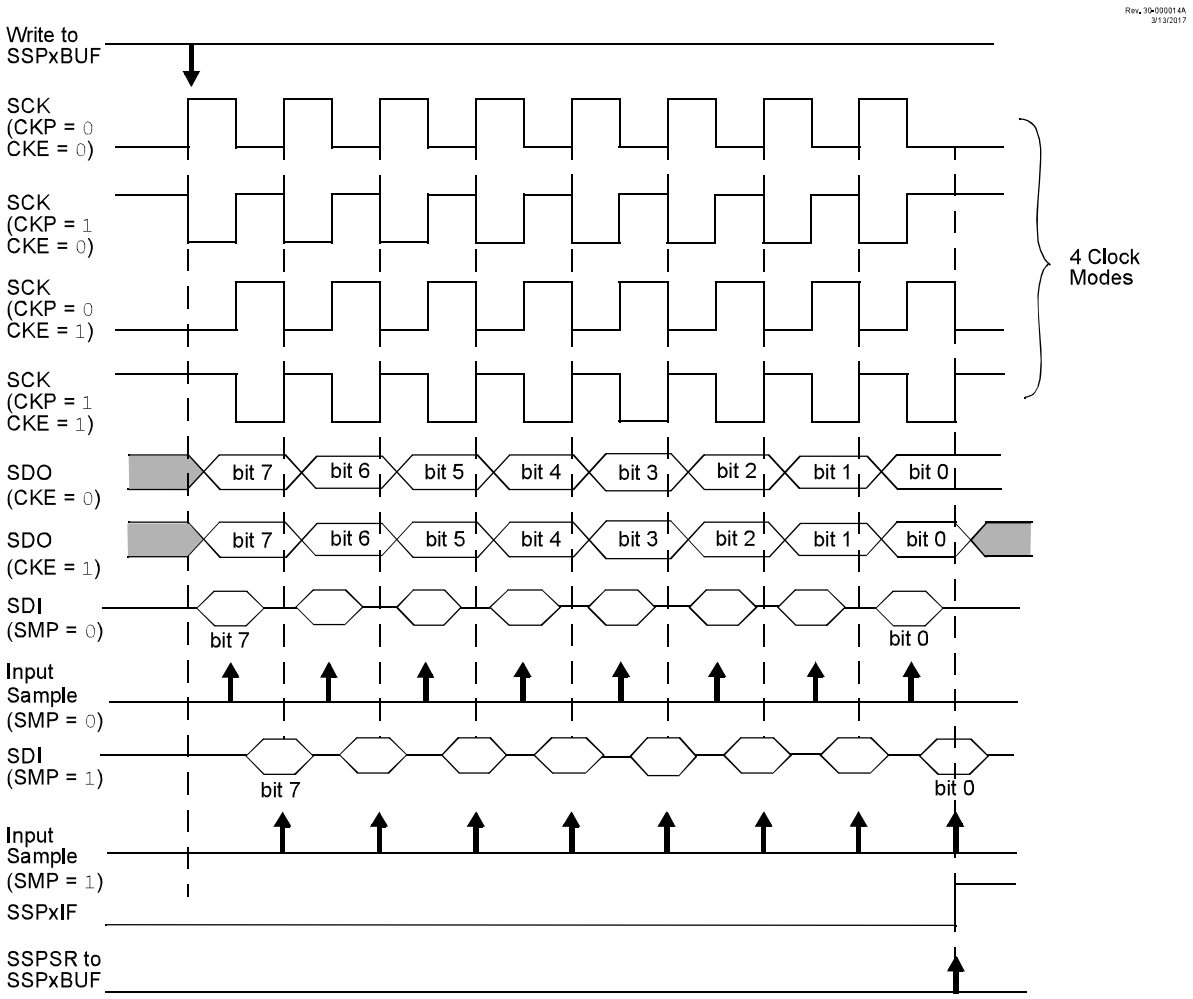
The SPI clock rate (bit rate) is user programmable to be one of the following:

- $F_{OSC}/4$ (or $T_{CY}$)
- $F_{OSC}/16$ (or 4 * $T_{CY}$)
- $F_{OSC}/64$ (or 16 * $T_{CY}$)

- Timer2 output/2
- $F_{OSC}/(4 * (SSPxADD + 1))$

**Important:** In Master mode the clock signal output to the SCK pin is also the clock signal input to the peripheral. The pin selected for output with the RxyPPS register must also be selected as the peripheral input with the SSPxCLKPPS register. The pin that is selected using the SSPxCLKPPS register should also be made a digital I/O. This is done by clearing the corresponding ANSEL bit.

**Figure 25-4. SPI Mode Waveform (Master Mode)**




### 25.1.2.2 SPI Slave Mode

In Slave mode, the data is transmitted and received as external clock pulses appear on SCK. When the last bit is latched, the 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 SCK pin. The Idle state is determined by the CKP bit.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the **"Electrical Specifications"** chapter.

While in Sleep mode, the slave can transmit/receive data. The Shift register is clocked from the SCK pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

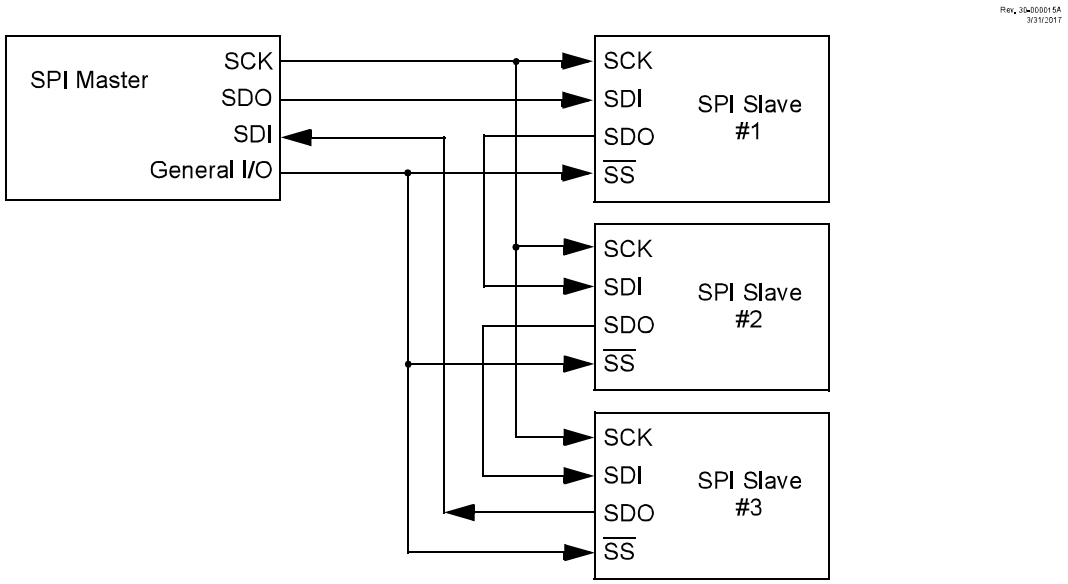
### 25.1.2.3 Daisy-Chain Configuration

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.

In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the Buffer Overwrite Enable (BOEN) bit 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.

Figure 25-5 shows the block diagram of a typical daisy-chain connection when operating in SPI mode.

**Figure 25-5. SPI Daisy-Chain Connection**




### 25.1.2.4 Slave Select Synchronization

The Slave Select can also be used to synchronize communication (see Figure 25-6). The Slave Select line is held high until the master device is ready to communicate. When the Slave Select line is pulled low, the slave knows that a new transmission is starting.

If the slave fails to receive the communication properly, it will be reset at the end of the transmission, when the Slave Select line returns to a high state. The slave is then ready to receive a new transmission when the Slave Select line is pulled low again. If the Slave Select line is not used, there is a risk that the slave will eventually become out of sync with the master. If the slave misses a bit, it will always be one bit off in future transmissions. Use of the Slave Select line allows the slave and master to align themselves at the beginning of each transmission.

The $\overline{SS}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with $\overline{SS}$ pin control enabled (MSSP Mode Select (SSPM) bits = 0100).

When the $\overline{SS}$ pin is low, transmission and reception are enabled and the SDO pin is driven.

When the $\overline{SS}$ pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the $\overline{SS}$ pin to a high level or clearing the SSPEN bit.

**Important:**

1. When the SPI is in Slave mode with $\overline{SS}$ pin control enabled (SSPM = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to $V_{DD}$.
2. When the SPI is used in Slave mode with CKE set; the user must enable $\overline{SS}$ pin control (see Figure 25-8. If CKE is clear, $\overline{SS}$ pin control is optional (see Figure 25-7).
3. While operated in SPI Slave mode the SMP bit must remain clear.

**Figure 25-6. Slave Select Synchronous Waveform**

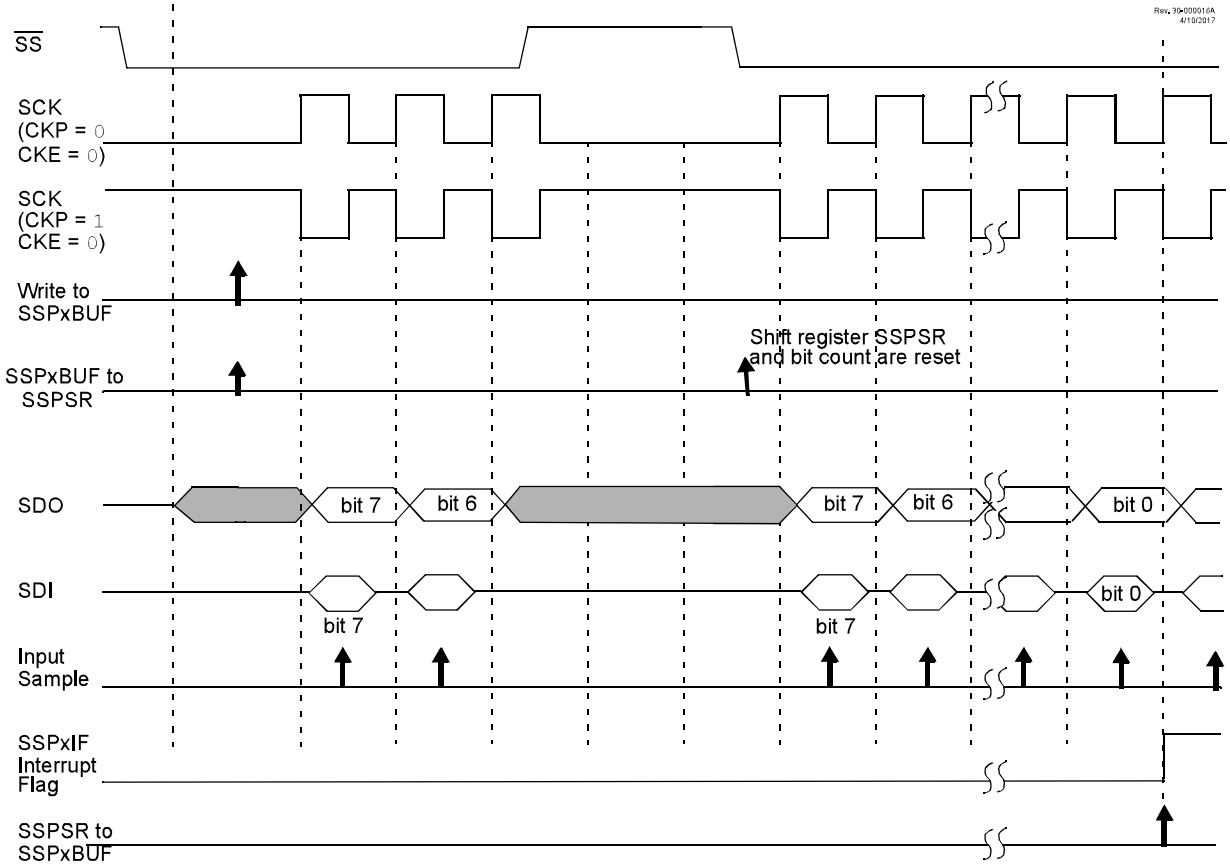


**Figure 25-7. SPI Mode Waveform (Slave Mode with CKE = 0)**

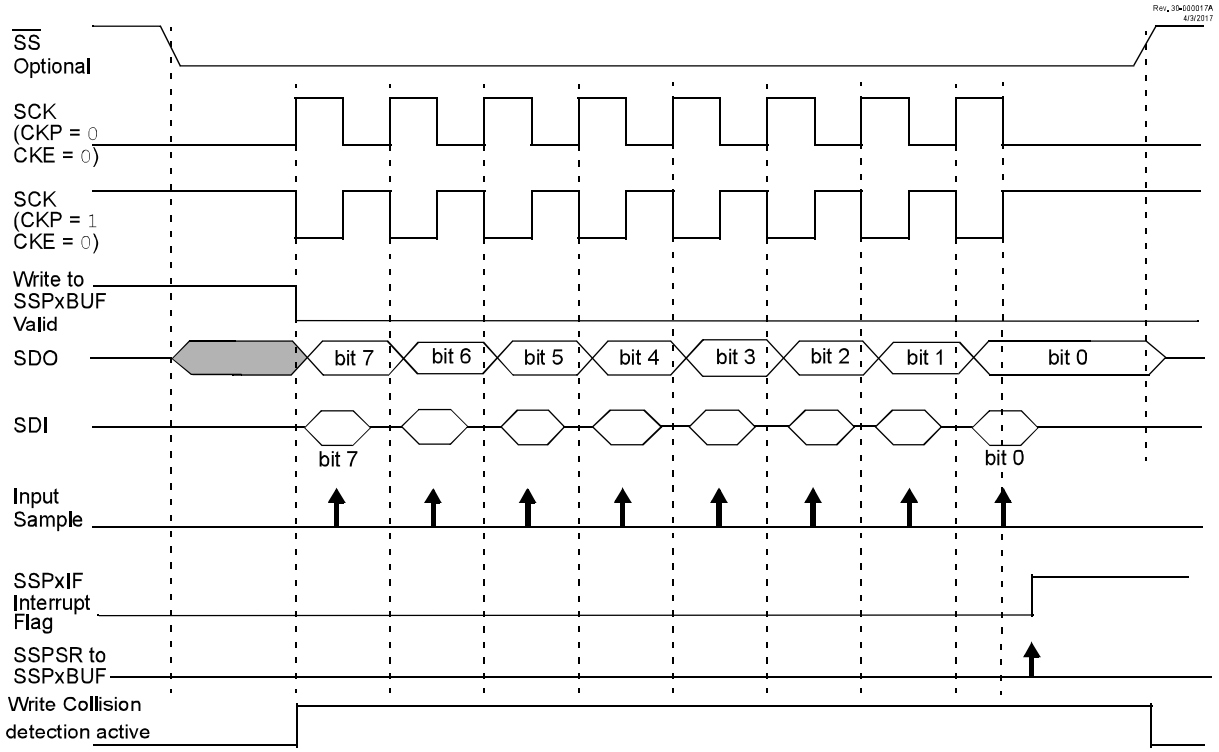
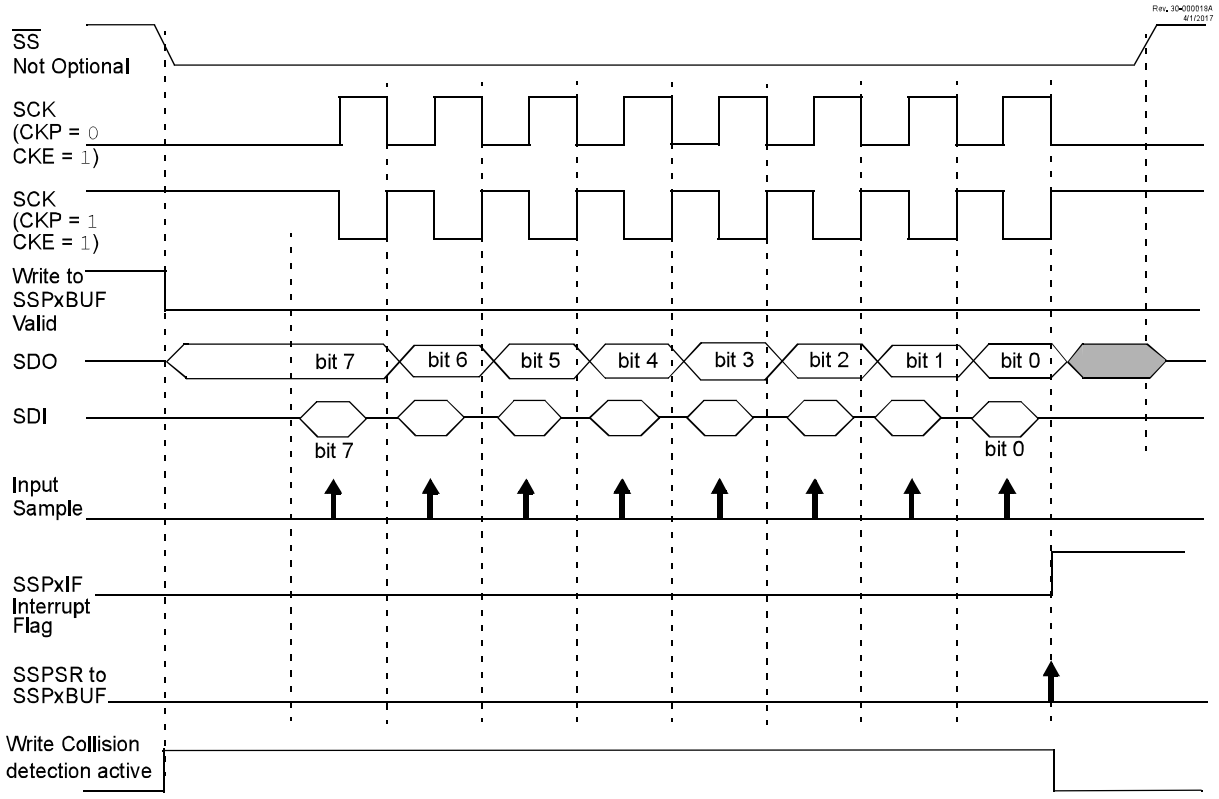



**Figure 25-8. SPI Mode Waveform (Slave Mode with CKE = 1)**



### 25.1.2.5 SPI Operation in Sleep Mode

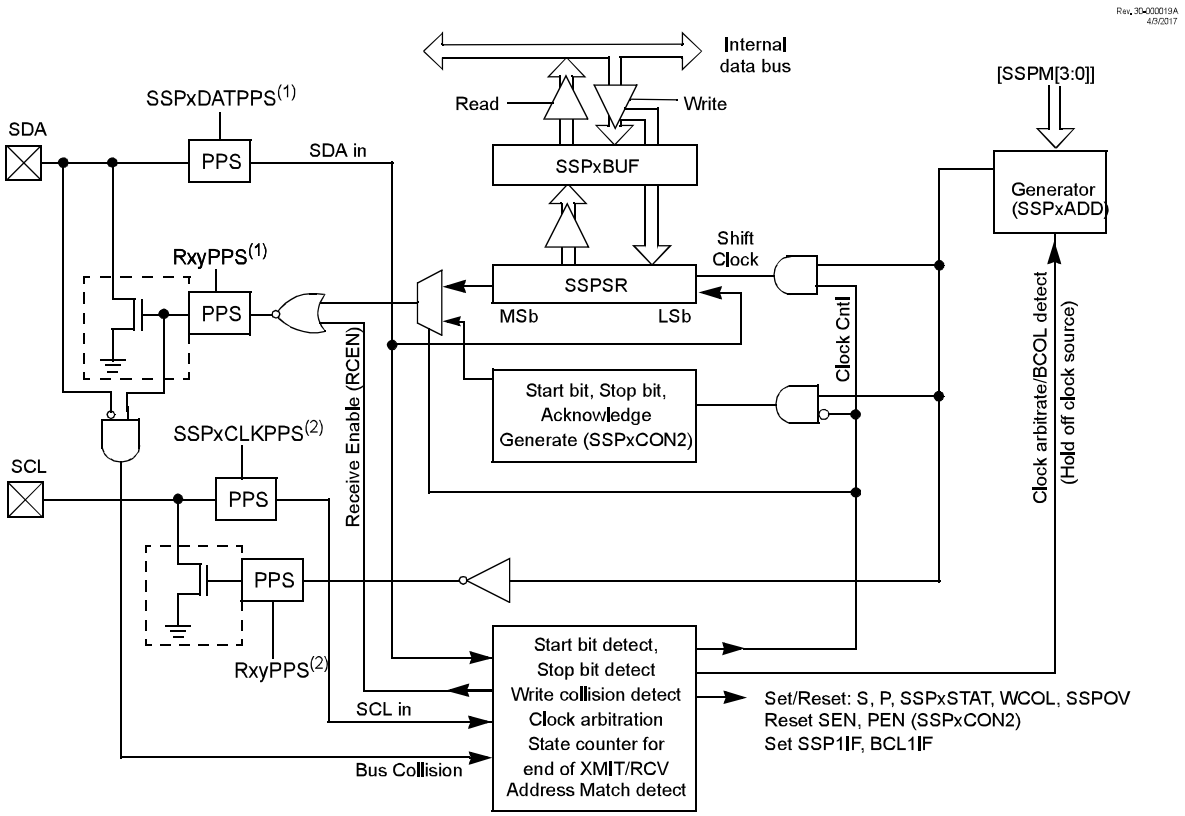
In SPI Master mode, when the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all eight bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

## 25.2 I²C Mode Overview

The Inter-Integrated Circuit ($I^2C$) bus 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. Figure 25-9 and Figure 25-10 show block diagrams of the $I^2C$ Master and Slave modes, respectively.

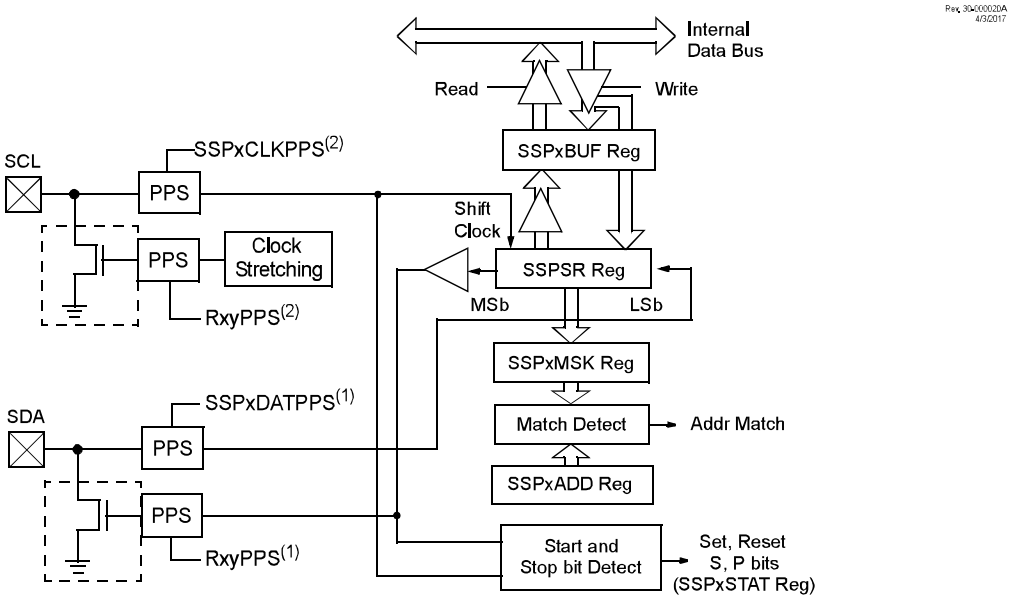
**Figure 25-9. MSSP Block Diagram ($I^2C$ Master mode)**




**Note 1:** SDA pin selections must be the same for input and output.

**2:** SCL pin selections must be the same for input and output.

**Figure 25-10. MSSP Block Diagram (I²C Slave mode)**




**Note 1:** SDA pin selections must be the same for input and output.
**2:** SCL pin selections must be the same for input and output.

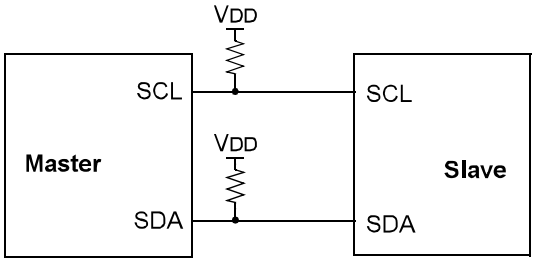
The I²C bus specifies two signal connections:

- Serial Clock (SCL)
- Serial Data (SDA)

Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors for the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one.

Figure 25-11 shows a typical connection between two processors configured as master and slave devices.

**Figure 25-11. I²C Master/Slave Connection**




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, the master device transmits a Start condition followed by the address byte of the slave it intends to communicate with. A Start condition is indicated by a high-to-low transition of the SDA line while the SCL line is held high. Address and data bytes are sent out, Most Significant bit (MSb) first. This is followed by a single Read/$\overline{\text{Write}}$ Information (R/$\overline{\text{W}}$) bit, which determines whether the master intends to transmit to or receive data from the slave device. The R/$\overline{\text{W}}$ 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.

If the requested slave exists on the bus, it will respond with an Acknowledge sequence, otherwise known as an $\overline{\text{ACK}}$. The Acknowledge sequence is an active-low signal, which holds the SDA line low to indicate to the transmitter that the slave device has received the transmitted data and is ready to receive more. The master then continues to either transmit to or receive data from the slave.

The transition of a data bit is always performed while the SCL line is held low. Transitions that occur while the SCL line is held high are used to indicate Start and Stop conditions.

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 $\overline{\text{ACK}}$ sequence. 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 $\overline{\text{ACK}}$ sequence. In this example, the master device is in Master Receive mode and the slave is in Slave Transmit mode.

On the last byte of data communicated, the master device may end the transmission by sending a Stop condition. If the master device is in Receive mode, it sends the Stop condition in place of the last $\overline{\text{ACK}}$ sequence. A Stop condition is indicated by a low-to-high transition of the SDA line while the SCL line is held high.

In some cases, the master may want to maintain control of the bus and re-initiate another transmission. If so, the master device may send Restart condition in place of the Stop condition or last $\overline{\text{ACK}}$ sequence when it is in Receive mode.

The $I^2C$ 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.

### 25.2.1 $I^2C$ Mode Registers

The MSSP module has eight registers for $I^2C$ operation.

These are:

- MSSP Status register (SSPxSTAT)
- MSSP Control register 1 (SSPxCON1)
- MSSP Control register 2 (SSPxCON2)
- MSSP Control register 3 (SSPxCON3)
- Serial Receive/Transmit Buffer register (SSPxBUF)
- MSSP Address register (SSPxADD)
- $I^2C$ Slave Address Mask register (SSPxMSK)
- MSSP Shift register (SSPSR) – not directly accessible

SSPxCON1, SSPxCON2, SSPxCON3 and SSPxSTAT are the Control and Status registers in $I^2C$ mode operation. The SSPxCON1, SSPxCON2, and SSPxCON3 registers are readable and writable. The lower six bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write. SSPxMSK holds the slave address mask value used in address comparison. SSPxADD contains the slave device address when the MSSP is configured in $I^2C$ Slave mode. When the MSSP is configured in Master mode, SSPxADD acts as the Baud Rate Generator reload value.

SSPSR is the Shift register used for shifting data in or out. SSPxBUF is the Buffer register to which data bytes are written to or read from. In receive operations, SSPSR and SSPxBUF together, create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set. During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPSR.

### 25.2.2 I²C Mode Operation

All MSSP I²C communication is byte oriented and shifted out MSb first. Eight SFR registers and two interrupt flags interface the module with the PIC® microcontroller and user software. Two pins, SDA and SCL, are exercised by the module to communicate with other external I²C devices.

#### 25.2.2.1 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/NXP I²C Specification.

| TERM | Description |
|---|---|
| Transmitter | The device that shifts data out onto the bus. |
| Receiver | The device that shifts data in from the bus. |
| Master | The device that initiates a transfer, generates clock signals and terminates a transfer. |
| Slave | The device addressed by the master. |
| Multi-master | A bus with more than one device that can initiate data transfers. |
| Arbitration | Procedure to ensure that only one master at a time controls the bus. Winning arbitration ensures that the message is not corrupted. |
| Synchronization | Procedure to synchronize the clocks of two or more devices on the bus. |
| Idle | No master is controlling the bus, and both SDA and SCL lines are high. |
| Active | Any time one or more master devices are controlling the bus. |
| Addressed Slave | Slave device that has received a matching address and is actively being clocked by a master. |
| Matching Address | Address byte that is clocked into a slave that matches the value stored in SSPxADD. |
| Write Request | Slave receives a matching address with R/$\overline{W}$ bit clear, and is ready to clock in data. |
| Read Request | Master sends an address byte with the R/$\overline{W}$ bit set, indicating that it wishes to clock data out of the Slave. This data is the next and all following bytes until a Restart or Stop. |
| Clock Stretching | When a device on the bus hold SCL low to stall communication. |
| Bus Collision | Any time the SDA line is sampled low by the module while it is outputting and expected high state. |

#### 25.2.2.2 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 sequence sent back. After the eighth falling edge of the SCL line, the device outputting data on the SDA changes that pin to an input and reads the Acknowledge value on the next clock pulse.

The clock signal, SCL, is provided by the master. Data is valid to change while the SCL signal is low, and sampled on the rising edge of the clock. Changes on the SDA line while the SCL line is high define special conditions on the bus, such as a Start or Stop condition.

#### 25.2.2.3 SDA and SCL Pins

Selection of any I²C mode with the SSPEN bit set forces the SCL and SDA pins to be open-drain. These pins must be configured as inputs by setting the appropriate TRIS bits.

**Important:** Any device pin can be selected for SDA and SCL functions with the PPS peripheral. These functions are bidirectional. The SDA input is selected with the SSPxDATPPS registers. The SCL input is selected with the SSPxCLKPPS registers. Outputs are selected with the RxyPPS registers. It is the user's responsibility to make the selections so that both the input and the output for each function is on the same pin.

#### 25.2.2.4 SDA Hold Time

The hold time of the SDA pin is selected by the SDA Hold Time Selection (SDAHT) bit. Hold time is the time SDA is held valid after the falling edge of SCL. Setting the SDAHT bit selects a longer 300 ns minimum hold time and may help buses with large capacitance.

#### 25.2.2.5 Clock Stretching

Clock stretching occurs when a device on the bus holds the SCL line low, effectively pausing communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. A master device is not concerned with stretching as anytime it is active on the bus and not transferring data it is stretching. Any stretching done by a slave is invisible to the master software and handled by the hardware that generates SCL.

The CKP bit is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. Setting CKP will release SCL and allow more communication.

#### 25.2.2.6 Arbitration

Each master device must monitor the bus for Start and Stop conditions. If the device detects that the bus is busy, it cannot begin a new message until the bus returns to an Idle state.

However, two master devices may try to initiate a transmission on or about the same time. When this occurs, the process of arbitration begins. Each transmitter checks the level of the SDA data line and compares it to the level that it expects to find. The first transmitter to observe that the two levels do not match, loses arbitration, and must stop transmitting on the SDA line.

For example, if one transmitter holds the SDA line to a logical one (lets SDA float) and a second transmitter holds it to a logical zero (pulls SDA low), the result is that the SDA line will be low. The first transmitter then observes that the level of the line is different than expected and concludes that another transmitter is communicating.

The first transmitter to notice this difference is the one that loses arbitration and must stop driving the SDA line. If this transmitter is also a master device, it also must stop driving the SCL line. It then can monitor the lines for a Stop condition before trying to reissue its transmission. In the meantime, the other device that has not noticed any difference between the expected and actual levels on the SDA line continues with its original transmission. It can do so without any complications, because so far, the transmission appears exactly as expected with no other transmitter disturbing the message.

Slave Transmit mode can also be arbitrated, when a master addresses multiple slaves, but this is less common.

#### 25.2.2.7 Start Condition

The $I^2C$ specification defines a Start condition as a transition of SDA from a high to a low state while SCL line is high. A Start condition is always generated by the master and signifies the transition of the bus from an Idle to an Active state. Figure 25-12 shows wave forms for Start and Stop conditions.

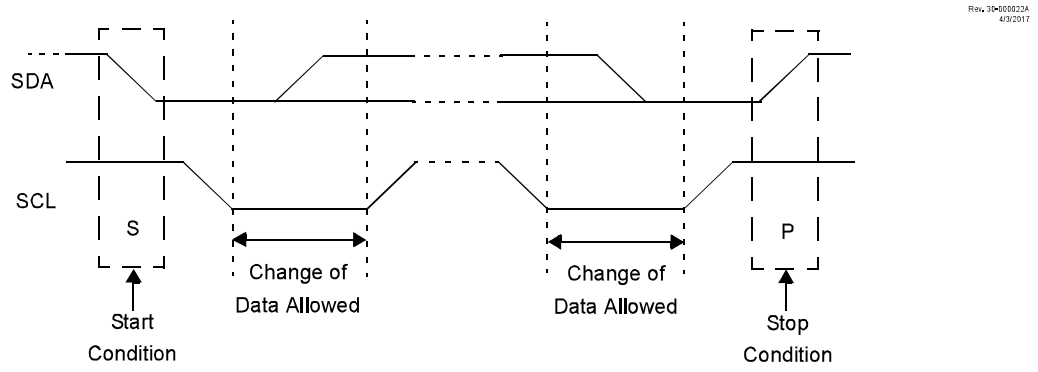
A bus collision can occur on a Start condition if the module samples the SDA line low before asserting it low. This does not conform to the $I^2C$ Specification that states no bus collision can occur on a Start.

#### 25.2.2.8 Stop Condition

A Stop condition is a transition of the SDA line from low-to-high state while the SCL line is high.

**Important:** At least one SCL low time must appear before a Stop is valid, therefore, if the SDA line goes low then high again while the SCL line stays high, only the Start condition is detected.

**Figure 25-12. I²C Start and Stop Conditions**




### 25.2.2.9 Start/Stop Condition Interrupt Masking

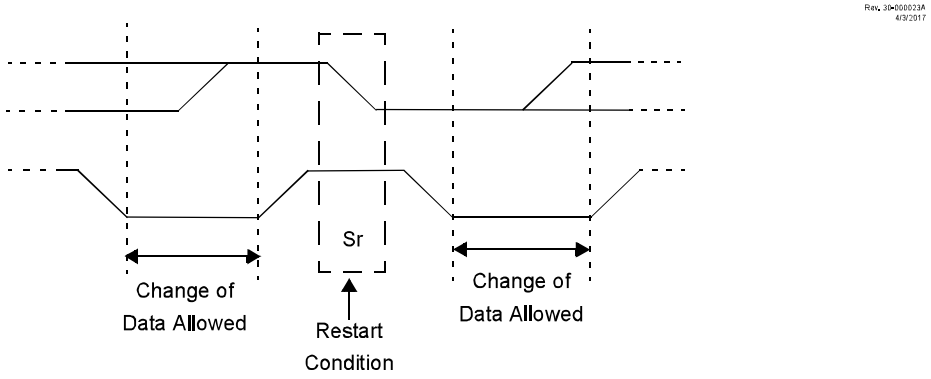
The Start Condition Interrupt Enable (SCIE) and Stop Condition Interrupt Enable (PCIE) bits can enable the generation of an interrupt in Slave modes that do not typically support this function. These bits will have no effect in Slave modes where interrupt on Start and Stop detect are already enabled.

### 25.2.2.10 Restart Condition

A Restart condition 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 25-13 shows the waveform 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.

**Figure 25-13. I²C Restart Condition**




### 25.2.2.11 Acknowledge Sequence

The ninth SCL pulse for any transferred byte in I²C is dedicated as an Acknowledge sequence ($\overline{ACK}$). It allows receiving devices to respond back to the transmitter by pulling the SDA line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge ($\overline{ACK}$) is an active-low signal, pulling the SDA line low indicates 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 Acknowledge Status (ACKSTAT) bit.

The slave software, when the Address Hold Enable (AHEN) and Data Hold Enable (DHEN) bits are set, allows the user to select the $\overline{ACK}$ value sent back to the transmitter. The Acknowledge Data (ACKDT) bit is set/cleared to determine the response.

The slave hardware will generate an $\overline{ACK}$ response under most circumstances. However, if the BF bit or the Receive Overflow Indicator (SSPOV) bit are set when a byte is received then the $\overline{ACK}$ will not be sent by the slave.

When the module is addressed, after the eighth falling edge of SCL on the bus, the Acknowledge Time Status (ACKTIM) bit is set. The ACKTIM bit indicates the acknowledge time of the active bus. The ACKTIM bit is only active when either the AHEN bit or DHEN bit is enabled.

### 25.2.3 I²C Slave Mode Operation

The MSSP Slave mode operates in one of four modes selected by the MSSP Mode Select (SSPM) bits. 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 condition interrupts operate the same as the other modes with SSPxIF additionally getting set upon detection of a Start, Restart, or Stop condition.

#### 25.2.3.1 Slave Mode Addresses

The SSPxADD register 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 SSPxMSK register affects the address matching process. See the "SSP Mask Register" section for more information.

##### 25.2.3.1.1 I²C Slave 7-bit Addressing Mode

In 7-bit Addressing mode, the LSb of the received data byte is ignored when determining if there is an address match.

##### 25.2.3.1.2 I²C Slave 10-bit Addressing Mode

In 10-bit Addressing mode, the first received byte is compared to the binary value of '1 1 1 1 0 A9 A8 0'. A9 and A8 are the two MSbs of the 10-bit address and stored in bits 2 and 1 of the SSPxADD register.

After the acknowledge of the high byte the Update Address (UA) bit is set and SCL is held low until the user updates SSPxADD with the low address. The low address byte is clocked in and all eight bits are compared to the low address value in SSPxADD. Even if there is not an address match; SSPxIF and UA are set, and SCL is held low until SSPxADD is updated to receive a high byte again. When SSPxADD is updated the UA bit is cleared. This ensures the module is ready to receive the high address byte on the next communication.

A high and low address match as a write request is required at the start of all 10-bit addressing communication. A transmission can be initiated by issuing a Restart once the slave is addressed and clocking in the high address with the R/$\overline{W}$ 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.

#### 25.2.3.2 Clock Stretching

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCL clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master that is communicating with the slave will attempt to raise the SCL line in order to transfer the next bit, but will detect that the clock line has not yet been released. Because the SCL connection is open-drain, the slave has the ability to hold that line low until it is ready to continue communicating.

Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

##### 25.2.3.2.1 Normal Clock Stretching

Following an $\overline{ACK}$ if the R/$\overline{W}$ bit 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 Stretch Enable (SEN) bit 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.

##### 25.2.3.2.2 10-bit Addressing Mode

In 10-bit Addressing mode, when the UA bit is set, the clock is always stretched. This is the only time the SCL is stretched without CKP being cleared. SCL is released immediately after a write to SSPxADD.

##### 25.2.3.2.3 Byte NACKing

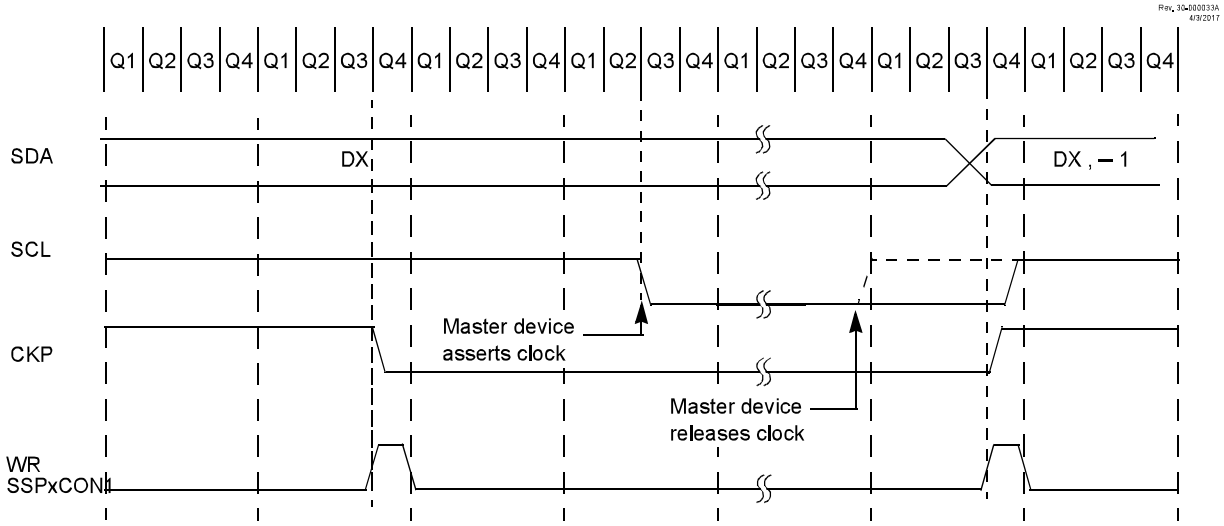
When the AHEN bit is set, CKP is cleared by hardware after the eighth falling edge of SCL for a received matching address byte. When the DHEN bit is set, CKP is cleared after the eighth falling edge of SCL for received data.

Stretching after the eighth falling edge of SCL allows the slave to look at the received address or data and decide if it wants to acknowledge ($\overline{ACK}$) the received address or data, or not acknowledge (NACK) the address or data.

### 25.2.3.3 Clock Synchronization and the CKP bit

Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. However, clearing the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external $I^2C$ master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the $I^2C$ bus have released SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 25-14).

**Figure 25-14. Clock Synchronization Timing**

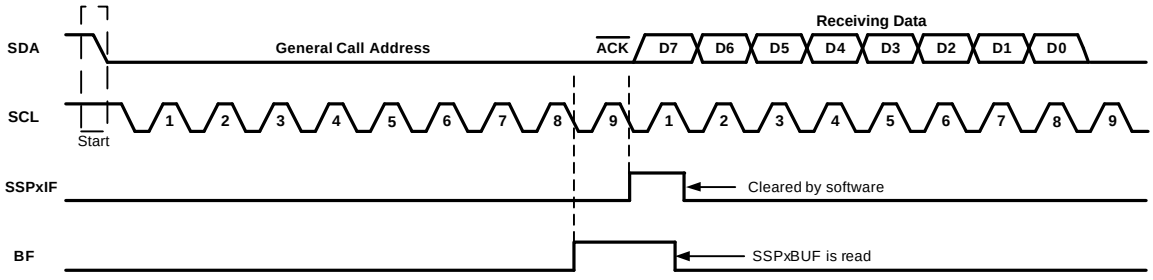



### 25.2.3.4 General Call Address Support

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 that can address all devices. When this address is used, all devices should, in theory, respond with an $\overline{ACK}$.

The general call address is a reserved address in the $I^2C$ protocol, defined as address 0x00. When the General Call Enable (GCEN) bit 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 25-15 shows a general call reception sequence.

**Figure 25-15. Slave Mode General Call Address 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 is set, just as with any other address reception, the slave hardware will stretch the clock after the eighth falling edge of SCL. The slave must then set its Acknowledge Sequence Enable (ACKEN) bit and release the clock with communication progressing as it would normally.

#### 25.2.3.5 SSP Mask Register

The MSSP Mask register (SSPxMSK) is available in I$^2$C Slave mode as a mask for the value held in the SSPSR register during an address comparison operation. A zero ('0') bit in the 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 MSSP operation until written with a mask value.

SSPxMSK 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 MSSP mask has no effect during the reception of the first (high) byte of the address.

#### 25.2.3.6 Slave Reception

When the R/$\overline{W}$ bit of a matching received address byte is clear, the R/$\overline{W}$ bit is cleared. The received address is loaded into the SSPxBUF register and acknowledged.

When the overflow condition exists for a received address, a Not Acknowledge (NACK) is transmitted and the Receive Overflow Indicator (SSPOV) bit is set. The Buffer Override Enable (BOEN) bit modifies this operation.

An MSSP interrupt is generated for each transferred data byte. The SSPxIF flag bit must be cleared by software.

When the SEN bit is set, SCL will be held low (clock stretch) following each received byte. The clock must be released by setting the CKP bit, except sometimes in 10-bit mode. See "10-Bit Addressing Mode" for more details.

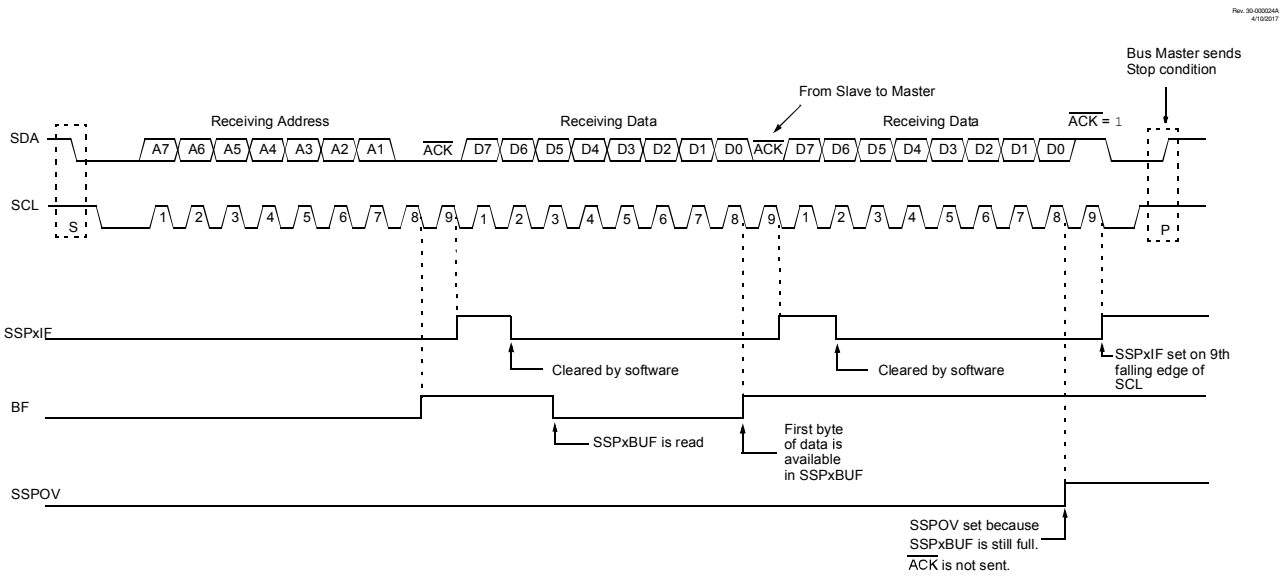
##### 25.2.3.6.1 7-bit Addressing Reception

This section describes a standard sequence of events for the MSSP module configured as an I$^2$C slave in 7-bit Addressing mode. Figure 25-16 and Figure 25-17 are used as a visual reference for this description.

This is a step by step process of what typically must be done to accomplish I$^2$C communication.

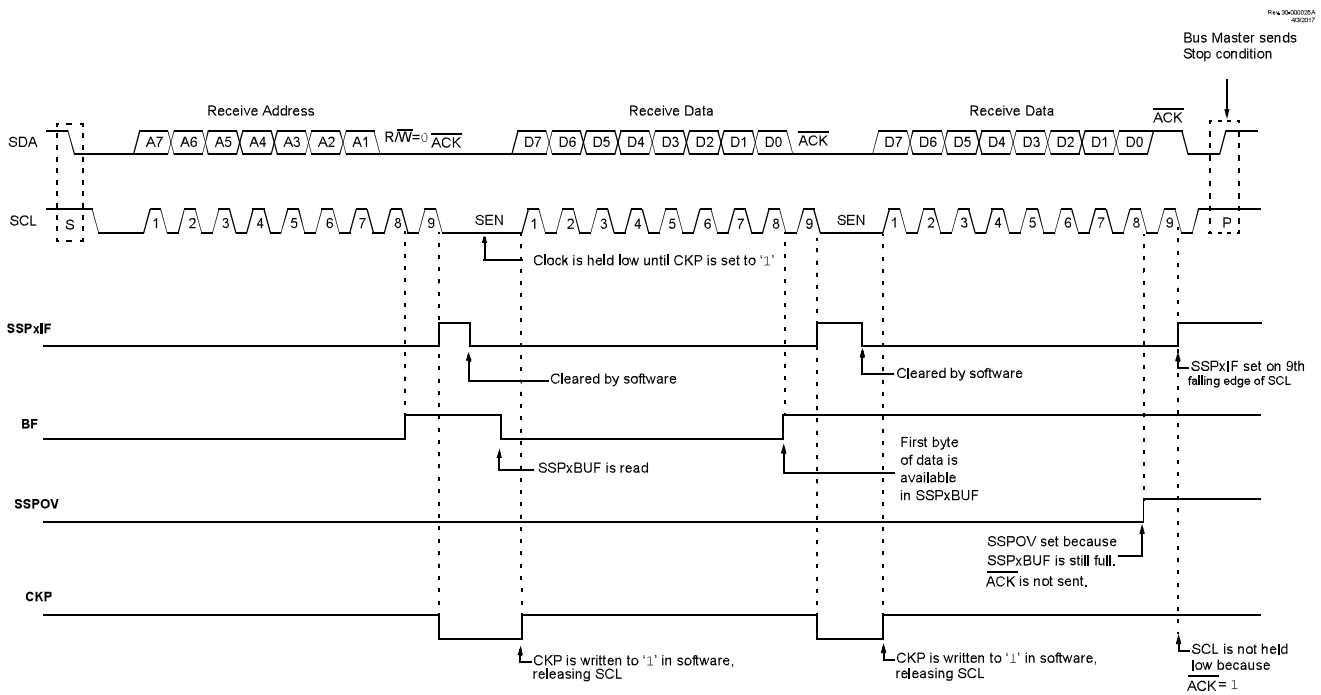
1. Start condition detected.
2. The Start (S) bit is set; SSPxIF is set if the Start Condition Interrupt Enable (SCIE) bit is set.
3. Matching address with R/$\overline{W}$ bit clear is received.
4. The slave pulls SDA low, sending an $\overline{ACK}$ to the master, and sets SSPxIF bit.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPxBUF, clearing the BF flag.
7. If SEN = 1; Slave software sets the CKP bit to release the SCL line.
8. The master clocks out a data byte.
9. Slave drives SDA 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 the Stop (P) bit, and the bus goes Idle.

**Figure 25-16. I²C Slave, 7-bit Address, Reception (SEN = 0, AHEN = 0, DHEN = 0)**




© 2020 Microchip Technology Inc.

**Figure 25-17. I²C Slave, 7-bit Address, Reception (SEN = 1, AHEN = 0, DHEN = 0)**




© 2020 Microchip Technology Inc.

#### 25.2.3.6.2 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 SCL. 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.

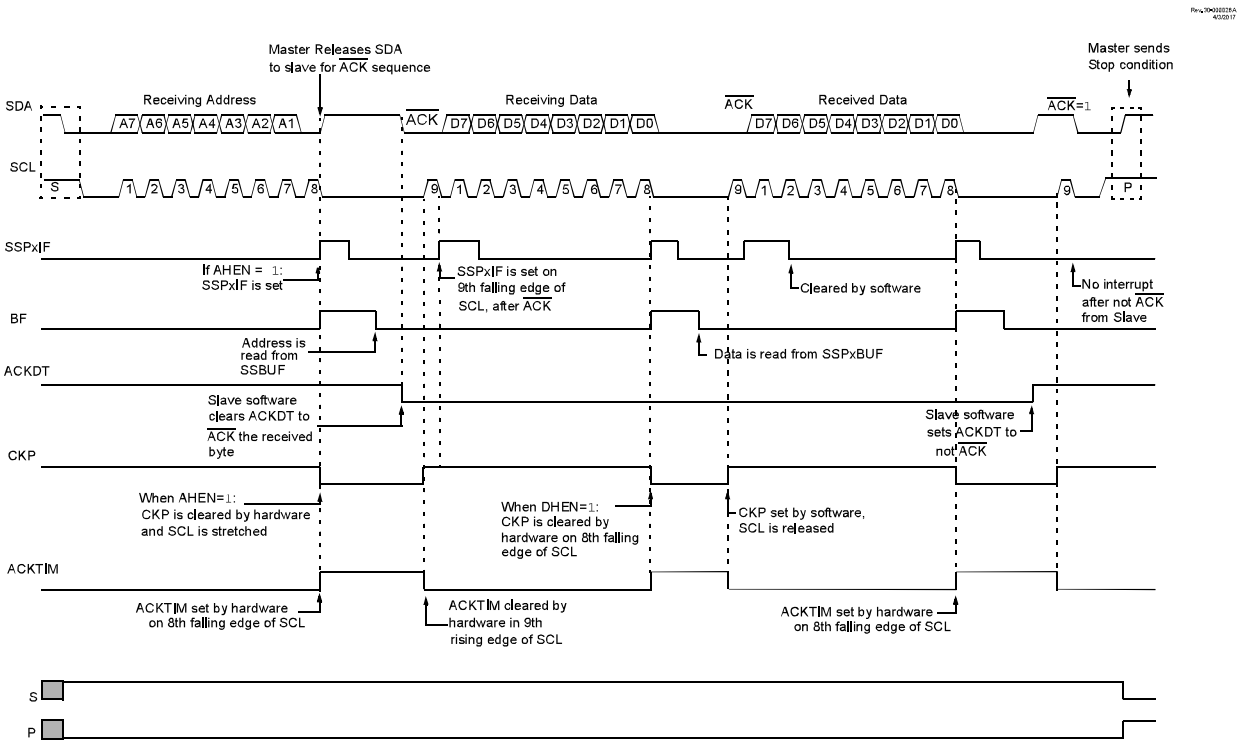
This list describes the steps that need to be taken by slave software to use these options for $I^2C$ communication. Figure 25-18 displays a module using both address and data holding. Figure 25-19 includes the operation with the SEN bit set.

1. The Start (S) bit is set; SSPxIF is set if SCIE is set.
2. Matching address with the R/$\overline{W}$ bit clear is clocked in. SSPxIF is set and CKP cleared after the eighth falling edge of SCL.
3. Software clears the SSPxIF.
4. Slave can look at the ACKTIM bit 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 transmits an $\overline{ACK}$ to the master by clearing 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.

**Important:** SSPxIF is still set after the ninth falling edge of SCL even if there is no clock stretching and BF has been cleared. Only if a NACK is sent to the master is SSPxIF not set.

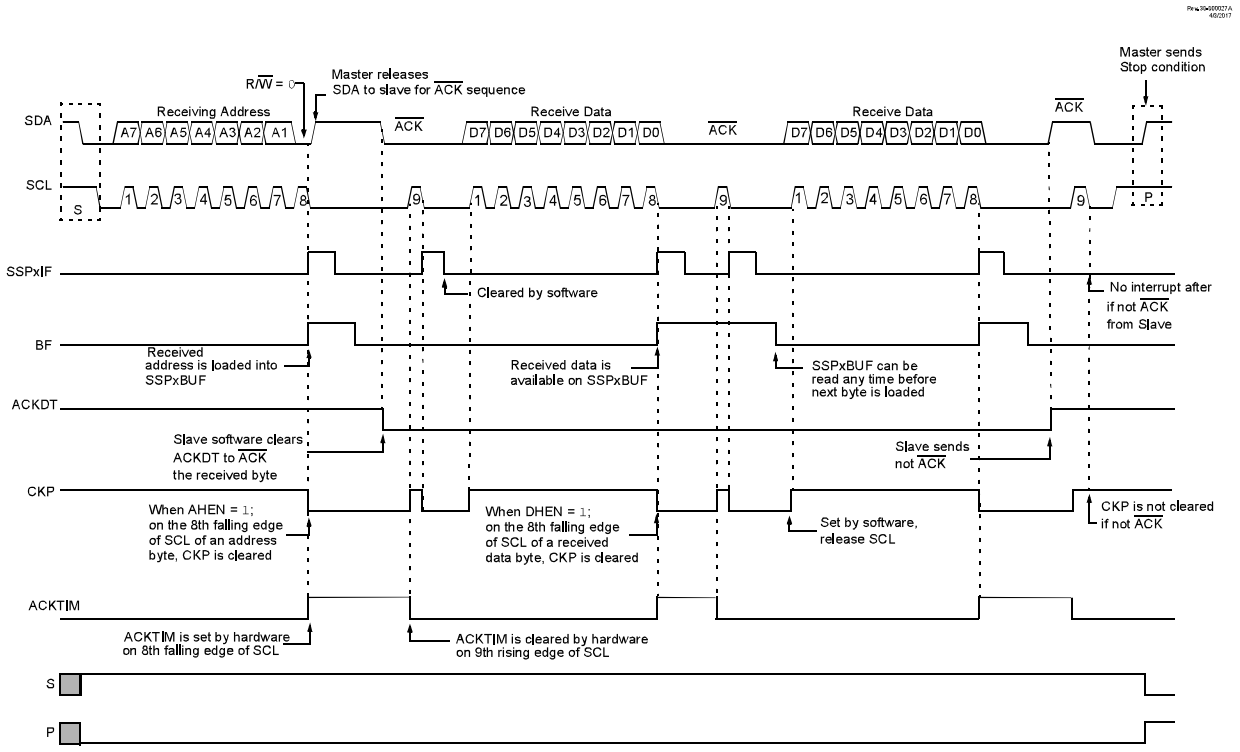
11. SSPxIF is set and CKP cleared after eighth falling edge of SCL for a received data byte.
12. Slave looks at the ACKTIM bit to determine the source of the interrupt.
13. Slave reads the received data from SSPxBUF, clearing BF.
14. Steps 7-14 are the same for each received data byte.
15. Communication is ended by either the slave sending a NACK, or the master sending a Stop condition. If a Stop is sent and the Stop Condition Interrupt Enable (PCIE) bit is clear, the slave will only know by polling the Stop (P) bit.

**Figure 25-18. I²C Slave, 7-bit Address, Reception (SEN = 0, AHEN = 1, DHEN = 1)**




© 2020 Microchip Technology Inc.

**Figure 25-19. I²C Slave, 7-bit Address, Reception (SEN = 1, AHEN = 1, DHEN = 1)**




© 2020 Microchip Technology Inc.

#### 25.2.3.6.3 Slave Mode 10-bit Address Reception

This section describes a standard sequence of events for the MSSP module configured as an $I^2C$ slave in 10-bit Addressing mode. Figure 25-20 shows a standard waveform for a slave receiver in 10-bit Addressing mode with clock stretching enabled.

This is a step-by-step process of what must be done by the slave software to accomplish $I^2C$ communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit is set; SSPxIF is set if SCIE is set.
3. Master sends matching high address with the R/$\overline{W}$ bit clear; the UA bit 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 SCL.
8. Master sends matching low address byte to the slave; UA bit is set.

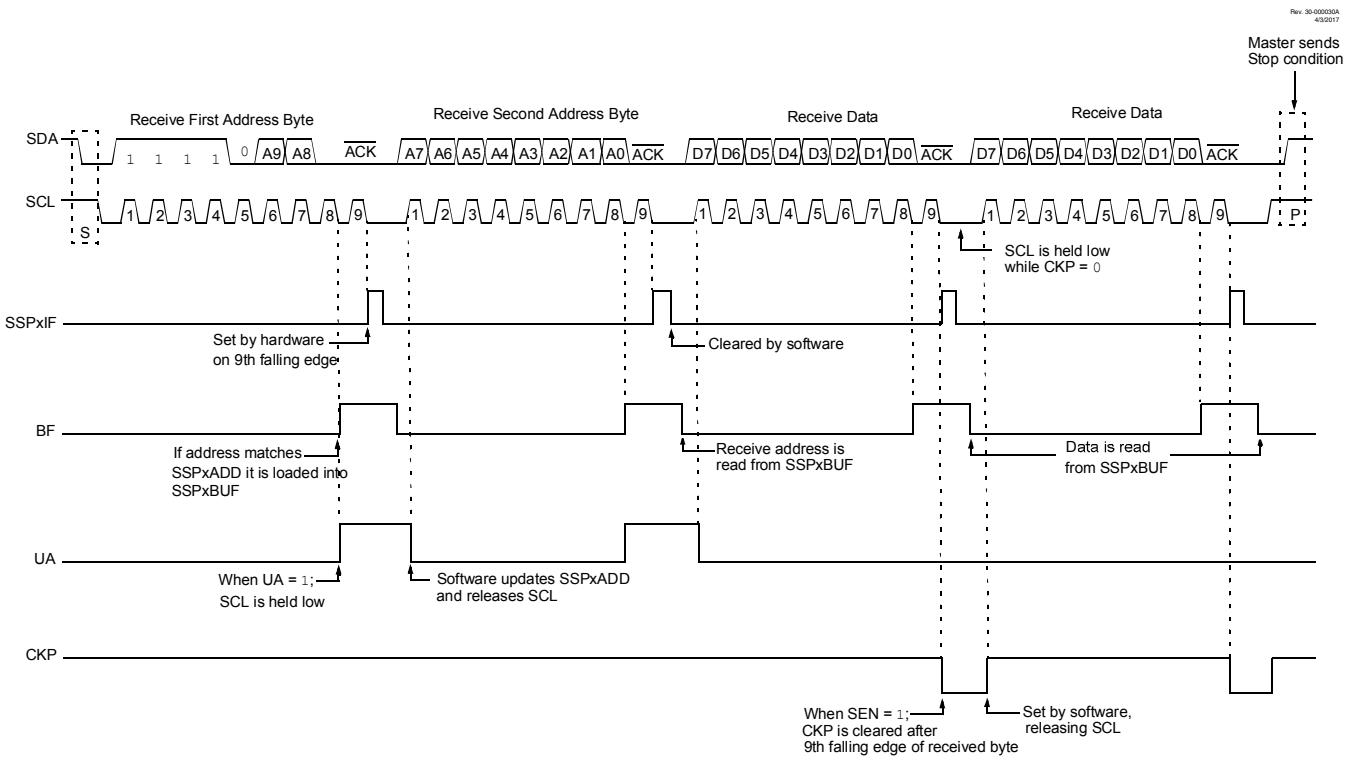
**Important:** Updates to the SSPxADD register are not allowed until after the $\overline{ACK}$ sequence.

9. Slave sends $\overline{ACK}$ and SSPxIF is set.

**Important:** 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 SCL pulse; SSPxIF is set.
14. If the SEN bit 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 software sets CKP to release the SCL.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

**Figure 25-20. I²C Slave, 10-bit Address, Reception (SEN = 1, AHEN = 0, DHEN = 0)**



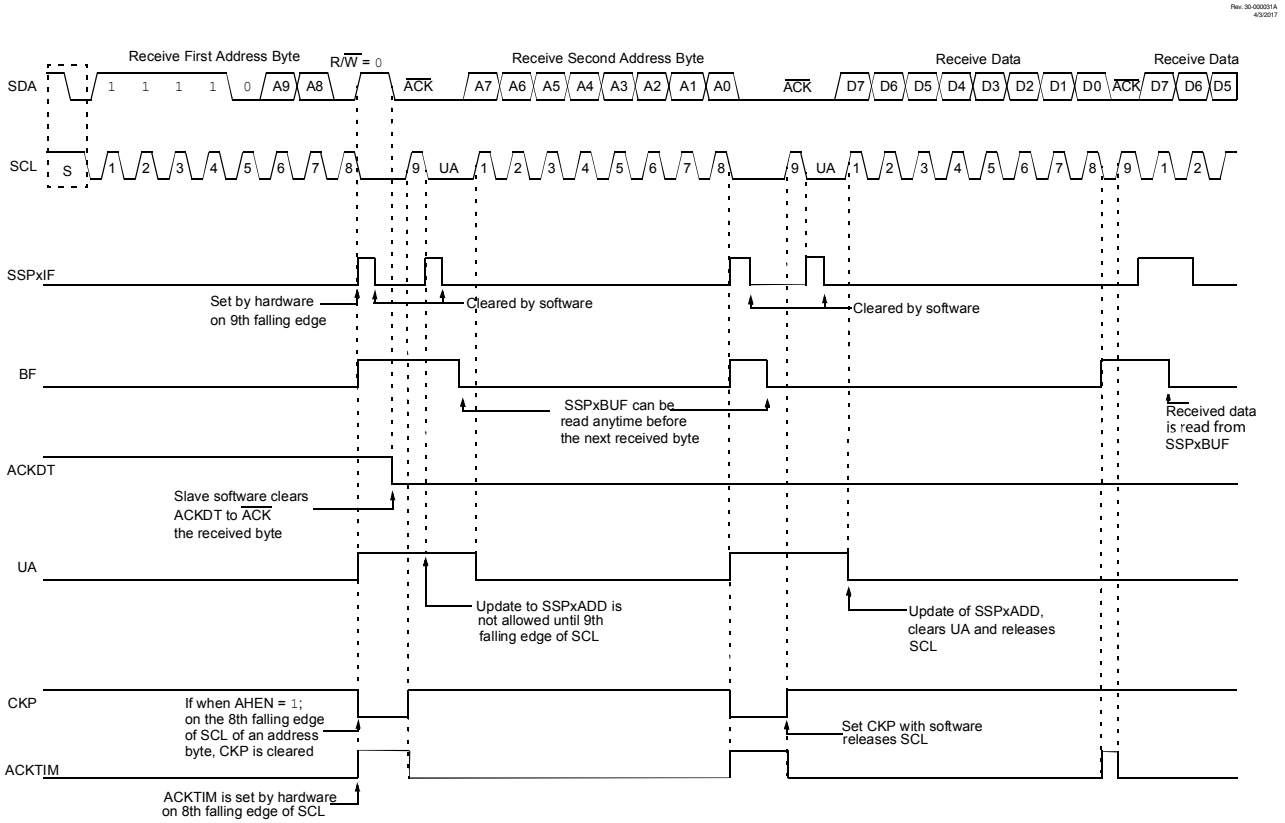

© 2020 Microchip Technology Inc.

#### 25.2.3.6.4 10-bit Addressing with Address or Data Hold

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPxADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCL line is held low are the same. Figure 25-21 can be used as a reference of a slave in 10-bit addressing with AHEN set.

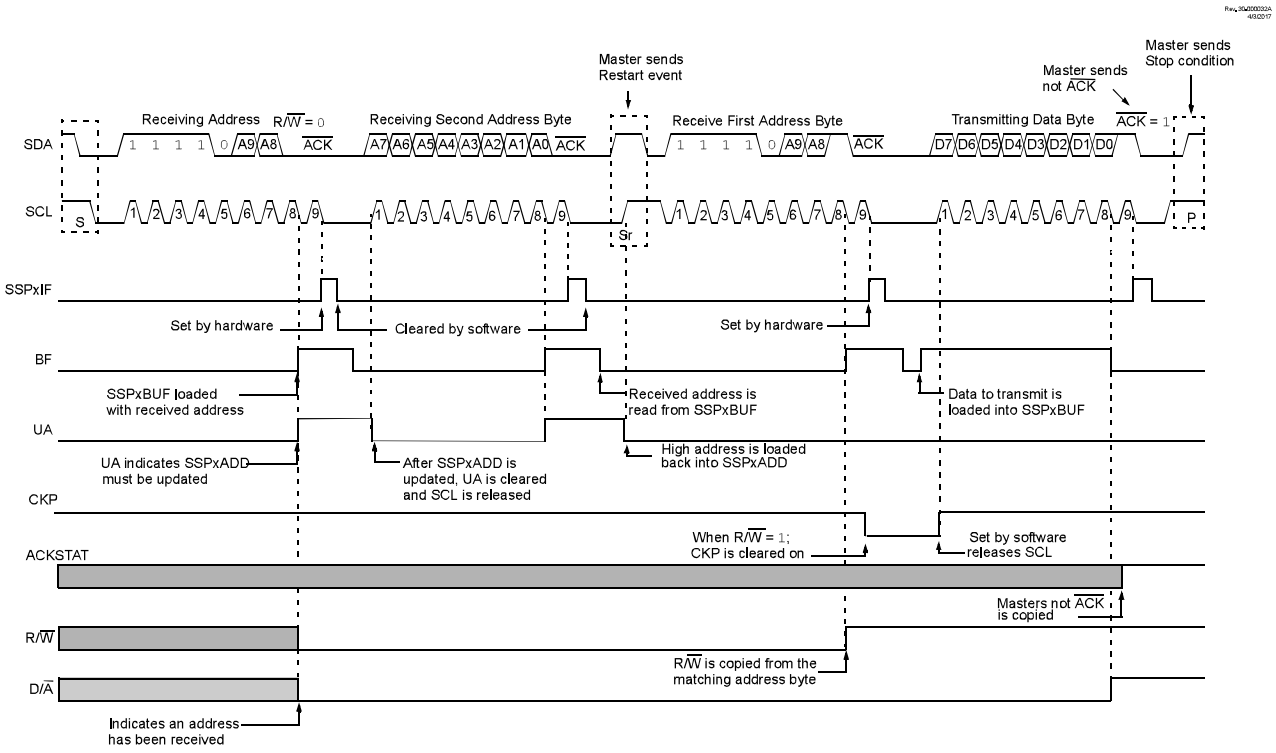
Figure 25-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

**Figure 25-21. I²C Slave, 10-bit Address, Reception (SEN = 0, AHEN = 1, DHEN = 0)**




© 2020 Microchip Technology Inc.

**Figure 25-22. I²C Slave, 10-bit Address, Transmission (SEN = 0, AHEN = 0, DHEN = 0)**




© 2020 Microchip Technology Inc.

#### 25.2.3.7 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 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 SCL pin is held low (see "Clock Stretching" for more details). 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 SSPSR register. Then the SCL pin should be released by setting the CKP bit. The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time.

The $\overline{ACK}$ pulse from the master receiver is latched on the rising edge of the ninth SCL input pulse. This $\overline{ACK}$ value is copied to the ACKSTAT bit. If ACKSTAT is set (NACK), then the data transfer is complete. In this case, when the NACK is latched by the slave, the slave goes Idle and waits for another occurrence of a Start condition. If the SDA line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPxBUF register. Again, the SCL pin must be released by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The 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.

##### 25.2.3.7.1 Slave Mode Bus Collision

A slave receives a read request and begins shifting data out on the SDA line. If a bus collision is detected and the Slave Mode Bus Collision Detect Enable (SBCDE) bit is set, the Bus Collision Interrupt Flag (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.

##### 25.2.3.7.2 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 25-23 can be used as a reference to this list.

1. Master sends a Start condition.
2. The Start (S) bit is set; SSPxIF is set if SCIE is set.
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. The SSPxIF bit is cleared by software.
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 by software, releasing SCL, 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 bit.
11. SSPxIF bit is cleared.
12. The slave software checks the ACKSTAT bit to see if the master wants to clock out more data.

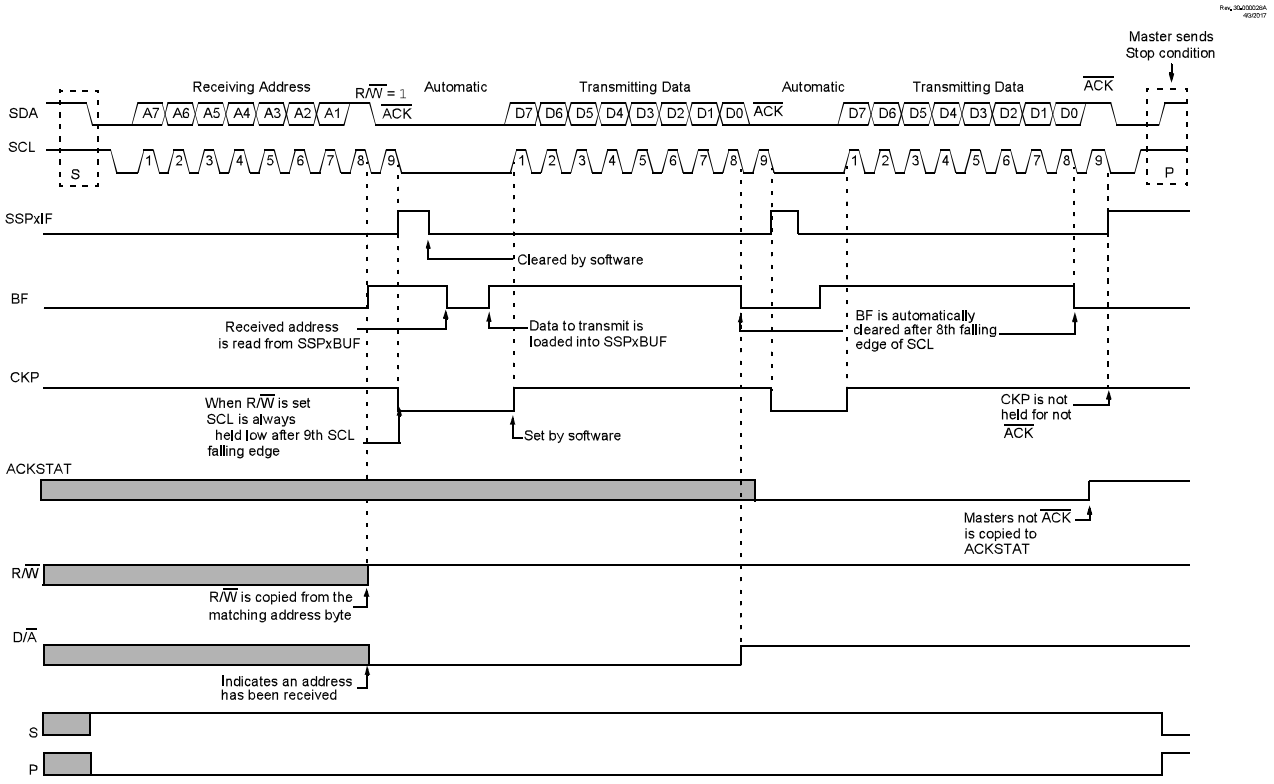


**Important:**

1. If the master $\overline{ACK}$s then the clock will be stretched.
2. ACKSTAT is the only bit updated on the rising edge of the ninth SCL clock instead of the falling edge.

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.

**Figure 25-23. I²C Slave, 7-bit Address, Transmission (AHEN = 0)**




© 2020 Microchip Technology Inc.

#### 25.2.3.7.3 7-bit Transmission with Address Hold Enabled

Setting the AHEN bit 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 25-24 displays a standard waveform of a 7-bit address slave transmission with AHEN enabled.

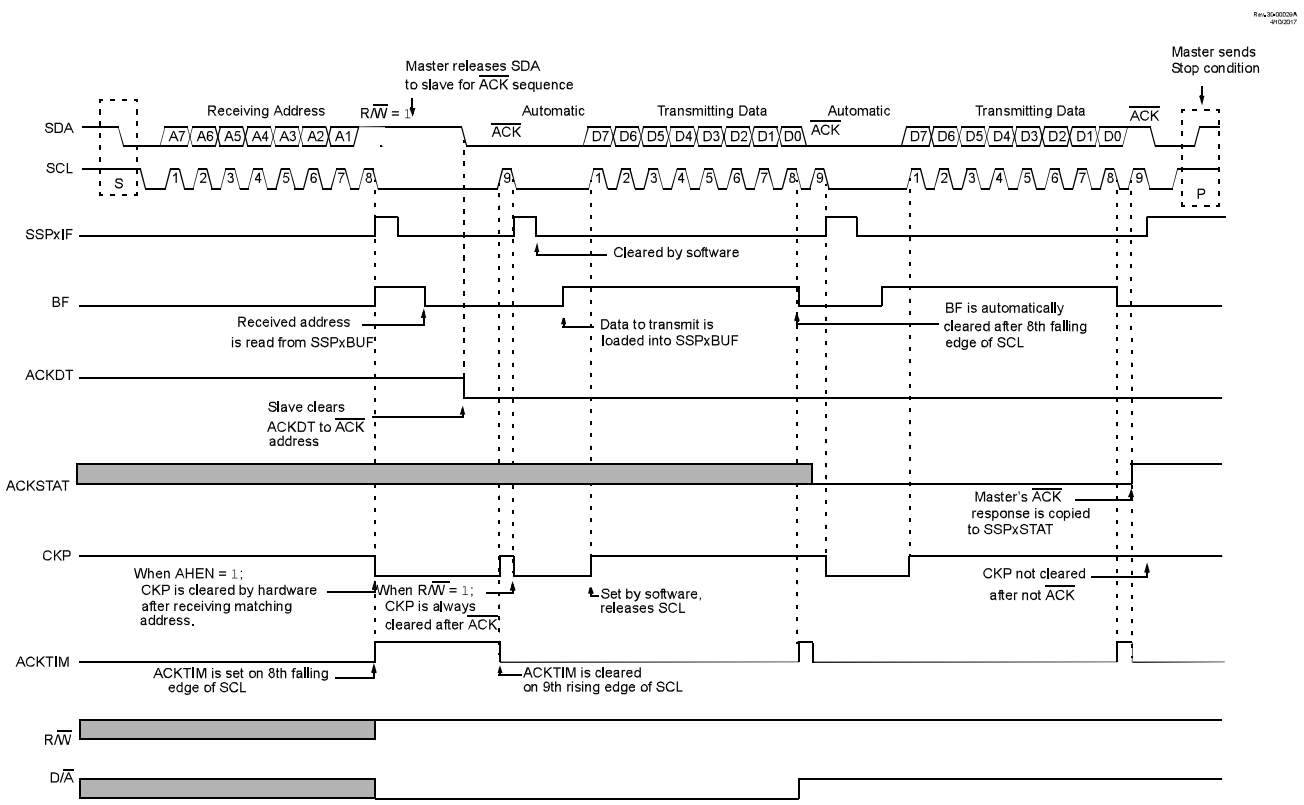
1. Bus starts Idle.
2. Master sends Start condition; the S bit is set; SSPxIF is set if SCIE is set.
3. Master sends matching address with the R/$\overline{W}$ bit set. After the eighth falling edge of the SCL line the CKP bit is cleared and SSPxIF interrupt is generated.
4. Slave software clears SSPxIF.
5. Slave software reads the ACKTIM, R/$\overline{W}$, and D/$\overline{A}$ bits 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 NACK and sets the ACKDT bit accordingly.
8. Slave software sets the CKP bit, releasing SCL.
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.

**Important:** SSPxBUF cannot be loaded until after the $\overline{ACK}$.

13. Slave software sets the CKP bit, releasing the clock.
14. Master clocks out the data from the slave and sends an $\overline{ACK}$ value on the ninth SCL pulse.
15. Slave hardware copies the $\overline{ACK}$ value into the ACKSTAT bit.
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.

**Important:** Master must send a not $\overline{ACK}$ on the last byte to ensure that the slave releases the SCL line to receive a Stop.

**Figure 25-24. I²C Slave, 7-bit Address, Transmission (AHEN = 1)**




© 2020 Microchip Technology Inc.

### 25.2.4 I²C Master Mode

Master mode is enabled by configuring the appropriate SSPM bits and setting the SSPEN bit. In Master mode, the SDA and SCL pins must be configured as inputs. The MSSP peripheral hardware will override the output driver TRIS controls when necessary to drive the pins low.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I²C bus may be taken when the P bit is set, or the bus is Idle.

In Firmware Controlled Master mode, user code conducts all I²C bus operations based on Start and Stop condition detection. Start and Stop condition detection is the only active circuitry in this mode. All other communication is done by the user software directly manipulating the SDA and SCL lines.

The following events will cause the MSSP Interrupt Flag bit (SSPxIF) to be set (MSSP interrupt, if enabled):

- Start condition detected.
- Stop condition detected.
- Data transfer byte transmitted/received.
- Acknowledge transmitted/received.
- Repeated Start generated.

**Important:**

1. The MSSP module, when configured in I²C Master mode, does not allow queuing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the Write Collision Detect (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 SSPxIF bit is set at the end of the Start/Stop generation when hardware clears the control bit.

#### 25.2.4.1 I²C Master Mode Operation

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

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

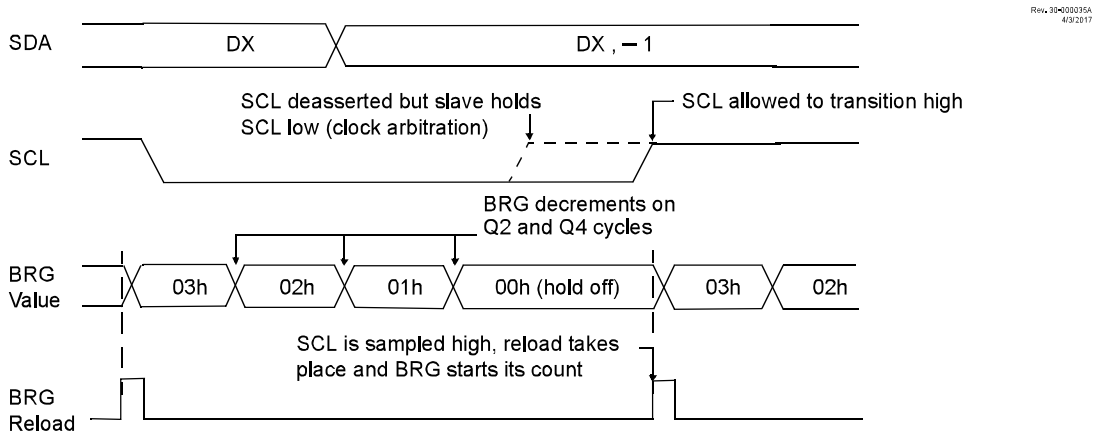
In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive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

A Baud Rate Generator is used to set the clock frequency output on SCL. See "Baud Rate Generator" for more details.

##### 25.2.4.1.1 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, releases the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device as shown in Figure 25-25.

**Figure 25-25. Baud Rate Generator Timing with Clock Arbitration**




#### 25.2.4.1.2 WCOL Status Flag

If the user writes the SSPxBUF when a Start, Restart, Stop, Receive or Transmit sequence is in progress, the Write Collision Detect (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.

**Important:** Because queuing of events is not allowed, writing to the lower five bits of SSPxCON2 is disabled until the Start condition is complete.

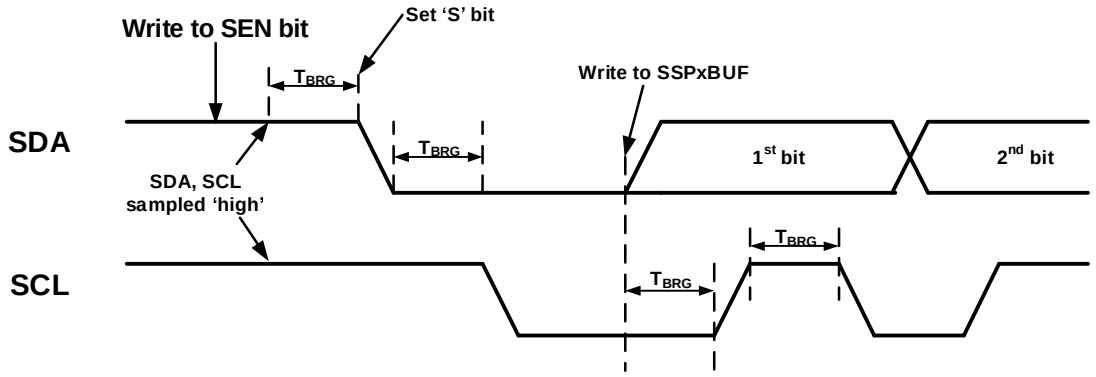
#### 25.2.4.1.3 $I^2C$ Master Mode Start Condition Timing

To initiate a Start condition (see Figure 25-26), the user sets the Start Condition Enable (SEN) bit. If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out ($T_{BRG}$), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the Start (S) bit to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPxADD and resumes its count. When the Baud Rate Generator times out ($T_{BRG}$), the SEN bit will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

**Important:**

1. If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag (BCLxIF) is set, the Start condition is aborted and the $I^2C$ module is reset into its Idle state.
2. The Philips $I^2C$ Specification states that a bus collision cannot occur on a Start.

**Figure 25-26. First Start Bit Timing**



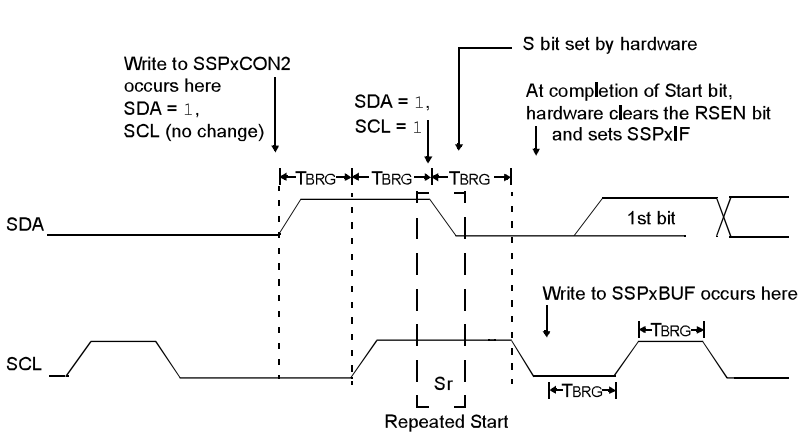

#### 25.2.4.1.4 I²C Master Mode Repeated Start Condition Timing

A Repeated Start condition (see Figure 25-27) occurs when the Repeated Start Condition Enable (RSEN) bit is programmed high and the master state machine is no longer active. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count ($T_{BRG}$). When the Baud Rate Generator times out, if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded and begins counting. SDA and SCL must be sampled high for one $T_{BRG}$. This action is then followed by assertion of the SDA pin (SDA = 0) for one $T_{BRG}$ while SCL is high. SCL is asserted low. Following this, the RSEN bit will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit will be set. The SSPxIF bit will not be set until the Baud Rate Generator has timed out.

**Important:**

1. If RSEN is programmed while any other event is in progress, it will not take effect.
2. A bus collision during the Repeated Start condition occurs if:
   - SDA is sampled low when SCL goes from low-to-high.
   - SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data '1'.

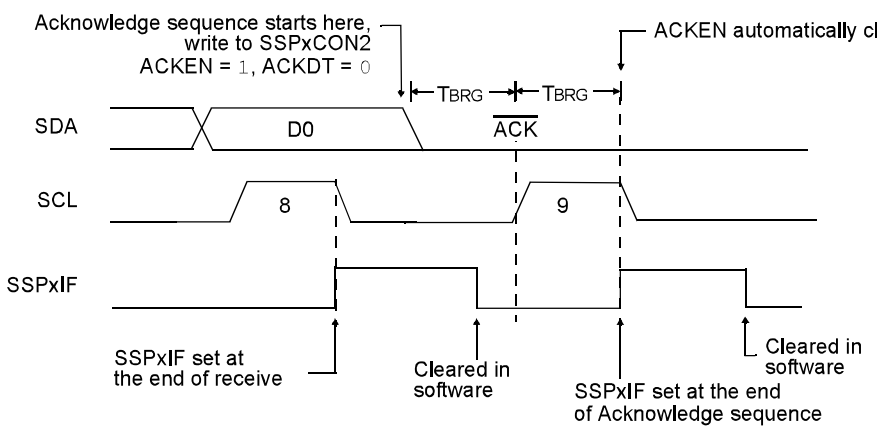
**Figure 25-27. Repeated Start Condition Waveform**




### 25.2.4.1.5 Acknowledge Sequence Timing

An Acknowledge sequence (see Figure 25-28) is enabled by setting the Acknowledge Sequence Enable (ACKEN) bit. When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge Data (ACKDT) bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period ($T_{BRG}$) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for $T_{BRG}$. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into Idle mode.

**Figure 25-28. Acknowledge Sequence Waveform**




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

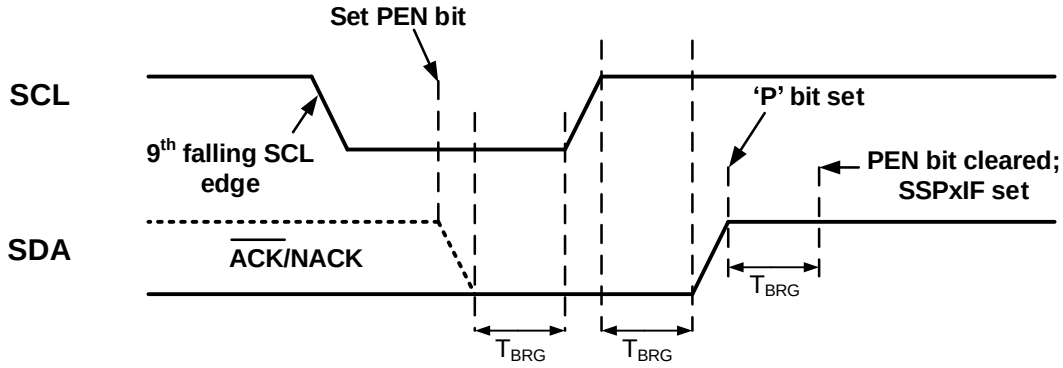
***Acknowledge Write Collision***

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).

### 25.2.4.1.6 Stop Condition Timing

A Stop condition (see Figure 25-29) is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Condition Enable (PEN) bit. At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCL pin will be brought high and one $T_{BRG}$ (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit is set. One $T_{BRG}$ later, the PEN bit is cleared and the SSPxIF bit is set.

**Figure 25-29. Stop Condition in Receive or Transmit Mode**




***Write Collision on Stop***

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).

#### 25.2.4.1.7 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 MSSP interrupt is enabled).

#### 25.2.4.1.8 Effects of a Reset

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

### 25.2.4.2 $I^2C$ 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 Status (BF) bit and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted. SCL is held low for one Baud Rate Generator rollover count ($T_{BRG}$). Data should be valid before SCL is released high. When the SCL pin is released high, it is held that way for $T_{BRG}$. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an $\overline{ACK}$ sequence 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 Acknowledge Status (ACKSTAT) bit on the rising edge of the ninth clock. If the master receives an $\overline{ACK}$, the ACKSTAT bit is cleared. If a NACK is received, ACKSTAT 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 SCL low and SDA unchanged (see Figure 25-30).

After the write to the SSPxBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/$\overline{W}$ bit are completed. On the falling edge of the eighth clock, the master will release the SDA pin, allowing the slave to respond with an $\overline{ACK}$. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the $\overline{ACK}$ bit is loaded into the ACKSTAT bit. 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 SCL low and allowing SDA to float.

#### 25.2.4.2.1 BF Status Flag

In Transmit mode, the Buffer Full Status (BF) bit is set when the CPU writes to SSPxBUF, and is cleared when all eight bits are shifted out.

#### 25.2.4.2.2 WCOL Status Flag

If the user writes the SSPxBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the Write Collision Detect (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.

#### 25.2.4.2.3 ACKSTAT Status Flag

In Transmit mode, the Acknowledge Status (ACKSTAT) bit is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0), and is set when the slave issues a NACK. A slave sends an $\overline{ACK}$ when it has recognized its address (including a General Call), or when the slave has properly received its data.

#### 25.2.4.2.4 Typical Transmit Sequence:

1. The Master generates a Start condition by setting the SEN bit.
2. SSPxIF is set by hardware on completion of the Start.
3. SSPxIF is cleared by software.
4. The MSSP module will wait the required start time before any other operation takes place.
5. Software loads the SSPxBUF with the slave address and the R/$\overline{W}$ bit. In Master Transmit mode, the R/$\overline{W}$ value is zero.
6. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
7. The MSSP module shifts in the $\overline{ACK}$ value from the slave device and writes its into the ACKSTAT bit.
8. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
9. Software loads the SSPxBUF with eight bits of data.
10. Data is shifted out the SDA pin until all eight bits are transmitted.
11. The MSSP module shifts in the $\overline{ACK}$ bit from the slave device and writes its value into the ACKSTAT bit.
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, respectively. An Interrupt is generated once the Stop/Restart condition is complete.

**Figure 25-30. I²C Master Mode Waveform (Transmission, 7-bit Address)**

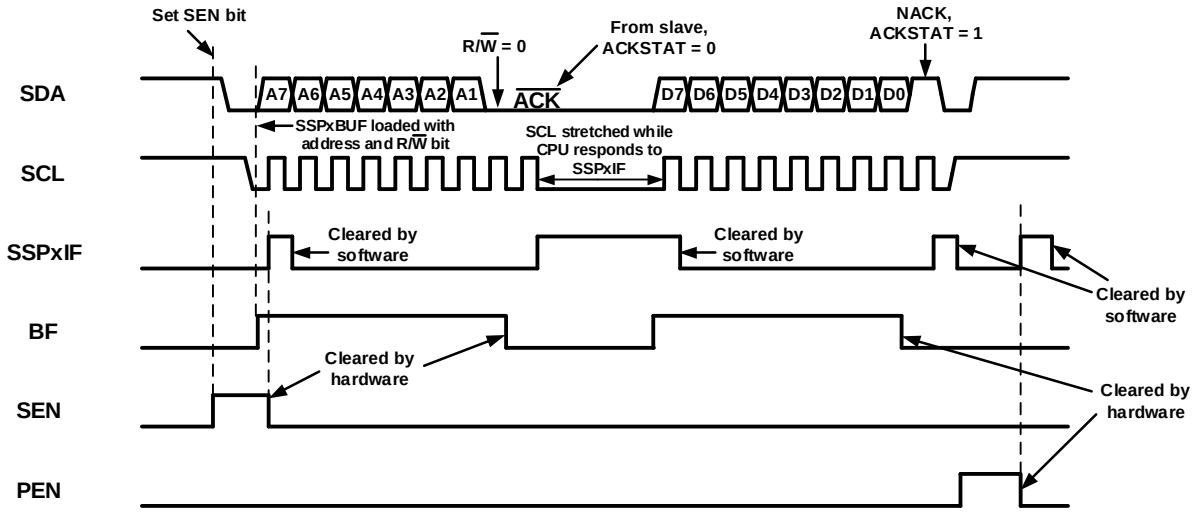
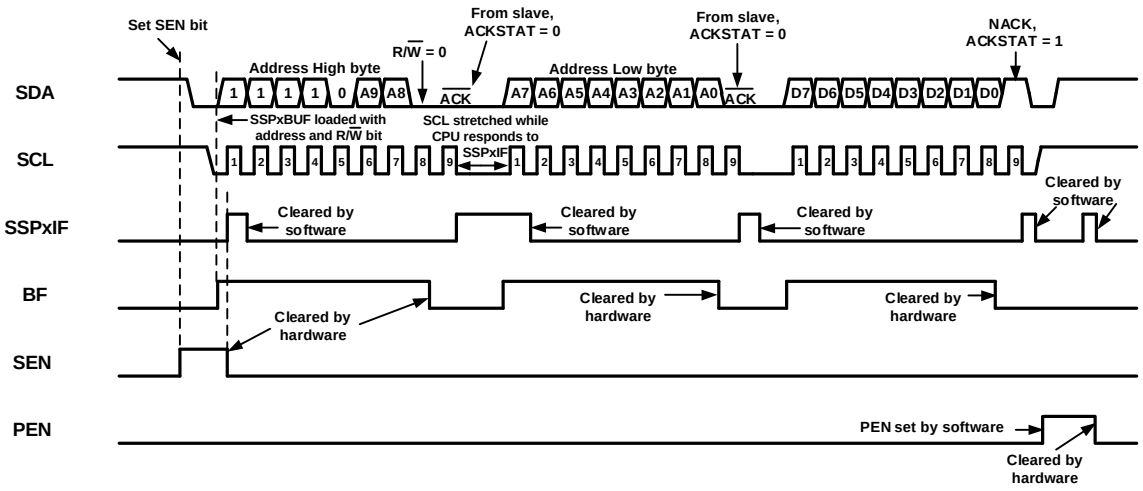


**Figure 25-31. I²C Master Mode Waveform (Transmission, 10-bit Address)**




### 25.2.4.3 I²C Master Mode Reception

Master mode reception (see Figure 25-32) is enabled by setting the Receive Enable (RCEN) bit.

**Important:** The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPxSR. After the falling edge of the eighth clock all the following events occur:

- RCEN is automatically cleared by hardware.
- The contents of the SSPxSR are loaded into the SSPxBUF.
- The BF flag bit is set.
- The SSPxIF flag bit is set.
- The Baud Rate Generator is suspended from counting.
- The SCL pin is held low.

The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The Master can then send an Acknowledge sequence at the end of reception by setting the Acknowledge Sequence Enable (ACKEN) bit.

#### 25.2.4.3.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPxBUF from SSPSR. It is cleared when the SSPxBUF register is read.

#### 25.2.4.3.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when eight bits are received into SSPxSR while the BF flag bit is already set from a previous reception.

#### 25.2.4.3.3 WCOL Status Flag

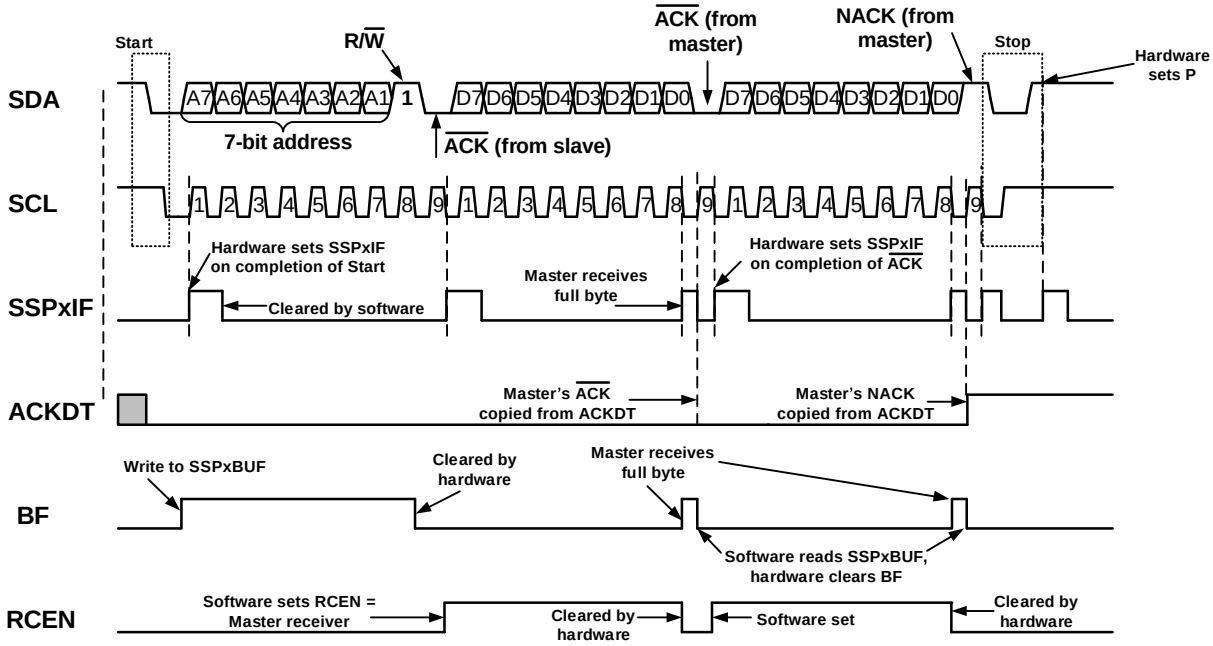
If the user writes the SSPxBUF when a receive is already in progress (i.e., SSPxSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

#### 25.2.4.3.4 Typical Receive Sequence:

1. The Master generates a Start condition by setting the SEN bit.
2. SSPxIF is set by hardware on completion of the Start.

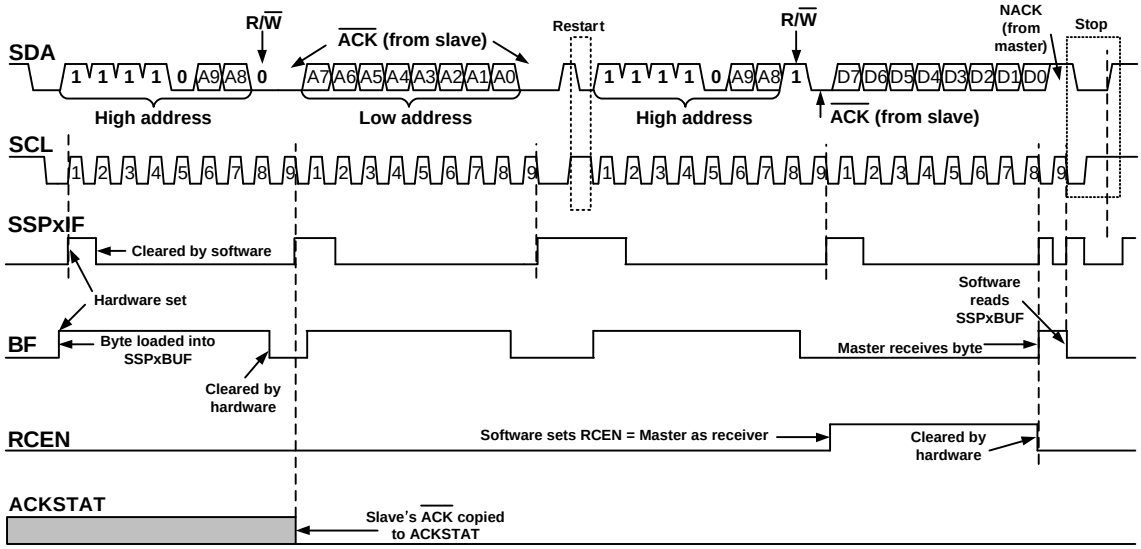
3. SSPxIF is cleared by software.
4. Software writes SSPxBUF with the slave address to transmit and the R/$\overline{W}$ bit set.
5. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
6. The MSSP module shifts in the $\overline{ACK}$ value from the slave device and writes it into the ACKSTAT bit.
7. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
8. Software sets the RCEN bit and the master clocks in a byte from the slave.
9. After the eighth falling edge of SCL, SSPxIF and BF are set.
10. Master clears SSPxIF and reads the received byte from SSPxBUF, which clears BF.
11. Master clears the ACKDT bit and initiates the $\overline{ACK}$ sequence by setting the ACKEN bit.
12. Master's $\overline{ACK}$ 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 NACK or Stop to end communication.

Figure 25-32. I²C Master Mode Waveform (Reception, 7-bit Address)




© 2020 Microchip Technology Inc.

**Figure 25-33. I²C Master Mode Waveform (Reception, 10-bit Address)**




### 25.2.5 Multi-Master Mode

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I²C bus may be taken when the P bit is set, or the bus is Idle, with both the S and P bits cleared. When the bus is busy, enabling the MSSP interrupt will generate an interrupt when the Stop condition occurs.

In Multi-Master operation, the SDA line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLxIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

#### 25.2.5.1 Multi-Master Communication, Bus Collision and Bus Arbitration

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag (BCLxIF) and reset the I²C port to its Idle state (see Figure 25-34).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are deasserted, and the SSPxBUF can be written to. When software services the Bus Collision Interrupt Service Routine, and if the I²C bus is free, software can resume communication by asserting a Start condition.

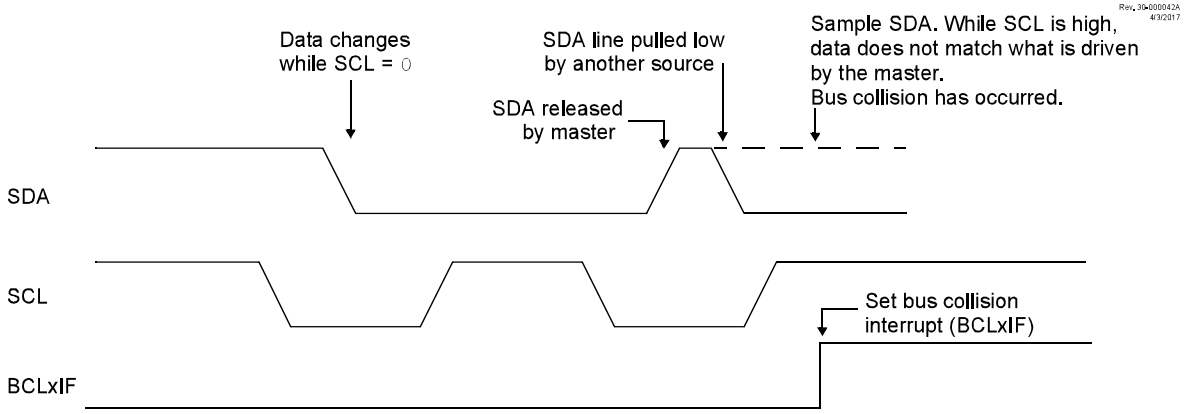
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted, and the respective control bits in the SSPxCON2 register are cleared. When software services the bus collision Interrupt Service Routine, and if the I²C bus is free, software can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the 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, or the bus is Idle and the S and P bits are cleared.

**Figure 25-34. Bus Collision Timing for Transmit and Acknowledge**




#### 25.2.5.1.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

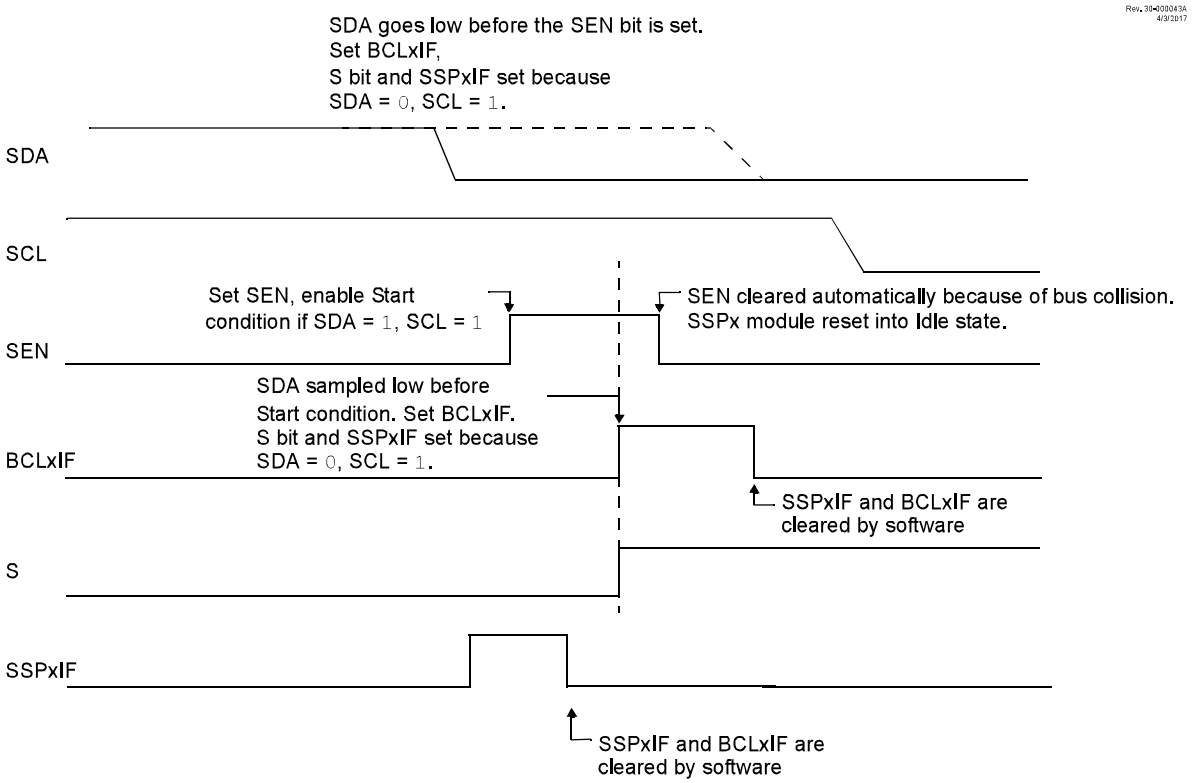
1. SDA or SCL are sampled low at the beginning of the Start condition (see Figure 25-35).
2. SCL is sampled low before SDA is asserted low (see Figure 25-36).

During a Start condition, both the SDA and the SCL pins are monitored.

If the SDA pin is already low, or the SCL pin is already low, then all of the following occur:

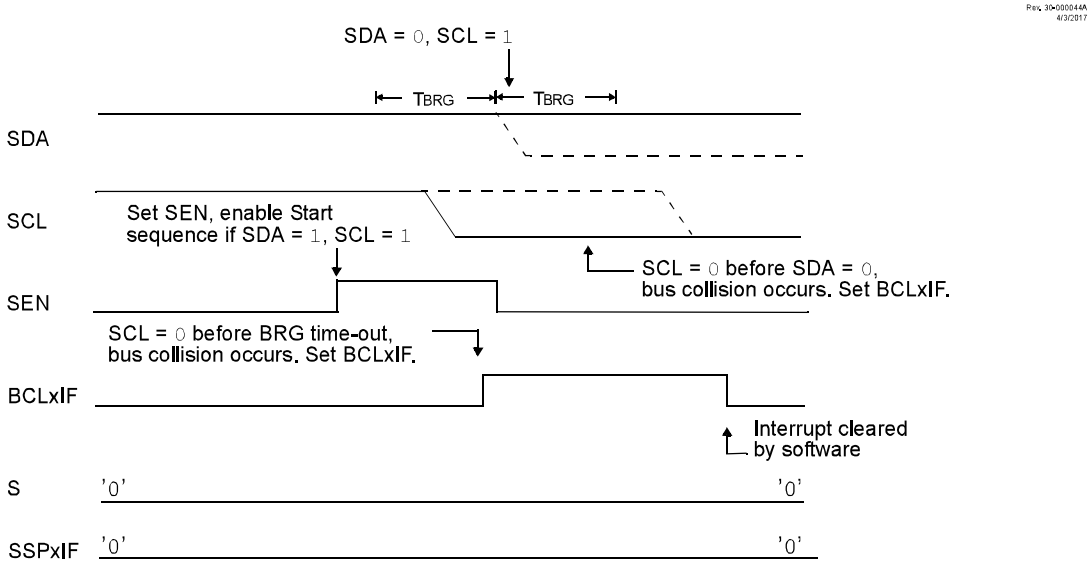
- the Start condition is aborted,
- the BCLxIF flag is set and
- the MSSP module is reset to its Idle state (see Figure 25-35).

**Figure 25-35. Bus Collision During Start Condition (SDA Only)**




The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded and counts down. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

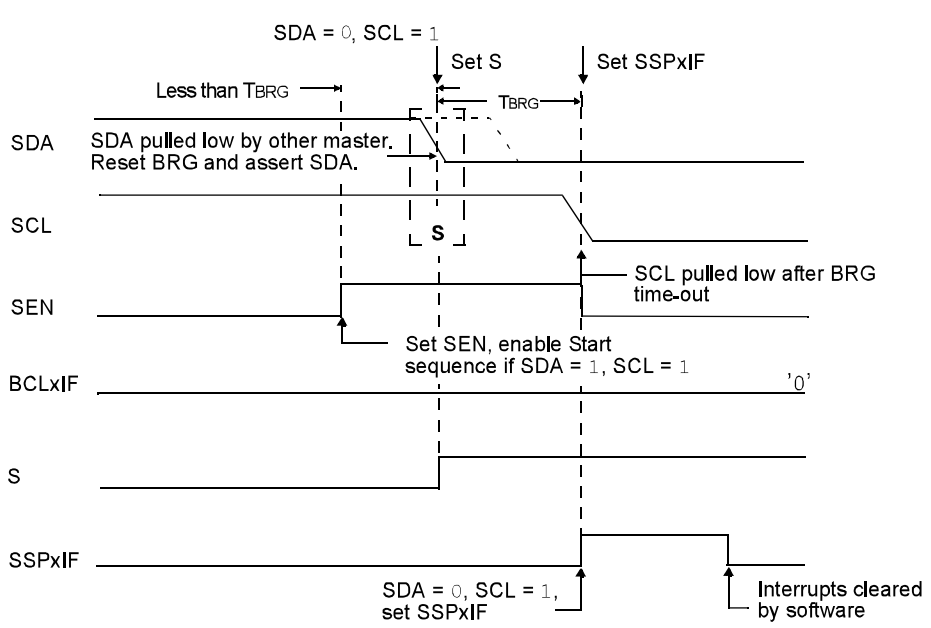
**Figure 25-36. Bus Collision During Start Condition (SCL = 0)**




If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (see Figure 25-37). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The

Baud Rate Generator is then reloaded and counts down to zero; if the SCL pin is sampled as '0' during this time, a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

**Figure 25-37. BRG Reset Due to SDA Arbitration During Start Condition**




**Important:** The reason that a bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

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

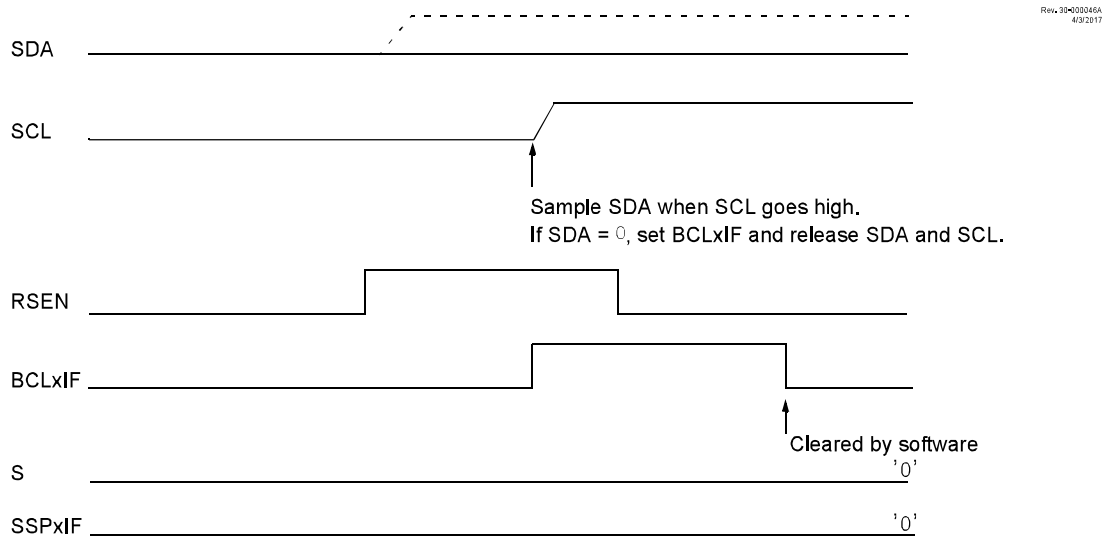
During a Repeated Start condition, a bus collision occurs if:

1. A low level is sampled on SDA when SCL goes from low level to high level (see Figure 25-38).
2. SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1' (see Figure 25-39).

When the user releases SDA and the pin is allowed to float high, the BRG is loaded with SSPxADD and counts down to zero. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', see Figure 25-38). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

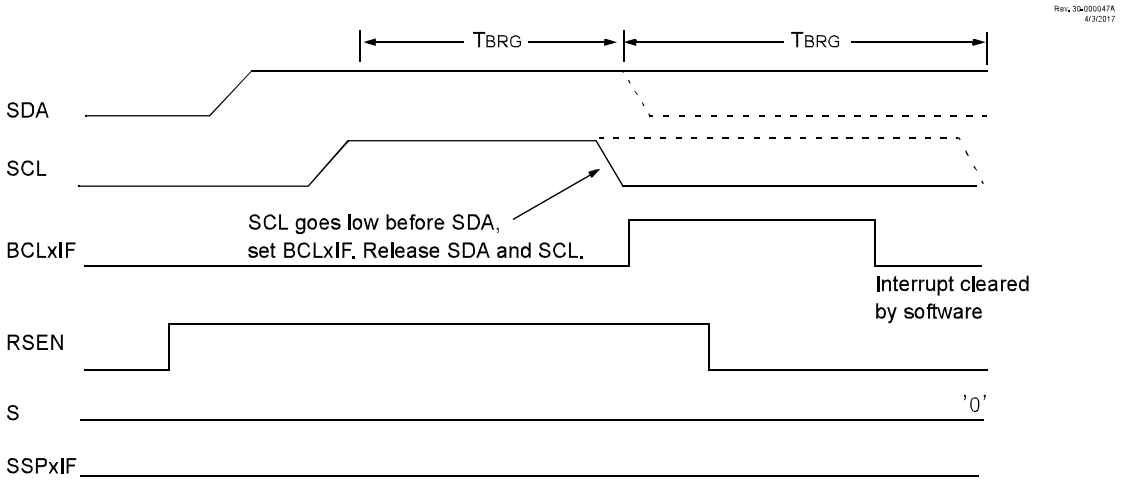
**Figure 25-38. Bus Collision During a Repeated Start Condition (Case 1)**




If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition (see Figure 25-39).

If, at the end of the BRG time-out, both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete.

**Figure 25-39. Bus Collision During Repeated Start Condition (Case 2)**




### 25.2.5.1.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

1. After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out (see Figure 25-40).
2. After the SCL pin is deasserted, SCL is sampled low before SDA goes high (see Figure 25-41).

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPxADD and counts down to zero. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (see Figure 25-40). If the SCL pin is sampled low before SDA is allowed

to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (see Figure 25-41).

**Figure 25-40. Bus Collision During a Stop Condition (Case 1)**

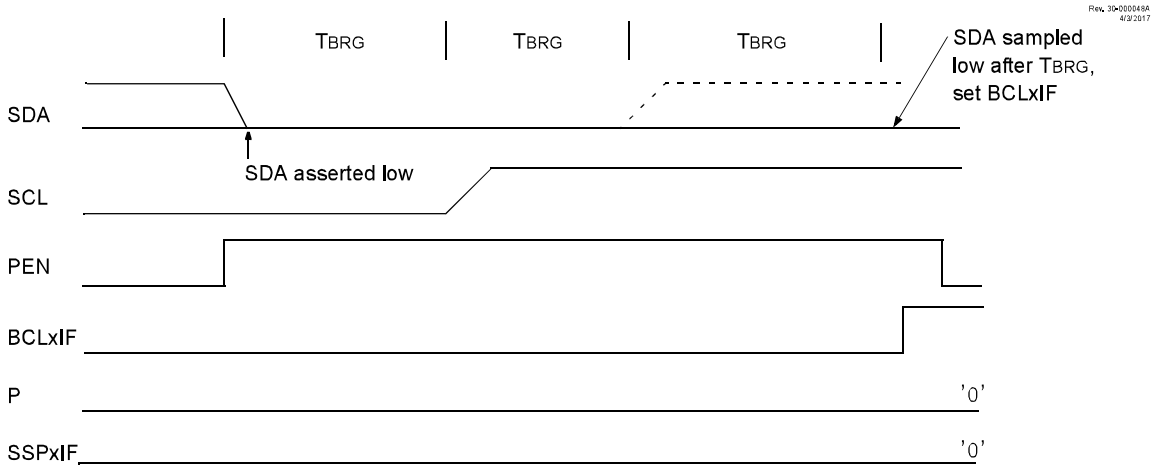
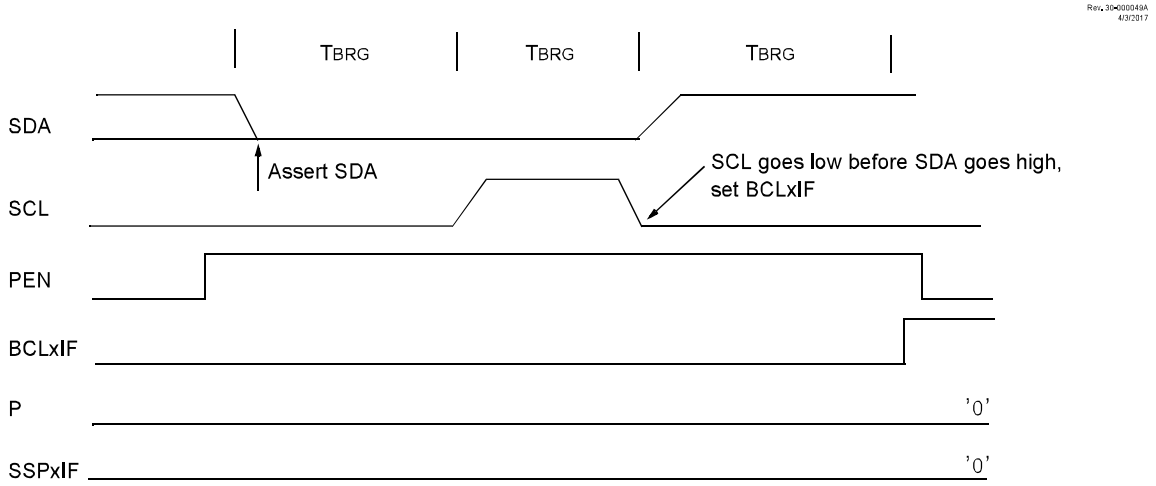



**Figure 25-41. Bus Collision During a Stop Condition (Case 2)**




## 25.3 Baud Rate Generator

The MSSP 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. When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting down. Example 25-1 shows how the value for SSPxADD is calculated.

Once the given operation is complete, the internal clock will automatically stop counting and the clock pin will remain in its last state.

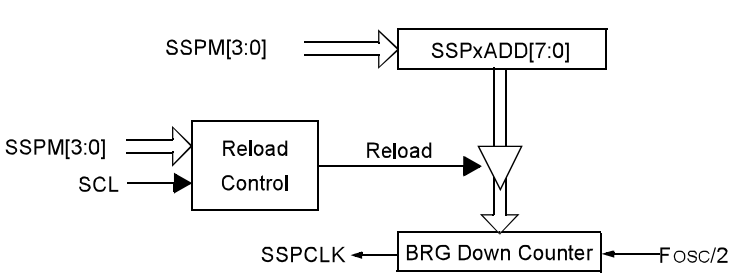
An internal signal "Reload", shown in Figure 25-42, triggers the value from SSPxADD to be loaded into the BRG counter. This occurs twice for each oscillation of the module clock line. The logic dictating when the reload signal is asserted depends on the mode in which the MSSP is being operated.

Table 25-1 illustrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD.

**Example 25-1. MSSP Baud Rate Generator Frequency Equation**

$$F_{CLOCK} = \frac{F_{OSC}}{4 \times (SSPxADD + 1)}$$

**Figure 25-42. Baud Rate Generator Block Diagram**




**Important:** 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 25-1. MSSP Clock Rate w/BRG**

| FOSC | FCY | BRG Value | FCLOCK (2 Rollovers of BRG) |
|---|---|---|---|
| 32 MHz | 8 MHz | 13h | 400 kHz |
| 32 MHz | 8 MHz | 19h | 308 kHz |
| 32 MHz | 8 MHz | 4Fh | 100 kHz |
| 16 MHz | 4 MHz | 09h | 400 kHz |
| 16 MHz | 4 MHz | 0Ch | 308 kHz |
| 16 MHz | 4 MHz | 27h | 100 kHz |
| 4 MHz | 1 MHz | 09h | 100 kHz |

**Note:** Refer to the I/O port electrical specifications in the "**Electrical Specifications**" chapter, Internal Oscillator Parameters, to ensure the system is designed to support all requirements.

## 25.4 Register Definitions: MSSP Control

### 25.4.1 SSPxBUF

**Name:** SSPxBUF
**Address:** 0x018C

MSSP Data Buffer Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BUF[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – BUF[7:0]** MSSP Input and Output Data Buffer bits

### 25.4.2 SSPxADD

**Name:** SSPxADD
**Address:** 0x018D

MSSP Baud Rate Divider and Address Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADD[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – ADD[7:0]**

- SPI and $I^2C$ Master: Baud rate divider
- $I^2C$ Slave: Address bits

| Value | Mode | Description |
|---|---|---|
| 11111111 – 00000011 | SPI and $I^2C$ Master | Baud rate divider. SCK/SCL pin clock period = ((n + 1) * 4)/$F_{OSC}$. Values less than 3 are not valid. |
| xxxxx11x – xxxxx00x | $I^2C$ 10-bit Slave MS Address | Bits [7:3] and Bit 0 are not used and are don't care. Bits [2:1] are bits [9:8] of the 10-bit Slave Most Significant Address |
| 11111111 – 00000000 | $I^2C$ 10-bit Slave LS Address | Bits [7:0] of 10-Bit Slave Least Significant Address |
| 1111111x – 0000000x | $I^2C$ 7-bit Slave | Bit 0 is not used and is don't care. Bits [7:1] are the 7-bit Slave Address |

### 25.4.3 SSPxMSK

**Name:** SSPxMSK
**Address:** 0x018E

MSSP Address Mask Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MSK[6:0] | | | | | | | MSK0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:1 – MSK[6:0]** Mask bits

| Value | Mode | Description |
|---|---|---|
| 1 | $I^2C$ Slave | The received address bit n is compared to SSPxADD bit n to detect $I^2C$ address match |
| 0 | $I^2C$ Slave | The received address bit n is not used to detect $I^2C$ address match |

**Bit 0 – MSK0**
Mask bit for $I^2C$ 10-bit Slave mode

| Value | Mode | Description |
|---|---|---|
| 1 | $I^2C$ 10-bit Slave | The received address bit 0 is compared to SSPxADD bit 0 to detect $I^2C$ address match |
| 0 | $I^2C$ 10-bit Slave | The received address bit 0 is not used to detect $I^2C$ address match |
| x | SPI or $I^2C$ 7-bit | Don't care |

### 25.4.4 SSPxSTAT

**Name:** SSPxSTAT
**Address:** 0x018F

MSSP Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| Access | R/W | R/W | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – SMP** Slew Rate Control bit

| Value | Mode | Description |
|---|---|---|
| 1 | SPI Master | Input data is sampled at the end of data output time |
| 0 | SPI Master | Input data is sampled at the middle of data output time |
| 0 | SPI Slave | Keep this bit cleared in SPI Slave mode |
| 1 | $I^2C$ | Slew rate control is disabled for Standard Speed mode (100 kHz and 1 MHz) |
| 0 | $I^2C$ | Slew rate control is enabled for High-Speed mode (400 kHz) |

**Bit 6 – CKE** SPI: Clock Select bit(4) I2C: SMBus Select bit

| Value | Mode | Description |
|---|---|---|
| 1 | SPI | Transmit occurs on the transition from active to Idle clock state |
| 0 | SPI | Transmit occurs on the transition from Idle to active clock state |
| 1 | $I^2C$ | Enables SMBus-specific inputs |
| 0 | $I^2C$ | Disables SMBus-specific inputs |

**Bit 5 – D/$\overline{A}$**
Data/$\overline{Address}$ bit

| Value | Mode | Description |
|---|---|---|
| x | SPI or $I^2C$ Master | Reserved |
| 1 | $I^2C$ Slave | Indicates that the last byte received or transmitted was data |
| 0 | $I^2C$ Slave | Indicates that the last byte received or transmitted was address |

**Bit 4 – P**
Stop bit(1)

| Value | Mode | Description |
|---|---|---|
| x | SPI | Reserved |
| 1 | $I^2C$ | Stop bit was detected last |
| 0 | $I^2C$ | Stop bit was not detected last |

**Bit 3 – S**
Start bit(1)

| Value | Mode | Description |
|---|---|---|
| x | SPI | Reserved |
| 1 | $I^2C$ | Start bit was detected last |
| 0 | $I^2C$ | Start bit was not detected last |

**Bit 2 – R/$\overline{W}$**
Read/$\overline{Write}$ Information bit(2,3)

| Value | Mode | Description |
|---|---|---|
| x | SPI | Reserved |
| 1 | $I^2C$ Slave | Read |
| 0 | $I^2C$ Slave | Write |
| 1 | $I^2C$ Master | Transmit is in progress |
| 0 | $I^2C$ Master | Transmit is not in progress |

**Bit 1 – UA** Update Address bit (10-Bit Slave mode only)

| Value | Mode | Description |
|---|---|---|
| x | All other modes | Reserved |
| 1 | $I^2C$ 10-bit Slave | Indicates that the user needs to update the address in the SSPxADD register |
| 0 | $I^2C$ 10-bit Slave | Address does not need to be updated |

**Bit 0 – BF**

Buffer Full Status bit(5)

| Value | Mode | Description |
|---|---|---|
| 1 | $I^2C$ Transmit | Character written to SSPxBUF has not been sent |
| 0 | $I^2C$ Transmit | SSPxBUF is ready for next character |
| 1 | SPI and $I^2C$ Receive | Received character in SSPxBUF has not been read |
| 0 | SPI and $I^2C$ Receive | Received character in SSPxBUF has been read |

**Notes:**

1. This bit is cleared on Reset and when SSPEN is cleared.
2. In $I^2C$ Slave mode 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.
3. ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Active mode.
4. Polarity of clock state is set by the CKP bit.
5. $I^2C$ receive status does not include $\overline{ACK}$ and Stop bits.

### 25.4.5 SSPxCON1

**Name:** SSPxCON1
**Address:** 0x0190

MSSP Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WCOL | SSPOV | SSPEN | CKP | SSPM[3:0] | | | |
| Access | R/W/HS | R/W/HS | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – WCOL**
Write Collision Detect bit

| Value | Mode | Description |
|---|---|---|
| 1 | SPI | A write to the SSPxBUF register was attempted while the previous byte was still transmitting (must be cleared by software) |
| 1 | $I^2C$ Master transmit | A write to the SSPxBUF register was attempted while the $I^2C$ conditions were not valid for a transmission to be started (must be cleared by software) |
| 1 | $I^2C$ Slave transmit | The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software) |
| 0 | SPI or $I^2C$ Master or Slave transmit | No collision |
| x | Master or Slave receive | Don't care |

**Bit 6 – SSPOV**
Receive Overflow Indicator bit(1)

| Value | Mode | Description |
|---|---|---|
| 1 | SPI Slave | A byte is received while the SSPxBUF register is still holding the previous byte. The user must read SSPxBUF, even if only transmitting data, to avoid setting overflow. (must be cleared in software) |
| 1 | $I^2C$ Receive | A byte is received while the SSPxBUF register is still holding the previous byte (must be cleared in software) |
| 0 | SPI Slave or $I^2C$ Receive | No overflow |
| x | SPI Master or $I^2C$ Master transmit | Don't care |

**Bit 5 – SSPEN**
Master Synchronous Serial Port Enable bit.(2)

| Value | Mode | Description |
|---|---|---|
| 1 | SPI | Enables the serial port. The SCKx, SDOx, SDIx, and $\overline{SSx}$ pin selections must be made with the PPS controls. Each signal must be configured with the corresponding TRIS control to the direction appropriate for the mode selected. |
| 1 | $I^2C$ | Enables the serial port. The SDAx and SCLx pin selections must be made with the PPS controls. Since both signals are bidirectional the PPS input pin and PPS output pin selections must be made that specify the same pin. Both pins must be configured as inputs with the corresponding TRIS controls. |
| 0 | All | Disables serial port and configures these pins as I/O PORT pins |

**Bit 4 – CKP**
SCK Release Control bit

| Value | Mode | Description |
|---|---|---|
| 1 | SPI | Idle state for the clock is a high level |
| 0 | SPI | Idle state for the clock is a low level |
| 1 | $I^2C$ Slave | Releases clock |
| 0 | $I^2C$ Slave | Holds clock low (clock stretch), used to ensure data setup time |
| x | $I^2C$ Master | Unused in this mode |

**Bits 3:0 – SSPM[3:0]**

Master Synchronous Serial Port Mode Select bits(4)

| Value | Description |
|---|---|
| 1111 | I²C Slave mode: 10-bit address with Start and Stop bit interrupts enabled |
| 1110 | I²C Slave mode: 7-bit address with Start and Stop bit interrupts enabled |
| 1101 | Reserved - do not use |
| 1100 | Reserved - do not use |
| 1011 | I²C Firmware Controlled Master mode (slave Idle) |
| 1010 | SPI Master mode: Clock = $F_{OSC}$/(4*(SSPxADD+1)). SSPxADD must be greater than 0.(3) |
| 1001 | Reserved - do not use |
| 1000 | I²C Master mode: Clock = $F_{OSC}$/(4 * (SSPxADD + 1)) |
| 0111 | I²C Slave mode: 10-bit address |
| 0110 | I²C Slave mode: 7-bit address |
| 0101 | SPI Slave mode: Clock = SCKx pin. $\overline{SSx}$ pin control is disabled |
| 0100 | SPI Slave mode: Clock = SCKx pin. $\overline{SSx}$ pin control is enabled |
| 0011 | SPI Master mode: Clock = TMR2 output/2 |
| 0010 | SPI Master mode: Clock = $F_{OSC}$/64 |
| 0001 | SPI Master mode: Clock = $F_{OSC}$/16 |
| 0000 | SPI Master mode: Clock = $F_{OSC}$/4 |

**Notes:**

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 inputs or outputs.
3. SSPxADD = 0 is not supported.
4. Bit combinations not specifically listed here are either reserved or implemented in I²C mode only.

### 25.4.6 SSPxCON2

**Name:** SSPxCON2
**Address:** 0x0191

MSSP Control Register 2

Control Register for $I^2C$ Operation Only

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| Access | R/W | R/W/HC | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – GCEN**
General Call Enable bit (Slave mode only)

| Value | Mode | Description |
|---|---|---|
| x | Master mode | Don't care |
| 1 | Slave mode | General Call is enabled |
| 0 | Slave mode | General Call is not enabled |

**Bit 6 – ACKSTAT** Acknowledge Status bit (Master Transmit mode only)

| Value | Description |
|---|---|
| 1 | Acknowledge was not received from slave |
| 0 | Acknowledge was received from slave |

**Bit 5 – ACKDT**
Acknowledge Data bit (Master Receive mode only)(1)

| Value | Description |
|---|---|
| 1 | Not Acknowledge |
| 0 | Acknowledge |

**Bit 4 – ACKEN**
Acknowledge Sequence Enable bit(2)

| Value | Description |
|---|---|
| 1 | Initiates Acknowledge sequence on SDAx and SCLx pins and transmits ACKDT data bit; automatically cleared by hardware |
| 0 | Acknowledge sequence is Idle |

**Bit 3 – RCEN**
Receive Enable bit (Master Receive mode only)(2)

| Value | Description |
|---|---|
| 1 | Enables Receive mode for $I^2C$ |
| 0 | Receive is Idle |

**Bit 2 – PEN**
Stop Condition Enable bit (Master mode only)(2)

| Value | Description |
|---|---|
| 1 | Initiates Stop condition on SDAx and SCLx pins; automatically cleared by hardware |
| 0 | Stop condition is Idle |

**Bit 1 – RSEN**
Repeated Start Condition Enable bit (Master mode only)(2)

| Value | Description |
|---|---|
| 1 | Initiates Repeated Start condition on SDAx and SCLx pins; automatically cleared by hardware |
| 0 | Repeated Start condition is Idle |

**Bit 0 – SEN**

Start Condition Enable bit (Master mode only)(2)

| Value | Description |
|---|---|
| 1 | Initiates Start condition on SDAx and SCLx pins; automatically cleared by hardware |
| 0 | Start condition is Idle |

**Notes:**

1. The value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
2. If the $I^2C$ module is active, these bits may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).

### 25.4.7 SSPxCON3

**Name:** SSPxCON3
**Address:** 0x0192

MSSP Control Register 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN |
| Access | R/HS/HC | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – ACKTIM** Acknowledge Time Status bit
Unused in Master mode.

| Value | Mode | Description |
|---|---|---|
| x | SPI or I²C Master | This bit is not used |
| 1 | I²C Slave and AHEN = 1 or DHEN = 1 | Eighth falling edge of SCL has occurred and the $\overline{ACK}$/NACK state is active |
| 0 | I²C Slave | $\overline{ACK}$/NACK state is not active. Transitions low on ninth rising edge of SCL. |

**Bit 6 – PCIE**
Stop Condition Interrupt Enable bit(1)

| Value | Mode | Description |
|---|---|---|
| x | SPI or SSPM = 1111 or 0111 | Don't care |
| 1 | SSPM ≠ 1111 and SSPM ≠ 0111 | Enable interrupt on detection of Stop condition |
| 0 | SSPM ≠ 1111 and SSPM ≠ 0111 | Stop detection interrupts are disabled |

**Bit 5 – SCIE** Start Condition Interrupt Enable bit

| Value | Mode | Description |
|---|---|---|
| x | SPI or SSPM = 1111 or 0111 | Don't care |
| 1 | SSPM ≠ 1111 and SSPM ≠ 0111 | Enable interrupt on detection of Start condition |
| 0 | SSPM ≠ 1111 and SSPM ≠ 0111 | Start detection interrupts are disabled |

**Bit 4 – BOEN**
Buffer Overwrite Enable bit(2)

| Value | Mode | Description |
|---|---|---|
| 1 | SPI | SSPxBUF is updated every time a new data byte is available, ignoring the BF bit |
| 0 | SPI | If a new byte is receive with BF set then SSPOV is set and SSPxBUF is not updated |
| 1 | I²C | SSPxBUF is updated every time a new data byte is available, ignoring the SSPOV effect on updating the buffer |
| 0 | I²C | SSPxBUF is only updated when SSPOV is clear |

**Bit 3 – SDAHT** SDA Hold Time Selection bit

| Value | Mode | Description |
|---|---|---|
| x | SPI | Not used in SPI mode |
| 1 | I²C | Minimum of 300 ns hold time on SDA after the falling edge of SCL |
| 0 | I²C | Minimum of 100 ns hold time on SDA after the falling edge of SCL |

**Bit 2 – SBCDE** Slave Mode Bus Collision Detect Enable bit
Unused in Master mode.

| Value | Mode | Description |
|---|---|---|
| x | SPI or I²C Master | Don't care |
| 1 | I²C Slave | Collision detection is enabled |
| 0 | I²C Slave | Collision detection is not enabled |

**Bit 1 – AHEN** Address Hold Enable bit

| Value | Mode | Description |
|---|---|---|
| x | SPI or $I^2C$ Master | Don't care |
| 1 | $I^2C$ Slave | Address hold is enabled. As a result CKP is cleared after the eighth falling SCL edge of an address byte reception. Software must set the CKP bit to resume operation. |
| 0 | $I^2C$ Slave | Address hold is not enabled |

**Bit 0 – DHEN** Data Hold Enable bit

| Value | Mode | Description |
|---|---|---|
| x | SPI or $I^2C$ Master | Don't care |
| 1 | $I^2C$ Slave | Data hold is enabled. As a result CKP is cleared after the eighth falling SCL edge of a data byte reception. Software must set the CKP bit to resume operation. |
| 0 | $I^2C$ Slave | Data hold is not enabled |

**Notes:**

1. This bit has no effect in Slave modes that Start and Stop condition detection is explicitly listed as enabled.
2. 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.

## 25.5 Register Summary: MSSP Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x018B | Reserved | | | | | | | | | |
| 0x018C | SSP1BUF | 7:0 | BUF[7:0] | | | | | | | |
| 0x018D | SSP1ADD | 7:0 | ADD[7:0] | | | | | | | |
| 0x018E | SSP1MSK | 7:0 | MSK[6:0] | | | | | | | MSK0 |
| 0x018F | SSP1STAT | 7:0 | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| 0x0190 | SSP1CON1 | 7:0 | WCOL | SSPOV | SSPEN | CKP | SSPM[3:0] | | | |
| 0x0191 | SSP1CON2 | 7:0 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| 0x0192 | SSP1CON3 | 7:0 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN |

# 26. FVR - Fixed Voltage Reference

The Fixed Voltage Reference (FVR) is a stable voltage reference, independent of $V_{DD}$, with 1.024V, 2.048V or 4.096V selectable output levels. The output of the FVR can be configured to supply a reference voltage to the ADC module as a positive reference and input channel.

The FVR can be enabled by setting the FVREN bit to '1'.

**Note:** Fixed Voltage Reference output cannot exceed $V_{DD}$.

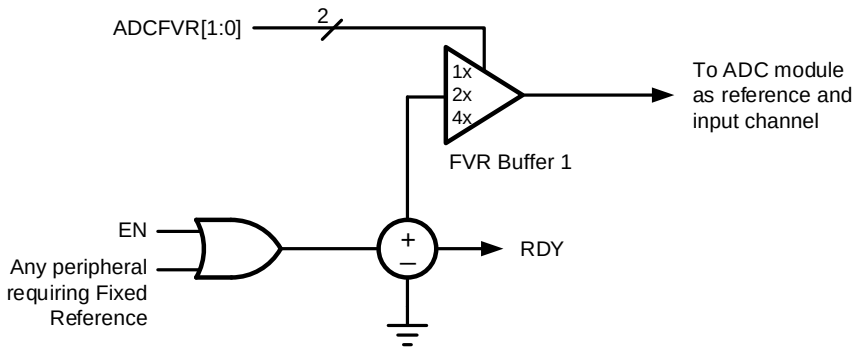
## 26.1 Independent Gain Amplifiers

The output of the FVR, which is connected to the ADC, is routed through an independent programmable gain amplifier. This amplifier can be programmed for a gain of 1x, 2x or 4x, to produce the three possible voltage levels.

The ADFVR bits are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module.

Refer to the figure below for block diagram of the FVR module.

**Figure 26-1. Fixed Voltage Reference Block Diagram**




## 26.2 FVR Stabilization Period

When the FVR 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 will be set.

## 26.3 Register Definitions: FVR

Long bit name prefixes for the FVR peripherals are shown in the following table. Refer to the **"Long Bit Names"** section in the **"Register and Bits Naming Conventions"** chapter for more information.

**Table 26-1. FVR Long bit name prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| FVR | FVR |

### 26.3.1 FVRCON

**Name:** FVRCON
**Address:** 0x90C

Fixed Voltage Reference Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FVREN | FVRRDY | | | | | ADFVR[1:0] | |
| Access | R/W | R | | | | | R/W | R/W |
| Reset | 0 | 0 | | | | | 0 | 0 |

**Bit 7 – FVREN** Fixed Voltage Reference Enable bit

| Value | Description |
|---|---|
| 1 | Fixed Voltage Reference is enabled |
| 0 | Fixed Voltage Reference is disabled |

**Bit 6 – FVRRDY** Fixed Voltage Reference Ready Flag bit (2)

| Value | Description |
|---|---|
| 1 | Fixed Voltage Reference output is ready for use |
| 0 | Fixed Voltage Reference output is not ready or not enabled |

**Bits 1:0 – ADFVR[1:0]** ADC FVR Buffer Gain Selection bit

| Value | Description |
|---|---|
| 11 | ADC FVR Buffer Gain is 4x, (4.096V)(1) |
| 10 | ADC FVR Buffer Gain is 2x, (2.048V)(1) |
| 01 | ADC FVR Buffer Gain is 1x, (1.024V) |
| 00 | ADC FVR Buffer is off |

**Notes:**

1. Fixed Voltage Reference output cannot exceed $V_{DD}$.
2. FVRRDY is always '1'.

## 26.4 Register Summary - FVR

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x090B | Reserved | | | | | | | | | |
| 0x090C | FVRCON | 7:0 | FVREN | FVRRDY | | | | | ADFVR[1:0] | |

# 27. ADC - Analog-to-Digital Converter

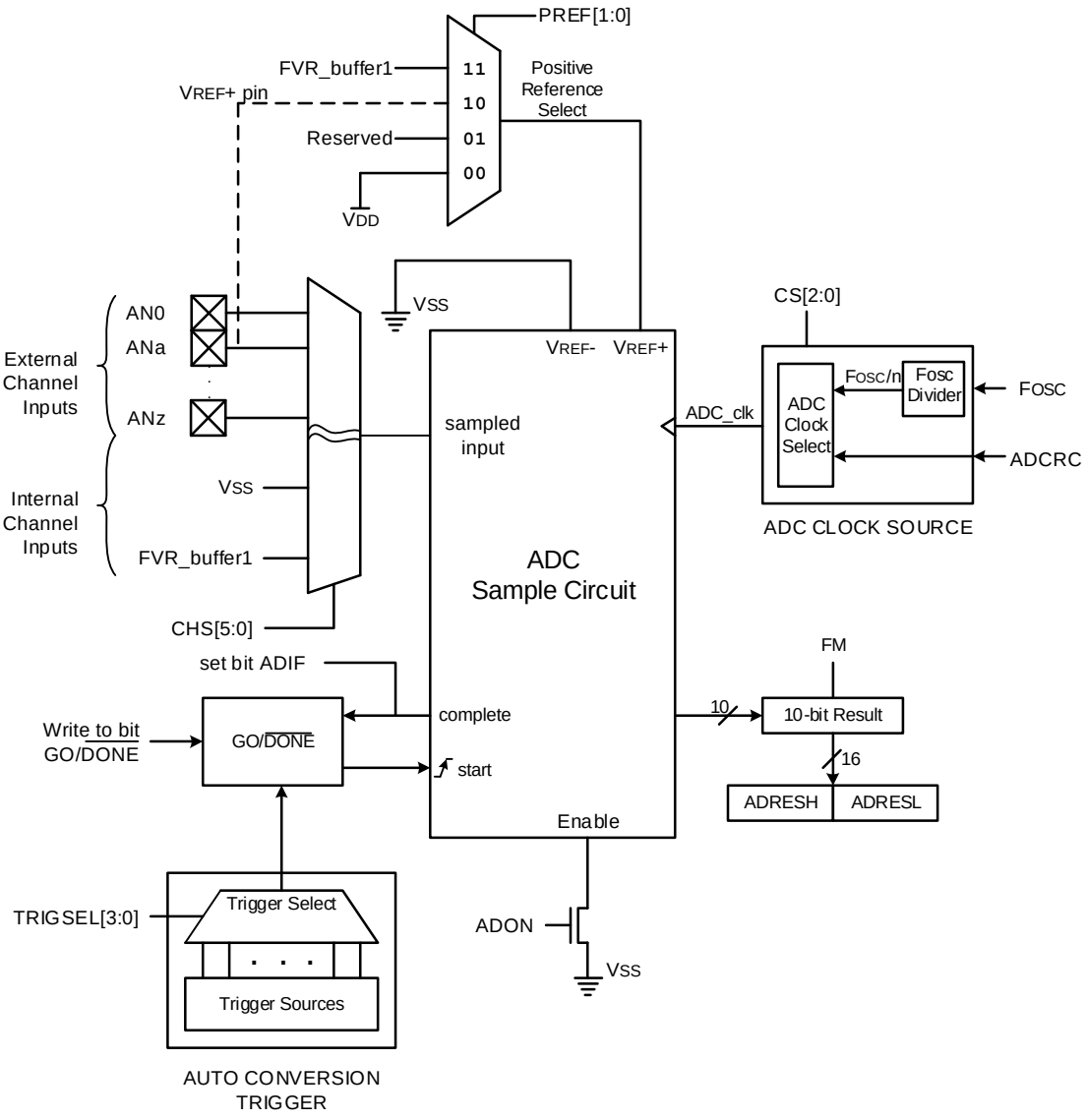
The Analog-to-Digital Converter (ADC) allows the conversion of an analog input signal into a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single Sample-and-Hold circuit. The output of the Sample-and-Hold circuit is connected to the input of the converter. The converter generates a 10-bit binary result via successive approximation and stores the results in the ADC Result registers (ADRESH:ADRESL 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.

Figure 27-1 shows the block diagram of the ADC.

**Figure 27-1. ADC Block Diagram**




## 27.1 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

### 27.1.1 Port Configuration

The ADC will convert the voltage on a pin whether or not the ANSEL bit is set. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits. Refer to the "**I/O Ports**" chapter for more information.

**Important:** Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

### 27.1.2 Channel Selection

The Analog Channel Select (CHS) bits determine which channel is connected to the Sample-and-Hold circuit for conversion. When switching channels, it is recommended to add an acquisition delay before starting the next conversion. Refer to the "ADC Operation" section for more information.

**Important:** To reduce the chance of measurement error, it is recommended to discharge the Sample-and-Hold capacitor when switching between ADC channels by starting a conversion on a channel connected to $V_{SS}$ and terminating the conversion after the acquisition time has elapsed. If the ADC does not have a dedicated $V_{SS}$ input channel, a free input channel can be connected to $V_{SS}$ and used in place of the dedicated input channel.

### 27.1.3 ADC Voltage Reference

The ADC Positive Voltage Reference Selection (PREF) bits provide control of the positive voltage reference. Refer to the ADC Control Register 1 (ADCON1) for the list of available positive voltage sources.

### 27.1.4 Conversion Clock

The conversion clock source is selected via the ADC Conversion Clock Select (CS) bits. The available clock sources include several derivatives of the system clock ($F_{OSC}$), as well as a dedicated internal fixed-frequency clock referred to as the ADCRC.

The time to complete one bit conversion is defined as the $T_{AD}$. Refer to Figure 27-2 for complete timing details of the ADC conversion.

For a correct conversion, the appropriate $T_{AD}$ specification must be met. Refer to the ADC Timing Specifications table in the **"Electrical Specifications"** chapter for more details. Table 27-1 gives examples of appropriate ADC clock selections.

**Important:**

- With the exception of the ADCRC clock source, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.
- The internal control logic of the ADC operates off of the clock selected by the CS bits. When the CS bits select the ADCRC, there may be unexpected delays in operation when setting the ADC control bits.

**Table 27-1. ADC Clock Period ($T_{AD}$) for Different Device Frequencies ($F_{OSC}$)**

| ADC Clock Source | CS[2:0] | ADC Clock Period ($T_{AD}$) for Different Device Frequencies ($F_{OSC}$) | | | | | |
|---|---|---|---|---|---|---|---|
| | | 32 MHz | 20 MHz | 16 MHz | 8 MHz | 4 MHz | 1 MHz |
| $F_{OSC}/2$ | 'b000 | 62.5 ns(2) | 100 ns(2) | 125 ns(2) | 250 ns(2) | 500 ns(2) | 2.0 µs |
| $F_{OSC}/4$ | 'b100 | 125 ns(2) | 200 ns(2) | 250 ns(2) | 500 ns(2) | 1.0 µs | 4.0 µs |

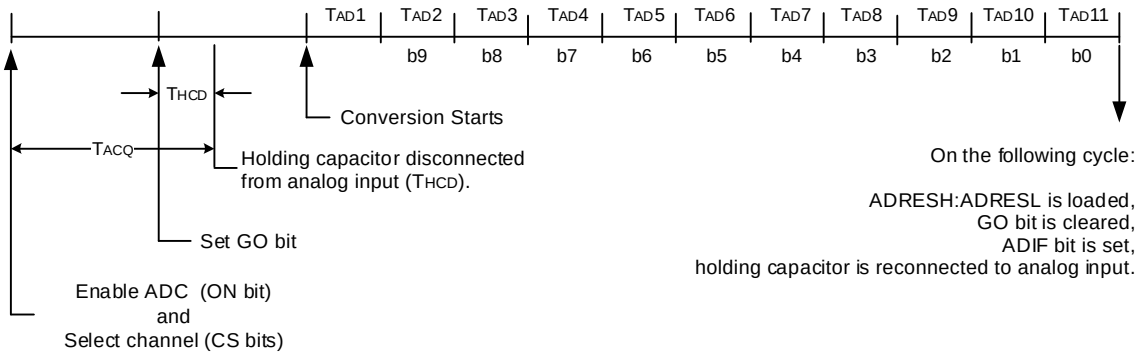
.....continued

| ADC Clock Source | CS[2:0] | ADC Clock Period ($T_{AD}$) for Different Device Frequencies ($F_{OSC}$) | | | | | |
|---|---|---|---|---|---|---|---|
| | | 32 MHz | 20 MHz | 16 MHz | 8 MHz | 4 MHz | 1 MHz |
| $F_{OSC}$/8 | 'b001 | 250 ns(2) | 400 ns(2) | 500 ns(2) | 1.0 µs | 2.0 µs | 8.0 µs |
| $F_{OSC}$/16 | 'b101 | 500 ns(2) | 800 ns(2) | 1.0 µs | 2.0 µs | 4.0 µs | 16.0 µs(2) |
| $F_{OSC}$/32 | 'b010 | 1.0 µs | 1.6 µs | 2.0 µs | 4.0 µs | 8.0 µs | 32.0 µs(2) |
| $F_{OSC}$/64 | 'b110 | 2.0 µs | 3.2 µs | 4.0 µs | 8.0 µs | 16.0 µs(2) | 64.0 µs(2) |
| ADCRC | 'bx11 | 1.0 - 6.0(1,3) | | | | | |

**Notes:**

1. Refer to the "**Electrical Specifications**" chapter to see the $T_{AD}$ parameter for the ADCRC source typical $T_{AD}$ value.
2. These values violate the required $T_{AD}$ time.
3. The ADC clock period ($T_{AD}$) and the total ADC conversion time can be minimized when the ADC clock is derived from the system clock $F_{OSC}$. However, the ADCRC oscillator source must be used when conversions are to be performed with the device in Sleep mode.

**Figure 27-2. 10-Bit Analog-to-Digital Conversion $T_{AD}$ Cycles**




### 27.1.5 Interrupts

The ADC module allows for the ability to generate an interrupt upon completion of an analog-to-digital conversion. The ADC Interrupt Flag (ADIF) bit is set upon the completion of each conversion. If the ADC Interrupt Enable (ADIE) bit is set, an ADC interrupt event occurs. The ADIF bit must be cleared by software.

**Important:**

1. The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC Interrupt is enabled.
2. The ADC operates in Sleep only when the ADCRC oscillator is selected as the clock source.

The ADC Interrupt can be generated while the device is operating or while in Sleep. While the device is operating in Sleep mode:

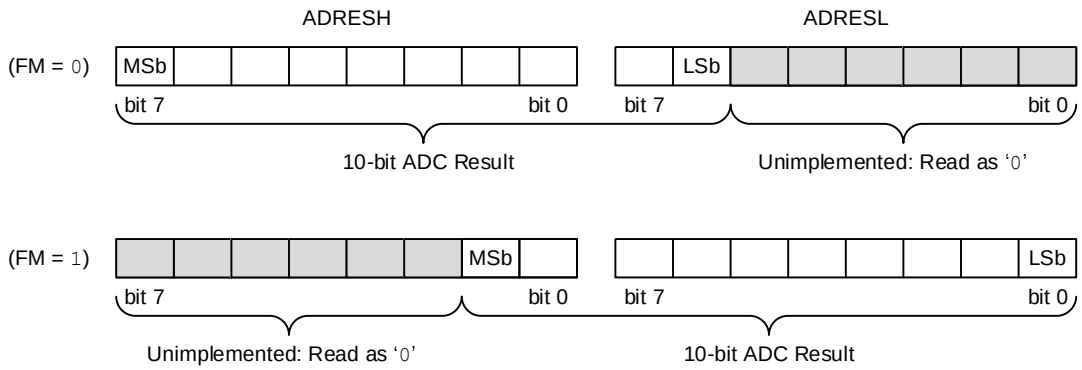
- If ADIE = 1, PEIE = 1, and GIE = 0: An interrupt will wake the device from Sleep. Upon waking from Sleep, the instructions following the SLEEP instruction are executed. The Interrupt Service Routine is not executed.

- If ADIE = 1, PEIE = 1, and GIE = 1: An interrupt will wake the device from Sleep. Upon waking from Sleep, the instruction following the SLEEP instruction is always executed. Then the execution will switch to the Interrupt Service Routine.

### 27.1.6 ADC Result Formatting

The 10-bit ADC conversion result can be supplied in two formats: left-justified or right-justified. The ADC Result Format/Alignment Selection (FM) bit controls the output format as shown in Figure 27-3.

**Figure 27-3. 10-Bit ADC Conversion Result Format**




**Important:** Writes to the ADRES register pair are always right-justified, regardless of the selected format mode. Therefore, data read after writing to ADRES when FM = 0 will be shifted left four places.

## 27.2 ADC Operation

### 27.2.1 Starting a Conversion

To enable the ADC module, the ON bit must be set to '1'. A conversion may be started by either of the following:

- Software setting the GO bit to '1'
- An external trigger (source selected by the ADC Auto-Conversion Trigger (ADACT) register

**Important:** The GO bit should not be set in the same instruction that turns on the ADC. Refer to the "ADC Conversion Procedure" section for more details.

### 27.2.2 Completion of a Conversion

When the conversion is complete, the ADC module will:

- Clear the GO bit
- Set the ADIF bit
- Update the ADRES registers with the new conversion result

### 27.2.3 ADC Operation During Sleep

The ADC module can operate in Sleep. This requires the ADC clock source to be set to the ADCRC option. When the ADC oscillator source is selected, the ADC waits one additional instruction before starting the conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled (ADIE = 1), the device will wake up from Sleep when the conversion completes. If the ADC interrupt is disabled (ADIE = 0), the device remains in Sleep and the ADC module is turned off, although the ON bit remains set.

When the ADC clock source is something other than the ADCRC, a SLEEP instruction causes the present conversion to be aborted and the ADC is turned off, although the ON bit remains set.

#### 27.2.3.1 External Trigger During Sleep

If an external trigger is received when the ADC is in Sleep, the trigger will be recorded, but the conversion will not begin until the device exits Sleep.

### 27.2.4 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 bit is set by hardware.

The auto-conversion trigger source is selected with the Auto-Conversion Trigger Select (ACT) bits.

**Important:** Using the auto-conversion trigger does not ensure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met.

### 27.2.5 ADC Conversion Procedure

This is an example procedure for using the ADC to perform an Analog-to-Digital Conversion:

1. Configure Port:
   1.1. Disable the pin output driver (Refer to the TRISx register)
   1.2. Configure the pin as analog (Refer to the ANSELx register)
2. Configure the ADC module:
   2.1. Select the ADC conversion clock
   2.2. Configure the voltage reference
   2.3. Select the ADC input channel
   2.4. Configure result format
   2.5. Turn on the ADC module
3. Configure ADC interrupt (optional):
   3.1. Clear ADC interrupt flag
   3.2. Enable ADC interrupt
   3.3. Enable the peripheral interrupt (PEIE bit)
   3.4. Enable global interrupt (GIE bit)(1)
4. Wait the required acquisition time(2).
5. Start conversion by setting the GO bit.
6. Wait for ADC conversion to complete by one of the following:
   – Polling the GO bit
   – Waiting for the ADC interrupt (if interrupt is enabled)
7. Read ADC Result.
8. Clear the ADC interrupt flag (if interrupt is enabled).

**Notes:**

1. With global interrupts disabled (GIE = 0), the device will wake from Sleep but will not enter an Interrupt Service Routine.
2. Refer to the "ADC Acquisition Requirements" section for more details.

**Example 27-1. ADC Conversion (assembly)**

```
; This code block configures the ADC for polling,
; VDD and VSS references,
; ADCRC oscillator, and AN0 input.
; Conversion start & polling for completion are included.

    BANKSEL TRISA
    BSF     TRISA,0     ; Set RA0 to input
    BANKSEL ANSEL
    BSF     ANSEL,0     ; Set RA0 to analog
    BANKSEL ADCON0
    CLRF    ADCON0
    CLRF    ADCON1
    CLRF    ADACT       ; Auto-conversion disabled
    BSF     ADCON0,0    ; CHS = RA0, ADC ON
    MOVLW   B'11110000' ; FM = Right-justified, CS = ADCRC, PREF = VDD
    MOVWF   ADCON1
    CALL    SampleTime  ; Acquisition delay
    BANKSEL ADCON0
    BSF     ADCON0,GO   ; Start conversion
    BTFSC   ADCON0,GO   ; Is conversion done?
    GOTO    $-1         ; No, test again
    BANKSEL ADRESH
    MOVF    ADRESH,W    ; Read upper byte
    MOVWF   RESULTHI    ; Store in GPR space
    MOVF    ADRESL,W    ; Read lower byte
    MOVWF   RESULTLO    ; Store in GPR space
```

**Example 27-2. ADC Conversion (C)**

```
/*This code block configures the ADC
for polling, VDD and VSS references,
ADCRC oscillator and AN0 input.
Conversion start & polling for completion
are included.
  */
    void main() {
        //System Initialize
        initializeSystem();

        // Configure Port
        TRISAbits.TRISA0 = 1;       // Set RA0 to input
        ANSELAbits.ANSELA0 = 1;     // Set RA0 to analog

        // Configure ADC
        ADCON1bits.CS = 1;          // ADCRC Clock
        ADCON1bits.PREF = 'b11;     // VDD
        ADCON0bits.CHS = 'b000000; // RA0
        ADCON1bits.FM = 1;          // Right justify
        ADCON0bits.ON = 1;          // Turn ADC On

    while (1) {
        ADCON0bits.GO = 1;          // Start conversion
        while (ADCON0bits.GO);      // Wait for conversion done
        resultHigh = ADRESH;        // Read result
        resultLow = ADRESL;         // Read result
    }
 }
```

## 27.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor ($C_{HOLD}$) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 27-4. The source impedance ($R_S$) and the internal sampling switch ($R_{SS}$) impedance directly affect the time required to charge the capacitor $C_{HOLD}$. The sampling switch ($R_{SS}$) impedance varies over the device voltage ($V_{DD}$). 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 time must be completed before the conversion can be started. To calculate the minimum acquisition time, Equation 27-1 may be used. This equation assumes an error of 1/2 LSb. The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

**Equation 27-1. Acquisition Time Example**

Assumptions: Temperature = 50°C; External impedance = 10kΩ; $V_{DD}$ = 5.0V

$T_{ACQ}$ = Amplifier Settling Time + Hold Capacitor Charging Time + Temperature Coefficient

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{ACQ} = 2\mu s + T_C + \left[\left(Temperature - 25°C\right)\left(0.05\mu s/°C\right)\right]$$

The value for $T_C$ can be approximated with the following equations:

$$V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right) = V_{CHOLD}$$ ; [1] $V_{CHOLD}$ charged to within ½ lsb

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD}$$ ; [2] $V_{CHOLD}$ charge response to $V_{APPLIED}$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right)$$ ; Combining [1] and [2]

Note: Where n = ADC resolution in bits

Solving for $T_C$:

$$T_C = -C_{HOLD}\left(R_{IC} + R_{SS} + R_S\right) ln\left(1/2047\right)$$

$$T_C = -10pF\left(1k\Omega + 7k\Omega + 10k\Omega\right) ln\left(0.0004885\right)$$

$$T_C = 1.37\mu s$$

Therefore:

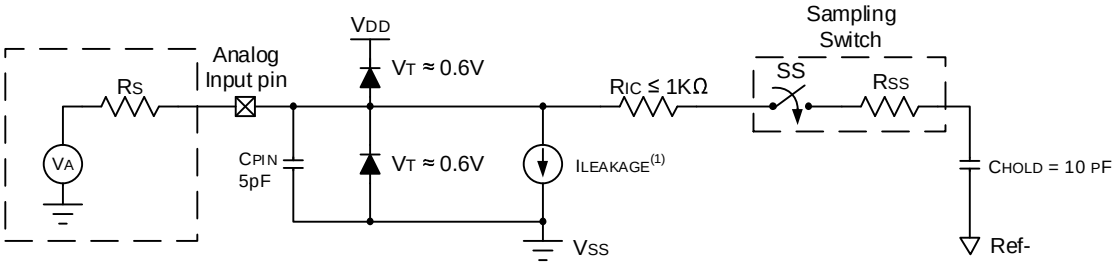
$$T_{ACQ} = 2\mu s + 1.37\mu s + \left[\left(50°C - 25°C\right)\left(0.05\mu s/°C\right)\right]$$

$$T_{ACQ} = 4.62\ \mu s$$

**Important:**

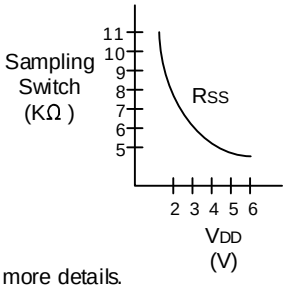
- The reference voltage ($V_{REF}$) has no effect on the equation, since it cancels itself out.
- The charge holding capacitor ($C_{HOLD}$) is not discharged after each conversion.
- The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

**Figure 27-4. Analog Input Model**



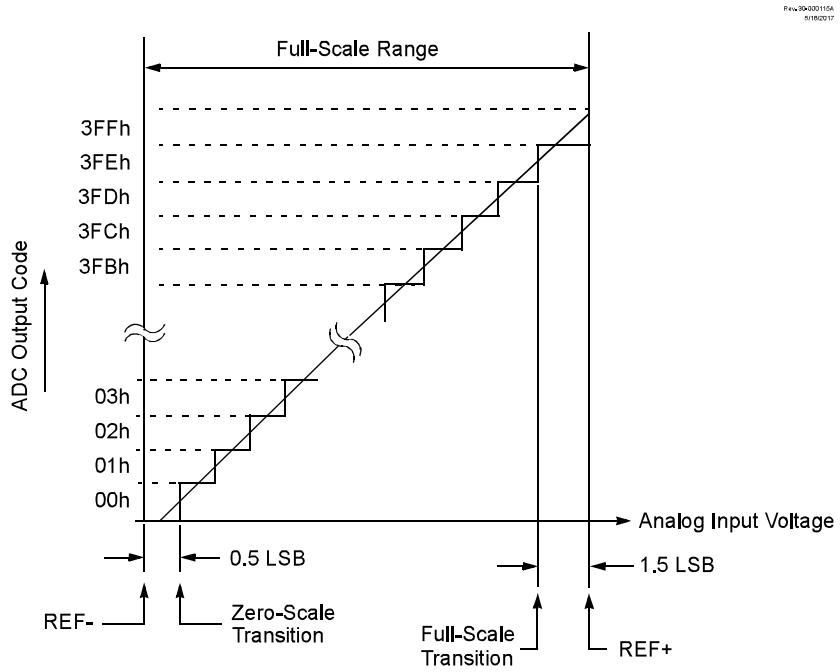

**Legend:**

| | | |
|---|---|---|
| CPIN | = | Input Capacitance |
| ILEAKAGE | = | Leakage Current at the pin due to various junctions |
| RIC | = | Interconnect Resistance |
| RS | = | Source Impedance |
| VA | = | Analog Voltage |
| VT | = | Diode Forward Voltage |
| SS | = | Sampling Switch |
| RSS | = | Resistance of the Sampling Switch |
| CHOLD | = | Sample/Hold Capacitance |




**Note:**
1. Refer to the **"Electrical Specifications"** chapter of the device data sheet for more details.

**Figure 27-5. ADC Transfer Function**



## 27.4 Register Definitions: ADC Control

**Table 27-2. ADC Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| ADC | AD |

### 27.4.1 ADCON0

**Name:** ADCON0
**Address:** 0x09D

ADC Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CHS[5:0] | | | | | | GO | ON |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W/HS/HC | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:2 – CHS[5:0]** Analog Channel Select

| CHS | Selected Channel |
|---|---|
| 111111-011111 | Reserved |
| 011110 | FVR Buffer 1 |
| 011101-011100 | Reserved |
| 011011 | $V_{SS}$ |
| 011010-010110 | Reserved |
| 010101 | RC5(1) |
| 010100 | RC4(1) |
| 010011 | RC3(1) |
| 010010 | RC2(1) |
| 010001-010000 | Reserved |
| 001111 | RB7(2) |
| 001110 | RB6(2) |
| 001101 | RB5(2) |
| 001100-000110 | Reserved |
| 000101 | RA5 |
| 000100 | RA4 |
| 000011 | Reserved |
| 000010 | RA2 |
| 000001 | RA1 |
| 000000 | RA0 |

**Notes:**
1. Not available on 8-pin devices (PIC16F15213/14).
2. Only available on 20-pin devices (PIC16F15243/44/45).

**Bit 1 – GO** ADC Conversion Status

| Value | Description |
|---|---|
| 1 | ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. This bit is automatically cleared by hardware when the ADC conversion has completed. |
| 0 | ADC conversion completed/not in progress |

**Bit 0 – ON** ADC Enable

| Value | Description |
|---|---|
| 1 | ADC is enabled |
| 0 | ADC is disabled an consumes no operating current |

### 27.4.2 ADCON1

**Name:** ADCON1
**Address:** 0x09E

ADC Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FM | CS[2:0] | | | | | PREF[1:0] | |
| Access | R/W | R/W | R/W | R/W | | | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | | 0 | 0 |

**Bit 7 – FM** ADC Result Format/Alignment Selection

| Value | Description |
|---|---|
| 1 | Right-justified. The six Most Significant bits of ADRESH are zero-filled. |
| 0 | Left-justified. The six Least Significant bits of ADRESL are zero-filled. |

**Bits 6:4 – CS[2:0]** ADC Conversion Clock Select

| Value | Description |
|---|---|
| 111 | ADCRC |
| 110 | $F_{OSC}/64$ |
| 101 | $F_{OSC}/16$ |
| 100 | $F_{OSC}/4$ |
| 011 | ADCRC |
| 010 | $F_{OSC}/32$ |
| 001 | $F_{OSC}/8$ |
| 000 | $F_{OSC}/2$ |

**Bits 1:0 – PREF[1:0]** ADC Positive Voltage Reference Configuration

| Value | Description |
|---|---|
| 11 | $V_{REF}+$ is connected to internal Fixed Voltage Reference (FVR) |
| 10 | $V_{REF}+$ is connected to external $V_{REF}+$ pin |
| 01 | Reserved |
| 00 | $V_{REF}+$ is connected to $V_{DD}$ |

### 27.4.3 ADACT

**Name:** ADACT
**Address:** 0x09F

ADC Auto-Conversion Trigger Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | ACT[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – ACT[3:0]** Auto-Conversion Trigger Select

| ACT | Auto-conversion Trigger Source |
|---|---|
| 1111-1110 | Reserved |
| 1101 | Software read of ADRESH |
| 1100-1010 | Reserved |
| 1001 | Interrupt-on-change Interrupt Flag |
| 001000 | PWM4_out |
| 0111 | PWM3_out |
| 0110 | CCP2_trigger |
| 0101 | CCP1_trigger |
| 0100 | TMR2_postscaled |
| 0011 | TMR1_overflow |
| 0010 | TMR0_overflow |
| 0001 | Pin selected by ADACTPPS |
| 0000 | External Trigger Disabled |

### 27.4.4 ADRES

**Name:** ADRES
**Address:** 0x09B

ADC Result Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | ADRES[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADRES[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – ADRES[15:0]** ADC Sample Result

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- ADRESH: Accesses the high byte ADRES[15:18]
- ADRESL: Accesses the low byte ADRES[7:0]

## 27.5 Register Summary - ADC

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00<br>...<br>0x9A | Reserved | | | | | | | | | |
| 0x9B | ADRES | 7:0 | ADRES[7:0] | | | | | | | |
| | | 15:8 | ADRES[15:8] | | | | | | | |
| 0x9D | ADCON0 | 7:0 | CHS[5:0] | | | | | | GO | ON |
| 0x9E | ADCON1 | 7:0 | FM | CS[2:0] | | | | | PREF[1:0] | |
| 0x9F | ADACT | 7:0 | | | | | ACT[3:0] | | | |

# 28. Charge Pump

This family of devices offers a dedicated charge pump, which is controlled through the Charge Pump Control (CPCON) register. The primary purpose of the charge pump is to supply a constant voltage to the gates of transistor devices contained in analog peripherals, signal and reference input pass-gates, and to prevent degradation of transistor performance at low operating voltages.

The charge pump offers the following modes:

- Manually Enabled
- Automatically Enabled
- Disabled

## 28.1 Manually Enabled

The charge pump can be manually enabled via the Charge Pump Enable (CPON) bits. When the CPON bits are configured as '11', the charge pump is enabled. In this case, the charge pump provides additional voltage to all analog systems, regardless of $V_{DD}$ levels, but also consumes additional current.

## 28.2 Automatically Enabled

The charge pump can also be enabled automatically. This allows the application to determine when to enable the charge pump. If the charge pump is enabled while $V_{DD}$ levels are above a sufficient threshold, the charge pump does not improve analog performance, but also consumes additional current. Allowing hardware to monitor $V_{DD}$ and determine when to enable the charge pump prevents unnecessary current consumption.

When the CPON bits are configured as '10', charge pump hardware monitors $V_{DD}$ and compares the $V_{DD}$ levels to a reference voltage threshold ($V_{AUTO}$), which is set to 4.096V. When hardware detects a $V_{DD}$ level lower than the threshold, the charge pump is automatically enabled. If $V_{DD}$ returns to a level above the threshold, hardware automatically disables the charge pump.

When the CPON bits are configured as '01', charge pump hardware waits for an analog peripheral, such as the ADC, to be enabled before monitoring $V_{DD}$. In this case, charge pump hardware monitors all analog peripherals, and once an analog peripheral is enabled, hardware begins to compare $V_{DD}$ to $V_{AUTO}$. When hardware detects a $V_{DD}$ level lower than the threshold, hardware enables the charge pump. If $V_{DD}$ returns to a level above the threshold, or if the analog peripheral is disabled, the charge pump is automatically disabled.

## 28.3 Disabled

The charge pump is disabled by default (CPON = 00). Clearing the CPON bits will disable the charge pump.

## 28.4 Charge Pump Threshold

The Charge Pump Threshold (CPT) bit indicates whether or not $V_{DD}$ is at an acceptable operating level. Charge pump hardware compares $V_{DD}$ to the threshold voltage ($V_{AUTO}$), which is set at 4.096V. If $V_{DD}$ is above $V_{AUTO}$, the CPT bit is set (CPT = 1). If $V_{DD}$ is below $V_{AUTO}$, CPT is clear (CPT = 0).

## 28.5 Charge Pump Ready

The Charge Pump Ready Status (CPRDY) bit indicates whether or not the charge pump is ready for use. When CPRDY is set (CPRDY = 1), the charge pump has reached a steady-state operation and is ready for use. When CPRDY is clear (CPRDY = 0), the charge pump is either in the OFF state or has not reached a steady-state operation.

## 28.6 Register Definitions: Charge Pump

### 28.6.1 CPCON

**Name:** CPCON
**Address:** 0x09A

Charge Pump Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CPON[1:0] | | | | | | CPT | CPRDY |
| Access | R/W | R/W | | | | | R | R |
| Reset | 0 | 0 | | | | | 0 | 0 |

**Bits 7:6 – CPON[1:0]**
Charge Pump Enable

| Value | Description |
|---|---|
| 11 | Charge pump is enabled |
| 10 | Charge pump is automatically enabled when $V_{DD} < V_{AUTO}$ ($V_{AUTO}$ = 4.096V) |
| 01 | Charge pump is automatically enabled when an analog peripheral is enabled (ADCON0.ON = 1) **AND** $V_{DD} < V_{AUTO}$ |
| 00 | Charge pump is disabled |

**Bit 1 – CPT**
Charge Pump Threshold

| Value | Description |
|---|---|
| 1 | $V_{DD}$ is above the charge pump auto-enable threshold ($V_{AUTO}$) |
| 0 | $V_{DD}$ is below the charge pump auto-enable threshold ($V_{AUTO}$) |

**Bit 0 – CPRDY**
Charge Pump Ready Status

| Value | Description |
|---|---|
| 1 | Charge pump has reached a steady-state operation |
| 0 | Charge pump is Off or has not reached a steady-state operation |

# 29. Instruction Set Summary

PIC16F152 devices incorporate the standard set of 50 PIC16 core instructions. Each instruction is a 14-bit word containing the operation code (opcode) and all required operands. The opcodes are broken into three broad categories:

- Byte Oriented
- Bit Oriented
- Literal and Control

The literal and control category contains the most varied instruction word format.

Table 29-3 lists the instructions recognized by the XC8 assembler.

All instructions are executed within a single instruction cycle, with the following exceptions, which may take two or three cycles:

- Subroutine entry takes two cycles (`CALL`, `CALLW`)
- Returns from interrupts or subroutines take two cycles (`RETURN`, `RETLW`, `RETFIE`)
- Program branching takes two cycles (`GOTO`, `BRA`, `BRW`, `BTFSS`, `BTFSC`, `DECFSZ`, `INCSFZ`)
- One additional instruction cycle will be used when any instruction references an indirect file register and the file select register is pointing to program memory.

One instruction cycle consists of 4 oscillator cycles; for an oscillator frequency of 4 MHz, this gives a nominal instruction execution rate of 1 MHz.

All instruction examples use the format '`0xhh`' to represent a hexadecimal number, where '`h`' signifies a hexadecimal digit.

## 29.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the working (W) register, or the originating file register, depending on the state of the destination designator 'd' (see Table 29-1 for more information). A read operation is performed on a register even if the instruction writes to that register.

**Table 29-1. Opcode Field Descriptions**

| Field | Description |
|---|---|
| f | Register file address (`0x00` to `0x7F`) |
| W | Working register (accumulator) |
| b | Bit address within an 8-bit file register |
| k | Literal field, constant data or label |
| x | Don't care location (= `0` or `1`).<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, d = `1`: store result in file register f. |
| n | FSR or INDF number. (0-1) |
| mm | Pre/post increment/decrement mode selection |

**Table 29-2. Abbreviation Descriptions**

| Field | Description |
|---|---|
| PC | Program Counter |

.....continued

| Field | Description |
|---|---|
| $\overline{TO}$ | Time-Out bit |
| C | Carry bit |
| DC | Digit Carry bit |
| Z | Zero bit |
| $\overline{PD}$ | Power-Down bit |

## 29.2 Standard Instruction Set

**Table 29-3. Instruction Set**

| Mnemonic, Operands | | Description | Cycles | 14-Bit Opcode | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **BYTE-ORIENTED OPERATIONS** | | | | | | | | | |
| ADDWF | f, d | Add WREG and f | 1 | 00 | 0111 | dfff | ffff | C, DC, Z | 2 |
| ADDWFC | f, d | Add WREG and CARRY bit to f | 1 | 11 | 1101 | dfff | ffff | C, DC, Z | 2 |
| ANDWF | f, d | AND WREG 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 WREG | 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 WREG with f | 1 | 00 | 0100 | dfff | ffff | Z | 2 |
| MOVF | f, d | Move f | 1 | 00 | 1000 | dfff | ffff | Z | 2 |
| MOVWF | f | Move WREG to f | 1 | 00 | 0000 | 1fff | ffff | None | 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 |

.....continued

| Mnemonic, Operands | | Description | Cycles | 14-Bit Opcode | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| SUBWF | f, d | Subtract WREG from f | 1 | 00 | 0010 | dfff | ffff | C, DC, Z | 2 |
| SUBWFB | f, d | Subtract WREG from f with borrow | 1 | 11 | 1011 | dfff | ffff | C, DC, Z | 2 |
| SWAPF | f, d | Swap nibbles in f | 1 | 00 | 1110 | dfff | ffff | None | 2 |
| XORWF | f, d | Exclusive OR WREG 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 | None | 1, 2 |
| INCFSZ | f, d | Increment f, Skip if 0 | 1(2) | 00 | 1111 | dfff | ffff | None | 1, 2 |
| **BIT-ORIENTED FILE REGISTER OPERATIONS** | | | | | | | | | |
| BCF | f, b | Bit Clear f | 1 | 01 | 00bb | bfff | ffff | None | 2 |
| BSF | f, b | Bit Set f | 1 | 01 | 01bb | bfff | ffff | None | 2 |
| **BIT-ORIENTED SKIP OPERATIONS** | | | | | | | | | |
| BTFSC | f, b | Bit Test f, Skip if Clear | 1(2) | 01 | 10bb | bfff | ffff | None | 1, 2 |
| BTFSS | f, b | Bit Test f, Skip if Set | 1(2) | 1010 | 11bb | bfff | ffff | None | 1, 2 |
| **LITERAL OPERATIONS** | | | | | | | | | |
| ADDLW | k | Add literal and WREG | 1 | 11 | 1110 | kkkk | kkkk | C, DC, Z | |
| ANDLW | k | AND literal with WREG | 1 | 11 | 1001 | kkkk | kkkk | Z | |
| IORLW | k | Inclusive OR literal with WREG | 1 | 11 | 1000 | kkkk | kkkk | Z | |
| MOVLB | k | Move literal to BSR | 1 | 00 | 000 | 0k | kkkk | None | |
| MOVLP | k | Move literal to PCLATH | 1 | 11 | 0001 | 1kkk | kkkk | None | |
| MOVLW | k | Move literal to W | 1 | 11 | 0000 | kkkk | kkkk | None | |
| 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 | |
| **CONTROL OPERATIONS** | | | | | | | | | |

.....continued

| Mnemonic, Operands | | Description | Cycles | 14-Bit Opcode | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| BRA | k | Relative Branch | 2 | 11 | 001k | kkkk | kkkk | None | |
| BRW | — | Relative Branch with WREG | 2 | 00 | 0000 | 0000 | 1011 | None | |
| CALL | k | Call Subroutine | 2 | 10 | 0kkk | kkkk | kkkk | None | |
| CALLW | — | Call Subroutine with WREG | 2 | 00 | 0000 | 0000 | 1010 | None | |
| GOTO | k | Go to address | 2 | 10 | 1kkk | kkkk | kkkk | None | |
| RETFIE | k | Return from interrupt | 2 | 00 | 0000 | 0000 | 1001 | None | |
| RETLW | k | Return with literal in WREG | 2 | 11 | 0100 | kkkk | kkkk | None | |
| RETURN | — | Return from Subroutine | 2 | 00 | 0000 | 0000 | 1000 | None | |
| **INHERENT OPERATIONS** | | | | | | | | | |
| CLRWDT | — | Clear Watchdog Timer | 1 | 00 | 0000 | 0110 | 0100 | $\overline{TO}$, $\overline{PD}$ | |
| NOP | — | No Operation | 1 | 00 | 0000 | 0000 | 0000 | None | |
| RESET | — | Software device Reset | 1 | 00 | 0000 | 0000 | 0001 | None | |
| SLEEP | — | Go into Standby mode | 1 | 00 | 0000 | 0110 | 0011 | $\overline{TO}$, $\overline{PD}$ | |
| TRIS | f | Load TRIS register with WREG | 1 | 00 | 0000 | 0110 | 0fff | None | |
| **C-COMPILER OPTIMIZED** | | | | | | | | | |
| ADDFSR | n, k | Add Literal k to FSRn | 1 | 11 | 0001 | 0nkk | kkkk | None | |
| MOVIW | n, mm | Move Indirect FSRn to WREG with pre/post inc/dec modifier, mm | 1 | 00 | 0000 | 0001 | 0nmm | Z | 2, 3 |
| | k[n] | Move INDFn to WREG, Indexed Indirect. | 1 | 11 | 1111 | 0nkk | kkkk | Z | 2 |
| MOVWI | n, mm | Move WREG to Indirect FSRn with pre/post inc/dec modifier, mm | 1 | 00 | 0000 | 0001 | 1nmm | None | 2, 3 |
| | k[n] | Move WREG to INDFn, Indexed Indirect. | 1 | 11 | 1111 | 1nkk | kkkk | None | 2 |

**Notes:**

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. Details on `MOVIW` and `MOVWI` instruction descriptions are available in the next section.

### 29.2.1 Standard Instruction Set

| 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>FSRn is limited to the range 0000h-FFFFh. Moving beyond these bounds will cause the FSR to wrap-around. |

| ADDLW | ADD literal to W |
|---|---|
| Syntax: | [ *label* ] ADDLW k |
| Operands: | 0 ≤ k ≤ 255 |
| Operation: | (W) + k → (W) |
| Status Affected: | C, DC, Z |
| Description: | The contents of W are added to the 8-bit literal 'k' and the result is placed in W. |

| ADDWF | ADD W to f |
|---|---|
| Syntax: | [ *label* ] ADDWF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (W) + (f) → dest |
| 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) |

| .....continued | |
|---|---|
| **ANDLW** | **AND literal with W** |
| Status Affected: | Z |
| Description: | The contents of W are AND'ed with the 8-bit literal 'k'.<br>The result is placed in W. |

| ANDWF | AND W with f |
|---|---|
| Syntax: | [ *label* ] ANDWF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (W) .AND. (f) → dest |
| Status Affected: | Z |
| Description: | AND the W register with register 'f'. If 'd' is '0', the result is stored in the W register.<br>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.<br>The MSb remains unchanged.<br>If 'd' is '0', the result is placed in W.<br>If 'd' is '1', the result is stored back in register 'f'.<br>**Register f → C** |

| 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 + ≤ 255<br>-256 ≤ k ≤ 255 |

.....continued

| BRA | Relative Branch |
|---|---|
| Operation: | (PC) + 1 + k → PC |
| Status Affected: | None |
| Description: | Add the signed 9-bit literal 'k' to the PC.<br>Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + k.<br>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.<br>Since the PC will have incremented to fetch the next instruction, the new address will be PC + 1 + (W).<br>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 File, 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,<br>and a `NOP` is executed instead, making this a 2-cycle instruction. |

| BTFSS | Bit Test File, Skip if Set |
|---|---|
| Syntax: | [ *label* ] BTFSS f, b |
| Operands: | 0 ≤ f ≤ 127<br>0 ≤ b < 7 |
| Operation: | skip if (f[b]) = 1 |

.....continued

| BTFSS | Bit Test File, Skip if Set |
|---|---|
| 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,<br>and a NOP is executed instead, making this a 2-cycle instruction. |

| CALL | Subroutine Call |
|---|---|
| Syntax: | [ *label* ] CALL k |
| Operands: | 0 ≤ k ≤ 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.<br>The 11-bit immediate address is loaded into PC bits [10:0].<br>The upper bits of the PC are loaded from PCLATH.<br>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.<br>First, the return address (PC + 1) is pushed onto the return stack.<br>Then, the contents of W is loaded into PC[7:0], and the contents of PCLATH into PC[14:8].<br>CALLW is a 2-cycle instruction. |

| CLRF | Clear f |
|---|---|
| Syntax: | [ *label* ] CLRF f |
| Operands: | 0 ≤ f ≤ 127 |
| Operation: | 000h → 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 |

.....continued

| CLRW | Clear W |
|---|---|
| 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>00h → WDT prescaler,<br>1 → $\overline{TO}$,<br>1 → $\overline{PD}$ |
| Status Affected: | $\overline{TO}$, $\overline{PD}$ |
| Description: | `CLRWDT` instruction resets the Watchdog Timer.<br>It also resets the prescaler of the WDT.<br>Status bits, $\overline{TO}$ and $\overline{PD}$, are set. |

| COMF | Complement f |
|---|---|
| Syntax: | [ *label* ] COMF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | ($\bar{f}$) → dest |
| Status Affected: | Z |
| Description: | The contents of register 'f' are complemented.<br>If 'd' is '0', the result is stored in W.<br>If 'd' is '1', the result is stored back in register 'f'. |

| DECF | Decrement f |
|---|---|
| Syntax: | [ *label* ] DECF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (f) – 1 → dest |
| Status Affected: | Z |
| Description: | Decrement register 'f'.<br>If 'd' is '0', the result is stored in the W register.<br>If 'd' is '1', the result is stored back in register 'f'. |

| DECFSZ | Decrement f, skip if 0 |
|---|---|
| Syntax: | [ *label* ] DECFSZ f, d |

.....continued

| DECFSZ | Decrement f, skip if 0 |
|---|---|
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (f) – 1 → dest,<br>skip if result = 0 |
| Description: | The contents of register 'f' are decremented.<br>If 'd' is '0', the result is placed in the W register.<br>If 'd' is '1', the result is placed back in register 'f'.<br>If the result is '1', the next instruction is executed.<br>If the result is '0', then a NOP is executed instead,<br>making it a 2-cycle instruction. |

| GOTO | Unconditional Branch |
|---|---|
| Syntax: | [ *label* ] GOTO k |
| Operands: | 0 ≤ k ≤ 2047 |
| Operation: | k → PC[10:0]<br>PCLATH[6:3] → PC[14:11] |
| Status Affected: | None |
| Description: | GOTO is an unconditional branch.<br>The 11-bit immediate value is loaded into PC bits [10:0].<br>The upper bits of PC are loaded from PCLATH[4:3].<br>GOTO is a 2-cycle instruction. |

| INCF | Increment f |
|---|---|
| Syntax: | [ *label* ] INCF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (f) + 1 → dest |
| Status Affected: | Z |
| Description: | The contents of register 'f' are incremented.<br>If 'd' is '0', the result is placed in the W register.<br>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 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (f) + 1 → dest,<br>skip if result = 0 |
| Status Affected: | None |

.....continued

| INCFSZ | Increment f, skip if 0 |
|---|---|
| Description: | The contents of register 'f' are incremented.<br>If 'd' is '0', the result is placed in the W register.<br>If 'd' is '1', the result is placed back in register 'f'.<br>If the result is '1', the next instruction is executed.<br>If the result is '0', a NOP is executed instead,<br>making it a 2-cycle instruction. |

| IORLW | Inclusive OR literal with W |
|---|---|
| Syntax: | [ *label* ] IORLW k |
| Operands: | 0 ≤ k ≤ 255 |
| Operation: | (W) .OR. k → (W) |
| Status Affected: | Z |
| Description: | The contents of W are ORed with the 8-bit literal 'k'.<br>The result is placed in W. |

| IORWF | Inclusive OR W with f |
|---|---|
| Syntax: | IORWF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (W) .OR. (f) → dest |
| Status Affected: | Z |
| Description: | Inclusive OR the W register with register 'f'.<br>If 'd' is '0', the result is placed in the W register.<br>If 'd' is '1', the result is placed back in register 'f'. |

| LSLF | Logical Left Shift |
|---|---|
| Syntax: | [ *label* ] LSLF f {,d} |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (f[7]) → C<br>(f[6:0]) → dest[7:1]<br>0 → dest[0] |
| Status Affected: | C, Z |
| Description: | The contents of register 'f' are shifted one bit to the left through the Carry flag.<br>A '0' is shifted into the LSb.<br>If 'd' is '0', the result is placed in W.<br>If 'd' is '1', the result is stored back in register 'f'.<br>**C ← Register f ← 0** |

| LSRF | Logical Right Shift |
|---|---|
| Syntax: | [ *label* ] LSRF f {,d} |

.....continued

| LSRF | Logical Right Shift |
|---|---|
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | 0 → dest[7]<br>(f[7:1]) → dest[6:0],<br>(f[0]) → C |
| Status Affected: | C, Z |
| Description: | The contents of register 'f' are shifted one bit to the right through the Carry flag.<br>A '0' is shifted into the MSb.<br>If 'd' is '0', the result is placed in W.<br>If 'd' is '1', the result is stored back in register 'f'.<br>**0 → register f → C** |

| MOVF | Move f | |
|---|---|---|
| Syntax: | [ *label* ] MOVF f, d | |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [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.<br>If d = 0, destination is W register.<br>If d = 1, the destination is file register f itself.<br>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<br>W = value in FSR register<br>Z = 1 | |

| MOVIW | Move INDFn to W |
|---|---|
| Syntax: | [ *label* ] MOVIW ++FSRn<br>[ *label* ] MOVIW --FSRn<br>[ *label* ] MOVIW FSRn++<br>[ *label* ] MOVIW FSRn--<br>[ *label* ] MOVIW k[FSRn] |
| Operands: | n ∈ [0,1]<br>mm ∈ [00,01,10,11]<br>-32 ≤ k ≤ 31 |

.....continued

| MOVIW | Move INDFn to W |
|---|---|
| Operation: | INDFn → (W)<br>Effective address is determined by<br>• FSR + 1 (preincrement)<br>• FSR - 1 (predecrement)<br>• FSR + k (relative offset)<br>After the Move, the FSR value will be either:<br>• FSR + 1 (all increments)<br>• FSR - 1 (all decrements)<br>• Unchanged |
| Status Affected: | Z |

| MODE | SYNTAX | mm |
|---|---|---|
| Preincrement | ++FSRn | 00 |
| Predecrement | --FSRn | 01 |
| Postincrement | FSRn++ | 10 |
| Postdecrement | FSRn-- | 11 |

| | |
|---|---|
| Description: | This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it. 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 ≤ 127 |
| Operation: | k → BSR |
| Status Affected: | None |
| Description: | The 6-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 |

.....continued

| MOVLW | Move literal to W | | | |
|---|---|---|---|---|
| Operation: | k → (W) | | | |
| Status Affected: | None | | | |
| Description: | The 8-bit literal 'k' is loaded into W register.<br>The "don't cares" will assemble as '0's. | | | |
| Words: | 1 | | | |
| Cycles: | 1 | | | |

| Example: | MOVLW | 5Ah |
|---|---|---|
| After Instruction<br>W = 5Ah | | |

| MOVWF | Move W to f | | | |
|---|---|---|---|---|
| Syntax: | [ *label* ] MOVWF f | | | |
| Operands: | 0 ≤ f ≤ 127 | | | |
| Operation: | (W) → f | | | |
| Status Affected: | None | | | |
| Description: | Move data from W to register 'f'. | | | |
| Words: | 1 | | | |
| Cycles: | 1 | | | |

| Example: | MOVWF | LATA |
|---|---|---|
| Before Instruction<br>LATA = FFh<br>W = 4Fh<br>After Instruction<br>LATA = 4Fh<br>W = 4Fh | | |

| MOVWI | Move W to INDFn |
|---|---|
| Syntax: | [ *label* ] MOVWI ++FSRn<br>[ *label* ] MOVWI --FSRn<br>[ *label* ] MOVWI FSRn++<br>[ *label* ] MOVWI FSRn--<br>[ *label* ] MOVWI k[FSRn] |
| Operands: | n ∈ [0,1]<br>mm ∈ [00,01,10,11]<br>-32 ≤ k ≤ 31 |

.....continued

| MOVWI | Move W to INDFn |
|---|---|
| Operation: | (W) → INDFn<br>Effective address is determined by<br>• FSR + 1 (preincrement)<br>• FSR - 1 (predecrement)<br>• FSR + k (relative offset)<br>After the Move, the FSR value will be either:<br>• FSR + 1 (all increments)<br>• FSR - 1 (all decrements)<br>• Unchanged |
| Status Affected: | None |

| MODE | SYNTAX | mm |
|---|---|---|
| Preincrement | ++FSRn | 00 |
| Predecrement | --FSRn | 01 |
| Postincrement | FSRn++ | 10 |
| Postdecrement | FSRn-- | 11 |

| | |
|---|---|
| Description: | This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it. 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 |
|---|---|
| None. | |

| RESET | Software Reset |
|---|---|
| Syntax: | [ *label* ] RESET |
| Operands: | None |
| Operation: | Execute a device Reset.<br>Resets the $\overline{RI}$ flag of the PCON register. |

.....continued

| RESET | Software Reset |
|---|---|
| Status Affected: | None |
| Description: | This instruction provides a way to execute a hardware Reset by software. |

| RETFIE | Return from Interrupt | | | |
|---|---|---|---|---|
| Syntax: | [ *label* ] RETFIE k | | | |
| Operands: | None | | | |
| Operation: | (TOS) → PC,<br>1 → GIE | | | |
| Status Affected: | None | | | |
| Description: | Return from Interrupt.<br>Stack is POPed and Top-of-Stack (TOS) is loaded in the PC.<br>Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON[7]).<br>This is a 2-cycle instruction. | | | |
| Words: | 1 | | | |
| Cycles: | 2 | | | |

| Example: | RETFIE |
|---|---|
| After Interrupt<br>PC = TOS<br>GIE = 1 | |

| RETLW | Return literal to W | | | |
|---|---|---|---|---|
| Syntax: | [ *label* ] RETLW k | | | |
| Operands: | 0 ≤ k ≤ 255 | | | |
| Operation: | k → (W),<br>(TOS) → PC, | | | |
| Status Affected: | None | | | |
| Description: | The W register is loaded with the 8-bit literal 'k'.<br>The program counter is loaded from the top of the stack (the return address).<br>This is a 2-cycle instruction. | | | |
| Words: | 1 | | | |
| Cycles: | 2 | | | |

Example:

```
      CALL TABLE            ; W contains table
; offset value
; W now has
; table value
   :
TABLE
ADDWF PC            ; W = offset
RETLW k1            ; Begin table
RETLW k2            ;
:
:
RETLW kn            ; End of table
```

Before Instruction
W = 07h

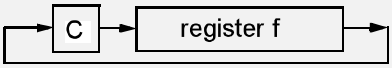
After Instruction

W = value of k8

| RETURN | Return from Subroutine | | | |
|---|---|---|---|---|
| Syntax: | [ *label* ] RETURN | | | |
| Operands: | None | | | |
| Operation: | (TOS) → PC, | | | |
| Status Affected: | None | | | |
| Encoding: | 0000 | 0000 | 0001 | 001s |
| Description: | Return from subroutine.<br>The stack is POPped and the top of the stack (TOS) is loaded into the Program Counter.<br>This is a 2-cycle instruction. | | | |

| RLF | Rotate Left f through Carry | | | |
|---|---|---|---|---|
| Syntax: | [ *label* ] RLF f, d | | | |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] | | | |
| Operation: | (f[n]) → dest[n + 1],<br>(f[7]) → C,<br>(C) → dest[0] | | | |
| Status Affected: | C | | | |
| Encoding: | 0011 | 01da | ffff | ffff |
| Description: | The contents of register 'f' are rotated one bit to the left through the CARRY flag.<br>If 'd' is '0', the result is placed in W.<br>If 'd' is '1', the result is stored back in register 'f' (default).<br> | | | |
| Words: | 1 | | | |
| Cycles: | 1 | | | |

| Example: | RLF | REG1, 0 |
|---|---|---|
| Before Instruction<br>REG1 = 1110 0110<br>C = 0<br>After Instruction<br>REG = 1110 0110<br>W = 1100 1100<br>C = 1 | | |

| RRF | Rotate Right f through Carry |
|---|---|
| Syntax: | [ *label* ] RRF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (f[n]) → dest[n – 1],<br>(f[0]) → C,<br>(C) → dest[7] |
| Status Affected: | C |
| Description: | The contents of register 'f' are rotated one bit to the right through the CARRY flag.<br>If 'd' is '0', the result is placed in W.<br>If 'd' is '1', the result is placed back in register 'f' (default).<br> |

| SLEEP | Enter Sleep mode |
|---|---|
| Syntax: | [ *label* ] SLEEP |
| Operands: | None |
| Operation: | 00h → WDT,<br>0 → WDT prescaler,<br>1 → $\overline{TO}$,<br>0 → $\overline{PD}$ |
| Status Affected: | $\overline{TO}$, $\overline{PD}$ |
| Description: | The Power-down Status bit ($\overline{PD}$) is cleared.<br>The Time-out Status bit ($\overline{TO}$) is set.<br>Watchdog Timer and its prescaler are cleared. |

| SUBLW | Subtract W from literal |
|---|---|
| Syntax: | [ *label* ] SUBLW k |
| Operands: | 0 ≤ k ≤ 255 |
| Operation: | k – (W) → (W) |
| Status Affected: | C, DC, Z |

| .....continued | |
|---|---|
| **SUBLW** | **Subtract W from literal** |
| Description | The W register is subtracted (2's complement method) from the 8-bit literal 'k'.<br>The result is placed in the W register.<br>C =0, W > k<br>C = 1, W ≤ k<br>DC = 0, W[3:0] > k[3:0]<br>DC = 1, W[3:0] ≤ k[3:0] |

| SUBWF | Subtract W from f |
|---|---|
| Syntax: | [ *label* ] SUBWF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (f) - (W) → (dest) |
| Status Affected: | C, DC, Z |
| Description | Subtract (2's complement method) W register from register 'f'.<br>If 'd' is '0', the result is stored in the W register.<br>If 'd' is '1', the result is stored back in register 'f.<br>C =0, W > f<br>C = 1, W ≤ f<br>DC = 0, W[3:0] > f[3:0]<br>DC = 1, W[3:0] ≤ f[3:0] |

| SUBWFB | Subtract W from f with Borrow |
|---|---|
| Syntax: | [ *label* ] SUBFWB f {,d} |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (W) – (f) – ($\overline{B}$) → dest |
| Status Affected: | C, DC, Z |
| Description: | Subtract W and the BORROW flag (CARRY) from register 'f' (2's complement method).<br>If 'd' is '0', the result is stored in W.<br>If 'd' is '1', the result is stored back in register 'f'. |

| SWAPF | Swap Nibbles in f |
|---|---|
| Syntax: | [ *label* ] SWAPF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (f[3:0]) → dest[7:4],<br>(f[7:4]) → dest[3:0] |
| Status Affected: | None |

.....continued

| SWAPF | Swap Nibbles in f |
|---|---|
| Description: | The upper and lower nibbles of register 'f' are exchanged.<br>If 'd' is '0', the result is placed in W.<br>If 'd' is '1', the result is placed in register 'f' (default). |

| TRIS | Load TRIS Register with W |
|---|---|
| Syntax: | [ *label* ] TRIS f |
| Operands: | 5 ≤ f ≤ 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 ≤ k ≤ 255 |
| Operation: | (W) .XOR. k → (W) |
| Status Affected: | Z |
| Description: | The contents of W are XORed with the 8-bit literal 'k'.<br>The result is placed in W. |

| XORWF | Exclusive OR W with f |
|---|---|
| Syntax: | [ *label* ] XORWF f, d |
| Operands: | 0 ≤ f ≤ 127<br>d ∈ [0,1] |
| Operation: | (W) .XOR. (f) → dest |
| Status Affected: | Z |
| Description: | Exclusive OR the contents of the W register with register 'f'.<br>If 'd' is '0', the result is stored in the W register.<br>If 'd' is '1', the result is stored back in register 'f'. |

# 30. ICSP™ - In-Circuit Serial Programming™

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
- $\overline{MCLR}/V_{PP}$
- $V_{DD}$
- $V_{SS}$

In Program/Verify mode the program memory, User IDs and the Configuration bits 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 **"Family Programming Specification"**

## 30.1 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 $\overline{MCLR}/V_{PP}$ to $V_{IHH}$.

## 30.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using $V_{DD}$ only, without high voltage. When the LVP Configuration bit is set to '1', the low-voltage ICSP programming entry is enabled. To disable the Low-Voltage ICSP mode, the LVP bit must be programmed to '0'.

Entry into the Low-Voltage Programming Entry mode requires the following steps:

1. $\overline{MCLR}$ 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 $V_{IL}$ for as long as Program/Verify mode is to be maintained.

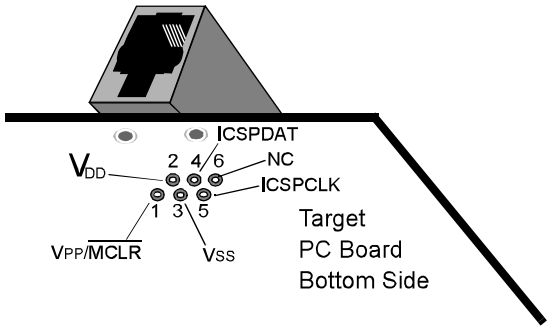
If low-voltage programming is enabled (LVP = 1), the $\overline{MCLR}$ Reset function is automatically enabled and cannot be disabled. See the $\overline{MCLR}$ Section for more information.

The LVP bit can only be reprogrammed to '0' by using the High-Voltage Programming mode.

## 30.3 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 30-1.

**Figure 30-1. ICD RJ-11 Style Connector Interface**




Pin Description

1 = $V_{PP}/\overline{MCLR}$

2 = $V_{DD}$ Target

3 = $V_{SS}$ (ground)

4 = ICSPDAT

5 = ICSPCLK

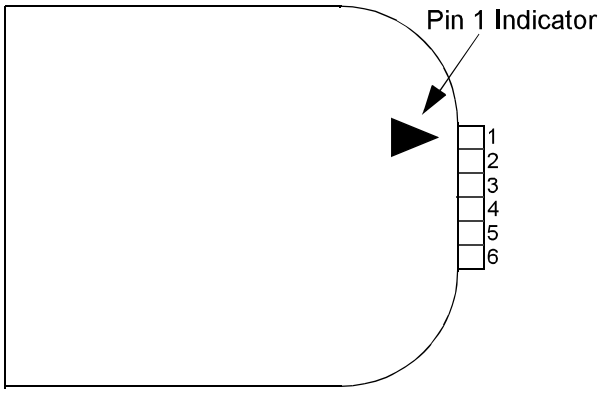
6 = No Connect

Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 30-2.

For additional interface recommendations, refer to the 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 30-3 for more information.

**Figure 30-2. PICkit™ Programmer Style Connector Interface**



Pin Description[1]:

1 = $V_{PP}/\overline{MCLR}$

2 = $V_{DD}$ Target

3 = $V_{SS}$ (ground)

4 = ICSPDAT

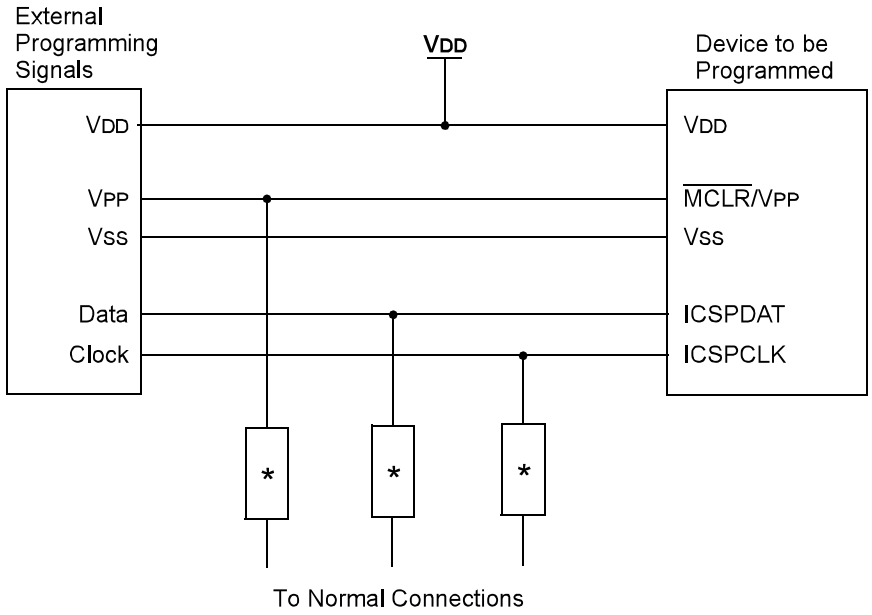
5 = ICSPCLK

6 = No Connect

**Note:**

1. The 6-pin header (0.100" spacing) accepts 0.025" square pins.

**Figure 30-3. Typical Connection for ICSP™ Programming**




* Isolation devices (as required).

# 31. Register Summary

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x00 | INDF0 | 7:0 | INDF0[7:0] | | | | | | | |
| 0x01 | INDF1 | 7:0 | INDF1[7:0] | | | | | | | |
| 0x02 | PCL | 7:0 | PCL[7:0] | | | | | | | |
| 0x03 | STATUS | 7:0 | | | | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C |
| 0x04 | FSR0 | 7:0 | FSR0[7:0] | | | | | | | |
| | | 15:8 | FSR0[15:8] | | | | | | | |
| 0x06 | FSR1 | 7:0 | FSR1[7:0] | | | | | | | |
| | | 15:8 | FSR1[15:8] | | | | | | | |
| 0x08 | BSR | 7:0 | | | BSR[5:0] | | | | | |
| 0x09 | WREG | 7:0 | WREG[7:0] | | | | | | | |
| 0x0A | PCLATH | 7:0 | | PCLATH[6:0] | | | | | | |
| 0x0B | Reserved | | | | | | | | | |
| 0x0C | PORTA | 7:0 | | | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 |
| 0x0D | PORTB | 7:0 | RB7 | RB6 | RB5 | RB4 | | | | |
| 0x0E | PORTC | 7:0 | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 |
| 0x0F ... 0x11 | Reserved | | | | | | | | | |
| 0x12 | TRISA | 7:0 | | | TRISA5 | TRISA4 | Reserved | TRISA2 | TRISA1 | TRISA0 |
| 0x13 | TRISB | 7:0 | TRISB7 | TRISB6 | TRISB5 | TRISB4 | | | | |
| 0x14 | TRISC | 7:0 | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| 0x15 ... 0x17 | Reserved | | | | | | | | | |
| 0x18 | LATA | 7:0 | | | LATA5 | LATA4 | | LATA2 | LATA1 | LATA0 |
| 0x19 | LATB | 7:0 | LATB7 | LATB6 | LATB5 | LATB4 | | | | |
| 0x1A | LATC | 7:0 | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 |
| 0x1B ... 0x99 | Reserved | | | | | | | | | |
| 0x9A | CPCON | 7:0 | CPON[1:0] | | | | | | CPT | CPRDY |
| 0x9B | ADRES | 7:0 | ADRES[7:0] | | | | | | | |
| | | 15:8 | ADRES[15:8] | | | | | | | |
| 0x9D | ADCON0 | 7:0 | CHS[5:0] | | | | | | GO | ON |
| 0x9E | ADCON1 | 7:0 | FM | CS[2:0] | | | | | PREF[1:0] | |
| 0x9F | ADACT | 7:0 | | | | | ACT[3:0] | | | |
| 0xA0 ... 0x010B | Reserved | | | | | | | | | |
| 0x010C | RB4I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x010D | RB6I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x010E | RC0I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x010F | RC1I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x0110 ... 0x0118 | Reserved | | | | | | | | | |
| 0x0119 | RC1REG | 7:0 | RCREG[7:0] | | | | | | | |
| 0x011A | TX1REG | 7:0 | TXREG[7:0] | | | | | | | |
| 0x011B | SP1BRG | 7:0 | SPBRG[7:0] | | | | | | | |
| | | 15:8 | SPBRG[15:8] | | | | | | | |
| 0x011D | RC1STA | 7:0 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| 0x011E | TX1STA | 7:0 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| 0x011F | BAUD1CON | 7:0 | ABDOVF | RCIDL | | SCKP | BRG16 | | WUE | ABDEN |
| 0x0120 ... 0x018B | Reserved | | | | | | | | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x018C | SSP1BUF | 7:0 | BUF[7:0] | | | | | | | |
| 0x018D | SSP1ADD | 7:0 | ADD[7:0] | | | | | | | |
| 0x018E | SSP1MSK | 7:0 | MSK[6:0] | | | | | | | MSK0 |
| 0x018F | SSP1STAT | 7:0 | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| 0x0190 | SSP1CON1 | 7:0 | WCOL | SSPOV | SSPEN | CKP | SSPM[3:0] | | | |
| 0x0191 | SSP1CON2 | 7:0 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| 0x0192 | SSP1CON3 | 7:0 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN |
| 0x0193 ... 0x020B | Reserved | | | | | | | | | |
| 0x020C | TMRx | 7:0 | TMR1[7:0] | | | | | | | |
| | | 15:8 | TMR1[15:8] | | | | | | | |
| 0x020E | TxCON | 7:0 | | | CKPS[1:0] | | | $\overline{SYNC}$ | RD16 | ON |
| 0x020F | TxGCON | 7:0 | GE | GPOL | GTM | GSPM | GGO/$\overline{DONE}$ | GVAL | | |
| 0x0210 | TxGATE | 7:0 | | | | GSS[4:0] | | | | |
| 0x0211 | TxCLK | 7:0 | | | | CS[4:0] | | | | |
| 0x0212 ... 0x028B | Reserved | | | | | | | | | |
| 0x028C | TxTMR | 7:0 | T2TMR[7:0] | | | | | | | |
| 0x028D | TxPR | 7:0 | T2PR[7:0] | | | | | | | |
| 0x028E | TxCON | 7:0 | ON | | CKPS[2:0] | | OUTPS[3:0] | | | |
| 0x028F | TxHLT | 7:0 | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| 0x0290 | TxCLKCON | 7:0 | | | | | | CS[2:0] | | |
| 0x0291 | TxRST | 7:0 | | | | | RSEL[3:0] | | | |
| 0x0292 ... 0x030B | Reserved | | | | | | | | | |
| 0x030C | CCPR1 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x030E | CCP1CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x030F | CCP1CAP | 7:0 | | | | | | | CTS[1:0] | |
| 0x0310 | CCPR2 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x0312 | CCP2CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x0313 | CCP2CAP | 7:0 | | | | | | | CTS[1:0] | |
| 0x0314 | PWM3DC | 7:0 | DCL[1:0] | | | | | | | |
| | | 15:8 | DCH[7:0] | | | | | | | |
| 0x0316 | PWM3CON | 7:0 | EN | | OUT | POL | | | | |
| 0x0317 | Reserved | | | | | | | | | |
| 0x0318 | PWM4DC | 7:0 | DCL[1:0] | | | | | | | |
| | | 15:8 | DCH[7:0] | | | | | | | |
| 0x031A | PWM4CON | 7:0 | EN | | OUT | POL | | | | |
| 0x031B ... 0x059B | Reserved | | | | | | | | | |
| 0x059C | TMR0L | 7:0 | TMR0L[7:0] | | | | | | | |
| 0x059D | TMR0H | 7:0 | TMR0H[7:0] | | | | | | | |
| 0x059E | T0CON0 | 7:0 | EN | | OUT | MD16 | OUTPS[3:0] | | | |
| 0x059F | T0CON1 | 7:0 | CS[2:0] | | | ASYNC | CKPS[3:0] | | | |
| 0x05A0 ... 0x070B | Reserved | | | | | | | | | |
| 0x070C | PIR0 | 7:0 | | | TMR0IF | IOCIF | | | | INTF |
| 0x070D | PIR1 | 7:0 | CCP1IF | TMR2IF | TMR1IF | RC1IF | TX1IF | BCL1IF | SSP1IF | ADIF |
| 0x070E | PIR2 | 7:0 | CCP2IF | NVMIF | TMR1GIF | | | | | |
| 0x070F ... 0x0715 | Reserved | | | | | | | | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0716 | PIE0 | 7:0 | | | TMR0IE | IOCIE | | | | INTE |
| 0x0717 | PIE1 | 7:0 | CCP1IE | TMR2IE | TMR1IE | RC1IE | TX1IE | BCL1IE | SSP1IE | ADIE |
| 0x0718 | PIE2 | 7:0 | CCP2IE | NVMIE | TMR1GIE | | | | | |
| 0x0719 ... 0x080B | Reserved | | | | | | | | | |
| 0x080C | WDTCON | 7:0 | CS | | PS[4:0] | | | | | SEN |
| 0x080D ... 0x0810 | Reserved | | | | | | | | | |
| 0x0811 | BORCON | 7:0 | SBOREN | | | | | | | BORRDY |
| 0x0812 | Reserved | | | | | | | | | |
| 0x0813 | PCON0 | 7:0 | STKOVF | STKUNF | | $\overline{RWDT}$ | $\overline{RMCLR}$ | $\overline{RI}$ | $\overline{POR}$ | $\overline{BOR}$ |
| 0x0814 | PCON1 | 7:0 | | | | | | | $\overline{MEMV}$ | |
| 0x0815 ... 0x0819 | Reserved | | | | | | | | | |
| 0x081A | NVMADR | 7:0 | NVMADR[7:0] | | | | | | | |
| | | 15:8 | | NVMADR[14:8] | | | | | | |
| 0x081C | NVMDAT | 7:0 | NVMDAT[7:0] | | | | | | | |
| | | 15:8 | | | NVMDAT[13:8] | | | | | |
| 0x081E | NVMCON1 | 7:0 | | NVMREGS | LWLO | FREE | WRERR | WREN | WR | RD |
| 0x081F | NVMCON2 | 7:0 | NVMCON2[7:0] | | | | | | | |
| 0x0820 ... 0x088D | Reserved | | | | | | | | | |
| 0x088E | OSCCON | 7:0 | | | COSC[1:0] | | | | | |
| 0x088F | Reserved | | | | | | | | | |
| 0x0890 | OSCSTAT | 7:0 | | HFOR | MFOR | LFOR | | ADOR | SFOR | |
| 0x0891 | OSCEN | 7:0 | | HFOEN | MFOEN | LFOEN | | ADOEN | | |
| 0x0892 | OSCTUNE | 7:0 | | | TUN[5:0] | | | | | |
| 0x0893 | OSCFRQ | 7:0 | | | | | | FRQ[2:0] | | |
| 0x0894 ... 0x090B | Reserved | | | | | | | | | |
| 0x090C | FVRCON | 7:0 | FVREN | FVRRDY | | | | | ADFVR[1:0] | |
| 0x090D ... 0x1E8F | Reserved | | | | | | | | | |
| 0x1E90 | INTPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x1E91 | T0CKIPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x1E92 | T1CKIPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x1E93 | T1GPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x1E94 ... 0x1E9B | Reserved | | | | | | | | | |
| 0x1E9C | T2INPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x1E9D ... 0x1EA0 | Reserved | | | | | | | | | |
| 0x1EA1 | CCP1PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x1EA2 | CCP2PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x1EA3 ... 0x1EC2 | Reserved | | | | | | | | | |
| 0x1EC3 | ADACTPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x1EC4 | Reserved | | | | | | | | | |
| 0x1EC5 | SSP1CLKPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x1EC6 | SSP1DATPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x1EC7 | SSP1SSPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1EC8 ... 0x1ECA | Reserved | | | | | | | | | |
| 0x1ECB | RX1PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1ECC | TX1PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x1ECD ... 0x1F0F | Reserved | | | | | | | | | |
| 0x1F10 | RA0PPS | 7:0 | | | RA0PPS[5:0] | | | | | |
| 0x1F11 | RA1PPS | 7:0 | | | RA1PPS[5:0] | | | | | |
| 0x1F12 | RA2PPS | 7:0 | | | RA2PPS[5:0] | | | | | |
| 0x1F13 | Reserved | | | | | | | | | |
| 0x1F14 | RA4PPS | 7:0 | | | RA4PPS[5:0] | | | | | |
| 0x1F15 | RA5PPS | 7:0 | | | RA5PPS[5:0] | | | | | |
| 0x1F16 ... 0x1F1B | Reserved | | | | | | | | | |
| 0x1F1C | RB4PPS | 7:0 | | | RB4PPS[5:0] | | | | | |
| 0x1F1D | RB5PPS | 7:0 | | | RB5PPS[5:0] | | | | | |
| 0x1F1E | RB6PPS | 7:0 | | | RB6PPS[5:0] | | | | | |
| 0x1F1F | RB7PPS | 7:0 | | | RB7PPS[5:0] | | | | | |
| 0x1F20 | RC0PPS | 7:0 | | | RC0PPS[5:0] | | | | | |
| 0x1F21 | RC1PPS | 7:0 | | | RC1PPS[5:0] | | | | | |
| 0x1F22 | RC2PPS | 7:0 | | | RC2PPS[5:0] | | | | | |
| 0x1F23 | RC3PPS | 7:0 | | | RC3PPS[5:0] | | | | | |
| 0x1F24 | RC4PPS | 7:0 | | | RC4PPS[5:0] | | | | | |
| 0x1F25 | RC5PPS | 7:0 | | | RC5PPS[5:0] | | | | | |
| 0x1F26 | RC6PPS | 7:0 | | | RC6PPS[5:0] | | | | | |
| 0x1F27 | RC7PPS | 7:0 | | | RC7PPS[5:0] | | | | | |
| 0x1F28 ... 0x1F37 | Reserved | | | | | | | | | |
| 0x1F38 | ANSELA | 7:0 | | | ANSELA5 | ANSELA4 | | ANSELA2 | ANSELA1 | ANSELA0 |
| 0x1F39 | WPUA | 7:0 | | | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 |
| 0x1F3A | ODCONA | 7:0 | | | ODCA5 | ODCA4 | | ODCA2 | ODCA1 | ODCA0 |
| 0x1F3B | SLRCONA | 7:0 | | | SLRA5 | SLRA4 | | SLRA2 | SLRA1 | SLRA0 |
| 0x1F3C | INLVLA | 7:0 | | | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 |
| 0x1F3D | IOCAP | 7:0 | | | IOCAP5 | IOCAP4 | IOCAP3 | IOCAP2 | IOCAP1 | IOCAP0 |
| 0x1F3E | IOCAN | 7:0 | | | IOCAN5 | IOCAN4 | IOCAN3 | IOCAN2 | IOCAN1 | IOCAN0 |
| 0x1F3F | IOCAF | 7:0 | | | IOCAF5 | IOCAF4 | IOCAF3 | IOCAF2 | IOCAF1 | IOCAF0 |
| 0x1F40 ... 0x1F42 | Reserved | | | | | | | | | |
| 0x1F43 | ANSELB | 7:0 | ANSELB7 | ANSELB6 | ANSELB5 | ANSELB4 | | | | |
| 0x1F44 | WPUB | 7:0 | WPUB7 | WPUB6 | WPUB5 | WPUB4 | | | | |
| 0x1F45 | ODCONB | 7:0 | ODCB7 | ODCB6 | ODCB5 | ODCB4 | | | | |
| 0x1F46 | SLRCONB | 7:0 | SLRB7 | SLRB6 | SLRB5 | SLRB4 | | | | |
| 0x1F47 | INLVLB | 7:0 | INLVLB7 | INLVLB6 | INLVLB5 | INLVLB4 | | | | |
| 0x1F48 | IOCBP | 7:0 | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | | | | |
| 0x1F49 | IOCBN | 7:0 | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | | | | |
| 0x1F4A | IOCBF | 7:0 | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | | | | |
| 0x1F4B ... 0x1F4D | Reserved | | | | | | | | | |
| 0x1F4E | ANSELC | 7:0 | ANSELC7 | ANSELC6 | ANSELC5 | ANSELC4 | ANSELC3 | ANSELC2 | ANSELC1 | ANSELC0 |
| 0x1F4F | WPUC | 7:0 | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 |
| 0x1F50 | ODCONC | 7:0 | ODCC7 | ODCC6 | ODCC5 | ODCC4 | ODCC3 | ODCC2 | ODCC1 | ODCC0 |
| 0x1F51 | SLRCONC | 7:0 | SLRC7 | SLRC6 | SLRC5 | SLRC4 | SLRC3 | SLRC2 | SLRC1 | SLRC0 |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x1F52 | INLVLC | 7:0 | INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 |
| 0x1F53 | IOCCP | 7:0 | IOCCP7 | IOCCP6 | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 |
| 0x1F54 | IOCCN | 7:0 | IOCCN7 | IOCCN6 | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 |
| 0x1F55 | IOCCF | 7:0 | IOCCF7 | IOCCF6 | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 |
| 0x1F56 ... 0x8004 | Reserved | | | | | | | | | |
| 80058005 | REVISIONID | 7:0 | MJRREV[1:0] | | MNRREV[5:0] | | | | | |
| | | 15:8 | | | Reserved | Reserved | MJRREV[5:2] | | | |
| 80068006 | DEVICEID | 7:0 | DEV[7:0] | | | | | | | |
| | | 15:8 | | | Reserved | Reserved | DEV[11:8] | | | |
| 0x8007 | CONFIG1 | 7:0 | | | RSTOSC[1:0] | | | | FEXTOSC[1:0] | |
| | | 15:8 | | | | VDDAR | | | | $\overline{\text{CLKOUTEN}}$ |
| 0x8008 | CONFIG2 | 7:0 | BOREN[1:0] | | | WDTE[1:0] | | PWRTS[1:0] | | MCLRE |
| | | 15:8 | | | $\overline{\text{DEBUG}}$ | STVREN | PPS1WAY | | BORV | |
| 0x8009 | CONFIG3 | 7:0 | | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x800A | CONFIG4 | 7:0 | $\overline{\text{WRTAPP}}$ | | | $\overline{\text{SAFEN}}$ | $\overline{\text{BBEN}}$ | BBSIZE[2:0] | | |
| | | 15:8 | | | LVP | | $\overline{\text{WRTSAF}}$ | | $\overline{\text{WRTC}}$ | $\overline{\text{WRTB}}$ |
| 0x800B | CONFIG5 | 7:0 | | | | | | | | $\overline{\text{CP}}$ |
| | | 15:8 | | | | | | | | |

# 32. Electrical Specifications

## 32.1 Absolute Maximum Ratings(†)

| Parameter | | Rating |
|---|---|---|
| **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 +6.5V |
| • **on $\overline{MCLR}$ pin:** | | -0.3V to +9.0V |
| • **on all other pins:** | | -0.3V to ($V_{DD}$ + 0.3V) |
| **Maximum current** | | |
| • **on $V_{SS}$ pin(1)** | -40°C ≤ $T_A$ ≤ +85°C | 300 mA |
| | 85°C < $T_A$ ≤ +125°C | 120 mA |
| • **on $V_{DD}$ pin(1)** | -40°C ≤ $T_A$ ≤ +85°C | 250 mA |
| | 85°C < $T_A$ ≤ +125°C | 85 mA |
| • **on any standard I/O pin** | | ±25 mA |
| **Clamp current, $I_K$ ($V_{PIN}$ < 0 or $V_{PIN}$ > $V_{DD}$)** | | ±20 mA |
| **Total power dissipation(2)** | | 800 mW |

**Notes:**

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 "Thermal Characteristics" to calculate device specifications.
2. Power dissipation is calculated as follows:
   $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma (V_{OI} \times I_{OL})$
3. Internal Power Dissipation is calculated as follows:
   $P_{INTERNAL} = I_{DD} \times V_{DD}$

   where $I_{DD}$ is current to run the chip alone without driving any load on the output pins.
4. I/O Power Dissipation is calculated as follows:
   $P_{I/O} = \Sigma(I_{OL} * V_{OL}) + \Sigma(I_{OH} * (V_{DD} - V_{OH}))$
5. Derated Power is calculated as follows:
   $P_{DER} = PD_{MAX}(T_J - T_A)/\theta_{JA}$

   where $T_A$=Ambient Temperature, $T_J$ = Junction Temperature.

† 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.

## 32.2 Standard Operating Conditions

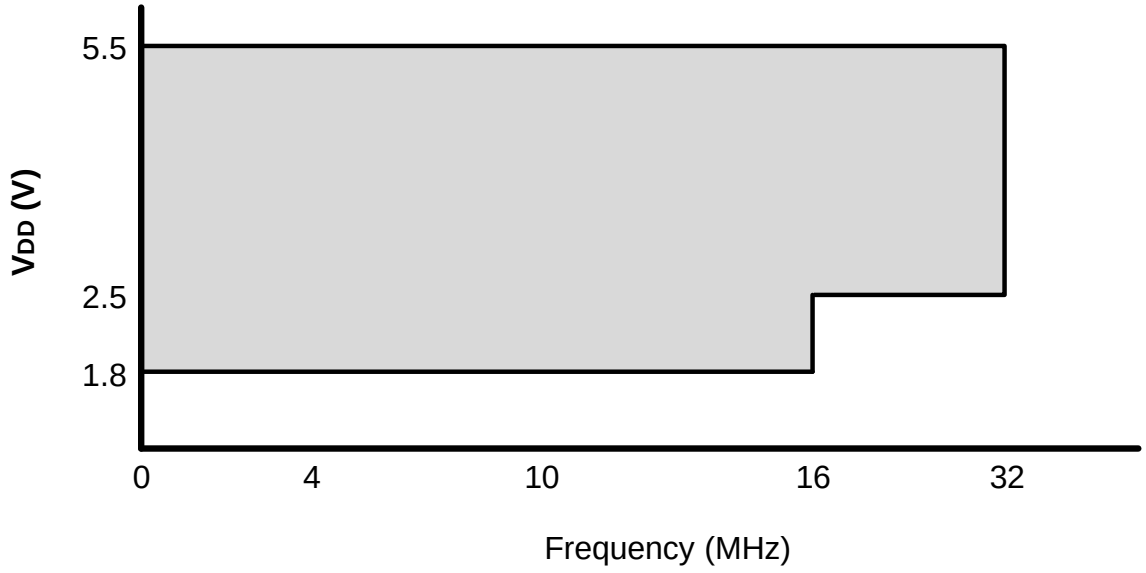
The standard operating conditions for any device are defined as:

| Parameter | Condition |
|---|---|
| Operating Voltage: | $V_{DDMIN} \le V_{DD} \le V_{DDMAX}$ |
| Operating Temperature: | $T_{AMIN} \le T_A \le T_{AMAX}$ |

| Parameter | | Ratings |
|---|---|---|
| **$V_{DD}$ — Operating Supply Voltage(1)** | | |
| | $V_{DDMIN}$ ($F_{OSC}$ ≤ 16 MHz) | +1.8V |
| | $V_{DDMIN}$ ($F_{OSC}$ ≤ 32 MHz) | +2.5V |
| | $V_{DDMAX}$ | +5.5V |
| **$T_A$ — Operating Ambient Temperature Range** | | |
| Industrial Temperature | $T_{A_MIN}$ | -40°C |
| | $T_{A_MAX}$ | +85°C |
| Extended Temperature | $T_{A_MIN}$ | -40°C |
| | $T_{A_MAX}$ | +125°C |

**Note:**
1. See Parameter **D002**, DC Characteristics: Supply Voltage.

**Figure 32-1. Voltage Frequency Graph, -40°C ≤ $T_A$ ≤ +125°C**




**Notes:**
1. The shaded region indicates the permissible combinations of voltage and frequency.
2. Refer to "External Clock/Oscillator Timing Requirements" section for each Oscillator mode's supported frequencies.

## 32.3 DC Characteristics

### 32.3.1 Supply Voltage

**Table 32-1.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | Min. | Typ.† | Max. | Units | Conditions |
| **Supply Voltage** | | | | | | | |

.....continued

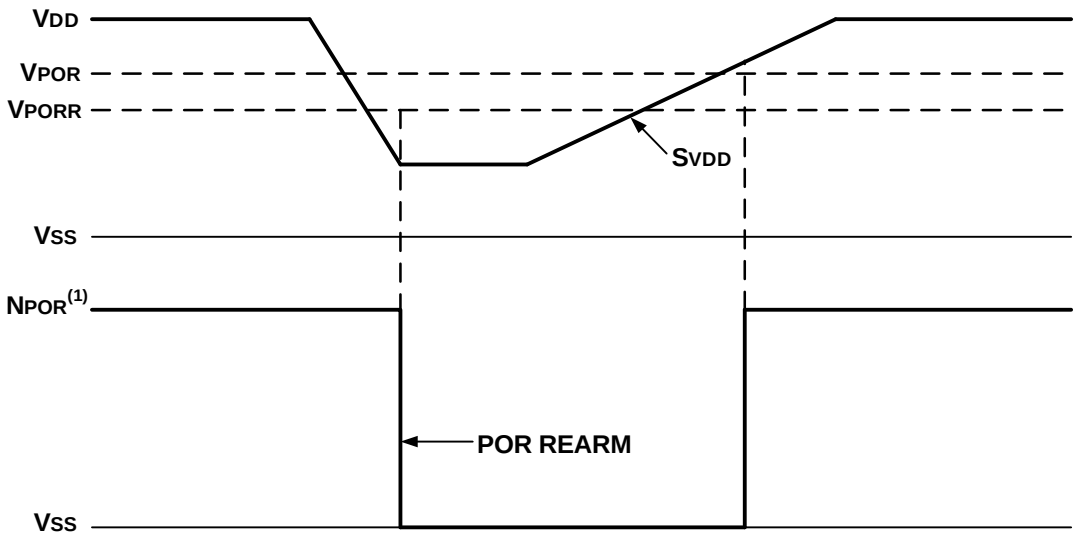
| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | Min. | Typ.† | Max. | Units | Conditions |
| D002 | $V_{DD}$ | | 1.8 | — | 5.5 | V | $F_{OSC} \leq 16$ MHz |
| | | | 2.5 | — | 5.5 | V | $F_{OSC} > 16$ MHz |
| **RAM Data Retention(1)** | | | | | | | |
| D003 | $V_{DR}$ | | 1.7 | — | — | V | Device in Sleep mode |
| **Power-on Reset Release Voltage(2)** | | | | | | | |
| D004 | $V_{POR}$ | | — | 1.6 | — | V | BOR disabled(3) |
| **Power-on Reset Rearm Voltage(2)** | | | | | | | |
| D005 | $V_{PORR}$ | | — | 0.8 | — | V | BOR disabled(3) |
| **$V_{DD}$ Rise Rate to ensure internal Power-on Reset signal(2)** | | | | | | | |
| D006 | $S_{VDD}$ | | 0.05 | — | — | V/ms | BOR disabled(3) |

† - Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. This is the limit to which $V_{DD}$ can be lowered in Sleep mode without losing RAM data.
2. See the following figure, POR and POR REARM with Slow Rising $V_{DD}$.
3. See "Reset, WDT, Power-up Timer, and Brown-Out Reset Specifications" for BOR trip point information.

**Figure 32-2. POR and POR Rearm with Slow Rising $V_{DD}$**




**Note:**

1. When $N_{POR}$ is low, the device is held in Reset.

### 32.3.2 Supply Current ($I_{DD}$)(1,2)

**Table 32-2.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Device Characteristics | Min. | Typ.† | Max. | Units | Conditions | |
| | | | | | | | $V_{DD}$ | Note |
| D101 | $I_{DD_{HFO16}}$ | HFINTOSC = 16 MHz | — | 1.5 | 2.1 | mA | 3.0V | |
| D102 | $I_{DD_{HFOPLL}}$ | HFINTOSC = 32 MHz | — | 2.7 | 3.6 | mA | 3.0V | |

† - Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. The test conditions for all $I_{DD}$ measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are outputs driven low; $\overline{MCLR}$ = $V_{DD}$; 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.

### 32.3.3 Power-Down Current ($I_{PD}$)(1,2,3)

**Table 32-3.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Device Characteristics | Min. | Typ.† | Max. +85°C | Max. +125°C | Units | Conditions | |
| | | | | | | | | $V_{DD}$ | Note |
| D200 | $I_{PD}$ | $I_{PD}$ Base | — | 0.40 | 2.5 | 12 | µA | 3.0V | |
| D201 | $I_{PD_WDT}$ | Low-Frequency Internal Oscillator/WDT | — | 0.5 | 3.6 | 6.1 | µA | 3.0V | |
| D203 | $I_{PD_FVR}$ | FVR | — | 40 | 64 | 66 | µA | 3.0V | |
| D204 | $I_{PD_BOR}$ | Brown-out Reset (BOR) | — | 25 | 33 | 37 | µA | 3.0V | |
| D207 | $I_{PD_ADCA}$ | ADC - Active | — | — | 3 | 5.6 | µA | 3.0V | ADC is non-converting (4) |

† - Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. The peripheral current is the sum of the base $I_{DD}$ 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 $V_{SS}$.
3. All peripheral currents listed are on a per-peripheral basis if more than one instance of a peripheral is available.
4. ADC clock source is ADCRC.

### 32.3.4 I/O Ports

**Table 32-4.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Device Characteristics** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| **Input Low Voltage** | | | | | | | |
| | $V_{IL}$ | I/O PORT: | | | | | |
| D300 | | • with TTL buffer | — | — | 0.8 | V | $4.5V \le V_{DD} \le 5.5V$ |
| D301 | | | — | — | 0.15 $V_{DD}$ | V | $1.8V \le V_{DD} \le 4.5V$ |
| D302 | | • with Schmitt Trigger buffer | — | — | 0.2 $V_{DD}$ | V | $1.8V \le V_{DD} \le 5.5V$ |
| D303 | | • with $I^2C$ levels | — | — | 0.3 $V_{DD}$ | V | |
| D304 | | • with SMBus levels | — | — | 0.8 | V | $2.7V \le V_{DD} \le 5.5V$ |
| D305 | | $\overline{MCLR}$ | — | — | 0.2 $V_{DD}$ | V | |
| **Input High Voltage** | | | | | | | |
| | $V_{IH}$ | I/O PORT: | | | | | |
| D320 | | • with TTL buffer | 2.0 | — | — | V | $4.5V \le V_{DD} \le 5.5V$ |
| D321 | | | 0.25 $V_{DD}$+0.8 | — | — | V | $1.8V \le V_{DD} \le 4.5V$ |
| D322 | | • with Schmitt Trigger buffer | 0.8$V_{DD}$ | — | — | V | $1.8V \le V_{DD} \le 5.5V$ |
| D323 | | • with $I^2C$ levels | 0.7 $V_{DD}$ | — | — | V | |
| D324 | | • with SMBus levels | 2.1 | — | — | V | $2.7V \le V_{DD} \le 5.5V$ |
| D325 | | $\overline{MCLR}$ | 0.8 $V_{DD}$ | — | — | V | |
| **Input Leakage Current(1)** | | | | | | | |
| D340 | $I_{IL}$ | I/O PORTS | — | ±5 | ±125 | nA | $V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 85°C |
| D341 | | | — | ±5 | ±1000 | nA | $V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 125°C |
| D342 | | $\overline{MCLR}$(2) | — | ±50 | ±200 | nA | $V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 85°C |
| **Weak Pull-up Current** | | | | | | | |
| D350 | $I_{PUR}$ | | 25 | 120 | 200 | µA | $V_{DD}$ = 3.0V, $V_{PIN} = V_{SS}$ |
| **Output Low Voltage** | | | | | | | |
| D360 | $V_{OL}$ | Standard I/O PORTS | — | — | 0.6 | V | $I_{OL}$ = 10 mA, $V_{DD}$ = 3.0V |
| **Output High Voltage** | | | | | | | |
| D370 | $V_{OH}$ | Standard I/O PORTS | $V_{DD}$-0.7 | — | — | V | $I_{OH}$ = 6 mA, $V_{DD}$ = 3.0V |
| **All I/O Pins** | | | | | | | |
| D380 | $C_{IO}$ | | — | 5 | 50 | pF | |

| .....continued | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
| **Param. No.** | **Sym.** | **Device Characteristics** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| † - Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested. | | | | | | | |
| **Notes:** | | | | | | | |
| 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. | | | | | | | |

### 32.3.5 Memory Programming Specifications

**Table 32-5.**

| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Device Characteristics** | **Min.** | **Typ†** | **Max.** | **Units** | **Conditions** |
| **High Voltage Entry Programming Mode Specifications** | | | | | | | |
| MEM01 | $V_{IHH}$ | Voltage on $\overline{MCLR}/V_{PP}$ pin to enter programming mode | 7.9 | — | 9 | V | **(Note 2)** |
| MEM02 | $I_{PPGM}$ | Current on $\overline{MCLR}/V_{PP}$ pin during programming mode | — | 1 | — | mA | **(Note 2)** |
| **Programming Mode Specifications** | | | | | | | |
| MEM10 | $V_{BE}$ | $V_{DD}$ for Bulk Erase | — | 2.7 | — | V | **(Note 3)** |
| MEM11 | $I_{DDPGM}$ | Supply Current during Programming operation | — | — | 10 | mA | |
| **Program Flash Memory Specifications** | | | | | | | |
| MEM30 | $E_P$ | Flash Memory Cell Endurance | 10k | — | — | E/W | -40°C ≤ $T_A$ ≤ +85°C **(Note 1)** |
| MEM32 | $T_{P_RET}$ | Characteristic Retention | — | 40 | — | Year | Provided no other specifications are violated |
| MEM33 | $V_{P_RD}$ | $V_{DD}$ for Read operation | $V_{DDMIN}$ | — | $V_{DDMAX}$ | V | |
| MEM34 | $V_{P_REW}$ | $V_{DD}$ for Row Erase or Write operation | $V_{DDMIN}$ | — | $V_{DDMAX}$ | V | |
| MEM35 | $T_{P_REW}$ | Self-Timed Row Erase or Self-Timed Write | — | 3.0 | — | ms | |
| † - Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested. | | | | | | | |
| **Notes:** | | | | | | | |
| 1. Flash Memory Cell Endurance for the Flash memory is defined as: One Row Erase operation and one Self-Timed Write. | | | | | | | |
| 2. Required only if the LVP bit of the Configuration Words is disabled. | | | | | | | |
| 3. Refer to the **"Family Programming Specification"** for more information. | | | | | | | |

### 32.3.6 Thermal Characteristics

**Table 32-6.**

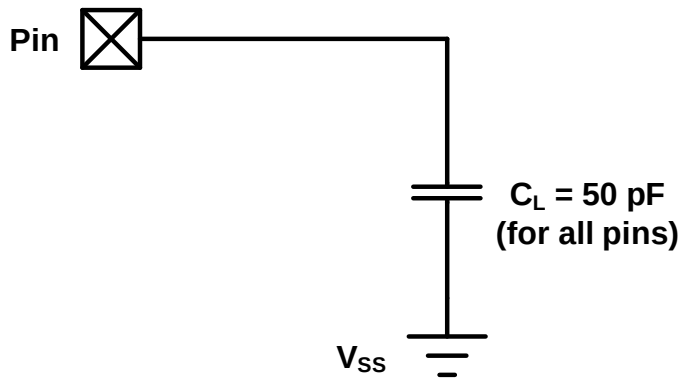
| Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ $T_A$ ≤ +125°C | | | | | |
|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Typ.** | **Units** | **Conditions** |
| TH01 | $\theta_{JA}$ | Thermal Resistance Junction to Ambient | — | °C/W | 8-pin SOIC package |
| | | | — | °C/W | 8-pin DFN package |
| | | | — | °C/W | 14-pin SOIC package |
| | | | — | °C/W | 14-pin TSSOP package |
| | | | — | °C/W | 16-pin VQFN 3x3mm package |
| | | | — | °C/W | 20-pin PDIP package |
| | | | — | °C/W | 20-pin SSOP package |
| | | | — | °C/W | 20-pin SOIC package |
| | | | — | °C/W | 20-pin VQFN 3x3mm package |
| TH03 | $T_{JMAX}$ | Maximum Junction Temperature | 150 | °C | |
| TH04 | PD | Power Dissipation | — | W | PD = $P_{INTERNAL}$+$P_{I/O}$ |
| TH05 | $P_{INTERNAL}$ | Internal Power Dissipation | — | W | $P_{INTERNAL}$ = $I_{DD}$ x $V_{DD}$(1) |
| TH06 | $P_{I/O}$ | I/O Power Dissipation | — | W | $P_{I/O}$ = Σ($I_{OL}$*$V_{OL}$)+Σ($I_{OH}$*($V_{DD}$-$V_{OH}$)) |
| TH07 | $P_{DER}$ | Derated Power | — | W | $P_{DER}$ = $PD_{MAX}$ ($T_J$-$T_A$)/$\theta_{JA}$(2) |

**Notes:**

1. $I_{DD}$ is current to run the chip alone without driving any load on the output pins.
2. $T_A$ = Ambient Temperature, $T_J$ = Junction Temperature.

## 32.4 AC Characteristics

**Figure 32-3. Load Conditions**



### 32.4.1 External Clock/Oscillator Timing Requirements

**Figure 32-4. Clock Timing**

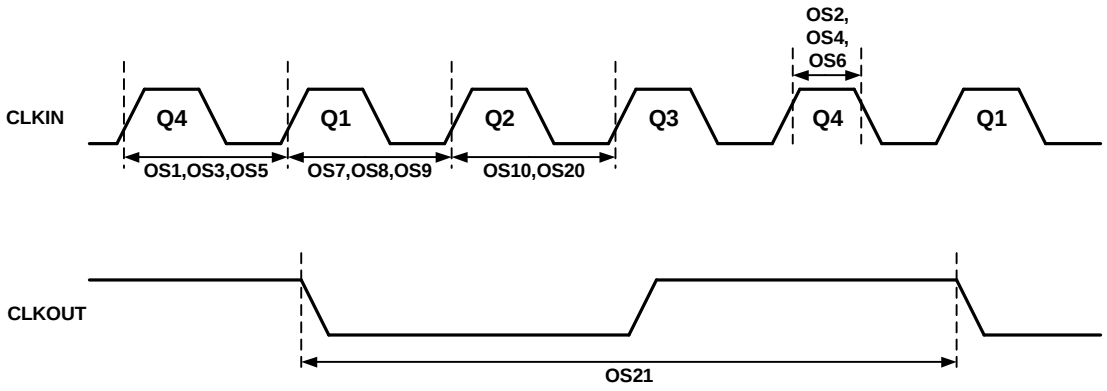



**Table 32-7.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| **ECL Oscillator** | | | | | | | |
| OS1 | $F_{ECL}$ | Clock Frequency | — | — | 16 | MHz | |
| OS2 | $T_{ECL_DC}$ | Clock Duty Cycle | 40 | — | 60 | % | |
| **ECH Oscillator** | | | | | | | |
| OS5 | $F_{ECH}$ | Clock Frequency | — | — | 32 | MHz | |
| OS6 | $T_{ECH_DC}$ | Clock Duty Cycle | 40 | — | 60 | % | |
| **System Oscillator** | | | | | | | |
| OS20 | $F_{OSC}$ | System Clock Frequency | — | — | 32 | MHz | **Note 2, Note 3** |
| OS21 | $F_{CY}$ | Instruction Frequency | — | $F_{OSC}/4$ | — | MHz | |
| OS22 | $T_{CY}$ | Instruction Period | 125 | $1/F_{CY}$ | — | ns | **Note 1** |

.....continued

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |

* 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.

**Notes:**

1. Instruction cycle period ($T_{CY}$) 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.
2. The system clock frequency ($F_{OSC}$) is selected by the "main clock switch controls" as described in the **"OSC - Oscillator Module"** chapter.
3. The system clock frequency ($F_{OSC}$) must meet the voltage requirements defined in the "Standard Operating Conditions" section.

### 32.4.2 Internal Oscillator Parameters(1)

**Table 32-8.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| OS50 | $F_{HFOSC}$ | Precision Calibrated HFINTOSC Frequency | — | 4<br>8<br>16<br>32 | — | MHz | **Note 2** |
| OS51 | $F_{HFOSCLP}$ | Low-Power Optimized HFINTOSC Frequency | 0.92<br>1.84<br>0.88<br>1.76 | 1<br>2<br>1<br>2 | 1.08<br>2.16<br>1.12<br>2.24 | MHz<br>MHz<br>MHz<br>MHz | -40°C to 85°C<br>-40°C to 85°C<br>-40°C to 125 °C<br>-40°C to 125°C |
| OS52 | $F_{MFOSC}$ | Internal Calibrated MFINTOSC Frequency | — | 500 | — | kHz | |
| OS53 | $F_{LFOSC}$ | Internal LFINTOSC Frequency | 26.5 | 31 | 36 | kHz | |
| OS54 | $T_{HFOSCST}$ | HFINTOSC Wake-up from Sleep Start-up Time | — | 4 | 10 | µs | |
| OS56 | $T_{LFOSCST}$ | LFINTOSC Wake-up from Sleep Start-up Time | — | 0.2 | 1 | ms | |

.....continued

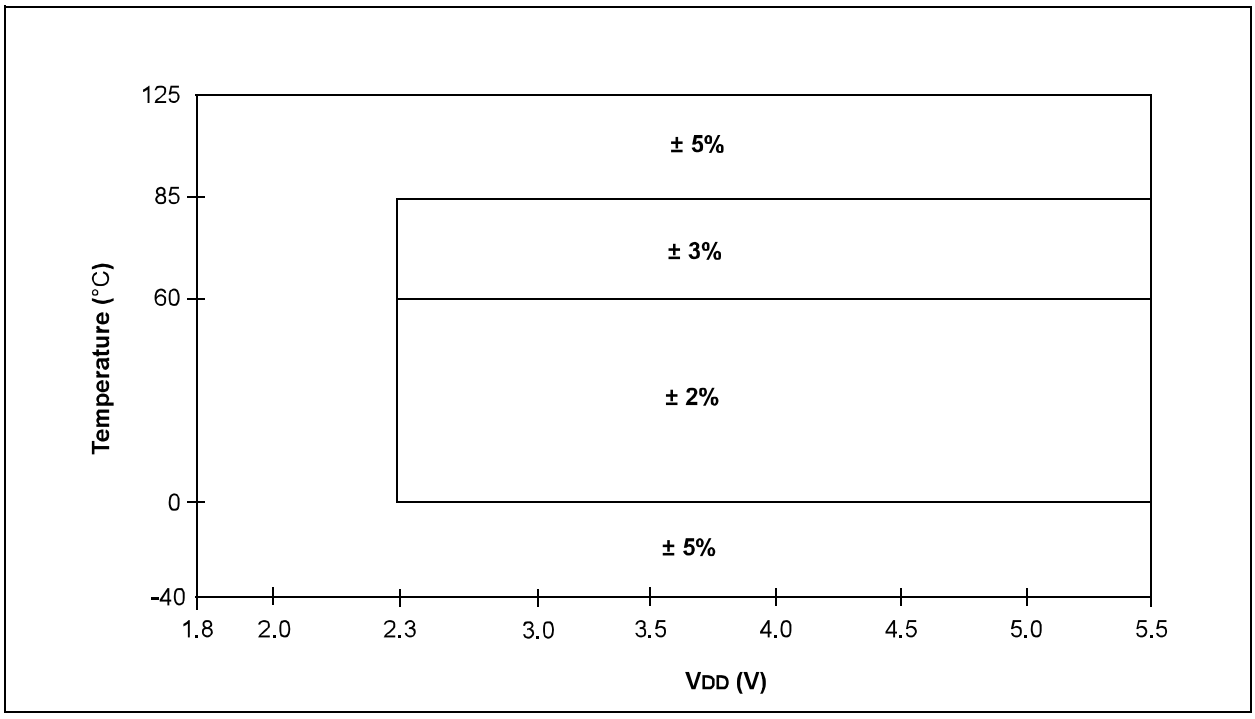
| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |

† - Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. To ensure these oscillator frequency tolerances, $V_{DD}$ and $V_{SS}$ must be capacitively decoupled as close to the device as possible. 0.1 µF and 0.01 µF values in parallel are recommended.
2. See Figure 32-5.

**Figure 32-5. Precision Calibrated HFINTOSC Frequency Accuracy Over Device $V_{DD}$ and Temperature**



### 32.4.3 I/O and CLKOUT Timing Specifications

**Figure 32-6. CLKOUT and I/O Timing**

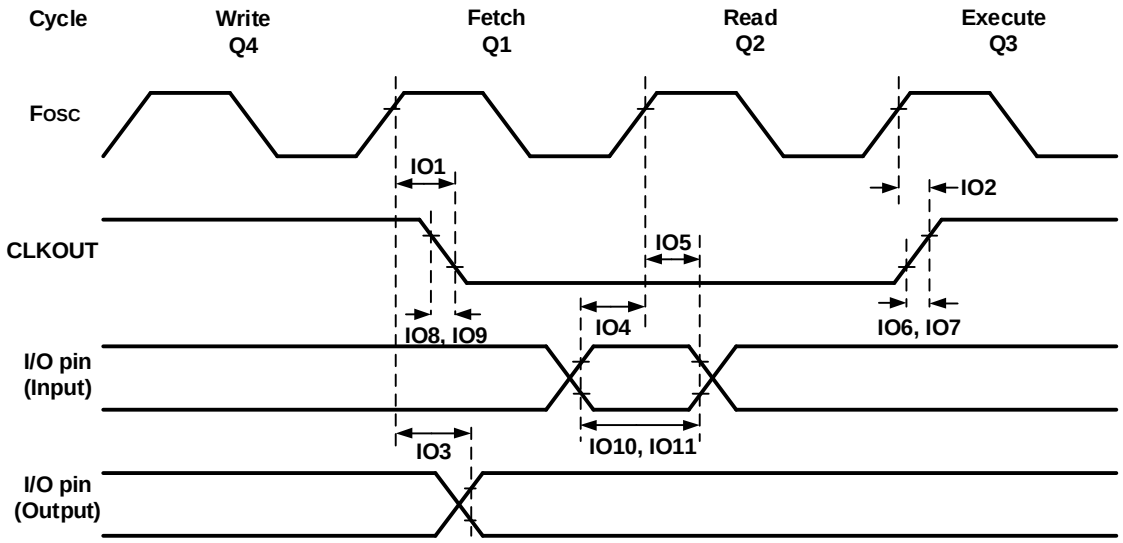
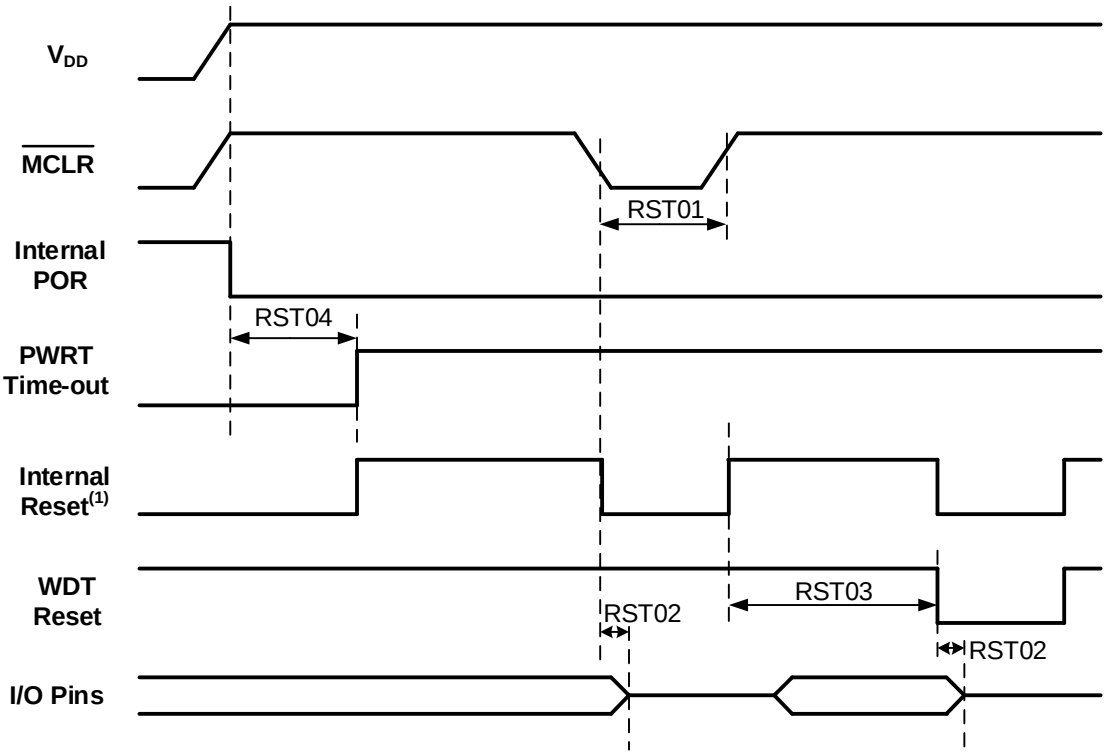



**Table 32-9. I/O and CLKOUT Timing Specifications**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| IO1* | $T_{CLKOUTH}$ | CLKOUT rising edge delay (rising edge $F_{OSC}$ (Q1 cycle) to falling edge CLKOUT | — | — | 70 | ns | |
| IO2* | $T_{CLKOUTL}$ | CLKOUT falling edge delay (rising edge $F_{OSC}$ (Q3 cycle) to rising edge CLKOUT | — | — | 72 | ns | |
| IO3* | $T_{IO_VALID}$ | Port output valid time (rising edge $F_{OSC}$ (Q1 cycle) to port valid) | — | 50 | 70 | ns | |
| IO4* | $T_{IO_SETUP}$ | Port input setup time (Setup time before rising edge $F_{OSC}$ – Q2 cycle) | 20 | — | — | ns | |
| IO5* | $T_{IO_HOLD}$ | Port input hold time (Hold time after rising edge $F_{OSC}$ – Q2 cycle) | 50 | — | — | ns | |
| IO6* | $T_{IOR_SLREN}$ | Port I/O rise time, slew rate enabled | — | 25 | — | ns | $V_{DD}$ = 3.0V |
| IO7* | $T_{IOR_SLRDIS}$ | Port I/O rise time, slew rate disabled | — | 5 | — | ns | $V_{DD}$ = 3.0V |
| IO8* | $T_{IOF_SLREN}$ | Port I/O fall time, slew rate enabled | — | 25 | — | ns | $V_{DD}$ = 3.0V |
| IO9* | $T_{IOF_SLRDIS}$ | Port I/O fall time, slew rate disabled | — | 5 | — | ns | $V_{DD}$ = 3.0V |
| IO10* | $T_{INT}$ | INT pin high or low time to trigger an interrupt | 25 | — | — | ns | |
| IO11* | $T_{IOC}$ | Interrupt-on-Change minimum high or low time to trigger interrupt | 25 | — | — | ns | |
| * - These parameters are characterized but not tested. | | | | | | | |

### 32.4.4 Reset, WDT, Power-up Timer, and Brown-Out Reset Specifications

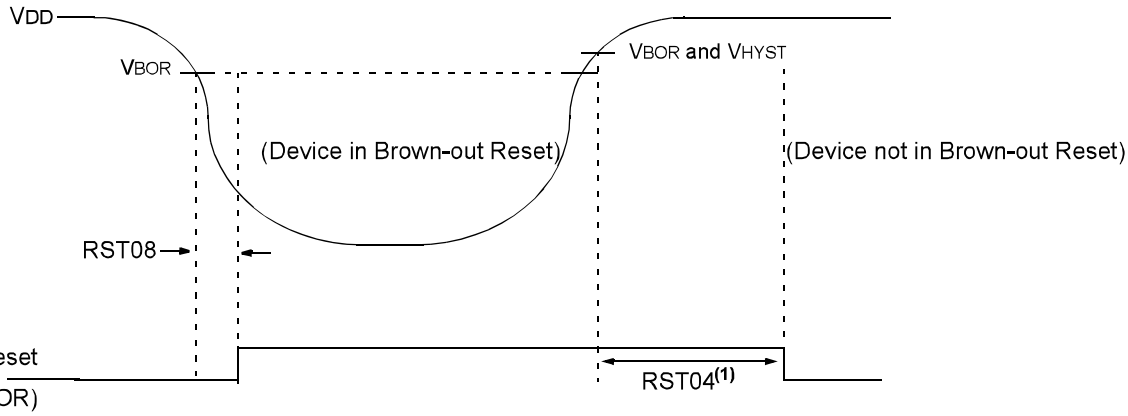
**Figure 32-7. Reset, Watchdog Timer, and Power-up Timer Timing**




**Note:**

1. Asserted low.

**Figure 32-8. Brown-out Reset Timing and Characteristics**




**Note:**

1. Only if $\overline{\text{PWRTE}}$ bit in the Configuration Word register is programmed to '1'; 2 ms delay if $\overline{\text{PWRTE}}$ = 0.

**Table 32-10.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| RST01* | $T_{MCLR}$ | $\overline{MCLR}$ Pulse Width Low to ensure Reset | 2 | — | — | µs | |
| RST02* | $T_{IOZ}$ | I/O high-impedance from Reset detection | — | — | 2 | µs | |
| RST03 | $T_{WDT}$ | Watchdog Timer Time-out Period | — | 16 | — | ms | 1:512 Prescaler |
| RST04* | $T_{PWRT}$ | Power-up Timer Period | — | 65 | — | ms | |
| RST06 | $V_{BOR}$ | Brown-out Reset Voltage | 2.55<br>— | 2.7<br>— | 2.85<br>1.9(1) | V<br>V | BORV = 0<br>BORV = 1 |
| RST07 | $V_{BORHYS}$ | Brown-out Reset Hysteresis | — | 40 | — | mV | |
| RST08 | $T_{BORDC}$ | Brown-out Reset Response Time | — | 3 | — | 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. This value corresponds to $V_{BORMAX}$.

### 32.4.5 Analog-to-Digital Converter (ADC) Accuracy Specifications(1,2)

**Table 32-11.**

| Standard Operating Conditions (unless otherwise stated)<br>$V_{DD}$ = 3.0V, $T_A$ = 25°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| AD01 | $N_R$ | Resolution | — | — | 10 | bit | |
| AD02 | $E_{IL}$ | Integral Error | — | — | 2 | LSb | $ADC_{REF}$+ = 3.0V, $ADC_{REF}$- = $V_{SS}$ |
| AD03 | $E_{DL}$ | Differential Error | — | — | 1 | LSb | $ADC_{REF}$+ = 3.0V, $ADC_{REF}$- = $V_{SS}$ |
| AD04 | $E_{OFF}$ | Offset Error | — | — | 2 | LSb | $ADC_{REF}$+ = 3.0V, $ADC_{REF}$- = $V_{SS}$ |
| AD05 | $E_{GN}$ | Gain Error | — | — | 1.5 | LSb | $ADC_{REF}$+ = 3.0V, $ADC_{REF}$- = $V_{SS}$ |
| AD06 | $V_{ADREF}$ | ADC Reference Voltage ($AD_{REF}$+) | 1.8 | — | $V_{DD}$ | V | |
| AD07 | $V_{AIN}$ | Full-Scale Range | $V_{SS}$ | — | $AD_{REF}$+ | V | |
| AD08 | $Z_{AIN}$ | Recommended Impedance of Analog Voltage Source | — | 10 | — | kΩ | |
| AD09 | $R_{VREF}$ | ADC Voltage Reference Ladder Impedance | — | 50 | — | kΩ | |

| .....continued | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Standard Operating Conditions (unless otherwise stated)** $V_{DD}$ = 3.0V, $T_A$ = 25°C | | | | | | | |
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| * - 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. | | | | | | | |
| **Notes:** 1. Total Absolute Error is the sum of the offset, gain and integral non-linearity (INL) errors. 2. The ADC conversion result never decreases with an increase in the input and has no missing codes. | | | | | | | |

### 32.4.6 Analog-to-Digital Converter (ADC) Conversion Timing Specifications

**Table 32-12.**

| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| AD20 | $T_{AD}$ | ADC Clock Period | 0.5 | — | 9 | µs | $F_{OSC}$ clock source |
| AD21 | | | 1 | 2 | 6 | µs | ADCRC clock source |
| AD22 | $T_{CNV}$ | Conversion Time | — | 11 | — | $T_{AD}$ | Set of GO bit to clear of GO bit |
| AD23 | $T_{ACQ}$ | Acquisition Time | — | 2 | — | µs | |
| AD24 | $T_{HCD}$ | Sample and Hold Capacitor Disconnect Time | — | — | — | — | $F_{OSC}$ clock source |
| * - 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 32-9. ADC Conversion Timing (ADC Clock $F_{OSC}$-Based)**

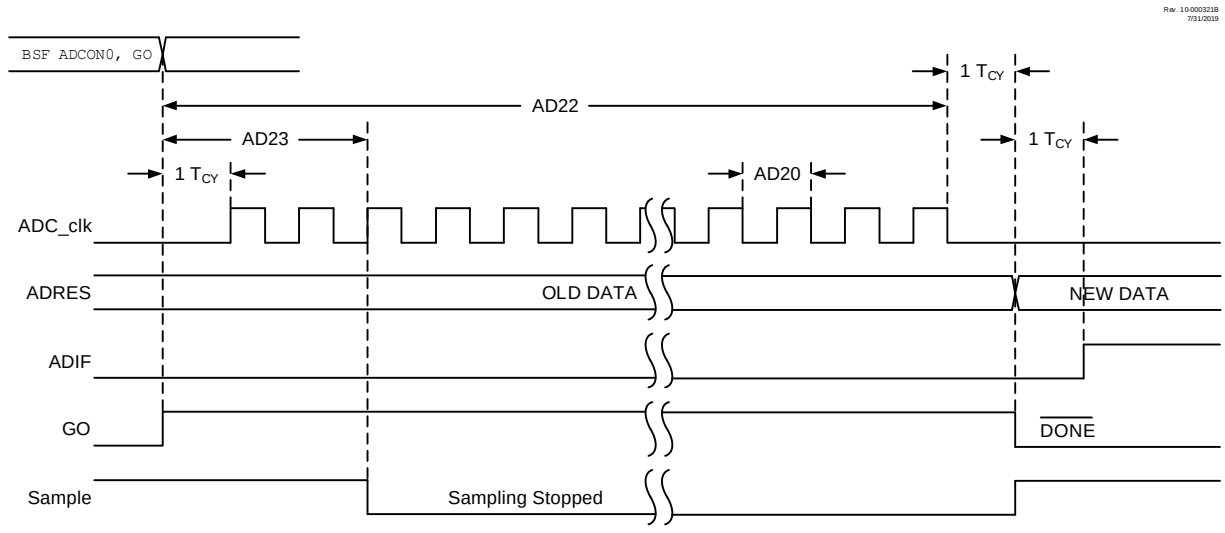
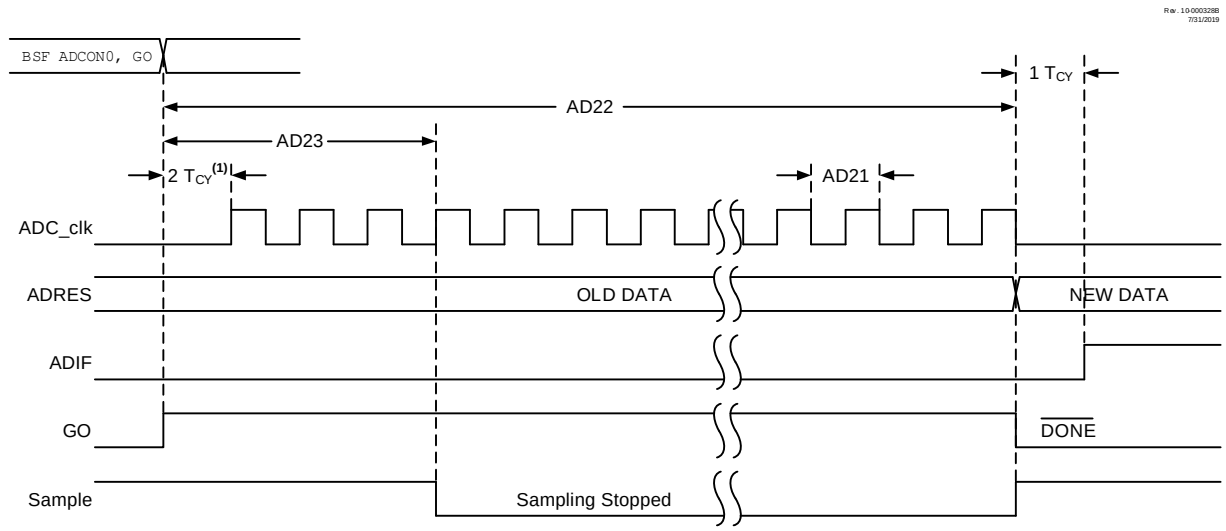


**Figure 32-10. ADC Conversion Timing (ADC Clock from $F_{RC}$)**




**Note:**

1. If the ADC clock source is selected as $F_{RC}$, a time of $T_{CY}$ is added before the ADC clock starts. This allows the `SLEEP` instruction to be executed.

### 32.4.7 Fixed Voltage Reference (FVR) Specifications

**Table 32-13.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| FVR01 | $V_{FVR}1$ | 1x Gain (1.024V) | -4 | — | +4 | % | $V_{DD} \geq 2.5V$, -40°C to 85°C |
| FVR02 | $V_{FVR}2$ | 2x Gain (2.048V) | -4 | — | +4 | % | $V_{DD} \geq 2.5V$, -40°C to 85°C |
| FVR03 | $V_{FVR}4$ | 4x Gain (4.096V) | -6 | — | +6 | % | $V_{DD} \geq 4.75V$, -40°C to 85°C |
| FVR04 | $T_{FVRST}$ | FVR Start-up Time | — | 60 | — | µs | |

### 32.4.8 Timer0 and Timer1 External Clock Requirements

**Table 32-14.**

| Standard Operating Conditions (unless otherwise stated)<br>Operating Temperature: -40°C ≤ $T_A$ ≤ +125°C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | | Min. | Typ. † | Max. | Units | Conditions |
| 40* | $T_T0H$ | T0CKI High Pulse Width | No Prescaler | $0.5T_{CY}$+20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 41* | $T_T0L$ | T0CKI Low Pulse Width | No Prescaler | $0.5T_{CY}$+20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 42* | $T_T0P$ | T0CKI Period | | — | — | — | ns | N = Prescale value |

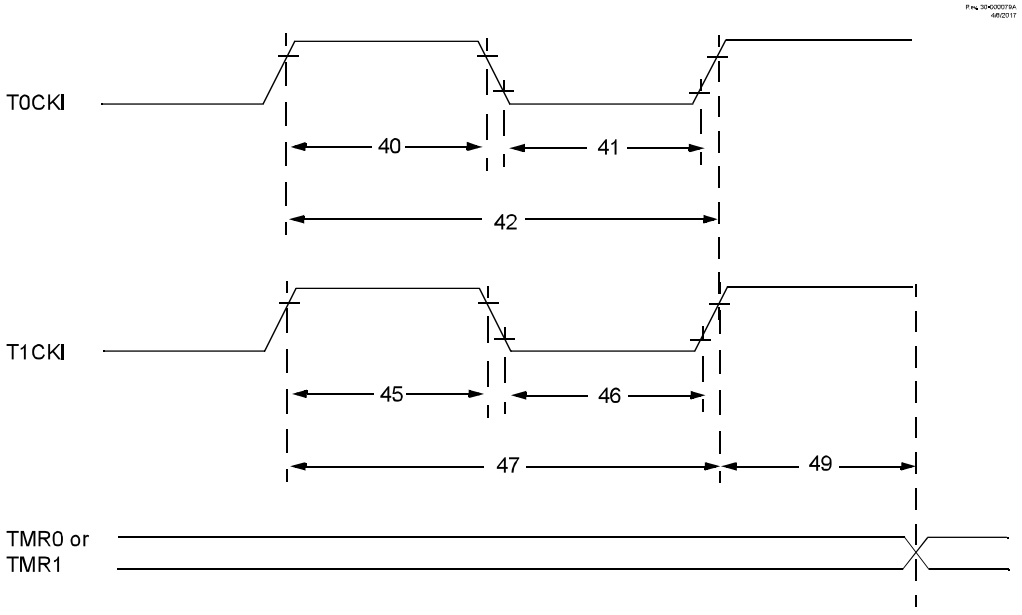
.....continued

| Standard Operating Conditions (unless otherwise stated)<br>Operating Temperature: -40°C ≤ $T_A$ ≤ +125°C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| 45* | $T_T1H$ | T1CKI High Time | Synchronous, No Prescaler | $0.5T_{CY}$+20 | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 46* | $T_T1L$ | T1CKI Low Time | Synchronous, No Prescaler | $0.5T_{CY}$+20 | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 47* | $T_T1P$ | T1CKI Input Period | Synchronous | — | — | — | ns | N = Prescale value |
| | | | Asynchronous | 60 | — | — | ns | |
| 49* | $TCKEZ_{TMR}1$ | Delay from External Clock Edge to Timer Increment | | 2 $T_{OSC}$ | — | 7 $T_{OSC}$ | — | Timers in Sync mode |

* - These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Figure 32-11. Timer0 and Timing1 External Clock Timings**



### 32.4.9 Capture/Compare/PWM Requirements (CCP)

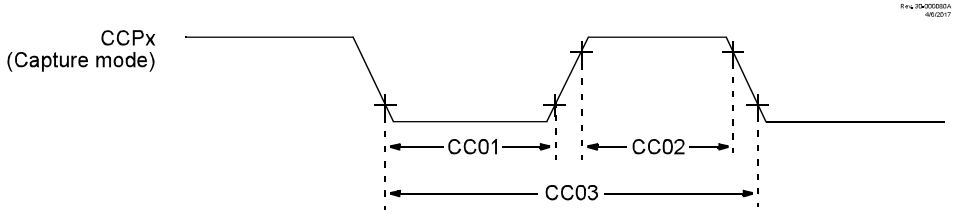
**Table 32-15.**

| Standard Operating Conditions (unless otherwise stated)<br>Operating Temperature: -40°C ≤ $T_A$ ≤ +125°C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| CC01* | $T_{CC}L$ | CCPx Input Low Time | No Prescaler | $0.5T_{CY}$+20 | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC02* | $T_{CC}H$ | CCPx Input High Time | No Prescaler | $0.5T_{CY}$+20 | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC03* | $T_{CC}P$ | CCPx Input Period | | $(3T_{CY}+40)/N$ | — | — | ns | N = Prescale value |

* - 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 32-12. Capture/Compare/PWM Timings (CCP)**



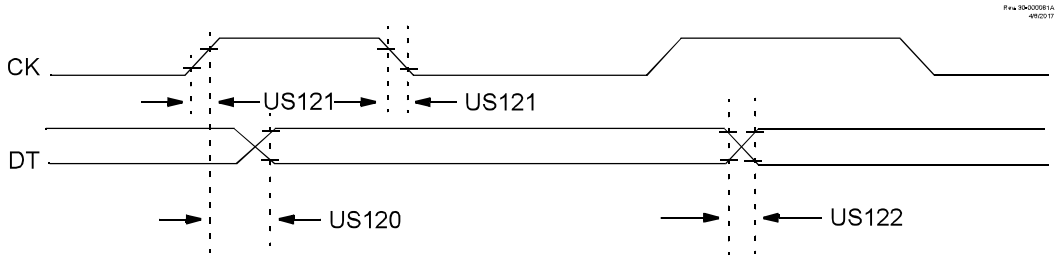

**Note:** Refer to Figure 32-3 for load conditions.

### 32.4.10 EUSART Synchronous Transmission Requirements

**Table 32-16.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Max.** | **Units** | **Conditions** |
| US120 | $T_{CK}H2_{DT}V$ | SYNC XMIT (Master and Slave)<br>Clock high to data-out valid | — | 80 | ns | 3.0V ≤ $V_{DD}$ ≤ 5.5V |
| | | | — | 100 | ns | 1.8V ≤ $V_{DD}$ ≤ 5.5V |
| US121 | $T_{CKRF}$ | Clock out rise time and fall time<br>(Master mode) | — | 45 | ns | 3.0V ≤ $V_{DD}$ ≤ 5.5V |
| | | | — | 50 | ns | 1.8V ≤ $V_{DD}$ ≤ 5.5V |
| US122 | $T_{DTRF}$ | Data-out rise time and fall time | — | 45 | ns | 3.0V ≤ $V_{DD}$ ≤ 5.5V |
| | | | — | 50 | ns | 1.8V ≤ $V_{DD}$ ≤ 5.5V |

**Figure 32-13. EUSART Synchronous Transmission (Master/Slave) Timing**



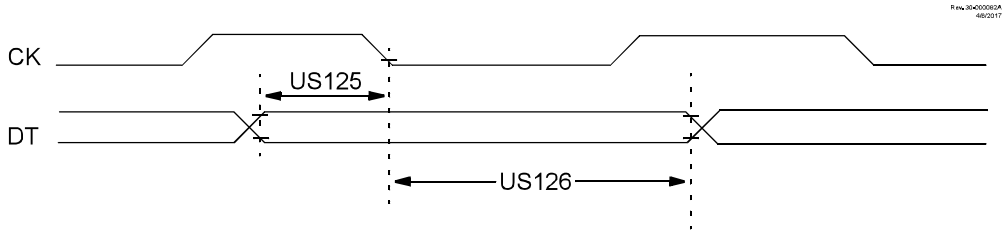

**Note:** Refer to Figure 32-3 for load conditions.

### 32.4.11 EUSART Synchronous Receive Requirements

**Table 32-17.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Max.** | **Units** | **Conditions** |
| US125 | $T_{DT}V2_{CKL}$ | SYNC RCV (Master and Slave)<br>Data-setup before CK ↓ (DT hold time) | 10 | — | ns | |
| US126 | $T_{CK}L2_{DTL}$ | Data-hold after CK ↓ (DT hold time) | 15 | — | ns | |

**Figure 32-14. EUSART Synchronous Receive (Master/Slave) Timing**




**Note:** Refer to Figure 32-3 for load conditions.

### 32.4.12 SPI Mode Requirements

**Table 32-18.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| SP70* | $T_{SS}L2_{SC}H$,<br>$T_{SS}L2_{SC}L$ | $\overline{SS}$↓ to SCK↓ or SCK↑ input | 2.25*$T_{CY}$ | — | — | ns | |
| SP71* | $T_{SC}H$ | SCK input high time (Slave mode) | $T_{CY}$ + 20 | — | — | ns | |
| SP72* | $T_{SC}L$ | SCK input low time (Slave mode) | $T_{CY}$ + 20 | — | — | ns | |
| SP73* | $T_{DI}V2_{SC}H$,<br>$T_{DI}V2_{SC}L$ | Setup time of SDI data input to SCK edge | 100 | — | — | ns | |
| SP74* | $T_{SC}H2_{DI}L$,<br>$T_{SC}L2_{DI}L$ | Hold time of SDI data input to SCK edge | 100 | — | — | ns | |

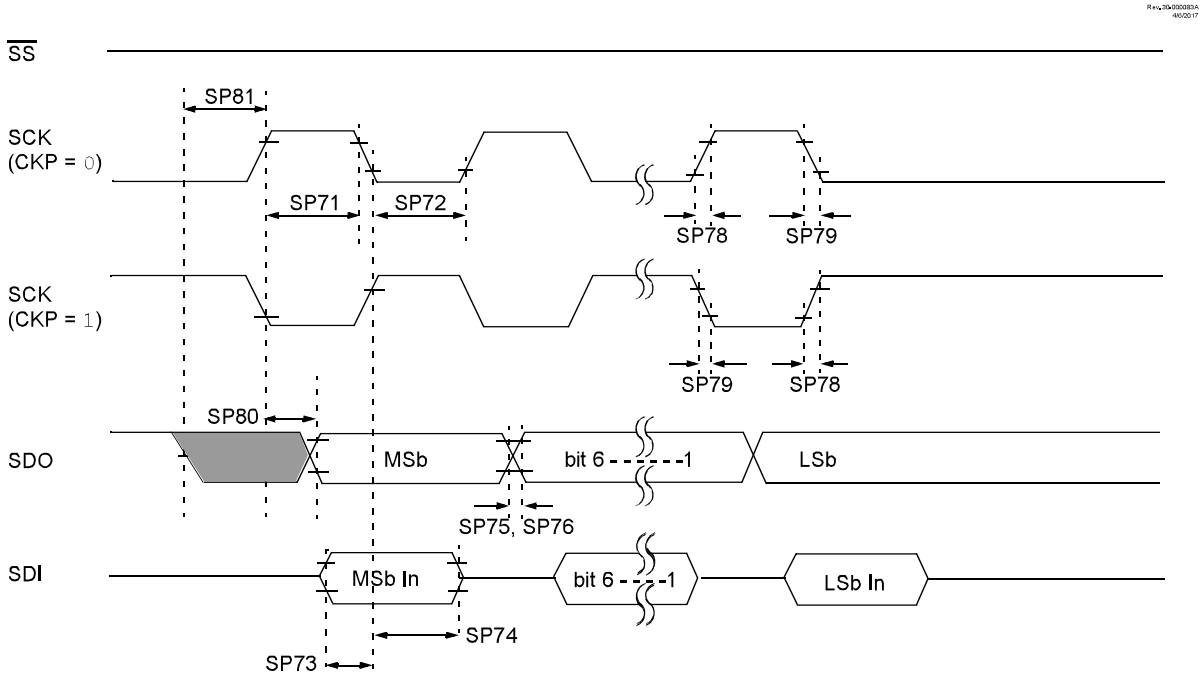
.....continued

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| SP75* | $T_{DO}R$ | SDO data output rise time | — | 25 | 50 | ns | 1.8V ≤ $V_{DD}$ ≤ 5.5V |
| SP76* | $T_{DO}F$ | SDO data output fall time | — | 10 | 25 | ns | |
| SP77* | $T_{SS}H2_{DO}Z$ | $\overline{SS}$↑ to SDO output high-impedance | 10 | — | 50 | ns | |
| SP78* | $T_{SC}R$ | SCK output rise time (Master mode) | — | 25 | 50 | ns | 1.8V ≤ $V_{DD}$ ≤ 5.5V |
| SP79* | $T_{SC}F$ | SCK output fall time (Master mode) | — | 10 | 25 | ns | |
| SP80* | $T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$ | SDO data output valid after SCK edge | — | — | 145 | ns | 1.8V ≤ $V_{DD}$ ≤ 5.5V |
| SP81* | $T_{DO}V2_{SC}H$, $T_{DO}V2_{SC}L$ | SDO data output setup to SCK edge | 1 $T_{CY}$ | — | — | ns | |
| SP82* | $T_{SS}L2_{DO}V$ | SDO data output valid after $\overline{SS}$↓ edge | — | — | 50 | ns | |
| SP83* | $T_{SC}H2_{SS}H$, $T_{SC}L2_{SS}H$ | $\overline{SS}$ ↑□after SCK edge | 1.5 $T_{CY}$ + 40 | — | — | ns | |

* - These parameters are characterized but not tested.

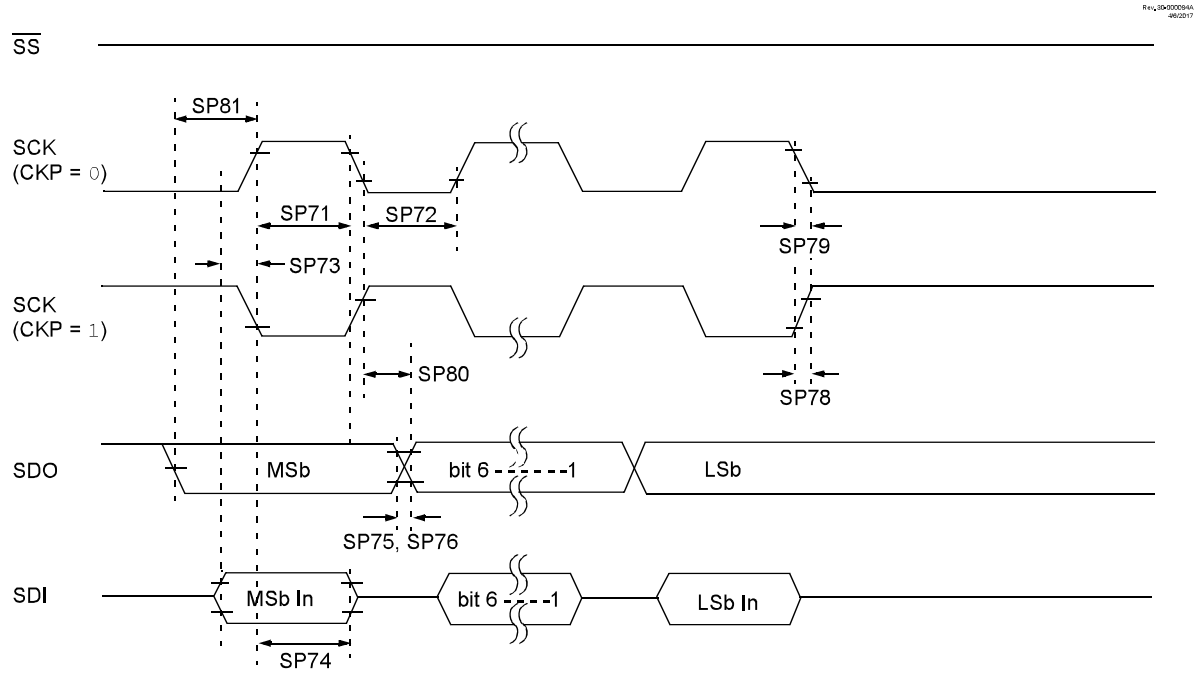
† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Figure 32-15. SPI Master Mode Timing (CKE = 0, SMP = 0)**



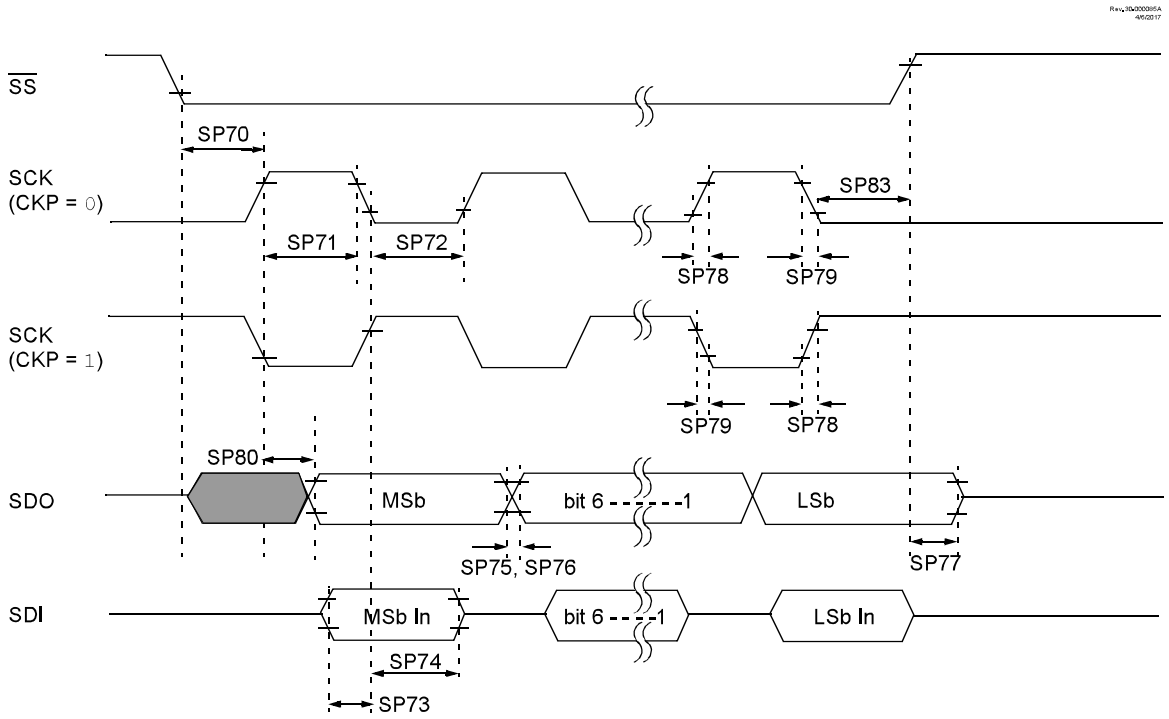

**Note:** Refer to Figure 32-3 for load conditions.

**Figure 32-16. SPI Master Mode Timing (CKE = 1, SMP = 1)**



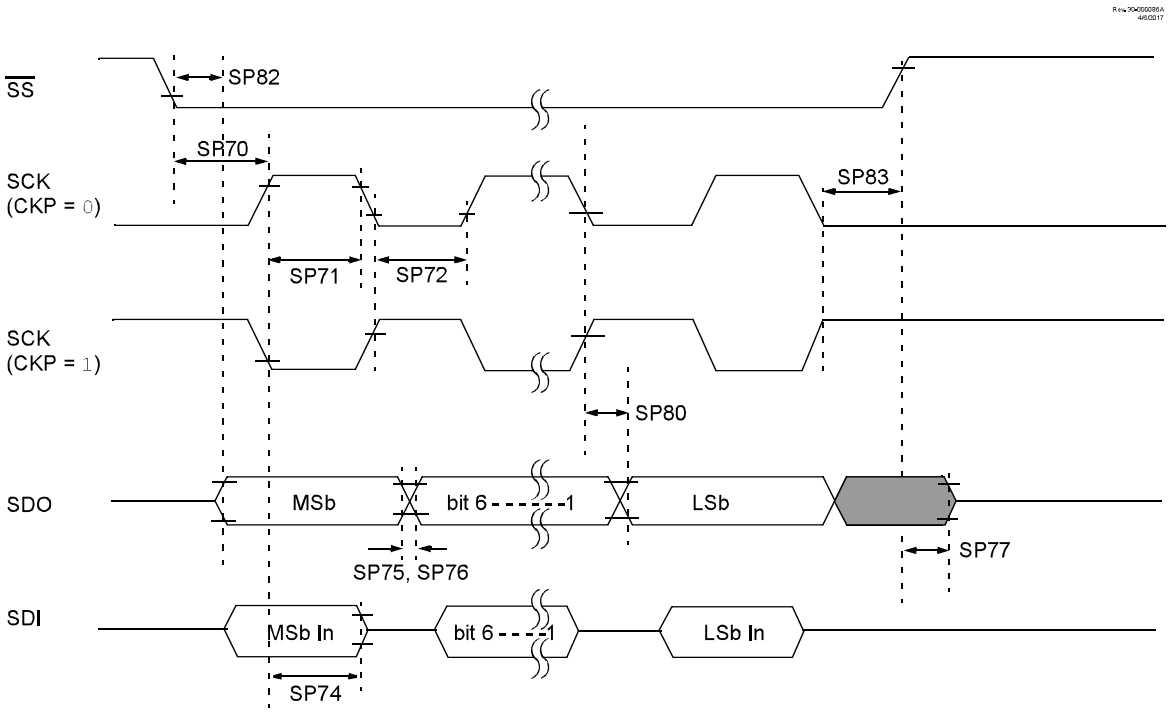

**Note:** Refer to Figure 32-3 for load conditions.

**Figure 32-17. SPI Slave Mode Timing (CKE = 0)**




**Note:** Refer to Figure 32-3 for load conditions.

**Figure 32-18. SPI Slave Mode Timing (CKE = 1)**



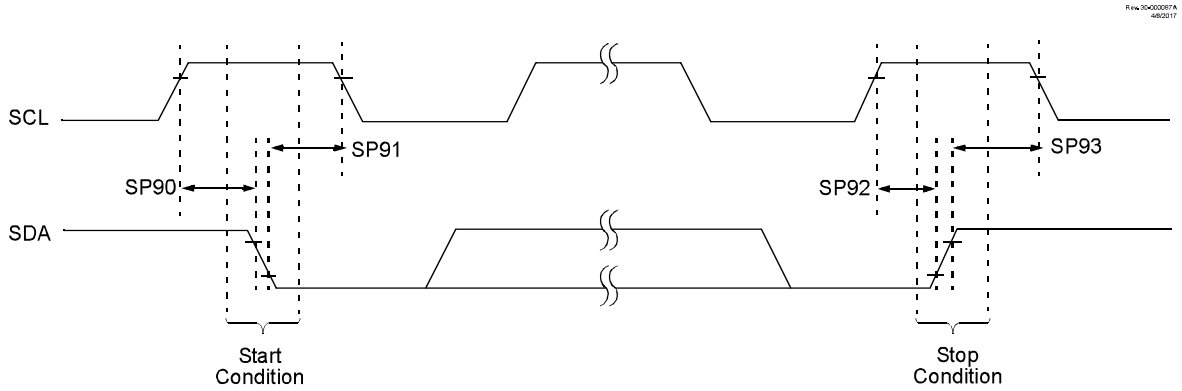

**Note:** Refer to Figure 32-3 for load conditions.

### 32.4.13 I²C Bus Start/Stop Bits Requirements

**Table 32-19.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | | Min. | Typ. † | Max. | Units | Conditions |
| SP90* | $T_{SU:STA}$ | Start condition Setup time | 100 kHz mode | 4700 | — | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 600 | — | — | | |
| SP91* | $T_{HD:STA}$ | Start condition Hold time | 100 kHz mode | 4000 | — | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 600 | — | — | | |
| SP92* | $T_{SU:STO}$ | Stop condition Setup time | 100 kHz mode | 4700 | — | — | ns | |
| | | | 400 kHz mode | 600 | — | — | | |
| SP93* | $T_{HD:STO}$ | Stop condition Hold time | 100 kHz mode | 4000 | — | — | ns | |
| | | | 400 kHz mode | 600 | — | — | | |
| * - These parameters are characterized but not tested. | | | | | | | | |

**Figure 32-19. I²C Bus Start/Stop Bits Timing**




**Note:** Refer to Figure 32-3 for load conditions.

### 32.4.14 I²C Bus Data Requirements

**Table 32-20.**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | | Min. | Max. | Units | Conditions |
| SP100* | $T_{HIGH}$ | Clock high time | 100 kHz mode | 4.0 | — | µs | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 0.6 | — | µs | Device must operate at a minimum of 10 MHz |
| | | | SSP module | $1.5T_{CY}$ | — | | |

.....continued

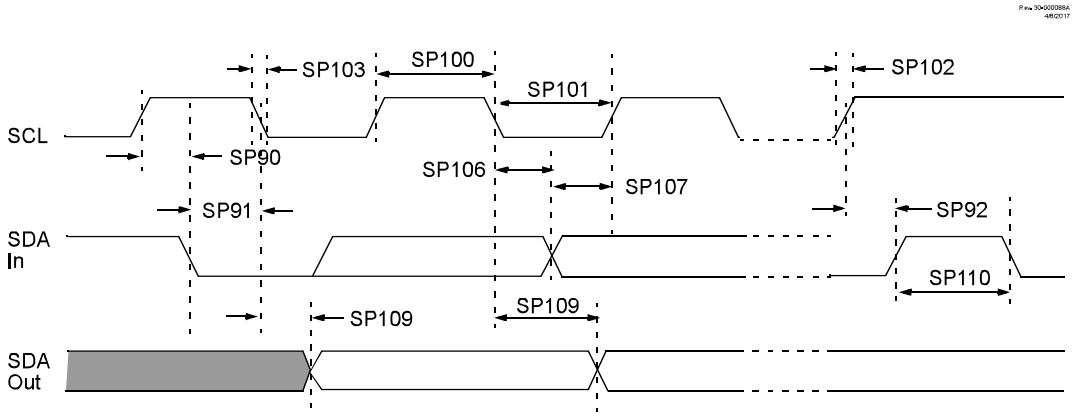
| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | | Min. | Max. | Units | Conditions |
| SP101* | $T_{LOW}$ | Clock low time | 100 kHz mode | 4.7 | — | µs | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 1.3 | — | µs | Device must operate at a minimum of 10 MHz |
| | | | SSP module | $1.5T_{CY}$ | — | | |
| SP102* | $T_R$ | SDA and SCL rise time | 100 kHz mode | — | 1000 | ns | |
| | | | 400 kHz mode | $20 + 0.1C_B$ | 300 | ns | $C_B$ is specified to be from 10-400 pF |
| SP103* | $T_F$ | SDA and SCL fall time | 100 kHz mode | — | 250 | ns | |
| | | | 400 kHz mode | $20 + 0.1C_B$ | 250 | ns | $C_B$ is specified to be from 10-400 pF |
| SP106* | $T_{HD:DAT}$ | Data input hold time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | µs | |
| SP107* | $T_{SU:DAT}$ | Data input setup time | 100 kHz mode | 250 | — | ns | **Note 2** |
| | | | 400 kHz mode | 100 | — | ns | |
| SP109* | $T_{AA}$ | Output valid from clock | 100 kHz mode | — | 3500 | ns | **Note 1** |
| | | | 400 kHz mode | — | — | ns | |
| SP110* | $T_{BUF}$ | Bus free time | 100 kHz mode | 4.7 | — | µs | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | µs | |
| SP111 | $C_B$ | Bus capacitive loading | | — | 400 | pF | |

* - These parameters are characterized but not tested.

**Notes:**

1. As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.
2. A Fast mode (400 kHz) $I^2C$ bus device can be used in a Standard mode (100 kHz) $I^2C$ bus system, but the requirement $T_{SU:DAT} \geq 250$ ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line TR max. + $T_{SU:DAT}$ = 1000 + 250 = 1250 ns (according to the Standard mode $I^2C$ bus specification), before the SCL line is released.

**Figure 32-20. I²C Bus Data Timing**




**Note:** Refer to Figure 32-3 for load conditions.

# 33. DC and AC Characteristics Graphs and Tables

Graphs and tables are not available at this time.

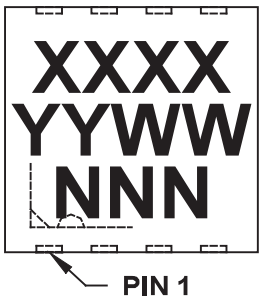
# 34. Packaging Information

Package Marking Information

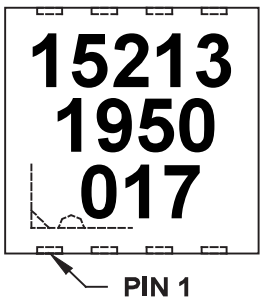
Rev. 30-000000A
5/17/2017

| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information or Microchip part number |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week '01') |
| | NNN | Alphanumeric traceability code |
| | (e3) | b- 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. | |

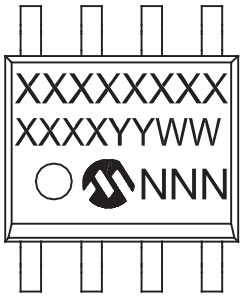
8-Lead DFN (3x3x0.9 mm)



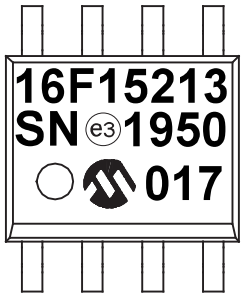

Example




8-Lead SOIC (3.90 mm)

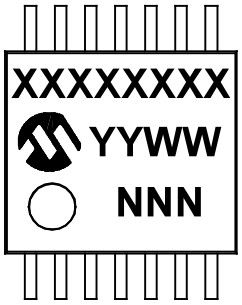



Example

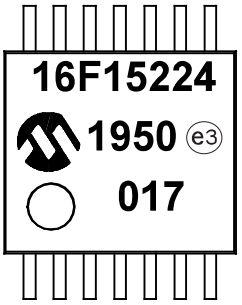


Rev. 30-0300114B
09/21/2017

14-Lead TSSOP (4.4 mm)

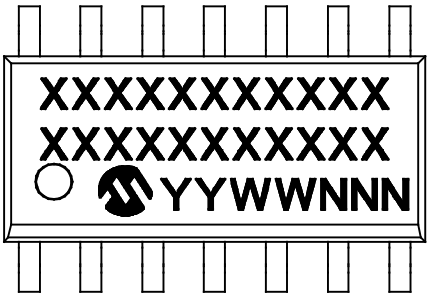



Example

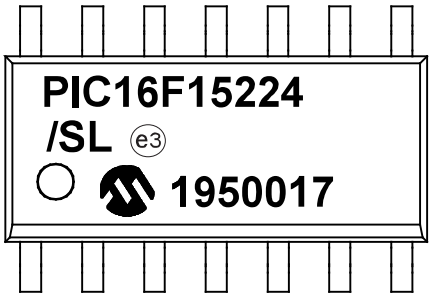



Rev. 30-0300114C
09/21/2017

14-Lead SOIC (3.90 mm)

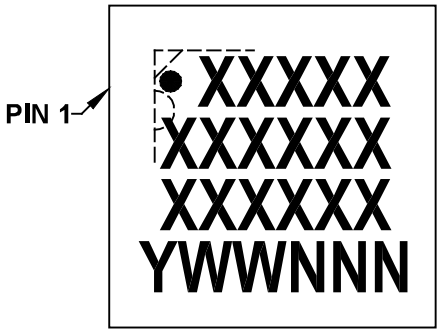



Example

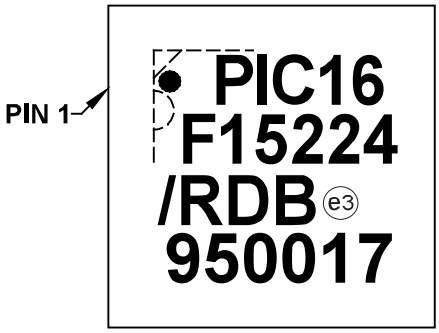



Rev. 30-0300165
12/10/2019

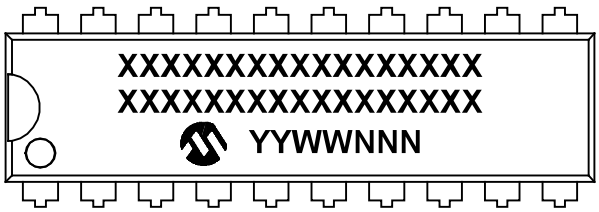
16-Lead VQFN (3x3x0.9 mm)



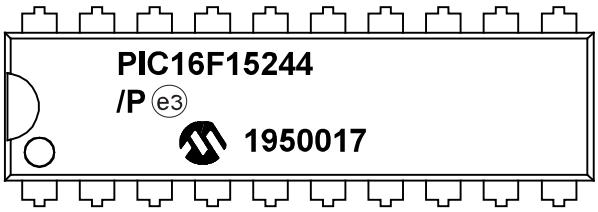

Example



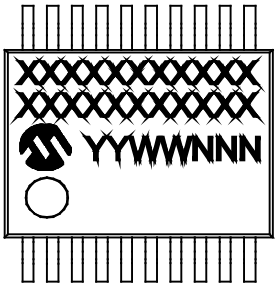
20-Lead PDIP (300 mil)



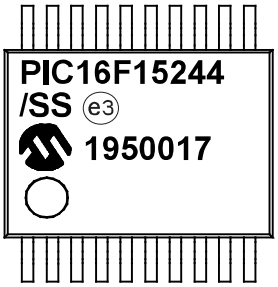

Example



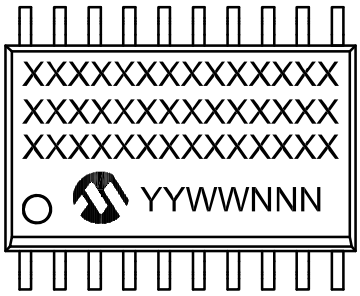

20-Lead SSOP (5.30 mm)



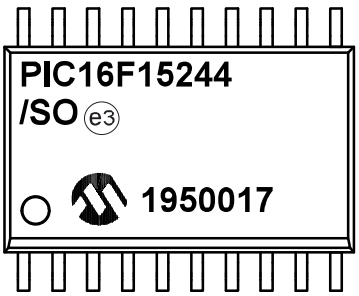

Example



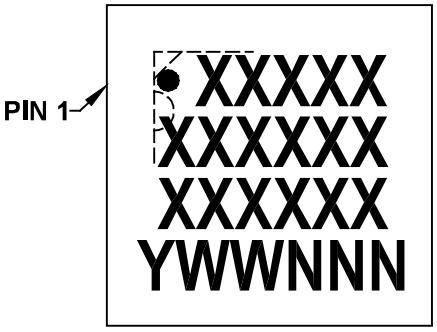

20-Lead SOIC (7.50 mm)



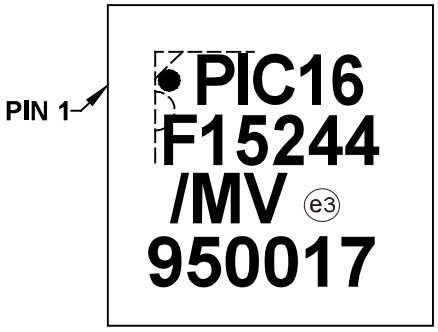

Example



20-Lead VQFN (3x3x0.9 mm)




Example

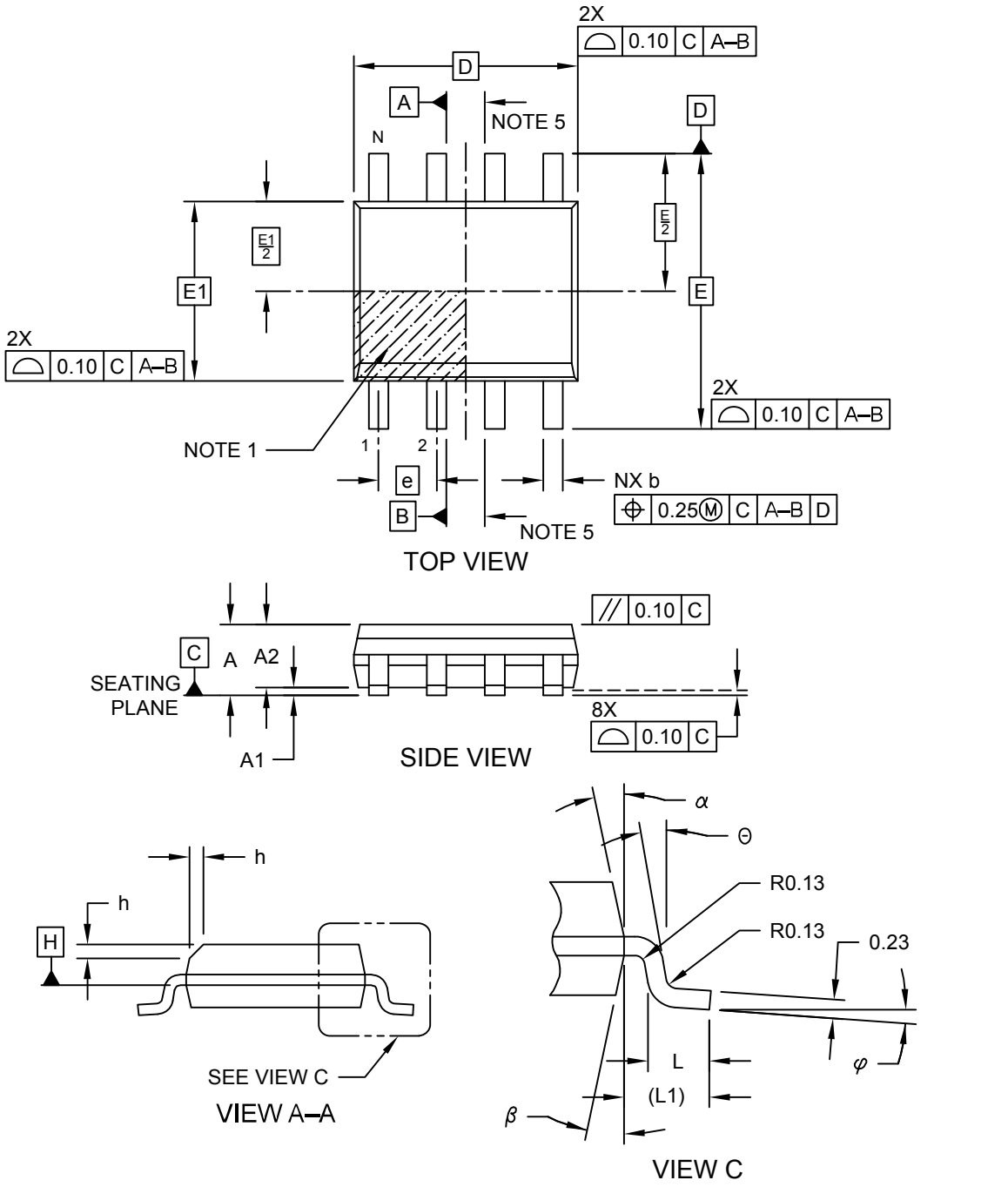



## 34.1 Package Details

The following sections give the technical details of the packages.

## 8-Lead Plastic Small Outline (SN) - Narrow, 3.90 mm (.150 In.) Body [SOIC]

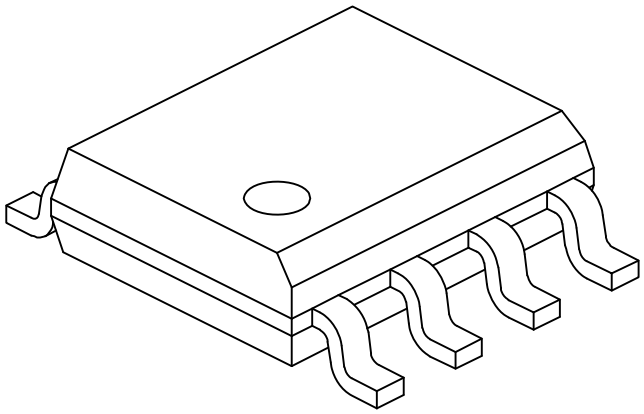
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




Microchip Technology Drawing No. C04-057-SN Rev F Sheet 1 of 2

## 8-Lead Plastic Small Outline (SN) - Narrow, 3.90 mm (.150 In.) Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 8 | |
| Pitch | e | | 1.27 BSC | |
| Overall Height | A | - | - | 1.75 |
| Molded Package Thickness | A2 | 1.25 | - | - |
| Standoff § | A1 | 0.10 | - | 0.25 |
| Overall Width | E | | 6.00 BSC | |
| Molded Package Width | E1 | | 3.90 BSC | |
| Overall Length | D | | 4.90 BSC | |
| Chamfer (Optional) | h | 0.25 | - | 0.50 |
| Foot Length | L | 0.40 | - | 1.27 |
| Footprint | L1 | | 1.04 REF | |
| Foot Angle | φ | 0° | - | 8° |
| Lead Thickness | c | 0.17 | - | 0.25 |
| Lead Width | b | 0.31 | - | 0.51 |
| Mold Draft Angle Top | α | 5° | - | 15° |
| Mold Draft Angle Bottom | β | 5° | - | 15° |

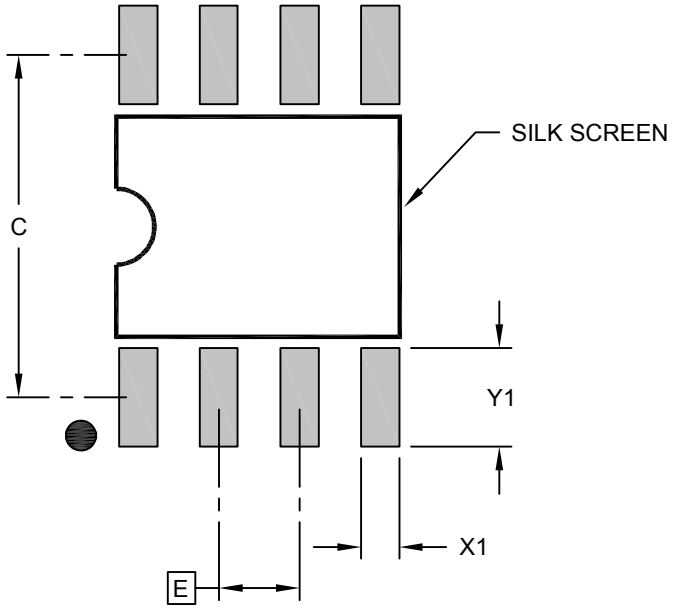
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing No. C04-057-SN Rev F Sheet 2 of 2

## 8-Lead Plastic Small Outline (SN) - Narrow, 3.90 mm (.150 In.) 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 | | 5.40 | |
| Contact Pad Width (X8) | X1 | | | 0.60 |
| Contact Pad Length (X8) | Y1 | | | 1.55 |

Notes:

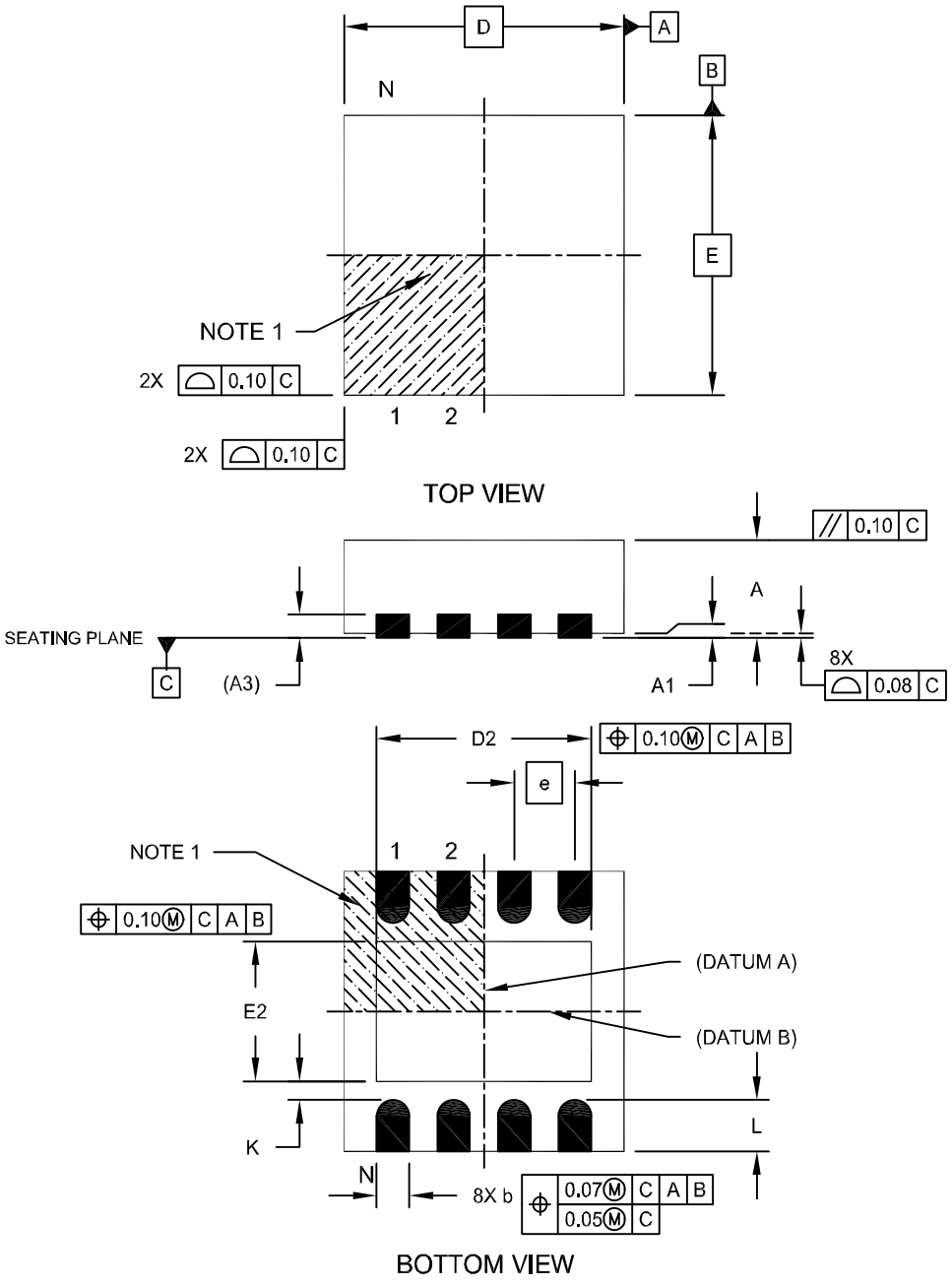
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-2057-SN Rev F

### 8-Lead Plastic Dual Flat, No Lead Package (MF) - 3x3x0.9mm Body [DFN]

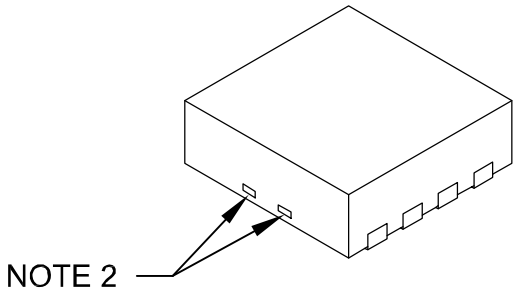
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




Microchip Technology Drawing No. C04-062C Sheet 1 of 2

### 8-Lead Plastic Dual Flat, No Lead Package (MF) - 3x3x0.9mm Body [DFN]

**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 | | 8 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Contact Thickness | A3 | | 0.20 REF | |
| Overall Length | D | | 3.00 BSC | |
| Exposed Pad Width | E2 | 1.34 | - | 1.60 |
| Overall Width | E | | 3.00 BSC | |
| Exposed Pad Length | D2 | 1.60 | - | 2.40 |
| Contact Width | b | 0.25 | 0.30 | 0.35 |
| Contact Length | L | 0.20 | 0.30 | 0.55 |
| Contact-to-Exposed Pad | K | 0.20 | - | - |

Notes:

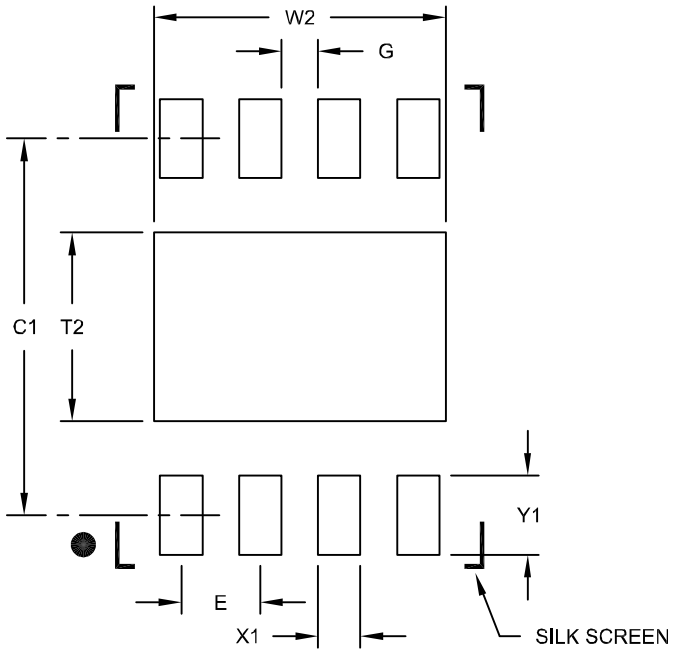
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package may have one or more exposed tie bars at ends.
3. Package is saw singulated
4. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing No. C04-062C Sheet 2 of 2

### 8-Lead Plastic Dual Flat, No Lead Package (MF) - 3x3x0.9mm Body [DFN]

**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 | |
| Optional Center Pad Width | W2 | | | 2.40 |
| Optional Center Pad Length | T2 | | | 1.55 |
| Contact Pad Spacing | C1 | | 3.10 | |
| Contact Pad Width (X8) | X1 | | | 0.35 |
| Contact Pad Length (X8) | Y1 | | | 0.65 |
| Distance Between Pads | G | 0.30 | | |

Notes:

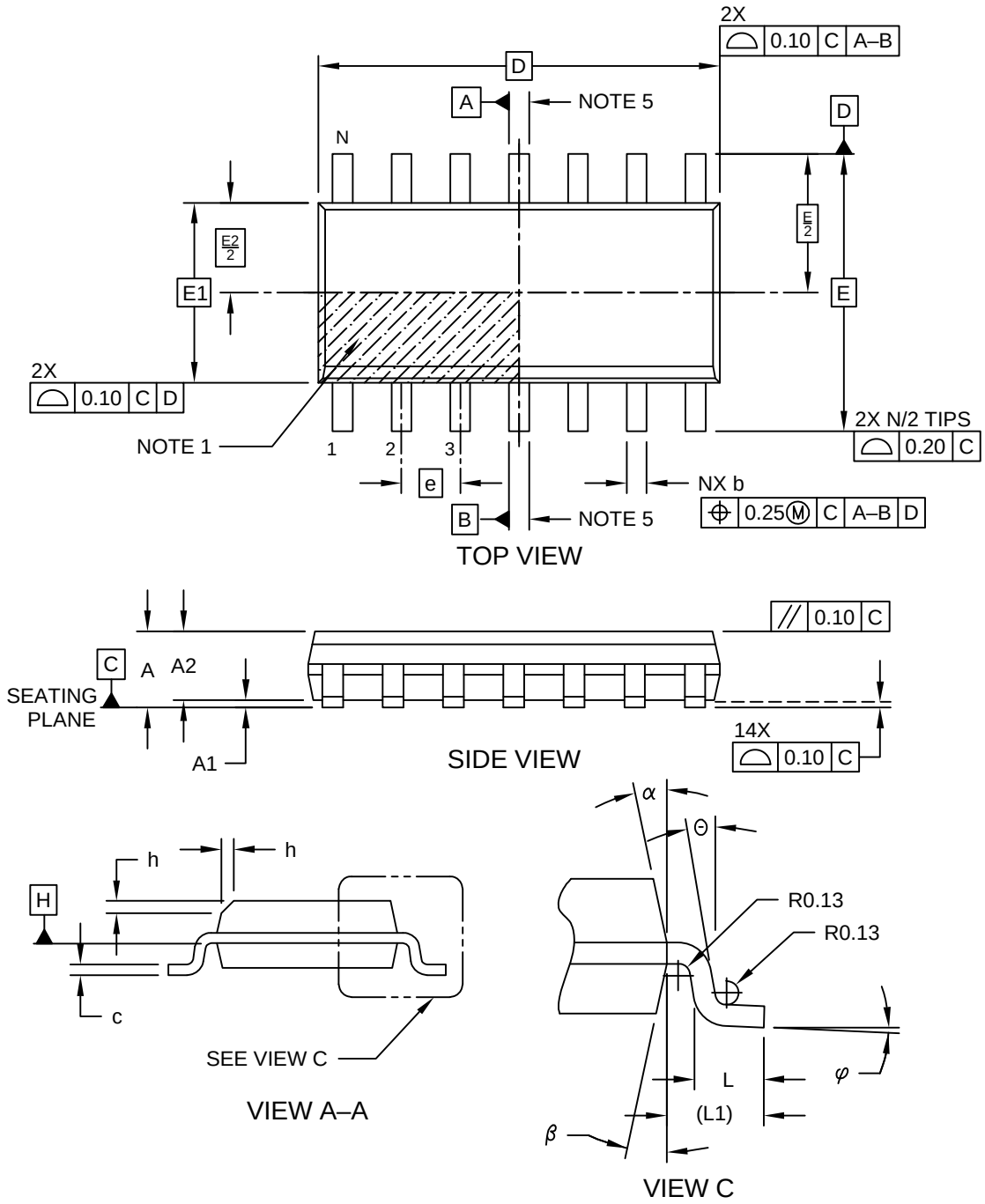
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2062B

## 14-Lead Plastic Small Outline (SL) - Narrow, 3.90 mm Body [SOIC]

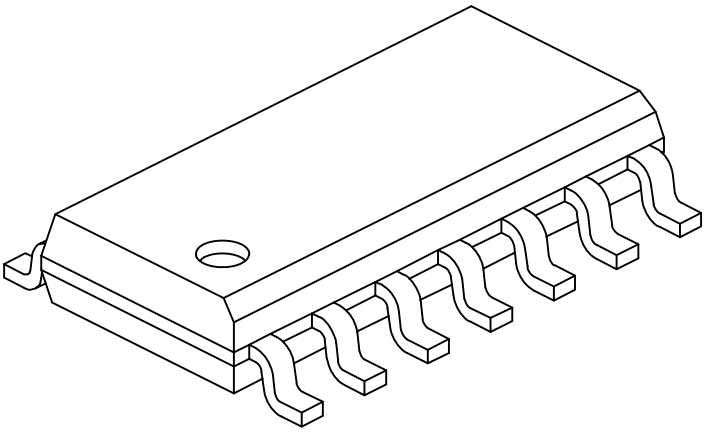
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




Microchip Technology Drawing No. C04-065-SL Rev D Sheet 1 of 2

## 14-Lead Plastic Small Outline (SL) - Narrow, 3.90 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 14 | |
| Pitch | e | | 1.27 BSC | |
| Overall Height | A | - | - | 1.75 |
| Molded Package Thickness | A2 | 1.25 | - | - |
| Standoff § | A1 | 0.10 | - | 0.25 |
| Overall Width | E | | 6.00 BSC | |
| Molded Package Width | E1 | | 3.90 BSC | |
| Overall Length | D | | 8.65 BSC | |
| Chamfer (Optional) | h | 0.25 | - | 0.50 |
| Foot Length | L | 0.40 | - | 1.27 |
| Footprint | L1 | | 1.04 REF | |
| Lead Angle | Θ | 0° | - | - |
| Foot Angle | φ | 0° | - | 8° |
| Lead Thickness | c | 0.10 | - | 0.25 |
| Lead Width | b | 0.31 | - | 0.51 |
| Mold Draft Angle Top | α | 5° | - | 15° |
| Mold Draft Angle Bottom | β | 5° | - | 15° |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

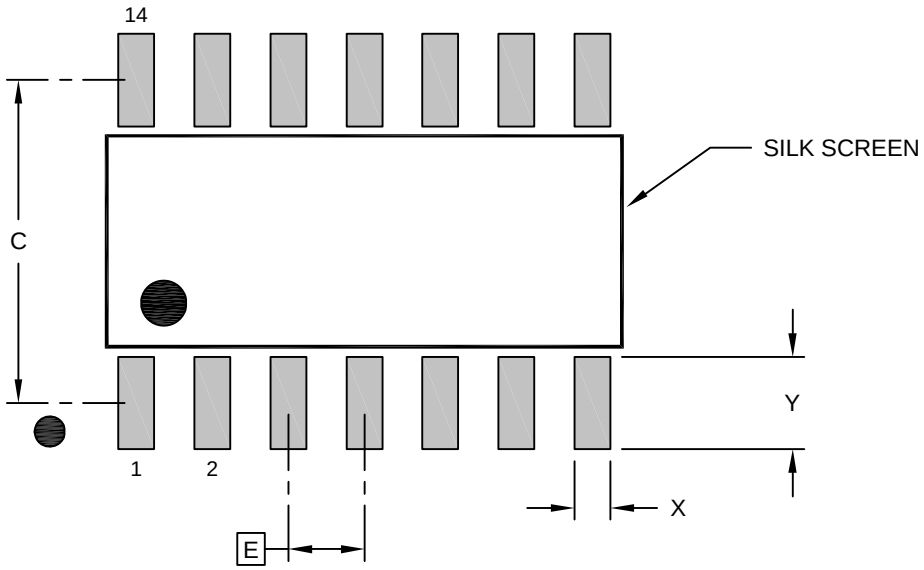
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing No. C04-065-SL Rev D Sheet 2 of 2

## 14-Lead Plastic Small Outline (SL) - Narrow, 3.90 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 | | 5.40 | |
| Contact Pad Width (X14) | X | | | 0.60 |
| Contact Pad Length (X14) | Y | | | 1.55 |

Notes:

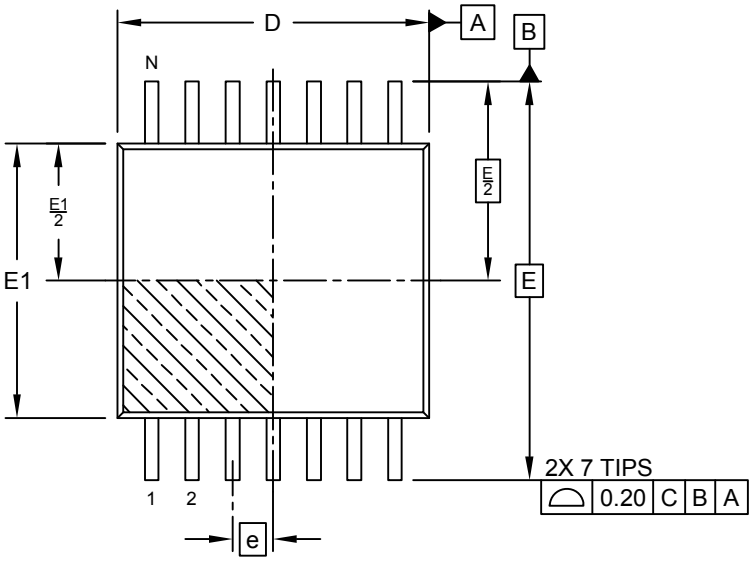
1. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

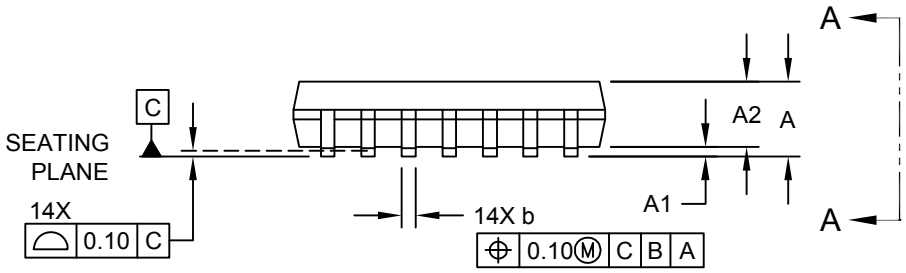
Microchip Technology Drawing No. C04-2065-SL Rev D

## 14Lead Thin Shrink Small Outline Package [ST] 4.4 mm Body [TSSOP]

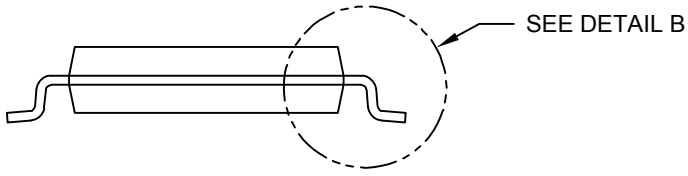
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




TOP VIEW




SIDE VIEW

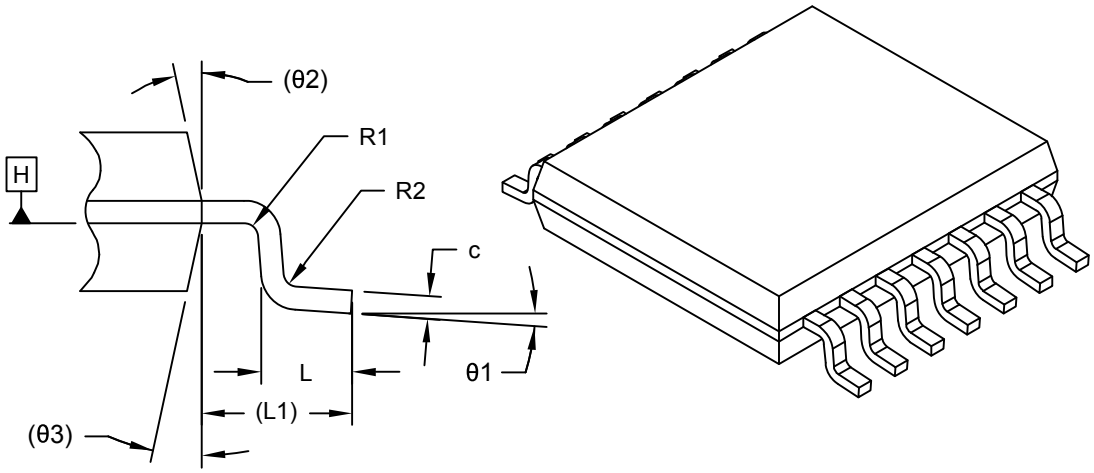



VIEW A–A

Microchip Technology Drawing C04-087 Rev D Sheet 1 of 2

## 14Lead Thin Shrink Small Outline Package [ST] 4.4 mm Body [TSSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




DETAIL B

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Terminals | N | | 14 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | – | – | 1.20 |
| Standoff | A1 | 0.05 | – | 0.15 |
| Molded Package Thickness | A2 | 0.80 | 1.00 | 1.05 |
| Overall Length | D | 4.90 | 5.00 | 5.10 |
| Overall Width | E | | 6.40 BSC | |
| Molded Package Width | E1 | 4.30 | 4.40 | 4.50 |
| Terminal Width | b | 0.19 | – | 0.30 |
| Terminal Thickness | c | 0.09 | – | 0.20 |
| Terminal Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | | 1.00 REF | |
| Lead Bend Radius | R1 | 0.09 | – | – |
| Lead Bend Radius | R2 | 0.09 | – | – |
| Foot Angle | θ1 | 0° | – | 8° |
| Mold Draft Angle | θ2 | – | 12° REF | – |
| Mold Draft Angle | θ3 | – | 12° REF | – |

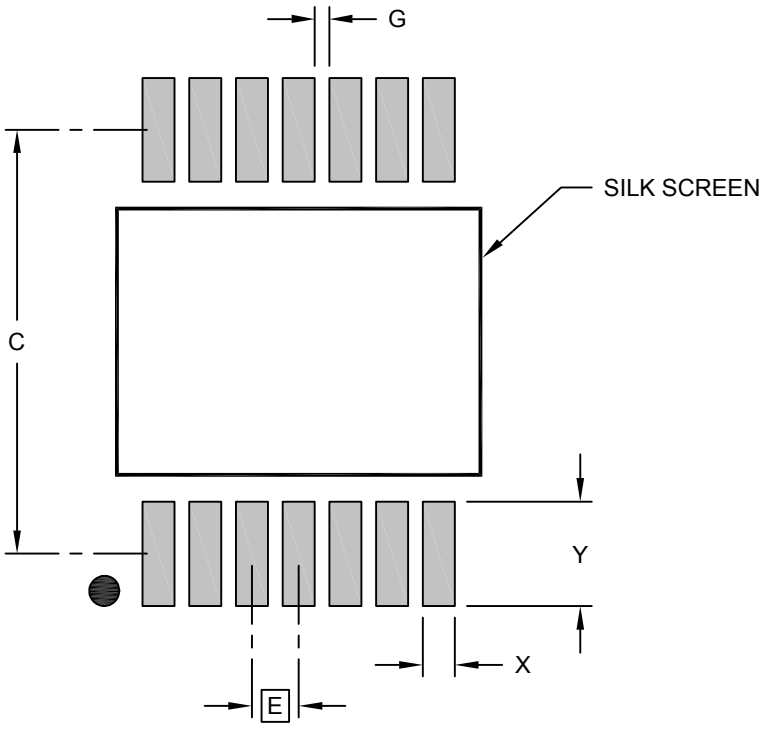
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. 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 C04-087 Rev D Sheet 2 of 2

### 14Lead Thin Shrink Small Outline Package [ST] 4.4 mm Body [TSSOP]

**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 | | 5.90 | |
| Contact Pad Width (Xnn) | X | | | 0.45 |
| Contact Pad Length (Xnn) | Y | | | 1.45 |
| Contact Pad to Contact Pad (Xnn) | G | 0.20 | | |

Notes:

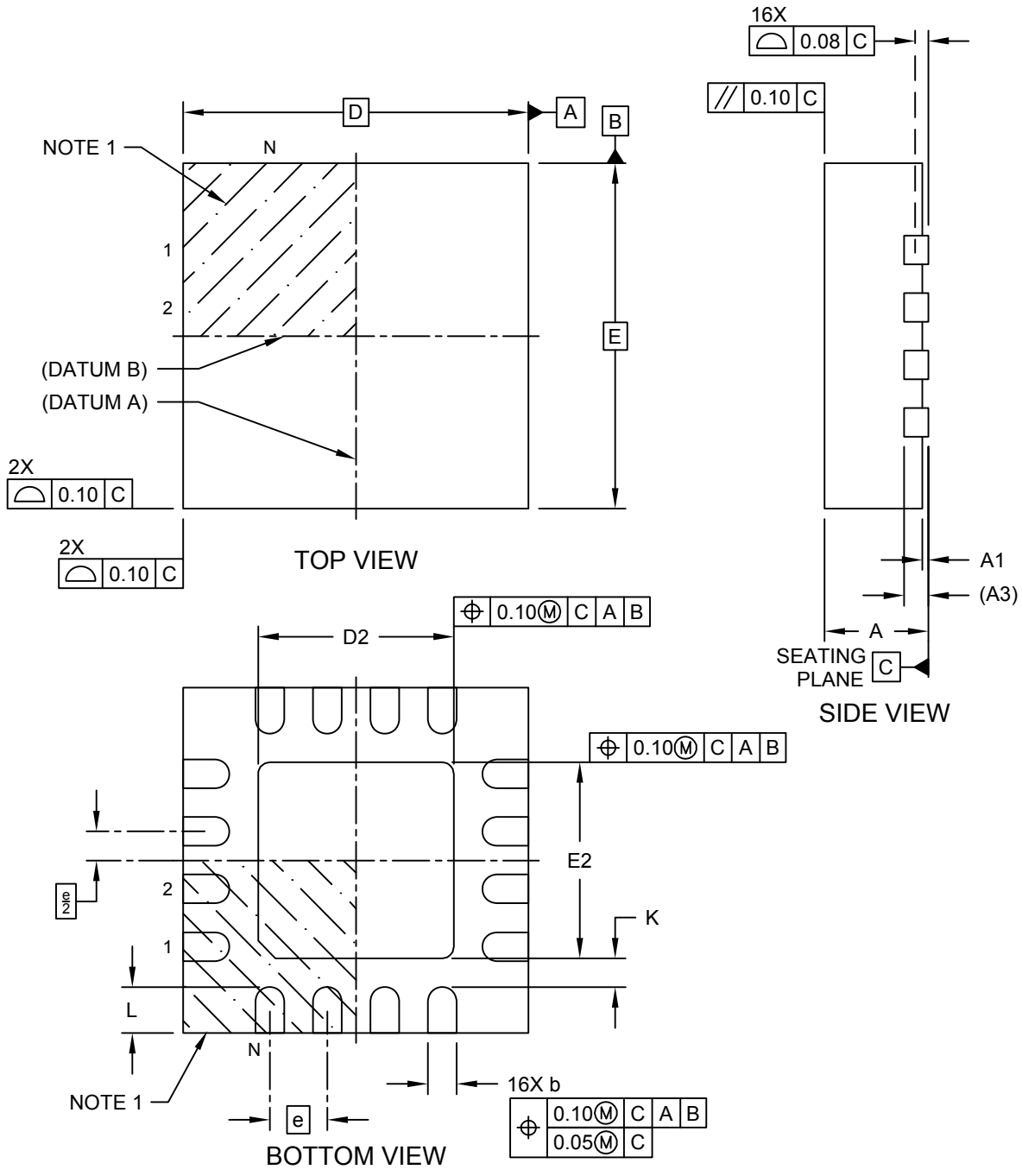
1. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-2087 Rev D

**16-Lead Very Thin Plastic Quad Flat, No Lead Package (RDB) - 3x3x0.9 mm Body [VQFN] With 1.7 mm Exposed Pad; Atmel Legacy Package ZPX**

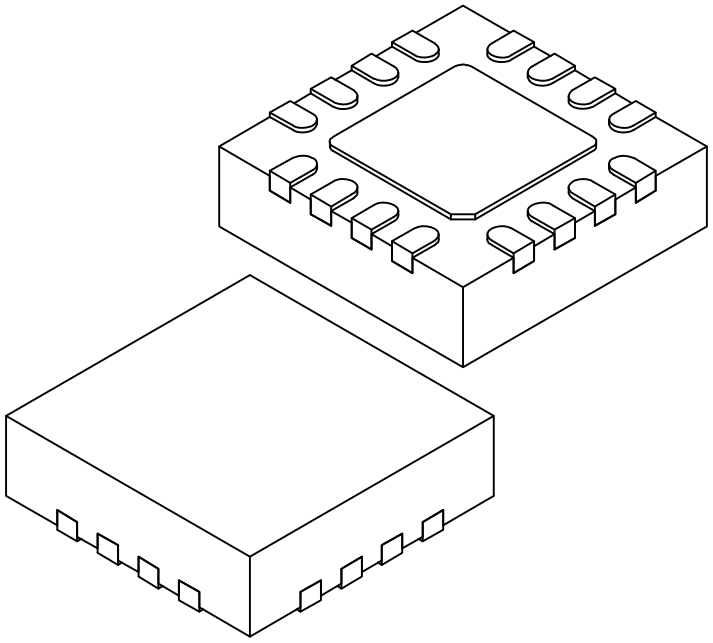
| **Note:** | For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging |
|---|---|




Microchip Technology Drawing C04-21379 Rev A Sheet 1 of 2

### 16-Lead Very Thin Plastic Quad Flat, No Lead Package (RDB) - 3x3x0.9 mm Body [VQFN] With 1.7 mm Exposed Pad; Atmel Legacy Package ZPX

**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 Terminals | N | | 16 | |
| Pitch | e | | 0.50 BSC | |
| Overall Height | A | 0.80 | 0.85 | 0.90 |
| Standoff | A1 | 0.00 | 0.035 | 0.05 |
| Terminal Thickness | A3 | | 0.21 REF | |
| Overall Length | D | | 3.00 BSC | |
| Exposed Pad Length | D2 | 1.60 | 1.70 | 1.80 |
| Overall Width | E | | 3.00 BSC | |
| Exposed Pad Width | E2 | 1.60 | 1.70 | 1.80 |
| Terminal Width | b | 0.20 | 0.25 | 0.30 |
| Terminal Length | L | 0.35 | 0.40 | 0.45 |
| Terminal-to-Exposed-Pad | K | 0.20 | – | – |

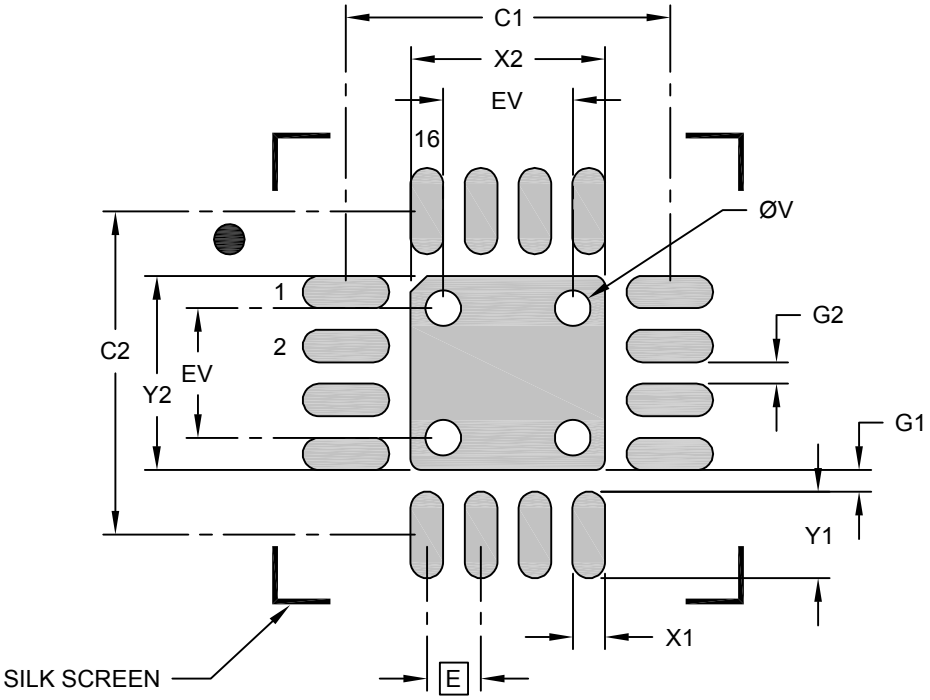
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-21379 Rev A Sheet 2 of 2

## 16-Lead Very Thin Plastic Quad Flat, No Lead Package (RDB) - 3x3x0.9 mm Body [VQFN] With 1.7 mm Exposed Pad; Atmel Legacy Package ZPX

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.50 BSC | |
| Optional Center Pad Width | X2 | | | 1.80 |
| Optional Center Pad Length | Y2 | | | 1.80 |
| Contact Pad Spacing | C1 | | 3.00 | |
| Contact Pad Spacing | C2 | | 3.00 | |
| Contact Pad Width (X16) | X1 | | | 0.30 |
| Contact Pad Length (X16) | Y1 | | | 0.80 |
| Contact Pad to Center Pad (X16) | G1 | 0.20 | | |
| Contact Pad to Contact Pad (X12) | G2 | 0.20 | | |
| Thermal Via Diameter | V | | 0.33 | |
| Thermal Via Pitch | EV | | 1.20 | |

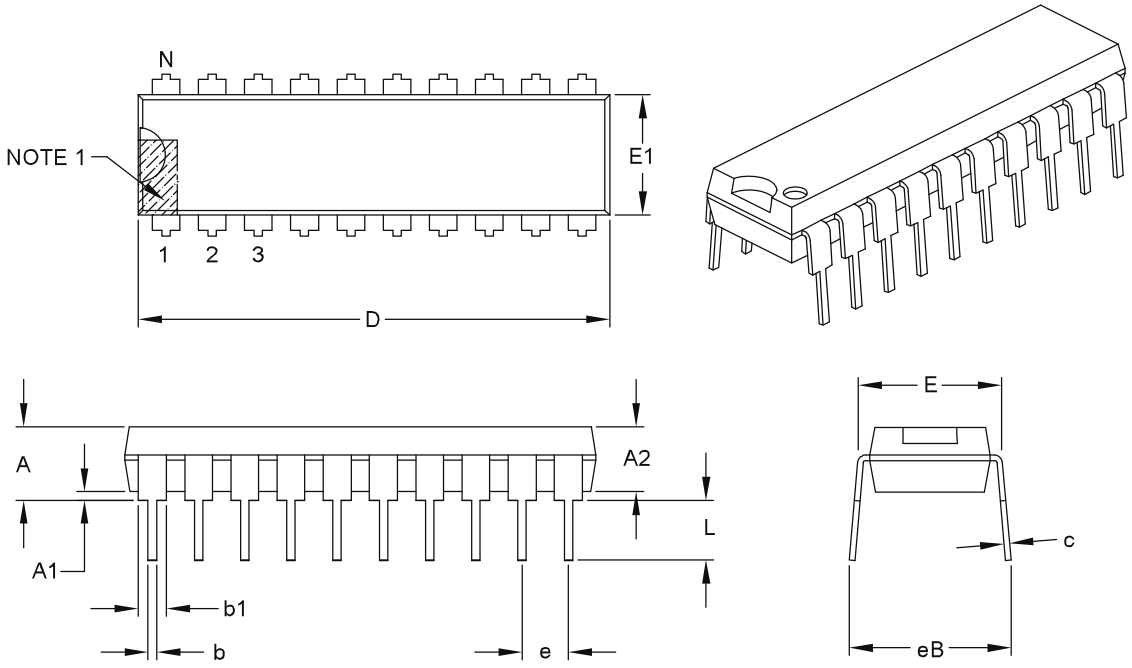
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23379 Rev A

### 20-Lead Plastic Dual In-Line (P) – 300 mil Body [PDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | INCHES | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 20 | |
| Pitch | e | | .100 BSC | |
| Top to Seating Plane | A | – | – | .210 |
| Molded Package Thickness | A2 | .115 | .130 | .195 |
| Base to Seating Plane | A1 | .015 | – | – |
| Shoulder to Shoulder Width | E | .300 | .310 | .325 |
| Molded Package Width | E1 | .240 | .250 | .280 |
| Overall Length | D | .980 | 1.030 | 1.060 |
| Tip to Seating Plane | L | .115 | .130 | .150 |
| Lead Thickness | c | .008 | .010 | .015 |
| Upper Lead Width | b1 | .045 | .060 | .070 |
| Lower Lead Width | b | .014 | .018 | .022 |
| Overall Row Spacing § | eB | – | – | .430 |

**Notes:**

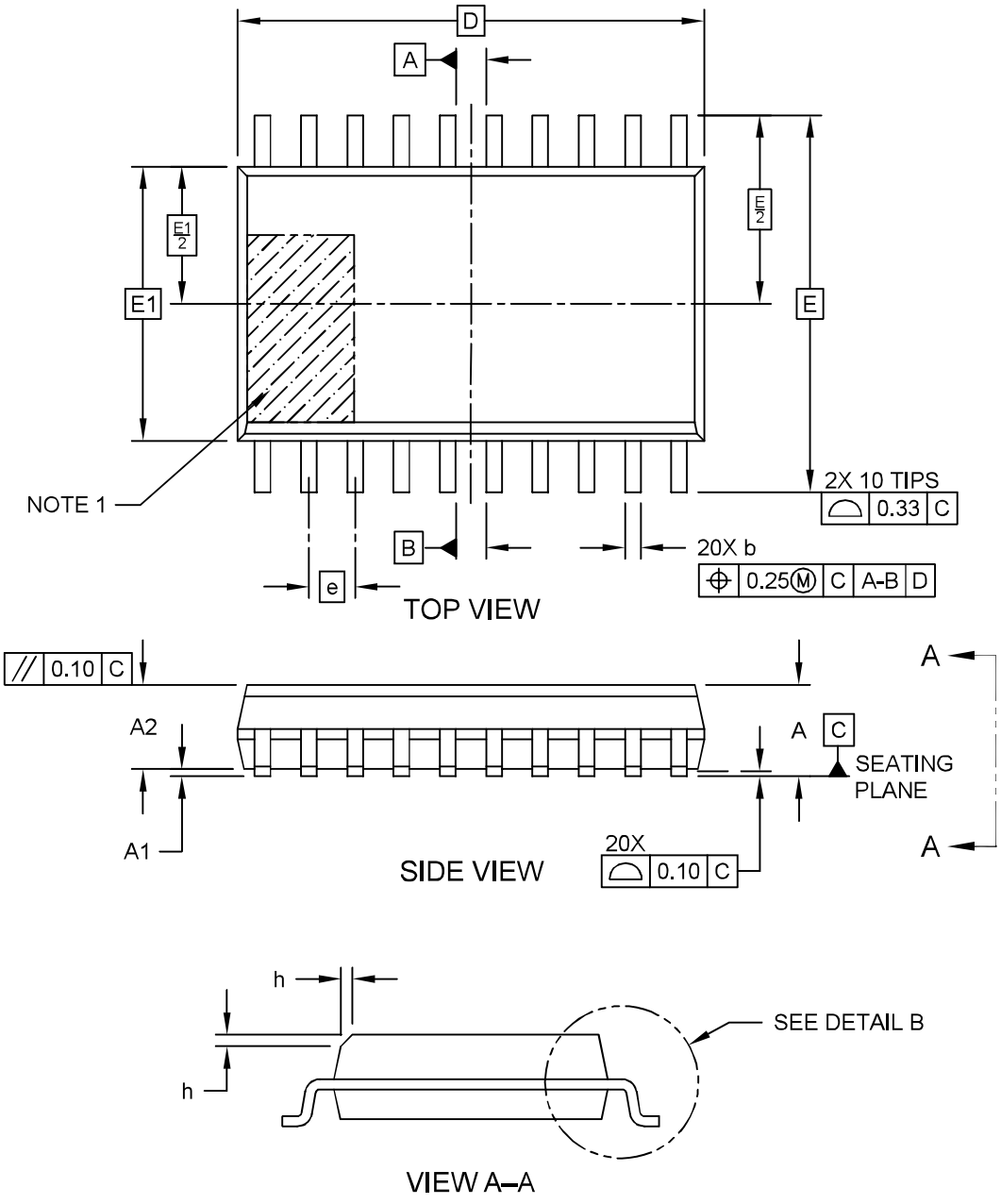
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-019B

## 20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

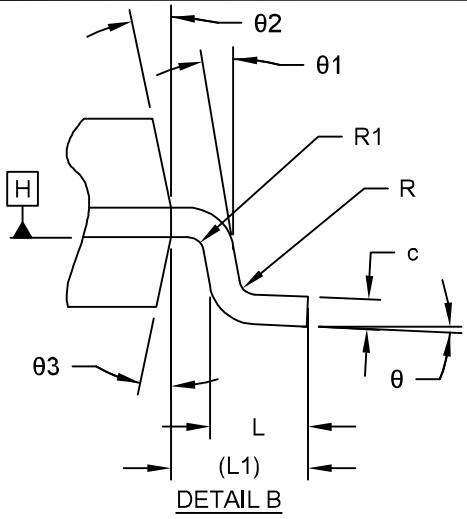
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

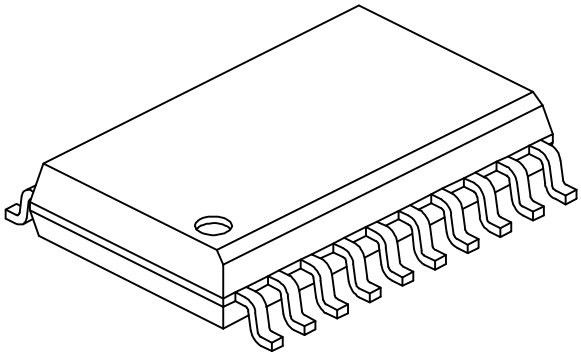



Microchip Technology Drawing C04-094 Rev D Sheet 1 of 2

## 20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Terminals | N | | 20 | |
| Pitch | e | | 1.27 BSC | |
| Overall Height | A | - | - | 2.65 |
| Standoff § | A1 | 0.10 | - | 0.30 |
| Molded Package Thickness | A2 | 2.05 | - | - |
| Overall Length | D | | 12.80 BSC | |
| Overall Width | E | | 10.30 BSC | |
| Molded Package Width | E1 | | 7.50 BSC | |
| Terminal Width | b | 0.31 | - | 0.51 |
| Terminal Thickness | c | 0.25 | - | 0.75 |
| Corner Chamfer | h | 0.25 | - | 0.41 |
| Terminal Length | L | 0.41 | 0.65 | 0.89 |
| Footprint | L1 | | 1.40 REF | |
| Lead Bend Radius | R1 | 0.07 | - | - |
| Lead Bend Radius | R2 | 0.07 | - | - |
| Foot Angle | θ | 0° | - | 8° |
| Lead Angle | θ1 | 0° | - | - |
| Mold Draft Angle | θ2 | 5° | - | 15° |
| Mold Draft Angle | θ3 | 5° | - | 15° |

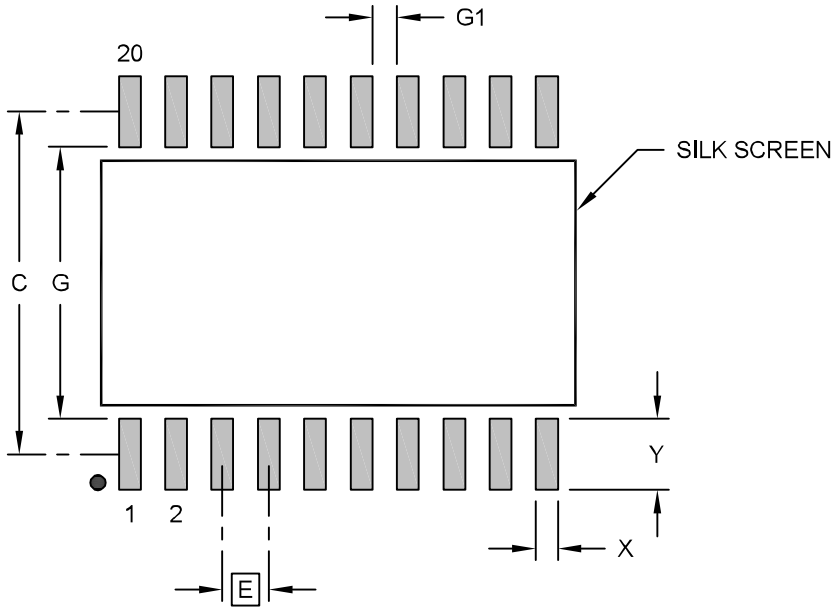
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. 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.
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. § Significant Characteristic

Microchip Technology Drawing C04-094 Rev D Sheet 2 of 2

### 20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 1.27 BSC | |
| Contact Pad Spacing | C | | 9.40 | |
| Contact Pad Width (X20) | X | | | 0.60 |
| Contact Pad Length (X20) | Y | | | 1.95 |
| Contact Pad to Contact Pad | G | 0.67 | | |
| Contact Pad to Contact Pad | G1 | 7.45 | | |

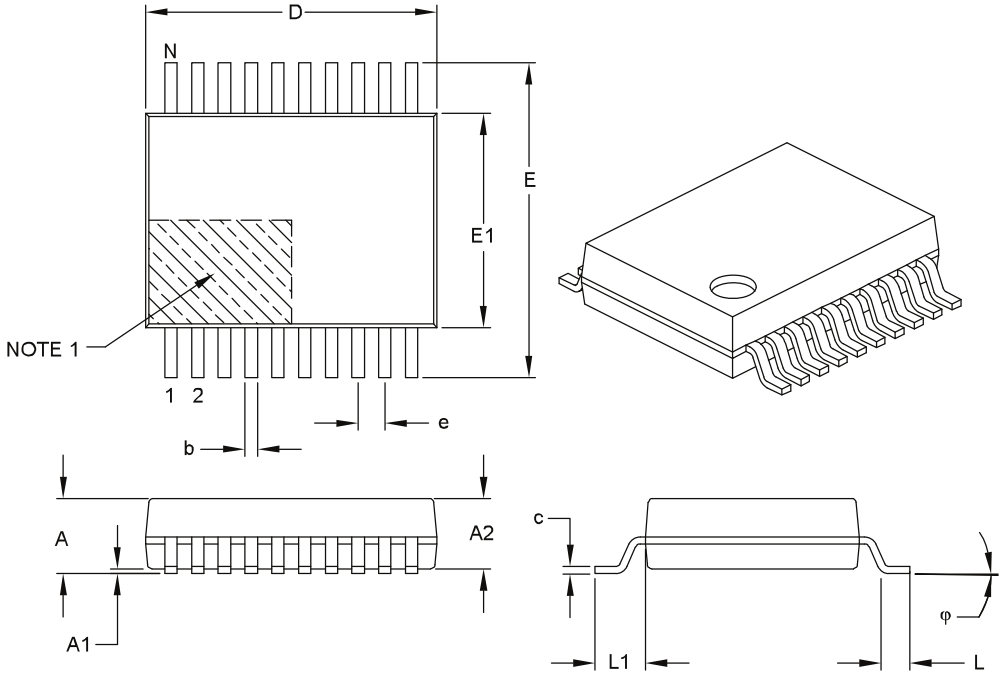
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2094 Rev D

## 20-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 20 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | – | – | 2.00 |
| Molded Package Thickness | A2 | 1.65 | 1.75 | 1.85 |
| Standoff | A1 | 0.05 | – | – |
| Overall Width | E | 7.40 | 7.80 | 8.20 |
| Molded Package Width | E1 | 5.00 | 5.30 | 5.60 |
| Overall Length | D | 6.90 | 7.20 | 7.50 |
| Foot Length | L | 0.55 | 0.75 | 0.95 |
| Footprint | L1 | | 1.25 REF | |
| Lead Thickness | c | 0.09 | – | 0.25 |
| Foot Angle | φ | 0° | 4° | 8° |
| Lead Width | b | 0.22 | – | 0.38 |

**Notes:**

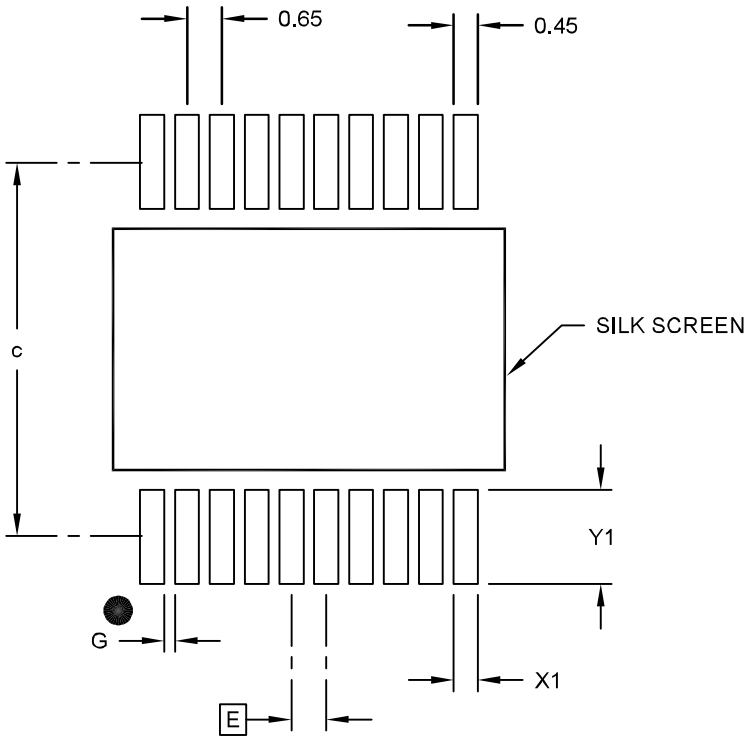
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-072B

## 20-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.65 BSC | |
| Contact Pad Spacing | C | | 7.20 | |
| Contact Pad Width (X20) | X1 | | | 0.45 |
| Contact Pad Length (X20) | Y1 | | | 1.75 |
| Distance Between Pads | G | 0.20 | | |

Notes:

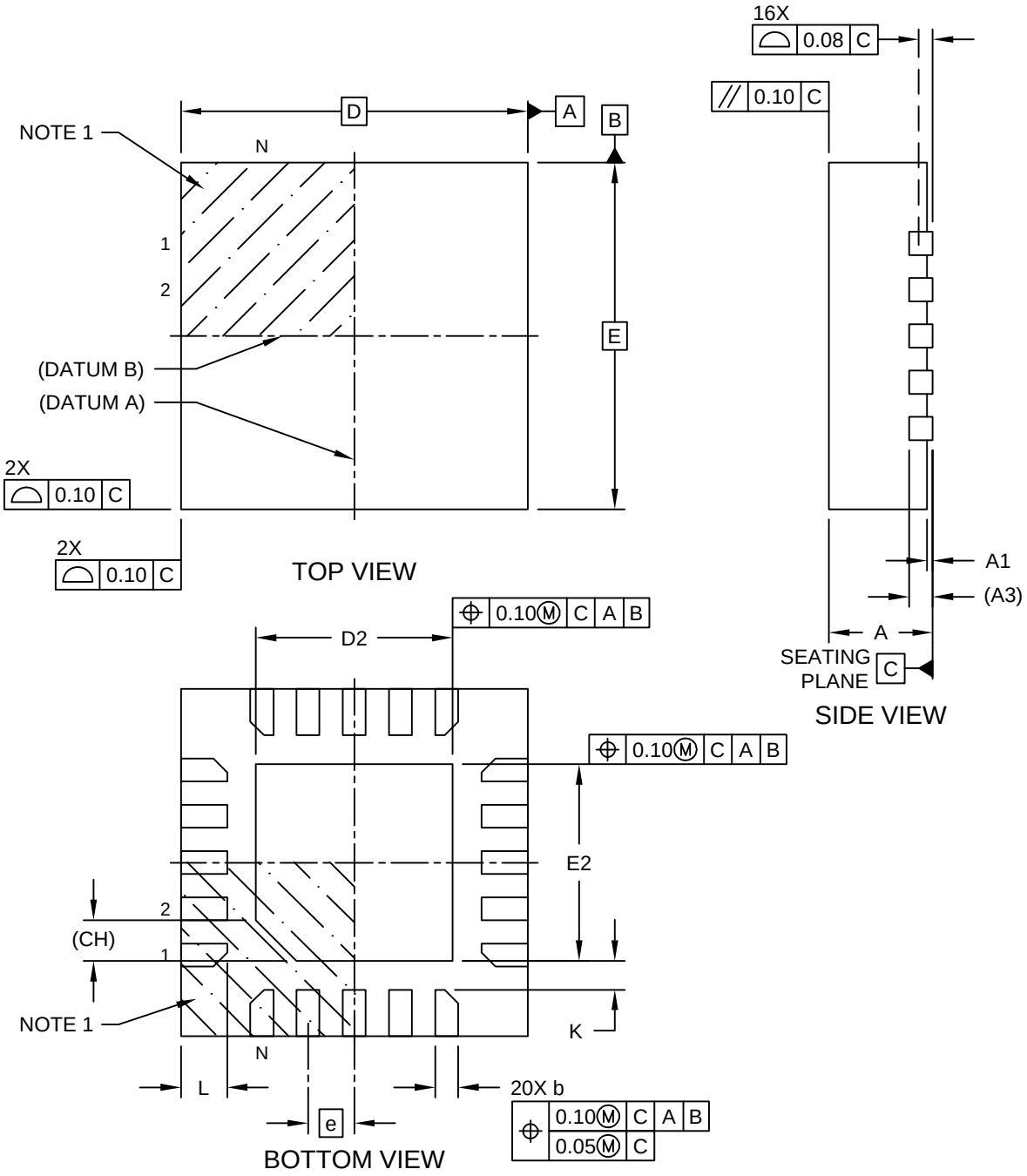
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2072B

**20-Lead Very Thin Plastic Quad Flat, No Lead Package (REB) - 3x3 mm Body [VQFN] With 1.7 mm Exposed Pad; Atmel Legacy Global Package Code ZCL**

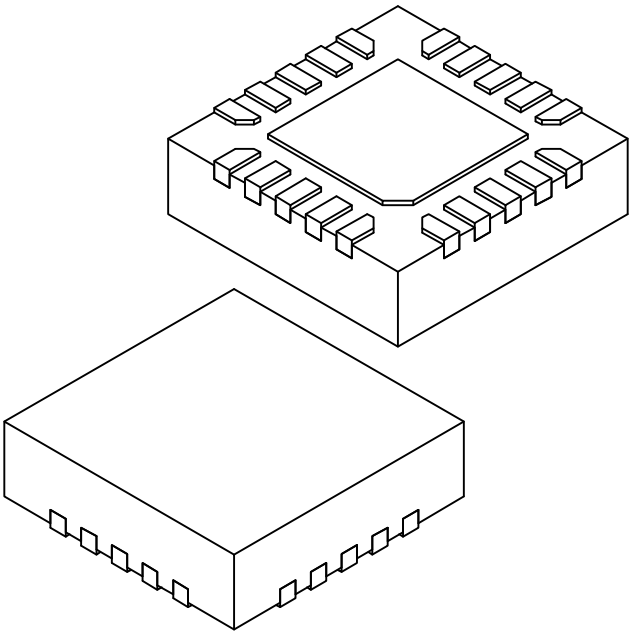
| **Note:** | For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging |
|---|---|




Microchip Technology Drawing C04-21380 Rev A Sheet 1 of 2

## 20-Lead Very Thin Plastic Quad Flat, No Lead Package (REB) - 3x3 mm Body [VQFN] With 1.7 mm Exposed Pad; Atmel Legacy Global Package Code ZCL

**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 Terminals | N | | 20 | |
| Pitch | e | | 0.40 BSC | |
| Overall Height | A | 0.80 | 0.85 | 0.90 |
| Standoff | A1 | 0.00 | 0.035 | 0.05 |
| Terminal Thickness | A3 | | 0.203 REF | |
| Overall Length | D | | 3.00 BSC | |
| Exposed Pad Length | D2 | 1.60 | 1.70 | 1.80 |
| Overall Width | E | | 3.00 BSC | |
| Exposed Pad Width | E2 | 1.60 | 1.70 | 1.80 |
| Terminal Width | b | 0.15 | 0.20 | 0.25 |
| Terminal Length | L | 0.35 | 0.40 | 0.45 |
| Terminal-to-Exposed-Pad | K | 0.20 | - | - |
| Pin 1 Index Chamfer | CH | | 0.35 REF | |

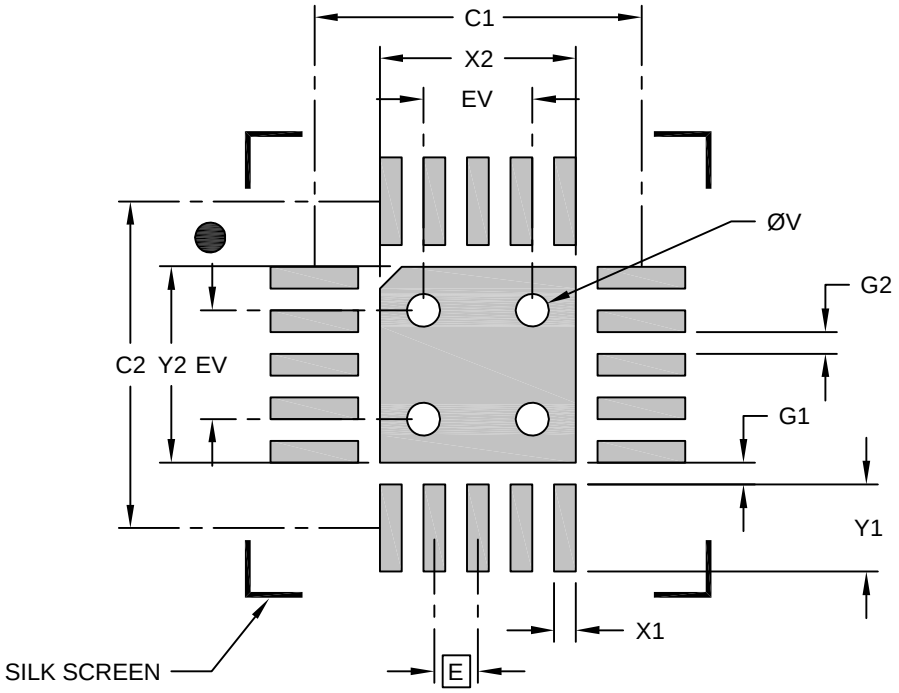
Notes:

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 C04-21380 Rev A Sheet 2 of 2

## 20-Lead Very Thin Plastic Quad Flat, No Lead Package (REB) - 3x3 mm Body [VQFN] With 1.7 mm Exposed Pad; Atmel Legacy Global Package Code ZCL

**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 | X2 | | | 1.80 |
| Optional Center Pad Length | Y2 | | | 1.80 |
| Contact Pad Spacing | C1 | | 3.00 | |
| Contact Pad Spacing | C2 | | 3.00 | |
| Contact Pad Width (X20) | X1 | | | 0.20 |
| Contact Pad Length (X20) | Y1 | | | 0.80 |
| Contact Pad to Center Pad (X20) | G1 | 0.20 | | |
| Contact Pad to Contact Pad (X16) | G2 | 0.20 | | |
| Thermal Via Diameter | V | | 0.30 | |
| Thermal Via Pitch | EV | | 1.00 | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23380 Rev A

# 35. Appendix A: Revision History

| Doc Rev. | Date | Comments |
|---|---|---|
| B | 07/2020 | Updated PWM block diagram; Updated I/O Ports registers; Updated Configuration Word 1 register in cpu chapter; Updated Memory Banks information; Other minor corrections. |
| A | 04/2020 | Initial document release. |

## The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **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 (FAQs), technical support requests, online discussion groups, Microchip design partner 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

## Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

## Customer Support

Users of Microchip products can receive assistance through several channels:

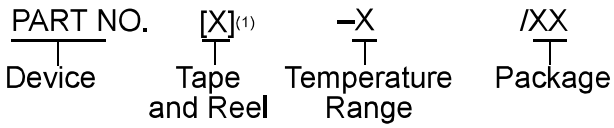
- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: 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.




| | | |
|---|---|---|
| Device: | Device A, Device B, etc | |
| Tape and Reel Option: | Blank | = Standard packaging (tube or tray) |
| | T | = Tape and Reel(1) |
| Temperature Range: | I | = -40°C to +85°C (Industrial) |
| | E | = -40°C to +125°C (Extended) |
| Package:(2) | JQ | = UQFN |
| | P | = PDIP |
| | ST | = TSSOP |
| | SL | = SOIC-14 |
| | SN | = SOIC-8 |
| | RF | = UDFN |
| Pattern: | QTP, SQTP$^{SM}$ (Serial Quick Turn Programming capability), Code or Special Requirements (blank otherwise) | |

- Device A - I/P Industrial temperature, PDIP package
- Device B - E/SS Extended temperature, SSOP package

PIS_NOTES

## Microchip Devices Code Protection Feature

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

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "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.

© 2020 Microchip Technology Inc.

## Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information 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.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". 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 ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., 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.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

© 2020, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-6649-9

## Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

# Worldwide Sales and Service

| AMERICAS | ASIA/PACIFIC | ASIA/PACIFIC | EUROPE |
|---|---|---|---|
| **Corporate Office** | **Australia - Sydney** | **India - Bangalore** | **Austria - Wels** |
| 2355 West Chandler Blvd. | Tel: 61-2-9868-6733 | Tel: 91-80-3090-4444 | Tel: 43-7242-2244-39 |
| Chandler, AZ 85224-6199 | **China - Beijing** | **India - New Delhi** | Fax: 43-7242-2244-393 |
| Tel: 480-792-7200 | Tel: 86-10-8569-7000 | Tel: 91-11-4160-8631 | **Denmark - Copenhagen** |
| Fax: 480-792-7277 | **China - Chengdu** | **India - Pune** | Tel: 45-4485-5910 |
| Technical Support: | Tel: 86-28-8665-5511 | Tel: 91-20-4121-0141 | Fax: 45-4485-2829 |
| www.microchip.com/support | **China - Chongqing** | **Japan - Osaka** | **Finland - Espoo** |
| Web Address: | Tel: 86-23-8980-9588 | Tel: 81-6-6152-7160 | Tel: 358-9-4520-820 |
| www.microchip.com | **China - Dongguan** | **Japan - Tokyo** | **France - Paris** |
| **Atlanta** | Tel: 86-769-8702-9880 | Tel: 81-3-6880- 3770 | Tel: 33-1-69-53-63-20 |
| Duluth, GA | **China - Guangzhou** | **Korea - Daegu** | Fax: 33-1-69-30-90-79 |
| Tel: 678-957-9614 | Tel: 86-20-8755-8029 | Tel: 82-53-744-4301 | **Germany - Garching** |
| Fax: 678-957-1455 | **China - Hangzhou** | **Korea - Seoul** | Tel: 49-8931-9700 |
| **Austin, TX** | Tel: 86-571-8792-8115 | Tel: 82-2-554-7200 | **Germany - Haan** |
| Tel: 512-257-3370 | **China - Hong Kong SAR** | **Malaysia - Kuala Lumpur** | Tel: 49-2129-3766400 |
| **Boston** | Tel: 852-2943-5100 | Tel: 60-3-7651-7906 | **Germany - Heilbronn** |
| Westborough, MA | **China - Nanjing** | **Malaysia - Penang** | Tel: 49-7131-72400 |
| Tel: 774-760-0087 | Tel: 86-25-8473-2460 | Tel: 60-4-227-8870 | **Germany - Karlsruhe** |
| Fax: 774-760-0088 | **China - Qingdao** | **Philippines - Manila** | Tel: 49-721-625370 |
| **Chicago** | Tel: 86-532-8502-7355 | Tel: 63-2-634-9065 | **Germany - Munich** |
| Itasca, IL | **China - Shanghai** | **Singapore** | Tel: 49-89-627-144-0 |
| Tel: 630-285-0071 | Tel: 86-21-3326-8000 | Tel: 65-6334-8870 | Fax: 49-89-627-144-44 |
| Fax: 630-285-0075 | **China - Shenyang** | **Taiwan - Hsin Chu** | **Germany - Rosenheim** |
| **Dallas** | Tel: 86-24-2334-2829 | Tel: 886-3-577-8366 | Tel: 49-8031-354-560 |
| Addison, TX | **China - Shenzhen** | **Taiwan - Kaohsiung** | **Israel - Ra'anana** |
| Tel: 972-818-7423 | Tel: 86-755-8864-2200 | Tel: 886-7-213-7830 | Tel: 972-9-744-7705 |
| Fax: 972-818-2924 | **China - Suzhou** | **Taiwan - Taipei** | **Italy - Milan** |
| **Detroit** | Tel: 86-186-6233-1526 | Tel: 886-2-2508-8600 | Tel: 39-0331-742611 |
| Novi, MI | **China - Wuhan** | **Thailand - Bangkok** | Fax: 39-0331-466781 |
| Tel: 248-848-4000 | Tel: 86-27-5980-5300 | Tel: 66-2-694-1351 | **Italy - Padova** |
| **Houston, TX** | **China - Xian** | **Vietnam - Ho Chi Minh** | Tel: 39-049-7625286 |
| Tel: 281-894-5983 | Tel: 86-29-8833-7252 | Tel: 84-28-5448-2100 | **Netherlands - Drunen** |
| **Indianapolis** | **China - Xiamen** | | Tel: 31-416-690399 |
| Noblesville, IN | Tel: 86-592-2388138 | | Fax: 31-416-690340 |
| Tel: 317-773-8323 | **China - Zhuhai** | | **Norway - Trondheim** |
| Fax: 317-773-5453 | Tel: 86-756-3210040 | | Tel: 47-72884388 |
| Tel: 317-536-2380 | | | **Poland - Warsaw** |
| **Los Angeles** | | | Tel: 48-22-3325737 |
| Mission Viejo, CA | | | **Romania - Bucharest** |
| Tel: 949-462-9523 | | | Tel: 40-21-407-87-50 |
| Fax: 949-462-9608 | | | **Spain - Madrid** |
| Tel: 951-273-7800 | | | Tel: 34-91-708-08-90 |
| **Raleigh, NC** | | | Fax: 34-91-708-08-91 |
| Tel: 919-844-7510 | | | **Sweden - Gothenberg** |
| **New York, NY** | | | Tel: 46-31-704-60-40 |
| Tel: 631-435-6000 | | | **Sweden - Stockholm** |
| **San Jose, CA** | | | Tel: 46-8-5090-4654 |
| Tel: 408-735-9110 | | | **UK - Wokingham** |
| Tel: 408-436-4270 | | | Tel: 44-118-921-5800 |
| **Canada - Toronto** | | | Fax: 44-118-921-5820 |
| Tel: 905-695-1980 | | | |
| Fax: 905-695-2078 | | | |