High-Performance RISC CPU
- C Compiler Optimized Architecture
- Only 49 Instructions
- 2K Words Linear Program Memory Addressing
- 256 bytes Linear Data Memory Addressing
- Operating Speed:
  - DC – 32 MHz clock input
  - DC – 125 ns instruction cycle
- Interrupt Capability with Automatic Context Saving
- 16-Level Deep Hardware Stack with Optional Overflow/Underflow Reset
- Direct, Indirect and Relative Addressing modes:
  - Two full 16-bit File Select Registers (FSRs)
  - FSRs can read program and data memory

Flexible Oscillator Structure
- 16 MHz Internal Oscillator Block:
  - Factory calibrated to ±1%, typical
  - Software selectable frequency range from 32 MHz to 31 kHz
- 4x Phase-Lock Loop (PLL), usable with 16 MHz internal oscillator
  - Allows 32 MHz software selectable clock frequency
- 31 kHz Low-Power Internal Oscillator
- Three External Clock modes up to 20 MHz

Special Microcontroller Features
- Operating Voltage Range:
  - 1.8V to 3.6V
- Self-Programmable under Software Control
- Power-on Reset (POR)
- Power-up Timer (PWRT)
- Programmable Low-Power Brown-Out Reset (LPBOR)
- Extended Watchdog Timer (WDT):
  - Programmable period from 1 ms to 256s
- Programmable Code Protection
- In-Circuit Serial Programming™ (ICSP™) via Two Pins
- Enhanced Low-Voltage Programming (LVP)
- Power-Saving Sleep mode:
  - Low-Power Sleep mode
  - Low-Power BOR (LPBOR)
- Integrated Temperature Indicator
- 128 Bytes High-Endurance Flash:
  - 100,000 write Flash endurance (minimum)

eXtreme Low-Power (XLP) Features
- Sleep Current:
  - 20 nA @ 1.8V, typical
- Watchdog Timer Current:
  - 200 nA @ 1.8V, typical
- Operating Current:
  - 30 µA/MHz @ 1.8V, typical

Peripheral Features
- Analog-to-Digital Converter (ADC):
  - 10-bit resolution
  - 5 external channels
  - 2 internal channels:
    - Fixed Voltage Reference
    - Temperature Indicator channel
  - Auto acquisition capability
  - Conversion available during Sleep
  - Special Event Triggers
- Hardware Capacitive Voltage Divider (CVD)
  - Double sample conversions
  - Two sets of result registers
  - Inverted acquisition
  - 7-bit pre-charge timer
  - 7-bit acquisition timer
  - Two guard ring output drives
  - Adjustable sample and hold capacitor array
- Voltage Reference module:
  - Fixed Voltage Reference (FVR) with 1.024V and 2.048V output levels
- 6 I/O Pins (1 Input-only Pin):
  - High current sink/source 25 mA/25 mA
  - Individually programmable weak pull-ups
  - Individually programmable interrupt-on-change (IOC) pins
- Timer0: 8-Bit Timer/Counter with 8-Bit Programmable Prescaler
- Master Synchronous Serial Port (MSSP) with SPI and I²C with:
  - 7-bit address masking
  - SMBus/PMBus™ compatibility
# PIC12LF1552

## TABLE 1: PIC12LF1552 Family Types

<table>
<thead>
<tr>
<th>Device</th>
<th>Data Sheet Index</th>
<th>Program Memory Flash (words)</th>
<th>Data EEPROM (bytes)</th>
<th>SRAM (bytes)</th>
<th>I/Os (1)</th>
<th>10-bit ADCs (4)</th>
<th>Analog Channels (3/9)</th>
<th>CVD RX Channels</th>
<th>CVD TX Channels (5)</th>
<th>Timers 8/16-bit</th>
<th>EUSART</th>
<th>MSSP</th>
<th>PWM</th>
<th>Debug</th>
<th>XLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC12LF1552</td>
<td>(A)</td>
<td>2048</td>
<td>0</td>
<td>256</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>1 / 0</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y</td>
</tr>
<tr>
<td>PIC16LF1554</td>
<td>(B)</td>
<td>4096</td>
<td>0</td>
<td>256</td>
<td>12</td>
<td>2</td>
<td>10</td>
<td>2 / 1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>I</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIC16LF1559</td>
<td>(B)</td>
<td>8192</td>
<td>0</td>
<td>512</td>
<td>18</td>
<td>2</td>
<td>16</td>
<td>2 / 1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>I</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIC16LF1566</td>
<td>(C)</td>
<td>8192</td>
<td>0</td>
<td>1024</td>
<td>25</td>
<td>2</td>
<td>23</td>
<td>3 / 1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>I</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIC16LF1567</td>
<td>(C)</td>
<td>8192</td>
<td>0</td>
<td>1024</td>
<td>36</td>
<td>2</td>
<td>34</td>
<td>34</td>
<td>3 / 1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>I</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** The MCLR pin is input only.

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

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

**Note 4:** If VDD > 2.4V, ADC may be overclocked 4x (TAD = 0.25 µs).

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

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

- **A:** DS40001674 PIC12LF1552 Data Sheet, 8-Pin Flash, 8-Bit Microcontrollers
- **B:** DS40001761 PIC16LF1554/1559 Data Sheet, 20-Pin Flash, 8-Bit Microcontrollers with XLP Technology
- **C:** DS40001817 PIC16LF1566/1567 Data Sheet 28/40/44-Pin Flash, 8-Bit Microcontrollers with XLP Technology

**Note:** For other small form-factor package availability and marking information, please visit [http://www.microchip.com/packaging](http://www.microchip.com/packaging) or contact your local sales office.
**FIGURE 1: 8-PIN PDIP, SOIC, MSOP, UDFN**

![Diagram of 8-PIN PDIP, SOIC, MSOP, UDFN]

**TABLE 2: 8-PIN ALLOCATION TABLE**

<table>
<thead>
<tr>
<th>IO</th>
<th>I/O-3</th>
<th>8-Pin PDIP/SOIC/MSOP/UDFN</th>
<th>I/O-2</th>
<th>ADC/Hardware CVD</th>
<th>Reference</th>
<th>Timer</th>
<th>MSSP</th>
<th>Interrupt</th>
<th>Pull-Up</th>
<th>Basic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0</td>
<td>7</td>
<td>AN0</td>
<td>—</td>
<td>—</td>
<td>SDO(1)</td>
<td>IOC</td>
<td>Y</td>
<td>ICSPDAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA1</td>
<td>6</td>
<td>AN1</td>
<td>VREF+</td>
<td>—</td>
<td>SCK</td>
<td>IOC</td>
<td>Y</td>
<td>ICSPCLK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA2</td>
<td>5</td>
<td>AN2-2</td>
<td>ADOUT</td>
<td>—</td>
<td>T0CKI</td>
<td>IOC</td>
<td>Y</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA3</td>
<td>4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SS(T)</td>
<td>IOC</td>
<td>Y</td>
<td>MCLR</td>
<td>VPP</td>
<td></td>
</tr>
<tr>
<td>RA4</td>
<td>3</td>
<td>AN3-2</td>
<td>ADGRDA</td>
<td>—</td>
<td>—</td>
<td>SDO(2)</td>
<td>IOC</td>
<td>Y</td>
<td>CLKOUT</td>
<td></td>
</tr>
<tr>
<td>RA5</td>
<td>2</td>
<td>AN4-2</td>
<td>ADGRDB</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOC</td>
<td>Y</td>
<td>CLarkin</td>
<td></td>
</tr>
<tr>
<td>VDD</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>VSS</td>
<td>8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>VSS</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** See Table 2 for location of all peripheral functions.

**Note 1:** Default location for peripheral pin function. Alternate location can be selected using the APFCON register.

**Note 2:** Alternate location for peripheral pin function selected by the APFCON register.
Table of Contents

1.0 Device Overview ................................................................. 5
2.0 Enhanced Mid-Range CPU ..................................................... 8
3.0 Memory Organization ........................................................... 10
4.0 Device Configuration ........................................................... 31
5.0 Oscillator Module ............................................................... 36
6.0 Resets .............................................................................. 44
7.0 Interrupts .......................................................................... 52
8.0 Power-down Mode (Sleep) .................................................... 62
9.0 Watchdog Timer (WDT) ....................................................... 64
10.0 Flash Program Memory Control ......................................... 68
11.0 I/O Ports ......................................................................... 84
12.0 Interrupt-on-Change .......................................................... 90
13.0 Fixed Voltage Reference (FVR) ........................................... 94
14.0 Temperature Indicator Module ........................................... 96
15.0 Analog-to-Digital Converter (ADC) Module ....................... 98
16.0 Hardware Capacitive Voltage Divider (CVD) Module ........... 111
17.0 Timer0 Module ................................................................. 130
18.0 Master Synchronous Serial Port Module ............................ 133
19.0 In-Circuit Serial Programming™ (ICSP™) ......................... 186
20.0 Instruction Set Summary ..................................................... 188
21.0 Electrical Specifications .................................................... 202
22.0 DC and AC Characteristics Graphs and Charts ................. 219
23.0 Development Support ........................................................ 223
24.0 Packaging Information ...................................................... 227

Appendix A: “Data Sheet Revision History” ............................... 239

The Microchip Web Site ......................................................... 240
Customer Change Notification Service ..................................... 240
Customer Support ................................................................... 240
Product Identification System .................................................. 241

TO OUR VALUED CUSTOMERS

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

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

Most Current Data Sheet

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

http://www.microchip.com

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

Errata

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

To determine if an errata sheet exists for a particular device, please check with one of the following:
• Microchip’s Worldwide Website: http://www.microchip.com
• Your local Microchip sales office (see last page)

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

Customer Notification System

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

The PIC12LF1552 are described within this data sheet. They are available in 8-pin packages. Figure 1-1 shows a block diagram of the PIC12LF1552 devices. Table 1-2 shows the pinout descriptions.

Reference Table 1-1 for peripherals available per device.

TABLE 1-1: DEVICE PERIPHERAL SUMMARY

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>PIC12LF1552</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog-to-Digital Converter (ADC)</td>
<td>●</td>
</tr>
<tr>
<td>Hardware Capacitor Voltage Divider (CVD)</td>
<td>●</td>
</tr>
<tr>
<td>Fixed Voltage Reference (FVR)</td>
<td>●</td>
</tr>
<tr>
<td>Temperature Indicator</td>
<td>●</td>
</tr>
<tr>
<td>Master Synchronous Serial Ports</td>
<td>MSSP1</td>
</tr>
<tr>
<td>Timers</td>
<td>Timer0</td>
</tr>
</tbody>
</table>
Note 1: See applicable chapters for more information on peripherals.
2: See Table 1-1 for peripherals available on specific devices.
### TABLE 1-2: PIC12LF1552 PINOUT DESCRIPTION

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Input Type</th>
<th>Output Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0/AN0/SDO(1)/SS(2)/ICSPDAT</td>
<td>RA0</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td></td>
<td>AN0</td>
<td>AN</td>
<td>—</td>
<td>ADC Channel input.</td>
</tr>
<tr>
<td></td>
<td>SDO</td>
<td>—</td>
<td>CMOS</td>
<td>SPI data output.</td>
</tr>
<tr>
<td></td>
<td>SS</td>
<td>ST</td>
<td>—</td>
<td>Slave Select input.</td>
</tr>
<tr>
<td></td>
<td>ICSPDAT</td>
<td>ST</td>
<td>CMOS</td>
<td>ICSP™ Data I/O.</td>
</tr>
<tr>
<td>RA1/AN1/VREF+/SCK/SCL/ICSPCLK</td>
<td>RA1</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td></td>
<td>AN1</td>
<td>AN</td>
<td>—</td>
<td>ADC Channel input.</td>
</tr>
<tr>
<td></td>
<td>VREF+</td>
<td>AN</td>
<td>—</td>
<td>ADC Positive Voltage Reference input.</td>
</tr>
<tr>
<td></td>
<td>SCK</td>
<td>ST</td>
<td>CMOS</td>
<td>SPI clock.</td>
</tr>
<tr>
<td></td>
<td>SCL</td>
<td>I^C</td>
<td>OD</td>
<td>I^C clock.</td>
</tr>
<tr>
<td></td>
<td>ICSPCLK</td>
<td>ST</td>
<td>—</td>
<td>ICSP™ Programming Clock.</td>
</tr>
<tr>
<td>RA2/AN2/ADOUT/T0CKI/SDI(1)/SDA(1)/INT</td>
<td>RA2</td>
<td>ST</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td></td>
<td>AN2</td>
<td>AN</td>
<td>—</td>
<td>ADC Channel input.</td>
</tr>
<tr>
<td></td>
<td>ADOUT</td>
<td>CMOS</td>
<td>—</td>
<td>ADC with CVD output.</td>
</tr>
<tr>
<td></td>
<td>T0CKI</td>
<td>ST</td>
<td>—</td>
<td>Timer0 clock input.</td>
</tr>
<tr>
<td></td>
<td>SDI</td>
<td>ST</td>
<td>—</td>
<td>SPI data input.</td>
</tr>
<tr>
<td></td>
<td>SDA</td>
<td>I^C</td>
<td>OD</td>
<td>I^C data input/output.</td>
</tr>
<tr>
<td></td>
<td>INT</td>
<td>ST</td>
<td>—</td>
<td>External interrupt.</td>
</tr>
<tr>
<td>RA3/MCLR/VPP/SS(1)/SDI(2)/SDA(2)</td>
<td>RA3</td>
<td>TTL</td>
<td>—</td>
<td>General purpose input.</td>
</tr>
<tr>
<td></td>
<td>MCLR</td>
<td>ST</td>
<td>—</td>
<td>Master Clear with internal pull-up.</td>
</tr>
<tr>
<td></td>
<td>VPP</td>
<td>HV</td>
<td>—</td>
<td>Programming voltage.</td>
</tr>
<tr>
<td></td>
<td>SS</td>
<td>ST</td>
<td>—</td>
<td>Slave Select input.</td>
</tr>
<tr>
<td></td>
<td>SDI</td>
<td>ST</td>
<td>—</td>
<td>SPI data input.</td>
</tr>
<tr>
<td></td>
<td>SDA</td>
<td>I^C</td>
<td>OD</td>
<td>I^C data input/output.</td>
</tr>
<tr>
<td>RA4/AN3/SDO(2)/CLKOUT/ADGRDA</td>
<td>RA4</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td></td>
<td>AN3</td>
<td>AN</td>
<td>—</td>
<td>ADC Channel input.</td>
</tr>
<tr>
<td></td>
<td>SDO</td>
<td>—</td>
<td>CMOS</td>
<td>SPI data output.</td>
</tr>
<tr>
<td></td>
<td>CLKOUT</td>
<td>—</td>
<td>CMOS</td>
<td>Fosc/4 output.</td>
</tr>
<tr>
<td></td>
<td>ADGRDA</td>
<td>—</td>
<td>CMOS</td>
<td>Guard ring output A.</td>
</tr>
<tr>
<td>RA5/AN4/CLKIN/ADGRDB</td>
<td>RA5</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td></td>
<td>AN4</td>
<td>AN</td>
<td>—</td>
<td>ADC Channel input.</td>
</tr>
<tr>
<td></td>
<td>CLKIN</td>
<td>CMOS</td>
<td>—</td>
<td>External clock input (EC mode).</td>
</tr>
<tr>
<td></td>
<td>ADGRDB</td>
<td>—</td>
<td>CMOS</td>
<td>Guard ring output B.</td>
</tr>
<tr>
<td>Vdd</td>
<td>Vdd</td>
<td>Power</td>
<td>—</td>
<td>Positive supply.</td>
</tr>
<tr>
<td>Vss</td>
<td>Vss</td>
<td>Power</td>
<td>—</td>
<td>Ground reference.</td>
</tr>
</tbody>
</table>

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

**Note 1:** Default location for peripheral pin function. Alternate location can be selected using the APFCON register.

**Note 2:** Alternate location for peripheral pin function selected by the APFCON register.
2.0 ENHANCED MID-RANGE CPU

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

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

FIGURE 2-1: CORE BLOCK DIAGRAM
2.1 Automatic Interrupt Context Saving

During interrupts, certain registers are automatically saved in shadow registers and restored when returning from the interrupt. This saves stack space and user code. See Section 7.5 “Automatic Context Saving”, for more information.

2.2 16-Level Stack with Overflow and Underflow

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

2.3 File Select Registers

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

2.4 Instruction Set

There are 49 instructions for the enhanced mid-range CPU to support the features of the CPU. See Section 20.0 “Instruction Set Summary” for more details.
3.0 MEMORY ORGANIZATION

These devices contain the following types of memory:

- Program Memory
  - Configuration Words
  - Device ID
  - User ID
  - Flash Program Memory
- Data Memory
  - Core Registers
  - Special Function Registers
  - General Purpose RAM
  - Common RAM

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

- PCL and PCLATH
- Stack
- Indirect Addressing

3.1 Program Memory Organization

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

<table>
<thead>
<tr>
<th>Device</th>
<th>Program Memory Space (Words)</th>
<th>Last Program Memory Address</th>
<th>High-Endurance Flash Memory Address Range (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC12LF1552</td>
<td>2,048</td>
<td>07FFh</td>
<td>0780h-07FFh</td>
</tr>
</tbody>
</table>

Note 1: High-endurance Flash applies to the low byte of each address in the range.
3.1.1 READING PROGRAM MEMORY AS DATA

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

3.1.1.1 RETLW Instruction

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

EXAMPLE 3-1: RETLW INSTRUCTION

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

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

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

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

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

Example 3-2: Accessing Program Memory via FSR

```
; constants
DW DATA0 ; First constant
DW DATA1 ; Second constant
DW DATA2
DW DATA3

my_function
;... LOTS OF CODE...
MOVLW DATA_INDEX
ADDLW LOW constants
MOVWF FSR1L
MOVWI HIGH constants; MSb is set automatically
MOVWF FSR1H
BTFSC STATUS,C ; carry from ADDLW?
INCF FSR1H,f ; yes
MOVWI 0[FSR1]
; THE PROGRAM MEMORY IS IN W
```

3.2 Data Memory Organization

The data memory is partitioned into 32 memory banks with 128 bytes in a bank. Each bank consists of (Figure 3-2):

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

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

<table>
<thead>
<tr>
<th>Addresses</th>
<th>BANKx</th>
</tr>
</thead>
<tbody>
<tr>
<td>x00h or x80h</td>
<td>INDF0</td>
</tr>
<tr>
<td>x01h or x81h</td>
<td>INDF1</td>
</tr>
<tr>
<td>x02h or x82h</td>
<td>PCL</td>
</tr>
<tr>
<td>x03h or x83h</td>
<td>STATUS</td>
</tr>
<tr>
<td>x04h or x84h</td>
<td>FSR0L</td>
</tr>
<tr>
<td>x05h or x85h</td>
<td>FSR0H</td>
</tr>
<tr>
<td>x06h or x86h</td>
<td>FSR1L</td>
</tr>
<tr>
<td>x07h or x87h</td>
<td>FSR1H</td>
</tr>
<tr>
<td>x08h or x88h</td>
<td>BSR</td>
</tr>
<tr>
<td>x09h or x89h</td>
<td>WREG</td>
</tr>
<tr>
<td>x0Ah or x8Ah</td>
<td>PCLATH</td>
</tr>
<tr>
<td>x0Bh or x8Bh</td>
<td>INTCON</td>
</tr>
</tbody>
</table>
3.2.1.1 STATUS Register
The STATUS register, shown in Register 3-1, contains:
• the arithmetic status of the ALU
• the Reset status
The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the TO and PD bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, CLRF STATUS will clear the upper three bits and set the Z bit. This leaves the STATUS register as '000u 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 Section 20.0 “Instruction Set Summary”).

Note 1: The C and DC bits operate as Borrow and Digit Borrow out bits, respectively, in subtraction.

3.3 Register Definitions: Status

REGISTER 3-1: STATUS: STATUS REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-1/q</th>
<th>R-1/q</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
<th>R/W-0/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TO</td>
<td>PD</td>
<td>Z</td>
<td>DC(1)</td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-5  Unimplemented: Read as ‘0’
bit 4    TO: Time-Out bit
  1 = After power-up, CLRWDT instruction or SLEEP instruction
  0 = A WDT time-out occurred
bit 3    PD: Power-Down bit
  1 = After power-up or by the CLRWDT instruction
  0 = By execution of the SLEEP instruction
bit 2    Z: Zero bit
  1 = The result of an arithmetic or logic operation is zero
  0 = The result of an arithmetic or logic operation is not zero
bit 1    DC: Digit Carry/Digit Borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
  1 = A carry-out from the 4th low-order bit of the result occurred
  0 = No carry-out from the 4th low-order bit of the result
bit 0    C: Carry/Borrow bit(1) (ADDWF, ADDLW, SUBLW, SUBWF instructions)(1)
  1 = A carry-out from the Most Significant bit of the result occurred
  0 = No carry-out from the Most Significant bit of the result occurred

Note 1: For Borrow, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high-order or low-order bit of the source register.
3.3.1 SPECIAL FUNCTION REGISTER
The Special Function Registers are registers used by the application to control the desired operation of peripheral functions in the device. The Special Function Registers occupy the 20 bytes after the core registers of every data memory bank (addresses x0Ch/x8Ch through x1Fh/x9Fh). The registers associated with the operation of the peripherals are described in the appropriate peripheral chapter of this data sheet.

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

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

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

3.3.4 DEVICE MEMORY MAPS
The memory maps for PIC12LF1552 are as shown in Table 3-3.
<table>
<thead>
<tr>
<th>BANK 0</th>
<th>BANK 1</th>
<th>BANK 2</th>
<th>BANK 3</th>
<th>BANK 4</th>
<th>BANK 5</th>
<th>BANK 6</th>
<th>BANK 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>000h</td>
<td>Core Registers (Table 3-2)</td>
<td>0100h</td>
<td>100h</td>
<td>200h</td>
<td>300h</td>
<td>400h</td>
<td>500h</td>
</tr>
<tr>
<td>080h</td>
<td>Core Registers (Table 3-2)</td>
<td>0100h</td>
<td>100h</td>
<td>200h</td>
<td>300h</td>
<td>400h</td>
<td>500h</td>
</tr>
<tr>
<td>00Bh</td>
<td>PORTA</td>
<td>0180h</td>
<td>1080h</td>
<td>2080h</td>
<td>3080h</td>
<td>4080h</td>
<td>5080h</td>
</tr>
<tr>
<td>08Bh</td>
<td>TRISA</td>
<td>0180h</td>
<td>1080h</td>
<td>2080h</td>
<td>3080h</td>
<td>4080h</td>
<td>5080h</td>
</tr>
<tr>
<td>100h</td>
<td></td>
<td>0180h</td>
<td>1080h</td>
<td>2080h</td>
<td>3080h</td>
<td>4080h</td>
<td>5080h</td>
</tr>
<tr>
<td>180h</td>
<td></td>
<td>0180h</td>
<td>1080h</td>
<td>2080h</td>
<td>3080h</td>
<td>4080h</td>
<td>5080h</td>
</tr>
<tr>
<td>200h</td>
<td></td>
<td>0180h</td>
<td>1080h</td>
<td>2080h</td>
<td>3080h</td>
<td>4080h</td>
<td>5080h</td>
</tr>
<tr>
<td>280h</td>
<td></td>
<td>0180h</td>
<td>1080h</td>
<td>2080h</td>
<td>3080h</td>
<td>4080h</td>
<td>5080h</td>
</tr>
<tr>
<td>300h</td>
<td></td>
<td>0180h</td>
<td>1080h</td>
<td>2080h</td>
<td>3080h</td>
<td>4080h</td>
<td>5080h</td>
</tr>
<tr>
<td>380h</td>
<td></td>
<td>0180h</td>
<td>1080h</td>
<td>2080h</td>
<td>3080h</td>
<td>4080h</td>
<td>5080h</td>
</tr>
<tr>
<td>00Bh</td>
<td>OPTION_REG</td>
<td>0115h</td>
<td>1015h</td>
<td>2015h</td>
<td>3015h</td>
<td>4015h</td>
<td>5015h</td>
</tr>
<tr>
<td>08Bh</td>
<td></td>
<td>0115h</td>
<td>1015h</td>
<td>2015h</td>
<td>3015h</td>
<td>4015h</td>
<td>5015h</td>
</tr>
<tr>
<td>100h</td>
<td></td>
<td>0115h</td>
<td>1015h</td>
<td>2015h</td>
<td>3015h</td>
<td>4015h</td>
<td>5015h</td>
</tr>
<tr>
<td>180h</td>
<td></td>
<td>0115h</td>
<td>1015h</td>
<td>2015h</td>
<td>3015h</td>
<td>4015h</td>
<td>5015h</td>
</tr>
<tr>
<td>200h</td>
<td></td>
<td>0115h</td>
<td>1015h</td>
<td>2015h</td>
<td>3015h</td>
<td>4015h</td>
<td>5015h</td>
</tr>
<tr>
<td>280h</td>
<td></td>
<td>0115h</td>
<td>1015h</td>
<td>2015h</td>
<td>3015h</td>
<td>4015h</td>
<td>5015h</td>
</tr>
<tr>
<td>300h</td>
<td></td>
<td>0115h</td>
<td>1015h</td>
<td>2015h</td>
<td>3015h</td>
<td>4015h</td>
<td>5015h</td>
</tr>
<tr>
<td>380h</td>
<td></td>
<td>0115h</td>
<td>1015h</td>
<td>2015h</td>
<td>3015h</td>
<td>4015h</td>
<td>5015h</td>
</tr>
<tr>
<td>00Ch</td>
<td>PORTA</td>
<td>0111h</td>
<td>1011h</td>
<td>2011h</td>
<td>3011h</td>
<td>4011h</td>
<td>5011h</td>
</tr>
<tr>
<td>08Ch</td>
<td>TRISA</td>
<td>0111h</td>
<td>1011h</td>
<td>2011h</td>
<td>3011h</td>
<td>4011h</td>
<td>5011h</td>
</tr>
<tr>
<td>10Ch</td>
<td></td>
<td>0111h</td>
<td>1011h</td>
<td>2011h</td>
<td>3011h</td>
<td>4011h</td>
<td>5011h</td>
</tr>
<tr>
<td>18Ch</td>
<td></td>
<td>0111h</td>
<td>1011h</td>
<td>2011h</td>
<td>3011h</td>
<td>4011h</td>
<td>5011h</td>
</tr>
<tr>
<td>20Ch</td>
<td></td>
<td>0111h</td>
<td>1011h</td>
<td>2011h</td>
<td>3011h</td>
<td>4011h</td>
<td>5011h</td>
</tr>
<tr>
<td>28Ch</td>
<td></td>
<td>0111h</td>
<td>1011h</td>
<td>2011h</td>
<td>3011h</td>
<td>4011h</td>
<td>5011h</td>
</tr>
<tr>
<td>30Ch</td>
<td></td>
<td>0111h</td>
<td>1011h</td>
<td>2011h</td>
<td>3011h</td>
<td>4011h</td>
<td>5011h</td>
</tr>
<tr>
<td>38Ch</td>
<td></td>
<td>0111h</td>
<td>1011h</td>
<td>2011h</td>
<td>3011h</td>
<td>4011h</td>
<td>5011h</td>
</tr>
<tr>
<td>011h</td>
<td>PORTA</td>
<td>0112h</td>
<td>1012h</td>
<td>2012h</td>
<td>3012h</td>
<td>4012h</td>
<td>5012h</td>
</tr>
<tr>
<td>091h</td>
<td>PIE1</td>
<td>0112h</td>
<td>1012h</td>
<td>2012h</td>
<td>3012h</td>
<td>4012h</td>
<td>5012h</td>
</tr>
<tr>
<td>111h</td>
<td></td>
<td>0112h</td>
<td>1012h</td>
<td>2012h</td>
<td>3012h</td>
<td>4012h</td>
<td>5012h</td>
</tr>
<tr>
<td>191h</td>
<td></td>
<td>0112h</td>
<td>1012h</td>
<td>2012h</td>
<td>3012h</td>
<td>4012h</td>
<td>5012h</td>
</tr>
<tr>
<td>211h</td>
<td></td>
<td>0112h</td>
<td>1012h</td>
<td>2012h</td>
<td>3012h</td>
<td>4012h</td>
<td>5012h</td>
</tr>
<tr>
<td>291h</td>
<td></td>
<td>0112h</td>
<td>1012h</td>
<td>2012h</td>
<td>3012h</td>
<td>4012h</td>
<td>5012h</td>
</tr>
<tr>
<td>311h</td>
<td></td>
<td>0112h</td>
<td>1012h</td>
<td>2012h</td>
<td>3012h</td>
<td>4012h</td>
<td>5012h</td>
</tr>
<tr>
<td>391h</td>
<td></td>
<td>0112h</td>
<td>1012h</td>
<td>2012h</td>
<td>3012h</td>
<td>4012h</td>
<td>5012h</td>
</tr>
<tr>
<td>012h</td>
<td>PORTA</td>
<td>0113h</td>
<td>1013h</td>
<td>2013h</td>
<td>3013h</td>
<td>4013h</td>
<td>5013h</td>
</tr>
<tr>
<td>092h</td>
<td>PIE2</td>
<td>0113h</td>
<td>1013h</td>
<td>2013h</td>
<td>3013h</td>
<td>4013h</td>
<td>5013h</td>
</tr>
<tr>
<td>113h</td>
<td></td>
<td>0113h</td>
<td>1013h</td>
<td>2013h</td>
<td>3013h</td>
<td>4013h</td>
<td>5013h</td>
</tr>
<tr>
<td>193h</td>
<td></td>
<td>0113h</td>
<td>1013h</td>
<td>2013h</td>
<td>3013h</td>
<td>4013h</td>
<td>5013h</td>
</tr>
<tr>
<td>213h</td>
<td></td>
<td>0113h</td>
<td>1013h</td>
<td>2013h</td>
<td>3013h</td>
<td>4013h</td>
<td>5013h</td>
</tr>
<tr>
<td>293h</td>
<td></td>
<td>0113h</td>
<td>1013h</td>
<td>2013h</td>
<td>3013h</td>
<td>4013h</td>
<td>5013h</td>
</tr>
<tr>
<td>313h</td>
<td></td>
<td>0113h</td>
<td>1013h</td>
<td>2013h</td>
<td>3013h</td>
<td>4013h</td>
<td>5013h</td>
</tr>
<tr>
<td>393h</td>
<td></td>
<td>0113h</td>
<td>1013h</td>
<td>2013h</td>
<td>3013h</td>
<td>4013h</td>
<td>5013h</td>
</tr>
</tbody>
</table>

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

Note 1: These ADC registers are the same as the registers in Bank 14.
### TABLE 3-3: PIC12LF1552 MEMORY MAP (CONTINUED)

<table>
<thead>
<tr>
<th>BANK 8</th>
<th>BANK 9</th>
<th>BANK 10</th>
<th>BANK 11</th>
<th>BANK 12</th>
<th>BANK 13</th>
<th>BANK 14</th>
<th>BANK 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>400h</td>
<td>408h</td>
<td>500h</td>
<td>508h</td>
<td>600h</td>
<td>608h</td>
<td>700h</td>
<td>708h</td>
</tr>
<tr>
<td>40Bh</td>
<td>40Ch</td>
<td>50Bh</td>
<td>50Ch</td>
<td>60Bh</td>
<td>60Ch</td>
<td>70Bh</td>
<td>70Ch</td>
</tr>
<tr>
<td>40Dh</td>
<td>40Eh</td>
<td>50Dh</td>
<td>50Eh</td>
<td>60Dh</td>
<td>60Eh</td>
<td>70Dh</td>
<td>70Eh</td>
</tr>
<tr>
<td>410h</td>
<td>411h</td>
<td>510h</td>
<td>511h</td>
<td>610h</td>
<td>611h</td>
<td>710h</td>
<td>71Ch</td>
</tr>
<tr>
<td>412h</td>
<td>413h</td>
<td>512h</td>
<td>513h</td>
<td>612h</td>
<td>613h</td>
<td>713h</td>
<td>71Ah</td>
</tr>
<tr>
<td>414h</td>
<td>415h</td>
<td>514h</td>
<td>515h</td>
<td>614h</td>
<td>615h</td>
<td>715h</td>
<td>71Bh</td>
</tr>
<tr>
<td>416h</td>
<td>417h</td>
<td>516h</td>
<td>517h</td>
<td>616h</td>
<td>617h</td>
<td>717h</td>
<td>71Ch</td>
</tr>
<tr>
<td>418h</td>
<td>419h</td>
<td>518h</td>
<td>519h</td>
<td>618h</td>
<td>619h</td>
<td>719h</td>
<td>71Ah</td>
</tr>
<tr>
<td>41Ah</td>
<td>41Bh</td>
<td>51Ah</td>
<td>51Bh</td>
<td>61Ah</td>
<td>61Bh</td>
<td>71Bh</td>
<td>71Ch</td>
</tr>
<tr>
<td>41Ch</td>
<td>41Dh</td>
<td>51Ch</td>
<td>51Dh</td>
<td>61Ch</td>
<td>61Dh</td>
<td>71Dh</td>
<td>71Ch</td>
</tr>
<tr>
<td>41Eh</td>
<td>41Fh</td>
<td>51Eh</td>
<td>51Fh</td>
<td>61Eh</td>
<td>61Fh</td>
<td>71Fh</td>
<td>71Ch</td>
</tr>
<tr>
<td>420h</td>
<td>440h</td>
<td>520h</td>
<td>540h</td>
<td>620h</td>
<td>640h</td>
<td>720h</td>
<td>740h</td>
</tr>
<tr>
<td>46Fh</td>
<td>4EFh</td>
<td>56Fh</td>
<td>5EFh</td>
<td>66Fh</td>
<td>6EFh</td>
<td>76Fh</td>
<td>7EFh</td>
</tr>
<tr>
<td>470h</td>
<td>4FFh</td>
<td>57Fh</td>
<td>5FFh</td>
<td>67Fh</td>
<td>6FFh</td>
<td>77Fh</td>
<td>7FFh</td>
</tr>
<tr>
<td>47Fh</td>
<td>4FFh</td>
<td>57Fh</td>
<td>5FFh</td>
<td>67Fh</td>
<td>6FFh</td>
<td>77Fh</td>
<td>7FFh</td>
</tr>
</tbody>
</table>

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

**Note 1:** These ADC registers are the same as the registers in Bank 1.
<table>
<thead>
<tr>
<th>BANK 24</th>
<th>BANK 25</th>
<th>BANK 26</th>
<th>BANK 27</th>
<th>BANK 28</th>
<th>BANK 29</th>
<th>BANK 30</th>
<th>BANK 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>C00h</td>
<td>C08h</td>
<td>D00h</td>
<td>D08h</td>
<td>E00h</td>
<td>E08h</td>
<td>F00h</td>
<td>F08h</td>
</tr>
<tr>
<td>C0Bh</td>
<td>C0Ch</td>
<td>D0Bh</td>
<td>D0Ch</td>
<td>E0Bh</td>
<td>E0Ch</td>
<td>F0Bh</td>
<td>F0Ch</td>
</tr>
<tr>
<td>C0Dh</td>
<td>C0Eh</td>
<td>D0Dh</td>
<td>D0Eh</td>
<td>E0Dh</td>
<td>E0Eh</td>
<td>F0Dh</td>
<td>F0Eh</td>
</tr>
<tr>
<td>C0Fh</td>
<td>C0Gh</td>
<td>D0Fh</td>
<td>D0Gh</td>
<td>E0Fh</td>
<td>E0Gh</td>
<td>F0Fh</td>
<td>F0Gh</td>
</tr>
<tr>
<td>C0Ah</td>
<td>C0Bh</td>
<td>D0Ah</td>
<td>D0Bh</td>
<td>E0Ah</td>
<td>E0Bh</td>
<td>F0Ah</td>
<td>F0Bh</td>
</tr>
<tr>
<td>C0Ch</td>
<td>C0Dh</td>
<td>D0Ch</td>
<td>D0Dh</td>
<td>E0Ch</td>
<td>E0Dh</td>
<td>F0Ch</td>
<td>F0Dh</td>
</tr>
<tr>
<td>C0Eh</td>
<td>C0Fh</td>
<td>D0Eh</td>
<td>D0Fh</td>
<td>E0Eh</td>
<td>E0Fh</td>
<td>F0Eh</td>
<td>F0Fh</td>
</tr>
<tr>
<td>C0Gh</td>
<td>C0Ah</td>
<td>D0Gh</td>
<td>D0Ah</td>
<td>E0Gh</td>
<td>E0Ah</td>
<td>F0Gh</td>
<td>F0Ah</td>
</tr>
<tr>
<td>C0Bh</td>
<td>C0Ch</td>
<td>D0Bh</td>
<td>D0Ch</td>
<td>E0Bh</td>
<td>E0Ch</td>
<td>F0Bh</td>
<td>F0Ch</td>
</tr>
<tr>
<td>C0Dh</td>
<td>C0Eh</td>
<td>D0Dh</td>
<td>D0Eh</td>
<td>E0Dh</td>
<td>E0Eh</td>
<td>F0Dh</td>
<td>F0Eh</td>
</tr>
<tr>
<td>C0Fh</td>
<td>C0Gh</td>
<td>D0Fh</td>
<td>D0Gh</td>
<td>E0Fh</td>
<td>E0Gh</td>
<td>F0Fh</td>
<td>F0Gh</td>
</tr>
<tr>
<td>C0Ah</td>
<td>C0Bh</td>
<td>D0Ah</td>
<td>D0Bh</td>
<td>E0Ah</td>
<td>E0Bh</td>
<td>F0Ah</td>
<td>F0Bh</td>
</tr>
<tr>
<td>C0Ch</td>
<td>C0Dh</td>
<td>D0Ch</td>
<td>D0Dh</td>
<td>E0Ch</td>
<td>E0Dh</td>
<td>F0Ch</td>
<td>F0Dh</td>
</tr>
<tr>
<td>C0Eh</td>
<td>C0Fh</td>
<td>D0Eh</td>
<td>D0Fh</td>
<td>E0Eh</td>
<td>E0Fh</td>
<td>F0Eh</td>
<td>F0Fh</td>
</tr>
<tr>
<td>C0Gh</td>
<td>C0Ah</td>
<td>D0Gh</td>
<td>D0Ah</td>
<td>E0Gh</td>
<td>E0Ah</td>
<td>F0Gh</td>
<td>F0Ah</td>
</tr>
<tr>
<td>C0Bh</td>
<td>C0Ch</td>
<td>D0Bh</td>
<td>D0Ch</td>
<td>E0Bh</td>
<td>E0Ch</td>
<td>F0Bh</td>
<td>F0Ch</td>
</tr>
<tr>
<td>C0Dh</td>
<td>C0Eh</td>
<td>D0Dh</td>
<td>D0Eh</td>
<td>E0Dh</td>
<td>E0Eh</td>
<td>F0Dh</td>
<td>F0Eh</td>
</tr>
<tr>
<td>C0Fh</td>
<td>C0Gh</td>
<td>D0Fh</td>
<td>D0Gh</td>
<td>E0Fh</td>
<td>E0Gh</td>
<td>F0Fh</td>
<td>F0Gh</td>
</tr>
<tr>
<td>C0Ah</td>
<td>C0Bh</td>
<td>D0Ah</td>
<td>D0Bh</td>
<td>E0Ah</td>
<td>E0Bh</td>
<td>F0Ah</td>
<td>F0Bh</td>
</tr>
<tr>
<td>C0Ch</td>
<td>C0Dh</td>
<td>D0Ch</td>
<td>D0Dh</td>
<td>E0Ch</td>
<td>E0Dh</td>
<td>F0Ch</td>
<td>F0Dh</td>
</tr>
<tr>
<td>C0Eh</td>
<td>C0Fh</td>
<td>D0Eh</td>
<td>D0Fh</td>
<td>E0Eh</td>
<td>E0Fh</td>
<td>F0Eh</td>
<td>F0Fh</td>
</tr>
<tr>
<td>C0Gh</td>
<td>C0Ah</td>
<td>D0Gh</td>
<td>D0Ah</td>
<td>E0Gh</td>
<td>E0Ah</td>
<td>F0Gh</td>
<td>F0Ah</td>
</tr>
<tr>
<td>C0Bh</td>
<td>C0Ch</td>
<td>D0Bh</td>
<td>D0Ch</td>
<td>E0Bh</td>
<td>E0Ch</td>
<td>F0Bh</td>
<td>F0Ch</td>
</tr>
<tr>
<td>C0Dh</td>
<td>C0Eh</td>
<td>D0Dh</td>
<td>D0Eh</td>
<td>E0Dh</td>
<td>E0Eh</td>
<td>F0Dh</td>
<td>F0Eh</td>
</tr>
<tr>
<td>C0Fh</td>
<td>C0Gh</td>
<td>D0Fh</td>
<td>D0Gh</td>
<td>E0Fh</td>
<td>E0Gh</td>
<td>F0Fh</td>
<td>F0Gh</td>
</tr>
<tr>
<td>C0Ah</td>
<td>C0Bh</td>
<td>D0Ah</td>
<td>D0Bh</td>
<td>E0Ah</td>
<td>E0Bh</td>
<td>F0Ah</td>
<td>F0Bh</td>
</tr>
<tr>
<td>C0Ch</td>
<td>C0Dh</td>
<td>D0Ch</td>
<td>D0Dh</td>
<td>E0Ch</td>
<td>E0Dh</td>
<td>F0Ch</td>
<td>F0Dh</td>
</tr>
<tr>
<td>C0Eh</td>
<td>C0Fh</td>
<td>D0Eh</td>
<td>D0Fh</td>
<td>E0Eh</td>
<td>E0Fh</td>
<td>F0Eh</td>
<td>F0Fh</td>
</tr>
<tr>
<td>C0Gh</td>
<td>C0Ah</td>
<td>D0Gh</td>
<td>D0Ah</td>
<td>E0Gh</td>
<td>E0Ah</td>
<td>F0Gh</td>
<td>F0Ah</td>
</tr>
<tr>
<td>C0Bh</td>
<td>C0Ch</td>
<td>D0Bh</td>
<td>D0Ch</td>
<td>E0Bh</td>
<td>E0Ch</td>
<td>F0Bh</td>
<td>F0Ch</td>
</tr>
<tr>
<td>C0Dh</td>
<td>C0Eh</td>
<td>D0Dh</td>
<td>D0Eh</td>
<td>E0Dh</td>
<td>E0Eh</td>
<td>F0Dh</td>
<td>F0Eh</td>
</tr>
<tr>
<td>C0Fh</td>
<td>C0Gh</td>
<td>D0Fh</td>
<td>D0Gh</td>
<td>E0Fh</td>
<td>E0Gh</td>
<td>F0Fh</td>
<td>F0Gh</td>
</tr>
<tr>
<td>C0Ah</td>
<td>C0Bh</td>
<td>D0Ah</td>
<td>D0Bh</td>
<td>E0Ah</td>
<td>E0Bh</td>
<td>F0Ah</td>
<td>F0Bh</td>
</tr>
<tr>
<td>C0Ch</td>
<td>C0Dh</td>
<td>D0Ch</td>
<td>D0Dh</td>
<td>E0Ch</td>
<td>E0Dh</td>
<td>F0Ch</td>
<td>F0Dh</td>
</tr>
<tr>
<td>C0Eh</td>
<td>C0Fh</td>
<td>D0Eh</td>
<td>D0Fh</td>
<td>E0Eh</td>
<td>E0Fh</td>
<td>F0Eh</td>
<td>F0Fh</td>
</tr>
<tr>
<td>C0Gh</td>
<td>C0Ah</td>
<td>D0Gh</td>
<td>D0Ah</td>
<td>E0Gh</td>
<td>E0Ah</td>
<td>F0Gh</td>
<td>F0Ah</td>
</tr>
</tbody>
</table>

Legend: = Unimplemented data memory locations, read as ‘0’.

Note 1: These ADC registers are the same as the registers in Bank 1.
<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F8Ch</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>FE3h</td>
<td>STATUS_SHAD</td>
</tr>
<tr>
<td>FE4h</td>
<td>WREG_SHAD</td>
</tr>
<tr>
<td>FE5h</td>
<td>BSR_SHAD</td>
</tr>
<tr>
<td>FE6h</td>
<td>PCLATH_SHAD</td>
</tr>
<tr>
<td>FE7h</td>
<td>FSR0L_SHAD</td>
</tr>
<tr>
<td>FE8h</td>
<td>FSR0H_SHAD</td>
</tr>
<tr>
<td>FE9h</td>
<td>FSR1L_SHAD</td>
</tr>
<tr>
<td>FEAh</td>
<td>FSR1H_SHAD</td>
</tr>
<tr>
<td>FEBh</td>
<td></td>
</tr>
<tr>
<td>FEDh</td>
<td>STKPTR</td>
</tr>
<tr>
<td>FEEh</td>
<td>TOSL</td>
</tr>
<tr>
<td>FEFh</td>
<td>TOSH</td>
</tr>
</tbody>
</table>

Legend: = Unimplemented data memory locations, read as '0'.
### 3.3.5 CORE FUNCTION REGISTERS

**SUMMARY**

The Core Function registers listed in Table 3-5 can be addressed from any Bank.

#### TABLE 3-5: CORE FUNCTION REGISTERS SUMMARY

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Value on all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>x00h or x80h</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register)</td>
<td>xxxx xxxx</td>
</tr>
<tr>
<td>x01h or x81h</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register)</td>
<td>xxxx xxxx</td>
</tr>
<tr>
<td>x02h or x82h</td>
<td>PCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Program Counter (PC) Least Significant Byte</td>
<td>0000 0000</td>
</tr>
<tr>
<td>x03h or x83h</td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>----1 1000</td>
<td>----q quuu</td>
</tr>
<tr>
<td>x04h or x84h</td>
<td>FSR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address 0 Low Pointer</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>x05h or x85h</td>
<td>FSR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address 0 High Pointer</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>x06h or x86h</td>
<td>FSR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address 1 Low Pointer</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>x07h or x87h</td>
<td>FSR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address 1 High Pointer</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>x08h or x88h</td>
<td>BSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BSR&lt;4:0&gt;</td>
<td>----0 0000</td>
<td>----0 0000</td>
</tr>
<tr>
<td>x09h or x89h</td>
<td>WREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Working Register</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>x0Ah or x8Ah</td>
<td>PCLATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write Buffer for the upper 7 bits of the Program Counter</td>
<td>--00 0000</td>
<td>--00 0000</td>
</tr>
<tr>
<td>x0Bh or x8Bh</td>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
</tbody>
</table>

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

Shaded locations are unimplemented, read as '0'.
### TABLE 3-6: SPECIAL FUNCTION REGISTER SUMMARY

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Value on all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00Ch</td>
<td>PORTA</td>
<td></td>
<td></td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00Eh to</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>011h</td>
<td>PIR1</td>
<td></td>
<td>ADIF</td>
<td></td>
<td></td>
<td>SSPIF</td>
<td></td>
<td></td>
<td></td>
<td>-0-- 0---- -0-- 0----</td>
<td></td>
</tr>
<tr>
<td>012h</td>
<td>PIR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCLIF</td>
<td></td>
<td></td>
<td></td>
<td>----- 0---- ----- 0----</td>
<td></td>
</tr>
<tr>
<td>013h</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>014h</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>015h</td>
<td>TMR0 Holding Register for the 8-bit Timer0 Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxx uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>016h to</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08Ch</td>
<td>TRISA</td>
<td></td>
<td></td>
<td>TRISA5</td>
<td>TRISA4</td>
<td></td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>--11 1111 --11 1111</td>
<td></td>
</tr>
<tr>
<td>08Dh</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08Eh</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>08Fh</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>090h</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>091h</td>
<td>PIE1</td>
<td></td>
<td>ADIE</td>
<td></td>
<td></td>
<td>SSPIE</td>
<td></td>
<td></td>
<td></td>
<td>-0-- 0---- -0-- 0----</td>
<td></td>
</tr>
<tr>
<td>092h</td>
<td>PIE2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCLIE</td>
<td></td>
<td></td>
<td></td>
<td>----- 0---- ----- 0----</td>
<td></td>
</tr>
<tr>
<td>093h</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>094h</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>095h</td>
<td>OPTION_REG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>096h</td>
<td>PCON</td>
<td></td>
<td>STKOVF</td>
<td>STKUNF</td>
<td></td>
<td>RWDT</td>
<td>RMCLR</td>
<td>RI</td>
<td>POR</td>
<td>RMCLR</td>
<td></td>
</tr>
<tr>
<td>097h</td>
<td>WDTCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SWDTEN</td>
<td>--01 0110 --01 0110</td>
</tr>
<tr>
<td>098h</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>099h</td>
<td>OSCCON</td>
<td></td>
<td>SPLLEN</td>
<td>IRCF&lt;3:0&gt;</td>
<td></td>
<td></td>
<td>SCs&lt;1:0&gt;</td>
<td></td>
<td></td>
<td>0011 1-00 0011 1-00</td>
<td></td>
</tr>
<tr>
<td>09Ah</td>
<td>OSCSTAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HIIOFR</td>
<td></td>
<td>LFIoFR</td>
<td>HIIOFS</td>
<td>-0-0-0--0-- -q-q --qq</td>
<td></td>
</tr>
<tr>
<td>09Bh</td>
<td>ADRESL&lt;2&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09Ch</td>
<td>ADRESH&lt;2&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09Dh</td>
<td>ADCON2&lt;2&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09Eh</td>
<td>ADCON1&lt;2&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09Fh</td>
<td>ADCON3&lt;2&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bank 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10Ch</td>
<td>LATA</td>
<td></td>
<td></td>
<td>LATA5</td>
<td>LATA4</td>
<td></td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
<td>--xx --xx --uu --uu</td>
<td></td>
</tr>
<tr>
<td>10Dh to</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>116h</td>
<td>BORCON</td>
<td></td>
<td>SBOREN</td>
<td>BORFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BORRDY</td>
<td>10-- --qq uu-- ----u</td>
<td></td>
</tr>
<tr>
<td>117h</td>
<td>FVRCON</td>
<td></td>
<td>FVREN</td>
<td>FVRRDY</td>
<td>TSEN</td>
<td>TSRSN</td>
<td></td>
<td></td>
<td>ADPVR&lt;1:0&gt;</td>
<td>0q00 --0q 0q00 --0q</td>
<td></td>
</tr>
<tr>
<td>118h to</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11Ch</td>
<td>APFCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11Dh</td>
<td>SDOSEL SSSEL SDSEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-000 ---- -000 ----</td>
<td></td>
</tr>
<tr>
<td>11Eh</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11Fh</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Note 1: Unimplemented, read as "-1".

Note 2: This register is available in Bank 1 and Bank 14 under similar register names. See Section 16.1.11 “Hardware CVD Register Mapping”.

---

**Legend:**
- **x** = unknown,
- **u** = unchanged,
- **q** = value depends on condition,
- **-** = unimplemented,
- **r** = reserved.

**Shaded locations are unimplemented, read as '0'**.
### TABLE 3-6: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

<table>
<thead>
<tr>
<th>Bank 3</th>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Value on all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>18Ch</td>
<td>ANSELA</td>
<td>—</td>
<td>—</td>
<td>ANSA5</td>
<td>ANSA4</td>
<td>—</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
<td>—</td>
<td>--11 1111</td>
<td>--11 1111</td>
</tr>
<tr>
<td>18Dh</td>
<td>—</td>
<td>Unimplemented</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>18Eh</td>
<td>—</td>
<td>Unimplemented</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>18Fh</td>
<td>—</td>
<td>Unimplemented</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>190h</td>
<td>—</td>
<td>Unimplemented</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>191h</td>
<td>PMADRL</td>
<td>Flash Program Memory Address Register Low Byte</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0000 0000 0000</td>
<td>—</td>
</tr>
<tr>
<td>192h</td>
<td>PMADRH</td>
<td>(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1000 0000 1000 0000</td>
<td>—</td>
</tr>
<tr>
<td>193h</td>
<td>PMDATL</td>
<td>Flash Program Memory Read Data Register Low Byte</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>xxx xxx xx xx xx</td>
<td>—</td>
</tr>
<tr>
<td>194h</td>
<td>PMDATH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>--xx xxx xx xx xx</td>
<td>—</td>
</tr>
<tr>
<td>195h</td>
<td>PMCON1</td>
<td>(1)</td>
<td>CFGS</td>
<td>LWLO</td>
<td>FREE</td>
<td>WRERR</td>
<td>WREN</td>
<td>WR</td>
<td>RD</td>
<td>0000 x000 0000 q000</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>196h</td>
<td>PMCON2</td>
<td>Flash Program Memory Control Register 2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0000 0000 0000</td>
<td>—</td>
</tr>
<tr>
<td>197h</td>
<td>to 19Fh</td>
<td>Unimplemented</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

### Bank 4

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Value on all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>20Ch</td>
<td>WPUA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WPUA5</td>
<td>WPUA4</td>
<td>WPUA3</td>
<td>WPUA2</td>
<td>WPUA1</td>
<td>WPUA0</td>
<td>--11 1111 111111 111111</td>
</tr>
<tr>
<td>20Dh</td>
<td>to 210h</td>
<td>Unimplemented</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>211h</td>
<td>SSPBUF</td>
<td>Synchronous Serial Port Receive Buffer/Transmit Register</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>212h</td>
<td>SSPADD</td>
<td>ADD&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>213h</td>
<td>SSPMSK</td>
<td>MSK&lt;7:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>214h</td>
<td>SSPSTAT</td>
<td>SMP</td>
<td>CKE</td>
<td>D/Ä</td>
<td>P</td>
<td>S</td>
<td>R/W</td>
<td>UA</td>
<td>BF</td>
<td>0000 0000 0000 0000</td>
<td>1111 1111 1111 1111</td>
</tr>
<tr>
<td>215h</td>
<td>SSPCON1</td>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPEN</td>
<td>CKP</td>
<td>SSM&lt;3:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0000 0000 0000</td>
<td>—</td>
</tr>
<tr>
<td>216h</td>
<td>SSPCON2</td>
<td>GCEN</td>
<td>ACKSTAT</td>
<td>ACKDT</td>
<td>ACKEN</td>
<td>RCEN</td>
<td>PEN</td>
<td>RSEN</td>
<td>SEN</td>
<td>0000 0000 0000 0000</td>
<td>—</td>
</tr>
<tr>
<td>217h</td>
<td>SSPCON3</td>
<td>ACKTIM</td>
<td>PCIe</td>
<td>SCIE</td>
<td>BOEN</td>
<td>SDAHT</td>
<td>SBCDE</td>
<td>AHEN</td>
<td>DHEN</td>
<td>0000 0000 0000 0000</td>
<td>—</td>
</tr>
<tr>
<td>218h</td>
<td>to 21Fh</td>
<td>Unimplemented</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

### Bank 5

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Value on all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>28Ch</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>29Fh</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

### Bank 6

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Value on all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>30Ch</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>31Fh</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**Legend:**
- `x` = unknown,
- `u` = unchanged,
- `q` = value depends on condition,
- `-` = unimplemented,
- `r` = reserved.

**Note:**
1. Unimplemented, read as ‘0’.
2. This register is available in Bank 1 and Bank 14 under similar register names. See Section 16.1.11 “Hardware CVD Register Mapping”.
### TABLE 3-6: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

<table>
<thead>
<tr>
<th>Address (Bank)</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Value on all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bank 7</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38Ch to 39h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>391h</td>
<td>IOCAP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCAP5</td>
<td>IOCAP4</td>
<td>IOCAP3</td>
<td>IOCAP2</td>
<td>IOCAP1</td>
<td>IOCAP0</td>
<td>---000000</td>
</tr>
<tr>
<td>392h</td>
<td>IOCAN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCAN5</td>
<td>IOCAN4</td>
<td>IOCAN3</td>
<td>IOCAN2</td>
<td>IOCAN1</td>
<td>IOCAN0</td>
<td>---000000</td>
</tr>
<tr>
<td>393h</td>
<td>IOCAF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCAF5</td>
<td>IOCAF4</td>
<td>IOCAF3</td>
<td>IOCAF2</td>
<td>IOCAF1</td>
<td>IOCAF0</td>
<td>---000000</td>
</tr>
<tr>
<td>394h to 39Fh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bank 8-13</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x0Ch/ x0Ch</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x1Fh/ x9Fh</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bank 14</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70Ch to 71h</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
</tr>
<tr>
<td>711h</td>
<td>AADCON0(2)</td>
<td>—</td>
<td></td>
<td>—</td>
<td></td>
<td>CHS&lt;4:0&gt;</td>
<td></td>
<td>GO/DONE</td>
<td>ADON</td>
<td>-000000</td>
<td>-000000</td>
</tr>
<tr>
<td>712h</td>
<td>AADCON1(2)</td>
<td>ADFM</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADCS&lt;2:0&gt;</td>
<td>—</td>
<td>ADPREF&lt;1:0&gt;</td>
<td>0000 ---00</td>
<td>0000 ---00</td>
</tr>
<tr>
<td>713h</td>
<td>AADCON2(2)</td>
<td>—</td>
<td></td>
<td>—</td>
<td>—</td>
<td>TRIGSEL&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>-000 -----</td>
<td>-000 -----</td>
</tr>
<tr>
<td>714h</td>
<td>AADCON3</td>
<td>ADEPPOL</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADIPOL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0-02</td>
</tr>
<tr>
<td>715h</td>
<td>AADSTAT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADCONV</td>
<td>ADSTG&lt;1:0&gt;</td>
<td>----- 0000</td>
<td>----- 0000</td>
</tr>
<tr>
<td>716h</td>
<td>AADPRE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADPRE&lt;6:0&gt;</td>
<td>-000 0000</td>
<td>-000 0000</td>
</tr>
<tr>
<td>717h</td>
<td>AADACQ</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADACQ&lt;6:0&gt;</td>
<td>-000 0000</td>
<td>-000 0000</td>
</tr>
<tr>
<td>718h</td>
<td>AADGRD</td>
<td>GRDBOE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>000- -----</td>
</tr>
<tr>
<td>719h</td>
<td>AADCAP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADCAP&lt;2:0&gt;</td>
</tr>
<tr>
<td>71Ah</td>
<td>AADDRS0L(2)</td>
<td>ADC Result Register 0 Low</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>xxxxx xxxxx</td>
</tr>
<tr>
<td>71Bh</td>
<td>AADDRS0H(2)</td>
<td>ADC Result Register 0 High</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>xxxxx xxxxx</td>
</tr>
<tr>
<td>71Ch</td>
<td>AADDRS1L(2)</td>
<td>ADC Result Register 1 Low</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>xxxxx xxxxx</td>
</tr>
<tr>
<td>71Dh</td>
<td>AADDRS1H(2)</td>
<td>ADC Result Register 1 High</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>xxxxx xxxxx</td>
</tr>
<tr>
<td>71 Eh</td>
<td>—</td>
<td>Unimplemented</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>71Fh</td>
<td>—</td>
<td>Unimplemented</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

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

**Note:**
1. Unimplemented, read as '1'.
2. This register is available in Bank 1 and Bank 14 under similar register names. See Section 16.1.11 “Hardware CVD Register Mapping”.
### TABLE 3-6: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Value on POR, BOR</th>
<th>Value on all other Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>x0Ch/</td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x0Ch/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x1Fh/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x1Fh/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F8Ch</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE3h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE4h</td>
<td>STATUS_ SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
<td>—</td>
<td></td>
<td>—</td>
<td></td>
<td>Z_SHAD</td>
<td>DC_SHAD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td>—</td>
<td></td>
<td>DC_SHAD</td>
<td>C_SHAD</td>
</tr>
<tr>
<td>FE5h</td>
<td>WREG_ SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FE6h</td>
<td>BSR_ SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FE7h</td>
<td>PCLATH_ SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FE8h</td>
<td>FSR0L_ SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FE9h</td>
<td>FSR0H_ SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FEAh</td>
<td>FSR1L_ SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FEBh</td>
<td>FSR1H_ SHAD</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FECb</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FEDh</td>
<td>STKPTR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FEEh</td>
<td>TOSL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
<tr>
<td>FEFh</td>
<td>TOSH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Unimplemented</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- `x` = unknown
- `u` = unchanged
- `q` = value depends on condition
- `-` = unimplemented
- `r` = reserved

**Shaded locations are unimplemented, read as ’0’.

**Note:**
1. Unimplemented, read as ’1’.
2. This register is available in Bank 1 and Bank 14 under similar register names. See Section 16.1.11 “Hardware CVD Register Mapping”.

© 2012-2016 Microchip Technology Inc.
3.4 PCL and PCLATH

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

FIGURE 3-3: LOADING OF PC IN DIFFERENT SITUATIONS

3.4.1 MODIFYING PCL

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

3.4.2 COMPUTED GOTO

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

3.4.3 COMPUTED FUNCTION CALLS

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

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

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

3.4.4 BRANCHING

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

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

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

All devices have a 16-level x 15-bit wide hardware stack (refer to Figures 3-4 through 3-7). The stack space is not part of either program or data space. The PC is PUSHed onto the stack when CALL or CALLW instructions are executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not affected by a PUSH or POP operation.

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

Note 1: There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the CALL, CALLW, RETURN, RETLW and RETFIE instructions or the vectoring to an interrupt address.

3.5.1 ACCESSING THE STACK

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

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

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

Reference Figure 3-4 through Figure 3-7 for examples of accessing the stack.

FIGURE 3-4: ACCESSING THE STACK EXAMPLE 1

<table>
<thead>
<tr>
<th>TOSH:TOSL</th>
<th>0x0F</th>
<th>STKPTR = 0x1F</th>
<th>Stack Reset Disabled (STVREN = 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0E</td>
<td></td>
<td></td>
<td>Initial Stack Configuration:</td>
</tr>
<tr>
<td>0x0D</td>
<td></td>
<td></td>
<td>After Reset, the stack is empty.</td>
</tr>
<tr>
<td>0x0C</td>
<td></td>
<td></td>
<td>The empty stack is initialized so</td>
</tr>
<tr>
<td>0x0B</td>
<td></td>
<td></td>
<td>the Stack Pointer is pointing at</td>
</tr>
<tr>
<td>0x0A</td>
<td></td>
<td></td>
<td>0x1F. If the Stack Overflow/</td>
</tr>
<tr>
<td>0x09</td>
<td></td>
<td></td>
<td>Underflow Reset is enabled, the</td>
</tr>
<tr>
<td>0x08</td>
<td></td>
<td></td>
<td>TOSH/TOSL registers will return 0.</td>
</tr>
<tr>
<td>0x07</td>
<td></td>
<td></td>
<td>If the Stack Overflow/Underflow</td>
</tr>
<tr>
<td>0x06</td>
<td></td>
<td></td>
<td>Reset is disabled, the TOSH/TOSL</td>
</tr>
<tr>
<td>0x05</td>
<td></td>
<td></td>
<td>registers will return the contents</td>
</tr>
<tr>
<td>0x04</td>
<td></td>
<td></td>
<td>of stack address 0x0F.</td>
</tr>
<tr>
<td>0x03</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| TOSH:TOSL | 0x1F | 0x0000 | STKPTR = 0x1F | Stack Reset Enabled (STVREN = 1) |

Initial Stack Configuration:
After Reset, the stack is empty. The empty stack is initialized so the Stack Pointer is pointing at 0x1F. If the Stack Overflow/Underflow Reset is enabled, the TOSH/TOSL registers will return ‘0’. If the Stack Overflow/Underflow Reset is disabled, the TOSH/TOSL registers will return the contents of stack address 0x0F.
FIGURE 3-5: ACCESSING THE STACK EXAMPLE 2

This figure shows the stack configuration after the first CALL or a single interrupt. If a RETURN instruction is executed, the return address will be placed in the Program Counter and the Stack Pointer decremented to the empty state (0x1F).

FIGURE 3-6: ACCESSING THE STACK EXAMPLE 3

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

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

3.6 Indirect Addressing

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

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

- Traditional Data Memory
- Linear Data Memory
- Program Flash Memory
FIGURE 3-8: INDIRECT ADDRESSING

Note: Not all memory regions are completely implemented. Consult device memory tables for memory limits.
3.6.1 TRADITIONAL DATA MEMORY

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

FIGURE 3-9: TRADITIONAL DATA MEMORY MAP

<table>
<thead>
<tr>
<th>Direct Addressing</th>
<th>Indirect Addressing</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSR</td>
<td>0</td>
</tr>
<tr>
<td>0000</td>
<td>0001</td>
</tr>
<tr>
<td>0x00</td>
<td>0x7F</td>
</tr>
</tbody>
</table>

Bank Select | Location Select

0x00 | 0x7F
3.6.2 LINEAR DATA MEMORY

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

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

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

FIGURE 3-10: LINEAR DATA MEMORY MAP

3.6.3 PROGRAM FLASH MEMORY

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

FIGURE 3-11: PROGRAM FLASH MEMORY MAP
4.0 DEVICE CONFIGURATION

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

4.1 Configuration Words

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

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

<table>
<thead>
<tr>
<th>U-1</th>
<th>U-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>U-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>CLKOUTEN</td>
<td>BOREN&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>bit 13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>U-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP</td>
<td>MCLRE</td>
<td>PWRTE</td>
<td>WDTE&lt;1:0&gt;</td>
<td>—</td>
<td>FOSC&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

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

| bit 13-12 | **Unimplemented**: Read as ‘1’ |
| bit 11    | **CLKOUTEN**: Clock Out Enable bit |
|          | 1 = CLKOUT function is disabled. I/O function on the CLKOUT pin |
|          | 0 = CLKOUT function is enabled on the CLKOUT pin |
| bit 10-9  | **BOREN<1:0>:** Brown-out Reset Enable bits(1) |
|          | 11 = BOR enabled |
|          | 10 = BOR enabled during operation and disabled in Sleep |
|          | 01 = BOR controlled by SBOREN bit of the BORCON register |
|          | 00 = BOR disabled |
| bit 8    | **Unimplemented**: Read as ‘1’ |
| bit 7    | **CP**: Code Protection bit(2) |
|          | 1 = Program memory code protection is disabled |
|          | 0 = Program memory code protection is enabled |
| bit 6    | **MCLRE**: MCLR/VPP Pin Function Select bit |
|          | If LVP bit = 1: |
|          | This bit is ignored. |
|          | If LVP bit = 0: |
|          | 1 = MCLR/VPP pin function is MCLR; Weak pull-up enabled. |
|          | 0 = MCLR/VPP pin function is digital input; MCLR internally disabled; Weak pull-up under control of WPUE3 bit. |
| bit 5    | **PWRTE**: Power-Up Timer Enable bit |
|          | 1 = PWRT disabled |
|          | 0 = PWRT enabled |
| bit 4-3  | **WDTE<1:0>:** Watchdog Timer Enable bits |
|          | 11 = WDT enabled |
|          | 10 = WDT enabled while running and disabled in Sleep |
|          | 01 = WDT controlled by the SWDTEN bit in the WDTCN register |
|          | 00 = WDT disabled |
| bit 2    | **Unimplemented**: Read as ‘1’ |
| bit 1-0  | **FOSC<1:0>:** Oscillator Selection bits |
|          | 11 = ECH: External Clock, High-Power mode: on CLKIN pin |
|          | 10 = ECM: External Clock, Medium-Power mode: on CLKIN pin |
|          | 01 = ECL: External Clock, Low-Power mode: on CLKIN pin |
|          | 00 = INTOSC oscillator: I/O function on CLKIN pin |

**Note 1:** Enabling Brown-out Reset does not automatically enable Power-up Timer.

**Note 2:** Once enabled, code-protect can only be disabled by bulk erasing the device.
### REGISTER 4-2: CONFIG2: CONFIGURATION WORD 2

<table>
<thead>
<tr>
<th>R/P-1</th>
<th>U-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>U-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVP</td>
<td></td>
<td>—</td>
<td>LPBOR</td>
<td>BORV</td>
<td>STVREN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bit 13</td>
<td></td>
<td></td>
<td>bit 8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U-1</th>
<th>U-1</th>
<th>U-1</th>
<th>U-1</th>
<th>U-1</th>
<th>U-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>WRT&lt;1:0&gt;</td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 0</td>
</tr>
</tbody>
</table>

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

**bit 13**
- **LVP**: Low-Voltage Programming Enable bit\(^{(1)}\)
  - 1 = Low-voltage programming enabled
  - 0 = High-voltage on MCLR must be used for programming

**bit 12**
- **Unimplemented**: Read as ‘1’

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

**bit 10**
- **BORV**: Brown-out Reset Voltage Selection bit\(^{(2)}\)
  - 1 = Brown-out Reset voltage (V\(_{\text{bor}}\)), low trip point selected
  - 0 = Brown-out Reset voltage (V\(_{\text{bor}}\)), high trip point selected

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

**bit 8-2**
- **Unimplemented**: Read as ‘1’

**bit 1-0**
- **WRT<1:0>**: Flash Memory Self-Write Protection bits
  - **2 kW Flash memory:**
    - 11 = Write protection off
    - 10 = 000h to 1FFh write-protected, 200h to 7FFh may be modified
    - 01 = 000h to 3FFh write-protected, 400h to 7FFh may be modified
    - 00 = 000h to 7FFh write-protected, no addresses may be modified

**Note 1:** The LVP bit cannot be programmed to ‘0’ when Programming mode is entered via LVP.

**Note 2:** See V\(_{\text{bor}}\) parameter for specific trip point voltages.
4.3 Code Protection

Code protection allows the device to be protected from unauthorized access. Internal access to the program memory is unaffected by any code protection setting.

4.3.1 PROGRAM MEMORY PROTECTION

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

4.4 Write Protection

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

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

4.5 User ID

Four memory locations (8000h-8003h) are designated as ID locations where the user can store checksum or other code identification numbers. These locations are readable and writable during normal execution. See Section 10.4 “User ID, Device ID and Configuration Word Access” for more information on accessing these memory locations. For more information on checksum calculation, see the “PIC12LF1552 Memory Programming Specification” (DS41642).
4.6 Device ID and Revision ID

The memory location 8006h is where the Device ID and Revision ID are stored. The upper nine bits hold the Device ID. The lower five bits hold the Revision ID. See Section 10.4 “User ID, Device ID and Configuration Word Access” for more information on accessing these memory locations.

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

4.7 Register Definitions: Device

REGISTER 4-3: DEVID: DEVICE ID REGISTER

<table>
<thead>
<tr>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DEV&lt;8:3&gt;</td>
</tr>
<tr>
<td>bit 13</td>
<td>bit 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DEV&lt;2:0&gt;</td>
<td>REV&lt;4:0&gt;</td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit
‘1’ = Bit is set ‘0’ = Bit is cleared

bit 13-5

DEV<8:0>: Device ID bits

<table>
<thead>
<tr>
<th>Device</th>
<th>DEVID&lt;13:0&gt; Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC12LF1552</td>
<td>0010 1011 110</td>
</tr>
</tbody>
</table>

bit 4-0

REV<4:0>: Revision ID bits

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

5.1 Overview

The oscillator module has a wide variety of clock sources and selection features that allow it to be used in a wide range of applications while maximizing performance and minimizing power consumption. Figure 5-1 illustrates a block diagram of the oscillator module.

Clock sources can be supplied from external clock oscillators. In addition, the system clock source can be supplied from one of two internal oscillators and PLL circuits, with a choice of speeds selectable via software. Additional clock features include:

- Selectable system clock source between external or internal sources via software.

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

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

Clock Source modes are selected by the FOSC<1:0> bits in the Configuration Words. The FOSC bits determine the type of oscillator that will be used when the device is first powered.

The EC clock mode relies on an external logic level signal as the device clock source.

The INTOSC internal oscillator block produces low and high-frequency clock sources, designated LFINTOSC and HFINTOSC. (see Internal Oscillator Block, Figure 5-1). A wide selection of device clock frequencies may be derived from these clock sources.

---

**FIGURE 5-1: SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM**
5.2 Clock Source Types

Clock sources can be classified as external or internal. External clock sources rely on external circuitry for the clock source to function. Examples are: oscillator modules (EC mode).

Internal clock sources are contained within the oscillator module. The oscillator block has two internal oscillators that are used to generate two system clock sources: the 16 MHz High-Frequency Internal Oscillator (HFINTOSC) and the 31 kHz Low-Frequency Internal Oscillator (LFINTOSC).

The system clock can be selected between external or internal clock sources via the System Clock Select (SCS) bits in the OSCCON register. See Section 5.3 “Clock Switching” for additional information.

5.2.1 EXTERNAL CLOCK SOURCES

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

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

See Section 5.3 “Clock Switching” for more information.

5.2.1.1 EC Mode

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

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

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

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

FIGURE 5-2: EXTERNAL CLOCK (EC) MODE OPERATION

![Diagram of External Clock (EC) Mode Operation]

Note 1: Output depends upon CLkinOUTEN bit of the Configuration Words.
5.2.2 INTERNAL CLOCK SOURCES

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

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

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

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

The internal oscillator block has two independent oscillators.

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

5.2.2.1 HFINTOSC

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

The outputs of the HFINTOSC connects to a prescaler and multiplexer (see Figure 5-1). One of multiple frequencies derived from the HFINTOSC can be selected via software using the IRCF<3:0> bits of the OSCCON register. See Section 5.2.2.4 “Internal Oscillator Clock Switch Timing” for more information.

The HFINTOSC is enabled by:

- Configure the IRCF<3:0> bits of the OSCCON register for the desired HF frequency, and
- FOSC<1:0> = 00, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '1x'.

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

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

5.2.2.2 LFINTOSC

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

The output of the LFINTOSC connects to a multiplexer (see Figure 5-1). Select 31 kHz, via software, using the IRCF<3:0> bits of the OSCCON register. See Section 5.2.2.4 “Internal Oscillator Clock Switch Timing” for more information. The LFINTOSC is also the source for the Power-up Timer (PWRT) and Watchdog Timer (WDT).

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

- Configure the IRCF<3:0> bits of the OSCCON register for the LF frequency, and
- FOSC<1:0> = 00, or
- Set the System Clock Source (SCS) bits of the OSCCON register to '1x'.

Peripherals that use the LFINTOSC are:

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

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

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

The outputs of the 16 MHz HFINTOSC postscaler and the LFINTOSC connect to a multiplexer (see Figure 5-1). The Internal Oscillator Frequency Select bits IRCF<3:0> of the OSCCON register select the frequency. One of the following frequencies can be selected via software:

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

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

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

5.2.2.4 Internal Oscillator Clock Switch Timing

When switching between the HFINTOSC and the LFINTOSC, the new oscillator may already be shut down to save power (see Figure 5-3). If this is the case, there is a delay after the IRCF<3:0> bits of the OSCCON register are modified before the frequency selection takes place. The OSCSTAT register will reflect the current active status of the HFINTOSC and LFINTOSC oscillators. The sequence of a frequency selection is as follows:

1. IRCF<3:0> bits of the OSCCON register are modified.
2. If the new clock is shut down, a clock start-up delay is started.
3. Clock switch circuitry waits for a falling edge of the current clock.
4. Clock switch is complete.

See Figure 5-3 for more details.

If the internal oscillator speed is switched between two clocks of the same source, there is no start-up delay before the new frequency is selected.

Start-up delay specifications are located in the oscillator tables of Section 21.0 “Electrical Specifications”.

5.2.2.5 32 MHz Internal Oscillator Frequency Selection

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

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

The 4x PLL is not available for use with the internal oscillator when the SCS bits of the OSCCON register are set to '1x'. The SCS bits must be set to '00' to use the 4x PLL with the internal oscillator.
FIGURE 5-3: INTERNAL OSCILLATOR SWITCH TIMING

HFINTOSC → LFINTOSC (WDT disabled)

HFINTOSC
LFINTOSC
IRCF <3:0> ≠ 0 = 0
System Clock

HFINTOSC → LFINTOSC (WDT enabled)

HFINTOSC
LFINTOSC
IRCF <3:0> ≠ 0 = 0
System Clock

LFINTOSC → HFINTOSC

LFINTOSC turns off unless WDT is enabled

LFINTOSC
HFINTOSC
IRCF <3:0> = 0 ≠ 0
System Clock
5.3 Clock Switching

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

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

5.3.1 SYSTEM CLOCK SELECT (SCS) BITS

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

- When the SCS bits of the OSCCON register = 00, the system clock source is determined by value of the FOSC<1:0> bits in the Configuration Words.
- When the SCS bits of the OSCCON register = 1x, the system clock source is chosen by the internal oscillator frequency selected by the IRCF<3:0> bits of the OSCCON register. After a Reset, the SCS bits of the OSCCON register are always cleared.

When switching between clock sources, a delay is required to allow the new clock to stabilize. These oscillator delays are shown in Table 5-2.

5.3.2 CLOCK SWITCHING BEFORE SLEEP

When clock switching from an old clock to a new clock is requested just prior to entering Sleep mode, it is necessary to confirm that the switch is complete before the SLEEP instruction is executed. Failure to do so may result in an incomplete switch and consequential loss of the system clock altogether. Clock switching is confirmed by monitoring the Clock Status bits in the OSCSTAT register. Switch confirmation can be accomplished by sensing that the Ready bit for the new clock is set or the Ready bit for the old clock is cleared. For example, when switching between the internal oscillator with the PLL and the internal oscillator without the PLL, monitor the PLLR bit. When PLLR is set, the switch to 32 MHz operation is complete. Conversely, when PLLR is cleared, the switch from 32 MHz operation to the selected internal clock is complete.

<table>
<thead>
<tr>
<th>Switch From</th>
<th>Switch To</th>
<th>Frequency</th>
<th>Oscillator Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>LFINTOSC(1)</td>
<td>31 kHz</td>
<td>Oscillator Warm-Up Delay T\textsubscript{WARM}(2)</td>
</tr>
<tr>
<td></td>
<td>MFINTOSC(1)</td>
<td>31.25 kHz-500 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HFINTOSC(1)</td>
<td>31.25 kHz-16 MHz</td>
<td></td>
</tr>
<tr>
<td>Sleep/POR</td>
<td>EC(1)</td>
<td>DC – 32 MHz</td>
<td>2 cycles</td>
</tr>
<tr>
<td>LFINTOSC</td>
<td>EC(1)</td>
<td>DC – 32 MHz</td>
<td>1 cycle of each</td>
</tr>
<tr>
<td>Any clock source</td>
<td>MFINTOSC(1)</td>
<td>31.25 kHz-500 kHz</td>
<td>2 (\mu)s (approx.)</td>
</tr>
<tr>
<td></td>
<td>HFINTOSC(1)</td>
<td>31.25 kHz-16 MHz</td>
<td></td>
</tr>
<tr>
<td>Any clock source</td>
<td>LFINTOSC(1)</td>
<td>31 kHz</td>
<td>1 cycle of each</td>
</tr>
<tr>
<td>PLL inactive</td>
<td>PLL active</td>
<td>16-32 MHz</td>
<td>2 ms (approx.)</td>
</tr>
</tbody>
</table>

Note 1: PLL inactive.
Note 2: See Section 21.0 “Electrical Specifications”.

TABLE 5-1: OSCILLATOR SWITCHING DELAYS
5.4 Register Definitions: Oscillator Control

**REGISTER 5-1: OSCCON: OSCILLATOR CONTROL REGISTER**

<table>
<thead>
<tr>
<th>Bit</th>
<th>SPLLEN</th>
<th>IRCF&lt;3:0&gt;</th>
<th>_</th>
<th>SCS&lt;1:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-1/1</td>
<td>U-0</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared

**bit 7**
- **SPLLEN**: Software PLL Enable bit
  - `1` = 4x PLL is enabled
  - `0` = 4x PLL is disabled

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

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

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

**Note 1**: Duplicate frequency derived from HFINTOSC.
REGISTER 5-2: OSCSTAT: OSCILLATOR STATUS REGISTER

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLLR</td>
<td>—</td>
<td>—</td>
<td>HFIOFR</td>
<td>—</td>
<td>—</td>
<td>LFIOFR</td>
<td>HFIOFS</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- q = Conditional

- Unimplemented: Read as ‘0’
- PLLR: 4x PLL Ready bit
  - 1 = 4x PLL is ready
  - 0 = 4x PLL is not ready
- HFIOFR: High-Frequency Internal Oscillator Ready bit
  - 1 = 16 MHz Internal Oscillator (HFINTOSC) is ready
  - 0 = 16 MHz Internal Oscillator (HFINTOSC) is not ready
- LFIOFR: Low-Frequency Internal Oscillator Ready bit
  - 1 = 31 kHz Internal Oscillator (LFINTOSC) is ready
  - 0 = 31 kHz Internal Oscillator (LFINTOSC) is not ready
- HFIOFS: High-Frequency Internal Oscillator Stable bit
  - 1 = 16 MHz Internal Oscillator (HFINTOSC) is stable
  - 0 = 16 MHz Internal Oscillator (HFINTOSC) is not yet stable

TABLE 5-2: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK SOURCES

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSCCON</td>
<td>SPLLEN</td>
<td>IRCF&lt;3:0&gt;</td>
<td>—</td>
<td>—</td>
<td>SCS&lt;1:0&gt;</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OSCSTAT</td>
<td>—</td>
<td>PLLR</td>
<td>—</td>
<td>HFIOFR</td>
<td>—</td>
<td>LFIOFR</td>
<td>HFIOFS</td>
<td>43</td>
<td></td>
</tr>
</tbody>
</table>

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

TABLE 5-3: SUMMARY OF CONFIGURATION WORD WITH CLOCK SOURCES

<table>
<thead>
<tr>
<th>Name</th>
<th>Bits</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 13/5</th>
<th>Bit 12/4</th>
<th>Bit 11/3</th>
<th>Bit 10/2</th>
<th>Bit 9/1</th>
<th>Bit 8/0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG1</td>
<td>13:3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLKOUTEN</td>
<td>BOREN&lt;1:0&gt;</td>
<td>—</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>CF</td>
<td>MCLRE</td>
<td>PWRT</td>
<td>WDTE&lt;1:0&gt;</td>
<td>—</td>
<td>FOSC&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

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

There are multiple ways to reset this device:
- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Low-Power Brown-out Reset (LPBOR)
- MCLR Reset
- WDT Reset
- RESET instruction
- Stack Overflow
- Stack Underflow
- Programming mode exit

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

A simplified block diagram of the On-chip Reset Circuit is shown in Figure 6-1.

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

**Note 1:** See Table 6-1 for BOR active conditions.
6.1 Power-on Reset (POR)

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

6.1.1 POWER-UP TIMER (PWRT)

The Power-up Timer provides a nominal 64 ms time-out on POR or Brown-out Reset. The device is held in Reset as long as PWRT is active. The PWRT delay allows additional time for the VDD to rise to an acceptable level. The Power-up Timer is enabled by clearing the PWRTEN bit in Configuration Words. The Power-up Timer starts after the release of the POR and BOR.

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

6.2 Brown-out Reset (BOR)

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

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

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

Refer to Table 6-1 for more information.

The Brown-out Reset voltage level is selectable by configuring the BORV bit in Configuration Words. A VDD noise rejection filter prevents the BOR from triggering on small events. If VDD falls below VBOR for a duration greater than parameter TBORDC, the device will reset. See Figure 6-2 for more information.

### Table 6-1: BOR OPERATING MODES

<table>
<thead>
<tr>
<th>BOREN&lt;1:0&gt;</th>
<th>SBOREN</th>
<th>Device Mode</th>
<th>BOR Mode</th>
<th>Instruction Execution upon: Release of POR or Wake-up from Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>X</td>
<td>X</td>
<td>Active</td>
<td>Waits for BOR ready&lt;sup&gt;(1)&lt;/sup&gt; (BORRDY = 1)</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>Awake</td>
<td>Active</td>
<td>Waits for BOR ready (BORRDY = 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sleep</td>
<td>Disabled</td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>X</td>
<td>Active</td>
<td>Waits for BOR ready&lt;sup&gt;(1)&lt;/sup&gt; (BORRDY = 1)</td>
</tr>
<tr>
<td>00</td>
<td>X</td>
<td>X</td>
<td>Disabled</td>
<td>Begins immediately (BORRDY = x)</td>
</tr>
</tbody>
</table>

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

6.2.1 BOR IS ALWAYS ON

When the BOREN bits of Configuration Words are programmed to ‘11’, the BOR is always on. The device start-up will be delayed until the BOR is ready and VDD is higher than the BOR threshold.

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

6.2.2 BOR IS OFF IN SLEEP

When the BOREN bits of Configuration Words are programmed to ‘10’, the BOR is on, except in Sleep. The device start-up will be delayed until the BOR is ready and VDD is higher than the BOR threshold.

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

6.2.3 BOR CONTROLLED BY SOFTWARE

When the BOREN bits of Configuration Words are programmed to ‘01’, the BOR is controlled by the SBOREN bit of the BORCON register. The device start-up is not delayed by the BOR ready condition or the VDD level.

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

BOR protection is unchanged by Sleep.
FIGURE 6-2: BROWN-OUT SITUATIONS

Note 1: TPWR delay only if PWRTE bit is programmed to '0'.

6.3 Register Definitions: BOR Control

REGISTER 6-1: BORCON: BROWN-OUT RESET CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-1/u</th>
<th>R/W-0/u</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-q/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBOREN</td>
<td>BORFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BORRDY</td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 6</td>
<td>bit 5-1</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7  SBOREN: Software Brown-out Reset Enable bit
  - If BOREN <1:0> in Configuration Words = 01:
    1 = BOR Enabled
    0 = BOR Disabled
  - If BOREN <1:0> in Configuration Words ≠ 00:
    SBOREN is read/write, but has no effect on the BOR.

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

bit 5-1 Unimplemented: Read as '0'

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

Note 1: BOREN<1:0> bits are located in Configuration Words.
6.4 Low-Power Brown-out Reset (LPBOR)

The Low-Power Brown-Out Reset (LPBOR) is an essential part of the Reset subsystem. Refer to Figure 6-1 to see how the BOR interacts with other modules.

The LPBOR is used to monitor the external VDD pin. When too low of a voltage is detected, the device is held in Reset. When this occurs, a register bit (BOR) is changed to indicate that a BOR Reset has occurred. The same bit is set for both the BOR and the LPBOR. Refer to Register 6-2.

6.4.1 ENABLING LPBOR

The LPBOR is controlled by the LPBOR bit of Configuration Words. When the device is erased, the LPBOR module defaults to disabled.

6.4.1.1 LPBOR Module Output

The output of the LPBOR module is a signal indicating whether or not a Reset is to be asserted. This signal is OR’ed together with the Reset signal of the BOR module to provide the generic BOR signal which goes to the PCON register and to the power control block.

6.5 MCLR

The MCLR is an optional external input that can reset the device. The MCLR function is controlled by the MCLRE bit of Configuration Words and the LVP bit of Configuration Words (Table 6-2).

<table>
<thead>
<tr>
<th>MCLRE</th>
<th>LVP</th>
<th>MCLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Enabled</td>
</tr>
<tr>
<td>x</td>
<td>1</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

6.5.1 MCLR ENABLED

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

The device has a noise filter in the MCLR Reset path. The filter will detect and ignore small pulses.

Note: A Reset does not drive the MCLR pin low.

6.5.2 MCLR DISABLED

When MCLR is disabled, the pin functions as a general purpose input and the internal weak pull-up is under software control. See Section 11.3 “PORTA Registers” for more information.

6.6 Watchdog Timer (WDT) Reset

The Watchdog Timer generates a Reset if the firmware does not issue a CLRWDT instruction within the time-out period. The TO and PD bits in the STATUS register are changed to indicate the WDT Reset. See Section 9.0 “Watchdog Timer (WDT)” for more information.

6.7 RESET Instruction

A reset instruction will cause a device Reset. The R bit in the PCON register will be set to ‘0’. See Table 6-4 for default conditions after a reset instruction has occurred.

6.8 Stack Overflow/Underflow Reset

The device can reset when the Stack Overflows or Underflows. The STKOVF or STKUNF bits of the PCON register indicate the Reset condition. These Resets are enabled by setting the STVREN bit in Configuration Words. See Section 3.5.2 “Overflow/Underflow Reset” for more information.

6.9 Programming Mode Exit

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

6.10 Power-up Timer

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

The Power-up Timer is controlled by the PWRTE bit of Configuration Words.

6.11 Start-up Sequence

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

1. Power-up Timer runs to completion (if enabled).
2. MCLR must be released (if enabled).

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

The Power-up Timer runs independently of MCLR Reset. If MCLR is kept low long enough, the Power-up Timer will expire. Upon bringing MCLR high, the device will begin execution immediately (see Figure 6-3). This is useful for testing purposes or to synchronize more than one device operating in parallel.
### FIGURE 6-3: RESET START-UP SEQUENCE

<table>
<thead>
<tr>
<th>VDD</th>
<th>Internal POR</th>
<th>Power-Up Timer</th>
<th>MCLR</th>
<th>Internal RESET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **VDD**
- **Internal POR**
- **Power-Up Timer**
- **MCLR**
- **Internal RESET**

**Internal Oscillator**

- **Oscillator**
- **FOSC**

**External Clock (EC)**

- **CLkin**
- **FOSC**
6.12 Determining the Cause of a Reset

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

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

<table>
<thead>
<tr>
<th>STKOVF</th>
<th>STKUNF</th>
<th>RWDT</th>
<th>RMCLR</th>
<th>RI</th>
<th>POR</th>
<th>BOR</th>
<th>TO</th>
<th>PD</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>Power-on Reset</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>0</td>
<td>x</td>
<td>Illegal, TO is set on POR</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>Illegal, PD is set on POR</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>u</td>
<td>1</td>
<td>1</td>
<td>u</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Brown-out Reset</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>0</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>0</td>
<td>u</td>
<td>WDT Reset</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>0</td>
<td>0</td>
<td>WDT Wake-up from Sleep</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>1</td>
<td>0</td>
<td>Interrupt Wake-up from Sleep</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>0</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>MCLR Reset during normal operation</td>
</tr>
<tr>
<td>u</td>
<td>u</td>
<td>u</td>
<td>0</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>1</td>
<td>0</td>
<td>MCLR Reset during Sleep</td>
</tr>
<tr>
<td>1</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>RESET Instruction Executed</td>
</tr>
<tr>
<td>u</td>
<td>1</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>Stack Overflow Reset (STVREN = 1)</td>
</tr>
</tbody>
</table>

**TABLE 6-4: RESET CONDITION FOR SPECIAL REGISTERS**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Program Counter</th>
<th>STATUS Register</th>
<th>PCON Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power-on Reset</td>
<td>0000h</td>
<td>---1 1000</td>
<td>00-- 110x</td>
</tr>
<tr>
<td>MCLR Reset during normal operation</td>
<td>0000h</td>
<td>---u uuuu</td>
<td>uuu-- 0uuu</td>
</tr>
<tr>
<td>MCLR Reset during Sleep</td>
<td>0000h</td>
<td>---1 0uuu</td>
<td>uuu-- 0uuu</td>
</tr>
<tr>
<td>WDT Reset</td>
<td>0000h</td>
<td>---0 uuuu</td>
<td>uuu-- uuuu</td>
</tr>
<tr>
<td>WDT Wake-up from Sleep</td>
<td>PC + 1</td>
<td>---0 0uuu</td>
<td>uuu-- uuuu</td>
</tr>
<tr>
<td>Brown-out Reset</td>
<td>0000h</td>
<td>---1 1uuu</td>
<td>00-- 11u0</td>
</tr>
<tr>
<td>Interrupt Wake-up from Sleep</td>
<td>PC + 1(1)</td>
<td>---1 0uuu</td>
<td>uuu-- uuuu</td>
</tr>
<tr>
<td>RESET Instruction Executed</td>
<td>0000h</td>
<td>---u uuuu</td>
<td>uuu-- u0uuu</td>
</tr>
<tr>
<td>Stack Overflow Reset (STVREN = 1)</td>
<td>0000h</td>
<td>---u uuuu</td>
<td>1u-- uuuu</td>
</tr>
<tr>
<td>Stack Underflow Reset (STVREN = 1)</td>
<td>0000h</td>
<td>---u uuuu</td>
<td>u1-- uuuu</td>
</tr>
</tbody>
</table>

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

Note 1: When the wake-up is due to an interrupt and the Global Interrupt Enable 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.
6.13 Power Control (PCON) Register

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

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

The PCON register bits are shown in Register 6-2.

6.14 Register Definitions: Power Control

REGISTER 6-2: PCON: POWER CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W/HS-0/q</th>
<th>R/W/HS-0/q</th>
<th>U-0</th>
<th>R/W/HC-1/q</th>
<th>R/W/HC-1/q</th>
<th>R/W/HC-1/q</th>
<th>R/W/HC-q/u</th>
<th>R/W/HC-q/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>STKOVF</td>
<td>STKUNF</td>
<td>--</td>
<td>RWDT</td>
<td>RMCLR</td>
<td>RI</td>
<td>POR</td>
<td>BOR</td>
</tr>
</tbody>
</table>

Legend:

- HC = Bit is cleared by hardware
- HS = Bit is set by hardware
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- u = Bit is unchanged
- x = Bit is unknown
- '1' = Bit is set
- '0' = Bit is cleared
- q = Value depends on condition

bit 7

STKOVF: Stack Overflow Flag bit

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

bit 6

STKUNF: Stack Underflow Flag bit

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

bit 5

Unimplemented: Read as '0'

bit 4

RWDT: Watchdog Timer Reset Flag bit

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

bit 3

RMCLR: MCLR Reset Flag bit

1 = A MCLR Reset has not occurred or set by firmware
0 = A MCLR Reset has occurred (cleared by hardware)

bit 2

RI: RESET Instruction Flag bit

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

bit 1

POR: Power-on Reset Status bit

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

bit 0

BOR: Brown-out Reset Status bit

1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Power-on Reset or Brown-out Reset occurs)
### TABLE 6-5: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BORCON</td>
<td>SBOREN</td>
<td>BORFS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BORRDY 46</td>
</tr>
<tr>
<td>PCON</td>
<td>STKOVF</td>
<td>STKUNF</td>
<td>—</td>
<td>RWDT</td>
<td>RMCLR</td>
<td>RI</td>
<td>POR</td>
<td>BOR</td>
<td>50</td>
</tr>
<tr>
<td>STATUS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TO</td>
<td>PD</td>
<td>Z</td>
<td>DC</td>
<td>C</td>
<td>13</td>
</tr>
<tr>
<td>WDTCON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SWDTEN 66</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented bit, reads as ‘0’. Shaded cells are not used by Resets.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bits</th>
<th>Bit -/7</th>
<th>Bit -/6</th>
<th>Bit 13/5</th>
<th>Bit 12/4</th>
<th>Bit 11/3</th>
<th>Bit 10/2</th>
<th>Bit 9/1</th>
<th>Bit 8/0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG1</td>
<td>13:8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>CP</td>
<td>MCLRE</td>
<td>PWRTE</td>
<td>WDTE&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>CONFIG2</td>
<td>13:8</td>
<td>—</td>
<td>—</td>
<td>LVP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

Legend: — = unimplemented location, read as ‘0’. Shaded cells are not used by Resets.
7.0 INTERRUPTS

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

This chapter contains the following information for Interrupts:

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

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

A block diagram of the interrupt logic is shown in Figure 7-1.

FIGURE 7-1: INTERRUPT LOGIC

[Block diagram image]
7.1 Operation

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

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

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

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

- Current prefetched instruction is flushed
- GIE bit is cleared
- Current Program Counter (PC) is pushed onto the stack
- Critical registers are automatically saved to the shadow registers (See “Section 7.5 “Automatic Context Saving”.”)
- PC is loaded with the interrupt vector 0004h

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

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

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

Note 1: Individual interrupt flag bits are set, regardless of the state of any other enable bits.

2: All interrupts will be ignored while the GIE bit is cleared. Any interrupt occurring while the GIE bit is clear will be serviced when the GIE bit is set again.

7.2 Interrupt Latency

Interrupt latency is defined as the time from when the interrupt event occurs to the time code execution at the interrupt vector begins. The latency for synchronous interrupts is three or four instruction cycles. For asynchronous interrupts, the latency is three to five instruction cycles, depending on when the interrupt occurs. See Figure 7-2 and Figure 7-3 for more details.
FIGURE 7-2: INTERRUPT LATENCY

- **Fosc**
- **CLKR**
- **Interrupt**
- **GIE**
- **PC**
- **Execute**

1. **1 Cycle Instruction at PC**
   - Inst(PC)
   - NOP
   - NOP
   - Inst(0004h)

2. **2 Cycle Instruction at PC**
   - Inst(PC)
   - NOP
   - NOP
   - Inst(0004h)

3. **3 Cycle Instruction at PC**
   - INST(PC)
   - NOP
   - NOP
   - NOP
   - Inst(0004h)
   - Inst(0005h)

**Interrupt Sampled during Q1**

- **New PC/PC+1**
- **FSR ADDR**
- **PC-1**
- **PC+1**

**Execute**

- **FSR ADDR**
- **PC-1**
- **PC+1**
- **PC+2**
- **0004h**
- **0005h**
FIGURE 7-3: INT PIN INTERRUPT TIMING

<table>
<thead>
<tr>
<th>Fosc</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKOUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT pin</td>
<td>(3)</td>
<td></td>
<td></td>
<td></td>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTF</td>
<td>(1)</td>
<td>(4)</td>
<td></td>
<td></td>
<td>Interrupt Latency (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

INSTRUCTION FLOW

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction Fetched</th>
<th>Instruction Executed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inst (PC)</td>
<td>Inst (PC – 1)</td>
</tr>
<tr>
<td></td>
<td>Inst (PC + 1)</td>
<td>Inst (PC)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Forced NOP</td>
</tr>
<tr>
<td>0004h</td>
<td>Inst (0004h)</td>
<td>Inst (0005h)</td>
</tr>
</tbody>
</table>

Note 1: INTF flag is sampled here (every Q1).
2: Asynchronous interrupt latency = 3-5 Tcy. Synchronous latency = 3-4 Tcy, where Tcy = instruction cycle time. Latency is the same whether Inst (PC) is a single cycle or a 2-cycle instruction.
3: For minimum width of INT pulse, refer to AC specifications in Section 21.0 "Electrical Specifications".
4: INTF is enabled to be set any time during the Q4-Q1 cycles.
7.3  Interrupts During Sleep

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

On waking from Sleep, if the GIE bit is also set, the processor will branch to the interrupt vector. Otherwise, the processor will continue executing instructions after the SLEEP instruction. The instruction directly after the SLEEP instruction will always be executed before branching to the ISR. Refer to Section 8.0 “Power-down Mode (Sleep)” for more details.

7.4  INT Pin

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

7.5  Automatic Context Saving

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

- W register
- STATUS register (except for TO and PD)
- BSR register
- FSR registers
- PCLATH register

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

REGISTER 7-1: INTCON: INTERRUPT CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>GIE: Global Interrupt Enable bit</td>
<td>1 = Enables all active interrupts, 0 = Disables all interrupts</td>
</tr>
<tr>
<td>6</td>
<td>PEIE: Peripheral Interrupt Enable bit</td>
<td>1 = Enables all active peripheral interrupts, 0 = Disables all peripheral interrupts</td>
</tr>
<tr>
<td>5</td>
<td>TMR0IE: Timer0 Overflow Interrupt Enable bit</td>
<td>1 = Enables the Timer0 interrupt, 0 = Disables the Timer0 interrupt</td>
</tr>
<tr>
<td>4</td>
<td>INTE: INT External Interrupt Enable bit</td>
<td>1 = Enables the INT external interrupt, 0 = Disables the INT external interrupt</td>
</tr>
<tr>
<td>3</td>
<td>IOCIE: Interrupt-on-Change Enable bit</td>
<td>1 = Enables the interrupt-on-change, 0 = Disables the interrupt-on-change</td>
</tr>
<tr>
<td>2</td>
<td>TMR0IF: Timer0 Overflow Interrupt Flag bit</td>
<td>1 = TMR0 register has overflowed, 0 = TMR0 register did not overflow</td>
</tr>
<tr>
<td>1</td>
<td>INTF: INT External Interrupt Flag bit</td>
<td>1 = The INT external interrupt occurred, 0 = The INT external interrupt did not occur</td>
</tr>
<tr>
<td>0</td>
<td>IOCIF: Interrupt-on-Change Interrupt Flag bit(1)</td>
<td>1 = When at least one of the interrupt-on-change pins changed state, 0 = None of the interrupt-on-change pins have changed state</td>
</tr>
</tbody>
</table>

Legend:

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

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

**Note:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE, of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
### REGISTER 7-2: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADIE</td>
<td>SSPIE</td>
</tr>
</tbody>
</table>

**Legend:**

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

- **bit 7** **Unimplemented:** Read as ‘0’
- **bit 6**  **ADIE:** Analog-to-Digital Converter (ADC) Interrupt Enable bit
  - 1 = Enables the ADC interrupt
  - 0 = Disables the ADC interrupt
- **bit 5-4** **Unimplemented:** Read as ‘0’
- **bit 3**  **SSPIE:** Synchronous Serial Port (MSSP) Interrupt Enable bit
  - 1 = Enables the MSSP interrupt
  - 0 = Disables the MSSP interrupt
- **bit 2-0** **Unimplemented:** Read as ‘0’

**Note:** Bit PEIE of the INTCON register must be set to enable any peripheral interrupt.
### REGISTER 7-3: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCLIE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

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

- **bit 7**
- **Unimplemented**: Read as ‘0’

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

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

**Note:** Bit PEIE of the INTCON register must be set to enable any peripheral interrupt.
REGISTER 7-4:  PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADIF</td>
<td></td>
<td></td>
<td>SSPIF</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 7 ADIF: ADC Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 6 Unimplemented: Read as '0'

bit 5-4 Unimplemented: Read as '0'

bit 3 SSPIF: Synchronous Serial Port (MSSP) Interrupt Flag bit
1 = Interrupt is pending
0 = Interrupt is not pending

bit 2-0 Unimplemented: Read as '0'

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as '0'
u = Bit is unchanged  x = Bit is unknown  -n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set  '0' = Bit is cleared

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE, of the INTCON register. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
### REGISTER 7-5: PIR2: PERIPHERAL INTERRUPT REQUEST REGISTER 2

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

#### TABLE 7-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>57</td>
</tr>
<tr>
<td>OPTION_REG</td>
<td>WPUEN</td>
<td>INTEDG</td>
<td>TMR0CS</td>
<td>TMR0SE</td>
<td>PSA</td>
<td>PS&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>132</td>
</tr>
<tr>
<td>PIE1</td>
<td></td>
<td>ADIE</td>
<td></td>
<td></td>
<td>SSPIE</td>
<td></td>
<td></td>
<td></td>
<td>58</td>
</tr>
<tr>
<td>PIE2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCLIE</td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>PIR1</td>
<td></td>
<td>ADIF</td>
<td></td>
<td></td>
<td>SSPIF</td>
<td></td>
<td></td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>PIR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BCLIF</td>
<td></td>
<td></td>
<td></td>
<td>61</td>
</tr>
</tbody>
</table>

#### Legend:
- **—** = unimplemented location, read as ‘0’. Shaded cells are not used by Interrupts.
8.0 POWER-DOWN MODE (SLEEP)

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

Upon entering Sleep mode, the following conditions exist:

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

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

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

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

I/O pins that are high-impedance inputs should be pulled to Vdd or Vss externally to avoid switching currents caused by floating inputs.

Examples of internal circuitry that might be sourcing current include the FVR module. See Section 13.0 “Fixed Voltage Reference (FVR)” for more information on this module.

8.1 Wake-up from Sleep

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

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

The first three events will cause a device Reset. The last three events are considered a continuation of program execution. To determine whether a device Reset or wake-up event occurred, refer to Section 6.12 “Determining the Cause of a Reset”.

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

The WDT is cleared when the device wakes up from Sleep, regardless of the source of wake-up.
8.1.1 WAKE-UP USING INTERRUPTS

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

- If the interrupt occurs before the execution of a SLEEP instruction:
  - SLEEP instruction will execute as a NOP.
  - WDT and WDT prescaler will not be cleared
  - TO bit of the STATUS register will not be set
  - PD bit of the STATUS register will not be cleared.

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

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

FIGURE 8-1: WAKE-UP FROM SLEEP THROUGH INTERRUPT

TABLE 8-1: SUMMARY OF REGISTERS ASSOCIATED WITH POWER-DOWN MODE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>57</td>
</tr>
<tr>
<td>IOCAF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCAF5</td>
<td>IOCAF4</td>
<td>IOCAF3</td>
<td>IOCAF2</td>
<td>IOCAF1</td>
<td>IOCAF0</td>
</tr>
<tr>
<td>IOCAN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCAN5</td>
<td>IOCAN4</td>
<td>IOCAN3</td>
<td>IOCAN2</td>
<td>IOCAN1</td>
<td>IOCAN0</td>
</tr>
<tr>
<td>IOCAP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>IOCAP5</td>
<td>IOCAP4</td>
<td>IOCAP3</td>
<td>IOCAP2</td>
<td>IOCAP1</td>
<td>IOCAP0</td>
</tr>
<tr>
<td>PIE1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADIE</td>
<td>—</td>
<td>—</td>
<td>SSPIE</td>
<td>58</td>
</tr>
<tr>
<td>PIE2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BCLIE</td>
<td>59</td>
</tr>
<tr>
<td>PIR1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADIF</td>
<td>60</td>
</tr>
<tr>
<td>PIR2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BCLIF</td>
</tr>
<tr>
<td>STATUS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TO</td>
<td>PD</td>
<td>Z</td>
<td>13</td>
</tr>
<tr>
<td>WDTCON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WDTPS&lt;4:0&gt;</td>
<td>66</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used in Power-Down mode.
9.0 WATCHDOG TIMER (WDT)

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

The WDT has the following features:

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

**FIGURE 9-1: WATCHDOG TIMER BLOCK DIAGRAM**
9.1 Independent Clock Source

The WDT derives its time base from the 31 kHz LFINTOSC internal oscillator. Time intervals in this chapter are based on a nominal interval of 1 ms. See Section 21.0 “Electrical Specifications” for the LFINTOSC tolerances.

9.2 WDT Operating Modes

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

9.2.1 WDT IS ALWAYS ON

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

9.2.2 WDT IS OFF IN SLEEP

When the WDTE bits of Configuration Words are set to '10', the WDT is on, except in Sleep.
WDT protection is not active during Sleep.

9.2.3 WDT CONTROLLED BY SOFTWARE

When the WDTE bits of Configuration Words are set to '01', the WDT is controlled by the SWDTEN bit of the WDTCON register.
WDT protection is unchanged by Sleep. See Table 9-1 for more details.

9.3 Time-out Period

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

9.4 Clearing the WDT

The WDT is cleared when any of the following conditions occur:
• Any Reset
• CLRWDT instruction is executed
• Device enters Sleep
• Device wakes up from Sleep
• Oscillator fail
• WDT is disabled
See Table 9-2 for more information.

9.5 Operation During Sleep

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

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 TO and PD bits in the STATUS register are changed to indicate the event. The RWDT bit in the PCON register can also be used. See Section 3.0 “Memory Organization” for more information.

---

**TABLE 9-1: WDT OPERATING MODES**

<table>
<thead>
<tr>
<th>WDTE&lt;1:0&gt;</th>
<th>SWDTEN</th>
<th>Device Mode</th>
<th>WDT Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>X</td>
<td>X</td>
<td>Active</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>Awake</td>
<td>Active</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sleep</td>
<td>Disabled</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>X</td>
<td>Active</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Disabled</td>
</tr>
<tr>
<td>00</td>
<td>X</td>
<td>X</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

**TABLE 9-2: WDT CLEARING CONDITIONS**

<table>
<thead>
<tr>
<th>Conditions</th>
<th>WDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDTE&lt;1:0&gt; = 00</td>
<td>Cleared</td>
</tr>
<tr>
<td>WDTE&lt;1:0&gt; = 01 and SWDTEN = 0</td>
<td></td>
</tr>
<tr>
<td>WDTE&lt;1:0&gt; = 10 and enter Sleep</td>
<td></td>
</tr>
<tr>
<td>CLRWDT Command</td>
<td></td>
</tr>
<tr>
<td>Oscillator Fail Detected</td>
<td></td>
</tr>
<tr>
<td>Exit Sleep + System Clock = INTOSC, EXTCCLK</td>
<td>Unaffected</td>
</tr>
<tr>
<td>Change INTOSC divider (IRCF bits)</td>
<td></td>
</tr>
</tbody>
</table>
9.6 Register Definitions: Watchdog Control

REGISTER 9-1: WDTCON: WATCHDOG TIMER CONTROL REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/1</th>
<th>R/W-1/1</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-0/0</th>
<th>SWDTEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>WDTPS&lt;4:0&gt;</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-6  Unimplemented: Read as ‘0’

bit 5-1  WDTPS<4:0>: Watchdog Timer Period Select bits\(^{(1)}\)
Bit Value = Prescale Rate
11111 = Reserved. Results in minimum interval (1:32)
11110 = Reserved. Results in minimum interval (1:32)
10110 = 1:32768 (Interval 1s nominal)
10101 = 1:16384 (Interval 2 ms nominal) (Reset value)
10100 = 1:8192 (Interval 4 ms nominal)
10010 = 1:4096 (Interval 8 ms nominal)
10001 = 1:2048 (Interval 16 ms nominal)
10000 = 1:1024 (Interval 32 ms nominal)
01111 = 1:512 (Interval 16 ms nominal)
01110 = 1:256 (Interval 8 ms nominal)
01101 = 1:128 (Interval 4 ms nominal)
01100 = 1:64 (Interval 2 ms nominal)
01011 = 1:32 (Interval 1 ms nominal)
01010 = 1:16 (Interval 2 ms nominal)
01001 = 1:8 (Interval 4 ms nominal)
01000 = 1:4 (Interval 8 ms nominal)
00111 = 1:2 (Interval 16 ms nominal)
00110 = 1:1 (Interval 32 ms nominal)
00101 = 1:0.5 (Interval 64 ms nominal)
00100 = 1:0.25 (Interval 128 ms nominal)
00011 = 1:0.128 (Interval 256 ms nominal)
00010 = 1:0.64 (Interval 512 ms nominal)
00001 = 1:0.32 (Interval 1 ms nominal)
00000 = 1:0.16 (Interval 2 ms nominal)

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

Note 1: Times are approximate. WDT time is based on 31 kHz LFINTOSC.
### TABLE 9-3: SUMMARY OF REGISTERS ASSOCIATED WITH WATCHDOG TIMER

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSCCON</td>
<td>PLL</td>
<td>—</td>
<td>IRCF&lt;3:0&gt;</td>
<td>—</td>
<td>SCS&lt;1:0&gt;</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCON</td>
<td>STKOVF</td>
<td>STKUNF</td>
<td>—</td>
<td>RWDT</td>
<td>RMCLR</td>
<td>RI</td>
<td>POR</td>
<td>BOR</td>
<td>50</td>
</tr>
<tr>
<td>STATUS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TO</td>
<td>PD</td>
<td>Z</td>
<td>DC</td>
<td>C</td>
<td>13</td>
</tr>
<tr>
<td>WDTCON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WDTPS&lt;4:0&gt;</td>
<td>—</td>
<td>SWDTEN</td>
<td>66</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:  
- \( \times \) = unknown, \( \_ \) = unchanged, \( - \) = unimplemented locations read as '0'. Shaded cells are not used by Watchdog Timer.

### TABLE 9-4: SUMMARY OF CONFIGURATION WORD WITH WATCHDOG TIMER

<table>
<thead>
<tr>
<th>Name</th>
<th>Bits</th>
<th>Bit -7</th>
<th>Bit -6</th>
<th>Bit 13/5</th>
<th>Bit 12/4</th>
<th>Bit 11/3</th>
<th>Bit 10/2</th>
<th>Bit 9/1</th>
<th>Bit 8/0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG1</td>
<td>13:8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CLKOUTEN</td>
<td>BOREN&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>CP</td>
<td>MCLRE</td>
<td>PWRE</td>
<td>WDTE&lt;1:0&gt;</td>
<td>—</td>
<td>FOSC&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

Legend:  
- \( - \) = unimplemented location, read as '0'. Shaded cells are not used by Watchdog Timer.
10.0  FLASH PROGRAM MEMORY CONTROL

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

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

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

The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump.

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

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

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

**Note 1:** Code protection of the entire Flash program memory array is enabled by clearing the CP bit of Configuration Words.

10.1  PMADRL and PMADRH Registers

The PMADRH:PMADRL register pair can address up to a maximum of 16K words of program memory. When selecting a program address value, the MSB of the address is written to the PMADRH register and the LSB is written to the PMADRL register.

10.1.1  PMCON1 AND PMCON2 REGISTERS

PMCON1 is the control register for Flash program memory accesses.

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

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

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

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

It is important to understand the Flash program memory structure for erase and programming operations. Flash program memory is arranged in rows. A row consists of a fixed number of 14-bit program memory words. A row is the minimum size that can be erased by user software. After a row has been erased, the user can reprogram all or a portion of this row. Data to be written into the program memory row is written to 14-bit wide data write latches. These write latches are not directly accessible to the user, but may be loaded via sequential writes to the PMDATH:PMDATL register pair.

Note: If the user wants to modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase. Then, new data and retained data can be written into the write latches to reprogram the row of Flash program memory. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations.

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

TABLE 10-1: FLASH MEMORY ORGANIZATION BY DEVICE

<table>
<thead>
<tr>
<th>Device</th>
<th>Row Erase (words)</th>
<th>Write Latches (words)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC12LF1552</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

10.2.1 READING THE FLASH PROGRAM MEMORY

To read a program memory location, the user must:

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

Once the read control bit is set, the program memory Flash controller will use the second instruction cycle to read the data. This causes the second instruction immediately following the “BSF PMCON1, RD” instruction to be ignored. The data is available in the very next cycle, in the PMDATH:PMDATL register pair; therefore, it can be read as two bytes in the following instructions.

Note: The two instructions following a program memory read are required to be NOPs. This prevents the user from executing a two-cycle instruction on the next instruction after the RD bit is set.
**EXAMPLE 10-1: FLASH PROGRAM MEMORY READ**

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

```
BANKSEL PMADRL ; Select Bank for PMCON registers
MOVLW PROG_ADDR_LO ;
MOVWF PMADRL ; Store LSB of address
MOVLW PROG_ADDR_HI ;
MOVWF PMADR ; Store MSB of address
BCF PMCON1,CFGS ; Do not select Configuration Space
BSF PMCON1,RD ; Initiate read
NOP ; Ignored (Figure 10-2)
NOP ; Ignored (Figure 10-2)

MOVF PMDATL,W ; Get LSB of word
MOVF PMDATA_LO ; Store in user location
MOVF PMDATH,W ; Get MSB of word
MOVF PROG_DATA_HI ; Store in user location
```
10.2.2 FLASH MEMORY UNLOCK SEQUENCE

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

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

The unlock sequence consists of the following steps:

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

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

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

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

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

See Example 10-2.

After the "BSF PMCON1, WR" instruction, the processor requires two cycles to set up the erase operation. The user must place two NOP instructions after the WR bit is set. The processor will halt internal operations for the typical 2 ms erase time. This is not Sleep mode as the clocks and peripherals will continue to run. After the erase cycle, the processor will resume operation with the third instruction after the PMCON1 write instruction.
EXAMPLE 10-2: ERASING ONE ROW OF PROGRAM MEMORY

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

BCF INTCON,GIE ; Disable ints so required sequences will execute properly
BANKSEL PMADRL
MOVF ADDRL,W ; Load lower 8 bits of erase address boundary
MOVF ADDRH,W ; Load upper 6 bits of erase address boundary
BCF PMCON1,CFGS ; Not configuration space
BSF PMCON1,FREE ; Specify an erase operation
BSF PMCON1,WREN ; Enable writes

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

BCF PMCON1,WREN ; Disable writes
BSF INTCON,GIE ; Enable interrupts
10.2.4 WRITING TO FLASH PROGRAM MEMORY

Program memory is programmed using the following steps:

1. Load the address in PMADRH:PMADRL of the row to be programmed.
2. Load each write latch with data.
3. Initiate a programming operation.
4. Repeat steps 1 through 3 until all data is written.

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

Program memory can be written one or more words at a time. The maximum number of words written at one time is equal to the number of write latches. See Figure 10-5 (row writes to program memory with 16 write latches) for more details.

The write latches are aligned to the Flash row address boundary defined by the upper eleven bits of PMADRH:PMADRL, (PMADRH<6:0>:PMADRL<7:4>) with the lower four bits of PMADRL, (PMADRL<3:0>) determining the write latch being loaded. Write operations do not cross these boundaries. At the completion of a program memory write operation, the data in the write latches is reset to contain 0x3FFF.

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

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

1. Set the WREN bit of the PMCON1 register.
2. Clear the CFGS bit of the PMCON1 register.
3. Set the LWLO bit of the PMCON1 register. When the LWLO bit of the PMCON1 register is '1', the write sequence will only load the write latches and will not initiate the write to Flash program memory.
4. Load the PMADRH:PMADRL register pair with the address of the location to be written.
5. Load the PMDATH:PMDATL register pair with the program memory data to be written.
6. Execute the unlock sequence (Section 10.2.2 “Flash Memory Unlock Sequence”). The write latch is now loaded.
7. Increment the PMADRH:PMADRL register pair to point to the next location.
8. Repeat steps 5 through 7 until all but the last write latch has been loaded.
9. Clear the LWLO bit of the PMCON1 register. When the LWLO bit of the PMCON1 register is '0', the write sequence will initiate the write to Flash program memory.
10. Load the PMDATH:PMDATL register pair with the program memory data to be written.
11. Execute the unlock sequence (Section 10.2.2 “Flash Memory Unlock Sequence”). The entire program memory latch content is now written to Flash program memory.

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

An example of the complete write sequence is shown in Example 10-3. The initial address is loaded into the PMADRH:PMADRL register pair; the data is loaded using indirect addressing.
FIGURE 10-5: BLOCK WRITES TO FLASH PROGRAM MEMORY WITH 16 WRITE LATCHES
FIGURE 10-6: FLASH PROGRAM MEMORY WRITE FLOWCHART

1. Start Write Operation
2. Disable Interrupts (GIE = 0)
3. Select Program or Config. Memory (CFGS)
4. Select Row Address (PMADRH:PMADRL)
5. Select Write Operation (FREE = 0)
6. Load Write Latches Only (LWLO = 1)
7. Determine number of words to be written into Program or Configuration Memory. The number of words cannot exceed the number of words per row. (word_cnt)
8. Enable Write/Erase Operation (WREN = 1)
9. Load the value to write (PMDATH:PMDATL)
10. Update the word counter (word_cnt--)
11. No delay when writing to Program Memory Latches
12. Increment Address (PMADRH:PMADRL++)
13. Last word to write ?
   - Yes
     14. Unlock Sequence Figure 10-3
     15. Write Latches to Flash (LWLO = 0)
   - No
     16. Unlock Sequence Figure 10-3
     17. CPU stalls while Write operation completes (2ms typical)
     18. Disable Write/Erase Operation (WREN = 0)
     19. Load Write Latches Only (LWLO = 1)
     20. Re-enable Interrupts (GIE = 1)
     21. End Write Operation
EXAMPLE 10-3: WRITING TO FLASH PROGRAM MEMORY

; This write routine assumes the following:
; 1. 32 bytes of data are loaded, starting at the address in DATA_ADDR
; 2. Each word of data to be written is made up of two adjacent bytes in DATA_ADDR,
;    stored in little endian format
; 3. A valid starting address (the least significant bits = 00000) is loaded in ADDRH:ADDRL
; 4. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F (common RAM)

BCF INTCON,GIE ; Disable ints so required sequences will execute properly
BANKSEL PMADRH ; Bank 3
MOVF ADDRH,W ; Load initial address
MOVWF PMADRH ;
MOVF ADDRL,W ;
MOVWF PMADRL ;
MOVLW LOW DATA_ADDR ; Load initial data address
MOVWF FSR0L ;
MOVLW HIGH DATA_ADDR ; Load initial data address
MOVWF FSR0H ;
BCF PMCON1,CFGS ; Not configuration space
BSF PMCON1,WREN ; Enable writes
BSF PMCON1,LWO ; Only Load Write Latches

LOOP
MOVLW FSR0++ ; Load first data byte into lower
MOVWF PMDATL ;
MOVLW FSR0++ ; Load second data byte into upper
MOVWF PMDATH ;
MOVLW PMADRL,W ; Check if lower bits of address are '00000'
XORLW 0x0F ; Check if we're on the last of 16 addresses
ANDLW 0x0F ;
BTFSC STATUS,Z ; Exit if last of 16 words,
GOTO START_WRITE ;

MOVLW 55h ; Start of required write sequence:
MOVWF PMCON2 ; Write 55h
MOVLW 0AAh ;
MOVWF PMCON2 ; Write AAh
BSF PMCON1,WR ; Set WR bit to begin write
NOP ; NOP instructions are forced as processor
NOP ; loads program memory write latches
NOP ;
INCF PMADRL,F ; Still loading latches Increment address
GOTO LOOP ; Write next latches

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

MOVLW 55h ; Start of required write sequence:
MOVWF PMCON2 ; Write 55h
MOVLW 0AAh ;
MOVWF PMCON2 ; Write AAh
BSF PMCON1,WR ; Set WR bit to begin write
NOP ; NOP instructions are forced as processor writes
NOP ; all the program memory write latches simultaneously
NOP ; to program memory.
NOP ; After NOPs, the processor
NOP ; stalls until the self-write process in complete
BCF PMCON1,WREN ; Disable writes
BSF INTCON,GIE ; Enable interrupts
10.3 Modifying Flash Program Memory

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

1. Load the starting address of the row to be modified.
2. Read the existing data from the row into a RAM image.
3. Modify the RAM image to contain the new data to be written into program memory.
4. Load the starting address of the row to be rewritten.
5. Erase the program memory row.
6. Load the write latches with data from the RAM image.
7. Initiate a programming operation.

FIGURE 10-7: FLASH PROGRAM MEMORY MODIFY FLOWCHART

Start
Modify Operation

Read Operation
Figure 10-2

An image of the entire row read must be stored in RAM

Modify Image
The words to be modified are changed in the RAM image

Erase Operation
Figure 10-4

Write Operation
use RAM image
Figure 10-5

End
Modify Operation
10.4 User ID, Device ID and Configuration Word Access

Instead of accessing program memory, the User ID’s, Device ID/Revision ID and Configuration Words can be accessed when CFGS = 1 in the PMCON1 register. This is the region that would be pointed to by PC<15> = 1, but not all addresses are accessible. Different access may exist for reads and writes. Refer to Table 10-2.

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

| TABLE 10-2: USER ID, DEVICE ID AND CONFIGURATION WORD ACCESS (CFGS = 1) |
|----------------------|------------------|------------------|
| Address              | Function         | Read Access      | Write Access |
| 8000h-8003h          | User IDs         | Yes              | Yes           |
| 8006h                | Device ID/Revision ID | Yes             | No            |
| 8007h-8008h          | Configuration Words 1 and 2 | Yes | No |

EXAMPLE 10-4: CONFIGURATION WORD AND DEVICE ID ACCESS

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

```assembly
BANKSEL PMADRL ; Select correct Bank
MOVLW PROG_ADDR_LO ;
MOVF PMADRL,W ; Store LSB of address
CLRF PMADRH ; Clear MSB of address
BSF PMCON1,CFGS ; Select Configuration Space
BCF INTCON,GIE ; Disable interrupts
BSF PMCON1,RD ; Initiate read
NOP ; Executed (See Figure 10-2)
NOP ; Ignored (See Figure 10-2)
BSF INTCON,GIE ; Restore interrupts
MOVF PMDATH,W ; Get MSB of word
MOVF PMDATL,W ; Store in user location
MOVF PROG_DATA_LO ; Store in user location
```
10.5 Write Verify

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

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

- **Start Verify Operation**
- **Read Operation**
  - **PMDAT = RAM image?**
    - **Yes**
      - **Fail Verify Operation**
    - **No**
      - **Last Word?**
        - **Yes**
          - **End Verify Operation**
        - **No**
          - **End Verify Operation**
10.6 Register Definitions: Flash Program Memory Control

REGISTER 10-1: PMDATL: PROGRAM MEMORY DATA LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMDAT&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Note 1: Unimplemented bit, read as ‘1’.

REGISTER 10-2: PMDATH: PROGRAM MEMORY DATA HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMDAT&lt;13:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

REGISTER 10-3: PMADRL: PROGRAM MEMORY ADDRESS LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMADR&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

REGISTER 10-4: PMADRH: PROGRAM MEMORY ADDRESS HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>U-1</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMADR&lt;14:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Note 1: Unimplemented bit, read as ‘1’.
REGISTER 10-5: PMCON1: PROGRAM MEMORY CONTROL 1 REGISTER

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-1</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W/HC-0/0</td>
<td>R/W/HC-x/q(2)</td>
<td>R/W-0/0</td>
<td>R/S/HC-0/0</td>
<td>R/S/HC-0/0</td>
</tr>
<tr>
<td>_-(1)</td>
<td>CFGS</td>
<td>LWLO</td>
<td>FREE</td>
<td>WRERR</td>
<td>WREN</td>
<td>WR</td>
<td>RD</td>
</tr>
</tbody>
</table>

Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **S** = Bit can only be set
- **x** = Bit is unknown
- **-n/n** = Value at POR and BOR/Value at all other Resets

| bit 7 | Unimplemented: Read as ‘1’ |
| bit 6 | **CFGs**: Configuration Select bit |
|      | 1 = Access Configuration, User ID and Device ID Registers |
|      | 0 = Access Flash program memory |
| bit 5 | **LWLO**: Load Write Latches Only bit(3) |
|      | 1 = Only the addressed program memory write latch is loaded/updated on the next WR command |
|      | 0 = The addressed program memory write latch is loaded/updated and a write of all program memory write latches will be initiated on the next WR command |
| bit 4 | **FREE**: Program Flash Erase Enable bit |
|      | 1 = Performs an erase operation on the next WR command (hardware cleared upon completion) |
|      | 0 = Performs an write operation on the next WR command |
| bit 3 | **WRERR**: Program/Erase Error Flag bit |
|      | 1 = Condition indicates an improper program or erase sequence attempt or termination (bit is set automatically on any set attempt (write ‘1’) of the WR bit). |
|      | 0 = The program or erase operation completed normally |
| bit 2 | **WREN**: Program/Erase Enable bit |
|      | 1 = Allows program/erase cycles |
|      | 0 = Inhibits programming/erasing of program Flash |
| bit 1 | **WR**: Write Control bit |
|      | 1 = Initiates a program Flash program/erase operation. |
|      | The operation is self-timed and the bit is cleared by hardware once operation is complete. |
|      | The WR bit can only be set (not cleared) in software. |
|      | 0 = Program/erase operation to the Flash is complete and inactive |
| bit 0 | **RD**: Read Control bit |
|      | 1 = Initiates a program Flash read. Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. |
|      | 0 = Does not initiate a program Flash read |

**Note**

1: Unimplemented bit, read as ‘1’.
2: The WRERR bit is automatically set by hardware when a program memory write or erase operation is started (WR = 1).
3: The LWLO bit is ignored during a program memory erase operation (FREE = 1).
REGISTER 10-6: PMCON2: PROGRAM MEMORY CONTROL 2 REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>Flash Memory Unlock Pattern bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>To unlock writes, a 55h must be written first, followed by an AAh, before setting the WR bit of the PMCON1 register. The value written to this register is used to unlock the writes. There are specific timing requirements on these writes.</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>57</td>
</tr>
<tr>
<td>PMCON1</td>
<td>__(1)</td>
<td>CFGS</td>
<td>LWLO</td>
<td>FREE</td>
<td>WRERR</td>
<td>WREN</td>
<td>WR</td>
<td>RD</td>
<td></td>
</tr>
<tr>
<td>PMCON2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>83</td>
</tr>
<tr>
<td>PMADRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>81</td>
</tr>
<tr>
<td>PMADRH</td>
<td>__(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMDATL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>81</td>
</tr>
<tr>
<td>PMDATH</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>81</td>
</tr>
<tr>
<td>Legend:</td>
<td>— = unimplemented location, read as ‘0’. Shaded cells are not used by Flash program memory.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note 1:</td>
<td>Unimplemented, read as ‘1’.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Bits</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 13/5</th>
<th>Bit 12/4</th>
<th>Bit 11/3</th>
<th>Bit 10/2</th>
<th>Bit 9/1</th>
<th>Bit 8/0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG1</td>
<td>13:8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>CP</td>
<td>MCLRE</td>
<td>PWRTE</td>
<td>WDTE&lt;1:0&gt;</td>
<td>—</td>
<td>FOSC&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>CONFIG2</td>
<td>13:8</td>
<td>—</td>
<td>—</td>
<td>LVP</td>
<td>—</td>
<td>LPBOR</td>
<td>BORV</td>
<td>STVREN</td>
<td>—</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>Legend:</td>
<td>— = unimplemented location, read as ‘0’. Shaded cells are not used by Flash program memory.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11.0 I/O PORTS

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

- TRISx registers (data direction)
- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)

Some ports may have one or more of the following additional registers. These registers are:

- ANSELx (analog select)
- WPUx (weak pull-up)

In general, when a peripheral is enabled on a port pin, that pin cannot be used as a general purpose output. However, the pin can still be read.

### TABLE 11-1: PORT AVAILABILITY PER DEVICE

<table>
<thead>
<tr>
<th>Device</th>
<th>PORTA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC12LF1552</td>
<td>●</td>
</tr>
</tbody>
</table>

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 the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

Ports that support analog inputs have an associated ANSELx register. When an ANSEL bit is set, the digital input buffer associated with that bit is disabled. Disabling the input buffer prevents analog signal levels on the pin between a logic high and low from causing excessive current in the logic input circuitry. A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 11-1.

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

#### EXAMPLE 11-1: INITIALIZING PORTA

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

```assembly
BANKSEL PORTA ;
CLRF PORTA ;Init PORTA
BANKSEL LATA ;Data Latch
CLRF LATA ;
BANKSEL ANSELA ;
CLRF ANSELA ;digital I/O
BANKSEL TRISA ;
MOVWF TRISA ;and set RA<2:0> as outputs
MOVWF TRISA ;
```

```
11.1 Alternate Pin Function

The Alternate Pin Function Control (APFCON) register is used to steer specific peripheral input and output functions between different pins. The APFCON register is shown in Register 11-1. For this device family, the following functions can be moved between different pins.

- SDO
- SSS
- SDA/SDI

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

11.2 Register Definitions: Alternate Pin Function Control

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

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>SDOSEL</td>
<td>SSSEL</td>
<td>SDSEL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Legend:

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

bit 7 Unimplemented: Read as ‘0’

bit 6 SDOSEL: Pin Selection bit
- 1 = SDO function is on RA4
- 0 = SDO function is on RA0

bit 5 SSSEL: Pin Selection bit
- 1 = SSS function is on RA0
- 0 = SSS function is on RA3

bit 4 SDSEL: Pin Selection bit
- 1 = SDA/SDI function is on RA3(1)
- 0 = SDA/SDI function is on RA2

bit 3-0 Unimplemented: Read as ‘0’

Note 1: The MSSP module has the ability to output low on RA3 when it is used as SDA/SDI.

(1) Note: The MSSP module has the ability to output low on RA3 when it is used as SDA/SDI.
11.3 PORTA Registers

11.3.1 DATA REGISTER

PORTA is a 6-bit wide, bidirectional port. The corresponding data direction register is TRISA (Register 11-3). Setting a TRISA bit (1) will make the corresponding PORTA pin an input (i.e., disable the output driver). Clearing a TRISA bit (0) will make the corresponding PORTA pin an output (i.e., enables output driver and puts the contents of the output latch on the selected pin). The exception is RA3, which is input-only and its TRIS bit will always read as '1'. Example 11-1 shows how to initialize an I/O port.

Reading the PORTA register (Register 11-2) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATA).

11.3.2 DIRECTION CONTROL

The TRISA register (Register 11-3) controls the PORTA pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISA register are maintained set when using them as analog inputs. I/O pins configured as analog input always read '0'.

11.3.3 ANSELA REGISTER

The ANSELA register (Register 11-5) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELA bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELA bits has no effect on digital output functions. A pin with TRIS clear and ANSEL set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

Note: The ANSELA bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to '0' by user software.

11.3.4 PORTA FUNCTIONS AND OUTPUT PRIORITIES

Each PORTA pin is multiplexed with other functions. The pins, their combined functions and their output priorities are shown in Table 11-2.

When multiple outputs are enabled, the actual pin control goes to the peripheral with the highest priority. Analog input functions, such as ADC inputs, are not shown in the priority lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown in Table 11-2.

### TABLE 11-2: PORTA OUTPUT PRIORITY

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function Priority^{(1)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0</td>
<td>ICSPDAT SDO^{(2)} SS^{(3)} RA0</td>
</tr>
<tr>
<td>RA1</td>
<td>SCL SCK RA1</td>
</tr>
<tr>
<td>RA2</td>
<td>ADOUT SDA^{(2)} SDI^{(2)} RA2</td>
</tr>
<tr>
<td>RA3</td>
<td>SDA^{(3)} SDI^{(3)} SS^{(4)} RA3</td>
</tr>
<tr>
<td>RA4</td>
<td>CLKOUT SDO^{(3)} ADGRDA RA4</td>
</tr>
<tr>
<td>RA5</td>
<td>ADGRDB RA5</td>
</tr>
</tbody>
</table>

Note 1: Priority listed from highest to lowest.
Note 2: Default pin (see APFCON register).
Note 3: Alternate pin (see APFCON register).
### 11.4 Register Definitions: PORTA

#### REGISTER 11-2: PORTA: PORTA REGISTER

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5-0</th>
<th>RA&lt;5:0&gt;</th>
<th>PORTA I/O Value bits(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7-6</td>
<td>bit 5-0</td>
<td>RA&lt;5:0&gt;</td>
<td>PORTA I/O Value bits(1)</td>
</tr>
</tbody>
</table>

#### REGISTER 11-3: TRISA: PORTA TRI-STATE REGISTER

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 5-4</th>
<th>TRISA&lt;5:4&gt;</th>
<th>PORTA Tri-State Control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7-6</td>
<td>bit 5-4</td>
<td>TRISA&lt;5:4&gt;</td>
<td>PORTA Tri-State Control bit</td>
</tr>
</tbody>
</table>

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

**Note 1:** Writes to PORTA are actually written to corresponding LATA register. Reads from PORTA register is return of actual I/O pin values.

**Note 1:** Unimplemented, read as ‘1’.
## REGISTER 11-4: LATA: PORTA DATA LATCH REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>U-0</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>LATA5</td>
<td>LATA4</td>
<td>—</td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
</table>

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

**bit 7-6**: **Unimplemented**: Read as ‘0’

**bit 5-4**: **LATA<5:4>:** RA<5:4> Output Latch Value bits(1)

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

**bit 2-0**: **LATA<2:0>:** RA<2:0> Output Latch Value bits(1)

**Note 1:** Writes to PORTA are actually written to corresponding LATA register. Reads from PORTA register is return of actual I/O pin values.

## REGISTER 11-5: ANSELA: PORTA ANALOG SELECT REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>U-0</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>ANSA5</td>
<td>ANSA4</td>
<td>—</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
</table>

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

**bit 7-6**: **Unimplemented**: Read as ‘0’

**bit 5-4**: **ANSA<5:4>:** Analog Select between Analog or Digital Function on pins RA<5:4>, respectively

- **1**: Analog input. Pin is assigned as analog input(1). Digital input buffer disabled.
- **0**: Digital I/O. Pin is assigned to port or digital special function.

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

**bit 2-0**: **ANSA<2:0>:** Analog Select between Analog or Digital Function on pins RA<2:0>, respectively

- **1**: Analog input. Pin is assigned as analog input(1). Digital input buffer disabled.
- **0**: Digital I/O. Pin is assigned to port or digital special function.

**Note 1:** When setting a pin to an analog input, the corresponding TRIS bit must be set to Input mode in order to allow external control of the voltage on the pin.
REGISTER 11-6: WPua: WEAK PULL-UP PORTA REGISTER\(^{(1,2)}\)

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>WPUA5</td>
<td>WPUA4</td>
<td>WPUA3</td>
<td>WPUA2</td>
<td>WPUA1</td>
</tr>
</tbody>
</table>

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

bit 7-6: **Unimplemented:** Read as ‘0’
bit 5-0: **WPua<5:0>:** Weak Pull-up Register bits\(^{(3)}\)

- 1 = Pull-up enabled
- 0 = Pull-up disabled

**Note 1:**
- Global WPUE\(N\) bit of the OPTION_REG register must be cleared for individual pull-ups to be enabled.
- 2: The weak pull-up device is automatically disabled if the pin is configured as an output.
- 3: For the WPua3 bit, when MCLRE = 1, weak pull-up is internally enabled, but not reported here.

**TABLE 11-3: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSELA</td>
<td>—</td>
<td>—</td>
<td>ANSA5</td>
<td>ANSA4</td>
<td>—</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
<td>88</td>
</tr>
<tr>
<td>APFCON</td>
<td>—</td>
<td>—</td>
<td>SDSEL</td>
<td>SSEL</td>
<td>SDSEL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>85</td>
</tr>
<tr>
<td>LATA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>LATA5</td>
<td>LATA4</td>
<td>—</td>
<td>LATA2</td>
<td>LATA1</td>
</tr>
<tr>
<td>OPTION_REG</td>
<td>WPUE</td>
<td>—</td>
<td>INTE</td>
<td>TMRO</td>
<td>TMRO</td>
<td>PSA</td>
<td>—</td>
<td>—</td>
<td>132</td>
</tr>
<tr>
<td>PORTA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
</tr>
<tr>
<td>TRISA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>—</td>
<td>TRISA2</td>
<td>TRISA1</td>
</tr>
<tr>
<td>WPua</td>
<td>—</td>
<td>—</td>
<td>WPUA5</td>
<td>WPUA4</td>
<td>WPUA3</td>
<td>WPUA2</td>
<td>WPUA1</td>
<td>WPUA0</td>
<td>89</td>
</tr>
</tbody>
</table>

Legend: x = unknown, u = unchanged, — = unimplemented locations read as ‘0’. Shaded cells are not used by PORTA.

**Note 1:** Unimplemented, read as ‘1’.

**TABLE 11-4: SUMMARY OF CONFIGURATION WORD WITH PORTA**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bits</th>
<th>Bit -/7</th>
<th>Bit -/6</th>
<th>Bit 13/5</th>
<th>Bit 12/4</th>
<th>Bit 11/3</th>
<th>Bit 10/2</th>
<th>Bit 9/1</th>
<th>Bit 8/0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG1</td>
<td>13:8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BOREN&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>CP</td>
<td>MCLRE</td>
<td>PWRE</td>
<td>WDTE&lt;1:0&gt;</td>
<td>—</td>
<td>FOSC&lt;1:0&gt;</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

Legend: — = unimplemented location, read as ‘0’. Shaded cells are not used by PORTA.
12.0 INTERRUPT-ON-CHANGE

The PORTA pins can be configured to operate as Interrupt-On-Change (IOC) pins. An interrupt can be generated by detecting a signal that has either a rising edge or a falling edge. Any individual port pin, or combination of port pins, can be configured to generate an interrupt. The interrupt-on-change module has the following features:

- Interrupt-on-Change enable (Master Switch)
- Individual pin configuration
- Rising and falling edge detection
- Individual pin interrupt flags

Figure 12-1 is a block diagram of the IOC module.

12.1 Enabling the Module

To allow individual port pins to generate an interrupt, the IOCIE bit of the INTCON register must be set. If the IOCIE bit is disabled, the edge detection on the pin will still occur, but an interrupt will not be generated.

12.2 Individual Pin Configuration

For each port pin, a rising edge detector and a falling edge detector are present. To enable a pin to detect a rising edge, the associated bit of the IOCxP register is set. To enable a pin to detect a falling edge, the associated bit of the IOCxN register is set.

A pin can be configured to detect rising and falling edges simultaneously by setting both associated bits of the IOCxP and IOCxN registers, respectively.

12.3 Interrupt Flags

The IOCAFx bits located in the IOCAF register, respectively, are status flags that correspond to the interrupt-on-change pins of the associated port. If an expected edge is detected on an appropriately enabled pin, then the status flag for that pin will be set, and an interrupt will be generated if the IOCIE bit is set. The IOCIF bit of the INTCON register reflects the status of all IOCAFx bits.

12.4 Clearing Interrupt Flags

The individual status flags, (IOCAFx bits), can be cleared by resetting them to zero. If another edge is detected during this clearing operation, the associated status flag will be set at the end of the sequence, regardless of the value actually being written.

In order to ensure that no detected edge is lost while clearing flags, only AND operations masking out known changed bits should be performed. The following sequence is an example of what should be performed.

**EXAMPLE 12-1: CLEARING INTERRUPT FLAGS (PORTA EXAMPLE)**

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

12.5 Operation in Sleep

The interrupt-on-change interrupt sequence will wake the device from Sleep mode, if the IOCIE bit is set.

If an edge is detected while in Sleep mode, the IOCxF register will be updated prior to the first instruction executed out of Sleep.
FIGURE 12-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTA EXAMPLE)

Data Bus = 0 or 1
Write IOCAFx
From all other IOCAFx individual pin detectors

IOC interrupt to CPU core

From all other IOCAFx individual pin detectors
12.6 Register Definitions: Interrupt-on-Change Control

REGISTER 12-1: IOCAP: INTERRUPT-ON-CHANGE PORTA POSITIVE EDGE REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>IOCAP5</td>
<td>IOCAP4</td>
<td>IOCAP3</td>
<td>IOCAP2</td>
<td>IOCAP1</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>Unimplemented: Read as ‘0’</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 5-0</td>
<td>IOCAP&lt;5:0&gt;: Interrupt-on-Change PORTA Positive Edge Enable bits</td>
</tr>
<tr>
<td>1 = Interrupt-on-Change enabled on the pin for a positive going edge. IOCAFx bit and IOCF flag will be set upon detecting an edge.</td>
<td></td>
</tr>
<tr>
<td>0 = Interrupt-on-Change disabled for the associated pin.</td>
<td></td>
</tr>
</tbody>
</table>

REGISTER 12-2: IOCAN: INTERRUPT-ON-CHANGE PORTA NEGATIVE EDGE REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>IOCAN5</td>
<td>IOCAN4</td>
<td>IOCAN3</td>
<td>IOCAN2</td>
<td>IOCAN1</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>Unimplemented: Read as ‘0’</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 5-0</td>
<td>IOCAN&lt;5:0&gt;: Interrupt-on-Change PORTA Negative Edge Enable bits</td>
</tr>
<tr>
<td>1 = Interrupt-on-Change enabled on the pin for a negative going edge. IOCAFx bit and IOCF flag will be set upon detecting an edge.</td>
<td></td>
</tr>
<tr>
<td>0 = Interrupt-on-Change disabled for the associated pin.</td>
<td></td>
</tr>
</tbody>
</table>

REGISTER 12-3: IOCAF: INTERRUPT-ON-CHANGE PORTA FLAG REGISTER

<table>
<thead>
<tr>
<th>R/W/H-S-0/0</th>
<th>R/W/H-S-0/0</th>
<th>R/W/H-S-0/0</th>
<th>R/W/H-S-0/0</th>
<th>R/W/H-S-0/0</th>
<th>R/W/H-S-0/0</th>
<th>R/W/H-S-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>IOCAF5</td>
<td>IOCAF4</td>
<td>IOCAF3</td>
<td>IOCAF2</td>
<td>IOCAF1</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- HS = Bit is set in hardware

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>Unimplemented: Read as ‘0’</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 5-0</td>
<td>IOCAF&lt;5:0&gt;: Interrupt-on-Change PORTA Flag bits</td>
</tr>
<tr>
<td>1 = An enabled change was detected on the associated pin.</td>
<td></td>
</tr>
<tr>
<td>Set when IOCAPx = 1 and a rising edge was detected on RAx, or when IOCANx = 1 and a falling edge was detected on RAx.</td>
<td></td>
</tr>
<tr>
<td>0 = No change was detected, or the user cleared the detected change</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 12-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSELA</td>
<td>—</td>
<td>—</td>
<td>ANSA5</td>
<td>ANSA4</td>
<td>—</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
<td>88</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCF</td>
<td>57</td>
</tr>
<tr>
<td>IOCAF</td>
<td>—</td>
<td>—</td>
<td>IOCAF5</td>
<td>IOCAF4</td>
<td>IOCAF3</td>
<td>IOCAF2</td>
<td>IOCAF1</td>
<td>IOCAF0</td>
<td>92</td>
</tr>
<tr>
<td>IOCAN</td>
<td>—</td>
<td>—</td>
<td>IOCAN5</td>
<td>IOCAN4</td>
<td>IOCAN3</td>
<td>IOCAN2</td>
<td>IOCAN1</td>
<td>IOCAN0</td>
<td>92</td>
</tr>
<tr>
<td>IOCAP</td>
<td>—</td>
<td>—</td>
<td>IOCAP5</td>
<td>IOCAP4</td>
<td>IOCAP3</td>
<td>IOCAP2</td>
<td>IOCAP1</td>
<td>IOCAP0</td>
<td>92</td>
</tr>
<tr>
<td>TRISA</td>
<td>—</td>
<td>—</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>—(1)</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>87</td>
</tr>
</tbody>
</table>

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

13.0  FIXED VOLTAGE REFERENCE (FVR)

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

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

13.1  Independent Gain Amplifier

The output of the FVR supplied to the ADC is routed through a programmable gain amplifier. Each amplifier can be programmed for a gain of 1x or 2x, to produce the two possible voltage levels.

The ADFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module. Reference Section 16.0 “Hardware Capacitive Voltage Divider (CVD) Module” for additional information.

13.2  FVR Stabilization Period

When the Fixed Voltage Reference module is enabled, it requires time for the reference and amplifier circuits to stabilize. Once the circuits stabilize and are ready for use, the FVRRDY bit of the FVRCON register will be set. See Section 21.0 “Electrical Specifications” for the minimum delay requirement.

FIGURE 13-1:  VOLTAGE REFERENCE BLOCK DIAGRAM

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

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>Conditions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HFINTOSC</td>
<td>FOSC&lt;1:0&gt; = 00 and IRCF&lt;3:0&gt; = 000x</td>
<td>INTOSC is active and device is not in Sleep.</td>
</tr>
<tr>
<td>BOR</td>
<td>BOREN&lt;1:0&gt; = 11</td>
<td>BOR always enabled.</td>
</tr>
<tr>
<td></td>
<td>BOREN&lt;1:0&gt; = 10 and BORFS = 1</td>
<td>BOR disabled in Sleep mode, BOR Fast Start enabled.</td>
</tr>
<tr>
<td></td>
<td>BOREN&lt;1:0&gt; = 01 and BORFS = 1</td>
<td>BOR under software control, BOR Fast Start enabled.</td>
</tr>
</tbody>
</table>
13.3 Register Definitions: FVR Control

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

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FVREN</td>
<td>FVRRDY(1)</td>
<td>TSEN</td>
<td>TSRNG</td>
<td>—</td>
<td>—</td>
<td>ADFVR&lt;1:0&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7  
FVREN: Fixed Voltage Reference Enable bit
1 = Fixed Voltage Reference is enabled
0 = Fixed Voltage Reference is disabled

bit 6  
FVRRDY: Fixed Voltage Reference Ready Flag bit(1)
1 = Fixed Voltage Reference output is ready for use
0 = Fixed Voltage Reference output is not ready or not enabled

bit 5  
TSEN: Temperature Indicator Enable bit(3)
1 = Temperature Indicator is enabled
0 = Temperature Indicator is disabled

bit 4  
TSRNG: Temperature Indicator Range Selection bit(3)
1 = VOUT = VDD - 4VT (High Range)
0 = VOUT = VDD - 2VT (Low Range)

bit 3-2  
Unimplemented: Read as ‘0’

bit 1-0  
ADFVR<1:0>: ADC Fixed Voltage Reference Selection bit
11 = ADC Fixed Voltage Reference Peripheral output is off
10 = ADC Fixed Voltage Reference Peripheral output is 2x (2.048V)(2)
01 = ADC Fixed Voltage Reference Peripheral output is 1x (1.024V)
00 = ADC Fixed Voltage Reference Peripheral output is off

Note 1: FVRRDY is always ‘1’ for the PIC12LF1552 devices.
2: Fixed Voltage Reference output cannot exceed VDD.
3: See Section 14.0 “Temperature Indicator Module” for additional information.

TABLE 13-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE FIXED VOLTAGE REFERENCE

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FVRCON</td>
<td>FVREN</td>
<td>FVRRDY</td>
<td>TSEN</td>
<td>TSRNG</td>
<td>—</td>
<td>—</td>
<td>ADFVR&lt;1:0&gt;</td>
<td>95</td>
<td></td>
</tr>
</tbody>
</table>

Legend: Shaded cells are unused by the Fixed Voltage Reference module.
14.0 TEMPERATURE INDICATOR MODULE

This family of devices is equipped with a temperature circuit designed to measure the operating temperature of the silicon die. The circuit's range of operating temperature falls between -40°C and +85°C. The output is a voltage that is proportional to the device temperature. The output of the temperature indicator is internally connected to the device ADC.

The circuit may be used as a temperature threshold detector or a more accurate temperature indicator, depending on the level of calibration performed. A one-point calibration allows the circuit to indicate a temperature closely surrounding that point. A two-point calibration allows the circuit to sense the entire range of temperature more accurately. Reference Application Note AN1333, “Use and Calibration of the Internal Temperature Indicator” (DS01333) for more details regarding the calibration process.

14.1 Circuit Operation

Figure 14-1 shows a simplified block diagram of the temperature circuit. The proportional voltage output is achieved by measuring the forward voltage drop across multiple silicon junctions.

**EQUATION 14-1:** \( \text{Vout} \) RANGES

<table>
<thead>
<tr>
<th>Range</th>
<th>( \text{Vout} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>High Range</td>
<td>( \text{VDD} - 4VT )</td>
</tr>
<tr>
<td>Low Range</td>
<td>( \text{VDD} - 2VT )</td>
</tr>
</tbody>
</table>

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

The circuit is enabled by setting the TSEN bit of the FVRCON register. When disabled, the circuit draws no current.

The circuit operates in either high or low range. The high range, selected by setting the TSRNG bit of the FVRCON register, provides a wider output voltage. This provides more resolution over the temperature range, but may be less consistent from part to part. This range requires a higher bias voltage to operate and thus, a higher \( \text{VDD} \) is needed.

The low range is selected by clearing the TSRNG bit of the FVRCON register. The low range generates a lower voltage drop and thus, a lower bias voltage is needed to operate the circuit. The low range is provided for low voltage operation.

14.2 Minimum Operating VDD

When the temperature circuit is operated in low range, the device may be operated at any operating voltage that is within specifications.

When the temperature circuit is operated in high range, the device operating voltage, \( \text{VDD} \), must be high enough to ensure that the temperature circuit is correctly biased.

Table 14-1 shows the recommended minimum \( \text{VDD} \) vs. range setting.

**TABLE 14-1:** RECOMMENDED \( \text{VDD} \) VS. RANGE

<table>
<thead>
<tr>
<th>( \text{Min. VDD, TSRNG} = 1 )</th>
<th>( \text{Min. VDD, TSRNG} = 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.6V</td>
<td>1.8V</td>
</tr>
</tbody>
</table>

14.3 Temperature Output

The output of the circuit is measured using the internal Analog-to-Digital Converter. A channel is reserved for the temperature circuit output. Refer to Section 16.0 “Hardware Capacitive Voltage Divider (CVD) Module” for detailed information.

14.4 ADC Acquisition Time

To ensure accurate temperature measurements, the user must wait at least 200 \( \mu \text{s} \) after the ADC input multiplexer is connected to the temperature indicator output before the conversion is performed. In addition, the user must wait 200 \( \mu \text{s} \) between sequential conversions of the temperature indicator output.
### TABLE 14-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FVRCON</td>
<td>FVREN</td>
<td>FVRRDY</td>
<td>TSEN</td>
<td>TSRNG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>118</td>
</tr>
</tbody>
</table>

**Legend:** Shaded cells are unused by the temperature indicator module.
15.0 ANALOG-TO-DIGITAL CONVERTER (ADC) MODULE

The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which are multiplexed into a single sample and hold circuit. The output of the sample and hold is connected to the input of the converter. The converter generates a 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESH:ADRESL register pair). Figure 15-1 shows the block diagram of the ADC.

The ADC voltage reference is software selectable to be either internally generated or externally supplied. The ADC can generate an interrupt upon completion of a conversion. This interrupt can be used to wake-up the device from Sleep.

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

Note 1: When ADON = 0, all multiplexer inputs are disconnected.

2: See AADCON0 register (Register 16-1) for detailed analog channel selection per device.

3: ADRES0H and AADRES0H are the same register in two locations, Bank 1 and Bank 14. See Table 3-3.

4: ADRES0L and AADRES0L are the same register in two locations, Bank 1 and Bank 14. See Table 3-3.
15.1 ADC Configuration

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

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

15.1.1 PORT CONFIGURATION

The ADC can be used to convert both analog and digital signals. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits. Refer to Section 11.0 “I/O Ports” for more information.

15.1.2 CHANNEL SELECTION

There are up to eight channel selections available:

- AN<4:0> pins
- VREF+ (ADC positive reference)
- Temperature Indicator
- FVR (Fixed Voltage Reference) Output

Refer to Section 13.0 “Fixed Voltage Reference (FVR)” and Section 14.0 “Temperature Indicator Module” for more information on these channel selections.

The CHS bits of the ADCON0 register determine which channel is connected to the sample and hold circuit.

When changing channels, a delay is required before starting the next conversion. Refer to Section 16.1 “Hardware CVD Operation” for more information.

15.1.3 ADC VOLTAGE REFERENCE

The ADPREF bits of the ADCON1 register provides control of the positive voltage reference. The positive voltage reference can be:

- VREF+ pin
- VDD
- FVR (Fixed Voltage Reference)

See Section 13.0 “Fixed Voltage Reference (FVR)” for more details on the fixed voltage reference.

15.1.4 CONVERSION CLOCK

The source of the conversion clock is software selectable via the ADCS bits of the ADCON1 register. There are seven possible clock options:

- Fosc/2
- Fosc/4
- Fosc/8
- Fosc/16
- Fosc/32
- Fosc/64
- FRC (dedicated internal oscillator)

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

For correct conversion, the appropriate TAD specification must be met. Refer to the ADC conversion requirements in Section 21.0 “Electrical Specifications” for more information. Table 15-1 gives examples of appropriate ADC clock selections.

Note: Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

Note: Unless using the FRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.
### TABLE 15-1: ADC CLOCK PERIOD (TAD) VS. DEVICE OPERATING FREQUENCIES

<table>
<thead>
<tr>
<th>ADC Clock Source</th>
<th>ADcS&lt;2:0&gt;</th>
<th>20 MHz</th>
<th>16 MHz</th>
<th>8 MHz</th>
<th>4 MHz</th>
<th>1 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fosc/2</td>
<td>000</td>
<td>100 ns (2)</td>
<td>125 ns (2)</td>
<td>250 ns (2)</td>
<td>500 ns (2)</td>
<td>2.0 µs</td>
</tr>
<tr>
<td>Fosc/4</td>
<td>100</td>
<td>125 ns (2)</td>
<td>250 ns (2)</td>
<td>500 ns (2)</td>
<td>1.0 µs</td>
<td>4.0 µs</td>
</tr>
<tr>
<td>Fosc/8</td>
<td>001</td>
<td>250 ns (2)</td>
<td>500 ns (2)</td>
<td>1.0 µs</td>
<td>2.0 µs</td>
<td>8.0 µs (3)</td>
</tr>
<tr>
<td>Fosc/16</td>
<td>101</td>
<td>500 ns (2)</td>
<td>1.0 µs</td>
<td>2.0 µs</td>
<td>4.0 µs</td>
<td>16.0 µs (3)</td>
</tr>
<tr>
<td>Fosc/32</td>
<td>010</td>
<td>1.0 µs</td>
<td>2.0 µs</td>
<td>4.0 µs</td>
<td>8.0 µs (3)</td>
<td>32.0 µs (3)</td>
</tr>
<tr>
<td>Fosc/64</td>
<td>110</td>
<td>3.2 µs</td>
<td>4.0 µs</td>
<td>8.0 µs (3)</td>
<td>16.0 µs (3)</td>
<td>64.0 µs (3)</td>
</tr>
<tr>
<td>FRC</td>
<td>x11</td>
<td>1.0-6.0 µs (1,4)</td>
<td>1.0-6.0 µs (1,4)</td>
<td>1.0-6.0 µs (1,4)</td>
<td>1.0-6.0 µs (1,4)</td>
<td>1.0-6.0 µs (1,4)</td>
</tr>
</tbody>
</table>

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

**Note 1:** The FRC source has a typical TAD time of 1.6 µs for VDD.

**Note 2:** These values violate the minimum required TAD time.

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

**Note 4:** The ADC clock period (TAD) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock FOSC. However, the FRC clock source must be used when conversions are to be performed with the device in Sleep mode.

### FIGURE 15-2: ANALOG-TO-DIGITAL CONVERSION TAD CYCLES

- **TCY - TAD1**
- **TAD1**
- **TAD2**
- **TAD3**
- **TAD4**
- **TAD5**
- **TAD6**
- **TAD7**
- **TAD8**
- **TAD9**
- **TAD10**
- **TAD11**

- **Conversion starts**

- **Holding capacitor is disconnected from analog input (typically 100 ns)**

- **Set GO bit**

On the following cycle:
- ADRESH:ADRESL is loaded, GO bit is cleared.
- ADIF bit is set, holding capacitor is connected to analog input.
15.1.5 INTERRUPTS

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion. The ADC Interrupt Flag is the ADIF bit in the PIR1 register. The ADC Interrupt Enable is the ADIE bit in the PIE1 register. The ADIF bit must be cleared in software.

**Note 1:** The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled.

2: The ADC operates during Sleep only when the FRC oscillator is selected.

This interrupt can be generated while the device is operating or while in Sleep. If the device is in Sleep, the interrupt will wake-up the device. Upon waking from Sleep, the next instruction following the **SLEEP** instruction is always executed. If the user is attempting to wake-up from Sleep and resume in-line code execution, the GIE and PEIE bits of the INTCON register must be disabled. If the GIE and PEIE bits of the INTCON register are enabled, execution will switch to the Interrupt Service Routine.

15.1.6 RESULT FORMATTING

The 10-bit ADC conversion result can be supplied in two formats, left justified or right justified. The ADFM bit of the ADCON1 register controls the output format. Figure 15-3 shows the two output formats.

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

<table>
<thead>
<tr>
<th></th>
<th>ADRESH</th>
<th>ADRESL</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ADFM = 0)</td>
<td>10-bit ADC Result</td>
<td>Unimplemented: Read as ‘0’</td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td>bit 7</td>
</tr>
<tr>
<td>(ADFM = 1)</td>
<td>10-bit ADC Result</td>
<td>Unimplemented: Read as ‘0’</td>
</tr>
<tr>
<td>bit 7</td>
<td>MSB</td>
<td>bit 7</td>
</tr>
<tr>
<td>Unimplemented: Read as ‘0’</td>
<td>10-bit ADC Result</td>
<td></td>
</tr>
</tbody>
</table>
15.2 ADC Operation

15.2.1 STARTING A CONVERSION
To enable the ADC module, the ADON bit of the ADCON0 register must be set to a ‘1’. Setting the GO/DONE bit of the ADCON0 register to a ‘1’ will start the Analog-to-Digital conversion.

Note: The GO/DONE bit should not be set in the same instruction that turns on the ADC. Refer to Section 15.2.6 “ADC Conversion Procedure”.

15.2.2 COMPLETION OF A CONVERSION
When the conversion is complete, the ADC module will:
• Clear the GO/DONE bit
• Set the ADIF Interrupt Flag bit
• Update the ADRESH and ADRESL registers with new conversion result

15.2.3 TERMINATING A CONVERSION
If a conversion must be terminated before completion, the GO/DONE bit can be cleared in software. The ADRESH and ADRESL registers will be updated with the partially complete Analog-to-Digital conversion sample. Incomplete bits will match the last bit converted.

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

15.2.4 ADC OPERATION DURING SLEEP
The ADC module can operate during Sleep. This requires the ADC clock source to be set to the FRC option. When the FRC clock source is selected, the ADC waits one additional instruction before starting the conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled, the device will wake-up from Sleep when the conversion completes. If the ADC interrupt is disabled, the ADC module is turned off after the conversion completes, although the ADON bit remains set.

When the ADC clock source is something other than FRC, a SLEEP instruction causes the present conversion to be aborted and the ADC module is turned off, although the ADON bit remains set.

15.2.5 SPECIAL EVENT TRIGGER
The Special Event Trigger allows periodic ADC measurements without software intervention, using the TRIGSEL bits of the AADC02 register. When this trigger occurs, the GO/DONE bit is set by hardware from the Timer0 Overflow.

<table>
<thead>
<tr>
<th>Device</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC12LF1552</td>
<td>TMR0</td>
</tr>
</tbody>
</table>

Table 15-2: SPECIAL EVENT TRIGGER

Using the Special Event Trigger does not assure proper ADC timing. It is the user’s responsibility to ensure that the ADC timing requirements are met.

Refer to Section 17.0 “Timer0 Module” for more information.
15.2.6 ADC CONVERSION PROCEDURE

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

1. Configure Port:
   • Disable pin output driver (Refer to the TRIS register)
   • Configure pin as analog (Refer to the ANSEL register)
   • Disable weak pull-ups either globally (Refer to the OPTION_REG register) or individually (Refer to the appropriate WPUx register).

2. Configure the ADC module:
   • Select ADC conversion clock
   • Configure voltage reference
   • Select ADC input channel
   • Turn on ADC module

3. Configure ADC interrupt (optional):
   • Clear ADC interrupt flag
   • Enable ADC interrupt
   • Enable peripheral interrupt
   • Enable global interrupt(1)

4. Wait the required acquisition time(2).

5. Start conversion by setting the GO/DONE bit.

6. Wait for ADC conversion to complete by one of the following:
   • Polling the GO/DONE bit
   • Waiting for the ADC interrupt (interrupts enabled)

7. Read ADC Result in ADRES0H and ADRES0L.

8. Clear the ADC interrupt flag (required if interrupt is enabled).

### EXAMPLE 15-1: ADC CONVERSION

```asm
; This code block configures the ADC for polling, Vdd and Vss references, Frc clock and AN0 input.
; Conversion start & polling for completion are included.
; BANKSEL ADCON1
MOVLW B'11110000'
; Right justify, Frc clock
MOVWF ADCON1
BANKSEL TRISA
BSF TRISA,0 ; Set RA0 to input
BANKSEL ANSEL
BANKSEL WPUA
BCF WPUA,0 ; Disable RA0 weak pull-up
BANKSEL ADCON0
MOVLW B'00000001'
; Select channel AN0
MOVWF ADCON0
CALL SampleTime ; Acquisition delay
BSF ADCON0,ADGO ; Start conversion
BTFSC ADCON0,ADGO ; Is conversion done?
GOTO $-1 ; No, test again
BANKSEL ADRES0H
MOVF ADRES0H,W ; Read upper 2 bits
MOVF RESULTHI ; store in GPR space
```

**Note 1:** The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

**2:** Refer to Section 15.4 “ADC Acquisition Requirements”.
15.3 ADC Register Definitions

The following registers are used to control the operation of the ADC.

**REGISTER 15-1: ADCON0: ADC CONTROL REGISTER 0**

<table>
<thead>
<tr>
<th>bit 7</th>
<th>CHS&lt;4:0&gt;</th>
<th>bit 6-2</th>
<th>GO/DONE</th>
<th>ADON</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
<tr>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>Unimplemented: Read as ‘0’</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 6-2</td>
<td>CHS&lt;4:0&gt;: Analog Channel Select bits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11111 = FVR (Fixed Voltage Reference) Buffer 1 Output(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11110 = Reserved. No channel connected.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11101 = Temperature Indicator(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11100 = Reserved. No channel connected.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11011 = Reserved. No channel connected.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11010 = VREFH (ADC Positive Reference)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11001 = Reserved. No channel connected.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00101 = Reserved. No channel connected.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00100 = AN4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00011 = AN3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00010 = AN2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00001 = AN1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000 = AN0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 1</td>
<td>GO/DONE: ADC Conversion Status bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This bit is automatically cleared by hardware when the ADC conversion has completed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = ADC conversion completed/not in progress</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 0</td>
<td>ADON: ADC Enable bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = ADC is enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = ADC is disabled and consumes no operating current</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

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

**Note:**

1: See Section 13.0 “Fixed Voltage Reference (FVR)” for more information.
2: See Section 14.0 “Temperature Indicator Module” for more information.
REGISTER 15-2:  ADCON1: ADC CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6-4</th>
<th>bit 3-2</th>
<th>bit 1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADFM</td>
<td>ADCS&lt;2:0&gt;</td>
<td>ADPREF&lt;1:0&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Legend:

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

bit 7  
ADFM: ADC Result Format Select bit
1 = Right justified. Six Most Significant bits of ADRESH are set to ‘0’ when the conversion result is loaded.
0 = Left justified. Six Least Significant bits of ADRESL are set to ‘0’ when the conversion result is loaded.

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

bit 3-2  
Unimplemented: Read as ‘0’

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

Note 1:  When selecting the FVR or the VREF+ pin as the source of the positive reference, be aware that a minimum voltage specification exists. See Section 21.0 “Electrical Specifications” for details.
REGISTER 15-3: ADRES0H: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 0

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRES&lt;9:2&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’  
u = Bit is unchanged  
x = Bit is unknown  
‘1’ = Bit is set  
‘0’ = Bit is cleared

bit 7-0  ADRES<9:2>: ADC Result Register bits  
Upper eight bits of 10-bit conversion result

REGISTER 15-4: ADRES0L: ADC RESULT REGISTER LOW (ADRESL) ADFM = 0

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRES&lt;1:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’  
u = Bit is unchanged  
x = Bit is unknown  
‘1’ = Bit is set  
‘0’ = Bit is cleared

bit 7-6  ADRES<1:0>: ADC Result Register bits  
Lower two bits of 10-bit conversion result

bit 5-0  Reserved: Do not use.
### REGISTER 15-5: ADRES0H: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 1

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>ADRES&lt;9:8&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>bit 6</td>
<td>bit 5</td>
<td>bit 4</td>
<td>bit 3</td>
<td>bit 2</td>
<td>bit 1</td>
</tr>
</tbody>
</table>

**Legend:**

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

- **bit 7-2**: Reserved: Do not use.
- **bit 1-0**: **ADRES<9:8>**: ADC Result Register bits
  Upper two bits of 10-bit conversion result

### REGISTER 15-6: ADRES0L: ADC RESULT REGISTER LOW (ADRESL) ADFM = 1

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>ADRES&lt;7:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>bit 6</td>
<td>bit 5</td>
<td>bit 4</td>
<td>bit 3</td>
<td>bit 2</td>
<td>bit 1</td>
<td>bit 0</td>
</tr>
</tbody>
</table>

**Legend:**

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

- **bit 7-0**: **ADRES<7:0>**: ADC Result Register bits
  Lower eight bits of 10-bit conversion result
15.4 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (\( \text{CHOLD} \)) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 15-4. The source impedance (\( \text{RS} \)) and the internal sampling switch (\( \text{RSS} \)) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (\( \text{RSS} \)) impedance varies over the device voltage (\( \text{V}_{\text{DD}} \)), refer to Figure 15-4. The maximum recommended impedance for analog sources is 10 k\( \Omega \). As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an ADC acquisition must be done before the conversion can be started. To calculate the minimum acquisition time, Equation 15-1 may be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

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

Assumptions: Temperature = 50°C and external impedance of 10\( k\Omega \) 3.3V \( \text{V}_{\text{DD}} \)

\[
\text{TACQ} = \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} = \text{T}_{\text{AMP}} + \text{Tc} + \text{T}_{\text{COFF}}
\]

\[
= 2\mu s + \text{Tc} + [(\text{Temperature} - 25^\circ\text{C})(0.05\mu\text{s}/^\circ\text{C})]
\]

The value for \( \text{Tc} \) can be approximated with the following equations:

\[
\text{V}_{\text{APPLIED}}\left(1 - \frac{1}{\left(2^n + 1\right)}\right) = \text{V}_{\text{CHOLD}} 
\]

\( ;[1]\) \( \text{V}_{\text{CHOLD}} \) charged to within 1/2 LSB

\[
\text{V}_{\text{APPLIED}}\left(1 - e^{-\frac{-\text{Tc}}{\text{RC}}}\right) = \text{V}_{\text{CHOLD}} 
\]

\( ;[2]\) \( \text{V}_{\text{CHOLD}} \) charge response to \( \text{V}_{\text{APPLIED}} \)

\[
\text{V}_{\text{APPLIED}}\left(1 - e^{-\frac{-\text{Tc}}{\text{RC}}}\right) = \text{V}_{\text{APPLIED}}\left(1 - \frac{1}{\left(2^n + 1\right)}\right) 
\]

;combining \([1]\) and \([2]\)

Note: Where \( n = \text{number of bits of the ADC} \).

Solving for \( \text{Tc} \):

\[
\text{Tc} = -\text{CHOLD}(\text{R}_{\text{IC}} + \text{RSS} + \text{RS}) \ln(1/511)
\]

\[
= -10\mu\text{F}(1k\Omega + 7k\Omega + 10k\Omega) \ln(0.001957)
\]

\[
= 1.12\mu\text{s}
\]

Therefore:

\[
\text{TACQ} = 2\mu s + 1.12\mu s + [(50^\circ\text{C} - 25^\circ\text{C})(0.05\mu\text{s}/^\circ\text{C})]
\]

\[
= 4.42\mu\text{s}
\]

**Note 1:** The reference voltage (\( \text{V}_{\text{REF}} \)) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (\( \text{CHOLD} \)) is not discharged after each conversion.

3: The maximum recommended impedance for analog sources is 10 k\( \Omega \). This is required to meet the pin leakage specification.
FIGURE 15-4: ANALOG INPUT MODEL

Legend:  
- CHOLD = Sample/Hold Capacitance  
- CPIN = Input Capacitance  
- I LEAKAGE = Leakage current at the pin due to various junctions  
- RIC = Interconnect Resistance  
- RSS = Resistance of Sampling Switch  
- SS = Sampling Switch  
- VT = Threshold Voltage

Note 1: Refer to Section 21.0 “Electrical Specifications”.

FIGURE 15-5: ADC TRANSFER FUNCTION

Legend:  
- 0.5 LSB  
- 1.5 LSB  
- 0.5 LSB  
- Ref-  
- Ref+  
- ADC Output Code  
- Analog Input Voltage  
- Full-Scale Range  
- Zero-Scale Transition  
- Full-Scale Transition
### TABLE 15-3: SUMMARY OF REGISTERS ASSOCIATED WITH ADC

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCON0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CHS&lt;4:0&gt;</td>
<td>GO/DONE, ADON</td>
</tr>
<tr>
<td>ADCON1</td>
<td>ADFM</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADCS&lt;2:0&gt;</td>
<td>ADPREF&lt;1:0&gt;</td>
</tr>
<tr>
<td>ADRES0H</td>
<td>ADC Result Register High</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>104</td>
</tr>
<tr>
<td>ADRES0L</td>
<td>ADC Result Register Low</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>106, 107</td>
</tr>
<tr>
<td>ANSELA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ANSA5</td>
<td>ANSA4</td>
<td>—</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
</tr>
<tr>
<td>FVRCON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>88</td>
</tr>
<tr>
<td>INTCON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>95</td>
</tr>
<tr>
<td>PIE1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>57</td>
</tr>
<tr>
<td>PIR1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>TRISA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>—&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
<td>87</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented read as ‘0’. Shaded cells are not used for ADC module.

**Note 1:** Unimplemented, read as ‘1’.
16.0 HARDWARE CAPACITIVE VOLTAGE DIVIDER (CVD) MODULE

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

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

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

- Automated double sample conversions
- Two result registers
- Inversion of second sample
- 7-bit pre-charge timer
- 7-bit acquisition timer
- Two guard ring output drives
- Adjustable sample and hold capacitor array

Note: For more information on capacitive voltage divider sensing method refer to the Application Note AN1478, “mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider” (DS01478).
FIGURE 16-2: HARDWARE CAPACITIVE VOLTAGE DIVIDER BLOCK DIAGRAM

- Additional Sample and Hold Cap
  - ANx Pads
  - VDD_ADOEN
  - ADIPPOL = 1
  - ADIPPOL = 0
  - ADDCAP<2:0>
  - VGND
  - ADOUT Pad
  - ADOUT
  - ADOUT Pad
  - ANx
  - ANx Pads
  - VGND

Note: The diagram illustrates the connections and configurations for the hardware capacitive voltage divider block diagram in the PIC12LF1552.
16.1 Hardware CVD Operation

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

16.1.1 PRE-CHARGE TIMER

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

During the pre-charge time, \text{CHOLD} is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either VDD or VSS, depending on the value of the ADIPPOL bit of the AADCON3 register. At the same time, the port pin logic of the selected analog channel is overridden to turn off the digital high/low output drivers so that they do not affect the final result of charge averaging. Also, the selected ADC channel is connected to \text{CHOLD}. This allows charge averaging to proceed between the precharged channel and the \text{CHOLD} capacitor. It is noted that the port pin logic override that occurs during acquisition related to the selected sample channel does not occur on the ADOUT pin (see Section 16.1.9 “Analog Bus Visibility”) for more information.

16.1.2 ACQUISITION TIMER

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

At the start of the acquisition stage, the port pin logic of the selected analog channel is again overridden to turn off the digital high/low output drivers so that they do not affect the final result of charge averaging. Also, the selected ADC channel is connected to \text{CHOLD}. This allows charge averaging to proceed between the precharged channel and the \text{CHOLD} capacitor. It is noted that the port pin logic override that occurs during acquisition related to the selected sample channel does not occur on the ADOUT pin (see Section 16.1.9 “Analog Bus Visibility”) for more information.

16.1.3 STARTING A CONVERSION

To enable the ADC module, the ADON bit of the AADCON0 register must be set. Setting the GO/DONE bit of the AADCON0 register or by the Special Event Trigger inputs will start the Analog-to-Digital conversion.

Once a conversion begins, it proceeds until complete, while the ADON bit is set. If the ADON bit is cleared, the conversion is halted. The GO/DONE bit of the AADCON0 register indicates that a conversion is occurring, regardless of the starting trigger.

\textbf{Note:} The GO/DONE bit should not be set in the same instruction that turns on the ADC. Refer to Section 16.1.10 “Hardware CVD Double Conversion Procedure”.

16.1.4 COMPLETION OF A CONVERSION

When the conversion is complete, the ADC module will:

- Clear the GO/DONE bit of the AADCON0 register.
- Set the ADIF Interrupt Flag bit of the PIR1 register.
- Update the AADRESxH and AADRESxL registers with new conversion results.

16.1.5 TERMINATING A CONVERSION

If a conversion must be terminated before completion, clear the GO/DONE bit. The AADRESxH and AADRESxL registers will be updated with the partially complete Analog-to-Digital conversion sample. Incomplete bits will match the last bit converted.
The AADSTAT register can be used to track the status of the hardware CVD module during a conversion.

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

### 16.1.6 DOUBLE SAMPLE CONVERSION

Double sampling can be enabled by setting the ADDSEN bit of the AADCON3 register. When this bit is set, two conversions are completed each time the GO/DONE bit is set or a Special Event Trigger occurs. The GO/DONE bit remains set for the duration of both conversions and is used to signal the end of the conversion.

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

The result of the first conversion is written to the AADRES0H and AADRES0L registers. The second conversion starts two clock cycles after the first has completed, while the GO/DONE bit remains set. When the ADIPEN bit of AADCON3 is set, the value used by the ADC for the ADEPPOL, ADIPPOL, and GRDPOL bits are inverted. The value stored in those bit locations is unchanged. All other control signals remain unchanged from the first conversion. The result of the second conversion is stored in the AADRES1H and AADRES1L registers. See Figure 16-4 and Figure 16-5 for more information.

### 16.1.7 GUARD RING OUTPUTS

The guard ring outputs consist of a pair of digital outputs from the hardware CVD module. This function is enabled by the GRDAOE and GRDBOE bits of the AADGRD register. Polarity of the output is controlled by the GRDPOL bit.

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

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

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

**FIGURE 16-3: GUARD RING CIRCUIT**
16.1.8 ADDITIONAL SAMPLE AND HOLD CAPACITOR

Additional capacitance can be added in parallel with the sample and hold capacitor (CHOLD) by setting the ADDCAP<2:0> bits of the AADCAP register. This bit connects a digitally programmable capacitance to the ADC conversion bus, increasing the effective internal capacitance of the sample and hold capacitor in the ADC module. This is used to improve the match between internal and external capacitance for a better sensing performance. The additional capacitance does not affect analog performance of the ADC because it is not connected during conversion. See Figure 16-1.

16.1.9 ANALOG BUS VISIBILITY

The ADOEN of the AADCON3 register can be used to connect the ADC conversion bus to the ADOUT pin. This connection can be used to monitor the state and behavior of the internal analog bus and it also can be used to improve the match between internal and external capacitance by connecting a external capacitor to increase the effective internal capacitance. The ADOEN bit provides the connection via a standard channel pass gate.

The ADOUT pin function can be overridden during the pre-charge stage of conversion. This override function is controlled by ADOOEN. The polarity of the override is set by the ADIPPOL bit. It should be noted that, outside of the pre-charge phase, no ADOUT override is in effect. Therefore, the user must manage the state of the ADOUT pin via the relevant TRIS bit in order to avoid unintended affects on conversion results. If the user wishes to have the ADOUT path active during conversions, then the relevant TRIS bit should be set to ensure that the ADOUT pin logic is in the Input mode during the acquisition phase of conversions.
FIGURE 16-6: DOUBLE SAMPLE CONVERSION SEQUENCE (ADDSEN = 1 AND ADIPEN = 0)

Conversion Clock

Pre-charge
AADPRE<6:0>

Acquisition
AADACQ<6:0>

(1) 1-127 TINST
(1) 1-127 TINST
(1) 1-127 TINST
(1) 1-127 TINST

10'h000 10'h000 10'h000 10'h000

10th 9th 8th 7th 6th 5th 4th 3rd 2nd 1st

Pre-charge
AADPRE<6:0>

Acquisition
AADACQ<6:0>

TPRE TACQ TCONV First result written to AADRES0L/H

2INST 2INST 2INST 2INST

10th 9th 8th 7th 6th 5th 4th 3rd 2nd 1st

Second result written to AADRES1L/H

Pre-charge Acquisition
AADPRE<6:0> AADACQ<6:0>

Pre-charge Acquisition
AADPRE<6:0> AADACQ<6:0>

ADGRDA
(GRDPOL = 0)

ADGRDB

Internal CHOLD
Charging
(ADIPPOL = 1)

External Channel
Charging
(ADEPPOL = 0)

External Channel
Connected
To Internal CHOLD

GO/DONE

ADIF

ADSTAT<2:0>

3'b001 3'b010 3'b011 3'b101 3'b110 3'b111 3'b000

Note 1: When the conversion clock is ADCRC, the pre-charge and acquisition timers are clocked by ADCRC.
2: The AADRES0L/H registers are set to zero during this period.
3: The AADRES1L/H registers are set to zero during this period.
FIGURE 16-7: DOUBLE SAMPLE CONVERSION SEQUENCE (ADDSEN = 1 AND ADIPEN = 1)

Note 1: When the conversion clock is ADCRC, the pre-charge and acquisition timers are clocked by ADCRC.
2: The AADRES0L/H registers are set to zero during this period.
3: The AADRES1L/H registers are set to zero during this period.
16.1.10 HARDWARE CVD DOUBLE CONVERSION PROCEDURE

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

1. Configure Port:
   • Enable pin output driver (Refer to the TRIS register).
   • Configure pin output low (Refer to the LAT register).
   • Disable weak pull-up (Refer to the WPU register).

2. Configure the ADC module:
   • Select an appropriate ADC conversion clock for your oscillator frequency.
   • Configure voltage reference.
   • Select ADC input channel.
   • Turn on the ADC module.

3. Configure the hardware CVD module:
   • Configure charge polarity and double conversion.
   • Configure pre-charge and acquisition timer.
   • Configure guard ring (optional).
   • Select additional capacitance (optional).

4. Configure ADC interrupt (optional):
   • Clear ADC interrupt flag
   • Enable ADC interrupt
   • Enable peripheral interrupt
   • Enable global interrupt(1)

5. Start conversion by setting the GO/DONE bit or by enabling the Special Event Trigger in the ADDCON2 register.

6. Wait for the ADC conversion to complete by one of the following:
   • Polling the GO/DONE bit.
   • Waiting for the ADC interrupt (interrupts enabled).

7. Read ADC result:
   • Conversion 1 result in ADDRES0H and ADDRES0L
   • Conversion 2 result in ADDRES1H and ADDRES1L

8. Clear the ADC interrupt flag (required if interrupt is enabled).

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

EXAMPLE 16-1: HARDWARE CVD DOUBLE CONVERSION

```plaintext
;This code block configures the ADC
;for polling, Vdd and Vss references, Fosc/16
;and clock and AN0 input.

;The Hardware CVD will perform an inverted double conversion, Guard A and B drive are both enabled.
;Conversion start & polling for completion are included.

;BANKSEL TRISA
BCF TRISA,0 ;Set RA0 to output
BANKSEL LATA
BCF LATA,0 ;RA0 output low
BANKSEL ANSELA
BCF ANSELA,0 ;Set RA0 to digital
BANKSEL WPUA
BCF WPUA,0 ;Disable pull-up on RA0

;Initialize ADC and Hardware CVD

BANKSEL AADCON0
MOVLW B'00000001' ;Select channel AN0
MOVF AADCON0
BANKSEL AADCON1
MOVLW B'11010000' ;Vdd and Vss Vref
MOVF AADCON1
BANKSEL AADCON3
MOVLW B'01000011' ;Double and inverted
MOVF AADCON3 ;ADOUT disabled
BANKSEL AADPRE
MOVLW .10
MOVF AADPRE ;Pre-charge Timer
BANKSEL AADACQ
MOVLW .10
MOVF AADACQ ;Acquisition Timer
BANKSEL AADGRD
MOVLW B'11000000' ;Guard on A and B
MOVF AADGRD
BANKSEL AADCAP
MOVLW B'00000000' ;No additional Capacitor
MOVF AADCAP

;RESULTS OF CONVERSION 1.

GOTO $-1 ;No, test again

;RESULTS OF CONVERSION 2.

;END
```
16.1.11 HARDWARE CVD REGISTER MAPPING

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

<table>
<thead>
<tr>
<th>[Bank 14 Address]</th>
<th>[Bank 1 Address]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware CVD</td>
<td>ADC</td>
</tr>
<tr>
<td>[711h] AADCON0(1)</td>
<td>[09Dh] ADCON0(1)</td>
</tr>
<tr>
<td>[712h] AADCON1(1)</td>
<td>[09Eh] ADCON1(1)</td>
</tr>
<tr>
<td>[713h] AADCON2</td>
<td>[09Fh] ADCON2(1)</td>
</tr>
<tr>
<td>[714h] AADCON3</td>
<td></td>
</tr>
<tr>
<td>[715h] AADSTAT</td>
<td></td>
</tr>
<tr>
<td>[716h] AADPRE</td>
<td></td>
</tr>
<tr>
<td>[717h] AADACQ</td>
<td></td>
</tr>
<tr>
<td>[718h] AADGRD</td>
<td></td>
</tr>
<tr>
<td>[719h] AADCAP</td>
<td></td>
</tr>
<tr>
<td>[71Ah] AADRES0L(1)</td>
<td>[09Bh] ADRES0L(1)</td>
</tr>
<tr>
<td>[71Bh] AADRES0H(1)</td>
<td>[09Ch] ADRES0H(1)</td>
</tr>
<tr>
<td>[71Ch] AADRES1L</td>
<td></td>
</tr>
<tr>
<td>[71Dh] AADRES1H</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Register is mapped in Bank 1 and Bank 14, using different names in each bank.
### 16.2 Register Definitions: Hardware CVD Control

#### REGISTER 16-1: AADCON0: HARDWARE CVD CONTROL REGISTER 0\(^{(1)}\)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Unimplemented</td>
<td>Read as '0'</td>
<td></td>
</tr>
<tr>
<td>6-2</td>
<td>CHS&lt;4:0&gt;: Analog Channel Select bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>GO/DONE: ADC Conversion Status bit</td>
<td>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</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>ADON: ADC Enable bit</td>
<td>1 = ADC is enabled</td>
<td>0 = ADC is disabled and consumes no operating current</td>
</tr>
</tbody>
</table>

**Legend:**

- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **u** = Bit is unchanged
- **x** = Bit is unknown
- \(-n/n\) = Value at POR and BOR/Value at all other Resets

- ‘1’ = Bit is set
- ‘0’ = Bit is cleared

Note:

1: See Section 16.1.11 “Hardware CVD Register Mapping” for more information.
2: See Section 13.0 “Fixed Voltage Reference (FVR)” for more information.
3: See Section 14.0 “Temperature Indicator Module” for more information.
### AADCON1: HARDWARE CVD CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 7   | ADFM: ADC Result Format Select bit    |       | 1 = Right justified. Six Most Significant bits of AADRESxH are set to '0' when the conversion result is loaded.  
      |                                      |       | 0 = Left justified. Six Least Significant bits of AADRESxL are set to '0' when the conversion result is loaded. |
| 6-4 | ADCS<2:0>: ADC Conversion Clock Select bits |       | 111 = FRC (clock supplied from a dedicated RC oscillator)  
      |                                      |       | 110 = FOSC/64  
      |                                      |       | 101 = FOSC/16  
      |                                      |       | 100 = FOSC/4  
      |                                      |       | 011 = FRC (clock supplied from a dedicated RC oscillator)  
      |                                      |       | 010 = FOSC/32  
      |                                      |       | 001 = FOSC/8  
      |                                      |       | 000 = FOSC/2  |
| 3-2 | Unimplemented: Read as '0'            |       |                                                                      |
| 1-0 | ADPREF<1:0>: ADC Positive Voltage Reference Configuration bits |       | 11 = VREF is connected to internal Fixed Voltage Reference (FVR) module  
      |                                      |       | 10 = VREF is connected to external VREF+ pin  
      |                                      |       | 01 = Reserved  
      |                                      |       | 00 = VREF is connected to VDD |

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

**Note 1:** See Section 16.1.11 “Hardware CVD Register Mapping” for more information.

**Note 2:** When selecting the FVR or the VREF+ pin as the source of the positive reference, be aware that a minimum voltage specification exists. See Section 21.0 “Electrical Specifications” for details.

**Note 3:** The Hardware CVD module only supports the FRC conversion clock while in Sleep. When the internal oscillator is running, ADCS should be set to an Fosc option.
REGISTER 16-3: AADCON2: HARDWARE CVD CONTROL REGISTER 2\(^{(1)}\)

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>TRIGSEL&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

bit 7

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

bit 7 \textbf{Unimplemented}: Read as ‘0’

bit 6-4 TRIGSEL<2:0>: ADC Special Event Trigger Source Selection bits

- 111 = Reserved. Auto-conversion Trigger disabled.
- 110 = Reserved. Auto-conversion Trigger disabled.
- 101 = Reserved. Auto-conversion Trigger disabled.
- 100 = Reserved. Auto-conversion Trigger disabled.
- 011 = TMR0 Overflow
- 010 = Reserved. Auto-conversion Trigger disabled.
- 001 = Reserved. Auto-conversion Trigger disabled.
- 000 = No Auto Conversion Trigger Selection bits\(^{(2,3)}\)

bit 3-0 \textbf{Unimplemented}: Read as ‘0’

\textbf{Note 1}: See \textit{Section 16.1.11 “Hardware CVD Register Mapping”} for more information.

2: This is a rising edge sensitive input for all sources.

3: Signal used to set the corresponding interrupt flag.
REGISTER 16-4: AADCON3: HARDWARE CVD CONTROL REGISTER 3

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADEPPOL</td>
<td>ADIPPOL</td>
<td>—</td>
<td>ADOEN</td>
<td>—</td>
<td>ADOOEN</td>
<td>—</td>
<td>ADIPEN</td>
<td>ADDSEN</td>
</tr>
</tbody>
</table>

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

- **bit 7** ADEPPOL: External Pre-charge Polarity bit
  - 1 = Selected channel is shorted to VDDIO during pre-charge time
  - 0 = Selected channel is shorted to VSS during pre-charge time

- **bit 6** ADIPPOL: Internal Pre-charge Polarity bit
  - 1 = CHOLD is shorted to VREFH during pre-charge time
  - 0 = CHOLD is shorted to VREFL during pre-charge time

- **bit 5** Unimplemented: Read as ‘0’

- **bit 4** ADOEN: ADOUT Output Enable bit
  - 1 = ADOUT pin is connected to ADC bus (normal passgate)
  - 0 = No external connection to ADC bus

- **bit 3** ADOOEN: ADOUT Override Enable bit
  - 1 = ADOUT pin is overridden during pre-charge with internal polarity value
  - 0 = ADOUT pin is not overridden

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

- **bit 1** ADIPEN: ADC Invert Polarity Enable bit
  - If ADDSEN = 1:
    - 1 = The output value of the ADEPPOL, ADIPPOL, and GRDPOL bits used by the ADC are inverted for the second conversion
    - 0 = The second ADC conversion proceeds like the first
  - If ADDSEN = 0:
    - This bit has no effect.

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

**Note 1:** When the ADDSEN = 1 and ADIPEN = 1; the polarity of this output is inverted for the second conversion time. The stored bit value does not change.
REGISTER 16-5: AADSTAT: HARDWARE CVD STATUS REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADCONV</td>
<td>ADSTG&lt;1:0&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’  
u = Bit is unchanged  
x = Bit is unknown  
‘1’ = Bit is set  
‘0’ = Bit is cleared

bit 7-3 Unimplemented: Read as ‘0’
bit 2 ADCONV: ADC Conversion Status bit
   1 = Indicates ADC in Conversion Sequence for AADRES1H:AADRES1L
   0 = Indicates ADC in Conversion Sequence for AADRES0H:AADRES0L (Also reads ‘0’ when GO/DONE = 0)
bit 1-0 ADSTG<1:0>: ADC Stage Status bit
   11 = ADC module is in conversion stage
   10 = ADC module is in acquisition stage
   01 = ADC module is in pre-charge stage
   00 = ADC module is not converting (same as GO/DONE = 0)

REGISTER 16-6: AADPRE: HARDWARE CVD PRE-CHARGE CONTROL REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADPRE&lt;6:0&gt;</td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’  
u = Bit is unchanged  
x = Bit is unknown  
‘1’ = Bit is set  
‘0’ = Bit is cleared

bit 7 Unimplemented: Read as ‘0’
bit 6-0 ADPRE<6:0>: Pre-charge Time Select bits(1)
   111 1111 = Pre-charge for 127 instruction cycles
   111 1110 = Pre-charge for 126 instruction cycles
   ...
   000 0010 = Pre-charge for 1 instruction cycle (Fosc/4)
   000 0000 = ADC pre-charge time is disabled

Note 1: When the FRC clock is selected as the conversion clock source, it is also the clock used for the pre-charge and acquisition times.
REGISTER 16-7: AADACQ: HARDWARE CVD ACQUISITION TIME CONTROL REGISTER

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6-0</th>
<th>ADACQ&lt;6:0&gt;</th>
<th>Acquisition/Charge Share Time Select bits^{(1)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
</tr>
<tr>
<td>—</td>
<td>ADACQ&lt;6:0&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

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

**bit 7**

**Unimplemented:** Read as ‘0’

**bit 6-0**

**ADACQ<6:0>:** Acquisition/Charge Share Time Select bits^{(1)}

- 111 1111 = Acquisition/charge share for 127 instruction cycles
- 111 1110 = Acquisition/charge share for 126 instruction cycles
- ...
- 000 0001 = Acquisition/charge share for one instruction cycle (Fosc/4)
- 000 0000 = ADC Acquisition/charge share time is disabled

**Note 1:** When the FRC clock is selected as the conversion clock source, it is also the clock used for the pre-charge and acquisition times.

REGISTER 16-8: AADGRD: HARDWARE CVD GUARD RING CONTROL REGISTER

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6-0</th>
<th>GRDBOE^{(2)}</th>
<th>GRDAOE^{(2)}</th>
<th>GRDPOL^{(1,2)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>R/W-0/0</td>
<td>U-0</td>
<td>U-0</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Legend:

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

**bit 7**

**GRDBOE:** Guard Ring B Output Enable bit^{(2)}

- 1 = ADC guard ring output is enabled to ADGRDB pin. Its corresponding TRISx bit must be clear.
- 0 = No ADC guard ring function to this pin is enabled

**bit 6**

**GRDAOE:** Guard Ring A Output Enable bit^{(2)}

- 1 = ADC Guard Ring Output is enabled to ADGRDA pin. Its corresponding TRISx, x bit must be clear.
- 0 = No ADC Guard Ring function is enabled

**bit 5**

**GRDPOL:** Guard Ring Polarity selection bit^{(1,2)}

- 1 = ADC guard ring outputs start as digital high during pre-charge stage
- 0 = ADC guard ring outputs start as digital low during pre-charge stage

**bit 4-0**

**Unimplemented:** Read as ‘0’

**Note 1:**

- When the ADDSEN = 1 and ADIPEN = 1; the polarity of this output is inverted for the second conversion time. The stored bit value does not change.
- Guard Ring outputs are maintained while ADON = 1. The ADGRDA output switches polarity at the start of the acquisition time.
## REGISTER 16-9: AADCAP: HARDWARE CVD ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>ADDCAP&lt;2:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U-0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bit 7-3</th>
<th>Unimplemented</th>
<th>Read as ‘0’</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 2-0</td>
<td>ADDCAP: ADC Additional Sample Capacitor Selection bits</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>Nominal additional sample capacitor of 28 pF</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>Nominal additional sample capacitor of 24 pF</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>Nominal additional sample capacitor of 20 pF</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>Nominal additional sample capacitor of 16 pF</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>Nominal additional sample capacitor of 12 pF</td>
<td></td>
</tr>
<tr>
<td>010</td>
<td>Nominal additional sample capacitor of  8 pF</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>Nominal additional sample capacitor of  4 pF</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>Additional sample capacitor is disabled</td>
<td></td>
</tr>
</tbody>
</table>
### REGISTER 16-10: AADRESxH: HARDWARE CVD RESULT REGISTER MSB ADFM = 0\(^{(1)}\)

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>ADRESx&lt;9:2&gt;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Note 1: See Section 16.1.11 “Hardware CVD Register Mapping” for more information.

### REGISTER 16-11: AADRESxL: HARDWARE CVD RESULT REGISTER LSL ADFM = 0\(^{(1)}\)

<table>
<thead>
<tr>
<th>R/W-x/u</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>ADRESx&lt;1:0&gt;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-6  
**AD<1:0>:** ADC Result Register bits  
Lower two bits of 10-bit conversion result

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

Note 1: See Section 16.1.11 “Hardware CVD Register Mapping” for more information.
**REGISTER 16-12: AADRESxH: HARDWARE CVD RESULT REGISTER MSB ADFM = 1(1)**

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADRESx&lt;9:8&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-2  **Reserved:** Do not use.
bit 1-0  **AD<9:8>:** Most Significant ADC results

Note 1: See Section 16.1.11 “Hardware CVD Register Mapping” for more information.

**REGISTER 16-13: AADRESxL: HARDWARE CVD RESULT REGISTER LSB ADFM = 1(1)**

<table>
<thead>
<tr>
<th></th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7-0  **AD<7:0>:** ADC Result Register bits
Lower two bits of 10-bit conversion result

Note 1: See Section 16.1.11 “Hardware CVD Register Mapping” for more information.
### TABLE 16-2: SUMMARY OF REGISTERS ASSOCIATED WITH HARDWARE CVD

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AADCAP</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>126</td>
</tr>
<tr>
<td>AADCON0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>120</td>
</tr>
<tr>
<td>AADCON1</td>
<td>ADFM</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>121</td>
</tr>
<tr>
<td>AADCON2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>122</td>
</tr>
<tr>
<td>AADCON3</td>
<td>ADEPPOL</td>
<td>—</td>
<td>—</td>
<td>ADON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>123</td>
<td></td>
</tr>
<tr>
<td>AADGRD</td>
<td>GRDBOE</td>
<td>—</td>
<td>—</td>
<td>GRDAOE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>125</td>
<td></td>
</tr>
<tr>
<td>AADPRE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>124</td>
</tr>
<tr>
<td>AADRES0H</td>
<td>ADC Result 0 Register High</td>
<td>127</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AADRES0L</td>
<td>ADC Result 0 Register Low</td>
<td>127</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AADRES1H</td>
<td>ADC Result 1 Register High</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AADRES1L</td>
<td>ADC Result 1 Register Low</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AADSTAT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADCONV</td>
<td>124</td>
</tr>
<tr>
<td>AADACQ</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ADACQ&lt;6:0&gt;</td>
<td>125</td>
</tr>
<tr>
<td>ANSELA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ANSA5</td>
<td>88</td>
</tr>
<tr>
<td>FVRCON</td>
<td>FVREN</td>
<td>FVRRDY</td>
<td>TSEN</td>
<td>TSRNG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>95</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCF</td>
<td>57</td>
</tr>
<tr>
<td>PIE1</td>
<td>ADIE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SSPIE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>PIR1</td>
<td>ADIF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SSPIF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>60</td>
</tr>
<tr>
<td>TRISA</td>
<td>—</td>
<td>—</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>—(1)</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>87</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented read as '0'. Shaded cells are not used for hardware CVD module.

**Note 1:** Unimplemented, read as ‘1’.
17.0 TIMER0 MODULE

The Timer0 module is an 8-bit timer/counter with the following features:

- 8-bit timer/counter register (TMR0)
- 8-bit prescaler (independent of Watchdog Timer)
- Programmable internal or external clock source
- Programmable external clock edge selection
- Interrupt on overflow

Figure 17-1 is a block diagram of the Timer0 module.

17.1 Timer0 Operation

The Timer0 module can be used as either an 8-bit timer or an 8-bit counter.

17.1.1 8-BIT TIMER MODE

The Timer0 module will increment every instruction cycle, if used without a prescaler. 8-bit Timer mode is selected by clearing the TMR0CS bit of the OPTION_REG register.

When TMR0 is written, the increment is inhibited for two instruction cycles immediately following the write.

Note: The value written to the TMR0 register can be adjusted, in order to account for the two instruction cycle delay when TMR0 is written.

17.1.2 8-BIT COUNTER MODE

In 8-Bit Counter mode, the Timer0 module will increment on every rising or falling edge of the T0CKI pin.

8-Bit Counter mode using the T0CKI pin is selected by setting the TMR0CS bit in the OPTION_REG register to ‘1’.

The rising or falling transition of the incrementing edge for either input source is determined by the TMR0SE bit in the OPTION_REG register.
17.1.3 SOFTWARE PROGRAMMABLE PRESCALER

A software programmable prescaler is available for exclusive use with Timer0. The prescaler is enabled by clearing the PSA bit of the OPTION_REG register.

Note: The Watchdog Timer (WDT) uses its own independent prescaler.

There are eight prescaler options for the Timer0 module ranging from 1:2 to 1:256. The prescale values are selectable via the PS<2:0> bits of the OPTION_REG register. In order to have a 1:1 prescaler value for the Timer0 module, the prescaler must be disabled by setting the PSA bit of the OPTION_REG register.

The prescaler is not readable or writable. All instructions writing to the TMR0 register will clear the prescaler.

17.1.4 TIMER0 INTERRUPT

Timer0 will generate an interrupt when the TMR0 register overflows from FFh to 00h. The TMR0IF interrupt flag bit of the INTCON register is set every time the TMR0 register overflows, regardless of whether or not the Timer0 interrupt is enabled. The TMR0IF bit can only be cleared in software. The Timer0 interrupt enable is the TMR0IE bit of the INTCON register.

Note: The Timer0 interrupt cannot wake the processor from Sleep since the timer is frozen during Sleep.

17.1.5 8-BIT COUNTER MODE SYNCHRONIZATION

When in 8-Bit Counter mode, the incrementing edge on the T0CKI pin must be synchronized to the instruction clock. Synchronization can be accomplished by sampling the prescaler output on the Q2 and Q4 cycles of the instruction clock. The high and low periods of the external clocking source must meet the timing requirements as shown in Section 21.0 “Electrical Specifications”.

17.1.6 OPERATION DURING SLEEP

Timer0 cannot operate while the processor is in Sleep mode. The contents of the TMR0 register will remain unchanged while the processor is in Sleep mode.
17.2 Register Definitions: Option Register

REGISTER 17-1: OPTION_REG: OPTION REGISTER

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPUEN</td>
<td>INTEDG</td>
<td>TMR0CS</td>
<td>TMR0SE</td>
<td>PSA</td>
<td>PS&lt;2:0&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- u = Bit is unchanged
- x = Bit is unknown
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared

- **bit 7** \( \text{WPUEN} \): Weak Pull-Up Enable bit
  - 1 = All weak pull-ups are disabled (except MCLR, if it is enabled)
  - 0 = Weak pull-ups are enabled by individual WPUx latch values

- **bit 6** \( \text{INTEDG} \): Interrupt Edge Select bit
  - 1 = Interrupt on rising edge of INT pin
  - 0 = Interrupt on falling edge of INT pin

- **bit 5** \( \text{TMR0CS} \): Timer0 Clock Source Select bit
  - 1 = Transition on T0CKI pin
  - 0 = Internal instruction cycle clock (Fosc/4)

- **bit 4** \( \text{TMR0SE} \): Timer0 Source Edge Select bit
  - 1 = Increment on high-to-low transition on T0CKI pin
  - 0 = Increment on low-to-high transition on T0CKI pin

- **bit 3** \( \text{PSA} \): Prescaler Assignment bit
  - 1 = Prescaler is not assigned to the Timer0 module
  - 0 = Prescaler is assigned to the Timer0 module

- **bit 2-0** \( \text{PS<2:0>} \): Prescaler Rate Select bits

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Timer0 Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1 : 2</td>
</tr>
<tr>
<td>001</td>
<td>1 : 4</td>
</tr>
<tr>
<td>010</td>
<td>1 : 8</td>
</tr>
<tr>
<td>011</td>
<td>1 : 16</td>
</tr>
<tr>
<td>100</td>
<td>1 : 32</td>
</tr>
<tr>
<td>101</td>
<td>1 : 64</td>
</tr>
<tr>
<td>110</td>
<td>1 : 128</td>
</tr>
<tr>
<td>111</td>
<td>1 : 256</td>
</tr>
</tbody>
</table>

TABLE 17-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER0

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AADCON2</td>
<td>—</td>
<td>TRIGSEL&lt;2:0&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>122</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>57</td>
</tr>
<tr>
<td>OPTION_REG</td>
<td>WPUEN</td>
<td>INTEDG</td>
<td>TMR0CS</td>
<td>TMR0SE</td>
<td>PSA</td>
<td>PS&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>132</td>
</tr>
<tr>
<td>TMR0</td>
<td>Holding Register for the 8-bit Timer0 Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>130*</td>
</tr>
<tr>
<td>TRISA</td>
<td>—</td>
<td>—</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>—</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>87</td>
</tr>
</tbody>
</table>

Legend:
- = Unimplemented location, read as ‘0’.
- Shaded cells are not used by the Timer0 module.

Note 1: Unimplemented, read as ‘1’.
18.0 MASTER SYNCHRONOUS SERIAL PORT MODULE

18.1 Master SSP (MSSP1) Module

Overview

The Master Synchronous Serial Port (MSSP1) 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 MSSP1 module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I²C)

The SPI interface supports the following modes and features:

- Master mode
- Slave mode
- Clock Parity
- Slave Select Synchronization (Slave mode only)
- Daisy-chain connection of slave devices

Figure 18-1 is a block diagram of the SPI interface module.

Note: Register names, I/O pins, and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module, when required.
The I²C interface supports the following modes and features:

- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited Multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection
- General call address matching
- Address masking
- Address Hold and Data Hold modes
- Selectable SDA hold times

*Figure 18-2* is a block diagram of the I²C interface module in Master mode. *Figure 18-3* is a diagram of the I²C interface module in Slave mode.

The PIC12LF1552 has one MSSP module.
FIGURE 18-3: MSSP1 BLOCK DIAGRAM (I^2C SLAVE MODE)

Internal Data Bus

Read

Write

SSPSR Reg

Match Detect

SSPADD Reg

Start and Stop bit Detect

Set, Reset S, P bits (SSPSTAT Reg)

SCL

Shift Clock

SSPBUF Reg

Addr Match

SSPMSK Reg

Match Detect

SDA

MSb

LSb

Internal Data Bus

Shift Clock
18.2 SPI Mode Overview

The Serial Peripheral Interface (SPI) bus is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select.

The SPI bus specifies four signal connections:
- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Slave Select (SS)

Figure 18-1 shows the block diagram of the MSSP1 module when operating in SPI mode.

The SPI bus operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

Figure 18-4 shows a typical connection between a master device and multiple slave devices.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

Transmissions involve two shift registers, eight bits in size, one in the master and one in the slave. With either the master or the slave device, data is always shifted out one bit at a time, with the Most Significant bit (MSb) shifted out first. At the same time, a new Least Significant bit (LSb) is shifted into the same register.

Figure 18-5 shows a typical connection between two processors configured as master and slave devices.

Data is shifted out of both shift registers on the programmed clock edge and latched on the opposite edge of the clock.

The master device transmits information out on its SDO output pin which is connected to, and received by, the slave’s SDI input pin. The slave device transmits information out on its SDO output pin, which is connected to, and received by, the master’s SDI input pin.

To begin communication, the master device first sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

The master device starts a transmission by sending out the MSb from its shift register. The slave device reads this bit from that same line and saves it into the LSb position of its shift register.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the master device is sending out the MSb from its shift register (on its SDO pin) and the slave device is reading this bit and saving it as the LSb of its shift register, that the slave device is also sending out the MSb from its shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its shift register.

After eight bits have been shifted out, the master and slave have exchanged register values.

If there is more data to exchange, the shift registers are loaded with new data and the process repeats itself.

Whether the data is meaningful or not (dummy data), depends on the application software. This leads to three scenarios for data transmission:
- Master sends useful data and slave sends dummy data.
- Master sends useful data and slave sends useful data.
- Master sends dummy data and slave sends useful data.

Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops sending the clock signal and it deselects the slave.

Every slave device connected to the bus that has not been selected through its slave select line must disregard the clock and transmission signals and must not transmit out any data of its own.
18.2.1 SPI MODE REGISTERS

The MSSP1 module has five registers for SPI mode operation. These are:

- MSSP1 STATUS register (SSPSTAT)
- MSSP1 Control Register 1 (SSPCON1)
- MSSP1 Control Register 3 (SSPCON3)
- MSSP1 Data Buffer register (SSPBUF)
- MSSP1 Address register (SSPADD)
- MSSP1 Shift register (SSPSR)
  (Not directly accessible)

SSPCON1 and SSPSTAT are the control and STATUS registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower six bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

In SPI master mode, SSPADD can be loaded with a value used in the Baud Rate Generator. More information on the Baud Rate Generator is available in Section 18.7 “Baud Rate Generator”

SSPSR is the shift register used for shifting data in and out. SSPBUF provides indirect access to the SSPSR register. SSPBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPSR and SSPBUF together create a buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

18.2.2 SPI MODE OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

To enable the serial port, SSP Enable bit, SSPEN of the SSPCON1 register, must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPCONx registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and SS pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI must have corresponding TRIS bit set
- SDO must have corresponding TRIS bit cleared
- SCK (Master mode) must have corresponding TRIS bit cleared
- SCK (Slave mode) must have corresponding TRIS bit set
- SS must have corresponding TRIS bit set

Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.
The MSSP1 consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready. Once the eight bits of data have been received, that byte is moved to the SSPBUF register. Then, the Buffer Full Detect bit, BF of the SSPSTAT register, and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the write collision detect bit WCOL of the SSPCON1 register, will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPBUF register to complete successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. The Buffer Full bit, BF of the SSPSTAT register, indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP1 interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur.

The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPBUF register. Additionally, the SSPSTAT register indicates the various Status conditions.
18.2.3 SPI MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK line. The master determines when the slave (Processor 2, Figure 18-5) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and Status bits appropriately set).

The clock polarity is selected by appropriately programming the CKP bit of the SSPCON1 register and the CKE bit of the SSPSTAT register. This then, would give waveforms for SPI communication as shown in Figure 18-6, Figure 18-8, Figure 18-9 and Figure 18-10, where the MSb is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or Tcy)
- FOSC/16 (or 4 * Tcy)
- FOSC/64 (or 16 * Tcy)
- Fosc/(4 * (SSPAD + 1))

Figure 18-6 shows the waveforms for Master mode.

When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

FIGURE 18-6: SPI MODE WAVEFORM (MASTER MODE)
18.2.4 SPI SLAVE MODE
In Slave mode, the data is transmitted and received as external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag bit is set.
Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit of the SSPCON1 register.
While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.
While in Sleep mode, the slave can transmit/receive data. The shift register is clocked from the SCK pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

18.2.4.1 Daisy-Chain Configuration
The SPI bus can sometimes be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole chain acts as one large communication shift register. The daisy-chain feature only requires a single Slave Select line from the master device.
Figure 18-7 shows the block diagram of a typical daisy-chain connection when operating in SPI mode.
In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the BOEN bit of the SSPCON3 register will enable writes to the SSPBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

18.2.5 SLAVE SELECT SYNCHRONIZATION
The Slave Select can also be used to synchronize communication. The Slave Select line is held high until the master device is ready to communicate. When the Slave Select line is pulled low, the slave knows that a new transmission is starting.
If the slave fails to receive the communication properly, it will be reset at the end of the transmission, when the Slave Select line returns to a high state. The slave is then ready to receive a new transmission when the Slave Select line is pulled low again. If the Slave Select line is not used, there is a risk that the slave will eventually become out of sync with the master. If the slave misses a bit, it will always be one bit off in future transmissions. Use of the Slave Select line allows the slave and master to align themselves at the beginning of each transmission.
The S S pin allows a Synchronous Slave mode. The SPI must be in Slave mode with S S pin control enabled (SSPCON1<3:0> = 0100).
When the S S pin is low, transmission and reception are enabled and the SDO pin is driven.
When the S S pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

Note 1: When the SPI is in Slave mode with S S pin control enabled (SSPCON1<3:0> = 0100), the SPI module will reset if the S S pin is set to VDD.
2: When the SPI is used in Slave mode with CKE set; the user must enable S S pin control.
3: While operated in SPI Slave mode the SMP bit of the SSPSTAT register must remain clear.

When the SPI module resets, the bit counter is forced to ‘0’. This can be done by either forcing the S S pin to a high level or clearing the SSPEN bit.
FIGURE 18-7: SPI DAISY-CHAIN CONNECTION

FIGURE 18-8: SLAVE SELECT SYNCHRONOUS WAVEFORM
FIGURE 18-9: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

FIGURE 18-10: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)
18.2.6 SPI OPERATION IN SLEEP MODE

In SPI Master mode, module clocks may be operating at a different speed than when in Full-Power mode; in the case of the Sleep mode, all clocks are halted.

Special care must be taken by the user when the MSSP1 clock is much faster than the system clock.

In Slave mode, when MSSP1 interrupts are enabled, after the master completes sending data, an MSSP1 interrupt will wake the controller from Sleep.

If an exit from Sleep mode is not desired, MSSP1 interrupts should be disabled.

In SPI Master mode, when the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all eight bits have been received, the MSSP1 interrupt flag bit will be set and if enabled, will wake the device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSELA</td>
<td>—</td>
<td>—</td>
<td>ANSA5</td>
<td>ANSA4</td>
<td>—</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
<td>85</td>
</tr>
<tr>
<td>APFCON</td>
<td>—</td>
<td>SDOSEL</td>
<td>SSSEL</td>
<td>SDSEL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>85</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>57</td>
</tr>
<tr>
<td>PIE1</td>
<td>—</td>
<td>ADIE</td>
<td>—</td>
<td>—</td>
<td>SSPIE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>PIR1</td>
<td>—</td>
<td>ADIF</td>
<td>—</td>
<td>—</td>
<td>SSPIF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>60</td>
</tr>
<tr>
<td>SSPBUF</td>
<td>Synchronous Serial Port Receive Buffer/Transmit Register</td>
<td>137*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPCON1</td>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPEN</td>
<td>CKP</td>
<td>SSPM&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPCON3</td>
<td>ACKTIM</td>
<td>PCIE</td>
<td>SCIE</td>
<td>BOEN</td>
<td>SDAHT</td>
<td>SBCDE</td>
<td>AHEN</td>
<td>DHEN</td>
<td>184</td>
</tr>
<tr>
<td>SSPSTAT</td>
<td>SMP</td>
<td>CKE</td>
<td>D/A</td>
<td>P</td>
<td>S</td>
<td>R/W</td>
<td>UA</td>
<td>BF</td>
<td>181</td>
</tr>
<tr>
<td>TRISA</td>
<td>—</td>
<td>—</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>—(1)</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>87</td>
</tr>
</tbody>
</table>

Legend: — = Unimplemented location, read as ‘0’. Shaded cells are not used by the MSSP1 in SPI mode.

* Page provides register information.

Note 1: Unimplemented, read as ‘1’.
18.3 \( \text{i}^2\text{C} \) MODE OVERVIEW

The Inter-Integrated Circuit Bus (\( \text{i}^2\text{C} \)) is a multi-master serial data communication bus. Devices communicate in a master/slave environment where the master devices initiate the communication. A Slave device is controlled through addressing.

The \( \text{i}^2\text{C} \) bus specifies two signal connections:
- Serial Clock (SCL)
- Serial Data (SDA)

Figure 18-2 and Figure 18-3 show the block diagrams of the MSSP1 module when operating in \( \text{i}^2\text{C} \) mode.

Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors for the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one.

Figure 18-11 shows a typical connection between two processors configured as master and slave devices.

The \( \text{i}^2\text{C} \) bus can operate with one or more master devices and one or more slave devices.

There are four potential modes of operation for a given device:
- Master Transmit mode (master is transmitting data to a slave)
- Master Receive mode (master is receiving data from a slave)
- Slave Transmit mode (slave is transmitting data to a master)
- Slave Receive mode (slave is receiving data from the master)

To begin communication, a master device starts out in Master Transmit mode. The master device sends out a Start bit followed by the address byte of the slave it intends to communicate with. This is followed by a single Read/Write bit, which determines whether the master intends to transmit to or receive data from the slave device.

If the requested slave exists on the bus, it will respond with an Acknowledge bit, otherwise known as an ACK. The master then continues in either Transmit mode or Receive mode and the slave continues in the complement, either in Receive mode or Transmit mode, respectively.

A Start bit is indicated by a high-to-low transition of the SDA line while the SCL line is held high. Address and data bytes are sent out, Most Significant bit (MSb) first. The Read/Write bit is sent out as a logical one when the master intends to read data from the slave, and is sent out as a logical zero when it intends to write data to the slave.

If the master intends to write to the slave, then it repeatedly sends out a byte of data, with the slave responding after each byte with an ACK bit. In this example, the master device is in Master Transmit mode and the slave is in Slave Receive mode.

If the master intends to read from the slave, then it repeatedly receives a byte of data from the slave, and responds after each byte with an ACK bit. In this example, the master device is in Master Receive mode and the slave is Slave Transmit mode.

On the last byte of data communicated, the master device may end the transmission by sending a Stop bit. If the master device is in Receive mode, it first sends a not ACK bit in place of an ACK and then terminates the transfer with a Stop bit.

In some cases, the master may want to maintain control of the bus and re-initiate another transmission. If so, the master device may send another Start bit in place of the Stop bit or last ACK bit when it is in receive mode.

The \( \text{i}^2\text{C} \) bus specifies three message protocols:
- Single message where a master writes data to a slave.
- Single message where a master reads data from a slave.
- Combined message where a master initiates a minimum of two writes, or two reads, or a combination of writes and reads, to one or more slaves.

When one device is transmitting a logical one, or letting the line float, and a second device is transmitting a logical zero, or holding the line low, the first device can detect that the line is not a logical one. This detection, when used on the SCL line, is called clock stretching. Clock stretching gives slave devices a mechanism to control the flow of data. When this detection is used on
the SDA line, it is called arbitration. Arbitration ensures that there is only one master device communicating at any single time.

18.3.1 CLOCK STRETCHING
When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCL clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master that is communicating with the slave will attempt to raise the SCL line in order to transfer the next bit, but will detect that the clock line has not yet been released. Because the SCL connection is open-drain, the slave has the ability to hold that line low until it is ready to continue communicating.

Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

18.3.2 ARBITRATION
Each master device must monitor the bus for Start and Stop bits. If the device detects that the bus is busy, it cannot begin a new message until the bus returns to an Idle state.

However, there are three conditions under which a collision can occur:

a) Two master devices may try to initiate a transmission on or about the same time
b) A device acting as multiple devices on the bus (one of which is acting as a master) may collide with another master which is trying to access a slave address on the first device
c) Two slaves may respond to a general call read at the same time

18.3.2.1 Multi-Master Collision
In the first condition each master transmitter checks the level of the SDA data line and compares it to the level that it expects to find. The first transmitter to observe that the two levels do not match loses arbitration and must stop transmitting on the SDA line. For example, if one transmitter holds the SDA line to a logical one (lets it float) and a second transmitter holds it to a logical zero (pulls it low), the result is that the SDA line will be low.

The first transmitter then observes that the level of the line is different than expected and concludes that another transmitter is communicating. The first transmitter to notice this difference is the one that loses arbitration and must stop driving the SDA line. If this transmitter is also a master device, it must also stop driving the SCL line. Then it 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.

18.3.2.2 Multi-Master with Slave Recovery
In the second condition there are essentially three entities on the bus: a master and a slave that reside within the same device, and a second master attempting to access the slave. If the master coupled to the slave loses the arbitration, then the slave must be able to recover and correctly respond to the second master. To accomplish this, the master which shares a device with the slave must use the I2C Firmware Controller Master mode of operation. This mode utilizes software to perform the master function, while the slave monitors the bus as a separate entity allowing it to respond to the second master.

To detect a collision, each device transmitting on the bus must check the level of the SDA data line and compare it to the level that it expects to find. The first transmitter to observe that the two levels do not match loses arbitration, and it must drop off the bus as well as stop transmitting on the SDA line if it is acting as a master. For example, if one transmitter holds the SDA line to a logical one (lets it float) and a second transmitter holds it to a logical zero (pulls it low), the result is that the SDA line will be low.

If two master devices send a message to two different slave devices at the address stage, the master sending the lower slave address always wins arbitration. When two master devices send messages to the same slave address, and addresses can sometimes refer to multiple slaves, the arbitration process must continue into the data stage.

18.3.2.3 Multi-Slave Collision
When a system attempts a slave read from the General Call address, there is the possibility that more than one slave devices may attempt to return data to the master. When this happens, a read collision occurs and the slaves which fail the arbitration must fall off the bus and allow the winning device to continue its data transmission. Once the winning device has completed its data transfer, the master can then initiate an additional read from the General Call address to retrieve the second slave’s data. In large systems this may require multiple reads by the master. Several protocols use this feature and the MSSP hardware incorporates a detection mechanism in the slave read to detect the condition and respond appropriately.

In most designs, arbitration usually occurs very rarely, but it is a necessary process for proper multi-master support or even multi-slave systems that use General Call address reads.
18.4 **I²C MODE OPERATION**

All MSSP1 I²C communication is byte oriented and shifted out MSb first. Six SFR registers and two interrupt flags interface the module with the PIC® microcontroller and user software. Two pins, SDA and SCL, are exercised by the module to communicate with other external I²C devices.

18.4.1 **BYTE FORMAT**

All communication in I²C is done in 9-bit segments. A byte is sent from a master to a slave or vice-versa, followed by an Acknowledge bit sent back. After the 8th falling edge of the SCL line, the device outputting data on the SDA changes that pin to an input and reads in an acknowledge value on the next clock pulse.

The clock signal, SCL, is provided by the master. Data is valid to change while the SCL signal is low, and sampled on the rising edge of the clock. Changes on the SDA line while the SCL line is high define special conditions on the bus, explained below.

18.4.2 **DEFINITION OF I²C TERMINOLOGY**

There is language and terminology in the description of I²C communication that have definitions specific to I²C. That word usage is defined below and may be used in the rest of this document without explanation. This table was adapted from the Philips I²C specification.

18.4.3 **SDA AND SCL PINS**

Selection of any I²C mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain. These pins should be set by the user to inputs by setting the appropriate TRIS bits.

| Note: | Data is tied to output zero when an I²C mode is enabled. |

18.4.4 **SDA HOLD TIME**

The hold time of the SDA pin is selected by the SDAHT bit of the SSPCON3 register. Hold time is the time SDA is held valid after the falling edge of SCL. Setting the SDAHT bit selects a longer 300 ns minimum hold time and may help on buses with large capacitance.

<table>
<thead>
<tr>
<th>TABLE 18-2: I²C BUS TERMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TERM</td>
</tr>
<tr>
<td>Transmitter</td>
</tr>
<tr>
<td>Receiver</td>
</tr>
<tr>
<td>Master</td>
</tr>
<tr>
<td>Slave</td>
</tr>
<tr>
<td>Multi-master</td>
</tr>
<tr>
<td>Arbitration</td>
</tr>
<tr>
<td>Synchronization</td>
</tr>
<tr>
<td>Idle</td>
</tr>
<tr>
<td>Active</td>
</tr>
<tr>
<td>Addressed Slave</td>
</tr>
<tr>
<td>Write Request</td>
</tr>
<tr>
<td>Read Request</td>
</tr>
<tr>
<td>Clock Stretching</td>
</tr>
<tr>
<td>Bus Collision</td>
</tr>
</tbody>
</table>
18.4.5 START CONDITION

The \textsuperscript{I\textsubscript{2}C} 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 18-12 shows waveforms for Start and Stop conditions.

A bus collision can occur on a Start condition if the module samples the SDA line low before asserting it low. This does not conform to the \textsuperscript{I\textsubscript{2}C} Specification that states no bus collision can occur on a Start.

18.4.6 STOP CONDITION

A Stop condition is a transition of the SDA line from low-to-high state while the SCL line is high.

**Note:** At least one SCL low time must appear before a Stop is valid, therefore, if the SDA line goes low then high again while the SCL line stays high, only the Start condition is detected.

18.4.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave. Figure 18-13 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 \textit{R/W} bit set. The slave logic will then hold the clock and prepare to clock out data.

After a full match with \textit{R/W} clear in 10-bit mode, a prior match flag is set and maintained. Until a Stop condition, a high address with \textit{R/W} clear, or high address match fails.

18.4.8 START/STOP CONDITION INTERRUPT MASKING

The SCIE and PCIE bits of the SSPCON3 register can enable the generation of an interrupt in Slave modes that do not typically support this function. Slave modes where interrupt on Start and Stop detect are already enabled, these bits will have no effect.

**FIGURE 18-12:** \textsuperscript{I\textsubscript{2}C} START AND STOP CONDITIONS

**FIGURE 18-13:** \textsuperscript{I\textsubscript{2}C} RESTART CONDITION
18.4.9 ACKNOWLEDGE SEQUENCE

The 9th SCL pulse for any transferred byte in \( \text{I}^2\text{C} \) is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDA line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge (\( \text{ACK} \)) is an active-low signal, pulling the SDA line low indicated to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an \( \text{ACK} \) is placed in the ACKSTAT bit of the SSPCON2 register.

Slave software, when the AHEN and DHEN bits are set, allow the user to set the \( \text{ACK} \) value sent back to the transmitter. The ACKDT bit of the SSPCON2 register is set/cleared to determine the response.

Slave hardware will generate an \( \text{ACK} \) response if the AHEN and DHEN bits of the SSPCON3 register are clear.

There are certain conditions where an \( \text{ACK} \) will not be sent by the slave. If the BF bit of the SSPSTAT register or the SSPOV bit of the SSPCON1 register are set when a byte is received.

When the module is addressed, after the 8th falling edge of SCL on the bus, the ACKTIM bit of the SSPCON3 register is set. The ACKTIM bit indicates the acknowledge time of the active bus. The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is enabled.

18.5 \( \text{I}^2\text{C} \) SLAVE MODE OPERATION

The MSSP1 Slave mode operates in one of four modes selected in the SSFM bits of SSPCON1 register. The modes can be divided into 7-bit and 10-bit Addressing mode. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

Modes with Start and Stop bit interrupts operate the same as the other modes with SSPIF additionally getting set upon detection of a Start, Restart, or Stop condition.

18.5.1 SLAVE MODE ADDRESSES

The SSPADD register (Register 18-6) contains the Slave mode address. The first byte received after a Start or Restart condition is compared against the value stored in this register. If the byte matches, the value is loaded into the SSPBUF register and an interrupt is generated. If the value does not match, the module goes idle and no indication is given to the software that anything happened.

The SSP Mask register (Register 18-5) affects the address matching process. See Section 18.5.9 “SSP Mask Register” for more information.

18.5.1.1 \( \text{I}^2\text{C} \) Slave 7-Bit Addressing Mode

In 7-bit Addressing mode, the LSb of the received data byte is ignored when determining if there is an address match.

18.5.1.2 \( \text{I}^2\text{C} \) Slave 10-Bit Addressing Mode

In 10-bit Addressing mode, the first received byte is compared to the binary value of \( '11110\ A9\ A8\ 0' \). A9 and A8 are the two MSb of the 10-bit address and stored in bits 2 and 1 of the SSPADD register.

After the acknowledge of the high byte the UA bit is set and SCL is held low until the user updates SSPADD with the low address. The low address byte is clocked in and all eight bits are compared to the low address value in SSPADD. Even if there is not an address match; SSPIF and UA are set, and SCL is held low until SSPADD is updated to receive a high byte again. When SSPADD is updated the UA bit is cleared. This ensures the module is ready to receive the high address byte on the next communication.

A high and low address match as a write request is required at the start of all 10-bit addressing communication. A transmission can be initiated by issuing a Restart once the slave is addressed, and clocking in the high address with the R/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.
18.5.2 SLAVE RECEPTION

When the R/W bit of a matching received address byte is clear, the R/W bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and acknowledged.

When the overflow condition exists for a received address, then not Acknowledge is given. An overflow condition is defined as either bit BF of the SSPSTAT register is set, or bit SSPOV of the SSPCON1 register is set. The BOEN bit of the SSPCON3 register modifies this operation. For more information see Register 18-4.

An MSSP1 interrupt is generated for each transferred data byte. Flag bit, SSPIF, must be cleared by software.

When the SEN bit of the SSPCON2 register is set, SCL will be held low (clock stretch) following each received byte. The clock must be released by setting the CKP bit of the SSPCON1 register, except sometimes in 10-bit mode. See Section 18.2.3 “SPI Master Mode” for more detail.

18.5.2.1 7-Bit Addressing Reception

This section describes a standard sequence of events for the MSSP1 module configured as an I2C Slave in 7-bit Addressing mode. Figure 18-14 and Figure 18-15 are used as visual references for this description.

This is a step by step process of what typically must be done to accomplish I2C communication.

1. Start bit detected.
2. S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Matching address with R/W bit clear is received.
4. The slave pulls SDA low sending an ACK to the master, and sets SSPIF bit.
5. Software clears the SSPIF bit.
6. Software reads received address from SSPBUF clearing the BF flag.
7. If SEN = 1; Slave software sets CKP bit to release the SCL line.
8. The master clocks out a data byte.
9. Slave drives SDA low sending an ACK to the master, and sets SSPIF bit.
10. Software clears SSPIF.
11. Software reads the received byte from SSPBUF clearing BF.
12. Steps 8-12 are repeated for all received bytes from the Master.

18.5.2.2 7-Bit Reception with AHEN and DHEN

Slave device reception with AHEN and DHEN set operate the same as without these options with extra interrupts and clock stretching added after the 8th falling edge of SCL. These additional interrupts allow the slave software to decide whether it wants to ACK the receive address or data byte, rather than the hardware. This functionality adds support for PMBus™ that was not present on previous versions of this module.

This list describes the steps that need to be taken by slave software to use these options for I2C communication. Figure 18-16 displays a module using both address and data holding. Figure 18-17 includes the operation with the SEN bit of the SSPCON2 register set.

1. S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
2. Matching address with R/W bit clear is clocked in. SSPIF is set and CKP cleared after the 8th falling edge of SCL.
3. Slave clears the SSPIF.
4. Slave can look at the ACKTIM bit of the SSPCON3 register to determine if the SSPIF was after or before the ACK.
5. Slave reads the address value from SSPBUF, clearing the BF flag.
6. Slave sets ACK value clocked out to the master by setting ACKDT.
7. Slave releases the clock by setting CKP.
8. SSPIF is set after an ACK, not after a NACK.
9. If SEN = 1 the slave hardware will stretch the clock after the ACK.
10. Slave clears SSPIF.

Note: SSPIF is still set after the 9th falling edge of SCL even if there is no clock stretching and BF has been cleared. Only if NACK is sent to Master is SSPIF not set

11. SSPIF set and CKP cleared after 8th falling edge of SCL for a received data byte.
12. Slave looks at ACKTIM bit of SSPCON3 to determine the source of the interrupt.
13. Slave reads the received data from SSPBUF clearing BF.
14. Steps 7-14 are the same for each received data byte.
15. Communication is ended by either the slave sending an ACK = 1, or the master sending a Stop condition. If a Stop is sent and Interrupt-on-Stop Detect is disabled, the slave will only know by polling the P bit of the SSPSTAT register.
FIGURE 18-14: \( i^2C \) SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 0, DHEN = 0)

Receiving Address

<table>
<thead>
<tr>
<th>Receiver Data</th>
<th>ACK</th>
<th>Receiving Data</th>
<th>ACK</th>
</tr>
</thead>
<tbody>
<tr>
<td>A7 A6 A5 A4 A3 A2 A1</td>
<td>A0</td>
<td>D7 D6 D5 D4 D3 D2 D1 D0</td>
<td>A0</td>
</tr>
</tbody>
</table>

ACK = 1

From Slave to Master

Bus Master sends Stop condition

First byte of data is available in SSPBUF

SSPOV set because SSPBUF is still full, ACK is not sent.

SSPOV set because SSPBUF is still full, ACK is not sent.

Cleared by software

Clears by software

SSPF set on 9th falling edge of SCL

SSPF set on 9th falling edge of SCL

First byte of data is available in SSPBUF

Cleared by software

SSPF set because SSPBUF is still full, ACK is not sent.
FIGURE 18-15: I2C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)

- **Receive Address**
  - SDA: A7 A6 A5 A4 A3 A2 A1 R/W=1
  - SCL: 1 2 3 4 5 6 7 8 9
  - SEN

- **Receive Data**
  - D7 D6 D5 D4 D3 D2 D1 D0 ACK
  - D7 D6 D5 D4 D3 D2 D1 D0

- **SSPIF**
  - SSPIF set on 9th
  - ACK
  - SSPIF set on 9th

- **BF**
  - BF

- **SSPOV**
  - SSPOV

- **CKP**
  - CKP
  - CKP is written to '1' in software, releasing SCL
  - CKP is written to 1 in software, releasing SCL

- **Bus Master sends**
  - Stop condition

- **Notes**
  - Clock is held low until CKP is set to '1'
  - SCL is not held low because ACK=1
  - ACK is not sent
  - SSPBUF is still full.
FIGURE 18-16: \( \text{I}^2\text{C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)} \)

- Master Releases SDA to slave for ACK sequence.
- Receiving Address:
  - SDA: A7 A6 A5 A4 A3 A2 A1
  - SCL: S S S S S S S
- Receiving Data:
  - ACK: 1
  - D7 D6 D5 D4 D3 D2 D1 D0
- ACK:
  - ACK: 1
  - D7 D6 D5 D4 D3 D2 D1 D0

- BF:
  - If AHEN = 1, SSPIF is set.
  - SSPIF is set on 9th falling edge of SCL, after ACK.
  - Cleared by software.
  - No interrupt after not ACK from Slave.

- ACKDT:
  - Address is read from SSBUF.
  - Slave software clears ACKDT to ACK the received byte.

- CKP:
  - When AHEN = 1, CKP is cleared by hardware and SCL is stretched.
  - When DHEN = 1, CKP is cleared by hardware on 8th falling edge of SCL.
  - CKP set by software, SCL is released.

- ACKTIM:
  - ACKTIM set by hardware on 8th falling edge of SCL.
  - ACKTIM cleared by hardware in 9th rising edge of SCL.
  - ACKTIM set by hardware on 8th falling edge of SCL.

- Master Releases SDA to slave for ACK sequence.
- Data is read from SSPBUF.
- Slave software clears ACKDT to ACK the received byte.
- ACKTIM set by hardware on 8th falling edge of SCL.
- ACKTIM cleared by hardware in 9th rising edge of SCL.
- ACKTIM set by hardware on 8th falling edge of SCL.
FIGURE 18-17: \(i^2C\) SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 1, DHEN = 1)

- **SDA**: Receiving Address
- **SCL**: Receive Data
- **SSPIF**: Received address is loaded into SSPBUF
- **BF**: Slave software clears ACKDT to ACK the received byte
- **ACKDT**: When AHEN = 1; on the 8th falling edge of SCL of an address byte, CKP is cleared
- **CKP**: When DHEN = 1; on the 8th falling edge of SCL of a received data byte, CKP is cleared
- **ACKTIM**: ACKTIM is set by hardware on 8th falling edge of SCL
- **ACKTIM**: ACKTIM is cleared by hardware on 9th rising edge of SCL
- **Master releases SDA to slave for ACK sequence**
- **S**: Master sends Stop condition
- **ACK**: No interrupt after if not ACK from Slave
- **BF**: Cleared by software
- **SSPBUF**: SSPBUF can be read any time before next byte is loaded
18.5.3 SLAVE TRANSMISSION

When the R/W bit of the incoming address byte is set and an address match occurs, the R/W bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register, and an ACK pulse is sent by the slave on the ninth bit.

Following the ACK, slave hardware clears the CKP bit and the SCL pin is held low (see Section 18.5.6 “Clock Stretching” for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data.

The transmit data must be loaded into the SSPBUF register which also loads the SSPSR register. Then the SCL pin should be released by setting the CKP bit of the SSPCON1 register. The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time.

The ACK pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. This ACK value is copied to the ACKSTAT bit of the SSPCON2 register. If ACKSTAT is set (not ACK), then the data transfer is complete. In this case, when the not ACK is latched by the slave, the slave goes idle and waits for another occurrence of the Start bit. If the SDA line was low (ACK), the next transmit data must be loaded into the SSPBUF register. Again, the SCL pin must be released by setting bit CKP.

An MSSP1 interrupt is generated for each data transfer byte. The SSPIF bit must be cleared by software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

18.5.3.1 Slave Mode Bus Collision

A slave receives a Read request and begins shifting data out on the SDA line. If a bus collision is detected and the SBCDE bit of the SSPCON3 register is set, the BCLIF bit of the PIRx register is set. Once a bus collision is detected, the slave goes Idle and waits to be addressed again. User software can use the BCLIF bit to handle a slave bus collision.

18.5.3.2 7-Bit Transmission

A master device can transmit a read request to a slave, and then clock data out of the slave. The list below outlines what software for a slave will need to do to accomplish a standard transmission. Figure 18-18 can be used as a reference to this list.

1. Master sends a Start condition on SDA and SCL.
2. S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Matching address with R/W bit set is received by the Slave setting SSPIF bit.
4. Slave hardware generates an ACK and sets SSPIF.
5. SSPIF bit is cleared by user.
6. Software reads the received address from SSPBUF, clearing BF.
7. R/W is set so CKP was automatically cleared after the ACK.
8. The slave software loads the transmit data into SSPBUF.
9. CKP bit is set releasing SCL, allowing the master to clock the data out of the slave.
10. SSPIF is set after the ACK response from the master is loaded into the ACKSTAT register.
11. SSPIF bit is cleared.
12. The slave software checks the ACKSTAT bit to see if the master wants to clock out more data.

Note 1: If the master ACKs the clock will be stretched.

Note 2: ACKSTAT is the only bit updated on the rising edge of SCL (9th) rather than the falling.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not ACK; the clock is not held, but SSPIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.
Figure 18-18: \( \text{I}^2 \text{C} \) Slave, 7-Bit Address, Transmission (AHEN = 0)

- **SDA**: Received address
- **SCL**: When R/W is set, SCL is always held low after 9th SCL falling edge
- **SSPIF**: ACK
- **BF**: Automatically cleared after 8th falling edge of SCL
- **CKP**: Set by software
- **ACKSTAT**: Masters not ACK is copied to ACKSTAT
- **R/W**: \( \text{R/W} \)
- **D/A**: Indicates an address has been received
- **S**: Master sends Stop condition

Key:
- 1: Received address
- 2: Data to transmit is loaded into SSPBUF
- 3: BF is automatically cleared after 8th falling edge of SCL
- 4: SCL is always held low after 9th SCL falling edge
- 5: ACK
- 6: CKP is not held for not ACK
18.5.3.3  7-Bit Transmission with Address Hold Enabled

Setting the AHEN bit of the SSPCON3 register enables additional clock stretching and interrupt generation after the 8th falling edge of a received matching address. Once a matching address has been clocked in, CKP is cleared and the SSPIF interrupt is set.

Figure 18-19 displays a standard waveform of a 7-bit Address Slave Transmission with AHEN enabled.

1. Bus starts Idle.
2. Master sends Start condition; the S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Master sends matching address with R/W bit set. After the 8th falling edge of the SCL line the CKP bit is cleared and SSPIF interrupt is generated.
4. Slave software clears SSPIF.
5. Slave software reads ACKTIM bit of SSPCON3 register, and R/W and D/A of the SSPSTAT register to determine the source of the interrupt.
6. Slave reads the address value from the SSPBUF register clearing the BF bit.
7. Slave software decides from this information if it wishes to ACK or not ACK and sets the ACKDT bit of the SSPCON2 register accordingly.
8. Slave sets the CKP bit releasing SCL.
9. Master clocks in the ACK value from the slave.
10. Slave hardware automatically clears the CKP bit and sets SSPIF after the ACK if the R/W bit is set.
11. Slave software clears SSPIF.
12. Slave loads value to transmit to the master into SSPBUF setting the BF bit.

**Note:** SSPBUF cannot be loaded until after the ACK.

13. Slave sets CKP bit releasing the clock.
14. Master clocks out the data from the slave and sends an ACK value on the 9th SCL pulse.
15. Slave hardware copies the ACK value into the ACKSTAT bit of the SSPCON2 register.
16. Steps 10-15 are repeated for each byte transmitted to the master from the slave.
17. If the master sends a not ACK the slave releases the bus allowing the master to send a Stop and end the communication.

**Note:** Master must send a not ACK on the last byte to ensure that the slave releases the SCL line to receive a Stop.
FIGURE 18-19: \( ^{2}C \) SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)

- **SDA**: Receiving Address
- **SCL**: Automatic
- **R/W**: Transmitting Data
- **BF**: Received address is read from SSPBUF.
- **ACKDT**: Data to transmit is loaded into SSPBUF.
- **ACKSTAT**: BF is automatically cleared after 8th falling edge of SCL.
- **SSPIF**: Master releases SDA to slave for ACK sequence.
- **ACKTIM**: Master's ACK response is copied to SSPSTAT.
- **CKP**: Set by software, releases SCL.
- **ACKTIM**:master sends Stop condition.
18.5.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSP1 module configured as an I²C Slave in 10-bit Addressing mode.

Figure 18-20 is used as a visual reference for this description.

This is a step by step process of what must be done by slave software to accomplish I²C communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit of SSPSTAT is set; SSPIF is set if interrupt on Start detect is enabled.
3. Master sends matching high address with R/W bit clear; UA bit of the SSPSTAT register is set.
4. Slave sends ACK and SSPIF is set.
5. Software clears the SSPIF bit.
6. Software reads received address from SSPBUF clearing the BF flag.
7. Slave loads low address into SSPADD, releasing SCL.
8. Master sends matching low address byte to the Slave; UA bit is set.

Note: Updates to the SSPADD register are not allowed until after the ACK sequence.

9. Slave sends ACK and SSPIF is set.

Note: If the low address does not match, SSPIF and UA are still set so that the slave software can set SSPADD back to the high address. BF is not set because there is no match. CKP is unaffected.

10. Slave clears SSPIF.
11. Slave reads the received matching address from SSPBUF clearing BF.
12. Slave loads high address into SSPADD.
13. Master clocks a data byte to the slave and clocks out the slaves ACK on the 9th SCL pulse; SSPIF is set.
14. If SEN bit of SSPCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPIF.
16. Slave reads the received byte from SSPBUF clearing BF.
17. If SEN is set the slave sets CKP to release the SCL.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

18.5.5 10-BIT ADDRESSING WITH ADDRESS OR DATA HOLD

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCL line is held low are the same. Figure 18-21 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 18-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.
FIGURE 18-20: I²C SLAVE, 10-BIT ADDRESS, RECESSION (SEN = 1, AHEN = 0, DHEN = 0)

- **Receive First Address Byte**
  - SDA: 11110 A9 A8
  - SCL: 12345678 91
  - SSPIF: Set by hardware on 9th falling edge
  - BF: If address matches SSPADD it is loaded into SSPBUF
  - UA: When UA = 1; SCL is held low
  - CKP: When SEN = 1; CKP is cleared after 9th falling edge of received byte

- **Receive Second Address Byte**
  - SDA: ACK A7 A6 A5 A4 A3 A2 A1 A0
  - SCL: 12345678 91
  - SSPIF: Cleared by software
  - BF: Receive address is read from SSPBUF
  - UA: Software updates SSPADD

- **Receive Data**
  - SDA: D7 D6 D5 D4 D3 D2 D1 D0
  - SCL: 12345678 91
  - SSPIF: Cleared by software
  - BF: Data is read from SSPBUF

- **Receive Data**
  - SDA: ACK D7 D6 D5 D4 D3 D2 D1 D0
  - SCL: 12345678 91
  - SSPIF: Cleared by software
  - BF: SCL is held low while CKP = 0

- **Master sends Stop condition**
FIGURE 18-21: \(\text{I}^2\text{C}\) SLAVE, 10-BIT ADDRESS, RECEPTION (\(\text{SEN} = 0\), \(\text{AHEN} = 1\), \(\text{DHEN} = 0\))

Receive First Address Byte

SDA: \[1\ 1\ 1\ 1\ 0\ \text{A9}\ \text{A8}\]

Receive Second Address Byte

SDA: \[1\ 1\ 1\ 1\ 0\ \text{A9}\ \text{A8}\]

Receive Data

SDA: \[1\ 1\ 1\ 1\ 0\ \text{A9}\ \text{A8}\]

SSPBUF is read from the received byte

SSPBUF can be read anytime before the next received byte

ACKDT is set by hardware on 9th falling edge

ACKDT is cleared by software

UA

Update to SSPADD is not allowed until 9th falling edge of SCL

Update of SSPADD, clears UA and releases SCL

CKP

If when \(\text{AHEN} = 1\), on the 8th falling edge of SCL of an address byte, CKP is cleared

ACKTIM

ACKTIM is set by hardware on 8th falling edge of SCL

ACKDT to ACK the received byte

ACK

ACK
FIGURE 18-22:  \(^2\text{C SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)}\)

- **SDA**: Receiving Address, ACK
- **SCL**: Receiving Second Address Byte
- **SSPIF**: Receive First Address Byte
- **BF**: Transmitting Data Byte
- **UA**: Master sends Restart event
- **ACKSTAT**: Master sends Stop condition
- **CKP**: Master sends not ACK
- **R/W**: Set by hardware
- **D/A**: Cleared by software
- **SSPBUF**: SSPADD loaded with received address
- **UA**: After SSPADD is updated, UA is cleared and SCL is released
- **SSP0ADD**: High address is loaded into SSPADD
- **ACK**: Data to transmit is loaded into SSPBUF
- **ACKSTAT**: When R/W = 1, CKP is cleared on 9th falling edge of SCL
- **R/W**: Set by software releases SCL
- **D/A**: Masters not ACK is copied
- **D/A**: Indicates an address has been received
18.5.6 CLOCK STRETCHING

Clock stretching occurs when a device on the bus holds the SCL line low, effectively pausing communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. A master device is not concerned with stretching as anytime it is active on the bus and not transferring data, it is stretching. Any stretching done by a slave is invisible to the master software and handled by the hardware that generates SCL.

The CKP bit of the SSPCON1 register is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. Setting CKP will release SCL and allow more communication.

18.5.6.1 Normal Clock Stretching

Following an ACK if the R/W bit of SSPSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPBUF with data to transfer to the master. If the SEN bit of SSPCON2 is set, the slave hardware will always stretch the clock after the ACK sequence. Once the slave is ready; CKP is set by software and communication resumes.

18.5.6.2 10-Bit Addressing Mode

In 10-bit Addressing mode, when the UA bit is set, the clock is always stretched. This is the only time the SCL is stretched without CKP being cleared. SCL is released immediately after a write to SSPADD.

**Note:** Previous versions of the module did not stretch the clock if the second address byte did not match.

18.5.6.3 Byte NACKing

When AHEN bit of SSPCON3 is set; CKP is cleared by hardware after the 8th falling edge of SCL for a received matching address byte. When DHEN bit of SSPCON3 is set; CKP is cleared after the 8th falling edge of SCL for received data.

Stretching after the 8th falling edge of SCL allows the slave to look at the received address or data and decide if it wants to ACK the received data.

18.5.7 CLOCK SYNCHRONIZATION AND THE CKP BIT

Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. However, clearing the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I^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 18-23).

**FIGURE 18-23:** CLOCK SYNCHRONIZATION TIMING

Table: SDA, SCL, CKP, WR, SSPCON1 timeline with Q1, Q2, Q3, Q4 transitions, Master device asserts and releases clock, DX, DX – 1.
18.5.8 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I2C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master device. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge.

The general call address is a reserved address in the I2C protocol, defined as address 0x00. When the GCEN bit of the SSPCON2 register is set, the slave module will automatically ACK the reception of this address regardless of the value stored in SSPADD. After the slave clocks in an address of all zeros with the R/W bit clear, an interrupt is generated and slave software can read SSPBUF and respond. Figure 18-24 shows a general call reception sequence.

In 10-bit Address mode, the UA bit will not be set on the reception of the general call address. The slave will prepare to receive the second byte as data, just as it would in 7-bit mode.

If the AHEN bit of the SSPCON3 register is set, just as with any other address reception, the slave hardware will stretch the clock after the 8th falling edge of SCL. The slave must then set its ACKDT value and release the clock with communication progressing as it would normally.

18.5.9 SSP MASK REGISTER

An SSP Mask (SSPMSK) register (Register 18-5) is available in I2C Slave mode as a mask for the value held in the SSPSR register during an address comparison operation. A zero (‘0’) bit in the SSPMSK register has the effect of making the corresponding bit of the received address a “don’t care”.

This register is reset to all ‘1’s upon any Reset condition and, therefore, has no effect on standard SSP operation until written with a mask value.

The SSP Mask register is active during:
• 7-bit Address mode: address compare of A<7:1>.
• 10-bit Address mode: address compare of A<7:0> only. The SSP mask has no effect during the reception of the first (high) byte of the address.
18.6 I²C MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in the SSPCON1 register and by setting the SSPEN bit. In Master mode, the SDA and SCK pins must be configured as inputs. The MSSP peripheral hardware will override the output driver TRIS controls when necessary to drive the pins low.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP1 module is disabled. Control of the I²C bus may be taken when the P bit is set, or the bus is Idle.

In Firmware Controlled Master mode, user code conducts all I²C bus operations based on Start and Stop bit condition detection. Start and Stop condition detection is the only active circuitry in this mode. All other communication is done by the user software directly manipulating the SDA and SCL lines.

The following events will cause the SSP Interrupt Flag bit, SSPIF, to be set (SSP interrupt, if enabled):
- Start condition detected
- Stop condition detected
- Data transfer byte transmitted/received
- Acknowledge transmitted/received
- Repeated Start generated

Note 1: The MSSP1 module, when configured in I²C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur

2: When in Master mode, Start/Stop detection is masked and an interrupt is generated when the SEN/PEN bit is cleared and the generation is complete.

18.6.1 I²C MASTER MODE OPERATION

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

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/W) bit. In this case, the R/W bit will be logic ‘0’. Serial data is transmitted eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/W bit. In this case, the R/W bit will be logic ‘1’. Thus, the first byte transmitted is a 7-bit slave address followed by a ‘1’ to indicate the receive bit.

Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received eight bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

A Baud Rate Generator is used to set the clock frequency output on SCL. See Section 18.7 “Baud Rate Generator” for more detail.

18.6.2 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, releases the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 18-25).
18.6.3 WCOL STATUS FLAG
If the user writes the SSPBUF when a Start, Restart, Stop, Receive or Transmit sequence is in progress, the WCOL 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 SSPBUF was attempted while the module was not Idle.

**Note:** Because queuing of events is not allowed, writing to the lower five bits of SSPCON2 is disabled until the Start condition is complete.

18.6.4 I²C MASTER MODE START CONDITION TIMING
To initiate a Start condition (Figure 18-26), the user sets the Start Enable bit, SEN bit of the SSPCON2 register. If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the S bit of the SSPSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit of the SSPCON2 register will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

**Note 1:** If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state.

2: The Philips I²C Specification states that a bus collision cannot occur on a Start.

**FIGURE 18-25: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**

**FIGURE 18-26: FIRST START BIT TIMING**
18.6.5 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition (Figure 18-27) occurs when the RSEN bit of the SSPCON2 register is programmed high and the master state machine is no longer active. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA = 0) for one TBRG while SCL is high. SCL is asserted low. Following this, the RSEN bit of the SSPCON2 register will be automatically cleared and the Baud Rate Generator will not be reloading, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit of the SSPSTAT register will be set. The SSPIF bit will not be set until the Baud Rate Generator has timed out.

Note 1: If RSEN is programmed while any other event is in progress, it will not take effect.

2: A bus collision during the Repeated Start condition occurs if:
   • SDA is sampled low when SCL goes from low-to-high.
   • SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data ‘1’.

FIGURE 18-27: REPEAT START CONDITION WAVEFORM

18.6.6 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPBUF register. This action will set the Buffer Full flag bit, BF and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted. SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high. When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an ACK bit during the ninth bit time if an address match occurred, or if data was received properly. The status of ACKSTAT bit on the rising edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared. If not, the bit is set. After the ninth clock, the SSPIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 18-28).

After the write to the SSPBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/W bit are completed. On the falling edge of the eighth clock, the master will release the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the ACK bit is loaded into the ACKSTAT Status bit of the SSPCON2 register. Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.
18.6.6.1 BF Status Flag
In Transmit mode, the BF bit of the SSPSTAT register is set when the CPU writes to SSPBUF and is cleared when all eight bits are shifted out.

18.6.6.2 WCOL Status Flag
If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).
WCOL must be cleared by software before the next transmission.

18.6.6.3 ACKSTAT Status Flag
In Transmit mode, the ACKSTAT bit of the SSPCON2 register is cleared when the slave has sent an Acknowledge (ACK = 0) and is set when the slave does not Acknowledge (ACK = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

18.6.6.4 Typical Transmit Sequence
1. The user generates a Start condition by setting the SEN bit of the SSPCON2 register.
2. SSPIF is set by hardware on completion of the Start.
3. SSPIF is cleared by software.
4. The MSSP1 module will wait the required start time before any other operation takes place.
5. The user loads the SSPBUF with the slave address to transmit.
6. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPBUF is written to.
7. The MSSP1 module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
8. The MSSP1 module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
9. The user loads the SSPBUF with eight bits of data.
10. Data is shifted out the SDA pin until all eight bits are transmitted.
11. The MSSP1 module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
12. Steps 8-11 are repeated for all transmitted data bytes.
13. The user generates a Stop or Restart condition by setting the PEN or RSEN bits of the SSPCON2 register. Interrupt is generated once the Stop/Restart condition is complete.
FIGURE 18-28: \( \text{i}^2\text{C} \) MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)

- **SDA**: Transmit Address to Slave
- **SCL**: SCL held low while CPU responds to SSPIF
- **SSPIF**: Cleared by software
- **BF (SSPSTAT<0>)**: SSPBUF written
- **SEN**: After Start condition, SEN cleared by hardware
- **R/W**: Transmitting Data or Second Half of 10-bit Address
- **ACK**: Transmitted Data or Second Half

**SSPBUF** written with 7-bit address and R/W start transmit

From slave, clear ACKSTAT bit SSPCON2<6>

Cleared by software service routine from SSP interrupt

Cleared by software

**SSPBUF** is written by software

Write SSPCON2<0> SEN = 1
Start condition begins

SEN = 0

Transmit Address to Slave

R/W = 0

ACK = 0

123456789 123456789 P

Cleared by software service routine

SSPBUF written by software

ACKSTAT in SSPCON2 = 1

Start condition begins

From slave, clear ACKSTAT bit SSPCON2<6>

Cleared by software service routine

SSPBUF is written

ACKSTAT in SSPCON2 = 1
18.6.7  I2C MASTER MODE RECEPTION

Master mode reception (Figure 18-29) is enabled by programming the Receive Enable bit, RCEN bit of the SSPCON2 register.

**Note:** The MSSP1 module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP1 is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable, ACKEN bit of the SSPCON2 register.

18.6.7.1  BF Status Flag
In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

18.6.7.2  SSPOV Status Flag
In receive operation, the SSPOV bit is set when eight bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

18.6.7.3  WCOL Status Flag
If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

18.6.7.4  Typical Receive Sequence
1. The user generates a Start condition by setting the SEN bit of the SSPCON2 register.
2. SSPIF is set by hardware on completion of the Start.
3. SSPIF is cleared by software.
4. User writes SSPBUF with the slave address to transmit and the R/W bit set.
5. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPBUF is written to.
6. The MSSP1 module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
7. The MSSP1 module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
8. User sets the RCEN bit of the SSPCON2 register and the Master clocks in a byte from the slave.
9. After the 8th falling edge of SCL, SSPIF and BF are set.
10. User clears SSPIF and reads the received byte from SSPBUF, which clears the BF flag.
11. User either clears the SSPCON2 ACKDT bit to receive another byte or sets the ACKDT bit to suppress further data and then initiates the acknowledge sequence by setting the ACKEN bit.
12. Master’s ACK or not ACK is clocked out to the slave and SSPIF is set.
13. User clears SSPIF.
14. Steps 8-13 are repeated for each received byte from the slave.
15. If the ACKDT bit was set in step 11, then the user can send a Stop to release the bus.
FIGURE 18-29: I2C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)

- **Write to SSPCON2<0> (SEN = 1), begin Start condition**
- **SEN = 0**
- **Write to SSPBUF occurs here, start XMIT**
- **Transmit Address to Slave**
  - A7 A6 A5 A4 A3 A2 A1 R/W
- **ACK from Slave**
- **Receiving Data from Slave**
  - D7 D6 D5 D4 D3 D2 D1 D0
  - ACK
- **Master configured as a receiver by programming SSPCON2<3> (RCEN = 1)**
- **RCEN cleared automatically**
- **ACK from Master**
  - SDA = ACKDT (SSPCON2<4>) = 0
  - SDA = 0, SCL = 1 while CPU responds to SSPIF
  - BF (SSPSTAT<0>) cleared by software
- **SSPIF**
- **Set SSPIF at end of receive**
- **Set P bit (SSPSTAT<4>) and SSPIF**
- **SSPOV is set because SSPBUF is still full**
- **ACKEN**
- **RCEN**
  - Master configured as a receiver by programming SSPCON2<3> (RCEN = 1)
  - RCEN cleared automatically
  - ACK from Master
  - SDA = ACKDT = 0
  - RCEN cleared automatically
18.6.8 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN bit of the SSPCON2 register. When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP1 module then goes into Idle mode (Figure 18-30).

18.6.8.1 WCOL Status Flag

If the user writes the SSPBUF 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).

FIGURE 18-30: ACKNOWLEDGE SEQUENCE WAVEFORM

![Acknowledge sequence waveform diagram]

Note: TBRG = one Baud Rate Generator period.

18.6.9 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN bit of the SSPCON2 register. At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to ‘0’. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit of the SSPSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 18-31).

18.6.9.1 WCOL Status Flag

If the user writes the SSPBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).
18.6.10 SLEEP OPERATION
While in Sleep mode, the I2C slave module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP1 interrupt is enabled).

18.6.11 EFFECTS OF A RESET
A Reset disables the MSSP1 module and terminates the current transfer.

18.6.12 MULTI-MASTER MODE
In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP1 module is disabled. Control of the I2C bus may be taken when the P bit of the SSPSTAT register is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the SSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDA line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLIF bit.

The states where arbitration can be lost are:
- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

18.6.13 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION
Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF and reset the I2C port to its Idle state (Figure 18-32).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are deasserted and the SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I2C bus is free, the user can resume communication by asserting a Start condition.

If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I2C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I2C bus can be taken when the P bit is set in the SSPSTAT register, or the bus is Idle and the S and P bits are cleared.
18.6.13.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the Start condition (Figure 18-33).
b) SCL is sampled low before SDA is asserted low (Figure 18-34).

During a Start condition, both the SDA and the SCL pins are monitored.

If the SDA pin is already low, or the SCL pin is already low, then all of the following occur:

• the Start condition is aborted,
• the BCLIF flag is set and
• the MSSP1 module is reset to its Idle state (Figure 18-33).

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded and counts down. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data ‘1’ during the Start condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 18-35). If, however, a ‘1’ is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to zero; if the SCL pin is sampled as ‘0’ during this time, a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

Note: The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.
FIGURE 18-33: BUS COLLISION DURING START CONDITION (SDA ONLY)

- SDA goes low before the SEN bit is set.
- Set BCLIF.
- S bit and SSPIF set because SDA = 0, SCL = 1.

SDA

SCL

SEN

SDA sampled low before Start condition if SDA = 1, SCL = 1

BCLIF

SSPIF

SDA sampled low before Start condition. Set BCLIF.

S bit and SSPIF set because SDA = 0, SCL = 1.

SEN cleared automatically because of bus collision.

SSPIF and BCLIF are cleared by software

Set SEN, enable Start condition if SDA = 1, SCL = 1

FIGURE 18-34: BUS COLLISION DURING START CONDITION (SCL = 0)

- SDA = 0, SCL = 1

SDA

SCL

SEN

BCLIF

S

SSPIF

SDA = 0, SCL = 1

SCL = 0 before SDA = 0, bus collision occurs. Set BCLIF.

Interruption cleared by software

`0`
FIGURE 18-35: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION

- **SDA**: SDA pulled low by other master. Reset BRG and assert SDA.
- **SCL**: Set SEN, enable Start sequence if SDA = 1, SCL = 1
- **SSPIF**: SDA = 0, SCL = 1, set SSPIF; Interrupts cleared by software.
18.6.13.2  Bus Collision During a Repeated Start Condition

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

a) A low level is sampled on SDA when SCL goes from low level to high level (Case 1).

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data ‘1’ (Case 2).

When the user releases SDA and the pin is allowed to float high, the BRG is loaded with SSPADD and counts down to zero. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data ‘0’, Figure 18-36).

If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data ‘1’ during the Repeated Start condition, see Figure 18-37.

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 18-36: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**

![Diagram of bus collision during a repeated start condition (Case 1)]

**FIGURE 18-37: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)**

![Diagram of bus collision during a repeated start condition (Case 2)]
18.6.13.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out (Case 1).

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high (Case 2).

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPADD and counts down to 0. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data ‘0’ (Figure 18-38). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data ‘0’ (Figure 18-39).

**FIGURE 18-38: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

![Diagram of bus collision during a stop condition (Case 1)]

**FIGURE 18-39: BUS COLLISION DURING A STOP CONDITION (CASE 2)**

![Diagram of bus collision during a stop condition (Case 2)]
### TABLE 18-3: SUMMARY OF REGISTERS ASSOCIATED WITH I²C OPERATION

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMROIF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>57</td>
</tr>
<tr>
<td>PIE1</td>
<td>—</td>
<td>ADIE</td>
<td>—</td>
<td>—</td>
<td>SSPIE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>PIE2</td>
<td>—</td>
<td>—</td>
<td>ADIF</td>
<td>—</td>
<td>BCLIE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>59</td>
</tr>
<tr>
<td>PIR1</td>
<td>—</td>
<td>ADIF</td>
<td>—</td>
<td>—</td>
<td>SSPIF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>60</td>
</tr>
<tr>
<td>PIR2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BCLIF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>61</td>
</tr>
<tr>
<td>SSPADD</td>
<td></td>
<td>ADD&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>185</td>
</tr>
<tr>
<td>SSPBUF</td>
<td>MSSP1 Receive Buffer/Transmit Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>137*</td>
</tr>
<tr>
<td>SSPCON1</td>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPEN</td>
<td>CKP</td>
<td>SSPM&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td>182</td>
</tr>
<tr>
<td>SSPCON2</td>
<td>GCEN</td>
<td>ACKSTAT</td>
<td>ACKDT</td>
<td>ACKN</td>
<td>RCEN</td>
<td>PEN</td>
<td>RSEN</td>
<td>SEN</td>
<td>183</td>
</tr>
<tr>
<td>SSPCON3</td>
<td>ACKTIM</td>
<td>PCIE</td>
<td>SCIE</td>
<td>BOEN</td>
<td>SDAHT</td>
<td>SBCDE</td>
<td>AHEN</td>
<td>DHEN</td>
<td>184</td>
</tr>
<tr>
<td>SSPMSK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MSK&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td>185</td>
</tr>
<tr>
<td>SSPSTAT</td>
<td>SMP</td>
<td>CKE</td>
<td>D/A</td>
<td>P</td>
<td>S</td>
<td>R/W</td>
<td>UA</td>
<td>BF</td>
<td>181</td>
</tr>
<tr>
<td>TRISA</td>
<td>—</td>
<td>—</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>—(f)</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>87</td>
</tr>
</tbody>
</table>

**Legend:**
- — = unimplemented location, read as ‘0’.
- Shaded cells are not used by the MSSP module in I²C mode.
- * Page provides register information.

**Note 1:** Unimplemented, read as ‘1’.
18.7 BAUD RATE GENERATOR

The MSSP1 module has a Baud Rate Generator available for clock generation in both I2C and SPI Master modes. The Baud Rate Generator (BRG) reload value is placed in the SSPADD register (Register 18-6). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting down.

Once the given operation is complete, the internal clock will automatically stop counting and the clock pin will remain in its last state.

An internal signal “Reload” in Figure 18-40 triggers the value from SSPADD to be loaded into the BRG counter. This occurs twice for each oscillation of the module clock line. The logic dictating when the reload signal is asserted depends on the mode the MSSP1 is being operated in.

Table 18-4 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

EQUATION 18-1:

\[ F_{\text{CLK}} = \frac{F_{\text{OSC}}}{(SSPxADD + 1)(4)} \]

FIGURE 18-40: BAUD RATE GENERATOR BLOCK DIAGRAM

TABLE 18-4: MSSP1 CLOCK RATE W/BRG

<table>
<thead>
<tr>
<th>Fosc</th>
<th>Fcy</th>
<th>BRG Value</th>
<th>FCLOCK (2 Rollovers of BRG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 MHz</td>
<td>8 MHz</td>
<td>13h</td>
<td>400 kHz(1)</td>
</tr>
<tr>
<td>32 MHz</td>
<td>8 MHz</td>
<td>19h</td>
<td>308 kHz</td>
</tr>
<tr>
<td>32 MHz</td>
<td>8 MHz</td>
<td>4Fh</td>
<td>100 kHz</td>
</tr>
<tr>
<td>16 MHz</td>
<td>4 MHz</td>
<td>09h</td>
<td>400 kHz(1)</td>
</tr>
<tr>
<td>16 MHz</td>
<td>4 MHz</td>
<td>0Ch</td>
<td>308 kHz</td>
</tr>
<tr>
<td>16 MHz</td>
<td>4 MHz</td>
<td>27h</td>
<td>100 kHz</td>
</tr>
<tr>
<td>4 MHz</td>
<td>1 MHz</td>
<td>09h</td>
<td>100 kHz</td>
</tr>
</tbody>
</table>

Note 1: Refer to the I/O port electrical and timing specifications in Table 21-3 and Figure 21-5 to ensure the system is designed to support the I/O timing requirements.

2: The I2C interface does not conform to the 400 kHz I2C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.
18.7.1 ALTERNATE PIN LOCATIONS

This module incorporates I/O pins that can be moved to other locations with the use of the alternate pin function register, APFCON. To determine which pins can be moved and what their default locations are upon a Reset, see Section 11.1 “Alternate Pin Function” for more information.
### 18.8 Register Definitions: MSSP Control

#### REGISTER 18-1: SSPSTAT: SSP STATUS REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
<th>R-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMP</td>
<td>CKE</td>
<td>D/A</td>
<td>P</td>
<td>S</td>
<td>R/W</td>
<td>UA</td>
<td>BF</td>
</tr>
</tbody>
</table>

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

- **‘1’** = Bit is set
- **‘0’** = Bit is cleared

#### bit 7
**SMP**: SPI Data Input Sample bit
- **SPI Master mode:**
  - 1 = Input data sampled at end of data output time
  - 0 = Input data sampled at middle of data output time
- **SPI Slave mode:**
  - SPI must be cleared when SPI is used in Slave mode
  - **In **I²C** Master or Slave mode:**
    - 1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)
    - 0 = Slew rate control enabled for High Speed mode (400 kHz)

#### bit 6
**CKE**: SPI Clock Edge Select bit (SPI mode only)
- **In SPI Master or Slave mode:**
  - 1 = Transmit occurs on transition from active to Idle clock state
  - 0 = Transmit occurs on transition from Idle to active clock state
- **In **I²C** mode only:**
  - 1 = Enable input logic so that thresholds are compliant with SMBus specification
  - 0 = Disable SMBus specific inputs

#### bit 5
**D/A**: Data/Address bit (**I²C** mode only)
- 1 = Indicates that the last byte received or transmitted was data
- 0 = Indicates that the last byte received or transmitted was address

#### bit 4
**P**: Stop bit
- **(I²C mode only. This bit is cleared when the MSSP1 module is disabled, SSPEN is cleared.)**
  - 1 = Indicates that a Stop bit has been detected last (this bit is ‘0’ on Reset)
  - 0 = Stop bit was not detected last

#### bit 3
**S**: Start bit
- **(I²C mode only. This bit is cleared when the MSSP1 module is disabled, SSPEN is cleared.)**
  - 1 = Indicates that a Start bit has been detected last (this bit is ‘0’ on Reset)
  - 0 = Start bit was not detected last

#### bit 2
**R/W**: Read/Write bit information (**I²C** mode only)
- This bit holds the **R/W** bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit, or not ACK bit.
- **In **I²C** Slave mode:**
  - 1 = Read
  - 0 = Write
- **In **I²C** Master mode:**
  - 1 = Transmit is in progress
  - 0 = Transmit is not in progress
  - OR-ing this bit with **SEN**, **RSEN**, **PEN**, **RCEN** or **ACKEN** will indicate if the MSSP1 is in Idle mode.

#### bit 1
**UA**: Update Address bit (**10-bit I²C** mode only)
- 1 = Indicates that the user needs to update the address in the **SSPADD** register
- 0 = Address does not need to be updated

#### bit 0
**BF**: Buffer Full Status bit
- **Receive (SPI and **I²C** modes):**
  - 1 = Receive complete, **SSPBUF** is full
  - 0 = Receive not complete, **SSPBUF** is empty
- **Transmit (**I²C** mode only):**
  - 1 = Data transmit in progress (does not include the **ACK** and Stop bits), **SSPBUF** is full
  - 0 = Data transmit complete (does not include the **ACK** and Stop bits), **SSPBUF** is empty
REGISTER 18-2: SSPCON1: SSP CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>R/C/HS-0/0</th>
<th>R/C/HS-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPEN</td>
<td>CKP</td>
<td>SSPM&lt;3:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7

WCOL: Write Collision Detect bit
- Master mode:
  - 1 = A write to the SSPBUF register was attempted while the I2C conditions were not valid for a transmission to be started
  - 0 = No collision
- Slave mode:
  - 1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
  - 0 = No collision

bit 6

SSPOV: Receive Overflow Indicator bit(1)
- In SPI mode:
  - 1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost.
  - 0 = No overflow
- In I2C mode:
  - 1 = A byte is received while the SSPBUF register is still holding the previous byte. SSPOV is a “don’t care” in Transmit mode (must be cleared in software).
  - 0 = No overflow

bit 5

SSPEN: Synchronous Serial Port Enable bit
- In both modes, when enabled, these pins must be properly configured as input or output
- In SPI mode:
  - 1 = Enables serial port and configures SCK, SDO, SDI and SS as the source of the serial port pins(2)
  - 0 = Disables serial port and configures these pins as I/O port pins
- In I2C mode:
  - 1 = Enables the serial port and configures the SDA and SCL pins as the source of the serial port pins(3)
  - 0 = Disables serial port and configures these pins as I/O port pins

bit 4

CKP: Clock Polarity Select bit
- In SPI mode:
  - 1 = Idle state for clock is a high level
  - 0 = Idle state for clock is a low level
- In I2C Slave mode:
  - 1 = Enable clock
  - 0 = Holds clock low (clock stretch). (Used to ensure data setup time.)
- In I2C Master mode:
  - Unused in this mode

bit 3-0

SSPM<3:0>: Synchronous Serial Port Mode Select bits
- 0000 = SPI Master mode, clock = Fosc/4
- 0001 = SPI Master mode, clock = Fosc/16
- 0010 = SPI Master mode, clock = Fosc/64
- 0011 = Reserved
- 0100 = SPI Slave mode, clock = SCK pin
- 0101 = SPI Slave mode, clock = SCK pin control enabled
- 0110 = I2C Slave mode, 7-bit address
- 0111 = I2C Slave mode, 10-bit address
- 1000 = I2C Master mode, clock = Fosc / (4 * (SSPADD+1))(4)
- 1001 = Reserved
- 1010 = SPI Master mode, clock = Fosc/(4 * (SSPADD+1))(5)
- 1011 = I2C firmware controlled Master mode (Slave idle)
- 1100 = Reserved
- 1101 = Reserved
- 1110 = I2C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
- 1111 = I2C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

Note
- 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.
- 2: When enabled, these pins must be properly configured as input or output.
- 3: When enabled, the SDA and SCL pins must be configured as inputs.
- 4: SSPADD values of 0, 1 or 2 are not supported for I2C mode.
- 5: SSPADD value of ‘0’ is not supported. Use SSPM = 0000 instead.
REGISTER 18-3: SSPCON2: SSP CONTROL REGISTER 2

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R-0/0</th>
<th>R/W-0/0</th>
<th>R/S-0/0</th>
<th>R/S/HS-0/0</th>
<th>R/S/HS-0/0</th>
<th>R/S/HS-0/0</th>
<th>R/W/HS-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCEN</td>
<td>ACKSTAT</td>
<td>ACKDT</td>
<td>ACKEN</td>
<td>RCEN</td>
<td>PEN</td>
<td>RSEN</td>
<td>SEN</td>
</tr>
</tbody>
</table>

bit 7    GCEN: General Call Enable bit (in I^2^C Slave mode only)
1 = Enable interrupt when a general call address (0x00 or 00h) is received in the SSPSR
0 = General call address disabled

bit 6    ACKSTAT: Acknowledge Status bit (in I^2^C mode only)
1 = Acknowledge was not received
0 = Acknowledge was received

bit 5    ACKDT: Acknowledge Data bit (in I^2^C mode only)
In Receive mode:
Value transmitted when the user initiates an Acknowledge sequence at the end of a receive
1 = Not Acknowledge
0 = Acknowledge

bit 4    ACKEN: Acknowledge Sequence Enable bit (in I^2^C Master mode only)
In Master Receive mode:
1 = Initiate Acknowledge sequence on SDA and SCL pins, and transmit ACKDT data bit.
   Automatically cleared by hardware.
0 = Acknowledge sequence idle

bit 3    RCEN: Receive Enable bit (in I^2^C Master mode only)
1 = Enables Receive mode for I^2^C
0 = Receive idle

bit 2    PEN: Stop Condition Enable bit (in I^2^C Master mode only)
SCK Release Control:
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1    RSEN: Repeated Start Condition Enable bit (in I^2^C Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

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

Note 1: For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I^2^C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).
REGISTER 18-4:  SSPCON3: SSP CONTROL REGISTER 3

<table>
<thead>
<tr>
<th>R-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKTIM</td>
<td>PCIE</td>
<td>SCIE</td>
<td>BOEN</td>
<td>SDAHT</td>
<td>SBCDE</td>
<td>AHEN</td>
<td>DHEN</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>ACKTIM: Acknowledge Time Status bit (I^2C mode only)(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Indicates the I^2C bus is in an Acknowledge sequence, set on 8^TH falling edge of SCL clock</td>
</tr>
<tr>
<td>0</td>
<td>Not an Acknowledge sequence, cleared on 9^TH rising edge of SCL clock</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6</th>
<th>PCIE: Stop Condition Interrupt Enable bit (I^2C mode only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enable interrupt on detection of Stop condition</td>
</tr>
<tr>
<td>0</td>
<td>Stop detection interrupts are disabled(2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5</th>
<th>SCIE: Start Condition Interrupt Enable bit (I^2C mode only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enable interrupt on detection of Start or Restart conditions</td>
</tr>
<tr>
<td>0</td>
<td>Start detection interrupts are disabled(2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 4</th>
<th>BOEN: Buffer Overwrite Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In SPI Slave mode:</td>
</tr>
<tr>
<td></td>
<td>1 = SSPBUF updates every time that a new data byte is shifted in ignoring the BF bit</td>
</tr>
<tr>
<td></td>
<td>0 = If new byte is received with BF bit of the SSPSTAT register already set, SSPOV bit of the SSPCON1 register is set, and the buffer is not updated</td>
</tr>
<tr>
<td></td>
<td>In I^2C Master mode and SPI Master mode:</td>
</tr>
<tr>
<td></td>
<td>This bit is ignored.</td>
</tr>
<tr>
<td></td>
<td>In I^2C Slave mode:</td>
</tr>
<tr>
<td></td>
<td>1 = SSPBUF is updated and ACK is generated for a received address/data byte, ignoring the state of the SSPOV bit only if the BF bit = 0.</td>
</tr>
<tr>
<td></td>
<td>0 = SSPBUF is only updated when SSPOV is clear</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 3</th>
<th>SDAHT: SDA Hold Time Selection bit (I^2C mode only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Minimum of 300 ns hold time on SDA after the falling edge of SCL</td>
</tr>
<tr>
<td>0</td>
<td>Minimum of 100 ns hold time on SDA after the falling edge of SCL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 2</th>
<th>SBCDE: Slave Mode Bus Collision Detect Enable bit (I^2C Slave mode only)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If on the rising edge of SCL, SDA is sampled low when the module is outputting a high state, the BCLIF bit of the PIR2 register is set, and bus goes idle</td>
</tr>
<tr>
<td></td>
<td>1 = Enable slave bus collision interrupts</td>
</tr>
<tr>
<td></td>
<td>0 = Slave bus collision interrupts are disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1</th>
<th>AHEN: Address Hold Enable bit (I^2C Slave mode only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Following the 8th falling edge of SCL for a matching received address byte; CKP bit of the SSPCON1 register will be cleared and the SCL will be held low.</td>
</tr>
<tr>
<td>0</td>
<td>Address holding is disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 0</th>
<th>DHEN: Data Hold Enable bit (I^2C Slave mode only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Following the 8th falling edge of SCL for a received data byte; slave hardware clears the CKP bit of the SSPCON1 register and SCL is held low.</td>
</tr>
<tr>
<td>0</td>
<td>Data holding is disabled</td>
</tr>
</tbody>
</table>

**Note 1:** For daisy-chained SPI operation; allows the user to ignore all but the last received byte. SSPOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPBUF.

**Note 2:** This bit has no effect in Slave modes that Start and Stop condition detection is explicitly listed as enabled.

**Note 3:** The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is set.
REGISTER 18-5: SSPMSK: SSP MASK REGISTER

<table>
<thead>
<tr>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MSK&lt;7:0&gt;</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

**bit 0**
- **MSK<0>:** Mask bit for \(i^2C\) Slave mode, 10-bit Address
  
  \(i^2C\) Slave mode, 10-bit address (SSPM<3:0> = 0111 or 1111):
  
  1 = The received address bit 0 is compared to SSPADD<0> to detect \(i^2C\) address match
  
  0 = The received address bit 0 is not used to detect \(i^2C\) address match

**REGISTER 18-6: SSPADD: MSSP1 ADDRESS AND BAUD RATE REGISTER (\(i^2C\) MODE)**

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD&lt;7:0&gt;</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Master mode:**

**bit 7-0**
- **ADD<7:0>:** Baud Rate Clock Divider bits
  
  SCL pin clock period = \(((ADD<7:0> + 1) \times 4)/FOSC\)

10-Bit Slave mode — Most Significant Address Byte:

**bit 7-3**
- **Not used:** Unused for Most Significant Address Byte. Bit state of this register is a "don't care". Bit pattern sent by master is fixed by \(i^2C\) specification and must be equal to '11110'. However, those bits are compared by hardware and are not affected by the value in this register.

**bit 2-1**
- **ADD<2:1>:** Two Most Significant bits of 10-bit address

**bit 0**
- **Not used:** Unused in this mode. Bit state is a "don't care".

10-Bit Slave mode — Least Significant Address Byte:

**bit 7-0**
- **ADD<7:0>:** Eight Least Significant bits of 10-bit address

7-Bit Slave mode:

**bit 7-1**
- **ADD<7:1>:** 7-bit address

**bit 0**
- **Not used:** Unused in this mode. Bit state is a "don't care".
19.0 IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

ICSP™ programming allows customers to manufacture circuit boards with unprogrammed devices. Programming can be done after the assembly process, allowing the device to be programmed with the most recent firmware or a custom firmware. Five pins are needed for ICSP™ programming:
• ICSPCLK
• ICSPDAT
• MCLR/VPP
• VDD
• VSS

In Program/Verify mode the Program Memory, User IDs and the Configuration Words are programmed through serial communications. The ICSPDAT pin is a bidirectional I/O used for transferring the serial data and the ICSPCLK pin is the clock input. For more information on ICSP™ refer to the “PIC12LF1552 Memory Programming Specification” (DS41642).

19.1 High-Voltage Programming Entry Mode

The device is placed into High-Voltage Programming Entry mode by holding the ICSPCLK and ICSPDAT pins low then raising the voltage on MCLR/VPP to VIHH.

19.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using VDD only, without high voltage. When the LVP bit of Configuration Words is set to ‘1’, the low-voltage ICSP programming entry is enabled. To disable the Low-Voltage ICSP mode, the LVP bit must be programmed to ‘0’.

Entry into the Low-Voltage Programming Entry mode requires the following steps:
1. MCLR is brought to VIL.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, MCLR must be held at VIL for as long as Program/Verify mode is to be maintained.

If low-voltage programming is enabled (LVP = 1), the MCLR Reset function is automatically enabled and cannot be disabled. See Section 6.5 “MCLR” for more information.

The LVP bit can only be reprogrammed to ‘0’ by using the High-Voltage Programming mode.

19.3 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 19-1.

FIGURE 19-1: ICD RJ-11 STYLE CONNECTOR INTERFACE

Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 19-2.
For additional interface recommendations, refer to your specific device programmer manual prior to PCB design.

It is recommended that isolation devices be used to separate the programming pins from other circuitry. The type of isolation is highly dependent on the specific application and may include devices such as resistors, diodes, or even jumpers. See Figure 19-3 for more information.

FIGURE 19-3: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING

* Isolation devices (as required).
20.0 INSTRUCTION SET SUMMARY

Each instruction is a 14-bit word containing the operation code (opcode) and all required operands. The opcodes are broken into three broad categories.

• Byte Oriented
• Bit Oriented
• Literal and Control

The literal and control category contains the most varied instruction word format.

Table 20-3 lists the instructions recognized by the MPASM™ assembler.

All instructions are executed within a single instruction cycle, with the following exceptions, which may take two or three cycles:

• Subroutine takes two cycles (CALL, CALLW)
• Returns from interrupts or subroutines take two cycles (RETURN, RETLW, RETFIE)
• Program branching takes two cycles (GOTO, BRA, BRW, BTFS, BTFC, DECFSZ, INCFSZ)
• 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.

20.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified, and the result is stored according to either the instruction, or the destination designator ‘d’. A read operation is performed on a register even if the instruction writes to that register.

TABLE 20-1: OPCODE FIELD DESCRIPTIONS

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>Register file address (0x00 to 0x7F)</td>
</tr>
<tr>
<td>W</td>
<td>Working register (accumulator)</td>
</tr>
<tr>
<td>b</td>
<td>Bit address within an 8-bit file register</td>
</tr>
<tr>
<td>k</td>
<td>Literal field, constant data or label</td>
</tr>
<tr>
<td>x</td>
<td>Don’t care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.</td>
</tr>
<tr>
<td>d</td>
<td>Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.</td>
</tr>
<tr>
<td>n</td>
<td>FSR or INDF number. (0-1)</td>
</tr>
<tr>
<td>mm</td>
<td>Pre-post increment-decrement mode selection</td>
</tr>
</tbody>
</table>

TABLE 20-2: ABBREVIATION DESCRIPTIONS

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>Program Counter</td>
</tr>
<tr>
<td>TO</td>
<td>Time-out bit</td>
</tr>
<tr>
<td>C</td>
<td>Carry bit</td>
</tr>
<tr>
<td>DC</td>
<td>Digit carry bit</td>
</tr>
<tr>
<td>Z</td>
<td>Zero bit</td>
</tr>
<tr>
<td>PD</td>
<td>Power-down bit</td>
</tr>
</tbody>
</table>
**FIGURE 20-1: GENERAL FORMAT FOR INSTRUCTIONS**

**Byte-oriented file register operations**

<table>
<thead>
<tr>
<th>13</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>d</td>
<td>f (FILE #)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

d = 0 for destination W  
d = 1 for destination f  
f = 7-bit file register address

**Bit-oriented file register operations**

<table>
<thead>
<tr>
<th>13</th>
<th>10</th>
<th>9</th>
<th>7</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>b (BIT #)</td>
<td>f (FILE #)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b = 3-bit address  
f = 7-bit file register address

**Literal and control operations**

**General**

<table>
<thead>
<tr>
<th>13</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

k = 8-bit immediate value

**CALL and GOTO instructions only**

<table>
<thead>
<tr>
<th>13</th>
<th>11</th>
<th>10</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

k = 11-bit immediate value

**MOVLP instruction only**

<table>
<thead>
<tr>
<th>13</th>
<th>7</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

k = 7-bit immediate value

**MOVLB instruction only**

<table>
<thead>
<tr>
<th>13</th>
<th>5</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

k = 5-bit immediate value

**BRA instruction only**

<table>
<thead>
<tr>
<th>13</th>
<th>9</th>
<th>8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

k = 9-bit immediate value

**FSR Offset instructions**

<table>
<thead>
<tr>
<th>13</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>n</td>
<td>k (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

n = appropriate FSR  
k = 6-bit immediate value

**FSR Increment instructions**

<table>
<thead>
<tr>
<th>13</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>n</td>
<td>m (mode)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

n = appropriate FSR  
m = 2-bit mode value

**OPCODE only**

<table>
<thead>
<tr>
<th>13</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td></td>
</tr>
</tbody>
</table>
## TABLE 20-3: PIC12LF1552 INSTRUCTION SET

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>14-Bit Opcode</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MSb</td>
<td>LSb</td>
<td></td>
</tr>
<tr>
<td><strong>BYTE-ORIENTED FILE REGISTER OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDWF f, d</td>
<td>Add W and f</td>
<td>1</td>
<td>00 1111 dfff ffff</td>
<td>C, DC, Z</td>
<td>2</td>
</tr>
<tr>
<td>ADDWFC f, d</td>
<td>Add with Carry W and f</td>
<td>1</td>
<td>11 1101 dfff ffff</td>
<td>C, DC, Z</td>
<td>2</td>
</tr>
<tr>
<td>ANDWF f, d</td>
<td>AND W with f</td>
<td>1</td>
<td>00 0101 dfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>ASRF f, d</td>
<td>Arithmetic Right Shift</td>
<td>1</td>
<td>11 0111 dfff ffff</td>
<td>C, DC, Z</td>
<td>2</td>
</tr>
<tr>
<td>LSLF f, d</td>
<td>Logical Left Shift</td>
<td>1</td>
<td>11 0101 dfff ffff</td>
<td>C, Z</td>
<td>2</td>
</tr>
<tr>
<td>LSRF f, d</td>
<td>Logical Right Shift</td>
<td>1</td>
<td>11 0110 dfff ffff</td>
<td>C, Z</td>
<td>2</td>
</tr>
<tr>
<td>CLRWF f, d</td>
<td>Clear f</td>
<td>1</td>
<td>00 0000 1fff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>CLRWF –</td>
<td>Clear W</td>
<td></td>
<td>00 0000 0000 00xx</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>COMF f, d</td>
<td>Complement f</td>
<td>1</td>
<td>00 1001 dfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>DECWF f, d</td>
<td>Decrement f</td>
<td>1</td>
<td>00 0011 dfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>INCWF f, d</td>
<td>Increment f</td>
<td>1</td>
<td>00 1010 dfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>IORWF f, d</td>
<td>Inclusive OR W with f</td>
<td>1</td>
<td>00 0100 dfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>MOVWF f, d</td>
<td>Move f</td>
<td>1</td>
<td>00 1000 dfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>MOVWF f, d</td>
<td>Move W to f</td>
<td>1</td>
<td>00 0000 1fff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>RLWF f, d</td>
<td>Rotate Left f through Carry</td>
<td>1</td>
<td>00 1101 dfff ffff</td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>RRWF f, d</td>
<td>Rotate Right f through Carry</td>
<td>1</td>
<td>00 1100 dfff ffff</td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>SUBWF f, d</td>
<td>Subtract W from f</td>
<td>1</td>
<td>00 0010 dfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>SWAPWF f, d</td>
<td>Swap nibbles in f</td>
<td>1</td>
<td>00 1110 dfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>XORWF f, d</td>
<td>Exclusive OR W with f</td>
<td>1</td>
<td>00 0110 dfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td><strong>BYTE ORIENTED SKIP OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECFSZ f, d</td>
<td>Decrement f, Skip if 0</td>
<td>1(2)</td>
<td>00 1111 dfff ffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INCFSZ f, d</td>
<td>Increment f, Skip if 0</td>
<td>1(2)</td>
<td>00 1111 dfff ffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BIT-ORIENTED REGISTER OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCF f, b</td>
<td>Bit Clear f</td>
<td>1</td>
<td>01 00bb bfff ffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BSF f, b</td>
<td>Bit Set f</td>
<td>1</td>
<td>01 01bb bfff ffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>BIT-ORIENTED SKIP OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BTFSC f, b</td>
<td>Bit Test f, Skip if Clear</td>
<td>1 (2)</td>
<td>01 10bb bfff ffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BTFSS f, b</td>
<td>Bit Test f, Skip if Set</td>
<td>1 (2)</td>
<td>01 11bb bfff ffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LITERAL OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDLW k</td>
<td>Add literal and W</td>
<td>1</td>
<td>11 1110 kkkk kkkk</td>
<td>C, DC, Z</td>
<td></td>
</tr>
<tr>
<td>ANDLW k</td>
<td>AND literal with W</td>
<td>1</td>
<td>11 1001 kkkk kkkk</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>IORLW k</td>
<td>Inclusive OR literal with W</td>
<td>1</td>
<td>11 1000 kkkk kkkk</td>
<td>Z</td>
<td></td>
</tr>
<tr>
<td>MOVLB k</td>
<td>Move literal to BSR</td>
<td>1</td>
<td>00 0000 001k kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVLP k</td>
<td>Move literal to PCLATH</td>
<td>1</td>
<td>11 0001 1kkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVLL k</td>
<td>Move literal to W</td>
<td>1</td>
<td>11 0000 kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBLW k</td>
<td>Subtract W from literal</td>
<td>1</td>
<td>11 1100 kkkk kkkk</td>
<td>C, DC, Z</td>
<td></td>
</tr>
<tr>
<td>XORLW k</td>
<td>Exclusive OR literal with W</td>
<td>1</td>
<td>11 1010 kkkk kkkk</td>
<td>Z</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

**Note 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.
## PIC12LF1552 INSTRUCTION SET (CONTINUED)

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>14-Bit Opcode</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CONTROL OPERATIONS</strong></td>
<td></td>
<td></td>
<td>MSb</td>
<td>LSb</td>
<td></td>
</tr>
<tr>
<td>BRA k</td>
<td>Relative Branch</td>
<td>2</td>
<td>11 001k kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRW –</td>
<td>Relative Branch with W</td>
<td>2</td>
<td>00 0000 0000 1011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALL k</td>
<td>Call Subroutine</td>
<td>2</td>
<td>10 0kkk kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CALLW –</td>
<td>Call Subroutine with W</td>
<td>2</td>
<td>00 0000 0000 1010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GOTO k</td>
<td>Go to address</td>
<td>2</td>
<td>10 1kkk kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETFIE k</td>
<td>Return from interrupt</td>
<td>2</td>
<td>00 0000 0000 1001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETLW k</td>
<td>Return with literal in W</td>
<td>2</td>
<td>11 0100 kkkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETURN –</td>
<td>Return from Subroutine</td>
<td>2</td>
<td>00 0000 0000 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>INHERENT OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLRWDW –</td>
<td>Clear Watchdog Timer</td>
<td>1</td>
<td>00 0000 0110 0100</td>
<td>TO, PD</td>
<td></td>
</tr>
<tr>
<td>NOP –</td>
<td>No Operation</td>
<td>1</td>
<td>00 0000 0000 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTION –</td>
<td>Load OPTION_REG register with W</td>
<td>1</td>
<td>00 0000 0110 0010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESET –</td>
<td>Software device Reset</td>
<td>1</td>
<td>00 0000 0000 0001</td>
<td>TO, PD</td>
<td></td>
</tr>
<tr>
<td>SLEEP –</td>
<td>Go into Standby mode</td>
<td>1</td>
<td>00 0000 0110 0011</td>
<td>TO, PD</td>
<td></td>
</tr>
<tr>
<td>TRIS f</td>
<td>Load TRIS register with W</td>
<td>1</td>
<td>00 0000 0110 0fff</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C-COMPILER OPTIMIZED</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDFSR n, k</td>
<td>Add Literal k to FSRn</td>
<td>1</td>
<td>11 0001 0nkk kkkk</td>
<td>Z</td>
<td>2, 3</td>
</tr>
<tr>
<td>MOVIW n mm</td>
<td>Move Indirect FSRn to W with pre/post inc/dec modifier, mm</td>
<td>1</td>
<td>00 0000 0001 0mm</td>
<td>kkkk</td>
<td></td>
</tr>
<tr>
<td>MOVWI n mm</td>
<td>Move W to Indirect FSRn with pre/post inc/dec modifier, mm</td>
<td>1</td>
<td>11 1111 0nkk 1mm</td>
<td>Z</td>
<td>2</td>
</tr>
</tbody>
</table>

**Note 1:** If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

**Note 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.

**Note 3:** See Table in the MOVIW and MOVWI instruction descriptions.
### 20.2 Instruction Descriptions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADDFSR</strong></td>
<td>Add Literal to FSRn</td>
</tr>
<tr>
<td>Syntax:</td>
<td>[ label ] ADDFSR  FSRn, k</td>
</tr>
<tr>
<td>Operands:</td>
<td>-32 ≤ k ≤ 31</td>
</tr>
<tr>
<td>Operation:</td>
<td>FSR(n) + k → FSR(n)</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Description:</td>
<td>The signed 6-bit literal 'k' is added to the contents of the FSRnH:FSRnL register pair.</td>
</tr>
<tr>
<td></td>
<td>FSRn is limited to the range 0000h-FFFFh. Moving beyond these bounds will cause the FSR to wrap-around.</td>
</tr>
<tr>
<td><strong>ANDLW</strong></td>
<td>AND literal with W</td>
</tr>
<tr>
<td>Syntax:</td>
<td>[ label ] ANDLW  k</td>
</tr>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 255</td>
</tr>
<tr>
<td>Operation:</td>
<td>(W) .AND. (k) → (W)</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>Z</td>
</tr>
<tr>
<td>Description:</td>
<td>The contents of W register are AND’ed with the 8-bit literal 'k'. The result is placed in the W register.</td>
</tr>
</tbody>
</table>

| **ADDLW**          | Add literal and W                                                                                      |
| Syntax:            | [ label ] ADDLW  k                                                                                      |
| Operands:          | 0 ≤ k ≤ 255                                                                                           |
| Operation:         | (W) + k → (W)                                                                                          |
| Status Affected:   | C, DC, Z                                                                                               |
| Description:       | The contents of the W register are added to the 8-bit literal 'k' and the result is placed in the W register.|

| **ADDWF**          | Add W and f                                                                                           |
| Syntax:            | [ label ] ADDWF  f,d                                                                                  |
| Operands:          | 0 ≤ f ≤ 127                                                                                           |
| Operation:         | (W) + (f) → (destination)                                                                             |
| Status Affected:   | C, DC, Z                                                                                               |
| Description:       | Add the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.|

| **ANDWF**          | AND W with f                                                                                           |
| Syntax:            | [ label ] ANDWF  f,d                                                                                  |
| Operands:          | 0 ≤ f ≤ 127                                                                                           |
| Operation:         | (W) .AND. (f) → (destination)                                                                           |
| Status Affected:   | Z                                                                                                       |
| Description:       | AND the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.|

| **ASRF**           | Arithmetic Right Shift                                                                                 |
| Syntax:            | [ label ] ASRF  f,d                                                                                     |
| Operands:          | 0 ≤ f ≤ 127                                                                                           |
| Operation:         | (f<7>) → dest<7>, (f<7:1>) → dest<6:0>, (f<0>) → C                                                      |
| Status Affected:   | C, Z                                                                                                    |
| Description:       | The contents of register 'f' are shifted one bit to the right through the Carry flag. The MSb remains unchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'.|

| **ADDWFC**         | ADD W and CARRY bit to f                                                                                 |
| Syntax:            | [ label ] ADDWFC  f,(d)                                                                                 |
| Operands:          | 0 ≤ f ≤ 127                                                                                           |
| 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'.|
### BCF  Bit Clear f

**Syntax:**
\[
\text{[ label] BCF f, b}
\]

**Operands:**
- \(0 \leq f \leq 127\)
- \(0 \leq b \leq 7\)

**Operation:**
\[0 \rightarrow (\langle b \rangle)\]

**Status Affected:** None

**Description:** Bit ‘b’ in register ‘f’ is cleared.

### BRA  Relative Branch

**Syntax:**
\[
\text{[ label] BRA label}
\]

**Operands:**
- \(-256 \leq \text{label} - \text{PC} + 1 \leq 255\)
- \(-256 \leq k \leq 255\)

**Operation:**
\[(\text{PC} + 1 + k) \rightarrow \text{PC}\]

**Status Affected:** None

**Description:** Add the signed 9-bit literal ‘k’ to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be \(\text{PC} + 1 + k\). This instruction is a 2-cycle instruction. This branch has a limited range.

### BRW  Relative Branch with W

**Syntax:**
\[
\text{[ label] BRW W}
\]

**Operands:** None

**Operation:**
\[(\text{PC} + (W)) \rightarrow \text{PC}\]

**Status Affected:** None

**Description:** Add the contents of W (unsigned) to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be \(\text{PC} + 1 + (W)\). This instruction is a 2-cycle instruction.

### BSF  Bit Set f

**Syntax:**
\[
\text{[ label] BSF f, b}
\]

**Operands:**
- \(0 \leq f \leq 127\)
- \(0 \leq b \leq 7\)

**Operation:**
\[1 \rightarrow (\langle b \rangle)\]

**Status Affected:** None

**Description:** Bit ‘b’ in register ‘f’ is set.

### BTFSC  Bit Test f, Skip if Clear

**Syntax:**
\[
\text{[ label] BTFSC f, b}
\]

**Operands:**
- \(0 \leq f \leq 127\)
- \(0 \leq b \leq 7\)

**Operation:**
skip if \(\langle \langle b \rangle \rangle = 0\)

**Status Affected:** None

**Description:** If bit ‘b’ in register ‘f’ is ‘1’, the next instruction is executed. If bit ‘b’, in register ‘f’, is ‘0’, the next instruction is discarded, and a \texttt{NOP} is executed instead, making this a 2-cycle instruction.

### BTFSS  Bit Test f, Skip if Set

**Syntax:**
\[
\text{[ label] BTFSS f, b}
\]

**Operands:**
- \(0 \leq f \leq 127\)
- \(0 \leq b < 7\)

**Operation:**
skip if \(\langle \langle b \rangle \rangle = 1\)

**Status Affected:** None

**Description:** If bit ‘b’ in register ‘f’ is ‘0’, the next instruction is executed. If bit ‘b’ is ‘1’, then the next instruction is discarded and a \texttt{NOP} is executed instead, making this a 2-cycle instruction.
**CALL**

**Call Subroutine**

Syntax:  
\[
\text{[ label]} \text{ CALL } k
\]

Operands:  
\[0 \leq k \leq 2047\]

Operation:  
\( (PC)+1 \rightarrow \text{TOS}, \)
\( k \rightarrow PC<10:0>, \)
\( (PCLATH<6:3>) \rightarrow PC<14:11> \)

Status Affected: None

Description: Call Subroutine. First, return address (PC + 1) is pushed onto the stack. The 11-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a 2-cycle instruction.

**CALLW**

**Subroutine Call With W**

Syntax:  
\[
\text{[ label]} \text{ CALLW}
\]

Operands: None

Operation:  
\( (PC) + 1 \rightarrow \text{TOS}, \)
\( (W) \rightarrow PC<7:0>, \)
\( (PCLATH<6:0>) \rightarrow PC<14:8> \)

Status Affected: None

Description: Subroutine call with W. First, the return address (PC + 1) is pushed onto the return stack. Then, the contents of W is loaded into PC<7:0>, and the contents of PCLATH into PC<14:8>. CALLW is a 2-cycle instruction.

**CLRF**

**Clear f**

Syntax:  
\[
\text{[ label]} \text{ CLRF } f
\]

Operands:  
\[0 \leq f \leq 127\]

Operation:  
\( 00h \rightarrow f \)
\( 1 \rightarrow Z \)

Status Affected: Z

Description: The contents of register 'f' are cleared and the Z bit is set.

**CLRW**

**Clear W**

Syntax:  
\[
\text{[ label]} \text{ CLRW}
\]

Operands: None

Operation:  
\( 00h \rightarrow W \)
\( 1 \rightarrow Z \)

Status Affected: Z

Description: W register is cleared. Zero bit (Z) is set.

**CLRWDT**

**Clear Watchdog Timer**

Syntax:  
\[
\text{[ label]} \text{ CLRWDT}
\]

Operands: None

Operation:  
\( 00h \rightarrow \text{WDT} \)
\( 0 \rightarrow \text{WDT prescaler}, \)
\( 1 \rightarrow \text{TO} \)
\( 1 \rightarrow \text{PD} \)

Status Affected: TO, PD

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are set.

**COMF**

**Complement f**

Syntax:  
\[
\text{[ label]} \text{ COMF } f,d
\]

Operands:  
\[0 \leq f \leq 127\]
\[d \in [0,1]\]

Operation:  
\( (f) \rightarrow (\text{destination}) \)

Status Affected: Z

Description: The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

**DECf**

**Decrement f**

Syntax:  
\[
\text{[ label]} \text{ DECF } f,d
\]

Operands:  
\[0 \leq f \leq 127\]
\[d \in [0,1]\]

Operation:  
\( (f) - 1 \rightarrow (\text{destination}) \)

Status Affected: Z

Description: Decrement register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.
### DECFSZ  
**Decrement f, Skip if 0**

<table>
<thead>
<tr>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[ label ]</code> DECFSZ <code>f,d</code></td>
</tr>
<tr>
<td>Operands:</td>
</tr>
<tr>
<td><code>0 ≤ f ≤ 127</code></td>
</tr>
<tr>
<td><code>d ∈ [0,1]</code></td>
</tr>
<tr>
<td>Operation:</td>
</tr>
<tr>
<td><code>(f) - 1 → (destination);</code></td>
</tr>
<tr>
<td><code>skip if result = 0</code></td>
</tr>
<tr>
<td>Status Affected:</td>
</tr>
<tr>
<td>None</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>The contents of register <code>f</code> are decremented. If <code>d</code> is <code>'0'</code>, the result is placed in the W register. If <code>d</code> is <code>'1'</code>, the result is placed back in register <code>f</code>. If the result is <code>'1'</code>, the next instruction is executed. If the result is <code>'0'</code>, a NOP is executed instead, making it a 2-cycle instruction.</td>
</tr>
</tbody>
</table>

### INCFSZ  
**Increment f, Skip if 0**

<table>
<thead>
<tr>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[ label ]</code> INCFSZ <code>f,d</code></td>
</tr>
<tr>
<td>Operands:</td>
</tr>
<tr>
<td><code>0 ≤ f ≤ 127</code></td>
</tr>
<tr>
<td><code>d ∈ [0,1]</code></td>
</tr>
<tr>
<td>Operation:</td>
</tr>
<tr>
<td><code>(f) + 1 → (destination),</code></td>
</tr>
<tr>
<td><code>skip if result = 0</code></td>
</tr>
<tr>
<td>Status Affected:</td>
</tr>
<tr>
<td>None</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>The contents of register <code>f</code> are incremented. If <code>d</code> is <code>'0'</code>, the result is placed in the W register. If <code>d</code> is <code>'1'</code>, the result is placed back in register <code>f</code>. If the result is <code>'1'</code>, the next instruction is executed. If the result is <code>'0'</code>, a NOP is executed instead, making it a 2-cycle instruction.</td>
</tr>
</tbody>
</table>

### GOTO  
**Unconditional Branch**

<table>
<thead>
<tr>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[ label ]</code> GOTO <code>k</code></td>
</tr>
<tr>
<td>Operands:</td>
</tr>
<tr>
<td><code>0 ≤ k ≤ 2047</code></td>
</tr>
<tr>
<td>Operation:</td>
</tr>
<tr>
<td><code>k → PC&lt;10:0&gt;</code></td>
</tr>
<tr>
<td><code>PCLATH&lt;6:3&gt; → PC&lt;14:11&gt;</code></td>
</tr>
<tr>
<td>Status Affected:</td>
</tr>
<tr>
<td>None</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>GOTO is an unconditional branch. The 11-bit immediate value is loaded into PC bits <code>&lt;10:0&gt;</code>. The upper bits of PC are loaded from PCLATH&lt;4:3&gt;. GOTO is a 2-cycle instruction.</td>
</tr>
</tbody>
</table>

### IORLW  
**Inclusive OR literal with W**

<table>
<thead>
<tr>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[ label ]</code> IORLW <code>k</code></td>
</tr>
<tr>
<td>Operands:</td>
</tr>
<tr>
<td><code>0 ≤ k ≤ 255</code></td>
</tr>
<tr>
<td>Operation:</td>
</tr>
<tr>
<td><code>(W) .OR. k</code></td>
</tr>
<tr>
<td>Status Affected:</td>
</tr>
<tr>
<td>Z</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>The contents of the W register are OR'ed with the 8-bit literal 'k'. The result is placed in the W register.</td>
</tr>
</tbody>
</table>

### INCF  
**Increment f**

<table>
<thead>
<tr>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[ label ]</code> INCF <code>f,d</code></td>
</tr>
<tr>
<td>Operands:</td>
</tr>
<tr>
<td><code>0 ≤ f ≤ 127</code></td>
</tr>
<tr>
<td><code>d ∈ [0,1]</code></td>
</tr>
<tr>
<td>Operation:</td>
</tr>
<tr>
<td><code>(f) + 1 → (destination)</code></td>
</tr>
<tr>
<td>Status Affected:</td>
</tr>
<tr>
<td>Z</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>The contents of register <code>f</code> are incremented. If <code>d</code> is <code>'0'</code>, the result is placed in the W register. If <code>d</code> is <code>'1'</code>, the result is placed back in register <code>f</code>.</td>
</tr>
</tbody>
</table>

### IORWF  
**Inclusive OR W with f**

<table>
<thead>
<tr>
<th>Syntax:</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>[ label ]</code> IORWF <code>f,d</code></td>
</tr>
<tr>
<td>Operands:</td>
</tr>
<tr>
<td><code>0 ≤ f ≤ 127</code></td>
</tr>
<tr>
<td><code>d ∈ [0,1]</code></td>
</tr>
<tr>
<td>Operation:</td>
</tr>
<tr>
<td><code>(W) .OR. (f)</code></td>
</tr>
<tr>
<td>Status Affected:</td>
</tr>
<tr>
<td>Z</td>
</tr>
<tr>
<td>Description:</td>
</tr>
<tr>
<td>Inclusive OR the W register with register <code>f</code>. If <code>d</code> is <code>'0'</code>, the result is placed in the W register. If <code>d</code> is <code>'1'</code>, the result is placed back in register <code>f</code>.</td>
</tr>
</tbody>
</table>
### LSLF - Logical Left Shift

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[ <em>label</em> ] LSLF f {,d}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>0 ≤ f ≤ 127</td>
</tr>
<tr>
<td></td>
<td>d ∈ [0,1]</td>
</tr>
<tr>
<td>Operation:</td>
<td>(f&lt;7&gt;) → C</td>
</tr>
<tr>
<td></td>
<td>(f&lt;6:0&gt;) → dest&lt;7:1&gt;</td>
</tr>
<tr>
<td></td>
<td>0 → dest&lt;0&gt;</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>C, Z</td>
</tr>
<tr>
<td>Description:</td>
<td>The contents of register ‘f’ are shifted one bit to the left through the Carry flag. A ‘0’ is shifted into the LSb. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’.</td>
</tr>
</tbody>
</table>

![Diagram](image1)

### LSRF - Logical Right Shift

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[ <em>label</em> ] LSRF f {,d}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>0 ≤ f ≤ 127</td>
</tr>
<tr>
<td></td>
<td>d ∈ [0,1]</td>
</tr>
<tr>
<td>Operation:</td>
<td>0 → dest&lt;7&gt;</td>
</tr>
<tr>
<td></td>
<td>(f&lt;7:1&gt;) → dest&lt;6:0&gt;</td>
</tr>
<tr>
<td></td>
<td>(f&lt;0&gt;) → C</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>C, Z</td>
</tr>
<tr>
<td>Description:</td>
<td>The contents of register ‘f’ are shifted one bit to the right through the Carry flag. A ‘0’ is shifted into the MSb. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’.</td>
</tr>
</tbody>
</table>

![Diagram](image2)

### MOVF - Move f

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[ <em>label</em> ] MOVF f,d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>0 ≤ f ≤ 127</td>
</tr>
<tr>
<td></td>
<td>d ∈ [0,1]</td>
</tr>
<tr>
<td>Operation:</td>
<td>(f) → (dest)</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>Z</td>
</tr>
<tr>
<td>Description:</td>
<td>The contents of register f is moved to a destination dependent upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.</td>
</tr>
</tbody>
</table>

**Example:**

```
MOVF FSR, 0
```

**After Instruction**

- W = value in FSR register
- Z = 1
MOVIW  Move INDFn to W

Syntax:  
[ label ] MOVIW ++FSRn  
[ label ] MOVIW --FSRn  
[ label ] MOVIW FSRn++  
[ label ] MOVIW FSRn--  
[ label ] MOVIW k[FSRn]

Operands:  
n ∈ [0, 1]  
mm ∈ [00, 01, 10, 11]  
-32 ≤ k ≤ 31

Operation: INDFn → W  
Effective address is determined by  
• FSR + 1 (preincrement)  
• FSR - 1 (predecrement)  
• FSR + k (relative offset)  

After the Move, the FSR value will be either:  
• FSR + 1 (all increments)  
• FSR - 1 (all decrements)  
• Unchanged

Status Affected:  
Z

Mode  Syntax  mm
Preincrement  ++FSRn  00
Predecrement  --FSRn  01
Postincrement  FSRn++  10
Postdecrement  FSRn--  11

Description:  
This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it.

Note: The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn.

FSRn is limited to the range 0000h-FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wrap-around.

MOVLP  Move literal to PCLATH

Syntax:  
[ label ] MOVLP k

Operands:  
0 ≤ k ≤ 127

Operation:  
k → PCLATH

Status Affected:  
None

Description:  
The 7-bit literal ‘k’ is loaded into the PCLATH register.

MOVLW  Move literal to W

Syntax:  
[ label ] MOVLW k

Operands:  
0 ≤ k ≤ 255

Operation:  
k → W

Status Affected:  
None

Description:  
The 8-bit literal ‘k’ is loaded into W register. The “don’t cares” will assemble as ’0’s.

Words:  
1

Cycles:  
1

Example:  
MOVLW 0x5A

Before Instruction  
W = 0x4F

After Instruction  
W = 0x5A

MOVWF  Move W to f

Syntax:  
[ label ] MOVWF f

Operands:  
0 ≤ f ≤ 127

Operation:  
(W) → (f)

Status Affected:  
None

Description:  
Move data from W register to register ‘f’.

Words:  
1

Cycles:  
1

Example:  
MOVWF OPTION_REG

Before Instruction  
OPTION_REG = 0xFF
W = 0x4F

After Instruction  
OPTION_REG = 0x4F
W = 0x4F

MOVLB  Move literal to BSR

Syntax:  
[ label ] MOVLB k

Operands:  
0 ≤ k ≤ 15

Operation:  
k → BSR

Status Affected:  
None

Description:  
The 5-bit literal ‘k’ is loaded into the Bank Select Register (BSR).
MOVWI | Move W to INDFn

Syntax:
- \([ label \] \) MOVWI ++FSRn
- \([ label \] \) MOVWI --FSRn
- \([ label \] \) MOVWI FSRn++
- \([ label \] \) MOVWI FSRn--
- \([ label \] \) MOVWI k[FSRn]

Operands:
- \( n \in [0,1] \)
- \( mm \in [00,01,10,11] \)
- \(-32 \leq k \leq 31\)

Operation: \( W \rightarrow INDFn \)
Effective address is determined by:
- FSR + 1 (preincrement)
- FSR - 1 (predecrement)
- FSR + k (relative offset)

After the Move, the FSR value will be:
- FSR + 1 (all increments)
- FSR - 1 (all decrements)
- Unchanged

Status Affected: None

Description: This instruction is used to move data between W and one of the indirect registers (INDFn). Before/after this move, the pointer (FSRn) is updated by pre/post incrementing/decrementing it.

Note: The INDFn registers are not physical registers. Any instruction that accesses an INDFn register actually accesses the register at the address specified by the FSRn.

FSRn is limited to the range 0000h-FFFFh. Incrementing/decrementing it beyond these bounds will cause it to wrap-around.

The increment/decrement operation on FSRn WILL NOT affect any Status bits.

NOP | No Operation

Syntax: \([ label \] \) NOP

Operands: None

Operation: No operation

Status Affected: None

Description: No operation

Words: 1
Cycles: 1

Example:
NOP

OPTION | Load OPTION_REG Register with W

Syntax: \([ label \] \) OPTION

Operands: None

Operation: \((W) \rightarrow OPTION_REG\)

Status Affected: None

Description: Move data from W register to OPTION_REG register.

RESET | Software Reset

Syntax: \([ label \] \) RESET

Operands: None

Operation: Execute a device Reset. Resets the RI flag of the PCON register.

Status Affected: None

Description: This instruction provides a way to execute a hardware Reset by software.
**RETIE**  Return from Interrupt

Syntax:  \[ \text{label} \] RETFIE \ k
Operands:  None
Operation:  TOS \rightarrow PC,  
            \ 1 \rightarrow \text{GIE}
Status Affected:  None
Description:  Return from Interrupt. Stack is POPed 
              and Top-of-Stack (TOS) is loaded in 
              the PC. Interrupts are enabled by 
              setting Global Interrupt Enable bit, 
              GIE (INTCON<7>). This is a 2-cycle 
              instruction.
Words:  1  
Cycles:  2
Example:  
After Interrupt 
PC = TOS 
GIE = 1

**RETLW**  Return with literal in W

Syntax:  \[ \text{label} \] RETLW \ k
Operands:  0 \leq \text{k} \leq 255
Operation:  \text{k} \rightarrow (W);  
            TOS \rightarrow PC
Status Affected:  None
Description:  The W register is loaded with the 8-bit 
              literal \text{k}. The program counter is 
              loaded from the top of the stack (the 
              return address). This is a 2-cycle 
              instruction.
Words:  1  
Cycles:  2
Example:  
CALL TABLE; W contains table  
          ; offset value  
          ; W now has table value  
TABLE  
ADDWF PC ;W = offset  
RETLW k1 ; Begin table  
RETLW k2 ;  
.  
.  
.  
RETLW kn ; End of table

Before Instruction 
W = 0x07
After Instruction 
W = value of k8

**RETURN**  Return from Subroutine

Syntax:  \[ \text{label} \] RETURN
Operands:  None
Operation:  TOS \rightarrow PC
Status Affected:  None
Description:  Return from subroutine. The stack is 
              POPed and the top of the stack (TOS) 
              is loaded into the program counter. 
              This is a 2-cycle instruction.

**RLF**  Rotate Left f through Carry

Syntax:  \[ \text{label} \] RLF \ f,d
Operands:  0 \leq \text{f} \leq 127  
            \ d \in \{0,1\}
Operation:  See description below
Status Affected:  C
Description:  The contents of register \text{f} are rotated 
              one bit to the left through the Carry 
              flag. If \text{d} is ‘0’, the result is placed in 
              the W register. If \text{d} is ‘1’, the result is 
              stored back in register \text{f}.
Words:  1  
Cycles:  1
Example:  
RLF REG1,0

Before Instruction  
REG1 = 1110 0110  
C = 0
After Instruction  
REG1 = 1110 0110  
W = 1100 1100  
C = 1
**PIC12LF1552**

<table>
<thead>
<tr>
<th>RRF</th>
<th>Rotate Right f through Carry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[label] RRF f,d</td>
</tr>
<tr>
<td>Operands:</td>
<td>$0 \leq f \leq 127$ $d \in {0,1}$</td>
</tr>
<tr>
<td>Operation:</td>
<td>See description below</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>C</td>
</tr>
<tr>
<td>Description:</td>
<td>The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SLEEP</th>
<th>Enter Sleep mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[label] SLEEP</td>
</tr>
<tr>
<td>Operands:</td>
<td>None</td>
</tr>
<tr>
<td>Operation:</td>
<td>$00h \rightarrow$ WDT, $0 \rightarrow$ WDT prescaler, $1 \rightarrow$ TO, $0 \rightarrow$ PD</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>TO, PD</td>
</tr>
<tr>
<td>Description:</td>
<td>The power-down Status bit, PD is cleared. Time-out Status bit, TO is set. Watchdog Timer and its prescaler are cleared. The processor is put into Sleep mode with the oscillator stopped.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBWFB</th>
<th>Subtract W from f with Borrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>SUBWFB f(.d)</td>
</tr>
<tr>
<td>Operands:</td>
<td>$0 \leq f \leq 127$ $d \in {0,1}$</td>
</tr>
<tr>
<td>Operation:</td>
<td>$(f) - (W) \rightarrow (\overline{B}) \rightarrow$ dest</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>C, DC, Z</td>
</tr>
<tr>
<td>Description:</td>
<td>Subtract W and the BORROW flag (CARRY) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBWF</th>
<th>Subtract W from f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[label] SUBWF f,d</td>
</tr>
<tr>
<td>Operands:</td>
<td>$0 \leq f \leq 127$ $d \in {0,1}$</td>
</tr>
<tr>
<td>Operation:</td>
<td>$(f) - (W) \rightarrow$ (destination)</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>C, DC, Z</td>
</tr>
<tr>
<td>Description:</td>
<td>Subtract (2's complement method) W register from register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUBLW</th>
<th>Subtract W from literal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[label] SUBLW k</td>
</tr>
<tr>
<td>Operands:</td>
<td>$0 \leq k \leq 255$</td>
</tr>
<tr>
<td>Operation:</td>
<td>$k - (W) \rightarrow (W)$</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>C, DC, Z</td>
</tr>
<tr>
<td>Description:</td>
<td>The W register is subtracted (2's complement method) from the 8-bit literal 'k'. The result is placed in the W register.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>W &gt; k</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DC</th>
<th>W&lt;3:0&gt; &gt; k&lt;3:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>W &lt; k</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DC</th>
<th>W&lt;3:0&gt; ≤ k&lt;3:0&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
### SWAPF

**Swap Nibbles in f**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[label] SWAPF f,d</th>
</tr>
</thead>
</table>
| Operands: | $0 \leq f \leq 127$  
| | $d \in \{0,1\}$  
| Operation: | $(f<3:0>) \rightarrow (\text{destination}<7:4>)$,  
| | $(f<7:4>) \rightarrow (\text{destination}<3:0>)$  
| Status Affected: | None |
| Description: | The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed in register 'f'. |

### XORLW

**Exclusive OR literal with W**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[label] XORLW k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>$0 \leq k \leq 255$</td>
</tr>
<tr>
<td>Operation:</td>
<td>$(W) \bowtie k \rightarrow (W)$</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>Z</td>
</tr>
<tr>
<td>Description:</td>
<td>The contents of the W register are XOR'ed with the 8-bit literal 'k'. The result is placed in the W register.</td>
</tr>
</tbody>
</table>

### TRIS

**Load TRIS Register with W**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[label] TRIS f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>$5 \leq f \leq 7$</td>
</tr>
<tr>
<td>Operation:</td>
<td>$(W) \rightarrow \text{TRIS register 'f'}$</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
</tbody>
</table>
| Description: | Move data from W register to TRIS register.  
| | When 'f' = 5, TRISA is loaded.  
| | When 'f' = 6, TRISB is loaded.  
| | When 'f' = 7, TRISC is loaded. |

### XORWF

**Exclusive OR W with f**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[label] XORWF f,d</th>
</tr>
</thead>
</table>
| Operands: | $0 \leq f \leq 127$  
| | $d \in \{0,1\}$  
| Operation: | $(W) \bowtie (f) \rightarrow (\text{destination})$ |
| Status Affected: | Z |
| Description: | Exclusive OR the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'. |
21.0 ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings\(^\dagger\)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambient temperature under bias</td>
<td>-40°C</td>
<td>+125°C</td>
</tr>
<tr>
<td>Storage temperature</td>
<td>-65°C</td>
<td>+150°C</td>
</tr>
<tr>
<td>Voltage on VDD with respect to VSS</td>
<td>-0.3V</td>
<td>+4.0V</td>
</tr>
<tr>
<td>Voltage on MCLR with respect to Vss</td>
<td>-0.3V</td>
<td>+9.0V</td>
</tr>
<tr>
<td>Voltage on all other pins with respect to Vss</td>
<td>-0.3V</td>
<td>(VDD + 0.3V)</td>
</tr>
<tr>
<td>Total power dissipation(^\dagger)</td>
<td></td>
<td>800 mW</td>
</tr>
<tr>
<td>Maximum current out of VSS pin, -40°C ≤ TA ≤ +85°C for industrial</td>
<td></td>
<td>210 mA</td>
</tr>
<tr>
<td>Maximum current out of VSS pin, -40°C ≤ TA ≤ +125°C for extended</td>
<td></td>
<td>95 mA</td>
</tr>
<tr>
<td>Maximum current into VDD pin, -40°C ≤ TA ≤ +85°C for industrial</td>
<td></td>
<td>150 mA</td>
</tr>
<tr>
<td>Maximum current into VDD pin, -40°C ≤ TA ≤ +125°C for extended</td>
<td></td>
<td>70 mA</td>
</tr>
<tr>
<td>Clamp current, IK (VPIN &lt; 0 or VPIN &gt; VDD)</td>
<td></td>
<td>±20 mA</td>
</tr>
<tr>
<td>Maximum output current sunk by any I/O pin</td>
<td></td>
<td>25 mA</td>
</tr>
<tr>
<td>Maximum output current sourced by any I/O pin</td>
<td></td>
<td>25 mA</td>
</tr>
</tbody>
</table>

Note 1: Power dissipation is calculated as follows: \( P_{\text{DIS}} = VDD \times (I_{DD} - \sum I_{OH}) + \sum \{VDD - VOH\} \times I_{OH} + \sum (VOL \times IOL) \).

\(^\dagger\) 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.

FIGURE 21-1: PIC12LF1552 VOLTAGE FREQUENCY GRAPH, -40°C ≤ TA ≤ +125°C

Note 1: The shaded region indicates the permissible combinations of voltage and frequency.
2: Refer to Table 21-1 for each Oscillator mode’s supported frequencies.
### 21.1 DC Characteristics: PIC12LF1552-I/E (Industrial, Extended)

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ. †</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D001</td>
<td>VDD</td>
<td>Supply Voltage (VDDMIN, VDDMAX)</td>
<td>1.8</td>
<td>—</td>
<td>3.6</td>
<td>V</td>
<td>FOSC ≤ 16 MHz: FOSC ≤ 32 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.5</td>
<td>—</td>
<td>3.6</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D002*</td>
<td>VDR</td>
<td>RAM Data Retention Voltage(1)</td>
<td>1.5</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>Device in Sleep mode</td>
</tr>
<tr>
<td>D002A*</td>
<td>VPOR*</td>
<td>Power-on Reset Release Voltage</td>
<td>—</td>
<td>1.6</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D002B*</td>
<td>VPORR*</td>
<td>Power-on Reset Rearm Voltage</td>
<td>—</td>
<td>0.8</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D003</td>
<td>VADFVR</td>
<td>Fixed Voltage Reference Voltage for ADC, Initial Accuracy</td>
<td>-7</td>
<td>—</td>
<td>6</td>
<td>%</td>
<td>1.024V, VDD ≥ 2.5V, 85°C (Note 2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-8</td>
<td>—</td>
<td>6</td>
<td>%</td>
<td>1.024V, VDD ≥ 2.5V, 125°C (Note 2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-7</td>
<td>—</td>
<td>6</td>
<td>%</td>
<td>2.048V, VDD ≥ 2.5V, 85°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-8</td>
<td>—</td>
<td>6</td>
<td>%</td>
<td>2.048V, VDD ≥ 2.5V, 125°C</td>
</tr>
<tr>
<td>D003C*</td>
<td>TCVFVR</td>
<td>Temperature Coefficient, Fixed Voltage Reference</td>
<td>—</td>
<td>-130</td>
<td>—</td>
<td>ppm/°C</td>
<td></td>
</tr>
<tr>
<td>D003D*</td>
<td>ΔVFVR/ΔVin</td>
<td>Line Regulation, Fixed Voltage Reference</td>
<td>—</td>
<td>0.270</td>
<td>—</td>
<td>%/V</td>
<td></td>
</tr>
<tr>
<td>D004*</td>
<td>SVDD</td>
<td>VDD Rise Rate to ensure internal Power-on Reset signal</td>
<td>0.05</td>
<td>—</td>
<td>—</td>
<td>V/ms</td>
<td>See Section 6.1 “Power-on Reset (POR)” for details.</td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.
† Data in “Typ.” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note:**
1. This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.
2. For proper operation, the minimum value of the ADC positive voltage reference must be 1.8V or greater. When selecting the FVR or the VREF+ pin as the source of the ADC positive voltage reference, be aware that the voltage must be 1.8V or greater.
**FIGURE 21-2: POR AND POR REARM WITH SLOW RISING VDD**

![Diagram of POR and POR rearm with slow rising VDD](image)

**Note**

1. When NPOR is low, the device is held in Reset.
2. TPOR 1 μs typical.
3. TVLOW 2.7 μs typical.
## 21.2 DC Characteristics: PIC12LF1552-I/E (Industrial, Extended)

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Device Characteristics</th>
<th>Min. Typ.†</th>
<th>Max. Units</th>
<th>Conditions</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>Supply Current (IDD)(1, 2)</td>
<td>Standard Operating Conditions (unless otherwise stated)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Operating temperature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-40°C ≤ TA ≤ +85°C for industrial</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-40°C ≤ TA ≤ +125°C for extended</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D010</td>
<td>—</td>
<td>2.5</td>
<td>18 µA</td>
<td>1.8</td>
<td>Fosc = 31 kHz</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>4</td>
<td>20 µA</td>
<td>3.0</td>
<td>LFINTOSC mode</td>
</tr>
<tr>
<td>D011</td>
<td>—</td>
<td>0.35</td>
<td>0.70 mA</td>
<td>1.8</td>
<td>Fosc = 8 MHz</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>0.5</td>
<td>1.10 mA</td>
<td>3.0</td>
<td>HFINTOSC mode</td>
</tr>
<tr>
<td>D012</td>
<td>—</td>
<td>0.5</td>
<td>1.2 mA</td>
<td>1.8</td>
<td>Fosc = 16 MHz</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>0.75</td>
<td>1.75 mA</td>
<td>3.0</td>
<td>HFINTOSC mode</td>
</tr>
<tr>
<td>D013</td>
<td>—</td>
<td>1.4</td>
<td>3.5 mA</td>
<td>3.0</td>
<td>Fosc = 32 MHz</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td>HFINTOSC mode with PLL</td>
</tr>
<tr>
<td>D014</td>
<td>—</td>
<td>3</td>
<td>17 µA</td>
<td>1.8</td>
<td>Fosc = 32 kHz</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>5</td>
<td>20 µA</td>
<td>3.0</td>
<td>ECL mode</td>
</tr>
<tr>
<td>D015</td>
<td>—</td>
<td>11</td>
<td>40 µA</td>
<td>1.8</td>
<td>Fosc = 500 kHz</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>16</td>
<td>60 µA</td>
<td>3.0</td>
<td>ECL mode</td>
</tr>
<tr>
<td>D016</td>
<td>—</td>
<td>23</td>
<td>65 µA</td>
<td>1.8</td>
<td>Fosc = 1 MHz</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>36</td>
<td>100 µA</td>
<td>3.0</td>
<td>ECM mode</td>
</tr>
<tr>
<td>D017</td>
<td>—</td>
<td>75</td>
<td>250 µA</td>
<td>1.8</td>
<td>Fosc = 4 MHz</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td>120</td>
<td>430 µA</td>
<td>3.0</td>
<td>ECM mode</td>
</tr>
<tr>
<td>D018</td>
<td>—</td>
<td>0.65</td>
<td>1.5 mA</td>
<td>3.0</td>
<td>Fosc = 20 MHz</td>
</tr>
<tr>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td>ECH mode</td>
</tr>
</tbody>
</table>

† Data in “Typ.” column is at 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The test conditions for all IDD measurements in active operation mode are: CLKIN = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD; MCLR = VDD; WDT disabled.

**Note 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.
## 21.3 DC Characteristics: PIC12LF1552-I/E (Power-Down)

### PIC12LF1552

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Device Characteristics</th>
<th>Min. Typ.</th>
<th>Max. +85°C</th>
<th>Max. +125°C</th>
<th>Units</th>
<th>Conditions</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>0.02</td>
<td>1.0</td>
<td>μA</td>
<td>1.8</td>
<td>WDT, BOR, FVR, and T1OSC disabled, all Peripherals Inactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>0.03</td>
<td>2</td>
<td>μA</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>D021</td>
<td>—</td>
<td>0.24</td>
<td>2</td>
<td>9</td>
<td>μA</td>
<td>1.8</td>
<td>LPWDT Current (Note 1)</td>
</tr>
<tr>
<td>D022</td>
<td>—</td>
<td>13</td>
<td>28</td>
<td>30</td>
<td>μA</td>
<td>1.8</td>
<td>FVR current (Note 1)</td>
</tr>
<tr>
<td>D023</td>
<td>—</td>
<td>6.5</td>
<td>17</td>
<td>20</td>
<td>μA</td>
<td>3.0</td>
<td>BOR Current (Note 1)</td>
</tr>
<tr>
<td>D024</td>
<td>—</td>
<td>0.1</td>
<td>4</td>
<td>10</td>
<td>μA</td>
<td>3.0</td>
<td>LPBOR Current</td>
</tr>
<tr>
<td>D025</td>
<td>—</td>
<td>0.03</td>
<td>3.5</td>
<td>9</td>
<td>μA</td>
<td>1.8</td>
<td>ADC Current (Note 1, Note 3), no conversion in progress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.04</td>
<td>4.0</td>
<td>10</td>
<td>μA</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>D026*</td>
<td></td>
<td>350</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>1.8</td>
<td>ADC Current (Note 1, Note 3), conversion in progress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>350</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>3.0</td>
<td></td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.

† Data in "Typ." column is at 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.

**Note 2:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD.

**Note 3:** ADC oscillator source is FRc.

---

21.3 DC Characteristics: PIC12LF1552-I/E (Power-Down)

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Device Characteristics</th>
<th>Min. Typ.</th>
<th>Max. +85°C</th>
<th>Max. +125°C</th>
<th>Units</th>
<th>Conditions</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>0.02</td>
<td>1.0</td>
<td>μA</td>
<td>1.8</td>
<td>WDT, BOR, FVR, and T1OSC disabled, all Peripherals Inactive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>—</td>
<td>0.03</td>
<td>2</td>
<td>μA</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>D021</td>
<td>—</td>
<td>0.24</td>
<td>2</td>
<td>9</td>
<td>μA</td>
<td>1.8</td>
<td>LPWDT Current (Note 1)</td>
</tr>
<tr>
<td>D022</td>
<td>—</td>
<td>13</td>
<td>28</td>
<td>30</td>
<td>μA</td>
<td>1.8</td>
<td>FVR current (Note 1)</td>
</tr>
<tr>
<td>D023</td>
<td>—</td>
<td>6.5</td>
<td>17</td>
<td>20</td>
<td>μA</td>
<td>3.0</td>
<td>BOR Current (Note 1)</td>
</tr>
<tr>
<td>D024</td>
<td>—</td>
<td>0.1</td>
<td>4</td>
<td>10</td>
<td>μA</td>
<td>3.0</td>
<td>LPBOR Current</td>
</tr>
<tr>
<td>D025</td>
<td>—</td>
<td>0.03</td>
<td>3.5</td>
<td>9</td>
<td>μA</td>
<td>1.8</td>
<td>ADC Current (Note 1, Note 3), no conversion in progress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.04</td>
<td>4.0</td>
<td>10</td>
<td>μA</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>D026*</td>
<td></td>
<td>350</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>1.8</td>
<td>ADC Current (Note 1, Note 3), conversion in progress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>350</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>3.0</td>
<td></td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.

† Data in "Typ." column is at 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.

**Note 2:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD.

**Note 3:** ADC oscillator source is FRc.
# DC Characteristics: PIC12LF1552-I/E

## DC CHARACTERISTICS

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
</table>

### VIL

| I/O PORT: | — | — | 0.15 VDD | V | 1.8V ≤ VDD ≤ 3.6V |
|—— | — | — | 0.2 VDD | V | 2.0V ≤ VDD ≤ 3.6V |

### VIH

| MCLR | — | — | 0.2 VDD | V | |

### VIN

| — | — | 0.2 VDD | V | |

### IIL

| — | ± 5 | ± 125 | nA | VSS ≤ VPIN ≤ VDD, Pin at high-impedance at 85°C |
|— | ± 5 | ± 1000 | nA | |
|— | ± 50 | ± 200 | nA | |

### IPUR

| — | 25 | 100 | 200 | μA | VDD = 3.3V, VPIN = VSS |

### VIL

| — | — | 6mA | V | IOL = 6mA, VDD = 3.3V |
|— | — | 1.8mA | V | IOL = 1.8mA, VDD = 1.8V |

### VOH

| — | 0.7 | — | — | V | IOH = 3mA, VDD = 3.3V |
|— | 1mA | — | — | V | IOH = 1mA, VDD = 1.8V |

### Capacitive Loading Specs on Output Pins

| All I/O pins | — | — | 50 | pF | |

* These parameters are characterized but not tested.
† Data in “Typ.” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Negative current is defined as current sourced by the pin.

**Note 2:** The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**Note 3:** Including OSC2 in CLKOUT mode.
### 21.5 Memory Programming Requirements

**DC CHARACTERISTICS**

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D110</td>
<td>VIHH</td>
<td>Voltage on MCLR/VPP pin</td>
<td>8.0</td>
<td>—</td>
<td>9.0</td>
<td>V</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>D111</td>
<td>IDDp</td>
<td>Supply Current during Programming</td>
<td>—</td>
<td>—</td>
<td>10</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>D112</td>
<td>VBE</td>
<td>VDD for Bulk Erase</td>
<td>2.7</td>
<td>—</td>
<td>VDDMAX</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D113</td>
<td>VPEW</td>
<td>VDD for Write or Row Erase</td>
<td>VDDMIN</td>
<td>—</td>
<td>VDDMAX</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D114</td>
<td>IPPGM</td>
<td>Current on MCLR/VPP during Erase/Write</td>
<td>—</td>
<td>—</td>
<td>1.0</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>D115</td>
<td>IDDPGM</td>
<td>Current on VDD during Erase/Write</td>
<td>—</td>
<td>—</td>
<td>5.0</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>D121</td>
<td>EP</td>
<td>Program Flash Memory Cell Endurance</td>
<td>10K</td>
<td>—</td>
<td>—</td>
<td>E/W</td>
<td>-40°C to +85°C (Note 1)</td>
</tr>
<tr>
<td>D122</td>
<td>VPRW</td>
<td>VDD for Read/Write</td>
<td>VDDMIN</td>
<td>—</td>
<td>VDDMAX</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D123</td>
<td>TIW</td>
<td>Self-timed Write Cycle Time</td>
<td>—</td>
<td>2</td>
<td>2.5</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>D124</td>
<td>TRET</td>
<td>Characteristic Retention</td>
<td>—</td>
<td>40</td>
<td>—</td>
<td>Year</td>
<td>Provided no other specifications are violated</td>
</tr>
<tr>
<td>D125</td>
<td>EHEFC</td>
<td>High-Endurance Flash Cell</td>
<td>100K</td>
<td>—</td>
<td>—</td>
<td>E/W</td>
<td>0°C to +60°C, Lower byte, Last 128 Addresses in Flash Memory</td>
</tr>
</tbody>
</table>

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Self-write and Block Erase.

**Note 2:** Required only if single-supply programming is disabled.
21.6 Thermal Considerations

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Typ.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH01</td>
<td>( \theta_{JA} )</td>
<td>Thermal Resistance Junction to Ambient</td>
<td>89.3</td>
<td>°C/W</td>
<td>8-pin PDIP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>149.5</td>
<td>°C/W</td>
<td>8-pin SOIC package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>211</td>
<td>°C/W</td>
<td>8-pin MSOP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>60</td>
<td>°C/W</td>
<td>8-pin UDFN 2X3mm package</td>
</tr>
<tr>
<td>TH02</td>
<td>( \theta_{JC} )</td>
<td>Thermal Resistance Junction to Case</td>
<td>43.1</td>
<td>°C/W</td>
<td>8-pin PDIP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>39.9</td>
<td>°C/W</td>
<td>8-pin SOIC package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>39</td>
<td>°C/W</td>
<td>8-pin MSOP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>°C/W</td>
<td>8-pin UDFN 2X3mm package</td>
</tr>
<tr>
<td>TH03</td>
<td>( T_{JMAX} )</td>
<td>Maximum Junction Temperature</td>
<td>150</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>TH04</td>
<td>PD</td>
<td>Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>( PD = P_{INTERNAL} + P_{I/O} )</td>
</tr>
<tr>
<td>TH05</td>
<td>( P_{INTERNAL} )</td>
<td>Internal Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>( P_{INTERNAL} = I_{DD} \times V_{DD} )</td>
</tr>
<tr>
<td>TH06</td>
<td>( P_{I/O} )</td>
<td>I/O Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>( P_{I/O} = \sum (I_{OL} \times V_{OL}) + \sum (I_{OH} \times (V_{DD} - V_{OH})) )</td>
</tr>
<tr>
<td>TH07</td>
<td>( P_{DER} )</td>
<td>Derated Power</td>
<td>—</td>
<td>W</td>
<td>( P_{DER} = P_{DMAX} \times \frac{(T_{J} - T_{A})}{\theta_{JA}} )</td>
</tr>
</tbody>
</table>

**Note 1:** \( I_{DD} \) is current to run the chip alone without driving any load on the output pins.

**Note 2:** \( T_{A} = \) Ambient Temperature; \( T_{J} = \) Junction Temperature.
21.7 Timing Parameter Symbology

The timing parameter symbols have been created with one of the following formats:

1. TppS2ppS
2. TppS

<table>
<thead>
<tr>
<th>T</th>
<th>Frequency</th>
<th>T</th>
<th>Time</th>
</tr>
</thead>
</table>

Lowercase letters (pp) and their meanings:

<table>
<thead>
<tr>
<th>pp</th>
<th></th>
<th>pp</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cc</td>
<td>CCP1</td>
<td>osc</td>
<td>CLKIN</td>
</tr>
<tr>
<td>ck</td>
<td>CLKOUT</td>
<td>rd</td>
<td>RD</td>
</tr>
<tr>
<td>cs</td>
<td>CS</td>
<td>rw</td>
<td>RD or WR</td>
</tr>
<tr>
<td>di</td>
<td>SDIx</td>
<td>sc</td>
<td>SCKx</td>
</tr>
<tr>
<td>do</td>
<td>SDO</td>
<td>ss</td>
<td>SS</td>
</tr>
<tr>
<td>dt</td>
<td>Data in</td>
<td>t0</td>
<td>T0CKI</td>
</tr>
<tr>
<td>io</td>
<td>I/O PORT</td>
<td>t1</td>
<td>T1CKI</td>
</tr>
<tr>
<td>mc</td>
<td>MCLR</td>
<td>wr</td>
<td>WR</td>
</tr>
</tbody>
</table>

Uppercase letters and their meanings:

<table>
<thead>
<tr>
<th>S</th>
<th></th>
<th>S</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Fall</td>
<td>P</td>
<td>Period</td>
</tr>
<tr>
<td>H</td>
<td>High</td>
<td>R</td>
<td>Rise</td>
</tr>
<tr>
<td>I</td>
<td>Invalid (High-impedance)</td>
<td>V</td>
<td>Valid</td>
</tr>
<tr>
<td>L</td>
<td>Low</td>
<td>Z</td>
<td>High-impedance</td>
</tr>
</tbody>
</table>

FIGURE 21-3: LOAD CONDITIONS

Legend: CL = 50 pF for all pins
21.8 AC Characteristics: PIC12LF1552-I/E

FIGURE 21-4: CLOCK TIMING

CLKIN

CLKOUT (CLKOUT Mode)

OS02

OS12

Q4 Q1 Q2 Q3 Q4 Q1

OS03

OS11

Note: See Table 21-3.
### TABLE 21-1: CLOCK OSCILLATOR TIMING REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)
Operating temperature -40°C ≤ TA ≤ +125°C

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS01</td>
<td>FOSC</td>
<td>External CLKIN Frequency(1)</td>
<td>DC</td>
<td>—</td>
<td>0.5</td>
<td>MHz</td>
<td>EC Oscillator mode (low)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DC</td>
<td>—</td>
<td>4</td>
<td>MHz</td>
<td>EC Oscillator mode (medium)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DC</td>
<td>—</td>
<td>20</td>
<td>MHz</td>
<td>EC Oscillator mode (high)</td>
</tr>
<tr>
<td>OS02</td>
<td>TOSC</td>
<td>External CLKIN Period(1)</td>
<td>50</td>
<td>—</td>
<td>∞</td>
<td>ns</td>
<td>EC mode</td>
</tr>
<tr>
<td>OS03</td>
<td>Tcy</td>
<td>Instruction Cycle Time(1)</td>
<td>200</td>
<td>—</td>
<td>DC</td>
<td>ns</td>
<td>TCY = FOSC/4</td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.
† Data in “Typ.” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Instruction cycle period (TCy) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min” values with an external clock applied to CLKIN pin. When an external clock input is used, the “max” cycle time limit is “DC” (no clock) for all devices.

### TABLE 21-2: OSCILLATOR PARAMETERS

Standard Operating Conditions (unless otherwise stated)
Operating Temperature -40°C ≤ TA ≤ +125°C

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS08</td>
<td>HFOSC</td>
<td>Internal Calibrated HFINTOSC Frequency(1)</td>
<td>—</td>
<td>16.0</td>
<td>—</td>
<td>MHz</td>
<td>0°C ≤ TA ≤ +85°C</td>
</tr>
<tr>
<td>OS08A</td>
<td>HFTOL</td>
<td>Frequency Tolerance</td>
<td>—</td>
<td>±3</td>
<td>—</td>
<td>%</td>
<td>25°C, 16MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>±6</td>
<td>—</td>
<td>%</td>
<td>0°C ≤ TA ≤ +85°C, 16 MHz</td>
</tr>
<tr>
<td>OS09</td>
<td>LFOSC</td>
<td>Internal LFINTOSC Frequency</td>
<td>—</td>
<td>31</td>
<td>—</td>
<td>kHz</td>
<td>-40°C ≤ TA ≤ +125°C</td>
</tr>
<tr>
<td>OS10*</td>
<td>TWARM</td>
<td>Wake-up from Sleep Start-up Time LFINTOSC</td>
<td>—</td>
<td>5</td>
<td>8</td>
<td>μs</td>
<td></td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.
† Data in “Typ.” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** To ensure these oscillator frequency tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.
TABLE 21-3: CLKOUT AND I/O TIMING PARAMETERS

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS11</td>
<td>TosH2ckL</td>
<td>Fosc↑ to CLKOUT↓ (1)</td>
<td>—</td>
<td>—</td>
<td>70 ns</td>
<td>VDD = 3.3-3.6V</td>
<td></td>
</tr>
<tr>
<td>OS12</td>
<td>TosH2ckH</td>
<td>Fosc↑ to CLKOUT↑ (1)</td>
<td>—</td>
<td>—</td>
<td>72 ns</td>
<td>VDD = 3.3-3.6V</td>
<td></td>
</tr>
<tr>
<td>OS13</td>
<td>TckL2ioV</td>
<td>CLKOUT↓ to Port out valid (1)</td>
<td>—</td>
<td>—</td>
<td>20 ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS14</td>
<td>TioV2ckH</td>
<td>Port input valid before CLKOUT↑ (1)</td>
<td>Tosc + 200 ns</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>OS15</td>
<td>TosH2ioV</td>
<td>Fosc↑ (Q1 cycle) to Port out valid</td>
<td>—</td>
<td>50</td>
<td>70* ns</td>
<td>VDD = 3.3-3.6V</td>
<td></td>
</tr>
<tr>
<td>OS16</td>
<td>TosH2iol</td>
<td>Fosc↑ (Q2 cycle) to Port input invalid (I/O in hold time)</td>
<td>50</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>OS17</td>
<td>TioV2osH</td>
<td>Port input valid to Fosc↑ (Q2 cycle) (I/O in setup time)</td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>OS18*</td>
<td>TioR</td>
<td>Port output rise time (2)</td>
<td>—</td>
<td>15</td>
<td>32 ns</td>
<td>VDD = 2.0V</td>
<td></td>
</tr>
<tr>
<td>OS19*</td>
<td>TioF</td>
<td>Port output fall time (2)</td>
<td>—</td>
<td>28</td>
<td>55 ns</td>
<td>VDD = 2.0V</td>
<td></td>
</tr>
<tr>
<td>OS20*</td>
<td>Tinp</td>
<td>INT pin input high or low time</td>
<td>25</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>OS21*</td>
<td>Tioc</td>
<td>Interrupt-on-change new input level time</td>
<td>25</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.
† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated.

**Note 1:** Measurements are taken in EC mode where CLKOUT output is 4 x Tosc.
FIGURE 21-6: RESET, WATCHDOG TIMER AND POWER-UP TIMER TIMING

Note 1: Asserted low.

FIGURE 21-7: BROWN-OUT RESET TIMING AND CHARACTERISTICS

Note 1: 64 ms delay only if PWRT bit in the Configuration Words is programmed to ‘0’. 2 ms delay if PWRT = 0.
### TABLE 21-4: RESET, WATCHDOG TIMER, POWER-UP TIMER AND BROWN-OUT RESET PARAMETERS

**Standard Operating Conditions (unless otherwise stated)**

Operating Temperature $-40^\circ C \leq T_A \leq +125^\circ C$

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>TMCL</td>
<td>MCLR Pulse Width (low)</td>
<td>2</td>
<td>—</td>
<td>—</td>
<td>µs</td>
<td>$-40^\circ C$ to $+85^\circ C$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>—</td>
<td>—</td>
<td>µs</td>
<td>$+85^\circ C$ to $+125^\circ C$</td>
</tr>
<tr>
<td>31</td>
<td>TWDTLP</td>
<td>Low-Power Watchdog Timer Time-out Period</td>
<td>10</td>
<td>16</td>
<td>27</td>
<td>ms</td>
<td>$V_{DD} = 3.3V$-$3.6V$, 1:512 Prescaler used</td>
</tr>
<tr>
<td>33*</td>
<td>TPWRT</td>
<td>Power-up Timer Period, PWRTE = 0</td>
<td>40</td>
<td>65</td>
<td>140</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>34*</td>
<td>TOZ</td>
<td>I/O high-impedance from MCLR Low or Watchdog Timer Reset</td>
<td>—</td>
<td>—</td>
<td>2.0</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>VBOR</td>
<td>Brown-out Reset Voltage(1)</td>
<td>2.55</td>
<td>1.80</td>
<td>1.90</td>
<td>2.05</td>
<td>$V$ BORV = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.70</td>
<td>2.85</td>
<td>2.05</td>
<td>$V$ BORV = 1</td>
<td></td>
</tr>
<tr>
<td>36*</td>
<td>VHYST</td>
<td>Brown-out Reset Hysteresis</td>
<td>0</td>
<td>25</td>
<td>50</td>
<td>mV</td>
<td>$-40^\circ C$ to $+85^\circ C$</td>
</tr>
<tr>
<td>37*</td>
<td>TBORDC</td>
<td>Brown-out Reset DC Response Time</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>µs</td>
<td>$V_{DD} \leq V_{BOR}$</td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.
† Data in “Typ.” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** To ensure these voltage tolerances, $V_{DD}$ 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.

### FIGURE 21-8: TIMER0 EXTERNAL CLOCK TIMINGS

[Diagram of TIM0CKI, TIM1CKI, and TMR0 clock timings]
## Table 21-5: Timer0 External Clock Requirements

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ. †</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>40*</td>
<td>T0H</td>
<td>T0CKI High Pulse Width</td>
<td>No Prescaler</td>
<td>0.5 Tcy + 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Prescaler</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>41*</td>
<td>T0L</td>
<td>T0CKI Low Pulse Width</td>
<td>No Prescaler</td>
<td>0.5 Tcy + 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With Prescaler</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>42*</td>
<td>T0P</td>
<td>T0CKI Period</td>
<td>Greater of:</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>N = prescale value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20 or Tcy + 40</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>(2, 4, ..., 256)</td>
</tr>
<tr>
<td>45*</td>
<td>T1H</td>
<td>T1CKI High Time</td>
<td>Synchronous, No Prescaler</td>
<td>0.5 Tcy + 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Synchronous, with Prescaler</td>
<td>15</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Asynchronous</td>
<td>30</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.
† Data in “Typ.” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

## Table 21-6: PIC12LF1552 A/D Converter (ADC) Characteristics

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ. †</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD01</td>
<td>NR</td>
<td>Resolution</td>
<td>—</td>
<td>—</td>
<td>10</td>
<td>bit</td>
<td></td>
</tr>
<tr>
<td>AD02</td>
<td>EIL</td>
<td>Integral Error</td>
<td>—</td>
<td>±0.4</td>
<td>±1</td>
<td>LSb</td>
<td>-40°C to +85°C, VREF ≥ 2.0V</td>
</tr>
<tr>
<td>AD03</td>
<td>EDL</td>
<td>Differential Error</td>
<td>—</td>
<td>±0.3</td>
<td>±1</td>
<td>LSb</td>
<td>-40°C to +85°C, VREF ≥ 2.0V</td>
</tr>
<tr>
<td>AD04</td>
<td>EOFF</td>
<td>Offset Error</td>
<td>—</td>
<td>1.2</td>
<td>±3</td>
<td>LSb</td>
<td>-40°C to +85°C, VREF ≥ 2.0V</td>
</tr>
<tr>
<td>AD05</td>
<td>EGN</td>
<td>Gain Error</td>
<td>—</td>
<td>1.0</td>
<td>±3</td>
<td>LSb</td>
<td>-40°C to +85°C, VREF ≥ 2.0V</td>
</tr>
<tr>
<td>AD06</td>
<td>VREF</td>
<td>Reference Voltage Range (VREFH – VREFL)</td>
<td>1.8</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>Absolute Minimum (Note 4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2.0</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>Minimum for 1LSb Accuracy</td>
</tr>
<tr>
<td>AD07</td>
<td>VAIN</td>
<td>Full-Range</td>
<td>VSS</td>
<td>—</td>
<td>VREF</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>AD08</td>
<td>ZAIN</td>
<td>Recommended Impedance of Analog Voltage Source</td>
<td>—</td>
<td>—</td>
<td>3</td>
<td>kΩ</td>
<td>Can go higher if external 0.01μF capacitor is present on input pin</td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.
† Data in “Typ.” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Total Absolute Error includes integral, differential, offset and gain errors.
2: The ADC conversion result never decreases with an increase in the input voltage and has no missing codes.
3: When ADC is off, it will not consume any current other than leakage current. The power-down current specification includes any such leakage from the ADC module.
4: ADC VREF is selected by ADPREF<1:0> bits.
### TABLE 21-7: PIC12LF1552 ADC CONVERSION REQUIREMENTS

**Standard Operating Conditions (unless otherwise stated)**

Operating temperature -40°C ≤ TA ≤ +125°C

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ.†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD130*</td>
<td>Tad</td>
<td>ADC Clock Period</td>
<td>0.25</td>
<td>25</td>
<td>0.7</td>
<td>µs</td>
<td>Tosc-based, -40°C to +85°C, VREF ≥ 2.4V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ADC Internal FRC Oscillator Period</td>
<td>1.0</td>
<td>1.6</td>
<td>6.0</td>
<td>µs</td>
<td>Tosc-based, +86°C to +125°C, VREF ≥ 2.4V</td>
</tr>
<tr>
<td>AD131</td>
<td>Tcnv</td>
<td>Conversion Time (not including Acquisition Time)(^{(1)})</td>
<td>—</td>
<td>11</td>
<td>—</td>
<td>—</td>
<td>Set GO/DONE bit to conversion complete</td>
</tr>
<tr>
<td>AD132*</td>
<td>Tacq</td>
<td>Acquisition Time</td>
<td>—</td>
<td>5.0</td>
<td>—</td>
<td>µs</td>
<td>—</td>
</tr>
</tbody>
</table>

\(^*\) These parameters are characterized but not tested.

\(^†\) Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** The ADRES register may be read on the following Tcy cycle.

---

### FIGURE 21-9: PIC12LF1552 ADC CONVERSION TIMING (NORMAL MODE)

![Diagram showing ADC conversion timing](image)

**Note 1:** If the ADC clock source is selected as FRC, a time of Tcy is added before the ADC clock starts. This allows the SLEEP instruction to be executed.
FIGURE 21-10:  PIC12LF1552 ADC CONVERSION TIMING (SLEEP MODE)

Note 1: If the ADC clock source is selected as FRC, a time of TCY is added before the ADC clock starts. This allows the SLEEP instruction to be executed.
22.0 DC AND AC CHARACTERISTICS GRAPHS AND CHARTS

The graphs and tables provided in this section are for design guidance and are not tested. In some graphs or tables, the data presented are outside specified operating range (i.e., outside specified V\text{DD} range). This is for information only and devices are ensured to operate properly only within the specified range.

**Note:** The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

“Typical” represents the mean of the distribution at 25°C. “MAXIMUM”, “Max.”, “MINIMUM” or “Min.” represents (mean + 3\(\sigma\)) or (mean - 3\(\sigma\)) respectively, where \(\sigma\) is a standard deviation, over each temperature range.
Note: Unless otherwise noted Cin = 0.1 µF and TA = 25°C.

**FIGURE 22-1:** IDD, EC Oscillator, Low-Power Mode, Fosc = 32 kHz

**FIGURE 22-2:** IDD, Typical, EC Oscillator, Low-Power Mode, Fosc = 500 kHz

**FIGURE 22-3:** IDD, Typical, EC Oscillator, High-Power Mode

**FIGURE 22-4:** IDD, Maximum, EC Oscillator, Medium-Power Mode

**FIGURE 22-5:** IDD, Typical, EC Oscillator, High-Power Mode

**FIGURE 22-6:** IDD, Maximum, EC Oscillator, High-Power Mode
Note: Unless otherwise noted CIN = 0.1 µF and TA = 25°C.

**FIGURE 22-7:** $I_{DD}$, LFINTOSC, $F_{OSC} = 31$ kHz

**FIGURE 22-8:** $I_{DD}$, Typical, HFINTOSC

**FIGURE 22-9:** $I_{DD}$, Maximum, EC Oscillator, High-Power Mode

**FIGURE 22-10:** $I_{PD}$, Base, Low-Power Sleep Mode

**FIGURE 22-11:** $I_{PD}$, Watchdog Timer (WDT)

**FIGURE 22-12:** $I_{PD}$, Fixed Voltage Reference (FVR)
Note: Unless otherwise noted CIN = 0.1 µF and TA = 25°C.

**FIGURE 22-13:** $IP_D$, Brown-out Reset (BOR)

**FIGURE 22-14:** $VO_H$ vs. $IO_H$, Over Temperature, $V_{DD} = 5.5V$, PIC12FXXXX Only

**FIGURE 22-15:** POR Release Voltage

**FIGURE 22-16:** POR Rearm Voltage, PIC12FXXXX Only
23.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/
    MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for
    Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICkit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards,
  Evaluation Kits and Starter Kits
- Third-party development tools

23.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:
- Color syntax highlighting
- Smart code completion makes suggestions and
  provides hints as you type
- Automatic code formatting based on user-defined
  rules
- Live parsing

User-Friendly, Customizable Interface:
- Fully customizable interface: toolbars, toolbar
  buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:
- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:
- Local file history feature
- Built-in support for Bugzilla issue tracker
23.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip’s 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

23.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

23.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

23.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility
23.6 **MPLAB X SIM Software Simulator**

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

23.7 **MPLAB REAL ICE In-Circuit Emulator System**

The MPLAB REAL ICE In-Circuit Emulator System is Microchip’s next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer’s PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

23.8 **MPLAB ICD 3 In-Circuit Debugger System**

The MPLAB ICD 3 In-Circuit Debugger System is Microchip’s most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer’s PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

23.9 **PICkit 3 In-Circuit Debugger/Programmer**

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer’s PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

23.10 **MPLAB PM3 Device Programmer**

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at $V_{DDMIN}$ and $V_{DDMAX}$ for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.
23.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, Keeloq® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

23.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®
24.0 PACKAGING INFORMATION

24.1 Package Marking Information

Legend:
- **XX...X**: Customer-specific information
- **Y**: Year code (last digit of calendar year)
- **YY**: Year code (last 2 digits of calendar year)
- **WW**: Week code (week of January 1 is week '01')
- **NNN**: Alphanumeric traceability code
- **(E3)**: Pb-free JEDEC® designator for Matte Tin (Sn)
- *****: This package is Pb-free. The Pb-free JEDEC designator (E3) can be found on the outer packaging for this package.

**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.
Package Marking Information (Continued)

8-Lead MSOP (3x3 mm) Example

Legend:
- XX...X Customer-specific information
- Y Year code (last digit of calendar year)
- YY Year code (last 2 digits of calendar year)
- WW Week code (week of January 1 is week '01')
- NNN Alphanumeric traceability code
- Pb-free JEDEC® designator for Matte Tin (Sn)
- * This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.
24.2 Package Details
The following sections give the technical details of the packages.

8-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

Microchip Technology Drawing No. C04-018D Sheet 1 of 2
8-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**

<table>
<thead>
<tr>
<th>Dimension Limits</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Pins</td>
<td>N</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
<td>.100 BSC</td>
<td></td>
</tr>
<tr>
<td>Top to Seating Plane</td>
<td>A</td>
<td>-</td>
<td>.210</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
<td>.115</td>
<td>.130</td>
</tr>
<tr>
<td>Base to Seating Plane</td>
<td>A1</td>
<td>.015</td>
<td>-</td>
</tr>
<tr>
<td>Shoulder to Shoulder Width</td>
<td>E</td>
<td>.390</td>
<td>.310</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
<td>.240</td>
<td>.250</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
<td>.348</td>
<td>.365</td>
</tr>
<tr>
<td>Tip to Seating Plane</td>
<td>L</td>
<td>.115</td>
<td>.130</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
<td>.008</td>
<td>.010</td>
</tr>
<tr>
<td>Upper Lead Width</td>
<td>b1</td>
<td>.040</td>
<td>.060</td>
</tr>
<tr>
<td>Lower Lead Width</td>
<td>b</td>
<td>.014</td>
<td>.018</td>
</tr>
<tr>
<td>Overall Row Spacing</td>
<td>eB</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**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.
8-Lead Plastic Small Outline (SN) - 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

TOP VIEW

SIDE VIEW

VIEW A-A

Microchip Technology Drawing No. C04-057C Sheet 1 of 2
8-Lead Plastic Small Outline (SN) - 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

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
</tr>
<tr>
<td>Standoff § A1</td>
<td>0.10</td>
</tr>
<tr>
<td>Overall Width E</td>
<td>6.00 BSC</td>
</tr>
<tr>
<td>Molded Package Width E1</td>
<td>3.90 BSC</td>
</tr>
<tr>
<td>Overall Length D</td>
<td>4.90 BSC</td>
</tr>
<tr>
<td>Chamfer (Optional) h</td>
<td>0.25</td>
</tr>
<tr>
<td>Foot Length L</td>
<td>0.40</td>
</tr>
<tr>
<td>Footprint L1</td>
<td>1.04 REF</td>
</tr>
<tr>
<td>Foot Angle φ</td>
<td>0&quot;</td>
</tr>
<tr>
<td>Lead Thickness c</td>
<td>0.17</td>
</tr>
<tr>
<td>Lead Width b</td>
<td>0.31</td>
</tr>
<tr>
<td>Mold Draft Angle Top α</td>
<td>5&quot;</td>
</tr>
<tr>
<td>Mold Draft Angle Bottom β</td>
<td>5&quot;</td>
</tr>
</tbody>
</table>

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.
8-Lead Plastic Small Outline (SN) – 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

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C</td>
</tr>
<tr>
<td>Contact Pad Width (X8)</td>
<td>X1</td>
</tr>
<tr>
<td>Contact Pad Length (X8)</td>
<td>Y1</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2057A
8-Lead Plastic Micro Small Outline Package (MS) [MSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

Microchip Technology Drawing C04-111C Sheet 1 of 2
8-Lead Plastic Micro Small Outline Package (MS) [MSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

![Diagram of 8-Lead Plastic Micro Small Outline Package (MS) [MSOP]](image)

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td>Limits</td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>A1</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A2</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>0.85</td>
</tr>
<tr>
<td>Standoff</td>
<td>E1</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E1</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>0.60</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Foot Length</td>
<td>L</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1</td>
</tr>
<tr>
<td>Foot Angle</td>
<td>Φ</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>Lead Width</td>
<td>b</td>
</tr>
</tbody>
</table>

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 0.15mm 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.
8-Lead Plastic Micro Small Outline Package (MS) [MSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

---

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C</td>
</tr>
<tr>
<td>Overall Width</td>
<td>Z</td>
</tr>
<tr>
<td>Contact Pad Width (X8)</td>
<td>X1</td>
</tr>
<tr>
<td>Contact Pad Length (X8)</td>
<td>Y1</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G1</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>GX</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2111A
8-Lead Plastic Dual Flat, No Lead Package (MU) – 2x3x0.5 mm Body [UDFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

![Diagram of 8-Lead Plastic Dual Flat, No Lead Package (MU)]

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
</tr>
<tr>
<td>Contact Thickness</td>
<td>A3</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>Exposed Pad Length</td>
<td>D2</td>
</tr>
<tr>
<td>Exposed Pad Width</td>
<td>E2</td>
</tr>
<tr>
<td>Contact Width</td>
<td>b</td>
</tr>
<tr>
<td>Contact Length</td>
<td>L</td>
</tr>
<tr>
<td>Contact-to-Exposed Pad</td>
<td>K</td>
</tr>
</tbody>
</table>

**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. C04-136B
8-Lead Plastic Dual Flat, No Lead Package (MU) – 2x3x0.5 mm Body [UDFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com-packaging

---

**DIMENSIONS (IN MILLIMETERS)**

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact Pitch</td>
<td>E</td>
</tr>
<tr>
<td>Optional Center Pad Width</td>
<td>W2</td>
</tr>
<tr>
<td>Optional Center Pad Length</td>
<td>T2</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1</td>
</tr>
<tr>
<td>Contact Pad Width (X8)</td>
<td>X1</td>
</tr>
<tr>
<td>Contact Pad Length (X8)</td>
<td>Y1</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>G</td>
</tr>
</tbody>
</table>

**Notes:**

1. Dimensioning and tolerancing per ASME Y14.5M
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2136A
APPENDIX A: DATA SHEET

REVISION HISTORY

Revision A (12/2012)
Original release.

Revision B (01/2013)
Revised Product ID System – Corrected packaged code MF to MU.

Revision C (12/2013)
Updated Sections 16, CVD and 21, Electrical Specifications; Other minor corrections.

Revision D (04/2014)
Updated Peripheral Features on page 1; Updated Figure 1-1 and Equation 15-1; Updated Sections 18.3, 18.3.2, 18.6.7.4, 21.2 and 21.3; Updated titles in Figure 21-6 and Table 21-4; Updated Note 1 in Figure 21-7 and Note 4 in Table 21-6; Added DC and AC Characteristics Graphs and Charts; Updated Chapter 24, Packaging Information; Other minor corrections.

Revision E (01/2015)
Updated Tables 21-6 and 21-7; Other minor corrections.

Revision F (10/2016)
Updated eXtreme Low-Power (XLP) Features; Updated Example 3-2; Updated Table 5-1; Added section 5.3.2; Updated Table 21-2. Updated section 15.2.6; Updated Example 15-1; Updated Register 6-2; Updated Table 18-4; Updated section 21.3; Updated Table 21-4. Added new char graphs. Other minor corrections.
THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user’s guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip’s customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under “Support”, click on “Customer Change Notification” and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at: http://www.microchip.com/support
# PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

## PART NO. [X][1] - X/XX/XXX

<table>
<thead>
<tr>
<th>Device</th>
<th>Tape and Reel Option</th>
<th>Temperature Range</th>
<th>Package</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC12LF1552</td>
<td>Blank = Standard packaging (tube or tray)</td>
<td>I = -40°C to +85°C (Industrial)</td>
<td>MU = Micro Lead Frame (UDFN) 2x3</td>
<td>QTP, SQTP, Code or Special Requirements (blank otherwise)</td>
</tr>
<tr>
<td></td>
<td>T = Tape and Reel(1)</td>
<td>E = -40°C to +125°C (Extended)</td>
<td>MS = MSOP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P = Plastic DIP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SN = SOIC</td>
<td></td>
</tr>
</tbody>
</table>

### Examples:

a) PIC12LF1552T - I/SN Tape and Reel, Industrial temperature, SOIC package
b) PIC12LF1552 - I/P Industrial temperature PDIP package
c) PIC12LF1552 - E/MU Extended temperature, UDFN package

### Note 1:
- Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.

### Note 2:
- Small form-factor packaging options may be available. Please check [www.microchip.com/packaging](http://www.microchip.com/packaging) for small-form-factor package availability, or contact your local Sales Office.
Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.

- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.

- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.

- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.
## Worldwide Sales and Service

### AMERICAS

- **Corporate Office**
  - 2355 West Chandler Blvd.
  - Chandler, AZ 85224-6199
  - Tel: 480-792-7200
  - Fax: 480-792-7277
  - Technical Support: [http://www.microchip.com/support](http://www.microchip.com/support)
  - Web Address: [www.microchip.com](http://www.microchip.com)

- **Atlanta**
  - Duluth, GA
  - Tel: 678-957-9614
  - Fax: 678-957-1455

- **Austin, TX**
  - Tel: 512-257-3370

- **Boston**
  - Westborough, MA
  - Tel: 774-760-0087
  - Fax: 774-760-0088

- **Chicago**
  - Itasca, IL
  - Tel: 630-285-0071
  - Fax: 630-285-0075

- **Cleveland**
  - Independence, OH
  - Tel: 216-447-0464
  - Fax: 216-447-0643

- **Dallas**
  - Addison, TX
  - Tel: 972-818-7423
  - Fax: 972-818-2924

- **Detroit**
  - Novi, MI
  - Tel: 248-848-4000

- **Houston, TX**
  - Tel: 281-894-5983

- **Indianapolis**
  - Noblesville, IN
  - Tel: 317-773-8323
  - Fax: 317-773-5453

- **Los Angeles**
  - Mission Viejo, CA
  - Tel: 949-462-9523
  - Fax: 949-462-9608

- **New York, NY**
  - Tel: 631-435-6000

- **San Jose, CA**
  - Tel: 408-735-9110

- **Canada - Toronto**
  - Tel: 905-673-0699
  - Fax: 905-673-6509

### ASIA/PACIFIC

- **Asia Pacific Office**
  - Suites 3707-14, 37th Floor
  - Tower 6, The Gateway
  - Harbour City, Kowloon
  - Hong Kong
  - Tel: 852-2943-5100
  - Fax: 852-2401-3431

- **Australia - Sydney**
  - Tel: 61-2-9868-6733
  - Fax: 61-2-9868-6755

- **China - Beijing**
  - Tel: 86-10-8569-7000
  - Fax: 86-10-8528-2104

- **China - Chengdu**
  - Tel: 86-28-8665-5511
  - Fax: 86-28-8665-7889

- **China - Chongqing**
  - Tel: 86-23-8960-9588
  - Fax: 86-23-8980-9500

- **China - Dongguan**
  - Tel: 86-769-8702-9880

- **China - Hangzhou**
  - Tel: 86-571-8792-8115
  - Fax: 86-571-8792-8116

- **China - Hong Kong SAR**
  - Tel: 852-2943-5100
  - Fax: 852-2401-3431

- **China - Nanjing**
  - Tel: 86-25-8473-2460
  - Fax: 86-25-8473-2470

- **China - Qingdao**
  - Tel: 86-532-8502-7355
  - Fax: 86-532-8502-7205

- **China - Shanghai**
  - Tel: 86-21-5407-5533
  - Fax: 86-21-5407-5066

- **China - Shenyang**
  - Tel: 86-24-2334-2829
  - Fax: 86-24-2334-2393

- **China - Shenzhen**
  - Tel: 86-755-8864-2200
  - Fax: 86-755-8203-1760

- **China - Wuhan**
  - Tel: 86-27-5980-5300
  - Fax: 86-27-5980-5118

- **China - Xian**
  - Tel: 86-29-8833-7252
  - Fax: 86-29-8833-7256

### ASIA/PACIFIC

- **China - Xiamen**
  - Tel: 86-592-2388138
  - Fax: 86-592-2388130

- **China - Zhuhai**
  - Tel: 86-756-3210040
  - Fax: 86-756-3210049

- **India - Bangalore**
  - Tel: 91-80-3090-4444
  - Fax: 91-80-3090-4313

- **India - New Delhi**
  - Tel: 91-11-4160-8631
  - Fax: 91-11-4160-8632

- **India - Pune**
  - Tel: 91-20-3019-1500

- **Japan - Osaka**
  - Tel: 81-6-6152-7160
  - Fax: 81-6-6152-9310

- **Japan - Tokyo**
  - Tel: 81-3-6880-3770
  - Fax: 81-3-6880-3771

- **Korea - Daegu**
  - Tel: 82-53-744-4301
  - Fax: 82-53-744-4302

- **Korea - Seoul**
  - Tel: 82-2-554-7200
  - Fax: 82-2-558-5932 or 82-2-558-5934

- **Malaysia - Kuala Lumpur**
  - Tel: 60-3-6201-9857
  - Fax: 60-3-6201-9859

- **Malaysia - Penang**
  - Tel: 60-4-227-8870
  - Fax: 60-4-227-4068

- **Philippines - Manila**
  - Tel: 63-2-634-9065
  - Fax: 63-2-634-9069

- **Singapore**
  - Tel: 65-6334-8870
  - Fax: 65-6334-8850

- **Taiwan - Hsin Chu**
  - Tel: 886-3-5778-366
  - Fax: 886-3-5770-955

- **Taiwan - Kaohsiung**
  - Tel: 886-7-213-7828

- **Taiwan - Taipei**
  - Tel: 886-2-2508-8600
  - Fax: 886-2-2508-0102

- **Thailand - Bangkok**
  - Tel: 66-2-694-1351
  - Fax: 66-2-694-1350

### EUROPE

- **Austria - Wels**
  - Tel: 43-7242-2244-39
  - Fax: 43-7242-2244-393

- **Denmark - Copenhagen**
  - Tel: 45-4450-2828
  - Fax: 45-4485-2829

- **France - Paris**
  - Tel: 33-1-69-53-63-20
  - Fax: 33-1-69-30-90-79

- **Germany - Dusseldorf**
  - Tel: 49-2129-3766400

- **Germany - Karlsruhe**
  - Tel: 49-721-625370

- **Germany - Munich**
  - Tel: 49-89-627-144-0
  - Fax: 49-89-627-144-44

- **Italy - Milan**
  - Tel: 39-0331-742611
  - Fax: 39-0331-466781

- **Italy - Venice**
  - Tel: 39-049-7625286

- **Netherlands - Drunen**
  - Tel: 31-416-690399
  - Fax: 31-416-690340

- **Poland - Warsaw**
  - Tel: 48-22-3325737

- **Spain - Madrid**
  - Tel: 34-91-708-08-90
  - Fax: 34-91-708-08-91

- **Sweden - Stockholm**
  - Tel: 46-8-5090-4654

- **UK - Wokingham**
  - Tel: 44-118-921-5800
  - Fax: 44-118-921-5820

© 2012-2016 Microchip Technology Inc. DS40001674F-page 243