PIC16F/LF1826/27
Data Sheet
18/20/28-Pin Flash Microcontrollers
with nanoWatt XLP Technology
Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

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

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

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, Hi-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MIWI, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Octopus, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, PIC18F logo, REAL ICE, rPICLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

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

© 2009, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.
High-Performance RISC CPU:
• C Compiler Optimized Architecture
• 256 bytes Data EEPROM
• Up to 4 Kbytes Linear Program Memory Addressing
• Up to 384 bytes Linear Data Memory Addressing
• 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:
• Precision 32 MHz Internal Oscillator Block:
  - Factory calibrated to ± 1%, typical
  - Software selectable frequencies range of 31 kHz to 32 MHz
• 31 kHz Low-Power Internal Oscillator
• Four Crystal modes up to 32 MHz
• Three External Clock modes up to 32 MHz
• 4X Phase Lock Loop (PLL)
• Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops
• Two-Speed Oscillator Start-up
• Reference Clock Module:
  - Programmable clock output frequency and duty-cycle

Special Microcontroller Features:
• Full 5.5V Operation – PIC16F1826/27
• 1.8V-3.6V Operation – PIC16LF1826/27
• Self-reprogrammable under Software Control
• Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
• Programmable Brown-out Reset (BOR)
• Extended Watchdog Timer (WDT):
  - Programmable period from 1ms to 268s
• Programmable Code Protection
• In-Circuit Serial Programming™ (ICSP™) via two pins
• In-Circuit Debug (ICD) via two pins
• Enhance Low-Voltage Programming
• Power-Saving Sleep mode

Extrem Low-Power Management
PIC16LF1826/27 with nanoWatt XLP:
• Sleep mode: 30 nA
• Watchdog Timer: 500 nA
• Timer1 Oscillator: 600 nA @ 32 kHz

Analog Features:
• Analog-to-Digital Converter (ADC) Module:
  - 10-bit resolution, 12 channels
  - Auto acquisition capability
  - Conversion available during Sleep
• Analog Comparator Module:
  - Two rail-to-rail analog comparators
  - Power mode control
  - Software controllable hysteresis
• Voltage Reference Module:
  - Fixed Voltage Reference (FVR) with 1.024V, 2.048V and 4.096V output levels
  - 5-bit rail-to-rail resistive DAC with positive and negative reference selection

Peripheral Highlights:
• 15 I/O Pins and 1 Input Only Pin:
  - High current sink/source 25 mA/25 mA
  - Programmable weak pull-ups
  - Programmable interrupt-on-change pins
• Timer0: 8-Bit Timer/Counter with 8-Bit Prescaler
• Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Gate Input mode
  - Dedicated, low-power 32 kHz oscillator driver
• Up to three Timer2-types: 8-Bit Timer/Counter with 8-Bit Period Register, Prescaler and Postscaler
• Up to two Capture, Compare, PWM (CCP) Modules
• Up to two Enhanced CCP (ECCP) Modules:
  - Software selectable time bases
  - Auto-shutdown and auto-restart
  - PWM steering
• Up to two Master Synchronous Serial Port (MSSP) with SPI and I2C™ with:
  - 7-bit address masking
  - SMBus/PMBus™ compatibility
• Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) Module
• mTouch™ Sensing Oscillator Module:
  - Up to 12 input channels
• Data Signal Modulator Module:
  - Selectable modulator and carrier sources
• SR Latch:
  - Multiple Set/Reset input options
  - Emulates 555 Timer applications
## PIC16F/LF1826/27 Family Types

<table>
<thead>
<tr>
<th>Device</th>
<th>Program Memory</th>
<th>Data Memory</th>
<th>I/O’s (1)</th>
<th>10-bit ADC (ch)</th>
<th>CapSense (ch)</th>
<th>Comparators</th>
<th>Timers (8/16-bit)</th>
<th>EUSART</th>
<th>MSSP</th>
<th>ECCP (Full-Bridge)</th>
<th>ECCP (Half-Bridge)</th>
<th>CCP</th>
<th>SR Latch</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC16LF1826</td>
<td>2K</td>
<td>256</td>
<td>256</td>
<td>16</td>
<td>12</td>
<td>12</td>
<td>2/1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>Yes</td>
</tr>
<tr>
<td>PIC16F1826</td>
<td>2K</td>
<td>256</td>
<td>256</td>
<td>16</td>
<td>12</td>
<td>12</td>
<td>2/1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>—</td>
<td>—</td>
<td>Yes</td>
</tr>
<tr>
<td>PIC16LF1827</td>
<td>4K</td>
<td>384</td>
<td>256</td>
<td>16</td>
<td>12</td>
<td>12</td>
<td>4/1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>PIC16F1827</td>
<td>4K</td>
<td>384</td>
<td>256</td>
<td>16</td>
<td>12</td>
<td>12</td>
<td>4/1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Note 1:** One pin is input only.
Pin Diagram – 18-Pin PDIP, SOIC (PIC16F/LF1826/27)

- RA0/AN0/CPS0/C12IN0-/SDO2(2)
- RA1/AN1/CPS1/C12IN1-/SS2
- RA2/AN2/CPS2/C12IN2-/C12IN+/VREF-/DACOUT
- RA3/AN3/CPS3/C12IN3-/C1IN+/VREF+/C1OUT/CCP3(2)/SRQ
- RA4/AN4/CPS4/C2OUT/T0CKI/CCP4(2)/SRNQ
- RA5/MCLR/VPP/SS1 (1,2)
- RA6/OSC2/CLKOUT/CLKR/P1D(1)/P2B(1,2)/SDO1(1)
- RA7/OSC1/CLKIN/P1C(1)/CCP2(1,2)/P2A(1,2)
- RA8/OSC3/CLKOUT/ACLK/P1D(1)/P2B(1,2)
- RA9/AN5/CPS5/T1CKI/T1OSI/P1C(1,3)/CCP2(1,2)/P2A(1,2)/ICSPCLK
- RB0/SRI/T1G/CCP1(1)/P1A(1)/INT/SRI/FLT0
- RB1/AN11/CPS11/RX(1)/DT(1)/SDA1/SDI1
- RB2/AN10/CPS10/MDMIN/TX(1)/CK(1,3)/RX(1)/DT(1)/SDA2(2)/SDI2(2)/SDO1(1,3)
- RB3/AN9/CPS9/MDOUT/CCP1(1)
- RB4/AN8/CPS8/SCL1/SCK1/MDCIN2
- RB5/AN7/CPS7/P1B/TX(1)/CK(1)/SCL2(2)/SCK2(2)/SS1 (1)
- RB6/AN5/CPS5/T1OSO/P1D(1)/P2B(1,2)/MDCIN1/ICSPDAT
- RB7/AN6/CPS6/T1OSI/T1OSO/P1D(1)/P2B(1,2)/ICSPCLK
- RB8/AN8/CPS8/T1OSI/T1OSO/P1D(1)/P2B(1,2)/ICSPDAT

Note 1: Pin feature is dependent on device configuration.
2: ECCP2, CCP3, CCP4, MSSP2 functions are only available on the PIC16F/LF1827.

Pin Diagram – 20-Pin SSOP (PIC16F/LF1826/27)

- RA0/AN0/CPS0/C12IN0-/SDO2(2)
- RA1/AN1/CPS1/C12IN1-/SS2
- RA2/AN2/CPS2/C12IN2-/C12IN+/VREF-/DACOUT
- RA3/AN3/CPS3/C12IN3-/C1IN+/VREF+/C1OUT/CCP3(2)/SRQ
- RA4/AN4/CPS4/C2OUT/T0CKI/CCP4(2)/SRNQ
- RA5/MCLR/VPP/SS1 (1,2)
- RA6/OSC2/CLKOUT/CLKR/P1D(1)/P2B(1,2)/SDO1(1)
- RA7/OSC1/CLKIN/P1C(1)/CCP2(1,2)/P2A(1,2)
- RA8/OSC3/CLKOUT/ACLK/P1D(1)/P2B(1,2)
- RA9/AN5/CPS5/T1CKI/T1OSI/P1C(1,3)/CCP2(1,2)/P2A(1,2)/ICSPCLK
- RB0/SRI/T1G/CCP1(1)/P1A(1)/INT/SRI/FLT0
- RB1/AN11/CPS11/RX(1,3)/DT(1,3)/SDA1/SDI1
- RB2/AN10/CPS10/MDMIN/TX(1,3)/CK(1,3)/RX(1)/DT(1)/SDA2(2)/SDI2(2)/SDO1(1,3)
- RB3/AN9/CPS9/MDOUT/CCP1(1)
- RB4/AN8/CPS8/SCL1/SCK1/MDCIN2
- RB5/AN7/CPS7/P1B/TX(1)/CK(1)/SCL2(2)/SCK2(2)/SS1 (1)
- RB6/AN5/CPS5/T1OSO/P1D(1)/P2B(1,2)/MDCIN1/ICSPDAT
- RB7/AN6/CPS6/T1OSI/T1OSO/P1D(1)/P2B(1,2)/ICSPCLK
- RB8/AN8/CPS8/T1OSI/T1OSO/P1D(1)/P2B(1,2)/ICSPDAT

Note 1: Pin feature is dependent on device configuration.
2: ECCP2, CCP3, CCP4, MSSP2 functions are only available on the PIC16F/LF1827.
Note 1: Pin feature is dependent on device configuration.
2: ECCP2, CCP3, CCP4, MSSP2 functions are only available on the PIC16F/LF1827.
<table>
<thead>
<tr>
<th>TABLE 1: 18/20/28-PIN SUMMARY (PIC16F/LF1826/27)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>I/O</strong></td>
</tr>
<tr>
<td>RA0</td>
</tr>
<tr>
<td>RA1</td>
</tr>
<tr>
<td>RA2</td>
</tr>
<tr>
<td>RA3</td>
</tr>
<tr>
<td>RA4</td>
</tr>
<tr>
<td>RA5</td>
</tr>
<tr>
<td>RA6</td>
</tr>
<tr>
<td>RB0</td>
</tr>
<tr>
<td>RB1</td>
</tr>
<tr>
<td>RB2</td>
</tr>
<tr>
<td>RB3</td>
</tr>
<tr>
<td>RB4</td>
</tr>
<tr>
<td>RB5</td>
</tr>
<tr>
<td>RB6</td>
</tr>
<tr>
<td>RB7</td>
</tr>
<tr>
<td>RB8</td>
</tr>
<tr>
<td>RB9</td>
</tr>
<tr>
<td>RB10</td>
</tr>
<tr>
<td>RB11</td>
</tr>
<tr>
<td>RB12</td>
</tr>
<tr>
<td>RB13</td>
</tr>
<tr>
<td>RB14</td>
</tr>
<tr>
<td>Vss</td>
</tr>
<tr>
<td>Vdd</td>
</tr>
<tr>
<td>RA0-RA5</td>
</tr>
<tr>
<td>RB0-RB13</td>
</tr>
<tr>
<td>RA0-RA1</td>
</tr>
<tr>
<td>RA2-RA3</td>
</tr>
<tr>
<td>RA4-RA5</td>
</tr>
<tr>
<td>RA6-RA7</td>
</tr>
<tr>
<td>RB8-RB9</td>
</tr>
<tr>
<td>Reference</td>
</tr>
<tr>
<td>A/D</td>
</tr>
<tr>
<td>ANSEL</td>
</tr>
<tr>
<td>28-Pin QFN/UQFN</td>
</tr>
<tr>
<td>20-Pin SSOP</td>
</tr>
<tr>
<td>18-Pin PDIP/SOIC</td>
</tr>
</tbody>
</table>

**Note 1:** Pin functions can be moved using the APFCON register(s).

**Note 2:** Functions are only available on the PIC16F/LF1827.

**Note 3:** Weak pull-up always enabled when MCLR is enabled, otherwise the pull-up is under user control.

**Note 4:** Default function location.
# Table of Contents

1.0 Device Overview ......................................................................................................................... 11
2.0 Enhanced Mid-range CPU ............................................................................................................. 17
3.0 Memory Organization .................................................................................................................... 19
4.0 Device Configuration .................................................................................................................... 49
5.0 Oscillator Module (With Fail-Safe Clock Monitor) ..................................................................... 55
6.0 Reference Clock Module .............................................................................................................. 71
7.0 Resets ......................................................................................................................................... 75
8.0 Interrupts ..................................................................................................................................... 83
9.0 Power-Down Mode (Sleep) .......................................................................................................... 99
10.0 Watchdog Timer (WDT) .......................................................................................................... 101
11.0 Data EEPROM and Flash Program Memory Control ............................................................... 105
12.0 I/O Ports .................................................................................................................................... 117
13.0 Interrupt-on-Change ................................................................................................................. 131
14.0 Fixed Voltage Reference (FVR) ............................................................................................... 135
15.0 Analog-to-Digital Converter (ADC) Module ............................................................................. 137
16.0 Digital-to-Analog Converter (DAC) Module ............................................................................. 151
17.0 Comparator Module .................................................................................................................... 157
18.0 SR Latch .................................................................................................................................... 167
19.0 Timer0 Module ........................................................................................................................ 173
21.0 Timer1 Module ........................................................................................................................ 177
22.0 Timer2/4/6 Modules ...................................................................................................................... 189
23.0 Data Signal Modulator (DSM) ................................................................................................ 193
24.0 Capture/Compare/PWM (ECCP1, ECCP2, ECCP3, CCP4) Modules ........................................... 203
25.0 Master Synchronous Serial Port (MSSP) Module ...................................................................... 233
26.0 Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) ............... 287
27.0 Capacitive Sensing Module ....................................................................................................... 315
28.0 In-Circuit Serial Programming™ (ICSP™) ............................................................................. 321
29.0 Instruction Set Summary ............................................................................................................ 323
30.0 Electrical Specifications ............................................................................................................. 337
31.0 DC and AC Characteristics Graphs and Tables ........................................................................ 371
32.0 Development Support ............................................................................................................... 373
33.0 Packaging Information .............................................................................................................. 377
Appendix A: Revision History ....................................................................................................... 387
Appendix B: Device Differences ................................................................................................... 387
Index ................................................................................................................................................ 389
The Microchip Web Site ................................................................................................................. 397
Customer Change Notification Service ......................................................................................... 397
Customer Support .......................................................................................................................... 397
Reader Response ............................................................................................................................. 398
Product Identification System ....................................................................................................... 399
## TO OUR VALUED CUSTOMERS

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

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

### Most Current Data Sheet

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

http://www.microchip.com

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

### Errata

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

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

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

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

### Customer Notification System

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

The PIC16F/LF1826/27 are described within this data sheet. They are available in 18/20/28-pin packages. Figure 1-1 shows a block diagram of the PIC16F/LF1826/27 devices. Table 1-2 shows the pinout descriptions.

Reference Table 1-1 for peripherals available per device.

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>PIC16F/LF1826</th>
<th>PIC16F/LF1827</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Capacitive Sensing Module</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Digital-to-Analog Converter (DAC)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Digital Signal Modulator (DSM)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>EUSART</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Fixed Voltage Reference (FVR)</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Reference Clock Module</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>SR Latch</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Capture/Compare/PWM Modules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECCP1</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>ECCP2</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>CCP3</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>CCP4</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Comparators</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>C2</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Master Synchronous Serial Ports</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSSP1</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>MSSP2</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Timers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timer0</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Timer1</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Timer2</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Timer4</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Timer6</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>
FIGURE 1-1: PIC16F/LF1826/27 BLOCK DIAGRAM

Note 1: See applicable chapters for more information on peripherals.
2: See Table 1-1 for peripherals available on specific devices.
# TABLE 1-2: PIC16F/LF1826/27 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/CPS0/C12IN0-/SDO2&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>RA0</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td>AN0</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 0 input.</td>
<td></td>
</tr>
<tr>
<td>CPS0</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 0.</td>
<td></td>
</tr>
<tr>
<td>C12IN0-</td>
<td>AN</td>
<td>—</td>
<td>Comparator C1 or C2 negative input.</td>
<td></td>
</tr>
<tr>
<td>SDO2</td>
<td>—</td>
<td>CMOS</td>
<td>SPI data output.</td>
<td></td>
</tr>
<tr>
<td>RA1/AN1/CPS1/C12IN1-/SS2&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>RA1</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td>AN1</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 1 input.</td>
<td></td>
</tr>
<tr>
<td>CPS1</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 1.</td>
<td></td>
</tr>
<tr>
<td>C12IN1-</td>
<td>AN</td>
<td>—</td>
<td>Comparator C1 or C2 negative input.</td>
<td></td>
</tr>
<tr>
<td>SS2</td>
<td>ST</td>
<td>—</td>
<td>Slave Select input 2.</td>
<td></td>
</tr>
<tr>
<td>RA2/AN2/CPS2/C12IN2-/C12IN+/VREF-/DACOUT</td>
<td>RA2</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td>AN2</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 2 input.</td>
<td></td>
</tr>
<tr>
<td>CPS2</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 2.</td>
<td></td>
</tr>
<tr>
<td>C12IN2-</td>
<td>AN</td>
<td>—</td>
<td>Comparator C1 or C2 negative input.</td>
<td></td>
</tr>
<tr>
<td>C12IN+</td>
<td>AN</td>
<td>—</td>
<td>Comparator C1 or C2 positive input.</td>
<td></td>
</tr>
<tr>
<td>VREF-</td>
<td>AN</td>
<td>—</td>
<td>A/D Negative Voltage Reference input.</td>
<td></td>
</tr>
<tr>
<td>DACOUT</td>
<td>—</td>
<td>AN</td>
<td>Voltage Reference output.</td>
<td></td>
</tr>
<tr>
<td>RA3/AN3/CPS3/C12IN3-/C1IN+/VREF+/C1OUT/CCP3&lt;sup&gt;(2)&lt;/sup&gt;/SRQ</td>
<td>RA3</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td>AN3</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 3 input.</td>
<td></td>
</tr>
<tr>
<td>CPS3</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 3.</td>
<td></td>
</tr>
<tr>
<td>C12IN3-</td>
<td>AN</td>
<td>—</td>
<td>Comparator C1 or C2 negative input.</td>
<td></td>
</tr>
<tr>
<td>C1IN+</td>
<td>AN</td>
<td>—</td>
<td>Comparator C1 positive input.</td>
<td></td>
</tr>
<tr>
<td>VREF+</td>
<td>AN</td>
<td>—</td>
<td>A/D Voltage Reference input.</td>
<td></td>
</tr>
<tr>
<td>C1OUT</td>
<td>—</td>
<td>CMOS</td>
<td>Comparator C1 output.</td>
<td></td>
</tr>
<tr>
<td>CCP3</td>
<td>ST</td>
<td>CMOS</td>
<td>Capture/Compare/PWM3.</td>
<td></td>
</tr>
<tr>
<td>SRQ</td>
<td>—</td>
<td>CMOS</td>
<td>SR Latch non-inverting output.</td>
<td></td>
</tr>
<tr>
<td>RA4/AN4/CPS4/C2OUT/T0CKI/CCP4&lt;sup&gt;(2)&lt;/sup&gt;/SRNQ</td>
<td>RA4</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td>AN4</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 4 input.</td>
<td></td>
</tr>
<tr>
<td>CPS4</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 4.</td>
<td></td>
</tr>
<tr>
<td>C2OUT</td>
<td>—</td>
<td>CMOS</td>
<td>Comparator C2 output.</td>
<td></td>
</tr>
<tr>
<td>T0CKI</td>
<td>ST</td>
<td>—</td>
<td>Timer0 clock input.</td>
<td></td>
</tr>
<tr>
<td>CCP4</td>
<td>ST</td>
<td>CMOS</td>
<td>Capture/Compare/PWM4.</td>
<td></td>
</tr>
<tr>
<td>SRNQ</td>
<td>—</td>
<td>CMOS</td>
<td>SR Latch inverting output.</td>
<td></td>
</tr>
<tr>
<td>RA5/MCLR/VPP/SS1&lt;sup&gt;(1,2)&lt;/sup&gt;</td>
<td>RA5</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td>MCLR</td>
<td>ST</td>
<td>—</td>
<td>Master Clear with internal pull-up.</td>
<td></td>
</tr>
<tr>
<td>VPP</td>
<td>HV</td>
<td>—</td>
<td>Programming voltage.</td>
<td></td>
</tr>
<tr>
<td>SS1</td>
<td>ST</td>
<td>—</td>
<td>Slave Select input 1.</td>
<td></td>
</tr>
</tbody>
</table>

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

**Note 1:** Pin functions can be moved using the APFCON register(s).
**Note 2:** Functions are only available on the PIC16F/LF1827.
**Note 3:** Default function location.
<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>RA6/OSC2/CLKOUT/CLKR/ P1D(1)/P2B(1,2)/SDO1(1)</td>
<td>RA6</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td>OSC2</td>
<td>—</td>
<td>XTAL</td>
<td>—</td>
<td>Crystal/Resonator (LP, XT, HS modes).</td>
</tr>
<tr>
<td>CLKOUT</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>Fosc/4 output.</td>
</tr>
<tr>
<td>CLKR</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>Clock Reference Output.</td>
</tr>
<tr>
<td>P1D</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>PWM output.</td>
</tr>
<tr>
<td>P2B</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>PWM output.</td>
</tr>
<tr>
<td>SDO1</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>SPI data output 1.</td>
</tr>
<tr>
<td>RA7/OSC1/CLKIN/P1C(1)/ CCP2(1,2)/P2A(1,2)</td>
<td>RA7</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O.</td>
</tr>
<tr>
<td>OSC1</td>
<td>XTAL</td>
<td>—</td>
<td>—</td>
<td>Crystal/Resonator (LP, XT, HS modes).</td>
</tr>
<tr>
<td>CLKIN</td>
<td>CMOS</td>
<td>—</td>
<td>—</td>
<td>External clock input (EC mode).</td>
</tr>
<tr>
<td>P1C</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>PWM output.</td>
</tr>
<tr>
<td>CCP2</td>
<td>ST</td>
<td>CMOS</td>
<td>—</td>
<td>Capture/Compare/PWM2.</td>
</tr>
<tr>
<td>P2A</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>PWM output.</td>
</tr>
<tr>
<td>RBO/T1G/CCP1(1)/P1A(1)/INT/ SRI/FLT0</td>
<td>RBO</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.</td>
</tr>
<tr>
<td>T1G</td>
<td>ST</td>
<td>—</td>
<td>—</td>
<td>Timer1 Gate input.</td>
</tr>
<tr>
<td>CCP1</td>
<td>ST</td>
<td>CMOS</td>
<td>—</td>
<td>Capture/Compare/PWM1.</td>
</tr>
<tr>
<td>P1A</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>PWM output.</td>
</tr>
<tr>
<td>INT</td>
<td>ST</td>
<td>—</td>
<td>—</td>
<td>External interrupt.</td>
</tr>
<tr>
<td>SRI</td>
<td>ST</td>
<td>—</td>
<td>—</td>
<td>SR Latch input.</td>
</tr>
<tr>
<td>FLT0</td>
<td>ST</td>
<td>—</td>
<td>—</td>
<td>ECCP Auto-Shutdown Fault input.</td>
</tr>
<tr>
<td>RB1/AN11/CPS11/RX(1,3)/ DT(1,3)/SDA1/SDI1</td>
<td>RB1</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.</td>
</tr>
<tr>
<td>AN11</td>
<td>AN</td>
<td>—</td>
<td>—</td>
<td>A/D Channel 11 input.</td>
</tr>
<tr>
<td>CPS11</td>
<td>AN</td>
<td>—</td>
<td>—</td>
<td>Capacitive sensing input 11.</td>
</tr>
<tr>
<td>RX</td>
<td>ST</td>
<td>—</td>
<td>—</td>
<td>USART asynchronous input.</td>
</tr>
<tr>
<td>DT</td>
<td>ST</td>
<td>CMOS</td>
<td>—</td>
<td>USART synchronous data.</td>
</tr>
<tr>
<td>SDA1</td>
<td>I²C™</td>
<td>OD</td>
<td>—</td>
<td>I²C™ data input/output 1.</td>
</tr>
<tr>
<td>SDI1</td>
<td>CMOS</td>
<td>—</td>
<td>—</td>
<td>SPI data input 1.</td>
</tr>
<tr>
<td>RB2/AN10/CPS10/MDMIN/ TX(1,3)/CK(1,3)/RX(1)/DT(1) /SDA2(2)/SDI2(2)/SDO1(1,3)</td>
<td>RB2</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.</td>
</tr>
<tr>
<td>AN10</td>
<td>AN</td>
<td>—</td>
<td>—</td>
<td>A/D Channel 10 input.</td>
</tr>
<tr>
<td>CPS10</td>
<td>AN</td>
<td>—</td>
<td>—</td>
<td>Capacitive sensing input 10.</td>
</tr>
<tr>
<td>MDMIN</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>Modulator source input.</td>
</tr>
<tr>
<td>TX</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>USART asynchronous transmit.</td>
</tr>
<tr>
<td>CK</td>
<td>ST</td>
<td>CMOS</td>
<td>—</td>
<td>USART synchronous clock.</td>
</tr>
<tr>
<td>RX</td>
<td>ST</td>
<td>—</td>
<td>—</td>
<td>USART asynchronous input.</td>
</tr>
<tr>
<td>DT</td>
<td>ST</td>
<td>CMOS</td>
<td>—</td>
<td>USART synchronous data.</td>
</tr>
<tr>
<td>SDA2</td>
<td>I²C™</td>
<td>OD</td>
<td>—</td>
<td>I²C™ data input/output 2.</td>
</tr>
<tr>
<td>SDI2</td>
<td>ST</td>
<td>—</td>
<td>—</td>
<td>SPI data input 2.</td>
</tr>
<tr>
<td>SDO1</td>
<td>—</td>
<td>CMOS</td>
<td>—</td>
<td>SPI data output 1.</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
- **HV** = High Voltage
- **XTAL** = Crystal
- **I²C™** = Schmitt Trigger input with I²C levels

**Note:**

1. Pin functions can be moved using the APFCON register(s).
2. Functions are only available on the PIC16F/LF1827.
3. Default function location.
### TABLE 1-2: PIC16F/LF1826/27 PINOUT DESCRIPTION (CONTINUED)

<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>RB3/AN9/CPS9/MDOU/CCP1(1,3)/P1A(1,3)</td>
<td>RB3</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.</td>
</tr>
<tr>
<td>AN9</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 9 input.</td>
<td></td>
</tr>
<tr>
<td>CPS9</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 9.</td>
<td></td>
</tr>
<tr>
<td>MDOU</td>
<td>—</td>
<td>CMOS</td>
<td>Modulator output.</td>
<td></td>
</tr>
<tr>
<td>CCP1</td>
<td>ST</td>
<td>CMOS</td>
<td>Capture/Compare/PWM1.</td>
<td></td>
</tr>
<tr>
<td>P1A</td>
<td>—</td>
<td>CMOS</td>
<td>PWM output.</td>
<td></td>
</tr>
<tr>
<td>RB4/AN8/CPS8/SCL1/SCK1/MDCIN2</td>
<td>RB4</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.</td>
</tr>
<tr>
<td>AN8</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 8 input.</td>
<td></td>
</tr>
<tr>
<td>CPS8</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 8.</td>
<td></td>
</tr>
<tr>
<td>SCL1</td>
<td>I²C™</td>
<td>OD</td>
<td>I²C™ clock 1.</td>
<td></td>
</tr>
<tr>
<td>SCK1</td>
<td>ST</td>
<td>CMOS</td>
<td>SPI clock 1.</td>
<td></td>
</tr>
<tr>
<td>MDCIN2</td>
<td>ST</td>
<td>—</td>
<td>Modulator Carrier Input 2.</td>
<td></td>
</tr>
<tr>
<td>RB5/AN7/CPS7/P1B/TX(1)/CK(1)/SCL2(2)/SCK2(2)/SS1</td>
<td>RB5</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.</td>
</tr>
<tr>
<td>AN7</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 7 input.</td>
<td></td>
</tr>
<tr>
<td>CPS7</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 7.</td>
<td></td>
</tr>
<tr>
<td>P1B</td>
<td>—</td>
<td>CMOS</td>
<td>PWM output.</td>
<td></td>
</tr>
<tr>
<td>TX</td>
<td>—</td>
<td>CMOS</td>
<td>USART asynchronous transmit.</td>
<td></td>
</tr>
<tr>
<td>CK</td>
<td>ST</td>
<td>CMOS</td>
<td>USART synchronous clock.</td>
<td></td>
</tr>
<tr>
<td>SCL2</td>
<td>I²C™</td>
<td>OD</td>
<td>I²C™ clock 2.</td>
<td></td>
</tr>
<tr>
<td>SCK2</td>
<td>ST</td>
<td>CMOS</td>
<td>SPI clock 2.</td>
<td></td>
</tr>
<tr>
<td>SS1</td>
<td>ST</td>
<td>—</td>
<td>Slave Select input 1.</td>
<td></td>
</tr>
<tr>
<td>RB6/AN5/CPS5/T1CKI/T1OSI/P1C(1,3)/CCP2(1,2,3)/P2A(1,2,3)/ICSPCLK</td>
<td>RB6</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.</td>
</tr>
<tr>
<td>AN5</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 5 input.</td>
<td></td>
</tr>
<tr>
<td>CPS5</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 5.</td>
<td></td>
</tr>
<tr>
<td>T1CKI</td>
<td>ST</td>
<td>—</td>
<td>Timer1 clock input.</td>
<td></td>
</tr>
<tr>
<td>T1OSO</td>
<td>XTAL</td>
<td>XTAL</td>
<td>Timer1 oscillator connection.</td>
<td></td>
</tr>
<tr>
<td>P1C</td>
<td>—</td>
<td>CMOS</td>
<td>PWM output.</td>
<td></td>
</tr>
<tr>
<td>CCP2</td>
<td>ST</td>
<td>CMOS</td>
<td>Capture/Compare/PWM2.</td>
<td></td>
</tr>
<tr>
<td>P2A</td>
<td>—</td>
<td>CMOS</td>
<td>PWM output.</td>
<td></td>
</tr>
<tr>
<td>ICSPCLK</td>
<td>ST</td>
<td>—</td>
<td>Serial Programming Clock.</td>
<td></td>
</tr>
<tr>
<td>RB7/AN6/CPS6/T1OSO/P1D(1,3)/P2B(1,2,3)/MDCIN1/ICSPDAT</td>
<td>RB7</td>
<td>TTL</td>
<td>CMOS</td>
<td>General purpose I/O. Individually controlled interrupt-on-change. Individually enabled pull-up.</td>
</tr>
<tr>
<td>AN6</td>
<td>AN</td>
<td>—</td>
<td>A/D Channel 6 input.</td>
<td></td>
</tr>
<tr>
<td>CPS6</td>
<td>AN</td>
<td>—</td>
<td>Capacitive sensing input 6.</td>
<td></td>
</tr>
<tr>
<td>T1OSO</td>
<td>XTAL</td>
<td>XTAL</td>
<td>Timer1 oscillator connection.</td>
<td></td>
</tr>
<tr>
<td>P1D</td>
<td>—</td>
<td>CMOS</td>
<td>PWM output.</td>
<td></td>
</tr>
<tr>
<td>P2B</td>
<td>—</td>
<td>CMOS</td>
<td>PWM output.</td>
<td></td>
</tr>
<tr>
<td>MDCIN1</td>
<td>ST</td>
<td>—</td>
<td>Modulator Carrier Input 1.</td>
<td></td>
</tr>
<tr>
<td>ICSPDAT</td>
<td>ST</td>
<td>CMOS</td>
<td>ICSP™ Data I/O.</td>
<td></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
- **I²C™** = Schmitt Trigger input with I²C levels
- **HV** = High Voltage
- **XTAL** = Crystal

**Note:**
1. Pin functions can be moved using the APFCON register(s).
2. Functions are only available on the PIC16F/LF1827.
3. Default function location.

© 2009 Microchip Technology Inc.
## TABLE 1-2: PIC16F/LF1826/27 PINOUT DESCRIPTION (CONTINUED)

<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>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, HV = High Voltage, XTAL = Crystal, I²C™ = Schmitt Trigger input with I²C levels.

Note 1: Pin functions can be moved using the APFCON register(s).
2: Functions are only available on the PIC16F/LF1827.
3: Default function location.
2.0 ENHANCED MID-RANGE CPU

This family of devices contains 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.

The Enhanced Mid-range CPU contains the following:

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

2.1 Automatic Interrupt Context Saving

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

2.2 16-level Stack with Overflow and Underflow

These devices have an external stack memory 15 bits wide and 16 words deep. A Stack Overflow or Underflow will set the appropriate bit (STKOVF or STKUNF) in the PCON register, and if enabled will cause a software Reset. See section Section 3.4 “Stack” for more details.

2.3 File Select Registers

There are two 16-bit File Select Registers (FSR). FSRs can access all file registers and program memory, which allows one data pointer for all memory. When an FSR points to program memory, there is 1 additional instruction cycle in instructions using INDFx to allow the data to be fetched. General purpose memory can also be addressed linearly, providing the ability to access contiguous data larger than 80 bytes. There are also new instructions to support the FSRs. See Section 3.5 “Indirect Addressing” for more details.

2.4 Instruction Set

There are 49 instructions for the enhanced mid-range CPU to support the features of the CPU. See Section 28.0 “Instruction Set Summary” for more details.
FIGURE 2-1: CORE BLOCK DIAGRAM
3.0 MEMORY ORGANIZATION

There are three types of memory in PIC16F/LF1826/27: Data Memory, Program Memory and Data EEPROM Memory\(^{(1)}\).

- Program Memory
- Data Memory
  - Core Registers
  - Special Function Registers
  - General Purpose RAM
  - Common RAM
  - Device Memory Maps
  - Special Function Registers Summary
- Data EEPROM memory\(^{(1)}\)

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

- PCL and PCLATH
- Stack
- Indirect Addressing

3.1 Program Memory Organization

The enhanced mid-range core has a 15-bit program counter capable of addressing a 32K x 14 program memory space. Table 3-1 shows the memory sizes implemented for the PIC16F/LF1826/27 family. Accessing a location above these boundaries will cause a wrap-around within the implemented memory space. The Reset vector is at 0000h and the interrupt vector is at 0004h (see Figures 3-1 and 3-2).

Note 1: The Data EEPROM Memory and the method to access Flash memory through the EECON registers is described in Section 11.0 “Data EEPROM and Flash Program Memory Control”.

<table>
<thead>
<tr>
<th>Device</th>
<th>Program Memory Space (Words)</th>
<th>Last Program Memory Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC16F/LF1826</td>
<td>2,048</td>
<td>07FFh</td>
</tr>
<tr>
<td>PIC16F/LF1827</td>
<td>4,096</td>
<td>0FFFh</td>
</tr>
</tbody>
</table>
FIGURE 3-1: PROGRAM MEMORY MAP AND STACK FOR PIC16F/LF1826

Stack Level 0
Stack Level 1
Stack Level 15

Reset Vector
0000h
Interrupt Vector
0004h
0005h
07FFh
0800h

On-chip Program Memory

Rollover to Page 0

PC<14:0>

CALL, CALLW
RETURN, RETLW
Interrupt, RETFIE

FIGURE 3-2: PROGRAM MEMORY MAP AND STACK FOR PIC16F/LF1827

Stack Level 0
Stack Level 1
Stack Level 15

Reset Vector
0000h
Interrupt Vector
0004h
0005h
07FFh
0800h

On-chip Program Memory

Rollover to Page 0

PC<14:0>

CALL, CALLW
RETURN, RETLW
Interrupt, RETFIE
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**

```plaintext
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 8 bits of the addressed word in the W register. Writes to the program memory cannot be performed via the INDF registers. Instructions that access the program memory via the FSR require one extra instruction cycle to complete. Example 3-2 demonstrates accessing the program memory via an FSR.

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

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

```assembly
; This code demonstrates accessing program memory via an FSR.

; Constants
RETLW DATA0 ; Index0 data
RETLW DATA1 ; Index1 data
RETLW DATA2
RETLW DATA3

; Function declaration
my_function

;... LOTS OF CODE...
MOVWF FSR1L
MOVWF FSR1H
MOVWI 0[INDF1] ; THE PROGRAM MEMORY IS IN W
```

3.2 Data Memory Organization

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

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

The active bank is selected by writing the bank number into the Bank Select Register (BSR). Unimplemented memory will read as '0'. All data memory can be accessed either directly (via instructions that use the file registers) or indirectly via the two File Select Registers (FSR). See Section 3.5 “Indirect Addressing” for more information.

3.2.1 CORE REGISTERS

The core registers contain the registers that directly affect the basic operation of the PIC16F/LF1826/27. These registers are listed below:

- INDF0
- INDF1
- PCL
- STATUS
- FSR0 Low
- FSR0 High
- FSR1 Low
- FSR1 High
- BSR
- WREG
- PCLATH
- INTCON

**Note:** The core registers are the first 12 addresses of every data memory bank.
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 28.0 “Instruction Set Summary”).

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

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>TO</td>
<td>PD</td>
<td>Z</td>
<td>DC(1)</td>
<td>C(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  -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(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)
   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.
3.2.2 SPECIAL FUNCTION REGISTER
The Special Function Registers are registers used by
the application to control the desired operation of
peripheral functions in the device. The registers asso-
ciated with the operation of the peripherals are
described in the appropriate peripheral chapter of this
data sheet.

3.2.3 GENERAL PURPOSE RAM
There are up to 80 bytes of GPR in each data memory
bank.

3.2.3.1 Linear Access to GPR
The general purpose RAM can be accessed in a
non-banked method via the FSRs. This can simplify
access to large memory structures. See Section 3.5.2
"Linear Data Memory" for more information.

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

FIGURE 3-3: BANKED MEMORY
PARTITIONING

<table>
<thead>
<tr>
<th>7-bit Bank Offset</th>
<th>Memory Region</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h</td>
<td>Core Registers (12 bytes)</td>
</tr>
<tr>
<td>0Bh</td>
<td>Special Function Registers (20 bytes maximum)</td>
</tr>
<tr>
<td>0Ch</td>
<td></td>
</tr>
<tr>
<td>1Fh</td>
<td>General Purpose RAM (80 bytes maximum)</td>
</tr>
<tr>
<td>20h</td>
<td></td>
</tr>
<tr>
<td>6Fh</td>
<td>Common RAM (16 bytes)</td>
</tr>
<tr>
<td>70h</td>
<td></td>
</tr>
<tr>
<td>7Fh</td>
<td></td>
</tr>
</tbody>
</table>

3.2.5 DEVICE MEMORY MAPS
The memory maps for the device family are as shown
in Table 3-2.

TABLE 3-2: MEMORY MAP TABLES

<table>
<thead>
<tr>
<th>Device</th>
<th>Banks</th>
<th>Table No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC16F/LF1826/27</td>
<td>0-7</td>
<td>Table 3-3</td>
</tr>
<tr>
<td></td>
<td>8-15</td>
<td>Table 3-4</td>
</tr>
<tr>
<td></td>
<td>16-23</td>
<td>Table 3-5</td>
</tr>
<tr>
<td></td>
<td>24-31</td>
<td>Table 3-6</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>Table 3-7</td>
</tr>
</tbody>
</table>
**TABLE 3-3: PIC16F/LF1826/27 MEMORY MAP, BANKS 0-7**

<table>
<thead>
<tr>
<th>BANK0</th>
<th>BANK1</th>
<th>BANK2</th>
<th>BANK3</th>
<th>BANK4</th>
<th>BANK5</th>
<th>BANK6</th>
<th>BANK7</th>
</tr>
</thead>
<tbody>
<tr>
<td>000h</td>
<td>INDF0</td>
<td>080h</td>
<td>INDF0</td>
<td>100h</td>
<td>INDF0</td>
<td>180h</td>
<td>INDF0</td>
</tr>
<tr>
<td>001h</td>
<td>INDF1</td>
<td>081h</td>
<td>INDF1</td>
<td>101h</td>
<td>INDF1</td>
<td>181h</td>
<td>INDF1</td>
</tr>
<tr>
<td>002h</td>
<td>PCL</td>
<td>082h</td>
<td>PCL</td>
<td>102h</td>
<td>PCL</td>
<td>182h</td>
<td>PCL</td>
</tr>
<tr>
<td>003h</td>
<td>STATUS</td>
<td>083h</td>
<td>STATUS</td>
<td>103h</td>
<td>STATUS</td>
<td>183h</td>
<td>STATUS</td>
</tr>
<tr>
<td>004h</td>
<td>FSROH</td>
<td>084h</td>
<td>FSROH</td>
<td>104h</td>
<td>FSROH</td>
<td>184h</td>
<td>FSROH</td>
</tr>
<tr>
<td>005h</td>
<td>FSROH</td>
<td>085h</td>
<td>FSROH</td>
<td>105h</td>
<td>FSROH</td>
<td>185h</td>
<td>FSROH</td>
</tr>
<tr>
<td>006h</td>
<td>FSRL</td>
<td>086h</td>
<td>FSRL</td>
<td>106h</td>
<td>FSRL</td>
<td>186h</td>
<td>FSRL</td>
</tr>
<tr>
<td>007h</td>
<td>FSRL</td>
<td>087h</td>
<td>FSRL</td>
<td>107h</td>
<td>FSRL</td>
<td>187h</td>
<td>FSRL</td>
</tr>
<tr>
<td>008h</td>
<td>BSR</td>
<td>088h</td>
<td>BSR</td>
<td>108h</td>
<td>BSR</td>
<td>188h</td>
<td>BSR</td>
</tr>
<tr>
<td>009h</td>
<td>BSR</td>
<td>089h</td>
<td>BSR</td>
<td>109h</td>
<td>BSR</td>
<td>189h</td>
<td>BSR</td>
</tr>
<tr>
<td>00Ah</td>
<td>WREG</td>
<td>08Ah</td>
<td>WREG</td>
<td>109h</td>
<td>WREG</td>
<td>18Ah</td>
<td>WREG</td>
</tr>
<tr>
<td>00Bh</td>
<td>WREG</td>
<td>08Bh</td>
<td>WREG</td>
<td>10Ah</td>
<td>WREG</td>
<td>18Bh</td>
<td>WREG</td>
</tr>
<tr>
<td>00Ch</td>
<td>PCLATH</td>
<td>08Ch</td>
<td>PCLATH</td>
<td>10Ah</td>
<td>PCLATH</td>
<td>18Ch</td>
<td>PCLATH</td>
</tr>
<tr>
<td>00Dh</td>
<td>PCLATH</td>
<td>08Dh</td>
<td>PCLATH</td>
<td>10Bh</td>
<td>PCLATH</td>
<td>18Dh</td>
<td>PCLATH</td>
</tr>
<tr>
<td>00Eh</td>
<td>—</td>
<td>08Eh</td>
<td>—</td>
<td>10Bh</td>
<td>—</td>
<td>18Eh</td>
<td>—</td>
</tr>
<tr>
<td>00Fh</td>
<td>—</td>
<td>08Fh</td>
<td>—</td>
<td>10Ch</td>
<td>—</td>
<td>18Fh</td>
<td>—</td>
</tr>
<tr>
<td>010h</td>
<td>—</td>
<td>09Fh</td>
<td>—</td>
<td>110h</td>
<td>—</td>
<td>19Fh</td>
<td>—</td>
</tr>
<tr>
<td>011h</td>
<td>PIR1</td>
<td>091h</td>
<td>PIE1</td>
<td>111h</td>
<td>CM1CON0</td>
<td>191h</td>
<td>EEDRL</td>
</tr>
<tr>
<td>012h</td>
<td>PIR2</td>
<td>092h</td>
<td>PIE2</td>
<td>112h</td>
<td>CM1CON1</td>
<td>192h</td>
<td>EEDRH</td>
</tr>
<tr>
<td>013h</td>
<td>PIR3(1)</td>
<td>093h</td>
<td>PIE3(1)</td>
<td>113h</td>
<td>CM2CON0</td>
<td>193h</td>
<td>EEDATL</td>
</tr>
<tr>
<td>014h</td>
<td>PIR4(1)</td>
<td>094h</td>
<td>PIE4(1)</td>
<td>114h</td>
<td>CM2CON1</td>
<td>194h</td>
<td>EEDATH</td>
</tr>
<tr>
<td>015h</td>
<td>TMRO</td>
<td>095h</td>
<td>OPTION</td>
<td>115h</td>
<td>GMOUT</td>
<td>195h</td>
<td>EEC0N</td>
</tr>
<tr>
<td>016h</td>
<td>TMRO</td>
<td>096h</td>
<td>PCON</td>
<td>116h</td>
<td>BORCON</td>
<td>196h</td>
<td>EEC0N</td>
</tr>
<tr>
<td>017h</td>
<td>TMRO</td>
<td>097h</td>
<td>PORTA</td>
<td>117h</td>
<td>WVRON</td>
<td>197h</td>
<td>FVRCON</td>
</tr>
<tr>
<td>018h</td>
<td>T1CON</td>
<td>098h</td>
<td>OSCI</td>
<td>118h</td>
<td>DACCON0</td>
<td>198h</td>
<td>—</td>
</tr>
<tr>
<td>019h</td>
<td>T1CON</td>
<td>099h</td>
<td>OSCCON</td>
<td>119h</td>
<td>DACCON1</td>
<td>199h</td>
<td>RCREG</td>
</tr>
<tr>
<td>01Ah</td>
<td>T2CON</td>
<td>09Ah</td>
<td>OSCST</td>
<td>120h</td>
<td>Src0A</td>
<td>19Ah</td>
<td>TXREG</td>
</tr>
<tr>
<td>01Bh</td>
<td>T2CON</td>
<td>09Bh</td>
<td>ADRESH</td>
<td>121h</td>
<td>SRC0B</td>
<td>19Bh</td>
<td>SBPRG</td>
</tr>
<tr>
<td>01Ch</td>
<td>T2CON</td>
<td>09Ch</td>
<td>ADRESH</td>
<td>122h</td>
<td>SRC0B</td>
<td>19Ch</td>
<td>SBPRG</td>
</tr>
<tr>
<td>01Dh</td>
<td>CPSCON</td>
<td>09Dh</td>
<td>ADCCON0</td>
<td>123h</td>
<td>APFC0N</td>
<td>19Dh</td>
<td>RCSTA</td>
</tr>
<tr>
<td>01 Eh</td>
<td>CPSCON</td>
<td>09 Eh</td>
<td>CPSCON0</td>
<td>124h</td>
<td>APFC0N</td>
<td>19Eh</td>
<td>TXSTA</td>
</tr>
<tr>
<td>01Fh</td>
<td>CPSCON</td>
<td>09Fh</td>
<td>—</td>
<td>125h</td>
<td>—</td>
<td>19Fh</td>
<td>BAUDC</td>
</tr>
</tbody>
</table>

| 020h  | —     | 0A0h  | —     | 130h  | —     | 1AFh | —     |

---

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

**Note 1:** Available only on PIC16F/LF1827.
<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>INDF0</td>
<td>480h</td>
<td>INDF0</td>
<td>500h</td>
<td>INDF0</td>
<td>580h</td>
<td>INDF0</td>
</tr>
<tr>
<td>401h</td>
<td>INDF1</td>
<td>481h</td>
<td>INDF1</td>
<td>501h</td>
<td>INDF1</td>
<td>581h</td>
<td>INDF1</td>
</tr>
<tr>
<td>402h</td>
<td>PCL</td>
<td>482h</td>
<td>PCL</td>
<td>502h</td>
<td>PCL</td>
<td>582h</td>
<td>PCL</td>
</tr>
<tr>
<td>403h</td>
<td>STATUS</td>
<td>483h</td>
<td>STATUS</td>
<td>503h</td>
<td>STATUS</td>
<td>583h</td>
<td>STATUS</td>
</tr>
<tr>
<td>404h</td>
<td>FSR0L</td>
<td>484h</td>
<td>FSR0L</td>
<td>504h</td>
<td>FSR0L</td>
<td>584h</td>
<td>FSR0L</td>
</tr>
<tr>
<td>405h</td>
<td>FSR0H</td>
<td>485h</td>
<td>FSR0H</td>
<td>505h</td>
<td>FSR0H</td>
<td>585h</td>
<td>FSR0H</td>
</tr>
<tr>
<td>406h</td>
<td>FSR1L</td>
<td>486h</td>
<td>FSR1L</td>
<td>506h</td>
<td>FSR1L</td>
<td>586h</td>
<td>FSR1L</td>
</tr>
<tr>
<td>407h</td>
<td>FSR1H</td>
<td>487h</td>
<td>FSR1H</td>
<td>507h</td>
<td>FSR1H</td>
<td>587h</td>
<td>FSR1H</td>
</tr>
<tr>
<td>408h</td>
<td>BSR</td>
<td>488h</td>
<td>BSR</td>
<td>508h</td>
<td>BSR</td>
<td>588h</td>
<td>BSR</td>
</tr>
<tr>
<td>409h</td>
<td>WREG</td>
<td>489h</td>
<td>WREG</td>
<td>509h</td>
<td>WREG</td>
<td>589h</td>
<td>WREG</td>
</tr>
<tr>
<td>40Ah</td>
<td>PCLATH</td>
<td>48Ah</td>
<td>PCLATH</td>
<td>50Ah</td>
<td>PCLATH</td>
<td>58Ah</td>
<td>PCLATH</td>
</tr>
<tr>
<td>40Bh</td>
<td>INTCN</td>
<td>48Bh</td>
<td>INTCN</td>
<td>50Bh</td>
<td>INTCN</td>
<td>58Bh</td>
<td>INTCN</td>
</tr>
<tr>
<td>40Ch</td>
<td>—</td>
<td>48Ch</td>
<td>—</td>
<td>50Ch</td>
<td>—</td>
<td>58Ch</td>
<td>—</td>
</tr>
<tr>
<td>40Dh</td>
<td>—</td>
<td>48Dh</td>
<td>—</td>
<td>50Dh</td>
<td>—</td>
<td>58Dh</td>
<td>—</td>
</tr>
<tr>
<td>40 Eh</td>
<td>—</td>
<td>48 Eh</td>
<td>—</td>
<td>50 Eh</td>
<td>—</td>
<td>58 Eh</td>
<td>—</td>
</tr>
<tr>
<td>40Fh</td>
<td>—</td>
<td>48Fh</td>
<td>—</td>
<td>50Fh</td>
<td>—</td>
<td>58Fh</td>
<td>—</td>
</tr>
<tr>
<td>410h</td>
<td>—</td>
<td>480h</td>
<td>—</td>
<td>510h</td>
<td>—</td>
<td>590h</td>
<td>—</td>
</tr>
<tr>
<td>411h</td>
<td>—</td>
<td>491h</td>
<td>—</td>
<td>511h</td>
<td>—</td>
<td>611h</td>
<td>—</td>
</tr>
<tr>
<td>412h</td>
<td>—</td>
<td>492h</td>
<td>—</td>
<td>512h</td>
<td>—</td>
<td>612h</td>
<td>—</td>
</tr>
<tr>
<td>413h</td>
<td>—</td>
<td>493h</td>
<td>—</td>
<td>513h</td>
<td>—</td>
<td>613h</td>
<td>—</td>
</tr>
<tr>
<td>414h</td>
<td>—</td>
<td>494h</td>
<td>—</td>
<td>514h</td>
<td>—</td>
<td>614h</td>
<td>—</td>
</tr>
<tr>
<td>415h</td>
<td>—</td>
<td>495h</td>
<td>—</td>
<td>515h</td>
<td>—</td>
<td>615h</td>
<td>—</td>
</tr>
<tr>
<td>416h</td>
<td>—</td>
<td>496h</td>
<td>—</td>
<td>516h</td>
<td>—</td>
<td>616h</td>
<td>—</td>
</tr>
<tr>
<td>417h</td>
<td>—</td>
<td>497h</td>
<td>—</td>
<td>517h</td>
<td>—</td>
<td>617h</td>
<td>—</td>
</tr>
<tr>
<td>418h</td>
<td>—</td>
<td>498h</td>
<td>—</td>
<td>518h</td>
<td>—</td>
<td>618h</td>
<td>—</td>
</tr>
<tr>
<td>419h</td>
<td>—</td>
<td>499h</td>
<td>—</td>
<td>519h</td>
<td>—</td>
<td>619h</td>
<td>—</td>
</tr>
<tr>
<td>41Ah</td>
<td>—</td>
<td>49Ah</td>
<td>—</td>
<td>51Ah</td>
<td>—</td>
<td>61Ah</td>
<td>—</td>
</tr>
<tr>
<td>41Bh</td>
<td>—</td>
<td>49Bh</td>
<td>—</td>
<td>51Bh</td>
<td>—</td>
<td>61Bh</td>
<td>—</td>
</tr>
<tr>
<td>41Ch</td>
<td>—</td>
<td>49Ch</td>
<td>—</td>
<td>51Ch</td>
<td>—</td>
<td>61Ch</td>
<td>—</td>
</tr>
<tr>
<td>41Dh</td>
<td>—</td>
<td>49Dh</td>
<td>—</td>
<td>51Dh</td>
<td>—</td>
<td>61Dh</td>
<td>—</td>
</tr>
<tr>
<td>41 Eh</td>
<td>—</td>
<td>49 Eh</td>
<td>—</td>
<td>51 Eh</td>
<td>—</td>
<td>61 Eh</td>
<td>—</td>
</tr>
<tr>
<td>41Fh</td>
<td>—</td>
<td>49Fh</td>
<td>—</td>
<td>51Fh</td>
<td>—</td>
<td>61Fh</td>
<td>—</td>
</tr>
</tbody>
</table>

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

TABLE 3-4: PIC16F/LF1826/27 MEMORY MAP, BANKS 8-15
### TABLE 3-5: PIC16F/LF1826/27 MEMORY MAP, BANKS 16-23)

<table>
<thead>
<tr>
<th>BANK16</th>
<th>BANK17</th>
<th>BANK18</th>
<th>BANK19</th>
<th>BANK20</th>
<th>BANK21</th>
<th>BANK22</th>
<th>BANK23</th>
</tr>
</thead>
<tbody>
<tr>
<td>800h</td>
<td>INDF0</td>
<td>800h</td>
<td>INDF0</td>
<td>900h</td>
<td>INDF0</td>
<td>A00h</td>
<td>B00h</td>
</tr>
<tr>
<td>801h</td>
<td>INDF1</td>
<td>801h</td>
<td>INDF1</td>
<td>901h</td>
<td>INDF1</td>
<td>A01h</td>
<td>B01h</td>
</tr>
<tr>
<td>802h</td>
<td>PCL</td>
<td>802h</td>
<td>PCL</td>
<td>902h</td>
<td>PCL</td>
<td>A02h</td>
<td>B02h</td>
</tr>
<tr>
<td>803h</td>
<td>STATUS</td>
<td>803h</td>
<td>STATUS</td>
<td>903h</td>
<td>STATUS</td>
<td>A03h</td>
<td>B03h</td>
</tr>
<tr>
<td>804h</td>
<td>FSROL</td>
<td>804h</td>
<td>FSROL</td>
<td>904h</td>
<td>FSROL</td>
<td>A04h</td>
<td>B04h</td>
</tr>
<tr>
<td>805h</td>
<td>FSROH</td>
<td>805h</td>
<td>FSROH</td>
<td>905h</td>
<td>FSROH</td>
<td>A05h</td>
<td>B05h</td>
</tr>
<tr>
<td>806h</td>
<td>FSRL1</td>
<td>806h</td>
<td>FSRL1</td>
<td>906h</td>
<td>FSRL1</td>
<td>A06h</td>
<td>B06h</td>
</tr>
<tr>
<td>807h</td>
<td>FSRIH</td>
<td>807h</td>
<td>FSRIH</td>
<td>907h</td>
<td>FSRIH</td>
<td>A07h</td>
<td>B07h</td>
</tr>
<tr>
<td>808h</td>
<td>BSR</td>
<td>808h</td>
<td>BSR</td>
<td>908h</td>
<td>BSR</td>
<td>A08h</td>
<td>B08h</td>
</tr>
<tr>
<td>809h</td>
<td>WREG</td>
<td>809h</td>
<td>WREG</td>
<td>909h</td>
<td>WREG</td>
<td>A09h</td>
<td>B09h</td>
</tr>
<tr>
<td>80Ah</td>
<td>PCLATH</td>
<td>80Ah</td>
<td>PCLATH</td>
<td>90Ah</td>
<td>PCLATH</td>
<td>A0Ah</td>
<td>B0Ah</td>
</tr>
<tr>
<td>80Bh</td>
<td>INTCN</td>
<td>80Bh</td>
<td>INTCN</td>
<td>90Bh</td>
<td>INTCN</td>
<td>A0Bh</td>
<td>B0Bh</td>
</tr>
<tr>
<td>80Ch</td>
<td>—</td>
<td>80Ch</td>
<td>—</td>
<td>90Ch</td>
<td>—</td>
<td>A0Ch</td>
<td>—</td>
</tr>
<tr>
<td>80Dh</td>
<td>—</td>
<td>80Dh</td>
<td>—</td>
<td>90Dh</td>
<td>—</td>
<td>A0Dh</td>
<td>—</td>
</tr>
<tr>
<td>80Eh</td>
<td>—</td>
<td>80Eh</td>
<td>—</td>
<td>90Eh</td>
<td>—</td>
<td>A0Eh</td>
<td>—</td>
</tr>
<tr>
<td>80Fh</td>
<td>—</td>
<td>80Fh</td>
<td>—</td>
<td>90Fh</td>
<td>—</td>
<td>A0Fh</td>
<td>—</td>
</tr>
<tr>
<td>810h</td>
<td>—</td>
<td>810h</td>
<td>—</td>
<td>910h</td>
<td>—</td>
<td>A10h</td>
<td>—</td>
</tr>
<tr>
<td>811h</td>
<td>—</td>
<td>811h</td>
<td>—</td>
<td>911h</td>
<td>—</td>
<td>A11h</td>
<td>—</td>
</tr>
<tr>
<td>812h</td>
<td>—</td>
<td>812h</td>
<td>—</td>
<td>912h</td>
<td>—</td>
<td>A12h</td>
<td>—</td>
</tr>
<tr>
<td>813h</td>
<td>—</td>
<td>813h</td>
<td>—</td>
<td>913h</td>
<td>—</td>
<td>A13h</td>
<td>—</td>
</tr>
<tr>
<td>814h</td>
<td>—</td>
<td>814h</td>
<td>—</td>
<td>914h</td>
<td>—</td>
<td>A14h</td>
<td>—</td>
</tr>
<tr>
<td>815h</td>
<td>—</td>
<td>815h</td>
<td>—</td>
<td>915h</td>
<td>—</td>
<td>A15h</td>
<td>—</td>
</tr>
<tr>
<td>816h</td>
<td>—</td>
<td>816h</td>
<td>—</td>
<td>916h</td>
<td>—</td>
<td>A16h</td>
<td>—</td>
</tr>
<tr>
<td>817h</td>
<td>—</td>
<td>817h</td>
<td>—</td>
<td>917h</td>
<td>—</td>
<td>A17h</td>
<td>—</td>
</tr>
<tr>
<td>818h</td>
<td>—</td>
<td>818h</td>
<td>—</td>
<td>918h</td>
<td>—</td>
<td>A18h</td>
<td>—</td>
</tr>
<tr>
<td>819h</td>
<td>—</td>
<td>819h</td>
<td>—</td>
<td>919h</td>
<td>—</td>
<td>A19h</td>
<td>—</td>
</tr>
<tr>
<td>81Ah</td>
<td>—</td>
<td>81Ah</td>
<td>—</td>
<td>91Ah</td>
<td>—</td>
<td>A1Ah</td>
<td>—</td>
</tr>
<tr>
<td>81Bh</td>
<td>—</td>
<td>81Bh</td>
<td>—</td>
<td>91Bh</td>
<td>—</td>
<td>A1Bh</td>
<td>—</td>
</tr>
<tr>
<td>81Ch</td>
<td>—</td>
<td>81Ch</td>
<td>—</td>
<td>91Ch</td>
<td>—</td>
<td>A1Ch</td>
<td>—</td>
</tr>
<tr>
<td>81Dh</td>
<td>—</td>
<td>81Dh</td>
<td>—</td>
<td>91Dh</td>
<td>—</td>
<td>A1Dh</td>
<td>—</td>
</tr>
<tr>
<td>81 Eh</td>
<td>—</td>
<td>81 Eh</td>
<td>—</td>
<td>91 Eh</td>
<td>—</td>
<td>A1 Eh</td>
<td>—</td>
</tr>
<tr>
<td>81Fh</td>
<td>—</td>
<td>81Fh</td>
<td>—</td>
<td>91Fh</td>
<td>—</td>
<td>A1Fh</td>
<td>—</td>
</tr>
</tbody>
</table>

Legend:  
- = Unimplemented data memory locations, read as ‘0’
### TABLE 3-6: PIC16F/LF1826/27 MEMORY MAP, BANKS 24-31

<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>INDF0</td>
<td>C80h</td>
<td>D00h</td>
<td>D80h</td>
<td>E00h</td>
<td>INDFO</td>
<td>F00h</td>
</tr>
<tr>
<td>C01h</td>
<td>INDF1</td>
<td>C81h</td>
<td>D01h</td>
<td>D81h</td>
<td>E01h</td>
<td>INDFO</td>
<td>F01h</td>
</tr>
<tr>
<td>C02h</td>
<td>PCL</td>
<td>C82h</td>
<td>D02h</td>
<td>D82h</td>
<td>E02h</td>
<td>PCL</td>
<td>F02h</td>
</tr>
<tr>
<td>C03h</td>
<td>STATUS</td>
<td>C83h</td>
<td>D03h</td>
<td>D83h</td>
<td>E03h</td>
<td>STATUS</td>
<td>F03h</td>
</tr>
<tr>
<td>C04h</td>
<td>FSR0H</td>
<td>C84h</td>
<td>D04h</td>
<td>D84h</td>
<td>E04h</td>
<td>FSR0H</td>
<td>F04h</td>
</tr>
<tr>
<td>C05h</td>
<td>FSR1H</td>
<td>C85h</td>
<td>D05h</td>
<td>D85h</td>
<td>E05h</td>
<td>FSR1H</td>
<td>F05h</td>
</tr>
<tr>
<td>C06h</td>
<td>BSR</td>
<td>C86h</td>
<td>D06h</td>
<td>D86h</td>
<td>E06h</td>
<td>BSR</td>
<td>F06h</td>
</tr>
<tr>
<td>C07h</td>
<td>WREG</td>
<td>C87h</td>
<td>D07h</td>
<td>D87h</td>
<td>E07h</td>
<td>WREG</td>
<td>F07h</td>
</tr>
<tr>
<td>C08h</td>
<td>PCLATH</td>
<td>C88h</td>
<td>D08h</td>
<td>D88h</td>
<td>E08h</td>
<td>PCLATH</td>
<td>F08h</td>
</tr>
<tr>
<td>C09h</td>
<td>—</td>
<td>C89h</td>
<td>D09h</td>
<td>D89h</td>
<td>E09h</td>
<td>—</td>
<td>F09h</td>
</tr>
<tr>
<td>C0Ah</td>
<td>—</td>
<td>C8Ah</td>
<td>D0Ah</td>
<td>D8Ah</td>
<td>E0Ah</td>
<td>—</td>
<td>F0Ah</td>
</tr>
<tr>
<td>C0Bh</td>
<td>—</td>
<td>C8Bh</td>
<td>D0Bh</td>
<td>D8Bh</td>
<td>E0Bh</td>
<td>—</td>
<td>F0Bh</td>
</tr>
<tr>
<td>C0Ch</td>
<td>—</td>
<td>C8Ch</td>
<td>D0Ch</td>
<td>D8Ch</td>
<td>E0Ch</td>
<td>—</td>
<td>F0Ch</td>
</tr>
<tr>
<td>C0Dh</td>
<td>—</td>
<td>C8Dh</td>
<td>D0Dh</td>
<td>D8Dh</td>
<td>E0Dh</td>
<td>—</td>
<td>F0Dh</td>
</tr>
<tr>
<td>C0Eh</td>
<td>—</td>
<td>C8Eh</td>
<td>D0Eh</td>
<td>D8Eh</td>
<td>E0Eh</td>
<td>—</td>
<td>F0Eh</td>
</tr>
<tr>
<td>C0Fh</td>
<td>—</td>
<td>C8Fh</td>
<td>D0Fh</td>
<td>D8Fh</td>
<td>E0Fh</td>
<td>—</td>
<td>F0Fh</td>
</tr>
<tr>
<td>C10h</td>
<td>—</td>
<td>C90h</td>
<td>D10h</td>
<td>D90h</td>
<td>E10h</td>
<td>—</td>
<td>F10h</td>
</tr>
<tr>
<td>C11h</td>
<td>—</td>
<td>C91h</td>
<td>D11h</td>
<td>D91h</td>
<td>E11h</td>
<td>—</td>
<td>F11h</td>
</tr>
<tr>
<td>C12h</td>
<td>—</td>
<td>C92h</td>
<td>D12h</td>
<td>D92h</td>
<td>E12h</td>
<td>—</td>
<td>F12h</td>
</tr>
<tr>
<td>C13h</td>
<td>—</td>
<td>C93h</td>
<td>D13h</td>
<td>D93h</td>
<td>E13h</td>
<td>—</td>
<td>F13h</td>
</tr>
<tr>
<td>C14h</td>
<td>—</td>
<td>C94h</td>
<td>D14h</td>
<td>D94h</td>
<td>E14h</td>
<td>—</td>
<td>F14h</td>
</tr>
<tr>
<td>C15h</td>
<td>—</td>
<td>C95h</td>
<td>D15h</td>
<td>D95h</td>
<td>E15h</td>
<td>—</td>
<td>F15h</td>
</tr>
<tr>
<td>C16h</td>
<td>—</td>
<td>C96h</td>
<td>D16h</td>
<td>D96h</td>
<td>E16h</td>
<td>—</td>
<td>F16h</td>
</tr>
<tr>
<td>C17h</td>
<td>—</td>
<td>C97h</td>
<td>D17h</td>
<td>D97h</td>
<td>E17h</td>
<td>—</td>
<td>F17h</td>
</tr>
<tr>
<td>C18h</td>
<td>—</td>
<td>C98h</td>
<td>D18h</td>
<td>D98h</td>
<td>E18h</td>
<td>—</td>
<td>F18h</td>
</tr>
<tr>
<td>C19h</td>
<td>—</td>
<td>C99h</td>
<td>D19h</td>
<td>D99h</td>
<td>E19h</td>
<td>—</td>
<td>F19h</td>
</tr>
<tr>
<td>C1Ah</td>
<td>—</td>
<td>CA0h</td>
<td>D20h</td>
<td>D10h</td>
<td>E20h</td>
<td>—</td>
<td>F20h</td>
</tr>
<tr>
<td>C1Bh</td>
<td>—</td>
<td>CA1h</td>
<td>D21h</td>
<td>D11h</td>
<td>E21h</td>
<td>—</td>
<td>F21h</td>
</tr>
<tr>
<td>C1Ch</td>
<td>—</td>
<td>CA2h</td>
<td>D22h</td>
<td>D12h</td>
<td>E22h</td>
<td>—</td>
<td>F22h</td>
</tr>
<tr>
<td>C1Dh</td>
<td>—</td>
<td>CA3h</td>
<td>D23h</td>
<td>D13h</td>
<td>E23h</td>
<td>—</td>
<td>F23h</td>
</tr>
<tr>
<td>C1Eh</td>
<td>—</td>
<td>CA4h</td>
<td>D24h</td>
<td>D14h</td>
<td>E24h</td>
<td>—</td>
<td>F24h</td>
</tr>
<tr>
<td>C1Fh</td>
<td>—</td>
<td>CA5h</td>
<td>D25h</td>
<td>D15h</td>
<td>E25h</td>
<td>—</td>
<td>F25h</td>
</tr>
</tbody>
</table>

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

Accesses: 70h – 7Fh

See Table 3-7 for more information on accesses.
### TABLE 3-7: PIC16F/LF1826/27 MEMORY MAP, BANK 31

<table>
<thead>
<tr>
<th>Bank 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA0h</td>
</tr>
</tbody>
</table>
| FE3h    | Unimplemented Read as '0'
| FE4h    | STATUS_SHAD
| FE5h    | WREG_SHAD
| FE6h    | BSR_SHAD
| FE7h    | PCLATH_SHAD
| FE8h    | FSR0L_SHAD
| FE9h    | FSR0H_SHAD
| FEAh    | FSR1L_SHAD
| FEBh    | FSR1H_SHAD
| FECh    | —
| FEDh    | STKPTR
| FEEh    | TOSL
| FEFh    | TOSH

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

### 3.2.6 SPECIAL FUNCTION REGISTERS SUMMARY

The Special Function Register Summary for the device family are as follows:

<table>
<thead>
<tr>
<th>Device</th>
<th>Bank(s)</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC16F/LF1826/27</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>9-30</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>40</td>
</tr>
</tbody>
</table>
**TABLE 3-8: 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>000h(R)</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>001h(R)</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>002h(R)</td>
<td>PCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>003h(R)</td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>T0</td>
<td>PD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Z</td>
<td>DC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C</td>
<td>- - 1 1000</td>
</tr>
<tr>
<td>004h(R)</td>
<td>FSR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>005h(R)</td>
<td>FSR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>006h(R)</td>
<td>FSR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>007h(R)</td>
<td>FSR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>008h(R)</td>
<td>BSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>- - 0 0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BSR4</td>
<td>BSR3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BSR2</td>
<td>BSR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BSR0</td>
<td>- - 0 0000</td>
</tr>
<tr>
<td>009h(R)</td>
<td>WREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>00Ah(R)</td>
<td>PCLATH</td>
<td></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>
</tr>
<tr>
<td>00Bh(R)</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>IOCF</td>
<td>0000 000x</td>
<td>0000 000u</td>
</tr>
<tr>
<td>00Ch</td>
<td>PORTA</td>
<td>RA7</td>
<td>RA6</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
<td>xxxxx xxxxx</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>00Dh</td>
<td>PORTB</td>
<td>RB7</td>
<td>RB6</td>
<td>RB5</td>
<td>RB4</td>
<td>RB3</td>
<td>RB2</td>
<td>RB1</td>
<td>RB0</td>
<td>xxxxx xxxxx</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>00Eh</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>00Fh</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>010h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>011h</td>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>012h</td>
<td>PIR2</td>
<td>NOSIF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>EEIF</td>
<td>BCL1IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>013h</td>
<td>PIR3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>014h</td>
<td>PIR4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>015h</td>
<td>TMR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer0 Module Register</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>016h</td>
<td>TMR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Holding Register for the Least Significant Byte of the 16-bit TMR1 Register</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>017h</td>
<td>TMR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Holding Register for the Most Significant Byte of the 16-bit TMR1 Register</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>018h</td>
<td>T1CON</td>
<td>TMR1CS1</td>
<td>TMR1CS0</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>T1SYNC</td>
<td>—</td>
<td>TMR1ON</td>
<td>0000 00-0</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<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>019h</td>
<td>T1GCON</td>
<td>TMR1GE</td>
<td>T1POL</td>
<td>T1TM</td>
<td>T1GPM</td>
<td>T1GO/ DONE</td>
<td>T1GVAL</td>
<td>T1GSS1</td>
<td>T1GSS0</td>
<td>0000 0x00</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>01Ah</td>
<td>TMR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer2 Module Register</td>
<td>0000 0000</td>
</tr>
<tr>
<td>01Bh</td>
<td>PR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer2 Period Register</td>
<td>1111 1111 1111 1111</td>
</tr>
<tr>
<td>01Ch</td>
<td>T2CON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>01Dh</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>01Eh</td>
<td>CPSCON0</td>
<td>CPSON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>01Fh</td>
<td>CPSCON1</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:** PIC16F/LF1827 only.

2: These registers can be addressed from any bank.
### TABLE 3-8: 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>080h</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxx xxxx xxxx xxxx</td>
<td>xxxx xxxx xxxx xxxx</td>
</tr>
<tr>
<td>081h</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxx xxxx xxxx xxxx</td>
<td>xxxx xxxx xxxx xxxx</td>
</tr>
<tr>
<td>082h</td>
<td>PCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>083h</td>
<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>---1 1000 ---q quu</td>
<td>---1 1000 ---q quu</td>
</tr>
<tr>
<td>084h</td>
<td>FSR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>085h</td>
<td>FSR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>086h</td>
<td>FSR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>087h</td>
<td>FSR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>088h</td>
<td>BSR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BSR4</td>
<td>BSR3</td>
<td>BSR2</td>
<td>---0 0000 ---0 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>089h</td>
<td>WREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>08Ah</td>
<td>PCLATH</td>
<td>—</td>
<td>Write Buffer for the upper 7 bits of the Program Counter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>---0 0000 ---0 0000</td>
<td>---0 0000 ---0 0000</td>
</tr>
<tr>
<td>08Bh</td>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TIMER0IE</td>
<td>INT</td>
<td>IOIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>08Ch</td>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>1111 1111 1111 1111</td>
<td>1111 1111 1111 1111</td>
</tr>
<tr>
<td>08Dh</td>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>1111 1111 1111 1111</td>
<td>1111 1111 1111 1111</td>
</tr>
<tr>
<td>08Eh</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>08Fh</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>090h</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>091h</td>
<td>PIE1</td>
<td>TMR1IE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>092h</td>
<td>PIE2</td>
<td>OSFIE</td>
<td>C2IE</td>
<td>C1IE</td>
<td>EEIE</td>
<td>BCL1IE</td>
<td>—</td>
<td>—</td>
<td>CCPIE</td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>093h</td>
<td>PIE3(T1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>094h</td>
<td>PIE4(T1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>095h</td>
<td>OPTION_REG</td>
<td>WPUEN</td>
<td>INTEG</td>
<td>TMRC5S</td>
<td>TMRS5E</td>
<td>PSA</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
<td>1111 1111 1111 1111</td>
<td>1111 1111 1111 1111</td>
</tr>
<tr>
<td>096h</td>
<td>PCON</td>
<td>STKOVF</td>
<td>STKUNF</td>
<td>—</td>
<td>—</td>
<td>RMCLR</td>
<td>RI</td>
<td>POR</td>
<td>BOR</td>
<td>00-- 11qq qq-- qquu</td>
<td>00-- 11qq qq-- qquu</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>—</td>
<td>—</td>
</tr>
<tr>
<td>098h</td>
<td>OSCTUNE</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>SPLLEN</td>
<td>IRCF3</td>
<td>IRCF2</td>
<td>IRCF1</td>
<td>IRCF0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>0011 1-00 0011 1--0</td>
<td>0011 1-00 0011 1--0</td>
</tr>
<tr>
<td>09Ah</td>
<td>OSCSTAT</td>
<td>T1OSCR</td>
<td>PLLR</td>
<td>OSTS</td>
<td>HFI0FR</td>
<td>HFI0FL</td>
<td>MI0F</td>
<td>MI0FR</td>
<td>HF10FS</td>
<td>10q0 0gq0 0qqq 0qqq</td>
<td>10q0 0gq0 0qqq 0qqq</td>
</tr>
<tr>
<td>09Bh</td>
<td>ADRESL</td>
<td>A/D Result Register Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxxx xxxxxx xxxxxx</td>
<td>xxxxxx xxxxxx xxxxxx</td>
</tr>
<tr>
<td>09Ch</td>
<td>ADRESH</td>
<td>A/D Result Register High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxxx xxxxxx xxxxxx</td>
<td>xxxxxx xxxxxx xxxxxx</td>
</tr>
<tr>
<td>09Dh</td>
<td>ADCON0</td>
<td>—</td>
<td>CHS4</td>
<td>CHS3</td>
<td>CHS2</td>
<td>CHS1</td>
<td>CHS0</td>
<td>GO/DONE</td>
<td>ADON</td>
<td>-000 0000 -000 0000</td>
<td>-000 0000 -000 0000</td>
</tr>
<tr>
<td>09Eh</td>
<td>ADCON1</td>
<td>ADFM</td>
<td>ADCS2</td>
<td>ADCS1</td>
<td>ADCS0</td>
<td>—</td>
<td>ADREN</td>
<td>ADREF</td>
<td>ADPREF1</td>
<td>ADPREF0</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>09Fh</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.
- Shaded locations are unimplemented, read as '0'.

**Note 1:**
- PIC16F/LF1827 only.
- These registers can be addressed from any bank.
### PIC16F/LF1826/27

#### TABLE 3-8: 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>
</table>

**Bank 2**

100h(2)  INDF0   Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register)  
101h(2)  INDF1   Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register)  
102h(2)  PCL     Program Counter (PC) Least Significant Byte  
103h(2)  STATUS  —       —       —       —       TO       PD       Z       DC       C       --1 1200  ---q quuu  
104h(2)  FSR0L   Indirect Data Memory Address 0 Low Pointer  
105h(2)  FSR0H   Indirect Data Memory Address 0 High Pointer  
106h(2)  FSR1L   Indirect Data Memory Address 1 Low Pointer  
107h(2)  FSR1H   Indirect Data Memory Address 1 High Pointer  
108h(2)  BSR     —       —       —       —       BSR4     BSR3     BSR2     BSR1     BSR0     --0 0000  ---0 0000  
109h(2)  WREG    Working Register  
10Ah(2)  PCLATH  — Write Buffer for the upper 7 bits of the Program Counter  
10Bh(2)  INTCON  GIIE       PEIE       TMROIE       INTE       IOCIE       TMROIF       INTF       IOCIF       0000 000x 0000 000u  
10Ch(2)  LATB    LATB7  LATB6  —       LATA4     LATA3     LATA2     LATA1     LATA0     xx-x xxxx uu-u uu uu  
10Dh(2)  LATB    LATB7  LATB6  LATB5  LATB4  LATB3  LATB2  LATB1  LATB0  xxxx xxxx uu uu uu uu uu  
10Eh     — Unimplemented  
10Fh     — Unimplemented  
110h     — Unimplemented  
111h     — Unimplemented  
112h     CM1CON0  CION      C1OUT     C1OE      C1POL  —       C1SP     C1HYS     C1SYNC     0000 -100 0000 -100  
113h     CM2CON0  C2ON      C2OUT     C2OE      C2POL  —       C2SP     C2HYS     C2SYNC     0000 -100 0000 -100  
114h     CM1CON1  C1INTP    C1INTN    C1PCH1    C1PCH0  —       —       C1CH1     C1CH0     0000 --00 0000 --00  
115h     CM1CON1  C1INTP    C1INTN    C1PCH1    C1PCH0  —       —       C2CH1     C2CH0     0000 --00 0000 --00  
116h     CMOUT    —       —       —       —       —       —       —       MC2OUT    MC1OUT    ----- ----- ----- -----  
117h     BORCON   SBOREN    —       —       —       —       —       —       —       BORRDY    1--- ---q u--- ---u  
118h     FVRCON   FVREN     FVRRDY    Reserved   Reserved   CDAFVR1   CDAFVR0   ADVR0     ADVF0     0gr 0000 0gr 0000  
119h     DACCON0  DACEN     DACLPS    DACOE     ---      DACPSS1   DACPSS0     ---      DACNSS     000- 09-0 090- 00-0  
11Ah     DACCON1  ---       ---       ---      DACR4     DACR3     DACR2     DACR1     DACR0     ---00 0000 -0000 0000  
11Bh     SRCON0   SRELN     SRCLK2    SRCLK1    SRCLK0    SRQEN     SRQEN     SRPS     SRPR     0000 0000 0000 0000  
11Ch     SRCON1   SRSPE     SRSCKE    SRSC2E    SRSC1E    SRRPE     SRRCE     SRRCE2    SRC1E     0000 0000 0000 0000  
11Dh     APFCON0  RXDSEL    SD01SEL   SS1SEL    PZSEL(t)  CCP2SEL(t)  P1DSEL    P1CSEL    CCP1SEL     0000 0000 0000 0000  
11Eh     APFCON1  ---       ---       ---      ---      ---      ---      ---      TXCKSEL    ----00 0----- 0-----0  
11Fh     — Unimplemented  

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

**Note**  
1: PIC16F/LF1827 only.  
2: These registers can be addressed from any bank.
### TABLE 3-8: 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>180h(2)</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>181h(2)</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>182h(2)</td>
<td>PCL</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>183h(2)</td>
<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>---1 1000 ---q quuu</td>
<td></td>
</tr>
<tr>
<td>184h(2)</td>
<td>FSR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>185h(2)</td>
<td>FSR0H</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>186h(2)</td>
<td>FSR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>187h(2)</td>
<td>FSR1H</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>188h(2)</td>
<td>BSR</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>BSR4</td>
<td>BSR3</td>
<td>BSR2</td>
<td>BSR1</td>
<td>BSR0</td>
<td>---0 0000 0000 0000</td>
</tr>
<tr>
<td>189h(2)</td>
<td>WREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>18Ah(2)</td>
<td>PCLATH</td>
<td></td>
<td></td>
<td>Write Buffer for the upper 7 bits of the Program Counter</td>
<td>-000 0000 -000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18Bh(2)</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 000x 0000 000u</td>
<td></td>
</tr>
<tr>
<td>18Ch</td>
<td>ANSELA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ANSA4</td>
<td>ANSA3</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
<td>---1 111   ---1 1111</td>
</tr>
<tr>
<td>18Dh</td>
<td>ANSELB</td>
<td>ANSB7</td>
<td>ANSB6</td>
<td>ANSB5</td>
<td>ANSB4</td>
<td>ANSB3</td>
<td>ANSB2</td>
<td>ANSB1</td>
<td>0000 0000</td>
<td>1111 1111 1111 1111</td>
<td></td>
</tr>
<tr>
<td>18Eh</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>18Fh</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>190h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>191h</td>
<td>EEADRL</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>EEADRH</td>
<td></td>
<td></td>
<td>EEPROM / Program Memory Address Register High Byte</td>
<td>-000 0000 -000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>193h</td>
<td>EEDATL</td>
<td></td>
<td></td>
<td>EEPROM / Program Memory Read Data Register Low Byte</td>
<td>xxxxx xxxxx uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>194h</td>
<td>EEDATH</td>
<td></td>
<td></td>
<td>EEPROM / Program Memory Read Data Register High Byte</td>
<td>---xx xxxx ---uu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>195h</td>
<td>EECON1</td>
<td>EEPOD</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 g000</td>
<td></td>
</tr>
<tr>
<td>196h</td>
<td>EECON2</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>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>198h</td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>199h</td>
<td>RCREG</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>19Ah</td>
<td>TXREG</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>19Bh</td>
<td>SPBRGL</td>
<td>Baud Rate Generator Data Register Low</td>
<td>0000 0000 0000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19Ch</td>
<td>SPBRGH</td>
<td>Baud Rate Generator Data Register High</td>
<td>0000 0000 0000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19Dh</td>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>0000 000x 0000 000x</td>
<td></td>
</tr>
<tr>
<td>19Eh</td>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>0000 0010 0000 0010</td>
<td></td>
</tr>
<tr>
<td>19Fh</td>
<td>BAUDCON</td>
<td>ABDOVF</td>
<td>RCIDL</td>
<td>—</td>
<td></td>
<td>SCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>01-0 0-0-0 01-0 0-0-0</td>
<td></td>
</tr>
</tbody>
</table>

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

**Note 1:** PIC16F/LF1827 only.
- 2: These registers can be addressed from any bank.
### TABLE 3-8: 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>200h(1)</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>201h(1)</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>202h(1)</td>
<td>PCL</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>203h(1)</td>
<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>——1 1200</td>
<td>——q quuu</td>
</tr>
<tr>
<td>204h(1)</td>
<td>FSR0L</td>
<td>Indirect Data Memory Address 0 Low Pointer</td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>205h(1)</td>
<td>FSR0H</td>
<td>Indirect Data Memory Address 0 High Pointer</td>
<td>0000 0000 0000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>206h(1)</td>
<td>FSR1L</td>
<td>Indirect Data Memory Address 1 Low Pointer</td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>207h(1)</td>
<td>FSR1H</td>
<td>Indirect Data Memory Address 1 High Pointer</td>
<td>0000 0000 0000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>208h(1)</td>
<td>BSR</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>BSR4</td>
<td>BSR3</td>
<td>BSR2</td>
<td>BSR1</td>
<td>BSR0</td>
<td>——0 0000</td>
<td>——0 0000</td>
</tr>
<tr>
<td>209h(1)</td>
<td>WREG</td>
<td>Working Register</td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20Ah(1)</td>
<td>PCLATH</td>
<td>Write Buffer for the upper 7 bits of the Program Counter</td>
<td>-000 0000 -000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20Bh(1)</td>
<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>0000 000x 0000 000u</td>
<td></td>
</tr>
<tr>
<td>20Ch</td>
<td>WPUA</td>
<td>——</td>
<td>——</td>
<td>WPUA5</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>-1- ----</td>
<td>-1- ----</td>
</tr>
<tr>
<td>20Dh</td>
<td>WPUA7</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>1111 1111 1111 1111</td>
<td></td>
</tr>
<tr>
<td>20 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>20Fh</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>210h</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>211h</td>
<td>SSP1BUF</td>
<td>Synchronous Serial Port Receive Buffer/Transmit Register</td>
<td>xxxxx xxxxx uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>212h</td>
<td>SSP1ADD</td>
<td>ADD7</td>
<td>ADD6</td>
<td>ADD5</td>
<td>ADD4</td>
<td>ADD3</td>
<td>ADD2</td>
<td>ADD1</td>
<td>ADD0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>213h</td>
<td>SSP1MSK</td>
<td>MSK7</td>
<td>MSK6</td>
<td>MSK5</td>
<td>MSK4</td>
<td>MSK3</td>
<td>MSK2</td>
<td>MSK1</td>
<td>MSK0</td>
<td>1111 1111 1111 1111</td>
<td></td>
</tr>
<tr>
<td>214h</td>
<td>SSP1STAT</td>
<td>SMP</td>
<td>CKE</td>
<td>DÂ</td>
<td>P</td>
<td>S</td>
<td>R/W</td>
<td>UA</td>
<td>BF</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>215h</td>
<td>SSP1CON1</td>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPEN</td>
<td>CKP</td>
<td>SSPM3</td>
<td>SSPM2</td>
<td>SSPM1</td>
<td>SSPM0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>216h</td>
<td>SSP1CON2</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>SSP1CON3</td>
<td>ACKTIM</td>
<td>PCIE</td>
<td>SCIIE</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>——</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>219h</td>
<td>SSP2BUF(1)</td>
<td>Synchronous Serial Port Receive Buffer/Transmit Register</td>
<td>xxxxx xxxxx uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21Ah</td>
<td>SSP2ADD(1)</td>
<td>ADD7</td>
<td>ADD6</td>
<td>ADD5</td>
<td>ADD4</td>
<td>ADD3</td>
<td>ADD2</td>
<td>ADD1</td>
<td>ADD0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>21Bh</td>
<td>SSP2MSK(1)</td>
<td>MSK7</td>
<td>MSK6</td>
<td>MSK5</td>
<td>MSK4</td>
<td>MSK3</td>
<td>MSK2</td>
<td>MSK1</td>
<td>MSK0</td>
<td>1111 1111 1111 1111</td>
<td></td>
</tr>
<tr>
<td>21Ch</td>
<td>SSP2STAT(1)</td>
<td>SMP</td>
<td>CKE</td>
<td>DÂ</td>
<td>P</td>
<td>S</td>
<td>R/W</td>
<td>UA</td>
<td>BF</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>21Dh</td>
<td>SSP2CON1(1)</td>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPEN</td>
<td>CKP</td>
<td>SSPM3</td>
<td>SSPM2</td>
<td>SSPM1</td>
<td>SSPM0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>21 Eh</td>
<td>SSP2CON2(1)</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>21Fh</td>
<td>SSP2CON3(1)</td>
<td>ACKTIM</td>
<td>PCIE</td>
<td>SCIIE</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>
</tbody>
</table>

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

**Note:**
1. PIC16F/LF1827 only.
2. These registers can be addressed from any bank.
### TABLE 3-8: 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>280h(2)</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>281h(2)</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>282h(2)</td>
<td>PCL</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>283h(2)</td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>284h(2)</td>
<td>FSR0L</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>285h(2)</td>
<td>FSR0H</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>286h(2)</td>
<td>FSR1L</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>287h(2)</td>
<td>FSR1H</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>288h(2)</td>
<td>BSR</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>289h(2)</td>
<td>WREG</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>28Ah(2)</td>
<td>PCLATH</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>28Bh(2)</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 0000 0000</td>
<td></td>
</tr>
<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>Unimplemented</td>
</tr>
<tr>
<td>28Dh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>28 Eh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>28Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>28 Gh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
</tr>
<tr>
<td>29h</td>
<td>CCPR1L</td>
<td>Capture/Compare/PWM Register 1 (LSB)</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>CCPR1H</td>
<td>Capture/Compare/PWM Register 1 (MSB)</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>CCP1CON</td>
<td>P1M1</td>
<td>P1M0</td>
<td>DC1B1</td>
<td>DC1B0</td>
<td>CCP1M3</td>
<td>CCP1M2</td>
<td>CCP1M1</td>
<td>CCP1M0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>PWM1CON</td>
<td>P1SEN</td>
<td>P1DC6</td>
<td>P1DC5</td>
<td>P1DC4</td>
<td>P1DC3</td>
<td>P1DC2</td>
<td>P1DC1</td>
<td>P1DC0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>CCPIAS</td>
<td>CCP1ASE</td>
<td>CCP1AS2</td>
<td>CCP1AS1</td>
<td>CCP1AS0</td>
<td>PSS1AC1</td>
<td>PSS1AC0</td>
<td>PSS1BD1</td>
<td>PSS1BD0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>PSTR1CON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STR1SYNC</td>
<td>STR1D</td>
<td>STR1C</td>
<td>STR1B</td>
<td>STR1A</td>
<td>---0 0001 ---0 0001</td>
</tr>
<tr>
<td>29h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>CCPR2L(1)</td>
<td>Capture/Compare/PWM Register 2 (LSB)</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>CCPR2R(1)</td>
<td>Capture/Compare/PWM Register 2 (MSB)</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>PWM2CON(1)</td>
<td>P2M1</td>
<td>P2M0</td>
<td>DC2B1</td>
<td>DC2B0</td>
<td>CCP2M3</td>
<td>CCP2M2</td>
<td>CCP2M1</td>
<td>CCP2M0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>CCPR2CON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STR2SYNC</td>
<td>STR2D</td>
<td>STR2C</td>
<td>STR2B</td>
<td>STR2A</td>
<td>---0 0001 ---0 0001</td>
</tr>
<tr>
<td>29h</td>
<td>PSTR2CON(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td>CCPTMR1S(1)</td>
<td>C4TSEL1</td>
<td>C4TSEL0</td>
<td>C3TSEL1</td>
<td>C3TSEL0</td>
<td>C2TSEL1</td>
<td>C2TSEL0</td>
<td>C1TSEL1</td>
<td>C1TSEL0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>29h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unimplemented</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:**
- PIC16F/LF1827 only.
- These registers can be addressed from any bank.
### Table 3-8: 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>Bank 6</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>3001h</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx XXXXXXXXX</td>
<td></td>
</tr>
<tr>
<td>3002h</td>
<td>PCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>3003h</td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FSR0L</td>
<td></td>
</tr>
<tr>
<td>3004h</td>
<td>FSR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>3005h</td>
<td>FSR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>3006h</td>
<td>FSR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>3007h</td>
<td>FSR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>3008h</td>
<td>BSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>3009h</td>
<td>WREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>300Ah</td>
<td>PCLATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>-000 0000</td>
</tr>
<tr>
<td>300Bh</td>
<td>INTCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
</tr>
<tr>
<td>300Ch</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>300Dh</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>300Eh</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>300Fh</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>3101h</td>
<td>CCPR3L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx XXXXXXXXX</td>
<td></td>
</tr>
<tr>
<td>3102h</td>
<td>CCPR3H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx XXXXXXXXX</td>
<td></td>
</tr>
<tr>
<td>3103h</td>
<td>CCP3CON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-000 0000</td>
<td>-000 0000</td>
</tr>
<tr>
<td>3104h</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>3105h</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>3106h</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>3107h</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>3108h</td>
<td>CCPR4L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx XXXXXXXXX</td>
<td></td>
</tr>
<tr>
<td>3109h</td>
<td>CCPR4H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx XXXXXXXXX</td>
<td></td>
</tr>
<tr>
<td>310Ah</td>
<td>CCP4CON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-000 0000</td>
<td>-000 0000</td>
</tr>
<tr>
<td>310Bh</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>310Ch</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>310Dh</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>310 Eh</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>310Fh</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.  
- Shaded locations are unimplemented, read as ‘0’.

**Note:**  
1. PIC16F/LF1827 only.  
2. These registers can be addressed from any bank.
### TABLE 3-8: 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>380h(2)</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>381h(2)</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>382h(2)</td>
<td>PCL</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>383h(2)</td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ZDC ---1 1000 ---q quuu</td>
<td></td>
</tr>
<tr>
<td>384h(2)</td>
<td>FSR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>385h(2)</td>
<td>FSR0H</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>386h(2)</td>
<td>FSR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>387h(2)</td>
<td>FSR1H</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>388h(2)</td>
<td>BSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>---0 0000 ---0 0000</td>
<td></td>
</tr>
<tr>
<td>389h(2)</td>
<td>WREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>38Ah(2)</td>
<td>PCLATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-000 0000 -000 0000</td>
<td></td>
</tr>
<tr>
<td>38Bh(2)</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 000x 0000 000u</td>
<td></td>
</tr>
<tr>
<td>38Ch</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>38Dh</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>38Eh</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>38Fh</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>390h</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>391h</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>392h</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>393h</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>394h</td>
<td>IOCBP</td>
<td>IOCBP7</td>
<td>IOCBP6</td>
<td>IOCBP5</td>
<td>IOCBP4</td>
<td>IOCBP3</td>
<td>IOCBP2</td>
<td>IOCBP1</td>
<td>IOCBP0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>395h</td>
<td>IOCBN</td>
<td>IOCBN7</td>
<td>IOCBN6</td>
<td>IOCBN5</td>
<td>IOCBN4</td>
<td>IOCBN3</td>
<td>IOCBN2</td>
<td>IOCBN1</td>
<td>IOCBN0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>396h</td>
<td>IOCBF</td>
<td>IOCBF7</td>
<td>IOCBF6</td>
<td>IOCBF5</td>
<td>IOCBF4</td>
<td>IOCBF3</td>
<td>IOCBF2</td>
<td>IOCBF1</td>
<td>IOCBF0</td>
<td>0000 0000 0000 0000</td>
<td></td>
</tr>
<tr>
<td>397h</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>398h</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>399h</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>39Ah</td>
<td>CLKRCN</td>
<td>CLKREN</td>
<td>CLKROE</td>
<td>CLKRSLR</td>
<td>CLKRDC1</td>
<td>CLKRDC0</td>
<td>CLKRDC2</td>
<td>CLKRDC1</td>
<td>CLKRDC0</td>
<td>0011 0000 0011 0000</td>
<td></td>
</tr>
<tr>
<td>39Bh</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>39Ch</td>
<td>MDCON</td>
<td>MDEN</td>
<td>MDOE</td>
<td>MDSLR</td>
<td>MDOPOL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MDDBIT</td>
<td>0010 ---0 0010 ---0</td>
<td></td>
</tr>
<tr>
<td>39Dh</td>
<td>MDSRC</td>
<td>MDSMODIS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MDMS3</td>
<td>MDMS2</td>
<td>MDMS1</td>
<td>MDMS0</td>
<td>x--- xxxx u--- uuuu</td>
<td></td>
</tr>
<tr>
<td>39Eh</td>
<td>MDCARL</td>
<td>MDCLODIS</td>
<td>MDCLPOL</td>
<td>MDCLSYNC</td>
<td>—</td>
<td>MDCL3</td>
<td>MDCL2</td>
<td>MDCL1</td>
<td>MDCL0</td>
<td>xxxx- xxxx uu- uu- uu- uu</td>
<td></td>
</tr>
<tr>
<td>39Fh</td>
<td>MDCARH</td>
<td>MDCOHDIS</td>
<td>MDCHPOL</td>
<td>MDCHSYNC</td>
<td>—</td>
<td>MDCH3</td>
<td>MDCH2</td>
<td>MDCH1</td>
<td>MDCH0</td>
<td>xxxx- xxxx uu- uu- uu</td>
<td></td>
</tr>
</tbody>
</table>

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

**Note 1:** PIC16F/LF1827 only.
2: These registers can be addressed from any bank.
### TABLE 3-8: 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>Bank 8</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>400h(2)</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx</td>
<td>Addressing this location uses contents of FSR0H/FSR0L to address data memory ( \text{(not a physical register)} )</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>401h(2)</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx</td>
<td>Addressing this location uses contents of FSR1H/FSR1L to address data memory ( \text{(not a physical register)} )</td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
</tr>
<tr>
<td>402h(2)</td>
<td>PCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000</td>
<td>Program Counter (PC) Least Significant Byte</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>403h(2)</td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>____</td>
<td>---1 1200 ---q quuu</td>
<td></td>
</tr>
<tr>
<td>404h(2)</td>
<td>FSR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000</td>
<td>Indirect Data Memory Address 0 Low Pointer</td>
<td>uuuu uuuu uuuu uuuu</td>
</tr>
<tr>
<td>405h(2)</td>
<td>FSR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000</td>
<td>Indirect Data Memory Address 0 High Pointer</td>
<td>uuuu uuuu uuuu uuuu</td>
</tr>
<tr>
<td>406h(2)</td>
<td>FSR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000</td>
<td>Indirect Data Memory Address 1 Low Pointer</td>
<td>uuuu uuuu uuuu uuuu</td>
</tr>
<tr>
<td>407h(2)</td>
<td>FSR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000</td>
<td>Indirect Data Memory Address 1 High Pointer</td>
<td>uuuu uuuu uuuu uuuu</td>
</tr>
<tr>
<td>408h(2)</td>
<td>BSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>____</td>
<td>---0 0000 ---0 0000</td>
<td></td>
</tr>
<tr>
<td>409h(2)</td>
<td>WREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000</td>
<td>Working Register</td>
<td>uuuu uuuu uuuu uuuu</td>
</tr>
<tr>
<td>40Ah(2)</td>
<td>PCLATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-000</td>
<td>Write Buffer for the upper 7 bits of the Program Counter</td>
<td>-000 0000 -000 0000</td>
</tr>
<tr>
<td>40Bh(2)</td>
<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>0000 000x 0000 000u</td>
<td></td>
</tr>
<tr>
<td>40Ch</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>40Dh</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>40 Eh</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>40Fh</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>410h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>411h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>412h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>413h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>414h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>415h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>416h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>417h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>418h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>419h</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>41Ah</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>41Bh</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>41Ch</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>41Dh</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
<tr>
<td>41 Eh</td>
<td></td>
<td>—</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>Unimplemented</td>
</tr>
</tbody>
</table>

**Legend:** \( x = \text{unknown}, \ u = \text{unchanged}, \ q = \text{value depends on condition}, \ - = \text{unimplemented}, \ r = \text{reserved}. \)

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

**Note**

1: PIC16F/LF1827 only.

2: These registers can be addressed from any bank.
### TABLE 3-8: 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:</th>
<th>Value on all other resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>x00h/</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td></td>
</tr>
<tr>
<td>x08h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxxxxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x00h/</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td></td>
</tr>
<tr>
<td>x81h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxxxxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x02h/</td>
<td>PCL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td></td>
</tr>
<tr>
<td>x82h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x03h/</td>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td>---1 1000 -q quuu</td>
</tr>
<tr>
<td>x83h</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>x04h/</td>
<td>FSR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td>0000 0000 uuuu uuuu</td>
</tr>
<tr>
<td>x84h</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>x05h/</td>
<td>FSR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>x85h</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>x06h/</td>
<td>FSR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td>0000 0000 uuuu uuuu</td>
</tr>
<tr>
<td>x86h</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>x07h/</td>
<td>FSR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>x87h</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>x08h/</td>
<td>BSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td>---0 0000 -0 0000</td>
</tr>
<tr>
<td>x88h</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>x09h/</td>
<td>WREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td>0000 0000 uuuu uuuu</td>
</tr>
<tr>
<td>x89h</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>x0Ah/</td>
<td>PCLATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td>-000 0000 -000 0000</td>
</tr>
<tr>
<td>x8Ah</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>x0Bh/</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>IO CIF</td>
<td>POR, BOR</td>
<td>0000 0000x 0000 0000x</td>
</tr>
<tr>
<td>x8Bh</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/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POR, BOR</td>
<td>—</td>
</tr>
<tr>
<td>x8Ch</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>POR, BOR</td>
<td>—</td>
</tr>
<tr>
<td>x3Fh</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:**  
- ≈ unknown  
- u = unchanged  
- q = value depends on condition  
- - = unimplemented  
- r = reserved  

**Note:**  
1. PIC16F/LF1827 only.  
2. These registers can be addressed from any bank.
### TABLE 3-8: 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>F80h(2)</td>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>F81h(2)</td>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxxxx xxxxx xxxxx xxxxx</td>
<td></td>
</tr>
<tr>
<td>F82h(2)</td>
<td>PCL</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>F83h(2)</td>
<td>STATUS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TO</td>
<td>PD</td>
<td>Z</td>
<td>DC</td>
<td>C --1 1200 --q quuu</td>
<td></td>
</tr>
<tr>
<td>F84h(2)</td>
<td>FSR0L</td>
<td>Indirect Data Memory Address 0 Low Pointer</td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F85h(2)</td>
<td>FSR0H</td>
<td>Indirect Data Memory Address 0 High Pointer</td>
<td>0000 0000 0000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F86h(2)</td>
<td>FSR1L</td>
<td>Indirect Data Memory Address 1 Low Pointer</td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F87h(2)</td>
<td>FSR1H</td>
<td>Indirect Data Memory Address 1 High Pointer</td>
<td>0000 0000 0000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F88h(2)</td>
<td>BSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BSR4</td>
<td>BSR3</td>
<td>BSR2</td>
<td>BSR1</td>
<td>BSR0 --0 0000 --0 0000</td>
<td></td>
</tr>
<tr>
<td>F89h(2)</td>
<td>WREG</td>
<td>Working Register</td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F8Ah(2)</td>
<td>PCLATH</td>
<td>Write Buffer for the upper 7 bits of the Program Counter</td>
<td>-000 0000 -000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F8Bh(2)</td>
<td>INTCON</td>
<td>GIE, PEIE, TMR0IE, INTE, IOCIE, TMR0IF, INTF, IOCIF</td>
<td>0000 000x 0000 000u</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>Unimplemented</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_</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Z DC C ---- -xxxx ---- -uuu</td>
<td></td>
</tr>
<tr>
<td>FE5h</td>
<td>WREG_</td>
<td>Working Register Shadow</td>
<td>0000 0000 uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE6h</td>
<td>BSR_</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Bank Select Register Shadow --x xxxx --u uuuu</td>
<td></td>
</tr>
<tr>
<td>FE7h</td>
<td>PCLATH_</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Program Counter Latch High Register Shadow --xxx xxxx uuuu uuuu</td>
<td></td>
</tr>
<tr>
<td>FE8h</td>
<td>FSR0L_</td>
<td>Indirect Data Memory Address 0 Low Pointer Shadow</td>
<td>xxxxx xxxxx uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FE9h</td>
<td>FSR0H_</td>
<td>Indirect Data Memory Address 0 High Pointer Shadow</td>
<td>xxxxx xxxxx uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEAh</td>
<td>FSR1L_</td>
<td>Indirect Data Memory Address 1 Low Pointer Shadow</td>
<td>xxxxx xxxxx uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEBh</td>
<td>FSR1H_</td>
<td>Indirect Data Memory Address 1 High Pointer Shadow</td>
<td>xxxxx xxxxx uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FECh</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>FEDh</td>
<td>STKPTR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Current Stack pointer --1 1111 --1 1111</td>
<td></td>
</tr>
<tr>
<td>FEEd</td>
<td>TOSL</td>
<td>Top-of-Stack Low byte</td>
<td>xxxxx xxxxx uuuu uuuu</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEfh</td>
<td>TOSH</td>
<td>Top-of-Stack High byte</td>
<td>-xxx xxxx -uuu uuuu</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. PIC16F/LF1827 only.
2. These registers can be addressed from any bank.
3.3 PCL and PCLATH

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

FIGURE 3-4: LOADING OF PC IN DIFFERENT SITUATIONS

3.3.1 MODIFYING PCL

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

3.3.2 COMPUTED GOTO

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

3.3.3 COMPUTED FUNCTION CALLS

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

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

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

3.3.4 BRANCHING

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

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

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

All devices have a 16-level x 15-bit wide hardware stack (refer to Figures 3-5 through 3-8). 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 = 0 (Configuration Word 2). This means that after the stack has been PUSHed sixteen times, the seventeenth PUSH overwrites the value that was stored from the first PUSH. The eighteenth PUSH overwrites the second PUSH (and so on). The STKOVF and STKUNF flag bits will be set on an Overflow/Underflow, regardless of whether the Reset is enabled.

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

3.4.1 ACCESSING THE STACK

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

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

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

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

FIGURE 3-5: ACCESSING THE STACK EXAMPLE 1

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.

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.

Stack Reset Disabled (STVREN = 0)

Stack Reset Enabled (STVREN = 1)
FIGURE 3-6: 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 3-7: 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.
3.4.2 OVERFLOW/UNDERFLOW RESET
If the STVREN bit in Configuration Word 2 is set 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.

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

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

- Traditional Data Memory
- Linear Data Memory
- Program Flash Memory
FIGURE 3-9: INDIRECT ADDRESSING

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

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

FIGURE 3-10: TRADITIONAL DATA MEMORY MAP

[Diagram showing direct and indirect addressing with banks and locations]
3.5.2 LINEAR DATA MEMORY

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

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

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

FIGURE 3-11: LINEAR DATA MEMORY MAP

3.5.3 PROGRAM FLASH MEMORY

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

FIGURE 3-12: PROGRAM FLASH MEMORY MAP
4.0 DEVICE CONFIGURATION

Device Configuration consists of Configuration Word 1 and Configuration Word 2, 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.
REGISTER 4-1: CONFIGURATION WORD 1

<table>
<thead>
<tr>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
<th>R/P-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCMEN</td>
<td>IESO</td>
<td>CLKOUTEN</td>
<td>BOREN1</td>
<td>BOREN0</td>
<td>CPD</td>
<td>CP</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 13  FCMEN: Fail-Safe Clock Monitor Enable bit
        1 = Fail-Safe Clock Monitor is enabled
        0 = Fail-Safe Clock Monitor is disabled

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

bit 11  CLKOUTEN: Clock Out Enable bit
        This bit is ignored, CLKOUT function is disabled. Oscillator function on the CLKOUT pin.
        All other FOSC modes:
        1 = CLKOUT function is disabled. I/O function on the CLKOUT pin
        0 = CLKOUT function is enabled on the CLKOUT pin

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

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

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

bit 6    MCLRE: RA5/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
         WPUA register.

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

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

Note 1: Enabling Brown-out Reset does not automatically enable Power-up Timer.
2: The entire data EEPROM will be erased when the code protection is turned off during an erase.
3: The entire program memory will be erased when the code protection is turned off.
REGISTER 4-1:  CONFIGURATION WORD 1 (CONTINUED)

bit 2-0  \textbf{FOSC}<2:0>: Oscillator Selection bits

111 = ECH: External Clock, High-Power mode (4-32 MHz): device clock supplied to CLKIN pin
110 = ECM: External Clock, Medium-Power mode (0.5-4 MHz): device clock supplied to CLKIN pin
101 = ECL: External Clock, Low-Power mode (0-0.5 MHz): device clock supplied to CLKIN pin
100 = INTOSC oscillator: I/O function on CLKIN pin
011 = EXTRC oscillator: External RC circuit connected to CLKIN pin
010 = HS oscillator: High-speed crystal/resonator connected between OSC1 and OSC2 pins
001 = XT oscillator: Crystal/resonator connected between OSC1 and OSC2 pins
000 = LP oscillator: Low-power crystal connected between OSC1 and OSC2 pins

\textbf{Note} 1: Enabling Brown-out Reset does not automatically enable Power-up Timer.
2: The entire data EEPROM will be erased when the code protection is turned off during an erase.
3: The entire program memory will be erased when the code protection is turned off.
### REGISTER 4-2: CONFIGURATION WORD 2

<table>
<thead>
<tr>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7-5</th>
<th>Bit 4</th>
<th>Bit 3-2</th>
<th>Bit 1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVP</td>
<td>DEBUG</td>
<td>—</td>
<td>BORV</td>
<td>STVREN</td>
<td>PLLEN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WRT1</td>
</tr>
<tr>
<td>U-1</td>
<td>U-1</td>
<td>R/P-1/1</td>
<td>U-1</td>
<td>U-1</td>
<td>R/P-1/1</td>
<td>R/P-1/1</td>
<td>U-1</td>
<td>R/P-1/1</td>
<td>WRT0</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 13**
- **LVP**: Low-Voltage Programming Enable bit<sup>(1)</sup>
  - 1 = Low-voltage programming enabled
  - 0 = High-voltage on MCLR must be used for programming

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

**bit 11**
- **Unimplemented**: Read as ‘1’

**bit 10**
- **BORV**: Brown-out Reset Voltage Selection bit
  - 1 = Brown-out Reset voltage set to 1.9V (typical)
  - 0 = Brown-out Reset voltage set to 2.5V (typical)

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

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

**bit 7-5**
- **Unimplemented**: Read as ‘1’

**bit 4**
- **Reserved**: This location should be programmed to a ‘1’.

**bit 3-2**
- **Unimplemented**: Read as ‘1’

**bit 1-0**
- **WRT<1:0>**: Flash Memory Self-Write Protection bits
  - 2 kW Flash memory (PIC16F/LF1826 only):
    - 11 = Write protection off
    - 10 = 000h to 1FFh write-protected, 200h to 7FFh may be modified by EECON control
    - 01 = 000h to 3FFh write-protected, 400h to 7FFh may be modified by EECON control
    - 00 = 000h to 7FFh write-protected, no addresses may be modified by EECON control
  - 4 kW Flash memory (PIC16F/LF1827 only):
    - 11 = Write protection off
    - 10 = 000h to 1FFh write-protected, 200h to FFFh may be modified by EECON control
    - 01 = 000h to 3FFh write-protected, 800h to FFFh may be modified by EECON control
    - 00 = 000h to FFFh write-protected, no addresses may be modified by EECON control

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

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

4.2.1 PROGRAM MEMORY PROTECTION

The entire program memory space is protected from external reads and writes by the CP bit in Configuration Word 1. When \( CP = 0 \), external reads and writes of program memory (0000h-7FFFh) 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 4.3 “Write Protection” for more information.

4.2.2 DATA EEPROM PROTECTION

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

4.3 Write Protection

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

The WRT<1:0> bits in Configuration Word 2 define the size of the program memory block that is protected.

4.4 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 11.4 “Configuration Word and Device ID Access” for more information on accessing these memory locations. For more information on checksum calculation, see the “PIC16F/LF1826/27 Memory Programming Specification” (DS41390).
4.5 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 11.4 “Configuration Word and Device ID Access” for more information on accessing these memory locations.

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

REGISTER 4-3: DEVICEID: DEVICE ID REGISTER\(^{(1)}\)

<table>
<thead>
<tr>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV8</td>
<td>DEV7</td>
<td>DEV6</td>
<td>DEV5</td>
<td>DEV4</td>
<td>DEV3</td>
<td>DEV2</td>
</tr>
</tbody>
</table>

bit 13

<table>
<thead>
<tr>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
<th>R/P-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV1</td>
<td>DEV0</td>
<td>REV4</td>
<td>REV3</td>
<td>REV2</td>
<td>REV1</td>
<td>REV0</td>
<td></td>
</tr>
</tbody>
</table>

bit 6

Legend:
P = Programmable bit
R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
’0’ = Bit is cleared
‘1’ = Bit is set
-n = Value at POR
x = Bit is unknown

bit 13-5 \(\text{DEV<8:0>}:\) Device ID bits

- 100111100 = PIC16F1826
- 100111101 = PIC16F1827
- 101000100 = PIC16LF1826
- 101000101 = PIC16LF1827

bit 4-0 \(\text{REV<4:0>}:\) Revision ID bits

These bits are used to identify the revision.

Note 1: This location cannot be written.
5.0 OSCILLATOR MODULE (WITH FAIL-SAFE CLOCK MONITOR)

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

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

The oscillator module can be configured in one of six clock modes.
1. EC – External clock.
2. LP – 32 kHz Low-Power Crystal mode.
3. XT – Medium Gain Crystal or Ceramic Resonator Oscillator mode.
4. HS – High Gain Crystal or Ceramic Resonator mode.
5. RC – External Resistor-Capacitor (RC).
6. INTOSC – Internal oscillator.

Clock Source modes are selected by the FOSC<2:0> bits in the Configuration Word 1. The FOSC bits determine the type of oscillator that will be used when the device is first powered.

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

The INTOSC internal oscillator block produces low, medium, and high frequency clock sources, designated LFINTOSC, MFINTOSC, and HFINTOSC. (see Internal Oscillator Block, Figure 5-1). A wide selection of device clock frequencies may be derived from these three 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), quartz crystal resonators or ceramic resonators (LP, XT and HS modes) and Resistor-Capacitor (RC) mode circuits.

Internal clock sources are contained internally within the oscillator module. The internal oscillator block has two internal oscillators and a dedicated phase-locked-loop (HFPLL) that are used to generate three internal system clock sources: the 16 MHz High-Frequency Internal Oscillator (HFINTOSC), 500 kHz (MFINTOSC) and the 31 kHz Low-Frequency Internal Oscillator (LFINTOSC).

The system clock can be selected between external or internal clock sources via the System Clock Select (SCS) bits in the OSCCON register. See Section 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<2:0> bits in the Configuration Word 1 to select an external clock source that will be used as the default system clock upon a device Reset.
• Write the SCS<1:0> bits in the OSCCON register to switch the system clock source to:
  - Timer1 Oscillator during run-time, or
  - An external clock source determined by the value of the FOSC bits.

See Section 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 OSC1 input. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. Figure 5-2 shows the pin connections for EC mode.

EC mode has 3 power modes to select from through Configuration Word 1:

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

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

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

5.2.1.2 LP, XT, HS Modes

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

LP Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is designed to drive only 32.768 kHz tuning-fork type crystals (watch crystals).

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

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

Figure 5-3 and Figure 5-4 show typical circuits for quartz crystal and ceramic resonators, respectively.
5.2.1.3 Oscillator Start-up Timer (OST)

If the oscillator module is configured for LP, XT or HS modes, the Oscillator Start-up Timer (OST) counts 1024 oscillations from OSC1. This occurs following a Power-on Reset (POR) and when the Power-up Timer (PWRT) has expired (if configured), or a wake-up from Sleep. During this time, the program counter does not increment and program execution is suspended. The OST ensures that the oscillator circuit, using a quartz crystal resonator or ceramic resonator, has started and is providing a stable system clock to the oscillator module.

In order to minimize latency between external oscillator start-up and code execution, the Two-Speed Clock Start-up mode can be selected (see Section 5.4 “Two-Speed Clock Start-up Mode”).

5.2.1.4 4X PLL

The oscillator module contains a 4X PLL that can be used with both external and internal clock sources to provide a system clock source. The input frequency for the 4X PLL must fall within specifications. See the PLL Clock Timing Specifications in Section 29.0 “Electrical Specifications”.

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

1. Program the PLLEN bit in Configuration Word 2 to a ‘1’.
2. Write the SPLLEN bit in the OSCCON register to a ‘1’. If the PLLEN bit in Configuration Word 2 is programmed to a ‘1’, then the value of SPLLEN is ignored.
5.2.1.5 **TIMER1 Oscillator**

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

The Timer1 Oscillator can be used as an alternate system clock source and can be selected during run-time using clock switching. Refer to Section 5.3 “Clock Switching” for more information.

5.2.1.6 **External RC Mode**

The external Resistor-Capacitor (RC) modes support the use of an external RC circuit. This allows the designer maximum flexibility in frequency choice while keeping costs to a minimum when clock accuracy is not required.

The RC circuit connects to OSC1. OSC2/CLKOUT is available for general purpose I/O. The function of the OSC2/CLKOUT pin is determined by the state of the CLKOUTEN bit in Configuration Word 1. Figure 5-5 shows the external RC mode connections.

![FIGURE 5-5: EXTERNAL RC MODES](image)

The RC oscillator frequency is a function of the supply voltage, the resistor ($R_{EXT}$) and capacitor ($C_{EXT}$) values and the operating temperature. Other factors affecting the oscillator frequency are:

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

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

5.2.2 **INTERNAL CLOCK SOURCES**

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

- Program the FOSC<2:0> bits in Configuration Word 1 to select the INTOSC clock source, which will be used as the default system clock upon a device Reset.
- Write the SCS<1:0> bits in the OSCCON register to switch the system clock source to the internal oscillator during run-time. See Section 5.3 “Clock Switching” for more information.

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

The function of the OSC2/CLKOUT pin is determined by the state of the CLKOUTEN bit in Configuration Word 1.

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

1. The **HFINTOSC** (High-Frequency Internal Oscillator) is factory calibrated and operates at 16 MHz. The HFINTOSC source is generated from the 500 kHz MFINTOSC source and the dedicated phase-locked-loop, HFPLL. The frequency of the HFINTOSC can be user-adjusted via software using the OSCTUNE register (Register 5-3).

2. The **MFINTOSC** (Medium-Frequency Internal Oscillator) is factory calibrated and operates at 500 kHz. The frequency of the MFINTOSC can be user-adjusted via software using the OSCTUNE register (Register 5-3).

3. 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 frequency of the HFINTOSC can be altered via software using the OSCTUNE register (Register 5-3).
The output of the HFINTOSC connects to a postscaler and multiplexer (see Figure 5-1). One of nine frequencies derived from the HFINTOSC can be selected via software using the IRCF<3:0> bits of the OSCCON register. See Section 5.2.2.7 “Internal Oscillator Clock Switch Timing” for more information.

The HFINTOSC is enabled by:
• Configure the IRCF<3:0> bits of the OSCCON register for the desired HF frequency, and
• FOSC<2:0> = 100, or
• Set the System Clock Source (SCS) bits of the OSCCON register to ‘1x’.

The High Frequency Internal Oscillator Ready bit (HFIOFR) of the OSCSTAT register indicates when the HFINTOSC is running and can be utilized.
The High Frequency Internal Oscillator Status Locked bit (HFIOFL) of the OSCSTAT register indicates when the HFINTOSC is running within 2% of its final value.
The High Frequency Internal Oscillator Status Stable bit (HFIOFS) of the OSCSTAT register indicates when the HFINTOSC is running within 0.5% of its final value.

5.2.2.2 MFINTOSC
The Medium-Frequency Internal Oscillator (MFINTOSC) is a factory calibrated 500 kHz internal clock source. The frequency of the MFINTOSC can be altered via software using the OSCTUNE register (Register 5-3).
The output of the MFINTOSC connects to a postscaler and multiplexer (see Figure 5-1). One of nine frequencies derived from the MFINTOSC can be selected via software using the IRCF<3:0> bits of the OSCCON register. See Section 5.2.2.7 “Internal Oscillator Clock Switch Timing” for more information.

The MFINTOSC is enabled by:
• Configure the IRCF<3:0> bits of the OSCCON register for the desired LF frequency, and
• FOSC<2:0> = 100, or
• Set the System Clock Source (SCS) bits of the OSCCON register to ‘1x’.

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

5.2.2.3 Internal Oscillator Frequency Adjustment
The 500 kHz internal oscillator is factory calibrated. This internal oscillator can be adjusted in software by writing to the OSCTUNE register (Register 5-3). Since the HFINTOSC and MFINTOSC clock sources are derived from the 500 kHz internal oscillator a change in the OSCTUNE register value will apply to both.
The default value of the OSCTUNE register is ‘0’. The value is a 5-bit two’s complement number. A value of 0Fh will provide an adjustment to the maximum frequency. A value of 10h will provide an adjustment to the minimum frequency.

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

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

5.2.2.4 LFINTOSC
The Low-Frequency Internal Oscillator (LFINTOSC) is an uncalibrated 31 kHz internal clock source.
The output of the LFINTOSC connects to a postscaler and multiplexer (see Figure 5-1). Select 31 kHz, via software, using the IRCF<3:0> bits of the OSCCON register. See Section 5.2.2.7 “Internal Oscillator Clock Switch Timing” for more information. The LFINTOSC is also the frequency for the Power-up Timer (PWRT), Watchdog Timer (WDT) and Fail-Safe Clock Monitor (FSCM).
The LFINTOSC is enabled by selecting 31 kHz (IRCF<3:0> bits of the OSCCON register = 000) as the system clock source (SCS bits of the OSCCON register = 1x), or when any of the following are enabled:
• Configure the IRCF<3:0> bits of the OSCCON register for the desired LF frequency, and
• FOSC<2:0> = 100, or
• Set the System Clock Source (SCS) bits of the OSCCON register to ‘1x’

Peripherals that use the LFINTOSC are:
• Power-up Timer (PWRT)
• Watchdog Timer (WDT)
• Fail-Safe Clock Monitor (FSCM)
The Low Frequency Internal Oscillator Ready bit (LFIOFR) of the OSCSTAT register indicates when the LFINTOSC is running and can be utilized.
5.2.2.5 Internal Oscillator Frequency Selection

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

The output of the 16 MHz HFINTOSC and 31 kHz LFINTOSC connects to a postscaler and multiplexer (see Figure 5-1). The Internal Oscillator Frequency Select bits IRCF<3:0> of the OSCCON register select the frequency output of the internal oscillators. One of the following frequencies can be selected via software:

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

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

5.2.2.6 32 MHz Internal Oscillator Frequency Selection

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

- The FOSC bits in Configuration Word 1 must be set to use the INTOSC source as the device system clock (FOSC<2:0> = 100).
- The IRCF bits in the OSCCON register must be set to the 8 MHz HFINTOSC selection (IRCF<3:0> = 1110).
- The SPLLEN bit in the OSCCON register must be set to enable the 4X PLL.

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

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

5.2.2.7 Internal Oscillator Clock Switch Timing

When switching between the HFINTOSC, MFINTOSC and the LFINTOSC, the new oscillator may already be shut down to save power (see Figure 5-6). If this is the case, there is a delay after the IRCF<3:0> bits of the OSCCON register are modified before the frequency selection takes place. The OSCSTAT register will reflect the current active status of the HFINTOSC, MFINTOSC and LFINTOSC oscillators. The sequence of a frequency selection is as follows:

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

See Figure 5-6 for more details.

If the internal oscillator speed is switched between two clocks of the same source, there is no start-up delay before the new frequency is selected. Clock switching time delays are shown in Table 5-1.

Start-up delay specifications are located in the oscillator tables of Section 29.0 “Electrical Specifications”.

Note: The 4X PLL may also be enabled for use with the Internal Oscillator Block by programming the PLLEN bit in Configuration Word 2 to a ‘1’. However, the 4X PLL cannot be disabled by software and the 8 MHz HFINTOSC option will no longer be available.
FIGURE 5-6: INTERNAL OSCILLATOR SWITCH TIMING

HFINTOSC / MFINTOSC

HFINTOSC / LFINTOSC (FSCM and WDT disabled)

HFINTOSC / MFINTOSC

HFINTOSC

Start-up Time

2-cycle Sync

Running

IRCF <3:0>

≠ 0

= 0

System Clock

HFINTOSC / LFINTOSC (Either FSCM or WDT enabled)

HFINTOSC / MFINTOSC

HFINTOSC

2-cycle Sync

Running

IRCF <3:0>

≠ 0

= 0

System Clock

LFINTOSC → HFINTOSC/MFINTOSC

LFINTOSC

Start-up Time

2-cycle Sync

Running

HFINTOSC / MFINTOSC

IRCF <3:0>

= 0

≠ 0

System Clock

LFINTOSC turns off unless WDT or FSCM is enabled
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 Word 1
- Timer1 32 kHz crystal oscillator
- 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<2:0> bits in the Configuration Word 1.
- When the SCS bits of the OSCCON register = 01, the system clock source is the Timer1 oscillator.
- When the SCS bits of the OSCCON register = 1x, the system clock source is chosen by the internal oscillator frequency selected by the IRCF<3:0> bits of the OSCCON register. After a Reset, the SCS bits of the OSCCON register are always cleared.

Note: Any automatic clock switch, which may occur from Two-Speed Start-up or Fail-Safe Clock Monitor, does not update the SCS bits of the OSCCON register. The user can monitor the OSTS bit of the OSCSTAT register to determine the current system clock source.

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

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

The Oscillator Start-up Time-out Status (OSTS) bit of the OSCSTAT register indicates whether the system clock is running from the external clock source, as defined by the FOSC<2:0> bits in the Configuration Word 1, or from the internal clock source. In particular, OSTS indicates that the Oscillator Start-up Timer (OST) has timed out for LP, XT or HS modes. The OST does not reflect the status of the Timer1 Oscillator.

5.3.3 TIMER1 OSCILLATOR

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

The Timer1 oscillator is enabled using the T1OSCEN control bit in the T1CON register. See Section 20.0 “Timer1 Module with Gate Control” for more information about the Timer1 peripheral.

5.3.4 TIMER1 OSCILLATOR READY (T1OSCR) BIT

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

Note: Any automatic clock switch, which may occur from Two-Speed Start-up or Fail-Safe Clock Monitor, does not update the SCS bits of the OSCCON register. The user can monitor the OSTS bit of the OSCSTAT register to determine the current system clock source.
5.4 Two-Speed Clock Start-up Mode

Two-Speed Start-up mode provides additional power savings by minimizing the latency between external oscillator start-up and code execution. In applications that make heavy use of the Sleep mode, Two-Speed Start-up will remove the external oscillator start-up time from the time spent awake and can reduce the overall power consumption of the device. This mode allows the application to wake-up from Sleep, perform a few instructions using the INTOSC internal oscillator block as the clock source and go back to Sleep without waiting for the external oscillator to become stable.

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

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

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

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

5.4.1 TWO-SPEED START-UP MODE CONFIGURATION

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

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

Two-Speed Start-up mode is entered after:

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

### TABLE 5-1: OSCILLATOR SWITCHING DELAYS

<table>
<thead>
<tr>
<th>Switch From</th>
<th>Switch To</th>
<th>Frequency</th>
<th>Oscillator Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep/POR</td>
<td>LFINTOSC&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>31 kHz</td>
<td>Oscillator Warm-up Delay (TWARM)</td>
</tr>
<tr>
<td></td>
<td>MFINTOSC&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>31.25 kHz-500 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HFINTOSC&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>31.25 kHz-16 MHz</td>
<td></td>
</tr>
<tr>
<td>Sleep/POR</td>
<td>EC, RC&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>DC – 32 MHz</td>
<td>2 cycles</td>
</tr>
<tr>
<td>LFINTOSC</td>
<td>EC, RC&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>DC – 32 MHz</td>
<td>1 cycle of each</td>
</tr>
<tr>
<td>Sleep/POR</td>
<td>Timer1 Oscillator</td>
<td>32 kHz-20 MHz</td>
<td>1024 Clock Cycles (OST)</td>
</tr>
<tr>
<td>LP, XT, HS&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Any clock source</td>
<td>MFINTOSC&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>31.25 kHz-500 kHz</td>
<td>2 µs (approx.)</td>
</tr>
<tr>
<td></td>
<td>HFINTOSC&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>31.25 kHz-16 MHz</td>
<td></td>
</tr>
<tr>
<td>Any clock source</td>
<td>LFINTOSC&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>31 kHz</td>
<td>1 cycle of each</td>
</tr>
<tr>
<td>Any clock source</td>
<td>Timer1 Oscillator</td>
<td>32 kHz</td>
<td>1024 Clock Cycles (OST)</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.
5.4.2 TWO-SPEED START-UP SEQUENCE

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

5.4.3 CHECKING TWO-SPEED CLOCK STATUS

Checking the state of the OSTS bit of the OSCSTAT register will confirm if the microcontroller is running from the external clock source, as defined by the FOSC<2:0> bits in the Configuration Word 1, or the internal oscillator.

FIGURE 5-7: TWO-SPEED START-UP

[Diagram showing the two-speed start-up sequence with INTOSC, OSC1, OSC2, Program Counter, and System Clock phases labeled.]
5.5 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM can detect oscillator failure any time after the Oscillator Start-up Timer (OST) has expired. The FSCM is enabled by setting the FCMEN bit in the Configuration Word 1. The FSCM is applicable to all external Oscillator modes (LP, XT, HS, EC, Timer1 Oscillator and RC).

FIGURE 5-8: FSCM BLOCK DIAGRAM

5.5.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64. See Figure 5-8. Inside the fail detector block is a latch. The external clock sets the latch on each falling edge of the external clock. The sample clock clears the latch on each rising edge of the sample clock. A failure is detected when an entire half-cycle of the sample clock elapses before the external clock goes low.

5.5.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM switches the device clock to an internal clock source and sets the bit flag OSFIF of the PIR2 register. Setting this flag will generate an interrupt if the OSFIE bit of the PIE2 register is also set. The device firmware can then take steps to mitigate the problems that may arise from a failed clock. The system clock will continue to be sourced from the internal clock source until the device firmware successfully restarts the external oscillator and switches back to external operation.

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

5.5.3 FAIL-SAFE CONDITION CLEARING

The Fail-Safe condition is cleared after a Reset, executing a SLEEP instruction or changing the SCS bits of the OSCCON register. When the SCS bits are changed, the OST is restarted. While the OST is running, the device continues to operate from the INTOSC selected in OSCCON. When the OST times out, the Fail-Safe condition is cleared and the device will be operating from the external clock source. The Fail-Safe condition must be cleared before the OSFIF flag can be cleared.

5.5.4 RESET OR WAKE-UP FROM SLEEP

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

Note: Due to the wide range of oscillator start-up times, the Fail-Safe circuit is not active during oscillator start-up (i.e., after exiting Reset or Sleep). After an appropriate amount of time, the user should check the Status bits in the OSCSTAT register to verify the oscillator start-up and that the system clock switchover has successfully completed.
Note: The system clock is normally at a much higher frequency than the sample clock. The relative frequencies in this example have been chosen for clarity.
5.6 Oscillator Control Registers

REGISTER 5-1: OSCCON: OSCILLATOR CONTROL REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>R/W-1/1</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPLLEN</td>
<td>IRCF3</td>
<td>IRCF2</td>
<td>IRCF1</td>
<td>IRCF0</td>
<td>—</td>
<td>SCS1</td>
<td>SCS0</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 SPLLEN: Software PLL Enable bit
If PLLEN in Configuration Word 1 = 1:
SPLLEN bit is ignored. 4x PLL is always enabled (subject to oscillator requirements)
If PLLEN in Configuration Word 1 = 0:
1 = 4x PLL is enabled
0 = 4x PLL is disabled

bit 6-3 IRCF<3:0>: Internal Oscillator Frequency Select bits
- 000x = 31 kHz LF
- 0010 = 31.25 kHz MF
- 0011 = 31.25 kHz HF\(^{(1)}\)
- 0100 = 62.5 kHz MF
- 0101 = 125 kHz MF
- 0110 = 250 kHz MF
- 0111 = 500 kHz MF (default upon Reset)
- 1000 = 125 kHz HF\(^{(1)}\)
- 1001 = 250 kHz HF\(^{(1)}\)
- 1010 = 500 kHz HF\(^{(1)}\)
- 1011 = 1 MHz HF
- 1100 = 2 MHz HF
- 1101 = 4 MHz HF
- 1110 = 8 MHz or 32 MHz HF (see Section 5.2.2.1 “HFINTOSC”)
- 1111 = 16 MHz HF

bit 2 Unimplemented: Read as '0'

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

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>T1OSCR</td>
<td>PLLR</td>
<td>OSTS</td>
<td>HFIOFR</td>
<td>HFIOFL</td>
<td>MFIOFR</td>
<td>LFIOFR</td>
<td>HFIOFS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Legend:</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = Readable bit</td>
</tr>
<tr>
<td>W = Writable bit</td>
</tr>
<tr>
<td>U = Unimplemented bit, read as '0'</td>
</tr>
<tr>
<td>u = Bit is unchanged</td>
</tr>
<tr>
<td>x = Bit is unknown</td>
</tr>
<tr>
<td>q = Conditional</td>
</tr>
<tr>
<td>-n/n = Value at POR and BOR/Value at all other Resets</td>
</tr>
<tr>
<td>'1' = Bit is set</td>
</tr>
<tr>
<td>'0' = Bit is cleared</td>
</tr>
</tbody>
</table>

- **bit 7** T1OSCR: Timer1 Oscillator Ready bit
  - If T1OSCEN = 1:
    - 1 = Timer1 oscillator is ready
    - 0 = Timer1 oscillator is not ready
  - If T1OSCEN = 0:
    - 1 = Timer1 clock source is always ready
- **bit 6** PLLR 4x PLL Ready bit
  - 1 = 4x PLL is ready
  - 0 = 4x PLL is not ready
- **bit 5** OSTS: Oscillator Start-up Time-out Status bit
  - 1 = Running from the clock defined by the FOSC<2:0> bits of the Configuration Word 1
  - 0 = Running from an internal oscillator (FOSC<2:0> = 100)
- **bit 4** HFIOFR: High Frequency Internal Oscillator Ready bit
  - 1 = HFINTOSC is ready
  - 0 = HFINTOSC is not ready
- **bit 3** HFIOFL: High Frequency Internal Oscillator Locked bit
  - 1 = HFINTOSC is at least 2% accurate
  - 0 = HFINTOSC is not 2% accurate
- **bit 2** MFIOFR: Medium Frequency Internal Oscillator Ready bit
  - 1 = MFINTOSC is ready
  - 0 = MFINTOSC is not ready
- **bit 1** LFIOFR: Low Frequency Internal Oscillator Ready bit
  - 1 = LFINTOSC is ready
  - 0 = LFINTOSC is not ready
- **bit 0** HFIOFS: High Frequency Internal Oscillator Stable bit
  - 1 = HFINTOSC is at least 0.5% accurate
  - 0 = HFINTOSC is not 0.5% accurate
### 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>IRCF3</td>
<td>IRCF2</td>
<td>IRCF1</td>
<td>IRCF0</td>
<td>—</td>
<td>SCS1</td>
<td>SCS0</td>
<td>67</td>
</tr>
<tr>
<td>OSCSTAT</td>
<td>T1OSCR</td>
<td>PLLR</td>
<td>OSTS</td>
<td>HFIOR</td>
<td>HFIOL</td>
<td>MFIOR</td>
<td>LFIOR</td>
<td>HFIOS</td>
<td>68</td>
</tr>
<tr>
<td>OSCTUNE</td>
<td>—</td>
<td>—</td>
<td>TUN5</td>
<td>TUN4</td>
<td>TUN3</td>
<td>TUN2</td>
<td>TUN1</td>
<td>TUN0</td>
<td>69</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSFIE</td>
<td>C2IE</td>
<td>C1IE</td>
<td>EEIE</td>
<td>BCL1E</td>
<td>—</td>
<td>—</td>
<td>CCP2IE(1)</td>
<td>92</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSFIF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>EEIF</td>
<td>BCL1F</td>
<td>—</td>
<td>—</td>
<td>CCP2IF(1)</td>
<td>95</td>
</tr>
<tr>
<td>T1CON</td>
<td>TMR1CS1</td>
<td>TMR1CS0</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>T1SYNC</td>
<td>—</td>
<td>TMR1ON</td>
<td>185</td>
</tr>
</tbody>
</table>

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

Note 1: PIC16F/LF1827 only.

### TABLE 5-3: SUMMARY OF CONFIGURATION WORD ASSOCIATED 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>13:8</td>
<td>—</td>
<td>—</td>
<td>FCMEN</td>
<td>IESO</td>
<td>CLKOUTEN</td>
<td>BOREN1</td>
<td>BOREN0</td>
<td>CPD</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>CP</td>
<td>MCLRE</td>
<td>PWRE</td>
<td>WDTE1</td>
<td>WDTE0</td>
<td>FOSC2</td>
<td>FOSC1</td>
<td>FOSC0</td>
<td></td>
</tr>
</tbody>
</table>

Legend: — = unimplemented locations read as ‘0’. Shaded cells are not used by clock sources.
6.0 REFERENCE CLOCK MODULE

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

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

The reference clock module is controlled by the CLKRCON register (Register 6-1) and is enabled when setting the CLKREN bit. To output the divided clock signal to the CLKR port pin, the CLKROE bit must be set. The CLKRCON<2:0> bits enable the selection of 8 different clock divider options. The CLKRCON<1:0> bits can be used to modify the duty cycle of the output clock. The CLKRSLR bit controls slew rate limiting.

For information on using the reference clock output with the modulator module, see Section 22.0 “Data Signal Modulator”.

6.1 Slew rate

When a reference clock signal of 20 MHz or greater is required, the slew rate limitation on the output port pin can be disabled. The slew rate limitation can be removed by clearing the CLKRSLR bit in the CLKRCON register.

6.2 Effects of a Reset

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

6.3 Conflicts with the CLKR pin

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

- LP, XT, or HS oscillator mode is selected.
- CLKOUT function is enabled.

Even if either of these cases are true, the module can still be enabled and the reference clock signal may be used in conjunction with the modulator module.

6.3.1 OSCILLATOR MODES

If LP, XT, or HS oscillator modes are selected, the OSC2/CLKR pin must be used as an oscillator input pin and the CLKR output cannot be enabled. See Section 5.2 “Clock Source Types” for more information on different oscillator modes.

6.3.2 CLKOUT FUNCTION

The CLKOUT function has a higher priority than the reference clock module. Therefore, if the CLKOUT function is enabled by the CLKOUTEN bit in Configuration Word 1, Fosc/4 will always be output on the port pin. Reference Section 4.0 “Device Configuration” for more information.

6.4 Operation During Sleep

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

Note 1: If the base clock rate is selected without a divider, the output clock will always have a duty cycle equal to that of the source clock, unless a 0% duty cycle is selected. If the clock divider is set to base clock/2, then 25% and 75% duty cycle accuracy will be dependent upon the source clock.
REGISTER 6-1: CLKRCON: REFERENCE CLOCK CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-1/1</th>
<th>R/W-1/1</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>
</tr>
</thead>
<tbody>
<tr>
<td>CLKREN</td>
<td>CLKROE</td>
<td>CLKRSLR</td>
<td>CLKRDC1</td>
<td>CLKRDC0</td>
<td>CLKRDIV2</td>
<td>CLKRDIV1</td>
<td>CLKRDIV0</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  CLKREN: Reference Clock Module Enable bit
1 = Reference Clock module is enabled
0 = Reference Clock module is disabled

bit 6  CLKROE: Reference Clock Output Enable bit(3)
1 = Reference Clock output is enabled on CLKR pin
0 = Reference Clock output disabled on CLKR pin

bit 5  CLKRSLR: Reference Clock Slew Rate Control Limiting Enable bit
1 = Slew Rate limiting is enabled
0 = Slew Rate limiting is disabled

bit 4-3  CLKRDC<1:0>: Reference Clock Duty Cycle bits
11 = Clock outputs duty cycle of 75%
10 = Clock outputs duty cycle of 50%
01 = Clock outputs duty cycle of 25%
00 = Clock outputs duty cycle of 0%

bit 2-0  CLKRDIV<2:0> Reference Clock Divider bits
111 = Base clock value divided by 128
110 = Base clock value divided by 64
101 = Base clock value divided by 32
100 = Base clock value divided by 16
011 = Base clock value divided by 8
010 = Base clock value divided by 4
001 = Base clock value divided by 2(1)
000 = Base clock value(2)

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

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

3: To route CLKR to pin, CLKOUTEN of Configuration Word 1 = 1 is required. CLKOUTEN of Configuration Word 1 = 0 will result in Fosc/4.
TABLE 6-1: SUMMARY OF REGISTERS ASSOCIATED WITH REFERENCE 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>CLKRCN</td>
<td>CLKREN</td>
<td>CLKROE</td>
<td>CLKRSLR</td>
<td>CLKRDC1</td>
<td>CLKRDC0</td>
<td>CLKRDC2</td>
<td>CLKRDIV2</td>
<td>CLKRDIV1</td>
<td>CLKRDIV0</td>
</tr>
</tbody>
</table>

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

TABLE 6-2: SUMMARY OF CONFIGURATION WORD ASSOCIATED WITH REFERENCE 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>13:8</td>
<td>—</td>
<td>—</td>
<td>FCMEN</td>
<td>IESO</td>
<td>CLKOUTEN</td>
<td>BOREN1</td>
<td>BOREN0</td>
<td>CPD</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>CP</td>
<td>MCLRE</td>
<td>PWRTE</td>
<td>WDTE1</td>
<td>WDTE0</td>
<td>FOSC2</td>
<td>FOSC1</td>
<td>FOSC0</td>
<td></td>
</tr>
</tbody>
</table>

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

There are multiple ways to reset this device:

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

To allow \( V_{DD} \) to stabilize, an optional power-up timer can be enabled to extend the Reset time after a BOR or POR event.

A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 7-1.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig7-1.png}
\caption{SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT}
\end{figure}
7.1 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.

7.1.1 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 Word 1.

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).

7.2 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 Word 1. 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 7-3 for more information.

The Brown-out Reset voltage level is selectable by configuring the BORV bit in Configuration Word 2.

A VDD noise rejection filter prevents the BOR from triggering on small events. If VDD falls below VBOR for a duration greater than parameter TBORD, the device will reset. See Figure 7-3 for more information.

### TABLE 7-1: BOR OPERATING MODES

<table>
<thead>
<tr>
<th>BOREn Config bits</th>
<th>SBOREn</th>
<th>Device Mode</th>
<th>BOR Mode</th>
<th>Device Operation upon release of POR</th>
<th>Device Operation upon wake-up from Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOR_ON (11)</td>
<td>X</td>
<td>X</td>
<td>Active</td>
<td>Waits for BOR ready(1)</td>
<td></td>
</tr>
<tr>
<td>BOR_NSLEEP (10)</td>
<td>X</td>
<td>Awake</td>
<td>Active</td>
<td>Waits for BOR ready</td>
<td></td>
</tr>
<tr>
<td>BOR_NSLEEP (10)</td>
<td>X</td>
<td>Sleep</td>
<td>Disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOR_SBOREN (01)</td>
<td>1</td>
<td>X</td>
<td>Active</td>
<td>Begins immediately</td>
<td></td>
</tr>
<tr>
<td>BOR_SBOREN (01)</td>
<td>0</td>
<td>X</td>
<td>Disabled</td>
<td>Begins immediately</td>
<td></td>
</tr>
<tr>
<td>BOR_OFF (00)</td>
<td>X</td>
<td>X</td>
<td>Disabled</td>
<td>Begins immediately</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Even though this case specifically waits for the BOR, the BOR is already operating, so there is no delay in start-up.

7.2.1 BOR IS ALWAYS ON

When the BOREn bits of Configuration Word 1 are set 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.

7.2.2 BOR IS OFF IN SLEEP

When the BOREn bits of Configuration Word 1 are set 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.

7.2.3 BOR CONTROLLED BY SOFTWARE

When the BOREn bits of Configuration Word 1 are set 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.
REGISTER 7-1:  BORCON: BROWN-OUT RESET CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-1/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>U-0</th>
<th>R-q/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBOREN</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BORRDY</td>
</tr>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</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
- -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 Word 1 ≠ 01:
      SBOREN is read/write, but has no effect on the BOR.
      If BOREN<1:0> in Configuration Word 1 = 01:
      1 = BOR Enabled
      0 = BOR Disabled

bit 6-1 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: TPWRT delay only if PWRTE bit is programmed to ‘0’.
7.3 MCLR

The MCLR is an optional external input that can reset the device. The MCLR function is controlled by the MCLRE bit of Configuration Word 1 and the LVP bit of Configuration Word 2 (Table 7-2).

<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>

7.3.1 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.

7.3.2 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 12.2 “PORTA Registers” for more information.

7.4 Watchdog Timer (WDT) Reset

The Watchdog Timer generates a Reset if the firmware does not issue a CLRWDT instruction within the time-out period. The TO and PD bits in the STATUS register are changed to indicate the WDT Reset. See Section 10.0 “Watchdog Timer” for more information.

7.5 RESET Instruction

A RESET instruction will cause a device Reset. The RI bit in the PCON register will be set to ‘0’. See Table 7-4 for default conditions after a RESET instruction has occurred.

7.6 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 Word 2. See Section 3.4.2 “Overflow/Underflow Reset” for more information.

7.7 Programming Mode Exit

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

7.8 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 Word 1.

7.9 Start-up Sequence

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

1. Power-up Timer runs to completion (if enabled).
2. Oscillator start-up timer runs to completion (if required for oscillator source).
3. 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 (With Fail-Safe Clock Monitor)” for more information.

The Power-up Timer and oscillator start-up timer run independently of MCLR Reset. If MCLR is kept low long enough, the Power-up Timer and oscillator start-up timer will expire. Upon bringing MCLR high, the device will begin execution immediately (see Figure 7-4). This is useful for testing purposes or to synchronize more than one device operating in parallel.
FIGURE 7-4:  RESET START-UP SEQUENCE

- Vdd
- Internal POR
- Power-up Timer
- MCLR
- Internal RESET

Oscillator Modes

External Crystal
- Oscillator Start Up Timer
- Oscillator
- Fosc

Internal Oscillator
- Oscillator
- Fosc

External Clock (EC)
- CLkin
- Fosc
7.10 Determining the Cause of a Reset

Upon any Reset, multiple bits in the STATUS and PCON register are updated to indicate the cause of the Reset. Table 7-3 and Table 7-4 show the Reset conditions of these registers.

### TABLE 7-3: RESET STATUS BITS AND THEIR SIGNIFICANCE

<table>
<thead>
<tr>
<th>STKOVF</th>
<th>STKUNF</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>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>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>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>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>u</td>
<td>u</td>
<td>u</td>
<td>u</td>
<td>u</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>
</tr>
<tr>
<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>Interrupt Wake-up from Sleep</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>0</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>1</td>
<td>0</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>Stack Overflow Reset (STVREN = 1)</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>Stack Underflow Reset (STVREN = 1)</td>
</tr>
</tbody>
</table>

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

### TABLE 7-4: RESET CONDITION FOR SPECIAL REGISTERS (2)

<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>uu-- 0uuu</td>
</tr>
<tr>
<td>MCLR Reset during Sleep</td>
<td>0000h</td>
<td>---1 0uuu</td>
<td>uu-- 0uuu</td>
</tr>
<tr>
<td>WDT Reset</td>
<td>0000h</td>
<td>---0 uuuuu</td>
<td>uu-- uuuuu</td>
</tr>
<tr>
<td>WDT Wake-up from Sleep</td>
<td>PC + 1</td>
<td>---0 0uuu</td>
<td>uu-- uuuuu</td>
</tr>
<tr>
<td>Brown-out Reset</td>
<td>0000h</td>
<td>---1 1uuu</td>
<td>00-- 11uu0</td>
</tr>
<tr>
<td>Interrupt Wake-up from Sleep</td>
<td>PC + 1(1)</td>
<td>---1 0uuu</td>
<td>uu-- uuuuu</td>
</tr>
<tr>
<td>RESET Instruction Executed</td>
<td>0000h</td>
<td>---u uuuu</td>
<td>uu-- uu0uuu</td>
</tr>
<tr>
<td>Stack Overflow Reset (STVREN = 1)</td>
<td>0000h</td>
<td>---u uuuu</td>
<td>1uu-- uuuu</td>
</tr>
<tr>
<td>Stack Underflow Reset (STVREN = 1)</td>
<td>0000h</td>
<td>---u uuuu</td>
<td>01uu-- 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 Global Enable bit (GIE) is set, the return address is pushed on the stack and PC is loaded with the interrupt vector (0004h) after execution of PC + 1.

2: If a Status bit is not implemented, that bit will be read as '0'.

1: When the wake-up is due to an interrupt and Global Enable bit (GIE) is set, the return address is pushed on the stack and PC is loaded with the interrupt vector (0004h) after execution of PC + 1.
7.11 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)
- Stack Overflow Reset (STKOVF)
- Stack Underflow Reset (STKUNF)
- MCLR Reset (RMCLR)

The PCON register bits are shown in Register 7-2.

**REGISTER 7-2: 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>U-0</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>—</td>
<td>RMCLR</td>
<td>Rl</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 set to ‘0’ by firmware

**bit 6 STKUNF**: Stack Underflow Flag bit  
1 = A Stack Underflow occurred  
0 = A Stack Underflow has not occurred or set to ‘0’ by firmware

**bit 5-4 Unimplemented**: Read as ‘0’

**bit 3 RMCLR**: MCLR Reset Flag bit  
1 = A MCLR Reset has not occurred or set to ‘1’ by firmware  
0 = A MCLR Reset has occurred (set to ‘0’ in hardware when a MCLR Reset occurs)

**bit 2 RI**: RESET Instruction Flag bit  
1 = A RESET instruction has not been executed or set to ‘1’ by firmware  
0 = A RESET instruction has been executed (set to ‘0’ in hardware upon executing a RESET instruction)

**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 7-5: 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>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BORRDY 77</td>
</tr>
<tr>
<td>PCON</td>
<td>STKOVF</td>
<td>STKUNF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BOR 81</td>
</tr>
<tr>
<td>STATUS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TO</td>
<td>PD</td>
<td>Z</td>
<td>DC</td>
<td>C 23</td>
</tr>
<tr>
<td>WDTCON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>WDTPS4</td>
<td>WDTPS3</td>
<td>WDTPS2</td>
<td>WDTPS1</td>
<td>WDTPS0</td>
<td>SWDTEN 103</td>
</tr>
</tbody>
</table>

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

**Note 1:** Other (non Power-up) Resets include MCLR Reset and Watchdog Timer Reset during normal operation.
8.0 INTERRUPTS

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

This chapter contains the following information for Interrupts:
- Operation
- Interrupt Latency
- Interrupts During Sleep
- INT Pin
- Automatic Context Saving

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

A block diagram of the interrupt logic is shown in Figure 8-1 and Figure 8-2.

**FIGURE 8-1: INTERRUPT LOGIC**
FIGURE 8-2: PERIPHERAL INTERRUPT LOGIC -

Note 1: These interrupts are available on PIC16F/LF1827 only
8.1 Operation

Interrupts are disabled upon any device Reset. They are enabled by setting the following bits:
• GIE bit of the INTCON register
• Interrupt Enable bit(s) for the specific interrupt event(s)
• PEIE bit of the INTCON register (if the Interrupt Enable bit of the interrupt event is contained in the PIEx register)

The INTCON, PIR1, PIR2, PIR3 and PIR4 registers record individual interrupts via interrupt flag bits. Interrupt flag bits will be set, regardless of the status of the GIE, PEIE and individual interrupt enable bits.

The following events happen when an interrupt event occurs while the GIE bit is set:
• Current prefetched instruction is flushed
• GIE bit is cleared
• Current Program Counter (PC) is pushed onto the stack
• Critical registers are automatically saved to the shadow registers (See Section 8.5 “Automatic Context Saving”)
• PC is loaded with the interrupt vector 0004h

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

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

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

8.2 Interrupt Latency

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

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.
FIGURE 8-3: INTERRUPT LATENCY

<table>
<thead>
<tr>
<th>OSC1</th>
<th>CLKOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
<td>Q4</td>
</tr>
<tr>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
<td>Q4</td>
</tr>
<tr>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>Q3</td>
<td>Q4</td>
</tr>
</tbody>
</table>

Interrupt Sampled during Q1

GIE

PC

Execute

1 Cycle Instruction at PC

| Inst(PC) | NOP | NOP | Inst(0004h) |

Interrupt

GIE

PC

Execute

2 Cycle Instruction at PC

| Inst(PC) | NOP | NOP | Inst(0004h) |

Interrupt

GIE

PC

Execute

3 Cycle Instruction at PC

| INST(PC) | NOP | NOP | NOP | Inst(0004h) | Inst(0005h) |

Interrupt

GIE

PC

Execute

3 Cycle Instruction at PC

| INST(PC) | NOP | NOP | NOP | NOP | NOP | Inst(0004h) |
FIGURE 8-4: INT PIN INTERRUPT TIMING

<table>
<thead>
<tr>
<th>OSC1</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>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKOUT (3)</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>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT pin</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>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTF</td>
<td></td>
<td></td>
<td></td>
<td>(1)</td>
<td>(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>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>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

INSTRUCTION FLOW

<table>
<thead>
<tr>
<th>PC</th>
<th>PC</th>
<th>PC + 1</th>
<th>PC + 1</th>
<th>0004h</th>
<th>0005h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Fetched</td>
<td>Inst (PC)</td>
<td>Inst (PC + 1)</td>
<td>—</td>
<td>Inst (0004h)</td>
<td>Inst (0005h)</td>
</tr>
<tr>
<td>Instruction Executed</td>
<td>Inst (PC – 1)</td>
<td>Inst (PC)</td>
<td>Dummy Cycle</td>
<td>Dummy Cycle</td>
<td>Inst (0004h)</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:** CLKOUT not available in all oscillator modes.

**4:** For minimum width of INT pulse, refer to AC specifications in Section 29.0 “Electrical Specifications”.

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

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

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

8.4 INT Pin

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

8.5 Automatic Context Saving

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

- W register
- STATUS register (except for 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.
8.5.1 INTCON REGISTER

The INTCON register is a readable and writable register, which contains the various enable and flag bits for TMR0 register overflow, interrupt-on-change and external INT pin interrupts.

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

REGISTER 8-1: INTCON: INTERRUPT CONTROL 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-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</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
- **-/-** = Value at POR and BOR/Value at all other Resets
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **HS** = Bit is set by hardware

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

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

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

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

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

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

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

bit 0   **IOCIF**: Interrupt-on-Change Interrupt Flag bit
       1 = When at least one of the interrupt-on-change pins changed state
       0 = None of the interrupt-on-change pins have changed state
8.5.2 PIE1 REGISTER

The PIE1 register contains the interrupt enable bits, as shown in Register 8-2.

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

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TMR1GIE: Timer1 Gate Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ADIE: A/D Converter (ADC) Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>RCIE: USART Receive Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TXIE: USART Transmit Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SSP1IE: Synchronous Serial Port 1 (MSSP1) Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CCP1IE: CCP1 Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>TMR2IE: TMR2 to PR2 Match Interrupt Enable bit</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>TMR1IE: Timer1 Overflow Interrupt Enable bit</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**
- **TMR1GIE:** Timer1 Gate Interrupt Enable bit
  - 1 = Enables the Timer1 Gate Acquisition interrupt
  - 0 = Disables the Timer1 Gate Acquisition interrupt

**bit 6**
- **ADIE:** A/D Converter (ADC) Interrupt Enable bit
  - 1 = Enables the ADC interrupt
  - 0 = Disables the ADC interrupt

**bit 5**
- **RCIE:** USART Receive Interrupt Enable bit
  - 1 = Enables the USART receive interrupt
  - 0 = Disables the USART receive interrupt

**bit 4**
- **TXIE:** USART Transmit Interrupt Enable bit
  - 1 = Enables the USART transmit interrupt
  - 0 = Disables the USART transmit interrupt

**bit 3**
- **SSP1IE:** Synchronous Serial Port 1 (MSSP1) Interrupt Enable bit
  - 1 = Enables the MSSP1 interrupt
  - 0 = Disables the MSSP1 interrupt

**bit 2**
- **CCP1IE:** CCP1 Interrupt Enable bit
  - 1 = Enables the CCP1 interrupt
  - 0 = Disables the CCP1 interrupt

**bit 1**
- **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
  - 1 = Enables the Timer2 to PR2 match interrupt
  - 0 = Disables the Timer2 to PR2 match interrupt

**bit 0**
- **TMR1IE:** Timer1 Overflow Interrupt Enable bit
  - 1 = Enables the Timer1 overflow interrupt
  - 0 = Disables the Timer1 overflow interrupt
8.5.3 PIE2 REGISTER

The PIE2 register contains the interrupt enable bits, as shown in Register 8-3.

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

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>OSFIE: Oscillator Fail Interrupt Enable bit</td>
</tr>
<tr>
<td>6</td>
<td>C2IE: Comparator C2 Interrupt Enable bit</td>
</tr>
<tr>
<td>5</td>
<td>C1IE: Comparator C1 Interrupt Enable bit</td>
</tr>
<tr>
<td>4</td>
<td>EEIE: EEPROM Write Completion Interrupt Enable bit</td>
</tr>
<tr>
<td>3</td>
<td>BCL1IE: MSSP1 Bus Collision Interrupt Enable bit</td>
</tr>
<tr>
<td>2-1</td>
<td>Unimplemented: Read as ‘0’</td>
</tr>
<tr>
<td>0</td>
<td>CCP2IE: CCP2 Interrupt Enable 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

- OSFIE: Oscillator Fail Interrupt Enable bit
  - 1 = Enables the Oscillator Fail interrupt
  - 0 = Disables the Oscillator Fail interrupt
- C2IE: Comparator C2 Interrupt Enable bit
  - 1 = Enables the Comparator C2 interrupt
  - 0 = Disables the Comparator C2 interrupt
- C1IE: Comparator C1 Interrupt Enable bit
  - 1 = Enables the Comparator C1 interrupt
  - 0 = Disables the Comparator C1 interrupt
- EEIE: EEPROM Write Completion Interrupt Enable bit
  - 1 = Enables the EEPROM Write Completion interrupt
  - 0 = Disables the EEPROM Write Completion interrupt
- BCL1IE: MSSP1 Bus Collision Interrupt Enable bit
  - 1 = Enables the MSSP1 Bus Collision Interrupt
  - 0 = Disables the MSSP1 Bus Collision Interrupt
- CCP2IE: CCP2 Interrupt Enable bit
  - 1 = Enables the CCP2 interrupt
  - 0 = Disables the CCP2 interrupt

**Note 1:** PIC16F/LF1827 only.
8.5.4 PIE3 REGISTER\(^{(1)}\)

The PIE3 register contains the interrupt enable bits, as shown in Register 8-4.

### REGISTER 8-4: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3\(^{(1)}\)

<table>
<thead>
<tr>
<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>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>TMR6IE</td>
<td>—</td>
<td>TMR4IE</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 7-6 | Unimplemented: Read as '0' |
| bit 5   | CCP4IE: CCP4 Interrupt Enable bit  |
| 1       | Enables the CCP4 interrupt        |
| 0       | Disables the CCP4 interrupt       |
| bit 4   | CCP3IE: CCP3 Interrupt Enable bit  |
| 1       | Enables the CCP3 interrupt        |
| 0       | Disables the CCP3 interrupt       |
| bit 3   | TMR6IE: TMR6 to PR6 Match Interrupt Enable bit  |
| 1       | Enables the TMR6 to PR6 Match interrupt |
| 0       | Disables the TMR6 to PR6 Match interrupt |
| bit 2   | Unimplemented: Read as '0'        |
| bit 1   | TMR4IE: TMR4 to PR4 Match Interrupt Enable bit  |
| 1       | Enables the TMR4 to PR4 Match interrupt |
| 0       | Disables the TMR4 to PR4 Match interrupt |
| bit 0   | Unimplemented: Read as '0'        |

**Note 1:** This register is only available on PIC16F/LF1827.
8.5.5 PIE4 REGISTER(1)

The PIE4 register contains the interrupt enable bits, as shown in Register 8-5.

**Note 1:** The PIE4 register is available only on the PIC16F/LF1827 device.

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

**REGISTER 8-5: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4(1)**

<table>
<thead>
<tr>
<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-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>BCL2IE</td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 0</td>
<td>SSP2IE</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** Unimplemented: Read as ‘0’

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

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

**Note 1:** This register is only available on PIC16F/LF1827.
8.5.6 PIR1 REGISTER

The PIR1 register contains the interrupt flag bits, as shown in Register 8-6.

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

REGISTER 8-6: PIR1: PERIPHERAL INTERRUPT REQUEST REGISTER 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TMR1GIF</td>
<td>Timer1 Gate Interrupt Flag bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Interrupt is pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Interrupt is not pending</td>
</tr>
<tr>
<td>6</td>
<td>ADIF</td>
<td>A/D Converter Interrupt Flag bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Interrupt is pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Interrupt is not pending</td>
</tr>
<tr>
<td>5</td>
<td>RCIF</td>
<td>USART Receive Interrupt Flag bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Interrupt is pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Interrupt is not pending</td>
</tr>
<tr>
<td>4</td>
<td>TXIF</td>
<td>USART Transmit Interrupt Flag bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Interrupt is pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Interrupt is not pending</td>
</tr>
<tr>
<td>3</td>
<td>SSP1IF</td>
<td>Synchronous Serial Port 1 (MSSP1) Interrupt Flag bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Interrupt is pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Interrupt is not pending</td>
</tr>
<tr>
<td>2</td>
<td>CCP1IF</td>
<td>CCP1 Interrupt Flag bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Interrupt is pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Interrupt is not pending</td>
</tr>
<tr>
<td>1</td>
<td>TMR2IF</td>
<td>Timer2 to PR2 Interrupt Flag bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Interrupt is pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Interrupt is not pending</td>
</tr>
<tr>
<td>0</td>
<td>TMR1IF</td>
<td>Timer1 Overflow Interrupt Flag bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Interrupt is pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Interrupt is not pending</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

- **TMR1GIF**: Timer1 Gate Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending

bit 6

- **ADIF**: A/D Converter Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending

bit 5

- **RCIF**: USART Receive Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending

bit 4

- **TXIF**: USART Transmit Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending

bit 3

- **SSP1IF**: Synchronous Serial Port 1 (MSSP1) Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending

bit 2

- **CCP1IF**: CCP1 Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending

bit 1

- **TMR2IF**: Timer2 to PR2 Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending

bit 0

- **TMR1IF**: Timer1 Overflow Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending
8.5.7 PIR2 REGISTER

The PIR2 register contains the interrupt flag bits, as shown in Register 8-7.

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

**REGISTER 8-7: PIR2: PERIPHERAL INTERRUPT REQUEST REGISTER 2**

<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 2-1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSFIF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>EEIF</td>
<td>BCL1IF</td>
<td>Unimplemented</td>
<td>CCP2IF(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
- **-n/n** = Value at POR and BOR/Value at all other Resets
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared

- **OSFIF:** Oscillator Fail Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending

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

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

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

- **BCL1IF:** MSSP1 Bus Collision Interrupt Flag bit
  - 1 = Interrupt is pending
  - 0 = Interrupt is not pending

- **Unimplemented:** Read as ‘0’

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

**Note 1:** PIC16F/LF1827 only.
8.5.8 PIR3 REGISTER(1)

The PIR3 register contains the interrupt flag bits, as shown in Register 8-8.

| bit 7-6 | Unimplemented: Read as ‘0’ |
| bit 5   | CCP4IF: CCP4 Interrupt Flag bit |
|        | 1 = Interrupt is pending         |
|        | 0 = Interrupt is not pending     |
| bit 4   | CCP3IF: CCP3 Interrupt Flag bit |
|        | 1 = Interrupt is pending         |
|        | 0 = Interrupt is not pending     |
| bit 3   | TMR6IF: TMR6 to PR6 Match Interrupt Flag bit |
|        | 1 = Interrupt is pending         |
|        | 0 = Interrupt is not pending     |
| bit 2   | Unimplemented: Read as ‘0’       |
| bit 1   | TMR4IF: TMR4 to PR4 Match Interrupt Flag bit |
|        | 1 = Interrupt is pending         |
|        | 0 = Interrupt is not pending     |
| bit 0   | Unimplemented: Read as ‘0’       |

**Note 1:** The PIR3 register is available only on the PIC16F/LF1827 device.

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

**REGISTER 8-8:** PIR3: PERIPHERAL INTERRUPT REQUEST REGISTER 3(1)

<table>
<thead>
<tr>
<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>U-0</th>
<th>R/W-0/0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>TMR6IF</td>
<td>—</td>
<td>TMR4IF</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

Note 1: This register is only available on PIC16F/LF1827.
8.5.9 PIR4 REGISTER(1)

The PIR4 register contains the interrupt flag bits, as shown in Register 8-9.

Note 1: The PIR4 register is available only on the PIC16F/LF1827 device.

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

REGISTER 8-9: PIR4: PERIPHERAL INTERRUPT REQUEST REGISTER 4(1)

| Bit 7-2 | Unimplemented: Read as ‘0’ |
| Bit 1 | BCL2IF: MSSP2 Bus Collision Interrupt Flag bit |
| 1 = A Bus Collision was detected (must be cleared in software) |
| 0 = No Bus collision was detected |
| Bit 0 | SSP2IF: Master Synchronous Serial Port 2 (MSSP2) Interrupt Flag bit |
| 1 = The Transmission/Reception/Bus Condition is complete (must be cleared in software) |
| 0 = Waiting to Transmit/Receive/Bus Condition in progress |

Note 1: This register is only available on PIC16F/LF1827.

TABLE 8-1: 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>89</td>
</tr>
<tr>
<td>OPTION_REG</td>
<td>WPUEN</td>
<td>INTEDG</td>
<td>TMR0CS</td>
<td>TMR0SE</td>
<td>PSA</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
<td>175</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSFIE</td>
<td>C2IE</td>
<td>C1IE</td>
<td>EEIE</td>
<td>BCL1IE</td>
<td></td>
<td></td>
<td></td>
<td>91</td>
</tr>
<tr>
<td>PIE3(1)</td>
<td></td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>92</td>
</tr>
<tr>
<td>PIE4(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>93</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSFIF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>EEIF</td>
<td>BCL1IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>95</td>
</tr>
<tr>
<td>PIR3(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>96</td>
</tr>
<tr>
<td>PIR4(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>97</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented locations read as ‘0’. Shaded cells are not used by Interrupts.

Note 1: PIC16F/LF1827 only.
9.0 POWER-DOWN MODE (SLEEP)

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

Upon entering Sleep mode, the following conditions exist:
1. WDT will be cleared but keeps running, if enabled for operation during Sleep.
2. 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. Timer1 oscillator is unaffected and peripherals that operate from it may continue operation in Sleep.
7. ADC is unaffected, if the dedicated FRC clock is selected.
8. Capacitive Sensing oscillator is unaffected.
9. I/O ports maintain the status they had before SLEEP was executed (driving high, low or high-impedance).
10. Resets other than WDT are not affected by Sleep mode.

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

To minimize current consumption, the following conditions should be considered:
• I/O pins should not be floating
• External circuitry sinking current from I/O pins
• Internal circuitry sourcing current from I/O pins
• Current draw from pins with internal weak pull-ups
• Modules using 31 kHz LFINTOSC
• Modules using Timer1 oscillator

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

Examples of internal circuitry that might be sourcing current include modules such as the DAC and FVR modules. See Section 16.0 “Digital-to-Analog Converter (DAC) Module” and Section 14.0 “Fixed Voltage Reference (FVR)” for more information on these modules.

9.1 Wake-up from Sleep

The device can wake-up from Sleep through one of the following events:
1. External Reset input on 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 7.10 “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 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 call the Interrupt Service Routine. In cases where the execution of the instruction following SLEEP is not desirable, the user should have a NOP after the SLEEP instruction.

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

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

- If the interrupt occurs before the execution of a **SLEEP** instruction
  - **SLEEP** instruction will execute as a **NOP**.
  - WDT and WDT prescaler will not be cleared
  - 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 9-1: WAKE-UP FROM SLEEP THROUGH INTERRUPT**

**TABLE 9-1: 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>IOIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>89</td>
</tr>
<tr>
<td>IOCBF</td>
<td>IOCBF7</td>
<td>IOCBF6</td>
<td>IOCBF5</td>
<td>IOCBF4</td>
<td>IOCBF3</td>
<td>IOCBF2</td>
<td>IOCBF1</td>
<td>IOCBF0</td>
<td>132</td>
</tr>
<tr>
<td>IOCBN</td>
<td>IOCBN7</td>
<td>IOCBN6</td>
<td>IOCBN5</td>
<td>IOCBN4</td>
<td>IOCBN3</td>
<td>IOCBN2</td>
<td>IOCBN1</td>
<td>IOCBN0</td>
<td>132</td>
</tr>
<tr>
<td>IOCBP</td>
<td>IOCBP7</td>
<td>IOCBP6</td>
<td>IOCBP5</td>
<td>IOCBP4</td>
<td>IOCBP3</td>
<td>IOCBP2</td>
<td>IOCBP1</td>
<td>IOCBP0</td>
<td>132</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSFIE</td>
<td>C2IE</td>
<td>C1IE</td>
<td>EEIE</td>
<td>BCL1IE</td>
<td>—</td>
<td>—</td>
<td>CCP2IE(1)</td>
<td>91</td>
</tr>
<tr>
<td>PIEL(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BCL2IE</td>
<td>93</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSFIF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>EEIF</td>
<td>BCL1IF</td>
<td>—</td>
<td>—</td>
<td>CCP2IF(1)</td>
<td>95</td>
</tr>
<tr>
<td>PIRL(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BCL2IF</td>
<td>97</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>23</td>
</tr>
<tr>
<td>WDTCN</td>
<td>—</td>
<td>—</td>
<td>WDTPS4</td>
<td>WDTPS3</td>
<td>WDTPS2</td>
<td>WDTPS1</td>
<td>WDTPS0</td>
<td>SWDTEN</td>
<td>103</td>
</tr>
</tbody>
</table>

**Legend:** — unimplemented, read as ‘0’. Shaded cells are not used in Power-down mode.

**Note 1:** PIC16F/LF1827 only.
10.0 WATCHDOG TIMER

The Watchdog Timer is a system timer that generates a Reset if the firmware does not issue a \texttt{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 268 seconds (typical)
- Multiple Reset conditions
- Operation during Sleep

FIGURE 10-1: WATCHDOG TIMER BLOCK DIAGRAM
10.1 Independent Clock Source
The WDT derives its time base from the 31 kHz LFINTOSC internal oscillator.

10.2 WDT Operating Modes
The Watchdog Timer module has four operating modes controlled by the WDTE<1:0> bits in Configuration Word 1. See Table 10-1.

10.2.1 WDT IS ALWAYS ON
When the WDTE bits of Configuration Word 1 are set to '11', the WDT is always on.
WDT protection is active during Sleep.

10.2.2 WDT IS OFF IN SLEEP
When the WDTE bits of Configuration Word 1 are set to '10', the WDT is on, except in Sleep.
WDT protection is not active during Sleep.

10.2.3 WDT CONTROLLED BY SOFTWARE
When the WDTE bits of Configuration Word 1 are set to '01', the WDT is controlled by the SWDTEN bit of the WDTCON register.
WDT protection is unchanged by Sleep. See Table 10-1 for more details.

TABLE 10-1: WDT OPERATING MODES

<table>
<thead>
<tr>
<th>WDTE Config bits</th>
<th>SWDTEN</th>
<th>Device Mode</th>
<th>WDT Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDT_ON (11)</td>
<td>X</td>
<td>X</td>
<td>Active</td>
</tr>
<tr>
<td>WDT_NSLEEP (10)</td>
<td>X</td>
<td>Awake</td>
<td>Active</td>
</tr>
<tr>
<td>WDT_NSLEEP (10)</td>
<td>X</td>
<td>Sleep</td>
<td>Disabled</td>
</tr>
<tr>
<td>WDT_SWDTEN (01)</td>
<td>1</td>
<td>X</td>
<td>Active</td>
</tr>
<tr>
<td>WDT_SWDTEN (01)</td>
<td>0</td>
<td>X</td>
<td>Disabled</td>
</tr>
<tr>
<td>WDT_OFF (00)</td>
<td>X</td>
<td>X</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

10.3 Time-Out Period
The WDTPS bits of the WDTCON register set the time-out period from 1ms to 268 seconds. After a Reset, the default time-out period is 2 seconds.

10.4 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 event
- WDT is disabled
- OST is running
See Table 10-2 for more information.

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

When the device exits Sleep, the WDT is cleared again. The WDT remains clear until the OST, if enabled, completes. See Section 5.0 “Oscillator Module (With Fail-Safe Clock Monitor)” for more information on the OST.

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

TABLE 10-2: 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>Cleared until the end of OST</td>
</tr>
<tr>
<td>WDTE&lt;1:0&gt; = 10 and enter Sleep</td>
<td>Unaffected</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 = T1OSC, EXTRC, INTOSC, EXTCCLK</td>
<td></td>
</tr>
<tr>
<td>Exit Sleep + System Clock = XT, HS, LP</td>
<td></td>
</tr>
<tr>
<td>Change INTOSC divider (IRCF bits)</td>
<td></td>
</tr>
</tbody>
</table>
## REGISTER 10-1: WDTCON: WATCHDOG TIMER CONTROL REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-1/0</th>
<th>R/W-0/0</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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>WDTPS4</td>
<td>WDTPS3</td>
<td>WDTPS2</td>
<td>WDTPS1</td>
<td>WDTPS0</td>
<td>SWDTEN</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
- **-m/n** = Value at POR and BOR/Value at all other Resets

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

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

<table>
<thead>
<tr>
<th>Bit Value</th>
<th>Prescale Rate</th>
<th>Interval</th>
<th>Prescale Rate</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>1:32</td>
<td>1 ms</td>
<td>1:64</td>
<td>2 ms</td>
</tr>
<tr>
<td>00001</td>
<td>1:64</td>
<td>2 ms</td>
<td>1:128</td>
<td>4 ms</td>
</tr>
<tr>
<td>00010</td>
<td>1:128</td>
<td>4 ms</td>
<td>1:256</td>
<td>8 ms</td>
</tr>
<tr>
<td>00100</td>
<td>1:256</td>
<td>8 ms</td>
<td>1:512</td>
<td>16 ms</td>
</tr>
<tr>
<td>00101</td>
<td>1:512</td>
<td>16 ms</td>
<td>1:1024</td>
<td>32 ms</td>
</tr>
<tr>
<td>00110</td>
<td>1:1024</td>
<td>32 ms</td>
<td>1:2048</td>
<td>64 ms</td>
</tr>
<tr>
<td>00111</td>
<td>1:2048</td>
<td>64 ms</td>
<td>1:4096</td>
<td>128 ms</td>
</tr>
<tr>
<td>01000</td>
<td>1:4096</td>
<td>128 ms</td>
<td>1:8192</td>
<td>256 ms</td>
</tr>
<tr>
<td>01001</td>
<td>1:8192</td>
<td>256 ms</td>
<td>1:16384</td>
<td>512 ms</td>
</tr>
<tr>
<td>01010</td>
<td>1:16384</td>
<td>512 ms</td>
<td>1:32768</td>
<td>1 s</td>
</tr>
<tr>
<td>01011</td>
<td>1:32768</td>
<td>1 s</td>
<td>1:65536</td>
<td>2 s</td>
</tr>
<tr>
<td>01100</td>
<td>1:65536</td>
<td>2 s</td>
<td>1:131072 (2^17)</td>
<td>4 s</td>
</tr>
<tr>
<td>01101</td>
<td>1:131072 (2^17)</td>
<td>4 s</td>
<td>1:262144 (2^18)</td>
<td>8 s</td>
</tr>
<tr>
<td>01110</td>
<td>1:262144 (2^18)</td>
<td>8 s</td>
<td>1:524288 (2^19)</td>
<td>16 s</td>
</tr>
<tr>
<td>01111</td>
<td>1:524288 (2^19)</td>
<td>16 s</td>
<td>1:1048576 (2^20)</td>
<td>32 s</td>
</tr>
<tr>
<td>10000</td>
<td>1:1048576 (2^20)</td>
<td>32 s</td>
<td>1:2097152 (2^21)</td>
<td>64 s</td>
</tr>
<tr>
<td>10001</td>
<td>1:2097152 (2^21)</td>
<td>64 s</td>
<td>1:4194304 (2^22)</td>
<td>128 s</td>
</tr>
<tr>
<td>10010</td>
<td>1:4194304 (2^22)</td>
<td>128 s</td>
<td>1:8388608 (2^23)</td>
<td>256 s</td>
</tr>
<tr>
<td>10011</td>
<td>1:8388608 (2^23)</td>
<td>256 s</td>
<td>Reserved. Results in minimum interval (1:32)</td>
<td></td>
</tr>
<tr>
<td>11111</td>
<td>Reserved. Results in minimum interval (1:32)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### bit 0
**SWDTEN:** Software Enable/Disable for Watchdog Timer bit

- If WDTE<1:0> = 00:
  - This bit is ignored.
- If WDTE<1:0> = 01:
  - 1 = WDT is turned on
  - 0 = WDT is turned off
- If WDTE<1:0> = 1x:
  - This bit is ignored.
11.0 DATA EEPROM AND FLASH PROGRAM MEMORY CONTROL

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

- EECON1
- EECON2
- EEDATL
- EEDATH
- EEADRL
- EEADRH

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

When accessing the program memory block of the PIC16F/LF1826/27 devices, the EEDATL and EEDATH registers form a 2-byte word that holds the 14-bit data for read/write, and the EEADRL and EEADRH registers form a 2-byte word that holds the 15-bit address of the program memory location being read.

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

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

Depending on the setting of the Flash Program Memory Self Write Enable bits WRT<1:0> of the Configuration Word 2, the device may or may not be able to write certain blocks of the program memory. However, reads from the program memory are always allowed.

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

11.1 EEADRL and EEADRH Registers

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

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

11.1.1 EECON1 AND EECON2 REGISTERS

EECON1 is the control register for EE memory accesses.

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

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

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

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

Reading EECON2 will read all ‘0’s. The EECON2 register is used exclusively in the data EEPROM write sequence. To enable writes, a specific pattern must be written to EECON2.
## REGISTER 11-1: EEDATL: EEPROM DATA 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>EEDATL7</td>
<td>EEDATL6</td>
<td>EEDATL5</td>
<td>EEDATL4</td>
<td>EEDATL3</td>
<td>EEDATL2</td>
<td>EEDATL1</td>
<td>EEDATL0</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 EEDATL<7:0>: 8 Least Significant data bits of data EEPROM or Read from program memory*

---

## REGISTER 11-2: EEDATH: EEPROM 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>—</td>
<td>—</td>
<td>EEDATH5</td>
<td>EEDATH4</td>
<td>EEDATH3</td>
<td>EEDATH2</td>
<td>EEDATH1</td>
<td>EEDATH0</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 EEDATH<5:0>: 6 Most Significant Data bits from program memory*

---

## REGISTER 11-3: EEADRL: EEPROM ADDRESS 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>EEADR7</td>
<td>EEADR6</td>
<td>EEADR5</td>
<td>EEADR4</td>
<td>EEADR3</td>
<td>EEADR2</td>
<td>EEADR1</td>
<td>EEADR0</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 EEADR<7:0>: 8 Least Significant Address bits for EEPROM or program memory*

---

## REGISTER 11-4: EEADRH: EEPROM ADDRESS HIGH BYTE 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>EEADRH6</td>
<td>EEADRH5</td>
<td>EEADRH4</td>
<td>EEADRH3</td>
<td>EEADRH2</td>
<td>EEADRH1</td>
<td>EEADRH0</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 EEADRH<6:0>: Specifies the 7 Most Significant Address bits or high bits for program memory reads*
REGISTER 11-5: EECON1: EEPROM CONTROL 1 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/HC-0/0</th>
<th>R/W-x/q</th>
<th>R/W-0/0</th>
<th>R/S/HC-0/0</th>
<th>R/S/HC-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEPGD</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  EEPGD: Flash Program/Data EEPROM Memory Select bit
1 = Accesses program space Flash memory
0 = Accesses data EEPROM memory

d bit 6  CFGS: Flash Program/Data EEPROM or Configuration Select bit
1 = Accesses Configuration, User ID and Device ID Registers
0 = Accesses Flash Program or data EEPROM Memory

bit 5  LWLO: Load Write Latches Only bit
If EEPGD = 1 or CFGS = 1: (accessing program Flash)
1 = The next WR command does not initiate a write to the PFM; only the program memory latches are updated.
0 = The next WR command writes a value from EEDATH:EEDATL into program memory latches and initiates a write to the PFM of all the data stored in the program memory latches.

If EEPGD = 0 and CFGS = 1: (Accessing data EEPROM)
LWLO is ignored. The next WR command initiates a write to the data EEPROM.

bit 4  FREE: Program Flash Erase Enable bit
If EEPGD = 1 or CFGS = 1: (accessing program Flash)
1 = Perform an program Flash erase operation on the next WR command (cleared by hardware after completion of erase).
0 = Perform a program Flash write operation on the next WR command.

If EEPGD = 0 and CFGS = 0: (Accessing data EEPROM)
FREE is ignored. The next WR command will initiate both a erase cycle and a write cycle.

bit 3  WRERR: EEPROM Error Flag bit
1 = Condition could indicate an improper program or erase sequence attempt or termination (bit is set automatically on any set attempt (write ‘1’) of the WR bit.
0 = The program or erase operation completed normally.

bit 2  WREN: Program/Erase Enable bit
1 = Allows program/erase cycles
0 = Inhibits programming/erasing of program Flash and data EEPROM

bit 1  WR: Write Control bit
1 = Initiates a program Flash or data EEPROM program/erase operation.
   The operation is self-timed and the bit is cleared by hardware once operation is complete.
   The WR bit can only be set (not cleared) in software.
0 = Program/erase operation to the Flash or data EEPROM is complete and inactive.

bit 0  RD: Read Control bit
1 = Initiates an program Flash or data EEPROM read. Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software.
0 = Does not initiate a program Flash or data EEPROM data read.
REGISTER 11-6: EECON2: EEPROM CONTROL 2 REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
<td>R-0/0</td>
</tr>
<tr>
<td>EEPROM control register 2</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'
S = Bit can only be set    x = Bit is unknown         -n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set           ‘0’ = Bit is cleared

Data EEPROM Unlock Pattern bits
To unlock writes, a 55h must be written first, followed by an AAh, before setting the WR bit of the EECON1 register. The value written to this register is used to unlock the writes. There are specific timing requirements on these writes. Refer to Section 11.1.3 “Writing to the Data EEPROM Memory” for more information.
11.1.2 READING THE DATA EEPROM MEMORY

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

**EXAMPLE 11-1: DATA EEPROM READ**

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

**Note:** Data EEPROM can be read regardless of the setting of the CPD bit.

11.1.3 WRITING TO THE DATA EEPROM MEMORY

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

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

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

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

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

**EXAMPLE 11-2: DATA EEPROM WRITE**

```asm
BANKSEL EEADRL ;
MOVLW DATA_EE_ADDR ;
MOVWF EEADRLOW ;Data Memory Address to write
MOVLW DATA_EE_DATA ;
MOVWF EEDATL ;Data Memory Value to write
BCF EECON1, CFGS ;Deselect Configuration space
BCF EECON1, EEPGD ;Point to DATA memory
BSF EECON1, WREN ;Enable writes
BCF INTCON, GIE ;Disable INTs.
MOVLW 55h ;
MOVWF EECON2 ;Write 55h
MOVLW 0AAh ;
MOVWF EECON2 ;Write AAh
BSF EECON1, WR ;Set WR bit to begin write
BSF INTCON, GIE ;Enable interrupts
BCF EECON1, WREN ;Disable writes
BTFSC EECON1, WR ;Wait for write to complete
GOTO $-2 ;Done
```

**Required Sequence**
11.1.4 READING THE FLASH PROGRAM MEMORY

To read a program memory location, the user must:

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

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

**EXAMPLE 11-3: FLASH PROGRAM READ**

```
BANKSEL EEADRL ;
MOVLW MS_PROG_EE_ADDR ;
MOVWF EEADRH ; MS Byte of Program Address to read
MOVLW LS_PROG_EE_ADDR ;
MOVWF EEADRL ; LS Byte of Program Address to read
BANKSEL EECON1 ;
BSF EECON1, EEPGD ; Point to PROGRAM memory
BSF EECON1, RD ; EE Read
 ; First instruction after BSF EECON1,RD executes normally
NOP
NOP ; Any instructions here are ignored as program memory is read in second cycle after BSF EECON1,RD

BANKSEL EEDATL ;
MOVF EEDATL, W ; W = LS Byte of Program Memory
MOVF LOWPMBYTE ;
MOVF EEDATH, W ; W = MS Byte of Program EEDATL
MOVF HIGHTPMBYTE ;
```

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

2: Data EEPROM can be read regardless of the setting of the CPD bit.
EXAMPLE 11-4: 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 EEADRRL ; Select Bank for EEPROM registers
MOVLW PROG_ADDR_LO ;
MOVWF EEADRRL ; Store LSB of address
MOVLW PROG_ADDR_HI ;
MOVWL EEADRRL ; Store MSB of address
BCF EECON1,CFGS ; Select Configuration Space
BSF EECON1,EEPGD ; Select Program Memory
BCF INTCON,GIE ; Disable interrupts
BSF EECON1,RD ; Initiate read
NOP ; Executed (Figure 11-1)
NOP ; Ignored (Figure 11-1)
BSF INTCON,GIE ; Restore interrupts
MOVF EEDATL,W ; Get LSB of word
MOVF EEDATH,W ; Get MSB of word
MOVF EEDATL,W ; Store in user location
MOVF EEDATH,W ; Store in user location

FIGURE 11-1: FLASH PROGRAM MEMORY READ CYCLE EXECUTION

<table>
<thead>
<tr>
<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>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash ADDR</td>
<td>PC</td>
<td>PC + 1</td>
<td>EEDRHEEADRRL</td>
<td>PC + 3</td>
<td>PC + 4</td>
<td>PC + 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash Data</td>
<td>EEDATH,EEDATL</td>
<td>INSTR(PC)</td>
<td>INSTR(PC + 1)</td>
<td>INSTR(PC + 3)</td>
<td>INSTR(PC + 4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RD bit</td>
<td>Forced NOP executed here</td>
<td>BSF EECON1,RD executed here</td>
<td>INSTR(PC + 1) executed here</td>
<td>EEDATH,EEDATL</td>
<td>INSTR(PC + 3) executed here</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEDATL</td>
<td>EERHLT</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>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11.2 Erasing Program Memory

While executing code, program memory can only be erased by rows. A row consists of 32 words where the EEADRL<4:0> = 0000. To erase a row:

1. Load the EEADRH and EEADRL registers with the address of new row to be erased.
2. Clear the CFGS bit of the EECON1 register.
3. Set the EEPGD bit of the EECON1 register.
4. Set the FREE bit of the EECON1 register.
5. Write 55h, then AAh, to EECON2 (Flash programming unlock sequence).
6. Set control bit WR of the EECON1 register to begin the write operation.

11.3 Writing to Flash Program Memory

Before writing, program memory should be erased using the Erase Program Memory command.

No automatic erase occurs upon the initiation of the write; if the program Flash needs to be erased before writing, the row (32 words) must be erased previously.

Flash program memory may only be written to if the destination address is in a segment of memory that is not write-protected, as defined in bits WRT<1:0> of the Configuration Word 2. Flash program memory must be written in eight-word blocks. See Figure 11-2 for more details. A block consists of eight words with sequential addresses, with a lower boundary defined by an address, where EEADRL<3:0> = 0000. All block writes to program memory are done as 32-word erase by eight-word write operations. The write operation is edge-aligned and cannot occur across boundaries.

When the LWLO bit is ‘1’, the write sequence will only load the buffer register and will not actually initiate the write to program Flash:

1. Set the EEPGD, WREN and LWLO bits of the EECON1 register.
2. Write 55h, then AAh, to EECON2 (Flash programming unlock sequence).
3. Set control bit WR of the EECON1 register to begin the write operation.

To write program data, it must first be loaded into the buffer registers (see Figure 11-1). This is accomplished by first writing the destination address to EEADR and EEADRH and then writing the data to EEDATA and EEDATH. After the address and data have been set up, then the following sequence of events must be executed:

1. Set the EEPGD control bit of the EECON1 register.
2. Set the LWLO bit of the EECON1 register.
3. Write 55h, then AAh, to EECON2 (Flash programming sequence).
4. Set the WR control bit of the EECON1 register.

Up to eight buffer register locations can be written to with correct data. If less than eight words are being written to in the block of eight words, then the data for the unprogrammed words should be set to all ones.

After the "BSF EECON1, WR" instruction, the processor requires two cycles to set up the erase/write operation. The user must place two NOP instructions after the WR bit is set. Since data is being written to buffer registers, the writing of the first seven words of the block appears to occur immediately. The processor will halt internal operations for the typical 2 ms, only during the cycle in which the erase takes place (i.e., the last word of the sixteen-word block erase). This is not Sleep mode as the clocks and peripherals will continue to run. After the eight-word write cycle, the processor will resume operation with the third instruction after the EECON1 write instruction.

An example of the complete eight-word write sequence is shown in Example 11-5. The initial address is loaded into the EEADR and EEADR register pair; the eight words of data are loaded using indirect addressing.

Note: The code sequence provided in Example 12-5 must be repeated 4 times to fully program an erased program memory row of 32 words.
FIGURE 11-2: BLOCK WRITES TO 8K FLASH PROGRAM MEMORY

First word of block to be written

EEADRL<2:0> = 000  EEADRL<2:0> = 001  EEADRL<2:0> = 010  EEADRL<2:0> = 111

Buffer Register  Buffer Register  Buffer Register  Buffer Register

Program Memory
EXAMPLE 11-5: WRITING TO FLASH PROGRAM MEMORY

; This write routine assumes the following:
; 1. The 16 bytes of data are loaded, starting at the address in DATA_ADDR
; 2. Each word of data to be written is made up of two adjacent bytes in DATA_ADDR,
;    stored in little endian format
; 3. A valid starting address (the least significant bits = 000) is loaded in ADDRH:ADDRL
; 4. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F

BCF INTCON,GIE ; Disable ints so required sequences will execute properly
BANKSEL EEARH ; Bank 3
MOVF ADDRH,W ; Load initial address
MOVF EEARH ;
MOVF ADDRL,W ;
MOVF EEARL ;
MOVLW LOW DATA_ADDR ; Load initial data address
MOVF FSRL ;
MOVLW HIGH DATA_ADDR ; Load initial data address
MOVF FSRO ;
LOOP
MOVIW INDF0++ ; Load first data byte into lower
MOVF EEDATL ;
MOVIW INDF0++ ; Load second data byte into upper
MOVF EEDATH ;
BSF EECON1,EEPGD ; Point to program memory
BCF EECON1,CFGS ; Not configuration space
BSF EECON1,WREN ; Enable writes
MOVF EEARL,W ; Check if lower bits of address are '000'
XORLW 0x07 ; Check if we're on the last of 8 addresses
ANDLW 0x07 ;
BTFSC STATUS,Z ; Exit if last of eight words,
GOTO START_WRITE ;
BSF EECON1,LWLO ; Only Load Write Latches

MOVLW 55h ; Start of required write sequence:
MOVF EECON2 ; Write 55h
MOVLW 0AAh ;
BSF EECON2 ; Write AAh
NOP ; Any instructions here are ignored as processor
NOP ; halts to begin write sequence
NOP ; Processor will stop here and wait for write to complete.
INCF EEARL,F ; After write processor continues with 3rd instruction.
GOTO LOOP ; Still loading latches Increment address

START_WRITE
BCF EECON1,LWLO ; No more Latches only - Actually start write

MOVLW 55h ; Start of required write sequence:
MOVF EECON2 ; Write 55h
MOVLW 0AAh ;
BSF EECON2 ; Write AAh
NOP ; Set WR bit to begin write
NOP ; Any instructions here are ignored as processor
NOP ; halts to begin write sequence
NOP ; Processor will stop here and wait for write complete.
INCF EEARL,F ; after write processor continues with 3rd instruction
GOTO LOOP ; Write next latches

BCF EECON1,LWLO ; Disable writes
BSF INTCON,GIE ; Enable interrupts
11.4 Configuration Word and Device ID Access

Instead of accessing program memory or EEPROM data memory, the User ID's, Device ID/Revision ID and Configuration Words can be accessed when CFGS = 1. 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 11-1.

When read access is initiated on an unallowed address, the EEDATH:EEDATL registers are cleared. Writes can be disabled via the WRT Configuration bits. Refer to the Configuration Word 2.

### TABLE 11-1: PFM AND FUSE ACCESS VIA EECON1/EEDATH:EEDATL REGISTERS (WHEN CFGS = 1)

<table>
<thead>
<tr>
<th>Address</th>
<th>Function</th>
<th>Read Access</th>
<th>Write Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000h-8003h</td>
<td>User IDs</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>8006h</td>
<td>Device ID/Revision ID</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>8007h-8008h</td>
<td>Configuration Words 1 and 2</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

### EXAMPLE 11-3: CONFIGURATION WORD AND DEVICE ID ACCESS

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

```
BANKSEL EEADRL ; Select Bank 2
MOVLW PROG_ADDR_LO ;
MOVF EEADRL,W ; Store LSB of address
MOVLW PROG_ADDR_HI ;
MOVF EEADRH,W ; Store MSB of address
BCF EECON1,CFGS ; Deselect Configuration Space
BSF EECON1,EEPGD ; Select Program Memory
BCF INTCON,GIE ; Disable interrupts
BSF EECON1,RD ; Initiate read
NOP ; Executed (Figure 11-1)
NOP ; Ignored (Figure 11-1)
BSF INTCON,GIE ; Restore interrupts
MOVF EEDATL,W ; Get LSB of word
MOVWF PROG_DATA_LO ; Store in user location
MOVF EEDATH,W ; Get MSB of word
MOVWF PROG_DATA_HI ; Store in user location
```
11.5 Write Verify

Depending on the application, good programming practice may dictate that the value written to the data EEPROM or program memory should be verified (see Example 11-6) to the desired value to be written.

**EXAMPLE 11-6: WRITE VERIFY**

```
BANKSEL EEDATL ;
MOVF EEDATL, W ;EEDATL not changed
;from previous write
BSF EECON1, RD ;YES, Read the
 ;value written
XORWF EEDATL, W ;
BTFSS STATUS, Z ;Is data the same
GOTO WRITE_ERR ;No, handle error
; ;Yes, continue
```

11.5.1 USING THE DATA EEPROM

The data EEPROM is a high-endurance, byte addressable array that has been optimized for the storage of frequently changing information (e.g., program variables or other data that are updated often). When variables in one section change frequently, while variables in another section do not change, it is possible to exceed the total number of write cycles to the EEPROM (specification D124) without exceeding the total number of write cycles to a single byte (specifications D120 and D120A). If this is the case, then a refresh of the array must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in Flash program memory.

11.6 Protection Against Spurious Write

There are conditions when the user may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built-in. On power-up, WREN is cleared. Also, the Power-up Timer (64 ms duration) prevents EEPROM write.

The write initiate sequence and the WREN bit together help prevent an accidental write during:
- Brown-out
- Power Glitch
- Software Malfunction

11.7 Data EEPROM Operation During Code-Protect

Data memory can be code-protected by programming the CPD bit in the Configuration Word 1 to ‘0’.

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

**TABLE 11-2: SUMMARY OF REGISTERS ASSOCIATED WITH DATA EEPROM**

<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>EECON1</td>
<td>EEPGD</td>
<td>CFGS</td>
<td>LWLO</td>
<td>FREE</td>
<td>WRERR</td>
<td>WREN</td>
<td>WR</td>
<td>RD</td>
<td>107</td>
</tr>
<tr>
<td>EECON2</td>
<td>EEEPROM Control Register 2 (not a physical register)</td>
<td>108*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEADR1</td>
<td>EEADRL7</td>
<td>EEADRL6</td>
<td>EEADRL5</td>
<td>EEADRL4</td>
<td>EEADRL3</td>
<td>EEADRL2</td>
<td>EEADRL1</td>
<td>EEADRL0</td>
<td>106</td>
</tr>
<tr>
<td>EEADR2</td>
<td>EEADRH6</td>
<td>EEADRH5</td>
<td>EEADRH4</td>
<td>EEADRH3</td>
<td>EEADRH2</td>
<td>EEADRH1</td>
<td>EEADRH0</td>
<td></td>
<td>106</td>
</tr>
<tr>
<td>EEADTL</td>
<td>EEDATL7</td>
<td>EEDATL6</td>
<td>EEDATL5</td>
<td>EEDATL4</td>
<td>EEDATL3</td>
<td>EEDATL2</td>
<td>EEDATL1</td>
<td>EEDATL0</td>
<td>106</td>
</tr>
<tr>
<td>EEADTH</td>
<td>EEDATH5</td>
<td>EEDATH4</td>
<td>EEDATH3</td>
<td>EEDATH2</td>
<td>EEDATH1</td>
<td>EEDATH0</td>
<td></td>
<td></td>
<td>106</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>89</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSFIE</td>
<td>C2IE</td>
<td>C1IE</td>
<td>EEIE</td>
<td>BCL1IE</td>
<td></td>
<td></td>
<td></td>
<td>91</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSFIF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>EEIF</td>
<td>BCL1IF</td>
<td></td>
<td></td>
<td></td>
<td>95</td>
</tr>
</tbody>
</table>

Legend: — unimplemented read as ‘0’. Shaded cells are not used by Data EEPROM module.

* Page provides register information.
12.0 I/O PORTS

Depending on the device selected and peripherals enabled, there are two ports available. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Each port has three registers for its operation. These registers are:
- TRISx registers (data direction register)
- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)

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

A write operation to the LATx register has the same affect as a write to the corresponding PORTx register. A read of the LATx register reads of the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

Ports with analog functions also have an ANSELx register which can disable the digital input and save power. A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 12-1.

12.1 Alternate Pin Function

The Alternate Pin Function Control (APFCONx) registers are used to steer specific peripheral input and output functions between different pins. The APFCONx registers are shown in Register 12-1 and Register 12-2. For this device family, the following functions can be moved between different pins.
- RX/DT
- SDO1
- SS1 (Slave Select 1)
- P2B
- CCP2/P2A
- P1D
- P1C
- CCP1/P1A
- TX/CK

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.
REGISTER 12-1: APFCON0: ALTERNATE PIN FUNCTION CONTROL REGISTER 0

<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>RXDTSEL</td>
<td>SDO1SEL</td>
<td>SS1SEL</td>
<td>P2BSEL</td>
<td>CCP2SEL</td>
<td>P1DSEL</td>
<td>P1CSEL</td>
<td>CCP1SEL</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

- RXDTSEL: Pin Selection bit
  0 = RX/DT function is on RB1
  1 = RX/DT function is on RB2

- SDO1SEL: Pin Selection bit
  0 = SDO1 function is on RB2
  1 = SDO1 function is on RA6

- SS1SEL: Pin Selection bit
  0 = SS1 function is on RB5
  1 = SS1 function is on RA5

- P2BSEL: Pin Selection bit
  0 = P2B function is on RB7
  1 = P2B function is on RA6

- CCP2SEL: Pin Selection bit
  0 = CCP2/P2A function is on RB6
  1 = CCP2/P2A function is on RA7

- P1DSEL: Pin Selection bit
  0 = P1D function is on RB7
  1 = P1D function is on RA6

- P1CSEL: Pin Selection bit
  0 = P1C function is on RB6
  1 = P1C function is on RA7

- CCP1SEL: Pin Selection bit
  0 = CCP1/P1A function is on RB3
  1 = CCP1/P1A function is on RB0

Note 1: PIC16F/LF1827 only.

REGISTER 12-2: APFCON1: ALTERNATE PIN FUNCTION CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-0</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

- TXCKSEL: Pin Selection bit
  0 = TX/CK function is on RB2
  1 = TX/CK function is on RB5

Note 1: PIC16F/LF1827 only.
12.2 PORTA Registers

PORTA is a 8-bit wide, bidirectional port. The corresponding data direction register is TRISA (Register 12-4). 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 RA5, which is input only and its TRIS bit will always read as ‘1’. Example 12-1 shows how to initialize PORTA.

Reading the PORTA register (Register 12-3) 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).

The TRISA register (Register 12-4) 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’.

**Note:** The ANSELA register must be initialized to configure an analog channel as a digital input. Pins configured as analog inputs will read ‘0’.

### EXAMPLE 12-1: INITIALIZING PORTA

```
BANKSEL PORTA ;
CLRF PORTA ;Init PORTA
BANKSEL LATA ;Data Latch
CLRF LATA ;
BANKSEL ANSELA ;
CLRF ANSELA ;digital I/O
BANKSEL TRISA ;
MOVLW 0Ch ;Set RA<3:2> as inputs
MOVWF TRISA ; and set RA<7:4,1:0> as outputs
```

### REGISTER 12-3: PORTA: PORTA REGISTER

<table>
<thead>
<tr>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA7</td>
<td>RA6</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</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:** Writes to PORTA are actually written to corresponding LATA register. Reads from PORTA register is return of actual I/O pin values.
REGISTER 12-4: TRISA: PORTA TRI-STATE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>TRISA7</th>
<th>TRISA6</th>
<th>TRISA5</th>
<th>TRISA4</th>
<th>TRISA3</th>
<th>TRISA2</th>
<th>TRISA1</th>
<th>TRISA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</td>
<td>1/1</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

bit 7-6  TRISA<7:6>: PORTA Tri-State Control bit
1 = PORTA pin configured as an input (tri-stated)
0 = PORTA pin configured as an output

bit 5  TRISA5: RA5 Port Tri-State Control bit
This bit is always '1' as RA5 is an input only

bit 4-0  TRISA<4:0>: PORTA Tri-State Control bit
1 = PORTA pin configured as an input (tri-stated)
0 = PORTA pin configured as an output

REGISTER 12-5: LATA: PORTA DATA LATCH REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>LATA7</th>
<th>LATA6</th>
<th>LATA5</th>
<th>LATA4</th>
<th>LATA3</th>
<th>LATA2</th>
<th>LATA1</th>
<th>LATA0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>U-0</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</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  LATA<7:6>: RA<7:6> Output Latch Value bits(1)
bit 5  Unimplemented: Read as '0'
bit 4-0  LATA<4:0>: RA<4:0> Output Latch Value bits(1)

Note: Writes to PORTA are actually written to corresponding LATA register. Reads from PORTA register is return of actual I/O pin values.
### REGISTER 12-6: WPUA: WEAK PULL-UP PORTA REGISTER

<table>
<thead>
<tr>
<th></th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-1/1</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>—</td>
<td>—</td>
<td>WPUA5</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 7-6**

- **Unimplemented**: Read as ‘0’

**bit 5**

- **WPUA5**: Weak Pull-up RA5 Control bit
  - If **MCLRE** in Configuration Word 1 = 0, MCLR is disabled:
    - 1 = Weak Pull-up enabled\(^{()1}\)
    - 0 = Weak Pull-up disabled
  - If **MCLRE** in Configuration Word 1 = 1, MCLR is enabled:
    - Weak Pull-up is always enabled.

**bit 4-0**

- **Unimplemented**: Read as ‘0’

**Note 1:**

- Global **WPUEN** bit of the OPTION register must be cleared for individual pull-ups to be enabled.

**Note 2:**

- The weak pull-up device is automatically disabled if the pin is configured as an output.
12.2.1 ANSELA REGISTER

The ANSELA register (Register 12-7) 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 affect 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.

The TRISA register (Register 12-4) 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’.

Note: The ANSELA register must be initialized to configure an analog channel as a digital input. Pins configured as analog inputs will read ‘0’.

REGISTER 12-7: ANSELA: PORTA ANALOG SELECT REGISTER

| bit 7-5 | Unimplemented: Read as ‘0’ |
| bit 4-0 | ANSA4: Analog Select between Analog or Digital Function on pins RA4, respectively |
|        | Digital I/O. Pin is assigned to port or digital special function. |
|        | Analog input. Pin is assigned as analog input(1). Digital input buffer disabled. |

Note 1: When setting a pin to an analog input, the corresponding TRIS bit must be set to Input mode in order to allow external control of the voltage on the pin.
12.2.2 PORTA FUNCTIONS AND OUTPUT PRIORITIES

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

When multiple outputs are enabled, the actual pin control goes to the peripheral with the lowest number in the following lists.

Analog input functions, such as ADC, comparator and CapSense inputs, are not shown in the priority lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx registers. Digital output functions may control the pin when it is in Analog mode with the priority shown below.

**RA0**
1. SDO2 (PIC16F/LF1827 only)
2. RA0

**RA1**
1. SS2 (PIC16F/LF1827 only)
2. RA1

**RA2**
1. DACOUT (DAC)
2. RA2

**RA3**
1. SRQ (SR Latch)
2. CCP3 (PIC16F/LF1827 only)
3. C1OUT (Comparator)
4. RA3

**RA4**
1. SRNQ (SR Latch)
2. CCP4 (PIC16F/LF1827 only)
3. T0CKI
4. C2OUT (Comparator)
5. RA4

**RA5**
Input only pin.

**RA6**
1. OSC2 (enabled by Configuration Word)
2. CLKOUT
3. CLKR
4. SDO1
5. P1D
6. P2B (PIC16F/LF1827 only)
7. RA6

**RA7**
1. OSC1/CLkin (enabled by Configuration Word)
2. P1C
3. CCP2 (PIC16F/LF1827 only)
4. P2A (PIC16F/LF1827 only)
5. RA7
### TABLE 12-1: 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>—</td>
<td>—</td>
<td>ANSA4</td>
<td>ANSA3</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
</tr>
<tr>
<td>LATA</td>
<td>LATA7</td>
<td>LATA6</td>
<td>—</td>
<td>—</td>
<td>LATA4</td>
<td>LATA3</td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
</tr>
<tr>
<td>OPTION_REG</td>
<td>WPUEN</td>
<td>INTEDG</td>
<td>TMR0CS</td>
<td>TMR0SE</td>
<td>PSA</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
<td></td>
</tr>
<tr>
<td>PORTA</td>
<td>RA7</td>
<td>RA6</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
<td></td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td></td>
</tr>
<tr>
<td>WPUA</td>
<td>—</td>
<td>—</td>
<td>WPUA5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

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

### TABLE 12-2: SUMMARY OF CONFIGURATION WORD ASSOCIATED 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>FCMEN</td>
<td>IESO</td>
<td>CLKOUTEN</td>
<td>BOREN1</td>
<td>BOREN0</td>
<td>CPD</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>7:0</td>
<td>CP</td>
<td>MCLRE</td>
<td>PWRTIE</td>
<td>WDTE1</td>
<td>WDTE0</td>
<td>FOSC2</td>
<td>FOSC1</td>
<td>FOSC0</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used by PORTA.
12.3 PORTB and TRISB Registers

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB (Register 12-9). Setting a TRISB bit ( = 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISB bit ( = 0) will make the corresponding PORTB pin an output (i.e., enable the output driver and put the contents of the output latch on the selected pin). Example 12-2 shows how to initialize PORTB.

Reading the PORTB register (Register 12-8) 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.

The TRISB register (Register 12-9) controls the PORTB pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISB register are maintained set when using them as analog inputs. I/O pins configured as analog input always read ‘0’. Example 12-2 shows how to initialize PORTB.

EXAMPLE 12-2: INITIALIZING PORTB

```
BANKSEL PORTB  ;
CLRF PORTB ; Init PORTB
BANKSEL ANSELB
CLRF ANSELB ; Make RB<7:0> digital
BANKSEL TRISB ;
MOVLW B’11110000’ ; Set RB<7:4> as inputs
                 ; and RB<3:0> as outputs
MOVWF TRISB ;
```

12.3.1 WEAK PULL-UPS

Each of the PORTB pins has an individually configurable internal weak pull-up. Control bits WPUB<7:0> enable or disable each pull-up (see Register 12-11). Each weak pull-up is automatically turned off when the port pin is configured as an output. All pull-ups are disabled on a Power-on Reset by the WPUEN bit of the OPTION register.

12.3.2 INTERRUPT-ON-CHANGE

All of the PORTB pins are individually configurable as an interrupt-on-change pin. Control bits IOCB<7:0> enable or disable the interrupt function for each pin. The interrupt-on-change feature is disabled on a Power-on Reset. Reference Section 13.0 “Interrupt-On-Change” for more information.

**Note:** The ANSELB register must be initialized to configure an analog channel as a digital input. Pins configured as analog inputs will read ‘0’.
REGISTER 12-8: PORTB: PORTB REGISTER

<table>
<thead>
<tr>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
<th>R/W-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB7</td>
<td>RB6</td>
<td>RB5</td>
<td>RB4</td>
<td>RB3</td>
<td>RB2</td>
<td>RB1</td>
<td>RB0</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’ = Value at POR and BOR/Value at all other Resets

bit 7-0  

RB<7:0>: PORTB I/O Pin bit
1 = Port pin is > VIH  
0 = Port pin is < VIL

REGISTER 12-9: TRISB: PORTB TRI-STATE 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>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</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’ = Value at POR and BOR/Value at all other Resets

bit 7-0  

TRISB<7:0>: PORTB Tri-State Control bit
1 = PORTB pin configured as an input (tri-stated)  
0 = PORTB pin configured as an output

REGISTER 12-10: LATB: PORTB DATA LATCH 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>
</tr>
</thead>
<tbody>
<tr>
<td>LATB7</td>
<td>LATB6</td>
<td>LATB5</td>
<td>LATB4</td>
<td>LATB3</td>
<td>LATB2</td>
<td>LATB1</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’ = Value at POR and BOR/Value at all other Resets

bit 7-0  

LATB<7:0>: PORTB Output Latch Value bits\(^{(1)}\)

Note 1: Writes to PORTB are actually written to corresponding LATB register. Reads from PORTB register is return of actual I/O pin values.
**REGISTER 12-11: WPUB: WEAK PULL-UP PORTB REGISTER**

<table>
<thead>
<tr>
<th></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>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WPUB7</td>
<td>WPUB6</td>
<td>WPUB5</td>
<td>WPUB4</td>
<td>WPUB3</td>
<td>WPUB2</td>
<td>WPUB1</td>
<td>WPUB0</td>
<td>bit 7</td>
</tr>
<tr>
<td>bit 7-0</td>
<td>WPUB&lt;7:0&gt;: Weak Pull-up Register bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Pull-up enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Pull-up disabled</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

**Note 1:** Global WPUEN bit of the OPTION register must be cleared for individual pull-ups to be enabled.

**Note 2:** The weak pull-up device is automatically disabled if the pin is configured as an output.
12.3.3 ANSELB REGISTER

The ANSELB register (Register 12-12) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELB bit high will cause all digital reads on the pin to be read as ‘0’ and allow analog functions on the pin to operate correctly.

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

The TRISB register (Register 12-9) controls the PORTB pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISB register are maintained set when using them as analog inputs. I/O pins configured as analog input always read ‘0’.

**Note:** The ANSELB register must be initialized to configure an analog channel as a digital input. Pins configured as analog inputs will read ‘0’.

**REGISTER 12-12: ANSELB: PORTB ANALOG SELECT 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>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANSB7</td>
<td>ANSB6</td>
<td>ANSB5</td>
<td>ANSB4</td>
<td>ANSB3</td>
<td>ANSB2</td>
<td>ANSB1</td>
</tr>
<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

- **‘1’** = Analog Select between Analog or Digital Function on Pins RB<7:1>, respectively
  - 0 = Digital I/O. Pin is assigned to port or digital special function.
  - 1 = Analog input. Pin is assigned as analog input[^1]. Digital input buffer disabled.

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

**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.
12.3.4 PORTB FUNCTIONS AND OUTPUT PRIORITIES

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

When multiple outputs are enabled, the actual pin control goes to the peripheral with the lowest number in the following lists.

Analog input and some digital input functions are not included in the list below. These input functions can remain active when the pin is configured as an output. Certain digital input functions, such as the EUSART RX signal, override other port functions and are included in the priority list.

**RB0**
1. P1A
2. RB0

**RB1**
1. SDA1
2. RX/DT
3. RB1

**RB2**
1. SDA2 (PIC16F/LF1827 only)
2. TX/CK
3. RX/DT
4. SDO1
5. RB2

**RB3**
1. MDOUT
2. CCP1/P1A
3. RB3

**RB4**
1. SCL1
2. SCK1
3. RB4

**RB5**
1. SCL2 (PIC16F/LF1827 only)
2. TX/CK
3. SCK2 (PIC16F/LF1827 only)
4. P1B
5. RB5

**RB6**
1. ICSPCLK (Programming)
2. T1OSI
3. P1C
4. CCP2 (PIC16F/LF1827 only)
5. P2A (PIC16F/LF1827 only)
6. RB6

**RB7**
1. ICSPDAT (Programming)
2. T1OSO
3. P1D
4. P2B (PIC16F/LF1827 only)
5. RB7

---

**TABLE 12-3: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB**

<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>ANSELB</td>
<td>ANSB7</td>
<td>ANSB6</td>
<td>ANSB5</td>
<td>ANSB4</td>
<td>ANSB3</td>
<td>ANSB2</td>
<td>ANSB1</td>
<td>—</td>
<td>128</td>
</tr>
<tr>
<td>LATB</td>
<td>LATB7</td>
<td>LATB6</td>
<td>LATB5</td>
<td>LATB4</td>
<td>LATB3</td>
<td>LATB2</td>
<td>LATB1</td>
<td>LATB0</td>
<td>126</td>
</tr>
<tr>
<td>OPTION_REG</td>
<td>WPUEN</td>
<td>INTEDG</td>
<td>TMROCS</td>
<td>TMROSE</td>
<td>PSA</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
<td>175</td>
</tr>
<tr>
<td>PORTB</td>
<td>RB7</td>
<td>RB6</td>
<td>RB5</td>
<td>RB4</td>
<td>RB3</td>
<td>RB2</td>
<td>RB1</td>
<td>RB0</td>
<td>126</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
<tr>
<td>WPUB</td>
<td>WPUB7</td>
<td>WPUB6</td>
<td>WPUB5</td>
<td>WPUB4</td>
<td>WPUB3</td>
<td>WPUB2</td>
<td>WPUB1</td>
<td>WPUB0</td>
<td>127</td>
</tr>
</tbody>
</table>

**Legend:**
— = unimplemented locations read as ‘0’. Shaded cells are not used by PORTB.
13.0 INTERRUPT-ON-CHANGE

The PORTB pins can be configured to operate as Interrupt-On-Change (IOC) pins. An interrupt can be generated by detecting a signal that has either a rising edge or a falling edge. Any individual PORTB pin, or combination of PORTB 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 13-1 is a block diagram of the IOC module.

13.1 Enabling the Module

To allow individual PORTB 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.

13.2 Individual Pin Configuration

For each PORTB pin, a rising edge detector and a falling edge detector are present. To enable a pin to detect a rising edge, the associated IOCBPx bit of the IOCBP register is set. To enable a pin to detect a falling edge, the associated IOCBNx bit of the IOCBN register is set.

A pin can be configured to detect rising and falling edges simultaneously by setting both the IOCBPx bit and the IOCBNx bit of the IOCBP and IOCBN registers, respectively.

13.3 Interrupt Flags

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

13.4 Clearing Interrupt Flags

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

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

**EXAMPLE 13-1:**

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

13.5 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 IOCBF register will be updated prior to the first instruction executed out of Sleep.
## REGISTER 13-1: IOCBP: INTERRUPT-ON-CHANGE POSITIVE EDGE REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>IOCBP&lt;7:0&gt;: Interrupt-on-Change Positive Edge Enable bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt-on-change enabled on the pin for a positive going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt-on-change disabled for the associated pin</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

### Example:

<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>IOCBP7</td>
<td>IOCBP6</td>
<td>IOCBP5</td>
<td>IOCBP4</td>
<td>IOCBP3</td>
<td>IOCBP2</td>
<td>IOCBP1</td>
<td>IOCBP0</td>
</tr>
</tbody>
</table>

### Bit Descriptions:
- **bit 7**
- **IOCBP7**
- **bit 0**


## REGISTER 13-2: IOCBN: INTERRUPT-ON-CHANGE NEGATIVE EDGE REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>IOCBN&lt;7:0&gt;: Interrupt-on-Change Negative Edge Enable bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interrupt-on-change enabled on the pin for a negative going edge. Associated Status bit and interrupt flag will be set upon detecting an edge.</td>
</tr>
<tr>
<td>0</td>
<td>Interrupt-on-change disabled for the associated pin</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

### Example:

<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>IOCBN7</td>
<td>IOCBN6</td>
<td>IOCBN5</td>
<td>IOCBN4</td>
<td>IOCBN3</td>
<td>IOCBN2</td>
<td>IOCBN1</td>
<td>IOCBN0</td>
</tr>
</tbody>
</table>

### Bit Descriptions:
- **bit 7**
- **IOCBN7**
- **bit 0**


## REGISTER 13-3: IOCBF: INTERRUPT-ON-CHANGE FLAG REGISTER

<table>
<thead>
<tr>
<th>bit 7-0</th>
<th>IOCBF&lt;7:0&gt;: Interrupt-on-Change Flag bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>An enabled change was detected on the associated pin. Set when IOCBPx = 1 and a rising edge was detected on RBx, or when IOCBNx = 1 and a falling edge was detected on RBx.</td>
</tr>
<tr>
<td>0</td>
<td>No change was detected, or the user cleared the detected change</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

### Example:

<table>
<thead>
<tr>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
<th>R/W/HS-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOCBF7</td>
<td>IOCBF6</td>
<td>IOCBF5</td>
<td>IOCBF4</td>
<td>IOCBF3</td>
<td>IOCBF2</td>
<td>IOCBF1</td>
<td>IOCBF0</td>
</tr>
</tbody>
</table>

### Bit Descriptions:
- **bit 7**
- **IOCBF7**
- **bit 0**

---

**PIC16F/LF1826/27**

© 2009 Microchip Technology Inc.
TABLE 13-1: 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>ANSELB</td>
<td>ANSB7</td>
<td>ANSB6</td>
<td>ANSB5</td>
<td>ANSB4</td>
<td>ANSB3</td>
<td>ANSB2</td>
<td>ANSB1</td>
<td>—</td>
<td>128</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMROIE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>89</td>
</tr>
<tr>
<td>IOCBF</td>
<td>IOCBF7</td>
<td>IOCBF6</td>
<td>IOCBF5</td>
<td>IOCBF4</td>
<td>IOCBF3</td>
<td>IOCBF2</td>
<td>IOCBF1</td>
<td>IOCBF0</td>
<td>132</td>
</tr>
<tr>
<td>IOCBN</td>
<td>IOCBN7</td>
<td>IOCBN6</td>
<td>IOCBN5</td>
<td>IOCBN4</td>
<td>IOCBN3</td>
<td>IOCBN2</td>
<td>IOCBN1</td>
<td>IOCBN0</td>
<td>132</td>
</tr>
<tr>
<td>IOCBP</td>
<td>IOCBP7</td>
<td>IOCBP6</td>
<td>IOCBP5</td>
<td>IOCBP4</td>
<td>IOCBP3</td>
<td>IOCBP2</td>
<td>IOCBP1</td>
<td>IOCBP0</td>
<td>132</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented locations read as ‘0’. Shaded cells are not used by interrupt-on-change.
14.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of VDD, with 1.024V, 2.048V or 4.096V selectable output levels. The output of the FVR can be configured to supply a reference voltage to the following:

- ADC input channel
- ADC positive reference
- Comparator positive input
- Digital-to-Analog Converter (DAC)

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

14.1 Independent Gain Amplifiers

The output of the FVR supplied to the ADC, Comparators, and DAC is routed through two independent programmable gain amplifiers. Each amplifier can be configured to amplify the reference voltage by 1x, 2x or 4x, to produce the three possible voltage levels.

The ADFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module. Reference Section 15.0 “Analog-to-Digital Converter (ADC) Module” for additional information.

The CDAFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the DAC and comparator module. Reference Section 16.0 “Digital-to-Analog Converter (DAC) Module” and Section 17.0 “Comparator Module” for additional information.

14.2 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 29.0 “Electrical Specifications” for the minimum delay requirement.

FIGURE 14-1: VOLTAGE REFERENCE BLOCK DIAGRAM

![Voltage Reference Block Diagram](image-url)
**REGISTER 14-1: FVRCON: FIXED VOLTAGE REFERENCE CONTROL REGISTER**

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R-q/q</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>FVREN</td>
<td>FVRRDY&lt;1&gt;</td>
<td>Reserved</td>
<td>Reserved</td>
<td>CDAFVR1</td>
<td>CDAFVR0</td>
<td>ADFVR1</td>
<td>ADFVR0</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  
**FVREN:** Fixed Voltage Reference Enable bit
0 = Fixed Voltage Reference is disabled
1 = Fixed Voltage Reference is enabled

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

bit 5-4  
Reserved: Read as '0'. Maintain these bits clear.

bit 3-2  
**CDAFVR<1:0>:** Comparator and DAC Fixed Voltage Reference Selection bit
00 = Comparator and DAC Fixed Voltage Reference Peripheral output is off.
01 = Comparator and DAC Fixed Voltage Reference Peripheral output is 1x (1.024V)
10 = Comparator and DAC Fixed Voltage Reference Peripheral output is 2x (2.048V)<sup>(2)</sup>
11 = Comparator and DAC Fixed Voltage Reference Peripheral output is 4x (4.096V)<sup>(2)</sup>

bit 1-0  
**ADFVR<1:0>:** ADC Fixed Voltage Reference Selection bit
00 = ADC Fixed Voltage Reference Peripheral output is off.
01 = ADC Fixed Voltage Reference Peripheral output is 1x (1.024V)
10 = ADC Fixed Voltage Reference Peripheral output is 2x (2.048V)<sup>(2)</sup>
11 = ADC Fixed Voltage Reference Peripheral output is 4x (4.096V)<sup>(2)</sup>

**Note 1:**  
FVRRDY is always '1' on devices with the LDO (PIC16F1826/27).

**2:** Fixed Voltage Reference output cannot exceed VDD.

**TABLE 14-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE FVR MODULE**

<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>Reserved</td>
<td>Reserved</td>
<td>CDAFVR1</td>
<td>CDAFVR0</td>
<td>ADFVR1</td>
<td>ADFVR0</td>
<td>136</td>
</tr>
</tbody>
</table>

Legend:  
- Shaded cells are unused by the FVR 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 register (ADRES). 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.

FIGURE 15-1: ADC BLOCK DIAGRAM

Note: When ADON = 0, all multiplexer inputs are disconnected.
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 12.0 “I/O Ports” for more information.

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

15.1.2 CHANNEL SELECTION

There are 14 channel selections available:

- AN<11:0> pins
- DAC Output
- FVR (Fixed Voltage Reference) Output

Refer to Section 16.0 “Digital-to-Analog Converter (DAC) Module” and Section 14.0 “Fixed Voltage Reference (FVR)” 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 15.2 “ADC 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

The ADNREF bits of the ADCON1 register provides control of the negative voltage reference. The negative voltage reference can be:

- VREF- pin
- VSS

See Section 14.0 “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 A/D conversion requirements in Section 29.0 “Electrical Specifications” for more information. Table 15-1 gives examples of appropriate ADC clock selections.

Note: Unless using the FRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.
### 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>32 MHz</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>62.5 ns</td>
<td>100 ns</td>
<td>125 ns</td>
<td>250 ns</td>
<td>500 ns</td>
<td>2.0 μs</td>
</tr>
<tr>
<td>Fosc/4</td>
<td>100</td>
<td>125 ns</td>
<td>200 ns</td>
<td>250 ns</td>
<td>500 ns</td>
<td>1.0 μs</td>
<td>4.0 μs</td>
</tr>
<tr>
<td>Fosc/8</td>
<td>001</td>
<td>0.5 μs</td>
<td>400 ns</td>
<td>0.5 μs</td>
<td>1.0 μs</td>
<td>4.0 μs</td>
<td>8.0 μs</td>
</tr>
<tr>
<td>Fosc/16</td>
<td>101</td>
<td>800 ns</td>
<td>800 ns</td>
<td>2.0 μs</td>
<td>4.0 μs</td>
<td>8.0 μs</td>
<td>16.0 μs</td>
</tr>
<tr>
<td>Fosc/32</td>
<td>010</td>
<td>1.0 μs</td>
<td>1.6 μs</td>
<td>2.0 μs</td>
<td>4.0 μs</td>
<td>8.0 μs</td>
<td>32.0 μs</td>
</tr>
<tr>
<td>Fosc/64</td>
<td>110</td>
<td>2.0 μs</td>
<td>3.2 μs</td>
<td>4.0 μs</td>
<td>8.0 μs</td>
<td>16.0 μs</td>
<td>64.0 μs</td>
</tr>
<tr>
<td>FRC</td>
<td>x11</td>
<td>1.0-6.0 μs</td>
<td>1.0-6.0 μs</td>
<td>1.0-6.0 μs</td>
<td>1.0-6.0 μs</td>
<td>1.0-6.0 μs</td>
<td>1.0-6.0 μs</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.
2. These values violate the minimum required TAD time.
3. For faster conversion times, the selection of another clock source is recommended.
4. When the device frequency is greater than 1 MHz, the FRC clock source is only recommended if the conversion will be performed during Sleep.

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

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.

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.

Please refer to Section 8.0 “Interrupts” for more information.

15.1.6 RESULT FORMATTING

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

Figure 15-3 shows the two output formats.

---

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

**Note 2:** The ADC operates during Sleep only when the FrC oscillator is selected.
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 “A/D 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 of the CCPx/ECCPX module allows periodic ADC measurements without software intervention. When this trigger occurs, the GO/DONE bit is set by hardware and the Timer1 counter resets to zero.

**TABLE 15-2: SPECIAL EVENT TRIGGER**

<table>
<thead>
<tr>
<th>Device</th>
<th>CCPx/ECCPX</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC16F/LF1826</td>
<td>ECCP1</td>
</tr>
<tr>
<td>PIC16F/LF1827</td>
<td>CCP4</td>
</tr>
</tbody>
</table>

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 23.0 “Capture/Compare/PWM Modules (ECCP1, ECCP2, CCP3, CCP4)” for more information.
15.2.6 A/D CONVERSION PROCEDURE

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

1. **Configure Port:**
   - Disable pin output driver (Refer to the TRIS register)
   - Configure pin as analog (Refer to the ANSEL register)

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

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

4. **Wait the required acquisition time (2).**

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

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

7. **Read ADC Result.**

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

---

**EXAMPLE 15-1: A/D CONVERSION**

```assembly
; 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 ;
MOVF ADRESH, W ; Read upper 2 bits
MOVWF RESULTHI ; Store in GPR space
BANKSEL ADRESH ;
MOVF ADRESL, W ; Read lower 8 bits
MOVWF RESULTLO ; 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 15.3 “A/D Acquisition Requirements”.

---

15.2.6 A/D CONVERSION PROCEDURE

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

1. **Configure Port:**
   - Disable pin output driver (Refer to the TRIS register)
   - Configure pin as analog (Refer to the ANSEL register)

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

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

4. **Wait the required acquisition time (2).**

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

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

7. **Read ADC Result.**

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

---

**EXAMPLE 15-1: A/D CONVERSION**

```assembly
; 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 ;
MOVF ADRESH, W ; Read upper 2 bits
MOVWF RESULTHI ; Store in GPR space
BANKSEL ADRESH ;
MOVF ADRESL, W ; Read lower 8 bits
MOVWF RESULTLO ; 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 15.3 “A/D Acquisition Requirements”.

---
15.2.7 ADC REGISTER DEFINITIONS

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

REGISTER 15-1: ADCON0: A/D CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6-2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>CHS4</td>
<td>GO/DONE</td>
<td>ADON</td>
</tr>
</tbody>
</table>

<p>| bit 7 | Unimplemented: Read as ‘0’ |</p>
<table>
<thead>
<tr>
<th>bit 6-2</th>
<th>CHS&lt;4:0&gt;: Analog Channel Select bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000</td>
<td>AN0</td>
</tr>
<tr>
<td>00001</td>
<td>AN1</td>
</tr>
<tr>
<td>00010</td>
<td>AN2</td>
</tr>
<tr>
<td>00011</td>
<td>AN3</td>
</tr>
<tr>
<td>00100</td>
<td>AN4</td>
</tr>
<tr>
<td>00101</td>
<td>AN5</td>
</tr>
<tr>
<td>00110</td>
<td>AN6</td>
</tr>
<tr>
<td>00111</td>
<td>AN7</td>
</tr>
<tr>
<td>01000</td>
<td>AN8</td>
</tr>
<tr>
<td>01001</td>
<td>AN9</td>
</tr>
<tr>
<td>01010</td>
<td>AN10</td>
</tr>
<tr>
<td>01011</td>
<td>AN11</td>
</tr>
<tr>
<td>01100</td>
<td>Reserved. No channel connected.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>11101</td>
<td>Reserved. No channel connected.</td>
</tr>
<tr>
<td>11110</td>
<td>DAC output(1)</td>
</tr>
<tr>
<td>11111</td>
<td>FVR (Fixed Voltage Reference) Buffer 1 Output(2)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1</th>
<th>GO/DONE: A/D Conversion Status bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.</td>
</tr>
<tr>
<td></td>
<td>This bit is automatically cleared by hardware when the A/D conversion has completed.</td>
</tr>
<tr>
<td>0</td>
<td>A/D conversion completed/not in progress</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 0</th>
<th>ADON: ADC Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ADC is enabled</td>
</tr>
<tr>
<td>0</td>
<td>ADC is disabled and consumes no operating current</td>
</tr>
</tbody>
</table>

**Note 1:** See Section 16.0 “Digital-to-Analog Converter (DAC) Module” for more information.

**Note 2:** See Section 14.0 “Fixed Voltage Reference (FVR)” for more information.
# REGISTER 15-2: ADCON1: A/D Control Register 1

<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>ADFM</td>
<td>ADCS2</td>
<td>ADCS1</td>
<td>ADCS0</td>
<td>Unimplemented</td>
<td>ADNREF</td>
<td>ADPREF1</td>
<td>ADPREF0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Legend:</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = Readable bit</td>
</tr>
<tr>
<td>u = Bit is unchanged</td>
</tr>
<tr>
<td>‘1’ = Bit is set</td>
</tr>
</tbody>
</table>

**bit 7**  
**ADFM**: A/D 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>**: A/D 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**  
**Unimplemented**: Read as ‘0’

**bit 2**  
**ADNREF**: A/D Negative Voltage Reference Configuration bit  
0 = VREF- is connected to AVSS  
1 = VREF- is connected to external VREF-

**bit 1-0**  
**ADPREF<1:0>**: A/D Positive Voltage Reference Configuration bits  
00 = VREF+ is connected to AVDD  
01 = Reserved  
10 = VREF+ is connected to external VREF+  
11 = VREF+ is connected to internal fixed voltage reference
REGISTER 15-3:  ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 0

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</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>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>ADRES9</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>6</td>
<td>ADRES8</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>5</td>
<td>ADRES7</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>4</td>
<td>ADRES6</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>3</td>
<td>ADRES5</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>2</td>
<td>ADRES4</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>1</td>
<td>ADRES3</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>0</td>
<td>ADRES2</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</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<9:2>: ADC Result Register bits
Upper 8 bits of 10-bit conversion result

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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</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>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>ADRES1</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
<td>R/W-x/u</td>
</tr>
<tr>
<td>6</td>
<td>ADRES0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</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
- **'1'** = Bit is set
- **'0'** = Bit is cleared
- **-n/n** = Value at POR and BOR/Value at all other Resets

Bit 7-6 ADRES<1:0>: ADC Result Register bits
Lower 2 bits of 10-bit conversion result

Bit 5-0 **Reserved**: Do not use.
**REGISTER 15-5: ADRESH: 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>R/W-x/u</th>
<th>R/W-x/u</th>
<th>ADRESH</th>
<th>ADRES8</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<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
- **’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 2 bits of 10-bit conversion result

**REGISTER 15-6: ADRESL: 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>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRES7</td>
<td>ADRES6</td>
<td>ADRES5</td>
<td>ADRES4</td>
<td>ADRES3</td>
<td>ADRES2</td>
<td>ADRES1</td>
<td>ADRES0</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 8 bits of 10-bit conversion result
15.3 A/D Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 15-4. The source impedance (Rs) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to Figure 15-4. The maximum recommended impedance for analog sources is 10 kΩ. As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an A/D 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 10kΩ VDD

\[ T_{ACQ} = \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \]
\[ = T_{AMP} + T_{C} + T_{COFF} \]
\[ = 2\mu s + T_{C} + [(\text{Temperature} - 25°C)(0.05\mu s/°C)] \]

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

\[ V_{APPLIED}(1 - \frac{1}{2^n + 1}) = V_{CHOLD} \quad ; [1] \quad V_{CHOLD} \text{charged to within 1/2 lsb} \]

\[ V_{APPLIED}(1 - e^{-\frac{T_{C}}{RC}}) = V_{CHOLD} \quad ; [2] \quad V_{CHOLD} \text{charge response to } V_{APPLIED} \]

\[ V_{APPLIED}(1 - e^{-\frac{T_{C}}{RC}}) = V_{APPLIED}(1 - \frac{1}{(2^n + 1) - 1}) \quad ; \text{combining [1] and [2]} \]

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

Solving for \( T_{C} \):

\[ T_{C} = -CHOLD(R_{IC} + RSS + RS) \ln(1/511) \]
\[ = -10pF(1kΩ + 7kΩ + 10kΩ) \ln(0.001957) \]
\[ = 1.12\mu s \]

Therefore:

\[ T_{ACQ} = 2\mu s + 1.12\mu s + [(50°C - 25°C)(0.05\mu s/°C)] \]
\[ = 4.42\mu s \]

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

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

3: The maximum recommended impedance for analog sources is 10 kΩ. 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 29.0 “Electrical Specifications”.

FIGURE 15-5: ADC TRANSFER FUNCTION

Legend:

- ADC Output Code

Note 1: Refer to Section 29.0 “Electrical Specifications”.

Legend:

- F Bh = Full-Scale Range
- 1 LSB ideal
- Zero-Scale Transition
- 1 LSB ideal
- Analog Input Voltage
- VREF
- VSS
### Table 15-3: Summary of Registers Associated with ADC

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
<th>Bit 5</th>
<th>Bit 6</th>
<th>Bit 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCON0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CHS4</td>
</tr>
<tr>
<td>ADCON1</td>
<td>ADFM</td>
<td>ADFM</td>
<td>ADFM</td>
<td>ADFM</td>
<td>ADFM</td>
<td>ADFM</td>
<td>ADFM</td>
<td>ADFM</td>
</tr>
<tr>
<td>ADRESH</td>
<td>ADON</td>
<td>ADON</td>
<td>ADON</td>
<td>ADON</td>
<td>ADON</td>
<td>ADON</td>
<td>ADON</td>
<td>ADON</td>
</tr>
<tr>
<td>ADRESL</td>
<td>A/D Result Register Low</td>
<td>146*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANSELA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ANSELB</td>
<td>ANSB7</td>
<td>ANSB7</td>
<td>ANSB7</td>
<td>ANSB7</td>
<td>ANSB7</td>
<td>ANSB7</td>
<td>ANSB7</td>
<td>ANSB7</td>
</tr>
<tr>
<td>CCPxCON</td>
<td>PxM1</td>
<td>PxM0</td>
<td>PxM1</td>
<td>PxM0</td>
<td>PxM1</td>
<td>PxM0</td>
<td>PxM0</td>
<td>PxM1</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMR0IE</td>
<td>INTIE</td>
<td>IOCE</td>
<td>TMR0IE</td>
<td>INTF</td>
<td>IOCF</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
</tr>
<tr>
<td>FVRCON</td>
<td>FVREN</td>
<td>FVRRDY</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>CDAFVR1</td>
<td>CDAFVR0</td>
<td>ADFVR1</td>
</tr>
<tr>
<td>DACCON0</td>
<td>DACEN</td>
<td>DACLPS</td>
<td>DACOE</td>
<td>—</td>
<td>DACPSS1</td>
<td>DACPSS0</td>
<td>—</td>
<td>DACNSS</td>
</tr>
<tr>
<td>DACCON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DACR4</td>
<td>DACR3</td>
<td>DACR2</td>
<td>DACR1</td>
</tr>
</tbody>
</table>

#### Legend:
- — = unimplemented read as ‘0’. Shaded cells are not used for ADC module.
- * Page provides register information.
16.0 DIGITAL-TO-ANALOG CONVERTER (DAC) MODULE

The Digital-to-Analog Converter supplies a variable voltage reference, ratiometric with the input source, with 32 selectable output levels.

The input of the DAC can be connected to:
- External VREF pins
- VDD supply voltage
- FVR (Fixed Voltage Reference)

The output of the DAC can be configured to supply a reference voltage to the following:
- Comparator positive input
- ADC input channel
- DACOUT pin

The Digital-to-Analog Converter (DAC) can be enabled by setting the DACEN bit of the DACCON0 register.

16.1 Output Voltage Selection

The DAC has 32 voltage level ranges. The 32 levels are set with the DACR<4:0> bits of the DACCON1 register.

The DAC output voltage is determined by the following equations:

$$EQUATION \ 16-1: \ \ DAC \ OUTPUT \ VOLTAGE$$

$$VOUT = \left( \frac{(V_{SRC}+ - V_{SRC}-) \times DACR<4:0>}{32} \right) + V_{SRC}-$$

$$V_{SRC}+ = VDD, \ VREF+ \ or \ FVR1$$

$$V_{SRC}- = VSS \ or \ VREF-$$

16.2 Ratiometric Output Level

The DAC output value is derived using a resistor ladder with each end of the ladder tied to a positive and negative voltage reference input source. If the voltage of either input source fluctuates, a similar fluctuation will result in the DAC output value.

The value of the individual resistors within the ladder can be found in Section 29.0 “Electrical Specifications”.

16.3 Low-Power Voltage State

In order for the DAC module to consume the least amount of power, one of the two voltage reference input sources to the resistor ladder must be disconnected. Either the positive voltage source, (V_{SRC}+), or the negative voltage source, (V_{SRC}-) can be disabled.

The negative voltage source is disabled by setting the DACLPS bit in the DACCON0 register. Clearing the DACLPS bit in the DACCON0 register disables the positive voltage source.

16.4 Output Clamped to Positive Voltage Source

The DAC output voltage can be set to V_{SRC}+ with the least amount of power consumption by performing the following:
- Clearing the DACEN bit in the DACCON0 register.
- Clearing the DACLPS bit in the DACCON0 register.
- Configuring the DACPSS bits to the proper positive source.
- Configuring the DACRx bits to ‘11111’ in the DACCON1 register.

This is also the method used to output the voltage level from the FVR to an output pin. See Section 16.6 “DAC Voltage Reference Output” for more information.

16.5 Output Clamped to Negative Voltage Source

The DAC output voltage can be set to V_{SRC}- with the least amount of power consumption by performing the following:
- Clearing the DACEN bit in the DACCON0 register.
- Configuring the DACPSS bits to the proper negative source.
- Configuring the DACRx bits to ‘00000’ in the DACCON1 register.

This allows the comparator to detect a zero-crossing while not consuming additional current through the DAC module.

16.6 DAC Voltage Reference Output

The DAC can be output to the DACOUT pin by setting the DACOE bit of the DACCON0 register to ‘1’. Selecting the DAC reference voltage for output on the DACOUT pin automatically overrides the digital output buffer and digital input threshold detector functions of that pin. Reading the DACOUT pin when it has been configured for DAC reference voltage output will always return a ‘0’.

Due to the limited current drive capability, a buffer must be used on the DAC voltage reference output for external connections to DACOUT. Figure 16-2 shows an example buffering technique.
FIGURE 16-1: DIGITAL-TO-ANALOG CONVERTER BLOCK DIAGRAM

Digital-to-Analog Converter (DAC)

VREF-
DACNSS
DACNSS
VSS
VSS
VSRC-
DACNSS

FVR BUFFER2
VDD
VREF+

DACNSS<1:0>
DACEN
DACLPS

VREF-
VSS

32 Steps
32x1 MUX
(To Comparator and ADC Modules)

DAC
DACOE
DACOUT

FIGURE 16-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE

PIC® MCU

DAC Module

R
Voltage Reference Output Impedance

DACNSS
DACNSS

Buffered DAC Output
16.7 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the DACCON0 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

16.8 Effects of a Reset

A device Reset affects the following:

- DAC is disabled.
- DAC output voltage is removed from the DACOUT pin.
- The DAC1R<4:0> range select bits are cleared.
REGISTER 16-1: DACCON0: VOLTAGE REFERENCE CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
<th>R/W-0/0</th>
<th>U-0</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DACEN</td>
<td>DACLPS</td>
<td>DACOE</td>
<td>---</td>
<td>DACPSS1</td>
<td>DACPSS0</td>
<td>---</td>
<td>DACNSS</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 DACEN**: DAC Enable bit
  - 1 = DAC is enabled
  - 0 = DAC is disabled

- **bit 6 DACLPS**: DAC Low-Power Voltage State Select bit
  - 1 = DAC Positive reference source selected
  - 0 = DAC Negative reference source selected

- **bit 5 DACOE**: DAC Voltage Output Enable bit
  - 1 = DAC voltage level is also an output on the DACOUT pin
  - 0 = DAC voltage level is disconnected from the DACOUT pin

- **bit 4 Unimplemented**: Read as ‘0’

- **bit 3-2 DACPSS<1:0>**: DAC Positive Source Select bits
  - 00 = VDD
  - 01 = VREF+
  - 10 = FVR Buffer2 output
  - 11 = Reserved, do not use

- **bit 1 Unimplemented**: Read as ‘0’

- **bit 0 DACNSS**: DAC Negative Source Select bits
  - 1 = VREF-
  - 0 = VSS

REGISTER 16-2: DACCON1: VOLTAGE REFERENCE CONTROL REGISTER 1

<table>
<thead>
<tr>
<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>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>DACR4</td>
<td>DACR3</td>
<td>DACR2</td>
<td>DACR1</td>
<td>DACR0</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 7-5 Unimplemented**: Read as ‘0’

- **bit 4-0 DACR<4:0>**: DAC Voltage Output Select bits
  \[ V\text{OUT} = ((V\text{SRC}+) - (V\text{SRC}-)) \times (\text{DACR}<4:0>/(2^5)) + V\text{SRC}- \]

**Note 1**: The output select bits are always right justified to ensure that any number of bits can be used without affecting the register layout.
## TABLE 16-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE DAC MODULE

<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>FVRDY</td>
<td>Reserved</td>
<td>Reserved</td>
<td>CDAFVR1</td>
<td>CDAFVR0</td>
<td>ADFVR1</td>
<td>ADFVR0</td>
<td>136</td>
</tr>
<tr>
<td>DACCON0</td>
<td>DACEN</td>
<td>DACLPS</td>
<td>DACOE</td>
<td>—</td>
<td>DACPSS1</td>
<td>DACPSS0</td>
<td>—</td>
<td>DACNSS</td>
<td>154</td>
</tr>
<tr>
<td>DACCON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>Dacr4</td>
<td>Dacr3</td>
<td>Dacr2</td>
<td>Dacr1</td>
<td>Dacr0</td>
</tr>
</tbody>
</table>

**Legend:**  
— = unimplemented, read as '0'. Shaded cells are unused by the DAC module.
17.0 COMPARATOR MODULE

Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. Comparators are very useful mixed signal building blocks because they provide analog functionality independent of program execution. The analog comparator module includes the following features:

- Independent comparator control
- Programmable input selection
- Comparator output is available internally/externally
- Programmable output polarity
- Interrupt-on-change
- Wake-up from Sleep
- Programmable Speed/Power optimization
- PWM shutdown
- Programmable and fixed voltage reference

17.1 Comparator Overview

A single comparator is shown in Figure 17-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at VIN+ is less than the analog voltage at VIN-, the output of the comparator is a digital low level. When the analog voltage at VIN+ is greater than the analog voltage at VIN-, the output of the comparator is a digital high level.

Note: The black areas of the output of the comparator represent the uncertainty due to input offsets and response time.
FIGURE 17-2: COMPARATOR 1 MODULE SIMPLIFIED BLOCK DIAGRAM

Note 1: When CxON = 0, the Comparator will produce a '0' at the output
2: When CxON = 0, all multiplexer inputs are disconnected.
3: Output of comparator can be frozen during debugging.

Note

1: When CxON = 0, the Comparator will produce a '0' at the output
2: When CxON = 0, all multiplexer inputs are disconnected.
3: Output of comparator can be frozen during debugging.
FIGURE 17-3: COMPARATOR 2 MODULE SIMPLIFIED BLOCK DIAGRAM

Note 1: When CxON = 0, the Comparator will produce a '0' at the output
2: When CxON = 0, all multiplexer inputs are disconnected.
3: Output of comparator can be frozen during debugging.
17.2 Comparator Control

Each comparator has 2 control registers: CMxCON0 and CMxCON1.
The CMxCON0 registers (see Register 17-1) contain Control and Status bits for the following:
- Enable
- Output selection
- Output polarity
- Speed/Power selection
- Hysteresis enable
- Output synchronization
The CMxCON1 registers (see Register 17-2) contain Control bits for the following:
- Interrupt enable
- Interrupt edge polarity
- Positive input channel selection
- Negative input channel selection

17.2.1 COMPARATOR ENABLE

Setting the CxON bit of the CMxCON0 register enables the comparator for operation. Clearing the CxON bit disables the comparator resulting in minimum current consumption.

17.2.2 COMPARATOR OUTPUT SELECTION

The output of the comparator can be monitored by reading either the CxOUT bit of the CMxCON0 register or the MCxOUT bit of the CMOUT register. In order to make the output available for an external connection, the following conditions must be true:
- CxOE bit of the CMxCON0 register must be set
- Corresponding TRIS bit must be cleared
- CxON bit of the CMxCON0 register must be set

17.2.3 COMPARATOR OUTPUT POLARITY

Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the CxPOL bit of the CMxCON0 register. Clearing the CxPOL bit results in a non-inverted output.

Table 17-1 shows the output state versus input conditions, including polarity control.

TABLE 17-1: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS

<table>
<thead>
<tr>
<th>Input Condition</th>
<th>CxPOL</th>
<th>CxOUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CxVN &gt; CxVP</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CxVN &lt; CxVP</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CxVN &gt; CxVP</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CxVN &lt; CxVP</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

17.2.4 COMPARATOR SPEED/POWER SELECTION

The trade-off between speed or power can be optimized during program execution with the CxSP control bit. The default state for this bit is ‘1’ which selects the normal speed mode. Device power consumption can be optimized at the cost of slower comparator propagation delay by clearing the CxSP bit to ‘0’.

---

Note 1: The CxOE bit of the CMxCON0 register overrides the PORT data latch. Setting the CxON bit of the CMxCON0 register has no impact on the port override.

2: The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.
17.3 Comparator Hysteresis

A selectable amount of separation voltage can be added to the input pins of each comparator to provide a hysteresis function to the overall operation. Hysteresis is enabled by setting the CxHYS bit of the CMxCON0 register.

These hysteresis levels change as a function of the comparator’s Speed/Power mode selection.

Table 17-2 shows the hysteresis levels.

<table>
<thead>
<tr>
<th>CxSP</th>
<th>CxHYS Enabled</th>
<th>CxHYS Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>± 3mV</td>
<td>&lt;&lt; ± 1mV</td>
</tr>
<tr>
<td>1</td>
<td>± 20mV</td>
<td>± 3mV</td>
</tr>
</tbody>
</table>

These levels are approximate.

See Section 29.0 “Electrical Specifications” for more information.

17.4 Timer1 Gate Operation

The output resulting from a comparator operation can be used as a source for gate control of Timer1. See Section 20.6 “Timer1 Gate” for more information. This feature is useful for timing the duration or interval of an analog event.

It is recommended that the comparator output be synchronized to Timer1. This ensures that Timer1 does not increment while a change in the comparator is occurring.

17.4.1 COMPARATOR OUTPUT SYNCHRONIZATION

The output from either comparator, C1 or C2, can be synchronized with Timer1 by setting the CxSYNC bit of the CMxCON0 register.

Once enabled, the comparator output is latched on the falling edge of the Timer1 source clock. If a prescaler is used with Timer1, the comparator output is latched after the prescaling function. To prevent a race condition, the comparator output is latched on the falling edge of the Timer1 clock source and Timer1 increments on the rising edge of its clock source. See the Comparator Block Diagram (Figure 17-2) and the Timer1 Block Diagram (Figure 20-1) for more information.

---

17.5 Comparator Interrupt

An interrupt can be generated upon a change in the output value of the comparator for each comparator, a rising edge detector and a Falling edge detector are present.

When either edge detector is triggered and its associated enable bit is set (CxINTP and/or CxINTN bits of the CMxCON1 register), the Corresponding Interrupt Flag bit (CxF bit of the PIR2 register) will be set.

To enable the interrupt, you must set the following bits:

- CxON, CxPOL and CxSP bits of the CMxCON0 register
- CxIE bit of the PIE2 register
- CxINTP bit of the CMxCON1 register (for a rising edge detection)
- CxINTN bit of the CMxCON1 register (for a falling edge detection)
- PEIE and GIE bits of the INTCON register

The associated interrupt flag bit, CxF bit of the PIR2 register, must be cleared in software. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

Note: Although a comparator is disabled, an interrupt can be generated by changing the output polarity with the CxPOL bit of the CMxCON0 register, or by switching the comparator on or off with the CxON bit of the CMxCON0 register.

17.6 Comparator Positive Input Selection

Configuring the CxPCH<1:0> bits of the CMxCON1 register directs an internal voltage reference or an analog pin to the non-inverting input of the comparator:

- C1IN+ or C2IN+ analog pin
- DAC
- FVR (Fixed Voltage Reference)
- Vss (Ground)

Note: For C1 on the PIC16F1826/7 devices, this selection changes to the C12IN+ pin.

See Section 14.0 “Fixed Voltage Reference (FVR)” for more information on the Fixed Voltage Reference module.

See Section 16.0 “Digital-to-Analog Converter (DAC) Module” for more information on the DAC input signal.

Any time the comparator is disabled (CxON = 0), all comparator inputs are disabled.
17.7 Comparator Negative Input Selection

The CxNCH<1:0> bits of the CMxCON0 register direct one of four analog pins to the comparator inverting input.

Note: To use CxIN+ and CxINx- pins as analog input, the appropriate bits must be set in the ANSEL register and the corresponding TRIS bits must also be set to disable the output drivers.

17.8 Comparator Response Time

The comparator output is indeterminate for a period of time after the change of an input source or the selection of a new reference voltage. This period is referred to as the response time. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response time to a comparator input change. See the Comparator and Voltage Reference Specifications in Section 29.0 "Electrical Specifications" for more details.

17.9 Interaction with ECCP Logic

The C1 and C2 comparators can be used as general purpose comparators. Their outputs can be brought out to the C1OUT and C2OUT pins. When the ECCP Auto-Shutdown is active it can use one or both comparator signals. If auto-restart is also enabled, the comparators can be configured as a closed loop analog feedback to the ECCP, thereby, creating an analog controlled PWM.

17.10 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 17-4. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur.

A maximum source impedance of 10 kΩ is recommended for the analog sources. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current to minimize inaccuracies introduced.

Note 1: When reading a PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert as an analog input, according to the input specification.

2: Analog levels on any pin defined as a digital input, may cause the input buffer to consume more current than is specified.
REGISTER 17-1: CMxCON0: COMPARATOR X CONTROL REGISTER 0

<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>CxON</td>
<td>CxOUT</td>
<td>CxOE</td>
<td>CxPOL</td>
<td>—</td>
<td>CxSP</td>
<td>CxHYS</td>
<td>CxSYNC</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 CxON:** Comparator Enable bit
- 1 = Comparator is enabled and consumes no active power
- 0 = Comparator is disabled

**bit 6 CxOUT:** Comparator Output bit
- If CxPOL = 1 (inverted polarity):
  - 1 = CxVP < CxVN
  - 0 = CxVP > CxVN
- If CxPOL = 0 (non-inverted polarity):
  - 1 = CxVP > CxVN
  - 0 = CxVP < CxVN

**bit 5 CxOE:** Comparator Output Enable bit
- 1 = CxOUT is present on the CxOUT pin. Requires that the associated TRIS bit be cleared to actually drive the pin. Not affected by CxON.
- 0 = CxOUT is internal only

**bit 4 CxPOL:** Comparator Output Polarity Select bit
- 1 = Comparator output is inverted
- 0 = Comparator output is not inverted

**bit 3 Unimplemented:** Read as ‘0’

**bit 2 CxSP:** Comparator Speed/Power Select bit
- 1 = Comparator operates in normal power, higher speed mode
- 0 = Comparator operates in low-power, low-speed mode

**bit 1 CxHYS:** Comparator Hysteresis Enable bit
- 1 = Comparator hysteresis enabled
- 0 = Comparator hysteresis disabled

**bit 0 CxSYNC:** Comparator Output Synchronous Mode bit
- 1 = Comparator output to Timer1 and I/O pin is synchronous to changes on the Timer1 clock source.
  - Output updated on the falling edge of Timer1 clock source.
- 0 = Comparator output to Timer1 and I/O pin is asynchronous
## REGISTER 17-2: CMxCON1: COMPARATOR CX CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR and BOR/Value at all other Resets</th>
<th>Readable bit</th>
<th>Writable bit</th>
<th>Unimplemented bit, read as ‘0’</th>
<th>Bit is unchanged</th>
<th>Bit is unknown</th>
<th>Set</th>
<th>Cleared</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>CxINTP: Comparator Interrupt on Positive Going Edge Enable bits</td>
<td>1 = The CxIF interrupt flag will be set upon a positive going edge of the CxOUT bit</td>
<td>1 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>CxINTN: Comparator Interrupt on Negative Going Edge Enable bits</td>
<td>1 = The CxIF interrupt flag will be set upon a negative going edge of the CxOUT bit</td>
<td>1 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-4</td>
<td>CxPCH&lt;1:0&gt;: Comparator Positive Input Channel Select bits</td>
<td>00 = CxVP connects to CxIN+ pin</td>
<td>00 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For C1:</td>
<td>11 = CxVP connects to C12IN+ pin</td>
<td>11 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For C2:</td>
<td>11 = CxVP connects to VSS</td>
<td>11 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-2</td>
<td>Unimplemented: Read as ‘0’</td>
<td>00 = CxVN connects to C12IN0- pin</td>
<td>00 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-0</td>
<td>CxNCH&lt;1:0&gt;: Comparator Negative Input Channel Select bits</td>
<td>00 = CxVN connects to C12IN1- pin</td>
<td>00 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For C1:</td>
<td>10 = CxVN connects to C12IN2- pin</td>
<td>10 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>For C2:</td>
<td>11 = CxVN connects to C12IN3- pin</td>
<td>11 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## REGISTER 17-3: CMOUT: COMPARATOR OUTPUT REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR and BOR/Value at all other Resets</th>
<th>Readable bit</th>
<th>Writable bit</th>
<th>Unimplemented bit, read as ‘0’</th>
<th>Bit is unchanged</th>
<th>Bit is unknown</th>
<th>Set</th>
<th>Cleared</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-2</td>
<td>Unimplemented: Read as ‘0’</td>
<td>00 = CxVN connects to C12IN3- pin</td>
<td>00 =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>MC2OUT: Mirror Copy of C2OUT bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>MC1OUT: Mirror Copy of C1OUT bit</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
- ’1’ = Bit is set
- ‘0’ = Bit is cleared
- -n/n = Value at POR and BOR/Value at all other Resets
### TABLE 17-3: SUMMARY OF REGISTERS ASSOCIATED WITH COMPARATOR MODULE

<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>—</td>
<td>ANSA4</td>
<td>ANSA3</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
<td>122</td>
</tr>
<tr>
<td>CMxCON0</td>
<td>CxON</td>
<td>CxOUT</td>
<td>CxOE</td>
<td>—</td>
<td>CxPOL</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>163</td>
</tr>
<tr>
<td>CMxCON1</td>
<td>CxNTP</td>
<td>CxINTN</td>
<td>CxPCH1</td>
<td>CxPCH0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>164</td>
</tr>
<tr>
<td>CMOUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>164</td>
</tr>
<tr>
<td>DACCON0</td>
<td>DACEN</td>
<td>DACLPS</td>
<td>DACOE</td>
<td>—</td>
<td>DACPS1</td>
<td>DACPS0</td>
<td>—</td>
<td>—</td>
<td>154</td>
</tr>
<tr>
<td>DACCON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>DACR4</td>
<td>DACR3</td>
<td>DACR2</td>
<td>DACR1</td>
<td>DACR0 154</td>
</tr>
<tr>
<td>FVRCON</td>
<td>FVREN</td>
<td>FVRDY</td>
<td>Reserved</td>
<td>Reserved</td>
<td>CDAFVR1</td>
<td>CDAFVR0</td>
<td>ADFVR1</td>
<td>ADFVR0</td>
<td>136</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>IO CIF</td>
<td>89</td>
</tr>
<tr>
<td>LATA</td>
<td>LATA7</td>
<td>LATA6</td>
<td>—</td>
<td>LATA4</td>
<td>LATA3</td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
<td>120</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSFIE</td>
<td>C2IE</td>
<td>C1IE</td>
<td>EEIE</td>
<td>BCL1IE</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>91</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSFIF</td>
<td>C2IF</td>
<td>C1F</td>
<td>EEIF</td>
<td>BCL1IF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>95</td>
</tr>
<tr>
<td>PORTA</td>
<td>RA7</td>
<td>RA6</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
<td>119</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>120</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

**Note 1:** PIC16F/LF1827 only.
18.0 SR LATCH

The module consists of a single SR Latch with multiple Set and Reset inputs as well as separate latch outputs. The SR Latch module includes the following features:

- Programmable input selection
- SR Latch output is available externally
- Separate Q and Q\bar{\text{Q}} outputs
- Firmware Set and Reset

The SR Latch can be used in a variety of analog applications, including oscillator circuits, one-shot circuit, hysteretic controllers, and analog timing applications.

18.1 Latch Operation

The latch is a Set-Reset Latch that does not depend on a clock source. Each of the Set and Reset inputs are active-high. The latch can be Set or Reset by:

- Software control (SRPS and SRPR bits)
- Comparator C1 output (SYNCC1OUT)
- Comparator C2 output (SYNCC2OUT)
- SRI pin
- Programmable clock (SRCLK)

The SRPS and the SRPR bits of the SRCON0 register may be used to Set or Reset the SR Latch, respectively. The latch is Reset-dominant. Therefore, if both Set and Reset inputs are high, the latch will go to the Reset state. Both the SRPS and SRPR bits are self resetting which means that a single write to either of the bits is all that is necessary to complete a latch Set or Reset operation.

The output from Comparator C1 or C2 can be used as the Set or Reset inputs of the SR Latch. The output of either Comparator can be synchronized to the Timer1 clock source. See Section 17.0 “Comparator Module” and Section 20.0 “Timer1 Module with Gate Control” for more information.

An external source on the SRI pin can be used as the Set or Reset inputs of the SR Latch.

An internal clock source is available that can periodically set or reset the SR Latch. The SRCLK<2:0> bits in the SRCON0 register are used to select the clock source period. The SRSCKE and SRRCKE bits of the SRCON1 register enable the clock source to Set or Reset the SR Latch, respectively.

Note: Enabling both the Set and Reset inputs from any one source at the same time may result in indeterminate operation, as the Reset dominance cannot be assured.

18.2 Latch Output

The SRQEN and SRNQEN bits of the SRCON0 register control the Q and Q\bar{\text{Q}} latch outputs. Both of the SR Latch outputs may be directly output to an I/O pin at the same time.

The applicable TRIS bit of the corresponding port must be cleared to enable the port pin output driver.

18.3 Effects of a Reset

Upon any device Reset, the SR Latch output is not initialized to a known state. The user’s firmware is responsible for initializing the latch output before enabling the output pins.
FIGURE 18-1: SR LATCH SIMPLIFIED BLOCK DIAGRAM

Note 1: If R = 1 and S = 1 simultaneously, Q = 0, \( \bar{Q} = 1 \)
2: Pulse generator causes a 1 Q-state pulse width.
3: Name denotes the connection point at the comparator output.
TABLE 18-1: SRCLK FREQUENCY TABLE

<table>
<thead>
<tr>
<th>SRCLK</th>
<th>Divider</th>
<th>FOSC = 32 MHz</th>
<th>FOSC = 20 MHz</th>
<th>FOSC = 16 MHz</th>
<th>FOSC = 4 MHz</th>
<th>FOSC = 1 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>512</td>
<td>62.5 kHz</td>
<td>39.0 kHz</td>
<td>31.3 kHz</td>
<td>7.81 kHz</td>
<td>1.95 kHz</td>
</tr>
<tr>
<td>110</td>
<td>256</td>
<td>125 kHz</td>
<td>78.1 kHz</td>
<td>62.5 kHz</td>
<td>15.6 kHz</td>
<td>3.90 kHz</td>
</tr>
<tr>
<td>101</td>
<td>128</td>
<td>250 kHz</td>
<td>156 kHz</td>
<td>125 kHz</td>
<td>31.25 kHz</td>
<td>7.81 kHz</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td>500 kHz</td>
<td>313 kHz</td>
<td>250 kHz</td>
<td>62.5 kHz</td>
<td>15.6 kHz</td>
</tr>
<tr>
<td>011</td>
<td>32</td>
<td>1 MHz</td>
<td>625 kHz</td>
<td>500 kHz</td>
<td>125 kHz</td>
<td>31.3 kHz</td>
</tr>
<tr>
<td>010</td>
<td>16</td>
<td>2 MHz</td>
<td>1.25 MHz</td>
<td>1 MHz</td>
<td>250 kHz</td>
<td>62.5 kHz</td>
</tr>
<tr>
<td>001</td>
<td>8</td>
<td>4 MHz</td>
<td>2.5 MHz</td>
<td>2 MHz</td>
<td>500 kHz</td>
<td>125 kHz</td>
</tr>
<tr>
<td>000</td>
<td>4</td>
<td>8 MHz</td>
<td>5 MHz</td>
<td>4 MHz</td>
<td>1 MHz</td>
<td>250 kHz</td>
</tr>
</tbody>
</table>

REGISTER 18-1: SRCON0: SR LATCH CONTROL 0 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/S-0/0</th>
<th>R/S-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRLEN</td>
<td>SRCLK2</td>
<td>SRCLK1</td>
<td>SRCLK0</td>
<td>SRQEN</td>
<td>SRNQEN</td>
<td>SRPS</td>
<td>SRPR</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  S = Bit is set only

bit 7  **SRLEN**: SR Latch Enable bit
1 = SR Latch is enabled
0 = SR Latch is disabled

bit 6-4  **SRCLK<2:0>**: SR Latch Clock Divider bits
000 = Generates a 1 Fosc wide pulse every 4th Fosc cycle clock
001 = Generates a 1 Fosc wide pulse every 8th Fosc cycle clock
010 = Generates a 1 Fosc wide pulse every 16th Fosc cycle clock
011 = Generates a 1 Fosc wide pulse every 32nd Fosc cycle clock
100 = Generates a 1 Fosc wide pulse every 64th Fosc cycle clock
101 = Generates a 1 Fosc wide pulse every 128th Fosc cycle clock
110 = Generates a 1 Fosc wide pulse every 256th Fosc cycle clock
111 = Generates a 1 Fosc wide pulse every 512th Fosc cycle clock

bit 3  **SRQEN**: SR Latch Q Output Enable bit
If SRLEN = 1:
1 = Q is present on the SRQ pin
0 = External Q output is disabled
If SRLEN = 0:
SR Latch is disabled

bit 2  **SRNQEN**: SR Latch Q̅ Output Enable bit
If SRLEN = 1:
1 = Q̅ is present on the SRnQ pin
0 = External Q̅ output is disabled
If SRLEN = 0:
SR Latch is disabled

bit 1  **SRPS**: Pulse Set Input of the SR Latch bit(1)
1 = Pulse set input for 1 Q-clock period
0 = No effect on set input.

bit 0  **SRPR**: Pulse Reset Input of the SR Latch bit(1)
1 = Pulse reset input for 1 Q-clock period
0 = No effect on reset input.

**Note 1**: Set only, always reads back ‘0’.
REGISTER 18-2: SRCON1: SR LATCH CONTROL 1 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>SRSPE</td>
<td>SRSCKE</td>
<td>SRSC2E</td>
<td>SRSC1E</td>
<td>SRRPE</td>
<td>SRRCKE</td>
<td>SRRC2E</td>
<td>SRRC1E</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  SRSPE: SR Latch Peripheral Set Enable bit
  1 = SR Latch is set when the SRI pin is high
  0 = SRI pin has no effect on the set input of the SR Latch

bit 6  SRSCKE: SR Latch Set Clock Enable bit
  1 = Set input of SR Latch is pulsed with SRCLK
  0 = SRCLK has no effect on the set input of the SR Latch

bit 5  SRSC2E: SR Latch C2 Set Enable bit
  1 = SR Latch is set when the C2 Comparator output is high
  0 = C2 Comparator output has no effect on the set input of the SR Latch

bit 4  SRSC1E: SR Latch C1 Set Enable bit
  1 = SR Latch is set when the C1 Comparator output is high
  0 = C1 Comparator output has no effect on the set input of the SR Latch

bit 3  SRRPE: SR Latch Peripheral Reset Enable bit
  1 = SR Latch is reset when the SRI pin is high
  0 = SRI pin has no effect on the reset input of the SR Latch

bit 2  SRRCKE: SR Latch Reset Clock Enable bit
  1 = Reset input of SR Latch is pulsed with SRCLK
  0 = SRCLK has no effect on the reset input of the SR Latch

bit 1  SRRC2E: SR Latch C2 Reset Enable bit
  1 = SR Latch is reset when the C2 Comparator output is high
  0 = C2 Comparator output has no effect on the reset input of the SR Latch

bit 0  SRRC1E: SR Latch C1 Reset Enable bit
  1 = SR Latch is reset when the C1 Comparator output is high
  0 = C1 Comparator output has no effect on the reset input of the SR Latch
### TABLE 18-2: SUMMARY OF REGISTERS ASSOCIATED WITH SR LATCH MODULE

<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>—</td>
<td>—</td>
<td>ANSA4</td>
<td>ANSA3</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0 122</td>
</tr>
<tr>
<td>SRCON0</td>
<td>SRLN</td>
<td>SRCLK2</td>
<td>SRCLK1</td>
<td>SRCLK0</td>
<td>SRQEN</td>
<td>SRQEN</td>
<td>SRPS</td>
<td>SRPR</td>
<td>169</td>
</tr>
<tr>
<td>SRCON1</td>
<td>SRSPE</td>
<td>SRSCKE</td>
<td>SRSC2E</td>
<td>SRSC1E</td>
<td>SRRPE</td>
<td>SRRPE</td>
<td>SRRCKE</td>
<td>SRRC2E</td>
<td>SRRC1E 170</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>120</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are unused by the SR Latch module.
19.0 TIMER0 MODULE

The Timer0 module is an 8-bit timer/counter with the following features:
- 8-bit timer/counter register (TMR0)
- 8-bit prescaler (independent of Watchdog Timer)
- Programmable internal or external clock source
- Programmable external clock edge selection
- Interrupt on overflow
- TMR0 can be used to gate Timer1

Figure 19-1 is a block diagram of the Timer0 module.

19.1 Timer0 Operation

The Timer0 module can be used as either an 8-bit timer or an 8-bit counter.

19.1.1 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 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.

19.1.2 8-BIT COUNTER MODE

In 8-Bit Counter mode, the Timer0 module will increment on every rising or falling edge of the T0CKI pin or the Capacitive Sensing Oscillator (CPSCLK) signal.

8-Bit Counter mode using the T0CKI pin is selected by setting the TMR0CS bit in the OPTION register to ‘1’ and resetting the T0XCS bit in the CPSCON0 register to ‘0’.

8-Bit Counter mode using the Capacitive Sensing Oscillator (CPSCLK) signal is selected by setting the TMR0CS bit in the OPTION register to ‘1’ and setting the T0XCS bit in the CPSCON0 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 register.
19.1.3 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 register.

| Note: | The Watchdog Timer (WDT) uses its own independent prescaler. |

There are 8 prescaler options for the Timer0 module ranging from 1:2 to 1:256. The prescale values are selectable via the PS<2:0> bits of the OPTION 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 register.

The prescaler is not readable or writable. All instructions writing to the TMR0 register will clear the prescaler.

19.1.4 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. |

19.1.5 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 29.0 “Electrical Specifications”.

19.1.6 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.
TABLE 19-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER0

<table>
<thead>
<tr>
<th>Register</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>CPSCON0</td>
<td>CPSON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPSRN</td>
<td>CPSRN0</td>
<td>CPSOUT</td>
<td>T0xCS</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>IO CIF</td>
<td>89</td>
</tr>
<tr>
<td>OPTION_REG</td>
<td>WPUEN</td>
<td>INTEDG</td>
<td>TMR0CS</td>
<td>TMR0SE</td>
<td>PSA</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
<td>175*</td>
</tr>
<tr>
<td>TMR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>173*</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>120</td>
</tr>
</tbody>
</table>

Legend: — = Unimplemented locations, read as ‘0’. Shaded cells are not used by the Timer0 module. * Page provides register information.
20.0 TIMER1 MODULE WITH GATE CONTROL

The Timer1 module is a 16-bit timer/counter with the following features:

• 16-bit timer/counter register pair (TMR1H:TMR1L)
• Programmable internal or external clock source
• 3-bit prescaler
• Dedicated 32 kHz oscillator circuit
• Optionally synchronized comparator out
• Multiple Timer1 gate (count enable) sources
• Interrupt on overflow
• Wake-up on overflow (external clock, Asynchronous mode only)
• Time base for the Capture/Compare function
• Special Event Trigger (with CCP/ECCP)
• Selectable Gate Source Polarity

• Gate Toggle Mode
• Gate Single-pulse Mode
• Gate Value Status
• Gate Event Interrupt

Figure 20-1 is a block diagram of the Timer1 module.

Figure 20-1: TIMER1 BLOCK DIAGRAM

Note 1: ST Buffer is high speed type when using T1CKI.
2: Timer1 register increments on rising edge.
3: Synchronize does not operate while in Sleep.
20.1 Timer1 Operation

The Timer1 module is a 16-bit incrementing counter which is accessed through the TMR1H:TMR1L register pair. Writes to TMR1H or TMR1L directly update the counter.

When used with an internal clock source, the module is a timer and increments on every instruction cycle. When used with an external clock source, the module can be used as either a timer or counter and increments on every selected edge of the external source.

Timer1 is enabled by configuring the TMR1ON and TMR1GE bits in the T1CON and T1GCON registers, respectively. Table 20-1 displays the Timer1 enable selections.

### TABLE 20-1: TIMER1 ENABLE SELECTIONS

<table>
<thead>
<tr>
<th>TMR1ON</th>
<th>TMR1GE</th>
<th>Timer1 Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Off</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Always On</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Count Enabled</td>
</tr>
</tbody>
</table>

Note: In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:
- Timer1 enabled after POR
- Write to TMR1H or TMR1L
- Timer1 is disabled
- Timer1 is disabled (TMR1ON = 0) when T1CKI is high then Timer1 is enabled (TMR1ON=1) when T1CKI is low.

20.2 Clock Source Selection

The TMR1CS<1:0> and T1OSCEN bits of the T1CON register are used to select the clock source for Timer1. Table 20-2 displays the clock source selections.

### 20.2.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected the TMR1H:TMR1L register pair will increment on multiples of Fosc as determined by the Timer1 prescaler.

### 20.2.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1 module may work as a timer or a counter.

When enabled to count, Timer1 is incremented on the rising edge of the external clock input T1CKI or the capacitive sensing oscillator signal. Either of these external clock sources can be synchronized to the microcontroller system clock or they can run asynchronously.

When used as a timer with a clock oscillator, an external 32.768 kHz crystal can be used in conjunction with the dedicated internal oscillator circuit.

### TABLE 20-2: CLOCK SOURCE SELECTIONS

<table>
<thead>
<tr>
<th>TMR1CS1</th>
<th>TMR1CS0</th>
<th>T1OSCEN</th>
<th>Clock Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>System Clock (Fosc)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>x</td>
<td>Instruction Clock (Fosc/4)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>x</td>
<td>Capacitive Sensing Oscillator</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>External Clocking on T1CKI Pin</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Osc.Circuit On T1OSI/T1OSO Pins</td>
</tr>
</tbody>
</table>
20.3 Timer1 Prescaler

Timer1 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The T1CKPS bits of the T1CON register control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMR1H or TMR1L.

20.4 Timer1 Oscillator

A dedicated low-power 32.768 kHz oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). This internal circuit is to be used in conjunction with an external 32.768 kHz crystal.

The oscillator circuit is enabled by setting the T1OSCEN bit of the T1CON register. The oscillator will continue to run during Sleep.

Note: The oscillator requires a start-up and stabilization time before use. Thus, T1OSCEN should be set and a suitable delay observed prior to enabling Timer1.

20.5 Timer1 Operation in Asynchronous Counter Mode

If control bit T1SYNC of the T1CON register is set, the external clock input is not synchronized. The timer increments asynchronously to the internal phase clocks. If external clock source is selected then the timer will continue to run during Sleep and can generate an interrupt on overflow, which will wake-up the processor. However, special precautions in software are needed to read/write the timer (see Section 20.5.1 “Reading and Writing Timer1 in Asynchronous Counter Mode”).

Note: When switching from synchronous to asynchronous operation, it is possible to skip an increment. When switching from asynchronous to synchronous operation, it is possible to produce an additional increment.

20.5.1 READING AND WRITING TIMER1 IN ASYNCHRONOUS COUNTER MODE

Reading TMR1H or TMR1L while the timer is running from an external asynchronous clock will ensure a valid read (taken care of in hardware). However, the user should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since the timer may overflow between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers, while the register is incrementing. This may produce an unpredictable value in the TMR1H:TMR1L register pair.

20.6 Timer1 Gate

Timer1 can be configured to count freely or the count can be enabled and disabled using Timer1 Gate circuitry. This is also referred to as Timer1 Gate Enable. Timer1 Gate can also be driven by multiple selectable sources.

20.6.1 TIMER1 GATE ENABLE

The Timer1 Gate Enable mode is enabled by setting the TMR1GE bit of the T1CON register. The polarity of the Timer1 Gate Enable mode is configured using the T1GPOL bit of the T1GCON register.

When Timer1 Gate Enable mode is enabled, Timer1 will increment on the rising edge of the Timer1 clock source. When Timer1 Gate Enable mode is disabled, no incrementing will occur and Timer1 will hold the current count. See Figure 20-3 for timing details.

TABLE 20-3: TIMER1 GATE ENABLE SELECTIONS

<table>
<thead>
<tr>
<th>T1CLK</th>
<th>T1GPOL</th>
<th>T1G</th>
<th>Timer1 Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>Counts</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>Holds Count</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>Holds Count</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>Counts</td>
</tr>
</tbody>
</table>

20.6.2 TIMER1 GATE SOURCE SELECTION

The Timer1 Gate source can be selected from one of four different sources. Source selection is controlled by the T1GSS bits of the T1GCON register. The polarity for each available source is also selectable. Polarity selection is controlled by the T1GPOL bit of the T1GCON register.

TABLE 20-4: TIMER1 GATE SOURCES

<table>
<thead>
<tr>
<th>T1GSS</th>
<th>Timer1 Gate Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Timer1 Gate Pin</td>
</tr>
<tr>
<td>01</td>
<td>Overflow of Timer0 (TMR0 increments from FFh to 00h)</td>
</tr>
<tr>
<td>10</td>
<td>Comparator 1 Output SYNCC1OUT (optionally synchronized out)</td>
</tr>
<tr>
<td>11</td>
<td>Comparator 2 Output SYNCC2OUT (optionally synchronized out)</td>
</tr>
</tbody>
</table>
20.6.2.1 T1G Pin Gate Operation

The T1G pin is one source for Timer1 Gate Control. It can be used to supply an external source to the Timer1 Gate circuitry.

20.6.2.2 Timer0 Overflow Gate Operation

When Timer0 increments from FFh to 00h, a low-to-high pulse will automatically be generated and internally supplied to the Timer1 Gate circuitry.

20.6.2.3 Comparator C1 Gate Operation

The output resulting from a Comparator 1 operation can be selected as a source for Timer1 Gate Control. The Comparator 1 output (SYNCC1OUT) can be synchronized to the Timer1 clock or left asynchronous. For more information see Section 17.4.1 “Comparator Output Synchronization”.

20.6.2.4 Comparator C2 Gate Operation

The output resulting from a Comparator 2 operation can be selected as a source for Timer1 Gate Control. The Comparator 2 output (SYNCC2OUT) can be synchronized to the Timer1 clock or left asynchronous. For more information see Section 17.4.1 “Comparator Output Synchronization”.

20.6.3 TIMER1 GATE TOGGLE MODE

When Timer1 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1 gate signal, as opposed to the duration of a single level pulse.

The Timer1 Gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 20-4 for timing details.

Timer1 Gate Toggle mode is enabled by setting the T1GTM bit of the T1GCON register. When the T1GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

Note: Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

20.6.4 TIMER1 GATE SINGLE-PULSE MODE

When Timer1 Gate Single-Pulse mode is enabled, it is possible to capture a single pulse gate event. Timer1 Gate Single-Pulse mode is first enabled by setting the T1GSPM bit in the T1GCON register. Next, the T1GGO/DONE bit in the T1GCON register must be set. The Timer1 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the T1GGO/DONE bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the T1GGO/DONE bit is once again set in software.

Clearing the T1GSPM bit of the T1GCON register will also clear the T1GGO/DONE bit. See Figure 20-5 for timing details.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1 Gate source to be measured. See Figure 20-6 for timing details.

20.6.5 TIMER1 GATE VALUE STATUS

When Timer1 Gate Value Status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the T1GVAL bit in the T1GCON register. The T1GCON bit is valid even when the Timer1 Gate is not enabled (TMR1GE bit is cleared).

20.6.6 TIMER1 GATE EVENT INTERRUPT

When Timer1 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of T1GVAL occurs, the TMR1GIF flag bit in the PIR1 register will be set. If the TMR1GIE bit in the PIE1 register is set, then an interrupt will be recognized.

The TMR1GIF flag bit operates even when the Timer1 Gate is not enabled (TMR1GE bit is cleared).
20.7 Timer1 Interrupt

The Timer1 register pair (TMR1H:TMR1L) increments to FFFFh and rolls over to 0000h. When Timer1 rolls over, the Timer1 interrupt flag bit of the PIR1 register is set. To enable the interrupt on rollover, you must set these bits:

- TMR1ON bit of the T1CON register
- TMR1IE bit of the PIE1 register
- PEIE bit of the INTCON register
- GIE bit of the INTCON register

The interrupt is cleared by clearing the TMR1IF bit in the Interrupt Service Routine.

Note: The TMR1H:TMR1L register pair and the TMR1IF bit should be cleared before enabling interrupts.

20.8 Timer1 Operation During Sleep

Timer1 can only operate during Sleep when setup in Asynchronous Counter mode. In this mode, an external crystal or clock source can be used to increment the counter. To set up the timer to wake the device:

- TMR1ON bit of the T1CON register must be set
- TMR1IE bit of the PIE1 register must be set
- PEIE bit of the INTCON register must be set
- T1SYNC bit of the T1CON register must be set
- TMR1CS bits of the T1CON register must be configured
- T1OSCEN bit of the T1CON register must be configured

The device will wake-up on an overflow and execute the next instructions. If the GIE bit of the INTCON register is set, the device will call the Interrupt Service Routine (0004h).

Timer1 oscillator will continue to operate in Sleep regardless of the T1SYNC bit setting.

20.9 ECCP/CCP Capture/Compare Time Base

The CCP modules use the TMR1H:TMR1L register pair as the time base when operating in Capture or Compare mode.

In Capture mode, the value in the TMR1H:TMR1L register pair is copied into the CCP1H:CCPR1L register pair on a configured event.

In Compare mode, an event is triggered when the value CCP1H:CCPR1L register pair matches the value in the TMR1H:TMR1L register pair. This event can be a Special Event Trigger.

For more information, see Section 23.0 “Capture/Compare/PWM Modules (ECCP1, ECCP2, CCP3, CCP4)”.

20.10 ECCP/CCP Special Event Trigger

When any of the CCP’s are configured to trigger a special event, the trigger will clear the TMR1H:TMR1L register pair. This special event does not cause a Timer1 interrupt. The CCP module may still be configured to generate a CCP interrupt.

In this mode of operation, the CCP1H:CCPR1L register pair becomes the period register for Timer1. Timer1 should be synchronized and Fosc/4 should be selected as the clock source in order to utilize the Special Event Trigger. Asynchronous operation of Timer1 can cause a Special Event Trigger to be missed.

In the event that a write to TMR1H or TMR1L coincides with a Special Event Trigger from the CCP, the write will take precedence.

For more information, see Section 15.2.5 “Special Event Trigger”.

FIGURE 20-2: TIMER1 INCREMENTING EDGE

Note 1: Arrows indicate counter increments.

2: In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge of the clock.
FIGURE 20-3: TIMER1 GATE ENABLE MODE

FIGURE 20-4: TIMER1 GATE TOGGLE MODE
**FIGURE 20-5: TIMER1 GATE SINGLE-PULSE MODE**

- **TMR1GE**: Set by software.
- **T1GPOL**: Set by hardware on falling edge of T1GVAL.
- **T1GSPM**: Cleared by software.
- **T1GO/DONE**: Cleared by hardware on falling edge of T1GVAL.
- **T1G_IN**: Counting enabled on rising edge of T1G.
- **T1CKI**:
- **T1GVAL**:
- **TIMR1**:
- **TMR1GIF**: Set by hardware on falling edge of T1GVAL.
- **TMR1GIF**: Cleared by software.
FIGURE 20-6: TIMER1 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE

- TMR1GE
- T1GPOL
- T1GSPM
- T1GTM
- T1GVAL
- T1GO/
- T1G_IN
- T1CKI
- TIMER1
- TMR1GIF

Set by software
Cleared by hardware on falling edge of T1G
Counting enabled on rising edge of T1G
Cleared by software
Set by hardware on falling edge of T1GVAL
Cleared by software
20.11 Timer1 Control Register

The Timer1 Control register (T1CON), shown in Register 20-1, is used to control Timer1 and select the various features of the Timer1 module.

REGISTER 20-1: T1CON: TIMER1 CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TMR1CS&lt;1:0&gt;: Timer1 Clock Source Select bits</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>T1CKPS&lt;1:0&gt;: Timer1 Input Clock Prescale Select bits</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>T1OSCEN: LP Oscillator Enable Control bit</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>T1SYNC: Timer1 External Clock Input Synchronization Control bit</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>TMR1ON: Timer1 On bit</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

**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 7-6**

**TMR1CS<1:0>: Timer1 Clock Source Select bits**

- 11 = Timer1 clock source is Capacitive Sensing Oscillator (CAPOSC)
- 10 = Timer1 clock source is pin or oscillator:
  - If T1OSCEN = 0:
    - External clock from T1CKI pin (on the rising edge)
  - If T1OSCEN = 1:
    - Crystal oscillator on T1OSI/T1OSO pins
- 01 = Timer1 clock source is system clock (Fosc)
- 00 = Timer1 clock source is instruction clock (Fosc/4)

**bit 5-4**

**T1CKPS<1:0>: Timer1 Input Clock Prescale Select bits**

- 11 = 1:8 Prescale value
- 10 = 1:4 Prescale value
- 01 = 1:2 Prescale value
- 00 = 1:1 Prescale value

**bit 3**

**T1OSCEN: LP Oscillator Enable Control bit**

- 1 = Dedicated Timer1 oscillator circuit enabled
- 0 = Dedicated Timer1 oscillator circuit disabled

**bit 2**

**T1SYNC: Timer1 External Clock Input Synchronization Control bit**

- **TMR1CS<1:0> = 1x**
  - This bit is ignored. Timer1 uses the internal clock when TMR1CS<1:0> = 1x.

**bit 1**

**Unimplemented:** Read as ‘0’

**bit 0**

**TMR1ON: Timer1 On bit**

- 1 = Enables Timer1
- 0 = Stops Timer1
  - Clears Timer1 Gate flip-flop
20.12 Timer1 Gate Control Register

The Timer1 Gate Control register (T1GCON), shown in Register 20-2, is used to control Timer1 Gate.

**REGISTER 20-2: T1GCON: TIMER1 GATE CONTROL REGISTER**

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR1GE</td>
<td>T1GSS0</td>
</tr>
<tr>
<td>T1GPPOL</td>
<td>T1GSS1</td>
</tr>
<tr>
<td>T1GTM</td>
<td>T1GVAL</td>
</tr>
<tr>
<td>T1GSPM</td>
<td>T1GGO/DONE</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
- **HC** = Bit is cleared by hardware
- ‘-n/n’ = Value at POR and BOR/Value at all other Resets

**bit 7**

**TMR1GE:** Timer1 Gate Enable bit

If TMR1ON = 0:
This bit is ignored

If TMR1ON = 1:
1 = Timer1 counting is controlled by the Timer1 gate function
0 = Timer1 counts regardless of Timer1 gate function

**bit 6**

**T1GPOL:** Timer1 Gate Polarity bit

1 = Timer1 gate is active-high (Timer1 counts when gate is high)
0 = Timer1 gate is active-low (Timer1 counts when gate is low)

**bit 5**

**T1GTM:** Timer1 Gate Toggle Mode bit

1 = Timer1 Gate Toggle mode is enabled
0 = Timer1 Gate Toggle mode is disabled and toggle flip-flop is cleared
Timer1 gate flip-flop toggles on every rising edge.

**bit 4**

**T1GSPM:** Timer1 Gate Single-Pulse Mode bit

1 = Timer1 gate Single-Pulse mode is enabled and is controlling Timer1 gate
0 = Timer1 gate Single-Pulse mode is disabled

**bit 3**

**T1GGO/DONE:** Timer1 Gate Single-Pulse Acquisition Status bit

1 = Timer1 gate single-pulse acquisition is ready, waiting for an edge
0 = Timer1 gate single-pulse acquisition has completed or has not been started
This bit is automatically cleared when T1GSPM is cleared.

**bit 2**

**T1GVAL:** Timer1 Gate Current State bit

Indicates the current state of the Timer1 gate that could be provided to TMR1H:TMR1L.
Unaffected by Timer1 Gate Enable (TMR1GE).

**bit 1-0**

**T1GSS<1:0>:** Timer1 Gate Source Select bits

00 = Timer1 Gate pin
01 = Timer0 overflow output
10 = Comparator 1 optionally synchronized output (SYNCC1OUT)
11 = Comparator 2 optionally synchronized output (SYNCC2OUT)
### TABLE 20-5: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1

<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>ANSELB</td>
<td>ANSB7</td>
<td>ANSB6</td>
<td>ANSB5</td>
<td>ANSB4</td>
<td>ANSB3</td>
<td>ANSB2</td>
<td>ANSB1</td>
<td>—</td>
<td>128</td>
</tr>
<tr>
<td>CCP1CON</td>
<td>PxM1</td>
<td>PxM0</td>
<td>DCxB1</td>
<td>DCxB0</td>
<td>CCPxM3</td>
<td>CCPxM2</td>
<td>CCPxM1</td>
<td>CCPxM0</td>
<td>204</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>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>PORTB</td>
<td>RB7</td>
<td>RB6</td>
<td>RB5</td>
<td>RB4</td>
<td>RB3</td>
<td>RB2</td>
<td>RB1</td>
<td>RB0</td>
<td>126</td>
</tr>
<tr>
<td>TMR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>181*</td>
</tr>
<tr>
<td>TMR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>181*</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
<tr>
<td>T1CON</td>
<td>TMR1CS1</td>
<td>TMR1CS0</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>T1SYNC</td>
<td>—</td>
<td>TMR1ON</td>
<td>185</td>
</tr>
<tr>
<td>T1GCON</td>
<td>TMR1GE</td>
<td>T1GPOI</td>
<td>T1GTM</td>
<td>T1GSPM</td>
<td>T1GGO/DONE</td>
<td>T1GVAL</td>
<td>T1GSS1</td>
<td>T1GSS0</td>
<td>186</td>
</tr>
</tbody>
</table>

**Legend:**
- — = unimplemented, read as ‘0’. Shaded cells are not used by the Timer1 module.
- * Page provides register information.
21.0 TIMER2/4/6 MODULES

There are up to three identical Timer2-type modules available. To maintain pre-existing naming conventions, the Timers are called Timer2, Timer4 and Timer6 (also Timer2/4/6).

Note: The ‘x’ variable used in this section is used to designate Timer2, Timer4, or Timer6. For example, TxCON references T2CON, T4CON, or T6CON. PRx references P2, P4, or P6.

The Timer2/4/6 modules incorporate the following features:
- 8-bit Timer and Period registers (TMRx and PRx, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16, and 1:64)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMRx match with PRx, respectively
- Optional use as the shift clock for the MSSPx modules (Timer2 only)

See Figure 21-1 for a block diagram of Timer2/4/6.

FIGURE 21-1: TIMER2/4/6 BLOCK DIAGRAM
21.1 Timer2/4/6 Operation

The clock input to the Timer2/4/6 modules is the system instruction clock (Fosc/4).

TMRx increments from 00h on each clock edge.

A 4-bit counter/prescaler on the clock input allows direct input, divide-by-4 and divide-by-16 prescale options. These options are selected by the prescaler control bits, TxCKPS<1:0> of the TxCON register. The value of TMRx is compared to that of the Period register, PRx, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMRx to 00h on the next cycle and drives the output counter/postscaler (see Section 21.2 “Timer2/4/6 Interrupt”).

The TMRx and PRx registers are both directly readable and writable. The TMRx register is cleared on any device Reset, whereas the PRx register initializes to FFh. Both the prescaler and postscaler counters are cleared on the following events:

• a write to the TMRx register
• a write to the TxCON register
• Power-on Reset (POR)
• Brown-out Reset (BOR)
• MCLR Reset
• Watchdog Timer (WDT) Reset
• Stack Overflow Reset
• Stack Underflow Reset
• RESET Instruction

Note: TMRx is not cleared when TxCON is written.

21.2 Timer2/4/6 Interrupt

Timer2/4/6 can also generate an optional device interrupt. The Timer2/4/6 output signal (TMRx-to-PRx match) provides the input for the 4-bit counter/postscaler. This counter generates the TMRx match interrupt flag which is latched in TMRxIF of the PIEx register. The interrupt is enabled by setting the TMRx Match Interrupt Enable bit, TMRxE of the PIEx register.

A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, TxOUTPS<3:0>, of the TxCON register.

21.3 Timer2/4/6 Output

The unscaled output of TMRx is available primarily to the CCP modules, where it is used as a time base for operations in PWM mode.

Timer2 can be optionally used as the shift clock source for the MSSPx modules operating in SPI mode. Additional information is provided in Section 24.0 “Master Synchronous Serial Port (MSSP1 and MSSP2) Module”.

21.4 Timer2/4/6 Operation During Sleep

The Timerx timers cannot be operated while the processor is in Sleep mode. The contents of the TMRx and PRx registers will remain unchanged while the processor is in Sleep mode.
## REGISTER 21-1: TXCON: TIMER2/TIMER4/TIMER6 CONTROL REGISTERS

<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>TxOUTPS3</td>
<td>TxOUTPS2</td>
<td>TxOUTPS1</td>
<td>TxOUTPS0</td>
<td>TMRxON</td>
<td>TxCKPS1</td>
<td>TxCKPS0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6-3</th>
<th>bit 2</th>
<th>bit 1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unimplemented: Read as '0'</td>
<td>TxOUTPS&lt;3:0&gt;: Timerx Output Postscaler Select bits</td>
<td>TMRxON: Timerx On bit</td>
<td>TxCKPS&lt;1:0&gt;: Timer2-type Clock Prescaler Select bits</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>Unimplemented: Read as '0'</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 6-3</td>
<td>TxOUTPS&lt;3:0&gt;: Timerx Output Postscaler Select bits</td>
</tr>
<tr>
<td>0000</td>
<td>1:1 Postscaler</td>
</tr>
<tr>
<td>0001</td>
<td>1:2 Postscaler</td>
</tr>
<tr>
<td>0010</td>
<td>1:3 Postscaler</td>
</tr>
<tr>
<td>0011</td>
<td>1:4 Postscaler</td>
</tr>
<tr>
<td>0100</td>
<td>1:5 Postscaler</td>
</tr>
<tr>
<td>0101</td>
<td>1:6 Postscaler</td>
</tr>
<tr>
<td>0110</td>
<td>1:7 Postscaler</td>
</tr>
<tr>
<td>0111</td>
<td>1:8 Postscaler</td>
</tr>
<tr>
<td>1000</td>
<td>1:9 Postscaler</td>
</tr>
<tr>
<td>1001</td>
<td>1:10 Postscaler</td>
</tr>
<tr>
<td>1010</td>
<td>1:11 Postscaler</td>
</tr>
<tr>
<td>1011</td>
<td>1:12 Postscaler</td>
</tr>
<tr>
<td>1100</td>
<td>1:13 Postscaler</td>
</tr>
<tr>
<td>1101</td>
<td>1:14 Postscaler</td>
</tr>
<tr>
<td>1110</td>
<td>1:15 Postscaler</td>
</tr>
<tr>
<td>1111</td>
<td>1:16 Postscaler</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 2</th>
<th>TMRxON: Timerx On bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Timerx is on</td>
</tr>
<tr>
<td>0</td>
<td>Timerx is off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1-0</th>
<th>TxCKPS&lt;1:0&gt;: Timer2-type Clock Prescaler Select bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Prescaler is 1</td>
</tr>
<tr>
<td>01</td>
<td>Prescaler is 4</td>
</tr>
<tr>
<td>10</td>
<td>Prescaler is 16</td>
</tr>
<tr>
<td>11</td>
<td>Prescaler is 64</td>
</tr>
</tbody>
</table>
### TABLE 21-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2/4/6

<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>IO CIF</td>
<td>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>PIE3(1)</td>
<td>—</td>
<td>—</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>TMR6IE</td>
<td>—</td>
<td>TMR4IE</td>
<td>—</td>
<td>92</td>
</tr>
<tr>
<td>PIR3(1)</td>
<td>—</td>
<td>—</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>TMR6IF</td>
<td>—</td>
<td>TMR4IF</td>
<td>—</td>
<td>96</td>
</tr>
<tr>
<td>PR2</td>
<td>Timer2 Module Period Register</td>
<td>189*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2CON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T2OUTPS3</td>
<td>—</td>
<td>—</td>
<td>T2OUTPS0</td>
<td>T2OUTPS1</td>
<td>TMR2ON</td>
</tr>
<tr>
<td>PR4</td>
<td>Timer4 Module Period Register</td>
<td>189*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR4</td>
<td>Holding Register for the 8-bit TMR4 Time Base(1)</td>
<td>189*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4CON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T4OUTPS3</td>
<td>—</td>
<td>—</td>
<td>T4OUTPS0</td>
<td>T4OUTPS1</td>
<td>T4OUTPS0</td>
</tr>
<tr>
<td>PR6</td>
<td>Timer6 Module Period Register</td>
<td>189*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR6</td>
<td>Holding Register for the 8-bit TMR6 Time Base(1)</td>
<td>189*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T6CON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>T6OUTPS3</td>
<td>—</td>
<td>—</td>
<td>T6OUTPS0</td>
<td>T6OUTPS1</td>
<td>T6OUTPS0</td>
</tr>
</tbody>
</table>

**Legend:**
- — = unimplemented read as ‘0’. Shaded cells are not used for Timer2 module.
- * Page provides register information.

**Note 1:** PIC16F/LF1827 only.
22.0 DATA SIGNAL MODULATOR

The Data Signal Modulator (DSM) is a peripheral which allows the user to mix a data stream, also known as a modulator signal, with a carrier signal to produce a modulated output.

Both the carrier and the modulator signals are supplied to the DSM module either internally, from the output of a peripheral, or externally through an input pin.

The modulated output signal is generated by performing a logical “AND” operation of both the carrier and modulator signals and then provided to the MDOUT pin.

The carrier signal is comprised of two distinct and separate signals. A carrier high (CARH) signal and a carrier low (CARL) signal. During the time in which the modulator (MOD) signal is in a logic high state, the DSM mixes the carrier high signal with the modulator signal. When the modulator signal is in a logic low state, the DSM mixes the carrier low signal with the modulator signal.

Using this method, the DSM can generate the following types of Key Modulation schemes:
- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)
- On-Off Keying (OOK)

Additionally, the following features are provided within the DSM module:
- Carrier Synchronization
- Carrier Source Polarity Select
- Carrier Source Pin Disable
- Programmable Modulator Data
- Modulator Source Pin Disable
- Modulated Output Polarity Select
- Slew Rate Control

Figure 22-1 shows a Simplified Block Diagram of the Data Signal Modulator peripheral.
22.1 DSM Operation

The DSM module can be enabled by setting the MDEN bit in the MDCON register. Clearing the MDEN bit in the MDCON register, disables the DSM module by automatically switching the carrier high and carrier low signals to the VSS signal source. The modulator signal source is also switched to the MDBIT in the MDCON register. This not only assures that the DSM module is inactive, but that it is also consuming the least amount of current.

The values used to select the carrier high, carrier low, and modulator sources held by the Modulation Source, Modulation High Carrier, and Modulation Low Carrier control registers are not affected when the MDEN bit is cleared and the DSM module is disabled. The values inside these registers remain unchanged while the DSM is inactive. The sources for the carrier high, carrier low and modulator signals will once again be selected when the MDEN bit is set and the DSM module is again enabled and active.

The modulated output signal can be disabled without shutting down the DSM module. The DSM module will remain active and continue to mix signals, but the output value will not be sent to the MDOUT pin. During the time that the output is disabled, the MDOUT pin will remain low. The modulated output can be disabled by clearing the MDOE bit in the MDCON register.

22.2 Modulator Signal Sources

The modulator signal can be supplied from the following sources:

- CCP1 Signal
- CCP2 Signal
- CCP3 Signal
- CCP4 Signal
- MSSP1 SDO1 Signal (SPI Mode Only)
- MSSP2 SDO2 Signal (SPI Mode Only)
- Comparator C1 Signal
- Comparator C2 Signal
- EUSART TX Signal
- External Signal on MDMIN1 pin
- MDBIT bit in the MDCON register

The modulator signal is selected by configuring the MDMS <3:0> bits in the MDSRC register.

22.3 Carrier Signal Sources

The carrier high signal and carrier low signal can be supplied from the following sources:

- CCP1 Signal
- CCP2 Signal
- CCP3 Signal
- CCP4 Signal
- Reference Clock Module Signal
- External Signal on MDCIN1 pin
- External Signal on MDCIN2 pin
- VSS

The carrier high signal is selected by configuring the MDCH <3:0> bits in the MDCARH register. The carrier low signal is selected by configuring the MDCL <3:0> bits in the MDCARL register.

22.4 Carrier Synchronization

During the time when the DSM switches between carrier high and carrier low signal sources, the carrier data in the modulated output signal can become truncated. To prevent this, the carrier signal can be synchronized to the modulator signal. When synchronization is enabled, the carrier pulse that is being mixed at the time of the transition is allowed to transition low before the DSM switches over to the next carrier source.

Synchronization is enabled separately for the carrier high and carrier low signal sources. Synchronization for the carrier high signal can be enabled by setting the MDCHSYNC bit in the MDCARH register. Synchronization for the carrier low signal can be enabled by setting the MDCLSYNC bit in the MDCARL register.

Figure 22-1 through Figure 22-5 show timing diagrams of using various synchronization methods.
FIGURE 22-2: ON OFF KEYING (OOK) SYNCHRONIZATION

<table>
<thead>
<tr>
<th>Carrier Low (CARL)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrier High (CARH)</td>
</tr>
<tr>
<td>Modulator (MOD)</td>
</tr>
<tr>
<td>MDCHSYNC = 1</td>
</tr>
<tr>
<td>MDCLSYNC = 0</td>
</tr>
<tr>
<td>MDCHSYNC = 1</td>
</tr>
<tr>
<td>MDCLSYNC = 1</td>
</tr>
<tr>
<td>MDCHSYNC = 0</td>
</tr>
<tr>
<td>MDCLSYNC = 0</td>
</tr>
<tr>
<td>MDCHSYNC = 0</td>
</tr>
<tr>
<td>MDCLSYNC = 1</td>
</tr>
</tbody>
</table>

EXAMPLE 22-1: NO SYNCHRONIZATION (MDSHSYNC = 0, MDCLSYNC = 0)

<table>
<thead>
<tr>
<th>Carrier High (CARH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrier Low (CARL)</td>
</tr>
<tr>
<td>Modulator (MOD)</td>
</tr>
<tr>
<td>MDCHSYNC = 0</td>
</tr>
<tr>
<td>MDCLSYNC = 0</td>
</tr>
</tbody>
</table>

Active Carrier State

| CARH | CARL | CARH | CARL |

FIGURE 22-3: CARRIER HIGH SYNCHRONIZATION (MDSHSYNC = 1, MDCLSYNC = 0)

<table>
<thead>
<tr>
<th>Carrier High (CARH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carrier Low (CARL)</td>
</tr>
<tr>
<td>Modulator (MOD)</td>
</tr>
<tr>
<td>MDCHSYNC = 1</td>
</tr>
<tr>
<td>MDCLSYNC = 0</td>
</tr>
<tr>
<td>MDCHSYNC = 1</td>
</tr>
<tr>
<td>MDCLSYNC = 0</td>
</tr>
</tbody>
</table>

Active Carrier State

| CARH | both | CARL | both | CARL |
### FIGURE 22-4: CARRIER LOW SYNCHRONIZATION (MDSHSYNC = 0, MDCLSYNC = 1)

<table>
<thead>
<tr>
<th>Carrier High (CARH)</th>
<th>Carrier Low (CARL)</th>
<th>Modulator (MOD)</th>
<th>MDSHSYNC = 0</th>
<th>MDCLSYNC = 1</th>
<th>Active Carrier State</th>
</tr>
</thead>
</table>

### FIGURE 22-5: FULL SYNCHRONIZATION (MDSHSYNC = 1, MDCLSYNC = 1)

<table>
<thead>
<tr>
<th>Carrier High (CARH)</th>
<th>Carrier Low (CARL)</th>
<th>Modulator (MOD)</th>
<th>MDSHSYNC = 1</th>
<th>MDCLSYNC = 1</th>
<th>Active Carrier State</th>
</tr>
</thead>
</table>

Falling edges used to sync
22.5 CARRIER SOURCE POLARITY SELECT

The signal provided from any selected input source for the carrier high and carrier low signals can be inverted. Inverting the signal for the carrier high source is enabled by setting the MDCHPOL bit of the MDCARH register. Inverting the signal for the carrier low source is enabled by setting the MDCLPOL bit of the MDCARL register.

22.6 CARRIER SOURCE PIN DISABLE

Some peripherals assert control over their corresponding output pin when they are enabled. For example, when the CCP1 module is enabled, the output of CCP1 is connected to the CCP1 pin.

This default connection to a pin can be disabled by setting the MDCHODIS bit in the MDCARH register for the carrier high source and the MDCLODIS bit in the MDCARL register for the carrier low source.

22.7 PROGRAMMABLE MODULATOR DATA

The MDBIT of the MDCON register can be selected as the source for the modulator signal. This gives the user the ability to program the value used for modulation.

22.8 MODULATOR SOURCE PIN DISABLE

The modulator source default connection to a pin can be disabled by setting the MDMSODIS bit in the MDSRC register.

22.9 MODULATED OUTPUT POLARITY

The modulated output signal provided on the MDOUT pin can also be inverted. Inverting the modulated output signal is enabled by setting the MDOPOL bit of the MDCON register.

22.10 SLEW RATE CONTROL

When modulated data streams of 20 MHz or greater are required, the slew rate limitation on the output port pin can be disabled. The slew rate limitation can be removed by clearing the MDSLR bit in the MDCON register.

22.11 OPERATION IN SLEEP MODE

The DSM module is not affected by Sleep mode. The DSM can still operate during Sleep, if the Carrier and Modulator input sources are also still operable during Sleep.

22.12 Effects of a Reset

Upon any device Reset, the data signal modulator module is disabled. The user’s firmware is responsible for initializing the module before enabling the output. The registers are reset to their default values.
### REGISTER 22-1: MDCON: MODULATION CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><strong>MDEN</strong>: Modulator Module Enable bit</td>
</tr>
<tr>
<td></td>
<td>1 = Modulator module is enabled and mixing input signals</td>
</tr>
<tr>
<td></td>
<td>0 = Modulator module is disabled and has no output</td>
</tr>
<tr>
<td>6</td>
<td><strong>MDOE</strong>: Modulator Module Pin Output Enable bit</td>
</tr>
<tr>
<td></td>
<td>1 = Modulator pin output enabled</td>
</tr>
<tr>
<td></td>
<td>0 = Modulator pin output disabled</td>
</tr>
<tr>
<td>5</td>
<td><strong>MDSLR</strong>: MDOUT Pin Slew Rate Limiting bit</td>
</tr>
<tr>
<td></td>
<td>1 = MDOUT pin slew rate limiting enabled</td>
</tr>
<tr>
<td></td>
<td>0 = MDOUT pin slew rate limiting disabled</td>
</tr>
<tr>
<td>4</td>
<td><strong>MDOPOL</strong>: Modulator Output Polarity Select bit</td>
</tr>
<tr>
<td></td>
<td>1 = Modulator output signal is inverted</td>
</tr>
<tr>
<td></td>
<td>0 = Modulator output signal is not inverted</td>
</tr>
<tr>
<td>3</td>
<td><strong>MDOUT</strong>: Modulator Output bit</td>
</tr>
<tr>
<td></td>
<td>Displays the current output value of the modulator module</td>
</tr>
<tr>
<td>2-1</td>
<td><strong>Unimplemented</strong>: Read as ‘0’</td>
</tr>
<tr>
<td>0</td>
<td><strong>MDBIT</strong>: Allows software to manually set modulation source input to module</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:** MDBIT must be selected as the modulation source in the MDSRC register for this operation.

**Note 2:** The modulated output frequency can be greater and asynchronous from the clock that updates this register bit, the bit value may not be valid for higher speed modulator or carrier signals.
**REGISTER 22-2: MDSRC: MODULATION SOURCE CONTROL REGISTER**

<table>
<thead>
<tr>
<th>R/W-x/u</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>
<th>R/W-x/u</th>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDMSODIS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MDMS3</td>
<td>MDMS2</td>
<td>MDMS1</td>
<td>MDMS0</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**  
- **MDMSODIS**: Modulation Source Output Disable bit
  - **1** = Output signal driving the peripheral output pin (selected by MDMS<3:0>) is disabled
  - **0** = Output signal driving the peripheral output pin (selected by MDMS<3:0>) is enabled

**bit 6-4**  
- **Unimplemented**: Read as '0'

**bit 3-0**  
- **MDMS<3:0>**: Modulation Source Selection bits
  - 1111 = Reserved. No channel connected.
  - 1110 = Reserved. No channel connected.
  - 1101 = Reserved. No channel connected.
  - 1100 = Reserved. No channel connected.
  - 1011 = Reserved. No channel connected.
  - 1010 = EUSART TX output
  - 1001 = MSSP2 SDOx output
  - 1000 = MSSP1 SDOx output
  - 0111 = Comparator2 output
  - 0110 = Comparator1 output
  - 0101 = CCP4 output (PWM Output mode only)
  - 0100 = CCP3 output (PWM Output mode only)
  - 0011 = CCP2 output (PWM Output mode only)
  - 0010 = CCP1 output (PWM Output mode only)
  - 0001 = MDMIN port pin
  - 0000 = MDBIT bit of MDCON register is modulation source

**Note 1**: Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.
### REGISTER 22-3: MDCARH: MODULATION HIGH CARRIER CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-x/u</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>
<th>R/W-x/u</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDCHODIS</td>
<td>MDCHPOL</td>
<td>MDCHSYNC</td>
<td>—</td>
<td>MDCH3</td>
<td>MDCH2</td>
<td>MDCH1</td>
<td>MDCH0</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
**MDCHODIS:** Modulator High Carrier Output Disable bit
- **1** = Output signal driving the peripheral output pin (selected by MDCH<3:0>) is disabled
- **0** = Output signal driving the peripheral output pin (selected by MDCH<3:0>) is enabled

#### bit 6
**MDCHPOL:** Modulator High Carrier Polarity Select bit
- **1** = Selected high carrier signal is inverted
- **0** = Selected high carrier signal is not inverted

#### bit 5
**MDCHSYNC:** Modulator High Carrier Synchronization Enable bit
- **1** = Modulator waits for a falling edge on the high time carrier signal before allowing a switch to the low time carrier
- **0** = Modulator Output is not synchronized to the high time carrier signal\(^{(1)}\)

#### bit 4
**Unimplemented:** Read as ‘0’

#### bit 3-0
**MDCH<3:0>:** Modulator Data High Carrier Selection bits \(^{(1)}\)
- **1111** = Reserved. No channel connected.
- **1110** = CCP4 output (PWM Output mode only)
- **1110** = CCP3 output (PWM Output mode only)
- **1101** = CCP2 output (PWM Output mode only)
- **1100** = CCP1 output (PWM Output mode only)
- **1011** = Reference clock module signal
- **1010** = MDCIN2 port pin
- **1001** = MDCIN1 port pin
- **1000** = Vss

**Note 1:** Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.
### TABLE 22-1: SUMMARY OF REGISTERS ASSOCIATED WITH DATA SIGNAL MODULATOR 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>MDCARH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>MDCARL</td>
<td>MDCHDIS</td>
<td>MDCHPOL</td>
<td>MDCHSYNC</td>
<td>—</td>
<td>MDCH3</td>
<td>MDCH2</td>
<td>MDCH1</td>
<td>MDCH0</td>
<td>201</td>
</tr>
<tr>
<td>MDCON</td>
<td>MDEN</td>
<td>MDOE</td>
<td>MDSLAR</td>
<td>MDOPOL</td>
<td>MDOUT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>198</td>
</tr>
<tr>
<td>MDSRC</td>
<td>MDMSODIS</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>MDMS3</td>
<td>MDMS2</td>
<td>MDMS1</td>
<td>MDMS0</td>
<td>199</td>
</tr>
</tbody>
</table>

**Legend:**
- — = unimplemented, read as ‘0’.
- Shaded cells are not used in the Data Signal Modulator mode.

**Note 1:** Narrowed carrier pulse widths or spur may occur in the signal stream if the carrier is not synchronized.
23.0 CAPTURE/COMPARE/PWM
MODULES (ECCP1, ECCP2, CCP3, CCP4)

This device family contains up to two Enhanced Capture/Compare/PWM (ECCP1, ECCP2) and up to two standard Capture/Compare/PWM modules (CCP3, CCP4). The CCP3 and CCP4 modules are identical in operation. The ECCP1 and ECCP2 modules may also be referred to as CCP1 and CCP2, as required.
23.1 Capture/Compare/PWM

The Capture/Compare/PWM module is a peripheral which allows the user to time and control different events. In Capture mode, the peripheral allows the timing of the duration of an event. The Compare mode allows the user to trigger an external event when a predetermined amount of time has expired. The PWM mode can generate a Pulse-Width Modulated signal of varying frequency and duty cycle.

Table 23-1 shows the timer resources required by the CCP module.

REGISTER 23-1: CCPXCON: CCPX CONTROL 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>PxM1(1)</td>
<td>PxM0(1)</td>
<td>DCxB1</td>
<td>DCxB0</td>
<td>CCPxM3</td>
<td>CCPxM2</td>
<td>CCPxM1</td>
<td>CCPxM0</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 Reset
‘1’ = Bit is set
‘0’ = Bit is cleared

bit 7-6  PxM<1:0>: Enhanced PWM Output Configuration bits(1)
If CCPxM<3:2> = 00, 01, 10:
xx = PxA assigned as Capture/Compare input; PxB, PxC, PxD assigned as port pins
If CCPxM<3:2> = 11:
00 = Single output; PxA modulated; PxB, PxC, PxD assigned as port pins
01 = Full-Bridge output forward; P1D modulated; P1A active; P1B, P1C inactive
10 = Half-Bridge output; P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins
11 = Full-Bridge output reverse; P1B modulated; P1C active; P1A, P1D inactive

bit 5-4  DCxB<1:0>: PWM Duty Cycle Least Significant bits
Capture mode:
Unused
Compare mode:
Unused
PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.
REGISTER 23-1:  CCPXCON: CCPX CONTROL REGISTER (CONTINUED)

bit 3-0  CCPxM<3:0>: ECCPx Mode Select bits
0000 = Capture/Compare/PWM off (resets ECCPx module)
0001 = Reserved
0010 = Compare mode: toggle output on match
0011 = Reserved
0100 = Capture mode: every falling edge
0101 = Capture mode: every rising edge
0110 = Capture mode: every 4th rising edge
0111 = Capture mode: every 16th rising edge
1000 = Compare mode: initialize ECCPx pin low; set output on compare match (set CCPxIF)
1001 = Compare mode: initialize ECCPx pin high; clear output on compare match (set CCPxIF)
1010 = Compare mode: generate software interrupt only; ECCPx pin reverts to I/O state
1011 = Compare mode: Special Event Trigger

CCP3/CCP4:
11xx = PWM mode

ECCP1/ECCP2:
1100 = PWM mode: PxA, PxC active-high; PxB, PxD active-high
1101 = PWM mode: PxA, PxC active-high; PxB, PxD active-low
1110 = PWM mode: PxA, PxC active-low; PxB, PxD active-high
1111 = PWM mode: PxA, PxC active-low; PxB, PxD active-low

Note 1:  Applies to ECCP modules only.
23.2 CCP Clock Selection

The PIC16F/LF1827 allows each individual CCP module to select the timer source that controls the CCP module. Each module has an independent selection.

As the PIC16F/LF1826/27 has only one 16-bit timer (Timer1), the Capture and Compare modes of the CCP modules always uses Timer1.

The PIC16F/LF1827 has three 8-bit timers with auto-reload (Timer2, Timer4 and Timer6), PWM mode on the CCP modules can use any of these timers. The PIC16F/LF1826 has one 8-bit timer (Timer2), which is used in PWM mode.

REGISTER 23-2: CCPTMRS: CCP TIMERS CONTROL REGISTER(1)

<table>
<thead>
<tr>
<th></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>C4TSEL1</td>
<td>C4TSEL0</td>
<td>C3TSEL1</td>
<td>C3TSEL0</td>
<td>C2TSEL1</td>
<td>C2TSEL0</td>
<td>C1TSEL1</td>
<td>C1TSEL0</td>
<td></td>
</tr>
</tbody>
</table>

bit 7-0

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

bit 7-6  C4TSEL<1:0>: CCP4 Timer Selection bits
00 = CCP4 is based off Timer 2 in PWM Mode
01 = CCP4 is based off Timer 4 in PWM Mode
10 = CCP4 is based off Timer 6 in PWM Mode
11 = Reserved

bit 5-4  C3TSEL<1:0>: CCP3 Timer Selection bits
00 = CCP3 is based off Timer 2 in PWM Mode
01 = CCP3 is based off Timer 4 in PWM Mode
10 = CCP3 is based off Timer 6 in PWM Mode
11 = Reserved

bit 3-2  C2TSEL<1:0>: CCP2 Timer Selection bits
00 = CCP2 is based off Timer 2 in PWM Mode
01 = CCP2 is based off Timer 4 in PWM Mode
10 = CCP2 is based off Timer 6 in PWM Mode
11 = Reserved

bit 1-0  C1TSEL<1:0>: CCP1 Timer Selection bits
00 = CCP1 is based off Timer 2 in PWM Mode
01 = CCP1 is based off Timer 4 in PWM Mode
10 = CCP1 is based off Timer 6 in PWM Mode
11 = Reserved

Note 1: PIC16F/LF1827 only.
23.3 Capture Mode

In Capture mode, the CCPRxH, CCPRxL register pair captures the 16-bit value of the TMR1 register when an event occurs on pin CCPx. An event is defined as one of the following and is configured by the CCPxM<3:0> bits of the CCPxCON register:

- Every falling edge
- Every rising edge
- Every 4th rising edge
- Every 16th rising edge

When a capture is made, the Interrupt Request Flag bit CCPxIF of the PIRx register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRxH, CCPRxL register pair is read, the old captured value is overwritten by the new captured value (see Figure 23-1).

23.3.1 CCPx PIN CONFIGURATION

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit. Also, the CCPx pin function can be moved to alternative pins using the APFCON register. Refer to Section 12.1 “Alternate Pin Function” for more details.

23.3.2 TIMER1 MODE SELECTION

Timer1 must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

23.3.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE interrupt enable bit of the PIEx register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the PIRx register following any change in Operating mode.

23.3.4 CCP PRESCALER

There are four prescaler settings specified by the CCPxM<3:0> bits of the CCPxCON register. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another does not clear the prescaler and may generate a false interrupt. To avoid this unexpected operation, turn the module off by clearing the CCPxCON register before changing the prescaler (see Example 23-1).

23.3.5 CAPTURE DURING SLEEP

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock (Fosc/4), or by an external clock source.

If Timer1 is clocked by Fosc/4, then Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

If Timer1 is clocked by an external clock source, then Capture mode will operate as defined in Section 23.1 “Capture/Compare/PWM”.

---

**FIGURE 23-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**

- Prescaler
- CCPx pin
- System Clock (Fosc)
- CCPRxH, CCPRxL
- Capture
- Enable
- CCPxCON<3:0>
- TMR1H, TMR1L
- Set Flag bit CCPxIF (PIRx register)
- and Edge Detect

**EXAMPLE 23-1: CHANGING BETWEEN CAPTURE PRESCALERS**

```
BANKSEL CCP1CON ;Set Bank bits to point
to CCP1CON
CLRF CCP1CON ;Turn CCP module off
MOVLW NEW_CAPT_PS ;Load the W reg with
                   ;the new prescaler
                   ;move value and CCP ON
MOVWF CCP1CON ;Load CCP1CON with this
               ;value
```
### TABLE 23-2: SUMMARY OF REGISTERS ASSOCIATED WITH CAPTURE

<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>CCPxCON</td>
<td>PxM1</td>
<td>PxM0</td>
<td>DCxH1</td>
<td>DCxH0</td>
<td>CCPxM3</td>
<td>CCPxM2</td>
<td>CCPxM1</td>
<td>CCPxM0</td>
<td>204</td>
</tr>
<tr>
<td>CCPRxL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>207*</td>
</tr>
<tr>
<td>CCPRxH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>207*</td>
</tr>
<tr>
<td>CMxCON0</td>
<td>CxON</td>
<td>CxOUT</td>
<td>CxOE</td>
<td>—</td>
<td>CxPOL</td>
<td>—</td>
<td>CxSP</td>
<td>CxHYS</td>
<td>CxSYNC</td>
</tr>
<tr>
<td>CMxCON1</td>
<td>CxINTP</td>
<td>CxINTN</td>
<td>CxPCH1</td>
<td>CxPCH0</td>
<td>—</td>
<td>—</td>
<td>CxNCH1</td>
<td>CxNCH0</td>
<td>164</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>IO CIF</td>
<td>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCEIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSFIE</td>
<td>C2IE</td>
<td>C1IE</td>
<td>EEIE</td>
<td>BCL1IE</td>
<td>—</td>
<td>—</td>
<td>CCP2IE(2)</td>
<td></td>
</tr>
<tr>
<td>PIE3(2)</td>
<td>—</td>
<td>—</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>TMR6IE</td>
<td>—</td>
<td>TMR4IE</td>
<td>—</td>
<td>91</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSFIF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>EEIF</td>
<td>BCL1IF</td>
<td>—</td>
<td>—</td>
<td>CCP2IE(2)</td>
<td></td>
</tr>
<tr>
<td>PIR3(2)</td>
<td>—</td>
<td>—</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>TMR6IF</td>
<td>—</td>
<td>TMR4IF</td>
<td>—</td>
<td>96</td>
</tr>
<tr>
<td>T1CON</td>
<td>TMR1CS1</td>
<td>TMR1CS0</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCE</td>
<td>T1SYNC</td>
<td>—</td>
<td>TMR1ON</td>
<td></td>
</tr>
<tr>
<td>T1GCON</td>
<td>TMR1GE</td>
<td>T1GPO</td>
<td>T1GTM</td>
<td>T1GSPM</td>
<td>T1GGO/DONE</td>
<td>T1GVAL</td>
<td>T1GSS1</td>
<td>T1GSS0</td>
<td>186</td>
</tr>
<tr>
<td>TMR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>181*</td>
</tr>
<tr>
<td>TMR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>181*</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>120</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
</tbody>
</table>

**Legend:** — = Unimplemented locations, read as ‘0’. Shaded cells are not used by the Capture.

* Page provides register information.

**Note 1:** Applies to ECCP modules only.

**Note 2:** PIC16F/LF1827 only.
23.4  Compare Mode

In Compare mode, the 16-bit CCPRx register value is constantly compared against the TMR1 register pair value. When a match occurs, the CCPx module may:

- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Generate a Special Event Trigger
- Generate a Software Interrupt

The action on the pin is based on the value of the CCPxM<3:0> control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set.

All Compare modes can generate an interrupt.

FIGURE 23-2:  COMPARE MODE OPERATION BLOCK DIAGRAM

23.4.1  CCPX PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the associated TRIS bit.

Also, the CCPx pin function can be moved to alternative pins using the APFCON register. Refer to Section 12.1 “Alternate Pin Function” for more details.

Note: Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

23.4.2  TIMER1 MODE SELECTION

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

Note: Clocking Timer1 from the system clock (Fosc) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (Fosc/4) or from an external clock source.

23.4.3  SOFTWARE INTERRUPT MODE

When Generate Software Interrupt mode is chosen (CCPxM<3:0> = 1010), the CCPx module does not assert control of the CCPx pin (see the CCP1CON register).

23.4.4  SPECIAL EVENT TRIGGER

When Special Event Trigger mode is chosen (CCPxM<3:0> = 1011), the CCPx module does the following:

- Resets Timer1
- Starts an ADC conversion if ADC is enabled

The CCPx module does not assert control of the CCPx pin in this mode (see the CCPxCON register).

The Special Event Trigger output of the CCP occurs immediately upon a match between the TMR1H, TMR1L register pair and the CCPRxH, CCPRXL register pair. The TMR1H, TMR1L register pair is not reset until the next rising edge of the Timer1 clock.

The Special Event Trigger output starts an A/D conversion (if the A/D module is enabled). This feature is only available on CCP4 for PIC16F/LF1827 and ECCP1 on PIC16F/LF1826. This allows the CCPRxH, CCPRXL register pair to effectively provide a 16-bit programmable period register for Timer1.

Note 1: The Special Event Trigger from the CCP module does not set interrupt flag bit TMR1IF of the PIR1 register.

Note 2: Removing the match condition by changing the contents of the CCPRxH and CCPRXL register pair, between the clock edge that generates the Special Event Trigger and the clock edge that generates the Timer1 Reset, will preclude the Reset from occurring.

23.4.5  COMPARE DURING SLEEP

The Compare mode is dependent upon the system clock (Fosc) for proper operation. Since Fosc is shut down during Sleep mode, the Compare mode will not function properly during Sleep.
### TABLE 23-3: SUMMARY OF REGISTERS ASSOCIATED WITH COMPARE

<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>CCPxCON</td>
<td>PxM1(1)</td>
<td>PxM0(1)</td>
<td>DCxB1</td>
<td>DCxB0</td>
<td>CCPxM3</td>
<td>CCPxM2</td>
<td>CCPxM1</td>
<td>CCPxM0</td>
<td>204</td>
</tr>
<tr>
<td>CCPRxL</td>
<td>Capture/Compare/PWM Register x Low Byte (LSB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>207*</td>
</tr>
<tr>
<td>CCPRxH</td>
<td>Capture/Compare/PWM Register x High Byte (MSB)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>207*</td>
</tr>
<tr>
<td>CM1CON0</td>
<td>C1ON</td>
<td>C1OUT</td>
<td>C1OE</td>
<td>C1POL</td>
<td>—</td>
<td>C1SP</td>
<td>C1HYS</td>
<td>C1SYNC</td>
<td>163</td>
</tr>
<tr>
<td>CM1CON1</td>
<td>C1INTP</td>
<td>C1INTN</td>
<td>C1PCH1</td>
<td>C1PCH0</td>
<td>—</td>
<td>—</td>
<td>C1NCH1</td>
<td>C1NCH0</td>
<td>164</td>
</tr>
<tr>
<td>CM2CON0</td>
<td>C2ON</td>
<td>C2OUT</td>
<td>C2OE</td>
<td>C2POL</td>
<td>—</td>
<td>C2SP</td>
<td>C2HYS</td>
<td>C2SYNC</td>
<td>163</td>
</tr>
<tr>
<td>CM2CON1</td>
<td>C2INTP</td>
<td>C2INTN</td>
<td>C2PCH1</td>
<td>C2PCH0</td>
<td>—</td>
<td>—</td>
<td>C2NCH1</td>
<td>C2NCH0</td>
<td>164</td>
</tr>
<tr>
<td>INTCN</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>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSPIE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSFIE</td>
<td>C2IE</td>
<td>C1IE</td>
<td>EEIE</td>
<td>BCL1IE</td>
<td>—</td>
<td>—</td>
<td>CCP2IE(2)</td>
<td>91</td>
</tr>
<tr>
<td>PIE3(3)</td>
<td>—</td>
<td>—</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>TMR6IE</td>
<td>—</td>
<td>TMR4IE</td>
<td>—</td>
<td>92</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSPIF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSFIF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>EEIF</td>
<td>BCLIF</td>
<td>—</td>
<td>—</td>
<td>CCP2IF(2)</td>
<td>95</td>
</tr>
<tr>
<td>PIR3(3)</td>
<td>—</td>
<td>—</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>TMR6IF</td>
<td>—</td>
<td>TMR4IF</td>
<td>—</td>
<td>96</td>
</tr>
<tr>
<td>T1CON</td>
<td>TMR1CS1</td>
<td>TMR1CS0</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>TTSYNC</td>
<td>—</td>
<td>TMR1ON</td>
<td>185</td>
</tr>
<tr>
<td>T1GCON</td>
<td>TMR1GIE</td>
<td>T1GPOL</td>
<td>T1GTM</td>
<td>T1GSPM</td>
<td>T1GGO/DONE</td>
<td>T1GVAL</td>
<td>T1GSS1</td>
<td>T1GSS0</td>
<td>186</td>
</tr>
<tr>
<td>TMR1L</td>
<td>Holding Register for the Least Significant Byte of the 16-bit TMR1 Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>181*</td>
</tr>
<tr>
<td>TMR1H</td>
<td>Holding Register for the Most Significant Byte of the 16-bit TMR1 Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>181*</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>120</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
</tbody>
</table>

**Legend:** — Unimplemented locations, read as ‘0’. Shaded cells are not used by the Compare.

* Page provides register information.

**Note 1:** Applies to ECCP modules only.

**Note 2:** PIC16F/LF1827 only.
23.5 PWM Mode

The PWM mode generates a Pulse-Width Modulated signal on the CCPx pin. The duty cycle, period and resolution are determined by the following registers:

- PRx
- TxCON
- CCPRxL
- CCPxCON

The ECCP modules have the following additional registers:

- ECCPxAS
- PSTRxCON
- PWMxCON

In Pulse-Width Modulation (PWM) mode, the CCPx module produces up to a 10-bit resolution PWM output on the CCPx pin. Since the CCPx pin is multiplexed with the PORT data latch, the TRIS for that pin must be cleared to enable the CCPx pin output driver.

**Note:** Clearing the CCPxCON register will relinquish CCPx control of the CCPx pin.

The CCPx module in PWM mode can have the PWM based off of either Timer2, Timer4 or Timer6. This is controlled by the CCPTMRS0 and CCPTMRS1 registers. Reference Section 23.2 “CCP Clock Selection” for more information.

Figure 23-3 shows a simplified block diagram of PWM operation.

Figure 23-4 shows a typical waveform of the PWM signal.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see Section 23.5.7 “Setup for PWM Operation”.

**FIGURE 23-3: SIMPLIFIED PWM BLOCK DIAGRAM**

The PWM output (Figure 23-4) has a time base (period) and a time that the output stays high (duty cycle).

**FIGURE 23-4: CCP PWM OUTPUT**

The 8-bit timer TMR2 register is concatenated with the 2-bit internal system clock (Fosc), or 2 bits of the prescaler, to create the 10-bit time base.

1: In PWM mode, CCPRxH is a read-only register.

2: Clear Timerx, toggle CCPx pin and latch duty cycle.
23.5.1 PWM PERIOD

The PWM period is specified by the PRx register of Timerx. The PWM period can be calculated using the formula of Equation 23-1.

EQUATION 23-1: PWM PERIOD

\[
PWM \ Period = \left(\frac{(PRx + 1)}{4}\right) \cdot T_{osc} \cdot (TMRx \ Prescale \ Value)
\]

Note 1: \( T_{osc} = \frac{1}{F_{OSC}} \)

When TMRx is equal to PRx, the following three events occur on the next increment cycle:

- TMRx is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is latched from CCPRxL into CCPRxH.

The Timerx postscaler (see Section 21.1 “Timer2/4/6 Operation”) is not used in the determination of the PWM frequency.

23.5.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to multiple registers: CCPRxL register and DCxB<1:0> bits of the CCPxCON register. The CCPRxL contains the eight MSbs and the DCxB<1:0> bits of the CCPxCON register contain the two LSbs. CCPRxL and DCxB<1:0> bits of the CCPxCON register can be written to at any time. The duty cycle value is not latched into CCPRxH until after the period completes (i.e., a match between PRx and TMRx registers occurs). While using the PWM, the CCPRxH register is read-only.

Equation 23-2 is used to calculate the PWM pulse width.

Equation 23-3 is used to calculate the PWM duty cycle ratio.

EQUATION 23-2: PULSE WIDTH

\[
Pulse \ Width = (CCPRxL:CCPxCON<5:4>) \cdot T_{osc} \cdot (TMRx \ Prescale \ Value)
\]

EQUATION 23-3: DUTY CYCLE RATIO

\[
Duty \ Cycle \ Ratio = \left(\frac{CCPRxL:CCPxCON<5:4>}{4(PRx + 1)}\right)
\]

The CCPRxH register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

The 8-bit timer TMRx register is concatenated with either the 2-bit internal system clock (Fosc), or 2 bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timerx prescaler is set to 1:1.

When the 10-bit time base matches the CCPRxH and 2-bit latch, then the CCPx pin is cleared (see Figure 23-3).
23.5.3 PWM RESOLUTION

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is 10 bits when PRx is 255. The resolution is a function of the PRx register value as shown by Equation 23-4.

\[
Resolution = \frac{\log[4(PRx + 1)]}{\log(2)} \text{ bits}
\]

Note: If the pulse width value is greater than the period the assigned PWM pin(s) will remain unchanged.

<table>
<thead>
<tr>
<th>TABLE 23-4:</th>
<th>EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 32 MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM Frequency</td>
<td>1.95 kHz</td>
</tr>
<tr>
<td>Timer Prescale (1, 4, 16)</td>
<td>16</td>
</tr>
<tr>
<td>PRx Value</td>
<td>0xFF</td>
</tr>
<tr>
<td>Maximum Resolution (bits)</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 23-5:</th>
<th>EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 20 MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM Frequency</td>
<td>1.22 kHz</td>
</tr>
<tr>
<td>Timer Prescale (1, 4, 16)</td>
<td>16</td>
</tr>
<tr>
<td>PRx Value</td>
<td>0xFF</td>
</tr>
<tr>
<td>Maximum Resolution (bits)</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 23-6:</th>
<th>EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 8 MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM Frequency</td>
<td>1.22 kHz</td>
</tr>
<tr>
<td>Timer Prescale (1, 4, 16)</td>
<td>16</td>
</tr>
<tr>
<td>PRx Value</td>
<td>0x65</td>
</tr>
<tr>
<td>Maximum Resolution (bits)</td>
<td>8</td>
</tr>
</tbody>
</table>
23.5.4 OPERATION IN SLEEP MODE

In Sleep mode, the TMRx register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, TMRx will continue from its previous state.

23.5.5 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency. Any changes in the system clock frequency will result in changes to the PWM frequency. See Section 5.0 “Oscillator Module (With Fail-Safe Clock Monitor)” for additional details.

23.5.6 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

23.5.7 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Disable the PWM pin (CCPx) output driver(s) by setting the associated TRIS bit(s).
2. Load the PRx register with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRxL register and the DCxBx bits of the CCPxCON register, with the PWM duty cycle value.
5. Configure and start Timerx:
   • Clear the TMRxIF interrupt flag bit of the PIRx register. See Note below.
   • Configure the TxCKPS bits of the TxCON register with the Timerx prescale value.
   • Enable Timerx by setting the TMRxON bit of the T2CON register.
6. Enable PWM output pin:
   • Wait until Timerx overflows, TMRxIF bit of the PIR1 register is set. See Note below.
   • Enable the PWM pin (CCPx) output driver(s) by clearing the associated TRIS bit(s).

Note: In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.
23.6  PWM (Enhanced Mode)

The Enhanced PWM mode can generate a PWM signal on up to four different output pins with up to 10-bits of resolution. It can do this through four different PWM output modes:

- Single PWM
- Half-Bridge PWM
- Full-Bridge PWM, Forward mode
- Full-Bridge PWM, Reverse mode

To select an Enhanced PWM mode, the P1M bits of the CCP1CON register must be set appropriately.

The PWM outputs are multiplexed with I/O pins and are designated P1A, P1B, P1C and P1D. The polarity of the PWM pins is configurable and is selected by setting the CCP1M bits in the CCP1CON register appropriately.

Table 23-7 shows the pin assignments for each Enhanced PWM mode.

Figure 23-5 shows an example of a simplified block diagram of the Enhanced PWM module.

Note:  To prevent the generation of an incomplete waveform when the PWM is first enabled, the ECCP module waits until the start of a new PWM period before generating a PWM signal.

### FIGURE 23-5:  EXAMPLE SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODE

### TABLE 23-7:  EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES

<table>
<thead>
<tr>
<th>ECCP Mode</th>
<th>PxM&lt;1:0&gt;</th>
<th>CCPx/P1A</th>
<th>P1B</th>
<th>P1C</th>
<th>P1D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>00</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
<td>Yes(1)</td>
</tr>
<tr>
<td>Half-Bridge</td>
<td>10</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Full-Bridge, Forward</td>
<td>01</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Full-Bridge, Reverse</td>
<td>11</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note 1: PWM Steering enables outputs in Single mode.

Note 1: The TRIS register value for each PWM output must be configured appropriately.

2: Clearing the CCPxCON register will relinquish ECCP control of all PWM output pins.

3: Any pin not used by an Enhanced PWM mode is available for alternate pin functions.
FIGURE 23-6: EXAMPLE PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)

<table>
<thead>
<tr>
<th>P&lt;sub&gt;x&lt;/sub&gt;M&lt;sub&gt;&lt;1:0&gt;&lt;/sub&gt;</th>
<th>Signal</th>
<th>PR&lt;sub&gt;x&lt;/sub&gt;+1</th>
<th>Pulse Width</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 (Single Output)</td>
<td>P1A Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 (Half-Bridge)</td>
<td>P1B Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1A Active</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 (Full-Bridge, Forward)</td>
<td>P1B Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1C Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1D Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1A Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1B Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1C Active</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1D Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relationships:
- Period = 4 * T<sub>osc</sub> * (PR<sub>x</sub> + 1) * (TMR<sub>x</sub> Prescale Value)
- Pulse Width = T<sub>osc</sub> * (CCPR<sub>x</sub>L<7:0>:CCPxCON<5:4>) * (TMR<sub>x</sub> Prescale Value)
- Delay = 4 * T<sub>osc</sub> * (PWM<sub>x</sub>CON<6:0>)

Note 1: Dead-band delay is programmed using the PWM<sub>x</sub>CON register (Section 23.6.6 “Programmable Dead-Band Delay Mode”).
FIGURE 23-7:  EXAMPLE ENHANCED PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)

<table>
<thead>
<tr>
<th>PxM&lt;1:0&gt;</th>
<th>Signal</th>
<th>0 Pulse Width</th>
<th>Period</th>
<th>PRx+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 (Single Output)</td>
<td>P1A Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 (Half-Bridge)</td>
<td>P1A Modulated</td>
<td>Delay(1)</td>
<td>Delay(1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1B Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01 (Full-Bridge, Forward)</td>
<td>P1B Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1C Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1D Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 (Full-Bridge, Reverse)</td>
<td>P1A Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1B Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1C Active</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1D Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relationships:

• Period = 4 \* Tosc \* (PRx + 1) \* (TMRx Prescale Value)
• Pulse Width = Tosc \* (CCPRxL<7:0>:CCPxCON<5:4>) \* (TMRx Prescale Value)
• Delay = 4 \* Tosc \* (PWMxCON<6:0>)

Note 1: Dead-band delay is programmed using the PWMxCON register (Section 23.6.6 “Programmable Dead-Band Delay Mode”).
23.6.1 HALF-BRIDGE MODE

In Half-Bridge mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the CCPx/P1A pin, while the complementary PWM output signal is output on the P1B pin (see Figure 23-9). This mode can be used for Half-Bridge applications, as shown in Figure 23-9, or for Full-Bridge applications, where four power switches are being modulated with two PWM signals.

In Half-Bridge mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of the PDC<6:0> bits of the PWMxCON register sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See Section 23.6.6 “Programmable Dead-Band Delay Mode” for more details of the dead-band delay operations.

Since the P1A and P1B outputs are multiplexed with the PORT data latches, the associated TRIS bits must be cleared to configure P1A and P1B as outputs.

FIGURE 23-8: EXAMPLE OF HALF-BRIDGE PWM OUTPUT

Note 1: At this time, the TMRx register is equal to the PRx register.
2: Output signals are shown as active-high.

FIGURE 23-9: EXAMPLE OF HALF-BRIDGE APPLICATIONS
23.6.2 FULL-BRIDGE MODE

In Full-Bridge mode, all four pins are used as outputs. An example of Full-Bridge application is shown in Figure 23-10.

In the Forward mode, pin CCPx/P1A is driven to its active state, pin P1D is modulated, while P1B and P1C will be driven to their inactive state as shown in Figure 23-11.

In the Reverse mode, P1C is driven to its active state, pin P1B is modulated, while P1A and P1D will be driven to their inactive state as shown Figure 23-11.

P1A, P1B, P1C and P1D outputs are multiplexed with the PORT data latches. The associated TRIS bits must be cleared to configure the P1A, P1B, P1C and P1D pins as outputs.

FIGURE 23-10: EXAMPLE OF FULL-BRIDGE APPLICATION
FIGURE 23-11: EXAMPLE OF FULL-BRIDGE PWM OUTPUT

Forward Mode

Period
Pulse Width

P1A (2)

P1B (2)

P1C (2)

P1D (2)

Reverse Mode

Period
Pulse Width

P1A (2)

P1B (2)

P1C (2)

P1D (2)

Note 1: At this time, the TMRx register is equal to the PRx register.
2: Output signal is shown as active-high.
23.6.2.1 Direction Change in Full-Bridge Mode

In the Full-Bridge mode, the PxM1 bit in the CCPxCON register allows users to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will change to the new direction on the next PWM cycle.

A direction change is initiated in software by changing the PxM1 bit of the CCPxCON register. The following sequence occurs four Timerx cycles prior to the end of the current PWM period:

- The modulated outputs (P1B and P1D) are placed in their inactive state.
- The associated unmodulated outputs (P1A and P1C) are switched to drive in the opposite direction.
- PWM modulation resumes at the beginning of the next period.

See Figure 23-12 for an illustration of this sequence.

The Full-Bridge mode does not provide dead-band delay. As one output is modulated at a time, dead-band delay is generally not required. There is a situation where dead-band delay is required. This situation occurs when both of the following conditions are true:

1. The direction of the PWM output changes when the duty cycle of the output is at or near 100%.
2. The turn off time of the power switch, including the power device and driver circuit, is greater than the turn on time.

Figure 23-13 shows an example of the PWM direction changing from forward to reverse, at a near 100% duty cycle. In this example, at time t1, the output P1A and P1D become inactive, while output P1C becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through power devices QC and QD (see Figure 23-10) for the duration of ‘t’. The same phenomenon will occur to power devices QA and QB for PWM direction change from reverse to forward.

If changing PWM direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

1. Reduce PWM duty cycle for one PWM period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

Other options to prevent shoot-through current may exist.

**FIGURE 23-12: EXAMPLE OF PWM DIRECTION CHANGE**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Period(1)</th>
<th>Pulse Width</th>
<th>Period</th>
<th>Pulse Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1A (Active-High)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1B (Active-High)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1C (Active-High)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1D (Active-High)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** The direction bit PxM1 of the CCPxCON register is written any time during the PWM cycle.

**Note 2:** When changing directions, the P1A and P1C signals switch before the end of the current PWM cycle. The modulated P1B and P1D signals are inactive at this time. The length of this time is four Timerx counts.
FIGURE 23-13: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE

Note 1: All signals are shown as active-high.
2: TON is the turn on delay of power switch QC and its driver.
3: TOFF is the turn off delay of power switch QD and its driver.
23.6.3 START-UP CONSIDERATIONS
When any PWM mode is used, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins.

**Note:** When the microcontroller is released from Reset, all of the I/O pins are in the high-impedance state. The external circuits must keep the power switch devices in the Off state until the microcontroller drives the I/O pins with the proper signal levels or activates the PWM output(s).

The CCPxM<1:0> bits of the CCPxCON register allow the user to choose whether the PWM output signals are active-high or active-low for each pair of PWM output pins (P1A/P1C and P1B/P1D). The PWM output polarities must be selected before the PWM pin output drivers are enabled. Changing the polarity configuration while the PWM pin output drivers are enabled is not recommended since it may result in damage to the application circuits.

The P1A, P1B, P1C and P1D output latches may not be in the proper states when the PWM module is initialized. Enabling the PWM pin output drivers at the same time as the Enhanced PWM modes may cause damage to the application circuit. The Enhanced PWM modes must be enabled in the proper Output mode and complete a full PWM cycle before enabling the PWM pin output drivers. The completion of a full PWM cycle is indicated by the TMRxIF bit of the PIRx register being set as the second PWM period begins.

23.6.4 ENHANCED PWM AUTO-SHUTDOWN MODE
The PWM mode supports an Auto-Shutdown mode that will disable the PWM outputs when an external shutdown event occurs. Auto-Shutdown mode places the PWM output pins into a predetermined state. This mode is used to help prevent the PWM from damaging the application.

The auto-shutdown sources are selected using the CCPxAS<2:0> bits of the CCPxAS register. A shutdown event may be generated by:
- A logic ‘0’ on the INT pin
- Comparator Cx
- Setting the CCPxASE bit in firmware

A shutdown condition is indicated by the CCPxASE (Auto-Shutdown Event Status) bit of the CCPxAS register. If the bit is a ‘0’, the PWM pins are operating normally. If the bit is a ‘1’, the PWM outputs are in the shutdown state.

When a shutdown event occurs, two things happen:
- The CCPxASE bit is set to ‘1’. The CCPxASE will remain set until cleared in firmware or an auto-restart occurs (see Section 23.6.5 “Auto-Restart Mode”).

The enabled PWM pins are asynchronously placed in their shutdown states. The PWM output pins are grouped into pairs [P1A/P1C] and [P1B/P1D]. The state of each pin pair is determined by the PSSxAC and PSSxBD bits of the CCPxAS register. Each pin pair may be placed into one of three states:
- Drive logic ‘1’
- Drive logic ‘0’
- Tri-state (high-impedance)
### REGISTER 23-3: CCPXAS: CCPX AUTO-SHUTDOWN CONTROL REGISTER

<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>CCPxASE: CCPx Auto-Shutdown Event Status bit</td>
<td>CCPx Auto-Shutdown Source Select bits</td>
<td>Pins P1A and P1C Shutdown State Control bits</td>
<td>Pins P1B and P1D Shutdown State Control bits</td>
</tr>
<tr>
<td>1 = A shutdown event has occurred; CCPx outputs are in shutdown state</td>
<td>000 = Auto-shutdown is disabled</td>
<td>00 = Drive pins P1A and P1C to ‘0’</td>
<td>00 = Drive pins P1B and P1D to ‘0’</td>
</tr>
<tr>
<td>0 = CCPx outputs are operating</td>
<td>001 = Comparator C1 output low</td>
<td>01 = Drive pins P1A and P1C to ‘1’</td>
<td>01 = Drive pins P1B and P1D to ‘1’</td>
</tr>
<tr>
<td></td>
<td>010 = Comparator C2 output low</td>
<td>1x = Pins P1A and P1C tri-state</td>
<td>1x = Pins P1B and P1D tri-state</td>
</tr>
<tr>
<td></td>
<td>011 = Either Comparator C1 or C2 low</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>100 = VIL on INT pin</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>101 = VIL on INT pin or Comparator C1 low</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>110 = VIL on INT pin or Comparator C2 low</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>111 = VIL on INT pin or Comparator C1 or Comparator C2 low</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:** If CxSYNC is enabled, the shutdown will be delayed by Timer1.

**Note 2:** The auto-shutdown condition is a level-based signal, not an edge-based signal. As long as the level is present, the auto-shutdown will persist.

**Note 3:** Writing to the CCPxASE bit is disabled while an auto-shutdown condition persists.

**Note 4:** Once the auto-shutdown condition has been removed and the PWM restarted (either through firmware or auto-restart) the PWM signal will always restart at the beginning of the next PWM period.

---

Register 23-3: CCPXAS: CCPX Auto-Shutdown Control Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>CCPxASE: CCPx Auto-Shutdown Event Status bit</td>
</tr>
<tr>
<td>6-4</td>
<td>CCPx Auto-Shutdown Source Select bits</td>
</tr>
<tr>
<td>3-2</td>
<td>Pins P1A and P1C Shutdown State Control bits</td>
</tr>
<tr>
<td>1-0</td>
<td>Pins P1B and P1D Shutdown State Control bits</td>
</tr>
</tbody>
</table>
23.6.5 AUTO-RESTART MODE

The Enhanced PWM can be configured to automatically restart the PWM signal once the auto-shutdown condition has been removed. Auto-restart is enabled by setting the PxRSEN bit in the PWMxCON register.

If auto-restart is enabled, the CCPxASE bit will remain set as long as the auto-shutdown condition is active. When the auto-shutdown condition is removed, the CCPxASE bit will be cleared via hardware and normal operation will resume.
23.6.6 PROGRAMMABLE DEAD-BAND DELAY MODE

In half-bridge applications where all power switches are modulated at the PWM frequency, the power switches normally require more time to turn off than to turn on. If both the upper and lower power switches are switched at the same time (one turned on, and the other turned off), both switches may be on for a short period of time until one switch completely turns off. During this brief interval, a very high current (*shoot-through current*) will flow through both power switches, shorting the bridge supply. To avoid this potentially destructive shoot-through current from flowing during switching, turning on either of the power switches is normally delayed to allow the other switch to completely turn off.

In Half-Bridge mode, a digitally programmable dead-band delay is available to avoid shoot-through current from destroying the bridge power switches. The delay occurs at the signal transition from the non-active state to the active state. See Figure 23-16 for illustration. The lower seven bits of the associated PWMxCON register (Register 23-4) sets the delay period in terms of microcontroller instruction cycles (TCY or 4 TOSC).

**FIGURE 23-16: EXAMPLE OF HALF-BRIDGE PWM OUTPUT**

- **td** = Dead-Band Delay

**Note 1:** At this time, the TMRx register is equal to the PRx register.
**Note 2:** Output signals are shown as active-high.

**FIGURE 23-17: EXAMPLE OF HALF-BRIDGE APPLICATIONS**

<table>
<thead>
<tr>
<th>Period</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1A(2)</td>
<td>td</td>
</tr>
<tr>
<td>P1B(2)</td>
<td>td</td>
</tr>
</tbody>
</table>

Standard Half-Bridge Circuit (“Push-Pull”)
**REGISTER 23-4: PWMxCON: ENHANCED PWM CONTROL 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>PxRSEN</td>
<td>PxDC6</td>
<td>PxDC5</td>
<td>PxDC4</td>
<td>PxDC3</td>
<td>PxDC2</td>
<td>PxDC1</td>
<td>PxDC0</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**  
PxRSEN: PWM Restart Enable bit  
- 1 = Upon auto-shutdown, the CCPxASE bit clears automatically once the shutdown event goes away; the PWM restarts automatically  
- 0 = Upon auto-shutdown, CCPxASE must be cleared in software to restart the PWM

**bit 6-0**  
PxDC<6:0>: PWM Delay Count bits  
PxDCx = Number of Fosc/4 (4 * Tosc) cycles between the scheduled time when a PWM signal should transition active and the actual time it transitions active

**Note 1:** Bit resets to ‘0’ with Two-Speed Start-up and LP, XT or HS selected as the Oscillator mode or Fail-Safe mode is enabled.
23.6.7 PWM STEERING MODE

In Single Output mode, PWM steering allows any of the PWM pins to be the modulated signal. Additionally, the same PWM signal can be simultaneously available on multiple pins.

Once the Single Output mode is selected (CCPxM<3:2> = 11 and PxM<1:0> = 00 of the CCPxCON register), the user firmware can bring out the same PWM signal to one, two, three or four output pins by setting the appropriate STRx<D:A> bits of the PSTRxCON register, as shown in Table 23-7.

Note: The associated TRIS bits must be set to output ('0') to enable the pin output driver in order to see the PWM signal on the pin.

While the PWM Steering mode is active, CCPxM<1:0> bits of the CCPxCON register select the PWM output polarity for the P1<D:A> pins.

The PWM auto-shutdown operation also applies to PWM Steering mode as described in Section 23.6.4 “Enhanced PWM Auto-shutdown mode”. An auto-shutdown event will only affect pins that have PWM outputs enabled.

REGISTER 23-5: PSTRXCON: PWM STEERING CONTROL REGISTER(1)

<table>
<thead>
<tr>
<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>R/W-0/0</th>
<th>R/W-1/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>STRxSYNC</td>
<td>STRxD</td>
<td>STRxC</td>
<td>STRxB</td>
<td>STRxA</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 7-5 Unimplemented: Read as ‘0’
bite 4 STRxSYNC: Steering Sync bit
  1 = Output steering update occurs on next PWM period
  0 = Output steering update occurs at the beginning of the instruction cycle boundary

bit 3 STRxD: Steering Enable bit D
  1 = P1D pin has the PWM waveform with polarity control from CCPxM<1:0>
  0 = P1D pin is assigned to port pin

bit 2 STRxC: Steering Enable bit C
  1 = P1C pin has the PWM waveform with polarity control from CCPxM<1:0>
  0 = P1C pin is assigned to port pin

bit 1 STRxB: Steering Enable bit B
  1 = P1B pin has the PWM waveform with polarity control from CCPxM<1:0>
  0 = P1B pin is assigned to port pin

bit 0 STRxA: Steering Enable bit A
  1 = P1A pin has the PWM waveform with polarity control from CCPxM<1:0>
  0 = P1A pin is assigned to port pin

Note 1: The PWM Steering mode is available only when the CCPxCON register bits CCPxM<3:2> = 11 and PxM<1:0> = 00.
FIGURE 23-18: SIMPLIFIED STEERING BLOCK DIAGRAM

Note 1: Port outputs are configured as shown when the CCPxCON register bits PxM<1:0> = 00 and CCPxM<3:2> = 11.

2: Single PWM output requires setting at least one of the STRx bits.
23.6.7.1 Steering Synchronization

The STRxSYNC bit of the PSTRxCON register gives the user two selections of when the steering event will happen. When the STRxSYNC bit is ‘0’, the steering event will happen at the end of the instruction that writes to the PSTRxCON register. In this case, the output signal at the P1<D:A> pins may be an incomplete PWM waveform. This operation is useful when the user firmware needs to immediately remove a PWM signal from the pin.

When the STRxSYNC bit is ‘1’, the effective steering update will happen at the beginning of the next PWM period. In this case, steering on/off the PWM output will always produce a complete PWM waveform.

Figures 23-19 and 23-20 illustrate the timing diagrams of the PWM steering depending on the STRXSYNC setting.

FIGURE 23-19: EXAMPLE OF STEERING EVENT AT END OF INSTRUCTION (STRXSYNC = 0)

FIGURE 23-20: EXAMPLE OF STEERING EVENT AT BEGINNING OF INSTRUCTION (STRXSYNC = 1)
### TABLE 23-8: SUMMARY OF REGISTERS ASSOCIATED WITH PWM

<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>CCPxCON</td>
<td>PxM1(1)</td>
<td>PxM0(1)</td>
<td>DCxB1</td>
<td>DCxB0</td>
<td>CCPxM3</td>
<td>CCPxM2</td>
<td>CCPxM1</td>
<td>CCPxM0</td>
<td>204</td>
</tr>
<tr>
<td>CCPxAS</td>
<td>CCPxASE</td>
<td>CCPxAS2</td>
<td>CCPxAS1</td>
<td>CCPxAS0</td>
<td>PSSxAC1</td>
<td>PSSxAC0</td>
<td>PSSxBD1</td>
<td>PSSxBD0</td>
<td>224</td>
</tr>
<tr>
<td>CCPTMRS</td>
<td>C4TSEL1</td>
<td>C4TSEL0</td>
<td>C3TSEL1</td>
<td>C3TSEL0</td>
<td>C2TSEL1</td>
<td>C2TSEL0</td>
<td>C1TSEL1</td>
<td>C1TSEL0</td>
<td>206</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>89</td>
</tr>
<tr>
<td>PRx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>89</td>
</tr>
<tr>
<td>PSTRxCON</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STRxSYNC</td>
<td>STRxD</td>
<td>STRxC</td>
<td>STRxB</td>
<td>228</td>
</tr>
<tr>
<td>PWMxCON</td>
<td>PxRSEN</td>
<td>PxDC6</td>
<td>PxDC5</td>
<td>PxDC4</td>
<td>PxDC3</td>
<td>PxDC2</td>
<td>PxDC1</td>
<td>PxDC0</td>
<td>227</td>
</tr>
<tr>
<td>TxCON</td>
<td></td>
<td>TxB0</td>
<td>TxA0</td>
<td>TxB2</td>
<td>TxA2</td>
<td>TxB1</td>
<td>TxA1</td>
<td>126</td>
<td></td>
</tr>
<tr>
<td>TMRx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TxB0</td>
<td>TxA0</td>
<td>TxB2</td>
<td>TxA2</td>
<td>126</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
</tbody>
</table>

**Legend:** — = Unimplemented locations, read as ‘0’. Shaded cells are not used by the PWM.

**Note 1:** Applies to ECCP modules only.

* Page provides register information.
24.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP1 AND MSSP2) MODULE

24.1 Master MSSPx (MSSPx) Module Overview

The Master Synchronous Serial Port (MSSPx) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be Serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSPx module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (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 24-1 is a block diagram of the SPI interface module.

**FIGURE 24-1: MSSPX BLOCK DIAGRAM (SPI MODE)**
The I²C interface supports the following modes and features:

- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited Multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection
- General call address matching
- Address masking
- Address Hold and Data Hold modes
- Selectable SDAx hold times

Figure 24-2 is a block diagram of the I²C interface module in Master mode. Figure 24-3 is a diagram of the I²C interface module in Slave mode.

The PIC16F1827 has two MSSP modules, MSSP1 and MSSP2, each module operating independently from the other.

**Note 1:** In devices with more than one MSSP module, it is very important to pay close attention to SSPxCONx register names. SSP1CON1 and SSP1CON2 registers control different operational aspects of the same module, while SSP1CON1 and SSP2CON1 control the same features for two different modules.

**2:** Throughout this section, generic references to an MSSP module in any of its operating modes may be interpreted as being equally applicable to MSSP1 or MSSP2. Register names, module I/O signals, and bit names may use the generic designator ‘x’ to indicate the use of a numeral to distinguish a particular module when required.
FIGURE 24-3: MSSPX BLOCK DIAGRAM (I²C™ SLAVE MODE)
24.2 SPI Mode Overview

The Serial Peripheral Interface (SPI) bus is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a chip select known as Slave Select.

The SPI bus specifies four signal connections:
- Serial Clock (SCKx)
- Serial Data Out (SDOx)
- Serial Data In (SDIx)
- Slave Select (SSx)

Figure 24-1 shows the block diagram of the MSSPx module when operating in SPI Mode.

The SPI bus operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

Figure 24-4 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 24-5 shows a typical connection between two processors configured as master and slave devices.

Data is shifted out of both shift registers on the programmed clock edge and latched on the opposite edge of the clock.

The master device transmits information out on its SDOx output pin which is connected to, and received by, the slave’s SDIx input pin. The slave device transmits information out on its SDOx output pin, which is connected to, and received by, the master’s SDIx input pin.

To begin communication, the master device first sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

The master device starts a transmission by sending out the MSb from it’s shift register. The slave device reads this bit from that same line and saves it into the LSb position of it’s shift register.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that at the same time, the slave device is sending out the MSb from it’s shift register and the master device is reading this bit from that same line and saving it as the LSb of it’s shift register.

After 8 bits have been shifted out, the master and slave have exchanged register values.

If there is more data to exchange, the shift registers are loaded with new data and the process repeats itself.

Whether the data is meaningful or not (dummy data), depends on the application software. This leads to three scenarios for data transmission:
- Master sends useful data and slave sends dummy data.
- Master sends useful data and slave sends useful data.
- Master sends dummy data and slave sends useful data.

Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops sending the clock signal and it deselects the slave.

Every slave device connected to the bus that has not been selected through its slave select line must disregard the clock and transmission signals and must not transmit out any data of its own.
24.2.1 SPI MODE REGISTERS

The MSSPx module has five registers for SPI mode operation. These are:

- MSSPx STATUS register (SSPxSTAT)
- MSSPx Control Register 1 (SSPxCON1)
- MSSPx Control Register 3 (SSPxCON3)
- MSSPx Data Buffer register (SSPxBUF)
- MSSPx Address register (SSPxADD)
- MSSPx Shift register (SSPxSR)
  (Not directly accessible)

SSPxCON1 and SSPxSTAT are the control and STATUS registers in SPI mode operation. The SSPxCON1 register is readable and writable. The lower 6 bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

In one SPI Master mode, SSPxADD can be loaded with a value used in the Baud Rate Generator. More information on the Baud Rate Generator is available in Section 24.7 “Baud Rate Generator”.

SSPxSR is the shift register used for shifting data in and out. SSPxBUF provides indirect access to the SSPxSR register. SSPxBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPxSR and SSPxBUF together create a buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxFIF interrupt is set.

During transmission, the SSPxBUF is not buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.
24.2.2 SPI MODE OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPxC<5:0> and SSPxSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCKx is the clock output)
- Slave mode (SCKx is the clock input)
- Clock Polarity (Idle state of SCKx)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCKx)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

To enable the serial port, SSPx Enable bit, SSPxEN of the SSPxC register, must be set. To reset or reconfigure SPI mode, clear the SSPxEN bit, re-initialize the SSPxC registers and then set the SSPxEN bit. This configures the SDIx, SDOx, SCKx and SSx pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDIx must have corresponding TRIS bit set
- SDOx must have corresponding TRIS bit cleared
- SCKx (Master mode) must have corresponding TRIS bit cleared
- SCKx (Slave mode) must have corresponding TRIS bit set
- SSx must have corresponding TRIS bit set

Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

The MSSPx consists of a transmit/receive shift register (SSPxSR) and a buffer register (SSPxBUF). The SSPxSR shifts the data in and out of the device, MSb first. The SSPxBUF holds the data that was written to the SSPxSR until the received data is ready. Once the 8 bits of data have been received, that byte is moved to the SSPxBUF register. Then, the Buffer Full Detect bit, BF of the SSPxSTAT register, and the interrupt flag bit, SSPxF, are set. This double-buffering of the received data (SSPxBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPxBUF register during transmission/reception of data will be ignored and the write collision detect bit WCOL of the SSPxC register, will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPxBUF register to complete successfully.

When the application software is expecting to receive valid data, the SSPxBUF should be read before the next byte of data to transfer is written to the SSPxBUF. The Buffer Full bit, BF of the SSPxSTAT register, indicates when SSPxBUF has been loaded with the received data (transmission is complete). When the SSPxBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSPx interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur.

FIGURE 24-5: SPI MASTER/SLAVE CONNECTION

![SPI Master/Slave Connection Diagram]

- SPI Master SSPxM<3:0> = 00xx = 1010
- SPI Slave SSPxM<3:0> = 010x
- Serial Input Buffer (BUF)
- Shift Register (SSPxSR)
- Serial Clock
- Slave Select (optional)
- Processor 1
- Processor 2
24.2.3 SPI MASTER MODE

The master can initiate the data transfer at any time because it controls the SCKx line. The master determines when the slave (Processor 2, Figure 24-5) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPxBUF register is written to. If the SPI is only going to receive, the SDOx output could be disabled (programmed as an input). The SSPxSR register will continue to shift in the signal present on the SDIx pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPxBUF register as if a normal received byte (interrupts and Status bits appropriately set).

The clock polarity is selected by appropriately programming the CKP bit of the SSPxCON1 register and the CKE bit of the SSPxSTAT register. This then, would give waveforms for SPI communication as shown in Figure 24-6, Figure 24-8 and Figure 24-9, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- Fosc/4 (or TCY)
- Fosc/16 (or 4 * TCY)
- Fosc/64 (or 16 * TCY)
- Timer2 output/2
- Fosc/(4 * (SSPxADD + 1))

Figure 24-6 shows the waveforms for Master mode. When the CKE bit is set, the SDOx data is valid before there is a clock edge on SCKx. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPxBUF is loaded with the received data is shown.

**FIGURE 24-6: SPI MODE WAVEFORM (MASTER MODE)**
24.2.4 SPI SLAVE MODE

In Slave mode, the data is transmitted and received as external clock pulses appear on SCKx. When the last bit is latched, the SSPxIF interrupt flag bit is set.

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCKx pin. The Idle state is determined by the CKP bit of the SSPxCON1 register.

While in Slave mode, the external clock is supplied by the external clock source on the SCKx pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. The shift register is clocked from the SCKx pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

24.2.4.1 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 24-7 shows the block diagram of a typical Daisy-Chain connection when operating in SPI Mode.

In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the BOEN bit of the SSPxCON3 register will enable writes to the SSPxBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

24.2.5 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 SSx pin allows a Synchronous Slave mode. The SPI must be in Slave mode with SSx pin control enabled (SSPxCON1<3:0> = 0100).

When the SSx pin is low, transmission and reception are enabled and the SDOx pin is driven.

When the SSx pin goes high, the SDOx pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

| Note 1: | When the SPI is in Slave mode with SSx pin control enabled (SSPxCON1<3:0> = 0100), the SPI module will reset if the SSx pin is set to VDD. |
| Note 2: | When the SPI is used in Slave mode with CKE set; the user must enable SSx pin control. |
| Note 3: | While operated in SPI Slave mode the SMP bit of the SSPxSTAT register must remain clear. |

When the SPI module resets, the bit counter is forced to ‘0’. This can be done by either forcing the SSx pin to a high level or clearing the SSPxEN bit.
FIGURE 24-7: SPI DAISY-CHAIN CONNECTION

FIGURE 24-8: SLAVE SELECT SYNCHRONOUS WAVEFORM
FIGURE 24-9: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

FIGURE 24-10: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)
24.2.6 SPI OPERATION IN SLEEP MODE

In SPI Master mode, module clocks may be operating at a different speed than when in Full-Power mode; in the case of the Sleep mode, all clocks are halted.

Special care must be taken by the user when the MSSPx clock is much faster than the system clock.

In Slave mode, when MSSPx interrupts are enabled, after the master completes sending data, an MSSPx interrupt will wake the controller from Sleep.

If an exit from Sleep mode is not desired, MSSPx interrupts should be disabled.

In SPI Master mode, when the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all 8 bits have been received, the MSSPx interrupt flag bit will be set and if enabled, will wake the device.

**TABLE 24-1: SUMMARY OF REGISTERS ASSOCIATED WITH SPI 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>Register on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>APFCON0</td>
<td>RXDTSEL</td>
<td>SDO1SEL</td>
<td>SS1SEL</td>
<td>P2BSEL</td>
<td>CCP2SEL</td>
<td>P1DSEL</td>
<td>P1CSEL</td>
<td>CCP1SEL</td>
<td>118</td>
</tr>
<tr>
<td>ANSELA</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ANSA4</td>
<td>ANSA3</td>
<td>ANSA2</td>
<td>ANSA1</td>
<td>ANSA0</td>
<td>122</td>
</tr>
<tr>
<td>ANSELB</td>
<td>ANSB7</td>
<td>ANSB6</td>
<td>ANSB5</td>
<td>ANSB4</td>
<td>ANSB3</td>
<td>ANSB2</td>
<td>ANSB1</td>
<td>—</td>
<td>128</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>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>MSSPxBUF</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>237*</td>
</tr>
<tr>
<td>MSSPxCON1</td>
<td>WCOL</td>
<td>SSPxOV</td>
<td>SSPxEN</td>
<td>CKP</td>
<td>SSPxM3</td>
<td>SSPxM2</td>
<td>SSPxM1</td>
<td>SSPxM0</td>
<td>282</td>
</tr>
<tr>
<td>MSSPxCON3</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>284</td>
</tr>
<tr>
<td>MSSPxSTAT</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>281</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRIS7</td>
<td>TRIS6</td>
<td>TRIS5</td>
<td>TRIS4</td>
<td>TRIS3</td>
<td>TRIS2</td>
<td>TRIS1</td>
<td>TRIS0</td>
<td>120</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRIS7</td>
<td>TRIS6</td>
<td>TRIS5</td>
<td>TRIS4</td>
<td>TRIS3</td>
<td>TRIS2</td>
<td>TRIS1</td>
<td>TRIS0</td>
<td>126</td>
</tr>
</tbody>
</table>

**Legend:** Shaded cells are not used by the MSSPx in SPI mode.

*Page provides register information.

**Note 1:** PIC16F/LF1827 only.
24.3 I²C MODE OVERVIEW

The Inter-Integrated Circuit Bus (I²C) is a multi-master serial data communication bus. Devices communicate in a master/slave environment where the master devices initiate the communication. A slave device is controlled through addressing.

The I²C bus specifies two signal connections:

• Serial Clock (SCLx)
• Serial Data (SDAx)

Figure 24-11 shows the block diagram of the MSSPx module when operating in I²C mode.

Both the SCLx and SDAx connections are bidirectional open-drain lines, each requiring pull-up resistors for the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one.

Figure 24-11 shows a typical connection between two processors configured as master and slave devices.

The I²C bus can operate with one or more master devices and one or more slave devices.

There are four potential modes of operation for a given device:

• Master Transmit mode
  (master is transmitting data to a slave)
• Master Receive mode
  (master is receiving data from a slave)
• Slave Transmit mode
  (slave is transmitting data to a master)
• Slave Receive mode
  (slave is receiving data from the master)

To begin communication, a master device starts out in Master Transmit mode. The master device sends out a Start bit followed by the address byte of the slave it intends to communicate with. This is followed by a single Read/Write bit, which determines whether the master intends to transmit to or receive data from the slave device.

If the requested slave exists on the bus, it will respond with an Acknowledge bit, otherwise known as an ACK. The master then continues in either Transmit mode or Receive mode and the slave continues in the complement, either in Receive mode or Transmit mode, respectively.

A Start bit is indicated by a high-to-low transition of the SDAx line while the SCLx line is held high. Address and data bytes are sent out, Most Significant bit (MSb) first. The Read/Write bit is sent out as a logical one when the master intends to read data from the slave, and is sent out as a logical zero when it intends to write data to the slave.

The Acknowledge bit (ACK) is an active-low signal, which holds the SDAx line low to indicate to the transmitter that the slave device has received the transmitted data and is ready to receive more.

The transition of a data bits is always performed while the SCLx line is held low. Transitions that occur while the SCLx line is held high are used to indicate Start and Stop bits.

If the master intends to write to the slave, then it repeatedly sends out a byte of data, with the slave responding after each byte with an ACK bit. In this example, the master device is in Master Transmit mode and the slave is in Slave Receive mode.

If the master intends to read from the slave, then it repeatedly receives a byte of data from the slave, and responds after each byte with an ACK bit. In this example, the master device is in Master Receive mode and the slave is Slave Transmit mode.

On the last byte of data communicated, the master device may end the transmission by sending a Stop bit. If the master device is in Receive mode, it sends the Stop bit in place of the last ACK bit. A Stop bit is indicated by a low-to-high transition of the SDAx line while the SCLx line is held high.

In some cases, the master may want to maintain control of the bus and re-initiate another transmission. If so, the master device may send another Start bit in place of the Stop bit or last ACK bit when it is in receive mode.

The I²C bus specifies three message protocols:

• Single message where a master writes data to a slave.
• Single message where a master reads data from a slave.
• Combined message where a master initiates a minimum of two writes, or two reads, or a combination of writes and reads, to one or more slaves.
When one device is transmitting a logical one, or letting the line float, and a second device is transmitting a logical zero, or holding the line low, the first device can detect that the line is not a logical one. This detection, when used on the SCLx line, is called clock stretching. Clock stretching give slave devices a mechanism to control the flow of data. When this detection is used on the SDAx line, it is called arbitration. Arbitration ensures that there is only one master device communicating at any single time.

24.3.1 CLOCK STRETCHING

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCLx clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master that is communicating with the slave will attempt to raise the SCLx line in order to transfer the next bit, but will detect that the clock line has not yet been released. Because the SCLx connection is open-drain, the slave has the ability to hold that line low until it is ready to continue communicating.

Clock stretching allow receivers that cannot keep up with a transmitter to control the flow of incoming data.

24.3.2 ARBITRATION

Each master device must monitor the bus for Start and Stop bits. If the device detects that the bus is busy, it cannot begin a new message until the bus returns to an Idle state.

However, two master devices may try to initiate a transmission on or about the same time. When this occurs, the process of arbitration begins. Each transmitter checks the level of the SDAx data line and compares it to the level that it expects to find. The first transmitter to observe that the two levels don’t match, loses arbitration, and must stop transmitting on the SDAx line.

For example, if one transmitter holds the SDAx line to a logical one (lets it float) and a second transmitter holds it to a logical zero (pulls it low), the result is that the SDAx line will be low. The first transmitter then observes that the level of the line is different than expected and concludes that another transmitter is communicating.

The first transmitter to notice this difference is the one that loses arbitration and must stop driving the SDAx line. If this transmitter is also a master device, it also must stop driving the SCLx line. It then can monitor the lines for a Stop condition before trying to reissue its transmission. In the meantime, the other device that has not noticed any difference between the expected and actual levels on the SDAx line continues with its original transmission. It can do so without any complications, because so far, the transmission appears exactly as expected with no other transmitter disturbing the message.

Slave Transmit mode can also be arbitrated, when a master addresses multiple slaves, but this is less common.

If two master devices are sending a message to two different slave devices at the address stage, the master sending the lower slave address always wins arbitration. When two master devices send messages to the same slave address, and addresses can sometimes refer to multiple slaves, the arbitration process must continue into the data stage.

Arbitration usually occurs very rarely, but it is a necessary process for proper multi-master support.
24.4 I²C MODE OPERATION

All MSSPx I²C communication is byte oriented and shifted out MSb first. Six SFR registers and 2 interrupt flags interface the module with the PIC® microcontroller and user software. Two pins, SDAx and SCLx, are exercised by the module to communicate with other external I²C devices.

24.4.1 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 SCLx line, the device outputting data on the SDAx changes that pin to an input and reads in an acknowledge value on the next clock pulse.

The clock signal, SCLx, is provided by the master. Data is valid to change while the SCLx signal is low, and sampled on the rising edge of the clock. Changes on the SDAx line while the SCLx line is high define special conditions on the bus, explained below.

24.4.2 DEFINITION OF I²C TERMINOLOGY

There is language and terminology in the description of I²C communication that have definitions specific to I²C. That word usage is defined below and may be used in the rest of this document without explanation. This table was adapted from the Phillips I²C specification.

24.4.3 SDAx AND SCLx PINS

Selection of any I²C mode with the SSPxEN bit set, forces the SCLx and SDAx pins to be open-drain. These pins should be set by the user to inputs by setting the appropriate TRIS bits.

**Note:** Data is tied to output zero when an I²C mode is enabled.

24.4.4 SDAx HOLD TIME

The hold time of the SDAx pin is selected by the SDAHT bit of the SSPxCON3 register. Hold time is the time SDAx is held valid after the falling edge of SCLx. Setting the SDAHT bit selects a longer 300 ns minimum hold time and may help on buses with large capacitance.

**TABLE 24-2: I²C™ BUS TERMS**

<table>
<thead>
<tr>
<th>TERM</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter</td>
<td>The device which shifts data out onto the bus.</td>
</tr>
<tr>
<td>Receiver</td>
<td>The device which shifts data in from the bus.</td>
</tr>
<tr>
<td>Master</td>
<td>The device that initiates a transfer, generates clock signals and terminates a transfer.</td>
</tr>
<tr>
<td>Slave</td>
<td>The device addressed by the master.</td>
</tr>
<tr>
<td>Multi-master</td>
<td>A bus with more than one device that can initiate data transfers.</td>
</tr>
<tr>
<td>Arbitration</td>
<td>Procedure to ensure that only one master at a time controls the bus. Winning arbitration ensures that the message is not corrupted.</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Procedure to synchronize the clocks of two or more devices on the bus.</td>
</tr>
<tr>
<td>Idle</td>
<td>No master is controlling the bus, and both SDAx and SCLx lines are high.</td>
</tr>
<tr>
<td>Active</td>
<td>Any time one or more master devices are controlling the bus.</td>
</tr>
<tr>
<td>Addressed Slave</td>
<td>Slave device that has received a matching address and is actively being clocked by a master.</td>
</tr>
<tr>
<td>Matching Address</td>
<td>Address byte that is clocked into a slave that matches the value stored in SSPxADD.</td>
</tr>
<tr>
<td>Write Request</td>
<td>Slave receives a matching address with R/W bit clear, and is ready to clock in data.</td>
</tr>
<tr>
<td>Read Request</td>
<td>Master sends an address byte with the R/W bit set, indicating that it wishes to clock data out of the Slave. This data is the next and all following bytes until a Restart or Stop.</td>
</tr>
<tr>
<td>Clock Stretching</td>
<td>When a device on the bus hold SCLx low to stall communication.</td>
</tr>
<tr>
<td>Bus Collision</td>
<td>Any time the SDAx line is sampled low by the module while it is outputting and expected high state.</td>
</tr>
</tbody>
</table>
24.4.5 START CONDITION

The I²C specification defines a Start condition as a transition of SDAx from a high to a low state while SCLx line is high. A Start condition is always generated by the master and signifies the transition of the bus from an Idle to an active state. Figure 24-10 shows wave forms for Start and Stop conditions.

A bus collision can occur on a Start condition if the module samples the SDAx line low before asserting it low. This does not conform to the I²C specification that states no bus collision can occur on a Start.

24.4.6 STOP CONDITION

A Stop condition is a transition of the SDAx line from low to high state while the SCLx line is high.

Note: At least one SCLx low time must appear before a Stop is valid, therefore, if the SDAx line goes low then high again while the SCLx line stays high, only the Start condition is detected.

24.4.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave.

In 10-bit Addressing Slave mode a Restart is required for the master to clock data out of the addressed slave. Once a slave has been fully addressed, matching both high and low address bytes, the master can issue a Restart and the high address byte with the R/W bit set. The slave logic will then hold the clock and prepare to clock out data.

After a full match with R/W clear in 10-bit mode, a prior match flag is set and maintained. Until a Stop condition, a high address with R/W clear, or high address match fails.

24.4.8 START/STOP CONDITION INTERRUPT MASKING

The SCIE and PCIE bits of the SSPxCON3 register can enable the generation of an interrupt in Slave modes that do not typically support this function. Slave modes where interrupt on Start and Stop detect are already enabled, these bits will have no effect.
24.4.9 ACKNOWLEDGE SEQUENCE

The 9th SCLx pulse for any transferred byte in \(^2\text{C}\) is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDAx line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge (\(\overline{\text{ACK}}\)) is an active-low signal, pulling the SDAx line low indicated to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an \(\overline{\text{ACK}}\) is placed in the ACKSTAT bit of the SSPxCON2 register.

Slave software, when the AHEN and DHEN bits are set, allow the user to set the \(\overline{\text{ACK}}\) value sent back to the transmitter. The ACKDT bit of the SSPxCON2 register is set/cleared to determine the response.

Slave hardware will generate an \(\overline{\text{ACK}}\) response if the AHEN and DHEN bits of the SSPxCON3 register are clear.

There are certain conditions where an \(\overline{\text{ACK}}\) will not be sent by the slave. If the BF bit of the SSPxSTAT register or the SSPxOV bit of the SSPxCON1 register are set when a byte is received.

When the module is addressed, after the 8th falling edge of SCLx on the bus, the ACKTIM bit of the SSPxCON3 register is set. The ACKTIM bit indicates the acknowledge time of the active bus.

24.5 \(^2\text{C}\) SLAVE MODE OPERATION

The MSSPx Slave mode operates in one of four modes selected in the SSPxM bits of SSPxCON1 register. The modes can be divided into 7-bit and 10-bit Addressing mode. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

Modes with Start and Stop bit interrupts operated the same as the other modes with SSPxIF additionally getting set upon detection of a Start, Restart, or Stop condition.

24.5.1 SLAVE MODE ADDRESSES

The SSPxADD register (Register 24-6) contains the Slave mode address. The first byte received after a Start or Restart condition is compared against the value stored in this register. If the byte matches, the value is loaded into the SSPxBUF register and an interrupt is generated. If the value does not match, the module goes idle and no indication is given to the software that anything happened.

The SSPx Mask register (Register 24-5) affects the address matching process. See Section 24.5.9 “SSPx Mask Register” for more information.

24.5.1.1 \(^2\text{C}\) 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.

24.5.1.2 \(^2\text{C}\) Slave 10-bit Addressing Mode

In 10-bit Addressing mode, the first received byte is compared to the binary value of ‘1 1 1 1 0 A9 A8 0’. A9 and A8 are the two MSb of the 10-bit address and stored in bits 2 and 1 of the SSPxADD register.

After the acknowledge of the high byte the UA bit is set and SCLx is held low until the user updates SSPxADD with the low address. The low address byte is clocked in and all 8 bits are compared to the low address value in SSPxADD. Even if there is not an address match; SSPxIF and UA are set, and SCLx is held low until SSPxADD is updated to receive a high byte again. When SSPxADD is updated the UA bit is cleared. This ensures the module is ready to receive the high address byte on the next communication.

A high and low address match as a write request is required at the start of all 10-bit addressing communication. A transmission can be initiated by issuing a Restart once the slave is addressed, and clocking in the high address with the R/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.
24.5.2 SLAVE RECEPTION

When the R/W bit of a matching received address byte is clear, the R/W bit of the SSPxSTAT register is cleared. The received address is loaded into the SSPxBUF register and acknowledged.

When the overflow condition exists for a received address, then not Acknowledge is given. An overflow condition is defined as either bit BF bit of the SSPxSTAT register is set, or bit SSPxOV bit of the SSPxCON1 register is set. The BOEN bit of the SSPxCON3 register modifies this operation. For more information see Register 24-4.

An MSSPx interrupt is generated for each transferred data byte. Flag bit, SSPxIF, must be cleared by software.

When the SEN bit of the SSPxCON2 register is set, SCLx will be held low (clock stretch) following each received byte. The clock must be released by setting the CKP bit of the SSPxCON1 register, except sometimes in 10-bit mode. See Section 24.2.3 “SPI Master Mode” for more detail.

24.5.2.1 7-bit Addressing Reception

This section describes a standard sequence of events for the MSSPx module configured as an I^2C slave in 7-bit Addressing mode. All decisions made by hardware or software and their effect on reception. Figure 24-13 and Figure 24-14 is used as a visual reference for this description.

This is a step by step process of what typically must be done to accomplish I^2C communication.
1. Start bit detected.
2. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with R/W bit clear is received.
4. The slave pulls SDAx low sending an ACK to the master, and sets SSPxIF bit.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPx-BUF clearing the BF flag.
7. If SEN = 1; Slave software sets CKP bit to release the SCLx line.
8. The master clocks out a data byte.
9. Slave drives SDAx low sending an ACK to the master, and sets SSPxIF bit.
10. Software clears SSPxIF.
11. Software reads the received byte from SSPx-BUF clearing BF.
12. Steps 8-12 are repeated for all received bytes from the master.

24.5.2.2 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 SCLx. 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 I^2C communication. Figure 24-15 displays a module using both address and data holding. Figure 24-16 includes the operation with the SEN bit of the SSPxCON2 register set.

1. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
2. Matching address with R/W bit clear is clocked in. SSPxIF is set and CKP cleared after the 8th falling edge of SCLx.
3. Slave clears the SSPxIF.
4. Slave can look at the ACKTIM bit of the SSPxCON3 register to determine if the SSPxIF was after or before the ACK.
5. Slave reads the address value from SSPxBUF, clearing the BF flag.
6. Slave sets ACK value clocked out to the master by setting ACKDT.
7. Slave releases the clock by setting CKP.
8. SSPxIF 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 SSPxIF.

Note: SSPxIF is still set after the 9th falling edge of SCLx even if there is no clock stretching and BF has been cleared. Only if NACK is sent to master is SSPxIF not set.

11. SSPxIF set and CKP cleared after 8th falling edge of SCLx for a received data byte.
12. Slave looks at ACKTIM bit of SSPxCON3 to determine the source of the interrupt.
13. Slave reads the received data from SSPxBUF clearing BF.
14. Steps 7-14 are the same for each received data byte.
15. Communication is ended by either the slave sending an ACK = 1, or the master sending a Stop condition. If a Stop is sent and Interrupt on Stop Detect is disabled, the slave will only know by polling the P bit of the SSTSTAT register.
FIGURE 24-14:  

\( \text{I}^2\text{C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 0, DHEN = 0)} \)
FIGURE 24-15: I2C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)
FIGURE 24-16: I2C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)

Master sends Stop condition

- SDAx
- SCLx

Data is read from SSPxBUF

- SDAx
- SCLx

SSPxIF is set on 9th falling edge of SCLx, after ACK

Slave software sets ACKDT to not ACK

When DHEN = 1:
- CKP is cleared by hardware on 8th falling edge of SCLx

Slave software clears ACKDT to ACK

the received byte

ACKTIM set by hardware on 8th falling edge of SCLx

ACK

Master Releases SDAx to slave for ACK sequence

No interrupt after not ACK from Slave

ACKDT

ACKTIM

SSPxIF

If AHEN = 1:
- SSPxF is set
- Address is read from SSBUF
- Slave software clears ACKDT to not ACK
- ACKTIM is set by hardware in 9th rising edge of SCLx

ACK

When AHEN = 1:
- CKP is cleared by hardware on 8th falling edge of SCLx

ACKTIM set by hardware on 8th falling edge of SCLx

ACK

ACKTIM set by hardware on 8th falling edge of SCLx

ACK
FIGURE 24-17: I 2C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 1, DHEN = 1)

- Receiving Address
- Receive Data
- ACK
- Receive Data
- ACK
- ACK
- ACK

- SDAx
- SCLx
- SSPxF
- BF
- ACKDT
- CKP
- ACKTIM
- S
- P

- Master releases SDAx to slave for ACK sequence
- ACKDT is set by hardware on 8th falling edge of SCLx
- ACKTIM is cleared by hardware on 9th rising edge of SCLx
- Master sends Stop condition
- SF

- Received address is loaded into SSPxFBUF
- Received data is available on SSPxFBUF
- SSPxFBUF can be read any time before next byte is loaded
- No interrupt after if not ACK from Slave

- Slave software clears ACKDT to ACK the received byte
- ACKTIM is cleared by hardware on 9th rising edge of SCLx

- When AHEN = 1; on the 8th falling edge of SCLx of an address byte, CKP is cleared
- When DHEN = 1; on the 8th falling edge of SCLx of a received data byte, CKP is cleared

- Slave sends not ACK
- CKP is not cleared if not ACK

- Master releases SDAx to slave for ACK sequence
- ACKDT is set by hardware on 8th falling edge of SCLx
- ACKTIM is cleared by hardware on 9th rising edge of SCLx

- SF
24.5.3 SLAVE TRANSMISSION

When the R/W bit of the incoming address byte is set and an address match occurs, the R/W bit of the SSPxSTAT register is set. The received address is loaded into the SSPxBUF register, and an ACK pulse is sent by the slave on the ninth bit.

Following the ACK, slave hardware clears the CKP bit and the SCLx pin is held low (see Section 24.5.6 “Clock Stretching” for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data.

The transmit data must be loaded into the SSPxBUF register which also loads the SSPxSR register. Then the SCLx pin should be released by setting the CKP bit of the SSPxCON1 register. The eight data bits are shifted out on the falling edge of the SCLx input. This ensures that the SDAx signal is valid during the SCLx high time.

The ACK pulse from the master-receiver is latched on the rising edge of the ninth SCLx input pulse. This ACK value is copied to the ACKSTAT bit of the SSPxCON2 register. If ACKSTAT is set (not ACK), then the data transfer is complete. In this case, when the not ACK is latched by the slave, the slave goes idle and waits for another occurrence of the Start bit. If the SDAx line was low (ACK), the next transmit data must be loaded into the SSPxBUF register. Again, the SCLx pin must be released by setting bit CKP.

An MSSPx interrupt is generated for each data transfer byte. The SSPxIF bit must be cleared by software and the SSPxSTAT register is used to determine the status of the byte. The SSPxIF bit is set on the falling edge of the ninth clock pulse.

24.5.3.1 Slave Mode Bus Collision

A slave receives a Read request and begins shifting data out on the SDAx line. If a bus collision is detected and the SBCDE bit of the SSPxCON3 register is set, the BCLxIF bit of the PIRx register is set. Once a bus collision is detected, the slave goes idle and waits to be addressed again. User software can use the BCLxIF bit to handle a slave bus collision.

24.5.3.2 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 24-17 can be used as a reference to this list.

1. Master sends a Start condition on SDAx and SCLx.
2. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with R/W bit set is received by the slave setting SSPxIF bit.
4. Slave hardware generates an ACK and sets SSPxIF.
5. SSPxIF bit is cleared by user.
6. Software reads the received address from SSPxBUF, clearing BF.
7. R/W is set so CKP was automatically cleared after the ACK.
8. The slave software loads the transmit data into SSPxBUF.
9. CKP bit is set releasing SCLx, allowing the master to clock the data out of the slave.
10. SSPxIF is set after the ACK response from the master is loaded into the ACKSTAT register.
11. SSPxIF bit is cleared.
12. The slave software checks the ACKSTAT bit to see if the master wants to clock out more data.
13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not ACK; the clock is not held, but SSPxIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.

Note 1: If the master ACKs the clock will be stretched.
2: ACKSTAT is the only bit updated on the rising edge of SCLx (9th) rather than the falling.
FIGURE 24-18: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 0)

Master sends
Stop condition

received address
is read from SSPxBUF

Data to transmit is loaded into SSPxBUF

BF is automatically cleared after 8th falling edge of SCLx

ACK STAT

ACK

ACK

ACK

R/W

SDAx

SCLx

SSPx IF

BF

CKP

ACK

R/W = 1

Master sends Stop condition

Set by software

Cleared by software

Indicates an address has been received

Masters not ACK

is copied to

ACKSTAT

CKP is not held for not ACK

BF is automatically cleared after 8th falling edge of SCLx

SDAx

SCLx

SSPx IF

BF

CKP

ACK STAT

R/W

D/A

S

P
24.5.3.3 7-bit Transmission with Address Hold Enabled

Setting the AHEN bit of the SSPxCON3 register enables additional clock stretching and interrupt generation after the 8th falling edge of a received matching address. Once a matching address has been clocked in, CKP is cleared and the SSPxIF interrupt is set.

Figure 24-18 displays a standard waveform of a 7-bit Address Slave Transmission with AHEN enabled.

1. Bus starts Idle.
2. Master sends Start condition; the S bit of SSPxSTAT is set; SSPxIF 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 SCLx line the CKP bit is cleared and SSPxIF interrupt is generated.
4. Slave software clears SSPxIF.
5. Slave software reads ACKTIM bit of SSPxCON3 register, and R/W and D/A of the SSPxSTAT register to determine the source of the interrupt.
6. Slave reads the address value from the SSPxBUF register clearing the BF bit.
7. Slave software decides from this information if it wishes to ACK or not ACK and sets ACKDT bit of the SSPxCON2 register accordingly.
8. Slave sets the CKP bit releasing SCLx.
9. Master clocks in the ACK value from the slave.
10. Slave hardware automatically clears the CKP bit and sets SSPxIF after the ACK if the R/W bit is set.
11. Slave software clears SSPxIF.
12. Slave loads value to transmit to the master into SSPxBUF setting the BF bit.

**Note:** SSPxBUF cannot be loaded until after the 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 SCLx pulse.
15. Slave hardware copies the ACK value into the ACKSTAT bit of the SSPxCON2 register.
16. Steps 10-15 are repeated for each byte transmitted to the master from the slave.
17. If the master sends a not ACK the slave releases the bus allowing the master to send a Stop and end the communication.

**Note:** Master must send a not ACK on the last byte to ensure that the slave releases the SCLx line to receive a Stop.
FIGURE 24-19: I2C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)

- Master sends Start condition.
- Master releases SDAx to slave for ACK sequence.
- Data to transmit is loaded into SSPxBUF.
- Slave clears address.
- ACKDT is set on 8th falling edge of SCLx.
- ACKTIM is set on 8th falling edge of SCLx.
- ACKTIM is cleared on 9th rising edge of SCLx.
- ACKSTAT is cleared after 8th falling edge of SCLx.
- BF is automatically cleared after 8th falling edge of SCLx.
- Master's ACK response is copied to SSPxSTAT.
- CKP is cleared by hardware after receiving matching address.
- When R/W = 1, CKP is always cleared after ACK.
- When AHEN = 1, CKP is cleared by hardware after receiving matching address.
- Received address is read from SSPxBUF.
- Slave clears address.
- Master's ACK response is cleared to ACK.
- Set by software, releases SCLx.
- ACKTIM is cleared on 9th rising edge of SCLx.
- ACKTIM is cleared on 8th falling edge of SCLx.
- ACKSTAT is cleared after not ACK.
- ACKDT is automatically cleared after 8th falling edge of SCLx.

Master sends Stop condition.
24.5.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSPx module configured as an I2C slave in 10-bit Addressing mode.

Figure 24-19 and is used as a visual reference for this description.

This is a step by step process of what must be done by slave software to accomplish I2C communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Master sends matching high address with R/W bit clear; UA bit of the SSPxSTAT register is set.
4. Slave sends ACK and SSPxIF is set.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. Slave loads low address into SSPxADD, releasing SCLx.
8. Master sends matching low address byte to the slave; UA bit is set.

**Note:** Updates to the SSPxADD register are not allowed until after the ACK sequence.

9. Slave sends ACK and SSPxIF is set.

**Note:** If the low address does not match, SSPxIF and UA are still set so that the slave software can set SSPxADD back to the high address. BF is not set because there is no match. CKP is unaffected.

10. Slave clears SSPxIF.
11. Slave reads the received matching address from SSPxBUF clearing BF.
12. Slave loads high address into SSPxADD.
13. Master clocks a data byte to the slave and clocks out the slaves ACK on the 9th SCLx pulse; SSPxIF is set.
14. If SEN bit of SSPxCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPxIF.
16. Slave reads the received byte from SSPxBUF clearing BF.
17. If SEN is set the slave sets CKP to release the SCLx.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

24.5.5 10-BIT ADDRESSING WITH ADDRESS OR DATA HOLD

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPxADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCLx line is held low are the same. Figure 24-20 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 24-21 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.
FIGURE 24-20: I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)
FIGURE 24-21: \( \text{i}^2\text{C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 0)} \)
FIGURE 24-22: \( i^2 C \) SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)
24.5.6 CLOCK STRETCHING

Clock stretching occurs when a device on the bus holds the SCLx line low effectively pausing communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. A master device is not concerned with stretching as anytime it is active on the bus and not transferring data it is stretching. Any stretching done by a slave is invisible to the master software and handled by the hardware that generates SCLx.

The CKP bit of the SSPxCON1 register is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCLx line to go low and then hold it. Setting CKP will release SCLx and allow more communication.

24.5.6.1 Normal Clock Stretching

Following an ACK if the R/W bit of SSPxSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPxBUF with data to transfer to the master. If the SEN bit of SSPxCON2 is set, the slave hardware will always stretch the clock after the ACK sequence. Once the slave is ready; CKP is set by software and communication resumes.

Note 1: The BF bit has no effect on whether the clock will be stretched or not. This is different than previous versions of the module that would not stretch the clock, clear CKP, if SSPxBUF was read before the 9th falling edge of SCLx.

2: Previous versions of the module did not stretch the clock for a transmission if SSPxBUF was loaded before the 9th falling edge of SCLx. It is now always cleared for read requests.

24.5.6.2 10-bit Addressing Mode

In 10-bit Addressing mode, when the UA bit is set, the clock is always stretched. This is the only time the SCLx is stretched without CKP being cleared. SCLx is released immediately after a write to SSPxADD.

Note: Previous versions of the module did not stretch the clock if the second address byte did not match.

24.5.6.3 Byte NACKing

When AHEN bit of SSPxCON3 is set; CKP is cleared by hardware after the 8th falling edge of SCLx for a received matching address byte. When DHEN bit of SSPxCON3 is set; CKP is cleared after the 8th falling edge of SCLx for received data.

Stretching after the 8th falling edge of SCLx allows the slave to look at the received address or data and decide if it wants to ACK the received data.

24.5.7 CLOCK SYNCHRONIZATION AND THE CKP BIT

Any time the CKP bit is cleared, the module will wait for the SCLx line to go low and then hold it. However, clearing the CKP bit will not assert the SCLx output low until the SCLx output is already sampled low. Therefore, the CKP bit will not assert the SCLx line until an external \( \text{I}^2\text{C} \) master device has already asserted the SCLx line. The SCLx output will remain low until the CKP bit is set and all other devices on the \( \text{I}^2\text{C} \) bus have released SCLx. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCLx (see Figure 24-22).

FIGURE 24-23: CLOCK SYNCHRONIZATION TIMING

| Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

SDAx

SCLx

CKP

WR

SSPxCON

Master device asserts clock

Master device releases clock

\( DX \)
24.5.8 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the \( \text{i}^2\text{C} \) 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 \( \text{i}^2\text{C} \) protocol, defined as address 0x00. When the GCEN bit of the SSPxCON2 register is set, the slave module will automatically ACK the reception of this address regardless of the value stored in SSPxADD. After the slave clocks in an address of all zeros with the R/W bit clear, an interrupt is generated and slave software can read SSPxBUF and respond. Figure 24-23 shows a general call reception sequence.

In 10-bit Address mode, the UA bit will not be set on the reception of the general call address. The slave will prepare to receive the second byte as data, just as it would in 7-bit mode.

If the AHEN bit of the SSPxCON3 register is set, just as with any other address reception, the slave hardware will stretch the clock after the 8th falling edge of SCLx. The slave must then set its ACKDT value and release the clock with communication progressing as it would normally.

FIGURE 24-24: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE

24.5.9 SSPX MASK REGISTER

An SSPx Mask (SSPxMSK) register (Register 24-5) is available in \( \text{i}^2\text{C} \) Slave mode as a mask for the value held in the SSPxSR register during an address comparison operation. A zero ('0') bit in the SSPxMSK register has the effect of making the corresponding bit of the received address a "don't care".

This register is reset to all '1's upon any Reset condition and, therefore, has no effect on standard SSPx operation until written with a mask value.

The SSPx Mask register is active during:
- 7-bit Address mode: address compare of A<7:1>.
- 10-bit Address mode: address compare of A<7:0> only. The SSPx mask has no effect during the reception of the first (high) byte of the address.
24.6  \(^2\)C MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPxM bits in the SSPxCON1 register and by setting the SSPxEN bit. In Master mode, the SCLx and SDAx lines are set as inputs and are manipulated by the MSSPx hardware.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSPx module is disabled. Control of the \(^2\)C bus may be taken when the P bit is set, or the bus is idle.

In Firmware Controlled Master mode, user code conducts all \(^2\)C bus operations based on Start and Stop bit condition detection. Start and Stop condition detection is the only active circuitry in this mode. All other communication is done by the user software directly manipulating the SDAx and SCLx lines.

The following events will cause the SSPx Interrupt Flag bit, SSPxIF, to be set (SSPx interrupt, if enabled):

- Start condition detected
- Stop condition detected
- Data transfer byte transmitted/received
- Acknowledge transmitted/received
- Repeated Start generated

Note 1: The MSSPx module, when configured in \(^2\)C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur.

Note 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.

24.6.1 \(^2\)C 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 \(^2\)C bus will not be released.

In Master Transmitter mode, serial data is output through SDAx, while SCLx outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/W) bit.

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 ‘0’. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/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 SDAx, while SCLx outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

A Baud Rate Generator is used to set the clock frequency output on SCLx. See Section 24.7 “Baud Rate Generator” for more detail.
24.6.2 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, releases the SCLx pin (SCLx allowed to float high). When the SCLx pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCLx pin is actually sampled high. When the SCLx pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and begins counting. This ensures that the SCLx high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 24-25).

FIGURE 24-25: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION

24.6.3 WCOL STATUS FLAG

If the user writes the SSPxBUF when a Start, Restart, Stop, Receive or Transmit sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write does not occur). Any time the WCOL bit is set it indicates that an action on SSPxBUF was attempted while the module was not idle.

Note: Because queueing of events is not allowed, writing to the lower 5 bits of SSPxCON2 is disabled until the Start condition is complete.
24.6.4 \( \text{i}^2\text{C} \) MASTER MODE START
CONDITION TIMING

To initiate a Start condition, the user sets the Start Enable bit, SEN bit of the SSPxCON2 register. If the SDAx and SCLx pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and starts its count. If SCLx and SDAx are both sampled high when the Baud Rate Generator times out (TBRG), the SDAx pin is driven low. The action of the SDAx being driven low while SCLx is high is the Start condition and causes the S bit of the SSPxSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit of the SSPxCON2 register will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDAx line held low and the Start condition is complete.

**Note 1:** If at the beginning of the Start condition, the SDAx and SCLx pins are already sampled low, or if during the Start condition, the SCLx line is sampled low before the SDAx line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLxIF, is set, the Start condition is aborted and the \( \text{i}^2\text{C} \) module is reset into its Idle state.

2: The Philips \( \text{i}^2\text{C} \) Specification states that a bus collision cannot occur on a Start.
24.6.5 I2C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit of the SSPxCON2 register is programmed high and the master state machine is no longer active. When the RSEN bit is set, the SCLx pin is asserted low. When the SCLx pin is sampled low, the Baud Rate Generator is loaded and begins counting. The SDAx pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDAx is sampled high, the SCLx pin will be deasserted (brought high). When SCLx is sampled high, the Baud Rate Generator is reloaded and begins counting. SDAx and SCLx must be sampled high for one TBRG. This action is then followed by assertion of the SDAx pin (SDAx = 0) for one TBRG while SCLx is high. SCLx is asserted low. Following this, the RSEN bit of the SSPxCON2 register will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDAx pin held low. As soon as a Start condition is detected on the SDAx and SCLx pins, the S bit of the SSPxSTAT register will be set. The SSPxIF bit will not be set until the Baud Rate Generator has timed out.

Note 1: If RSEN is programmed while any other event is in progress, it will not take effect.

2: A bus collision during the Repeated Start condition occurs if:
   - SDAx is sampled low when SCLx goes from low-to-high.
   - SCLx goes low before SDAx is asserted low. This may indicate that another master is attempting to transmit a data ‘1’. 

![FIGURE 24-27: REPEAT START CONDITION WAVEFORM](image)
24.6.6 I^2C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPxBUF register. This action will set the Buffer Full flag bit, BF and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDAx pin after the falling edge of SCLx is asserted. SCLx is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCLx is released high. When the SCLx pin is released high, it is held that way for TBRG. The data on the SDAx pin must remain stable for that duration and some hold time after the next falling edge of SCLx. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDAx. This allows the slave device to respond with an ACK bit during the ninth bit time if an address match occurred, or if data was received properly. The status of ACK is written into the ACKSTAT bit on the rising edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared. If not, the bit is set. After the ninth clock, the SSPxFIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPxBUF, leaving SCLx low and SDAx unchanged (Figure 24-27).

After the write to the SSPxBUF, each bit of the address will be shifted out on the falling edge of SCLx until all seven address bits and the R/W bit are completed. On the falling edge of the eighth clock, the master will release the SDAx pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDAx pin to see if the address was recognized by a slave. The status of the ACK bit is loaded into the ACKSTAT Status bit of the SSPxCON2 register. Following the falling edge of the ninth clock transmission of the address, the SSPxFIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPxBUF takes place, holding SCLx low and allowing SDAx to float.

24.6.6.1 BF Status Flag

In Transmit mode, the BF bit of the SSPxSTAT register is set when the CPU writes to SSPxBUF and is cleared when all 8 bits are shifted out.

24.6.6.2 WCOL Status Flag

If the user writes the SSPxBUF when a transmit is already in progress (i.e., SSPxSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write does not occur).

WCOL must be cleared by software before the next transmission.

24.6.6.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit of the SSPxCON2 register is cleared when the slave has sent an Acknowledge (ACK = 0) and is set when the slave does not Acknowledge (ACK = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

24.6.6.4 Typical Transmit Sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPxCON2 register.
2. SSPxFIF is set by hardware on completion of the Start.
3. SSPxFIF is cleared by software.
4. The MSSPx module will wait the required start time before any other operation takes place.
5. The user loads the SSPxBUF with the slave address to transmit.
6. Address is shifted out the SDAx pin until all 8 bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
7. The MSSPx module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
8. The MSSPx module generates an interrupt at the end of the ninth clock cycle by setting the SSPxFIF bit.
9. The user loads the SSPxBUF with eight bits of data.
10. Data is shifted out the SDAx pin until all 8 bits are transmitted.
11. The MSSPx module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
12. Steps 8-11 are repeated for all transmitted data bytes.
13. The user generates a Stop or Restart condition by setting the PEN or RSEN bits of the SSPxCON2 register. Interrupt is generated once the Stop/Restart condition is complete.
FIGURE 24-28: \( \text{I}^2\text{C} \text{ M} \text{A} \text{S} \text{T} \text{E} \text{R} \text{ M} \text{O} \text{D} \text{E} \text{ W} \text{A} \text{V} \text{E} \text{F} \text{O} \text{R} \text{M} \text{ (T} \text{R} \text{A} \text{N} \text{S} \text{M} \text{I} \text{S} \text{S} \text{I} \text{ON} \text{,} \text{ 7} \text{ O} \text{R} \text{ 1} \text{0}-\text{B} \text{I} \text{T} \text{ A} \text{D} \text{R} \text{E} \text{S} \text{S} \text{S} \text{S}) \)

1. Write SSPCON2<10> SEN = 1
2. If 10-bit address, write SSPCON2<6> = 1
3. Transmit Address to Slave
4. Start condition begins
5. Transmit Data or Second Half of 10-bit Address
6. CPU responds to SSPxIF
7. From slave, clear ACKSTAT bit SSPxCON2<6>
8. SSPBUF written with 7-bit address and R/W
9. Start condition begins
10. Transmit Address to Slave
11. Write SSPxCON2<0> SEN = 1
12. CPU responds to SSPxIF
13. Cleared by software
14. ACC cleared by software
15. SEN cleared by hardware
16. SSPBUF written
24.6.7  \textsuperscript{2}C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN bit of the SSPxCON2 register.

\textbf{Note:} The MSSPx module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCLx pin changes (high-to-low/low-to-high) and data is shifted into the SSPxSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPxSR are loaded into the SSPxBUF, the BF flag bit is set, the SSPxIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCLx low. The MSSPx is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable, ACKEN bit of the SSPxCON2 register.

24.6.7.1  BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPxBUF from SSPxSR. It is cleared when the SSPxBUF register is read.

24.6.7.2  SSPxOV Status Flag

In receive operation, the SSPxOV bit is set when 8 bits are received into the SSPxSR and the BF flag bit is already set from a previous reception.

24.6.7.3  WCOL Status Flag

If the user writes the SSPxBUF when a receive is already in progress (i.e., SSPxSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

24.6.7.4  Typical Receive Sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPxCON2 register.
2. SSPxIF is set by hardware on completion of the Start.
3. SSPxIF is cleared by software.
4. User writes SSPxBUF with the slave address to transmit and the \texttt{R/W} bit set.
5. Address is shifted out the SDAx pin until all 8 bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
6. The MSSPx module shifts in the \texttt{ACK} bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
7. The MSSPx module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
8. User sets the RCEN bit of the SSPxCON2 register and the Master clocks in a byte from the slave.
9. After the 8th falling edge of SCLx, SSPxIF and BF are set.
10. Master clears SSPxIF and reads the received byte from SSPxUF, clears BF.
11. Master sets \texttt{ACK} value sent to slave in ACKDT bit of the SSPxCON2 register and initiates the \texttt{ACK} by setting the ACKEN bit.
12. Masters \texttt{ACK} is clocked out to the slave and SSPxIF is set.
13. User clears SSPxIF.
14. Steps 8-13 are repeated for each received byte from the slave.
15. Master sends a not \texttt{ACK} or Stop to end communication.

\textbf{Note:} The MSSPx module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.
FIGURE 24-29: I²C™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)

- **SDAx**: Slave Data Input
- **SCLx**: Slave Clock
- **SSPxCON2**: SSP Control Register 2
- **SSPxBUF**: SSP Buffer Register
- **SSPxSTAT**: SSP Status Register
- **BF**: Busy Flag
- **SSPxOV**: SSP Overflow
- **ACKEN**: ACKnowledge Enable
- **RCEN**: Receiver Clear Enable

**Write to SSPxCON2<0>(SEN = 1), begin Start condition**

**Write to SSPxBUF occurs here**: Transmit Address to Slave

**ACK from Slave**: Receiving Data from Slave

**RCEN cleared automatically**

**Set SSPxOV if not already set**

**Set SSPxIF interrupt if not already set**

**Set P bit (SSPxSTAT<4>)**

**Last bit is shifted into SSPxSR and contents are unloaded into SSPxBUF**

**SSPxOV is set because SSPxBUF is bit full**

**SSPxIF is set at end of receive**

**Set ACKEN, start Acknowledge sequence**

**SDAx = ACKDT (SSPxCON2<5>) = 0**

**RCEN cleared automatically**

**Write to SSPxCON2<4> to start Acknowledge sequence**

**SDAx = ACKDT (SSPxCON2<5>) = 0**

**RCEN cleared automatically**

**ACK from Master**

**SSPxOV is reset because SSPxBUF is not full**

**SSPxIF is set at end of Acknowledge sequence**

**Set P bit (SSPxSTAT<4>) and SSPxIF**

**Bus master terminates transfer**

**Master configured as a receiver by programming SSPxCON2<3> (RCEN = 1)**

**RCEN cleared automatically**

**ACK from Master**

**SSPxOV is reset because SSPxBUF is not full**

**SSPxIF is set at end of Acknowledge sequence**

**Set P bit (SSPxSTAT<4>) and SSPxIF**

**PEN bit = 1 written here**

**Write to SSPxCON2<3> (RCEN = 1)**

**Write to SSPxBUF occurs here**, ACK from Slave

**RCEN cleared automatically**

**Set SSPxIF at end of receive**

**Set SSPxIF interrupt at end of receive**

**Set SSPxIF interrupt at end of Acknowledge sequence**

**PEN bit = 1 written here**

**Write to SSPxCON2<3> (RCEN = 1)**

**Write to SSPxBUF occurs here**, ACK from Slave

**RCEN cleared automatically**

**Set SSPxIF at end of receive**

**Set SSPxIF interrupt at end of Acknowledge sequence**

**Set SSPxIF interrupt at end of Acknowledge sequence**

**PEN bit = 1 written here**

**Write to SSPxCON2<3> (RCEN = 1)**

**Write to SSPxBUF occurs here**, ACK from Slave

**RCEN cleared automatically**

**Set SSPxIF at end of receive**

**Set SSPxIF interrupt at end of Acknowledge sequence**

**Set SSPxIF interrupt at end of Acknowledge sequence**

**PEN bit = 1 written here**

**Write to SSPxCON2<3> (RCEN = 1)**

**Write to SSPxBUF occurs here**, ACK from Slave

**RCEN cleared automatically**

**Set SSPxIF at end of receive**

**Set SSPxIF interrupt at end of Acknowledge sequence**

**Set SSPxIF interrupt at end of Acknowledge sequence**

**PEN bit = 1 written here**
24.6.8 ACKNOWLEDGE SEQUENCE TIMING
An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN bit of the SSPxCON2 register. When this bit is set, the SCLx pin is pulled low and the contents of the Acknowledge data bit are presented on the SDAx pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCLx pin is deasserted (pulled high). When the SCLx pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCLx pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSPx module then goes into Idle mode (Figure 24-29).

24.6.8.1 WCOL Status Flag
If the user writes the SSPxBUF when an Acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write does not occur).

FIGURE 24-30: ACKNOWLEDGE SEQUENCE WAVEFORM

Note: TBRG = one Baud Rate Generator period.

24.6.9 STOP CONDITION TIMING
A Stop bit is asserted on the SDAx pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN bit of the SSPxCON2 register. At the end of a receive/transmit, the SCLx line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDAx line low. When the SDAx line is sampled low, the Baud Rate Generator is reloaded and counts down to ‘0’. When the Baud Rate Generator times out, the SCLx pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDAx pin will be deasserted. When the SDAx pin is sampled high while SCLx is high, the P bit of the SSPxSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPxCF bit is set (Figure 24-30).

24.6.9.1 WCOL Status Flag
If the user writes the SSPxBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).
24.6.10 SLEEP OPERATION
While in Sleep mode, the \( \text{I}^2\text{C} \) slave module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSPx interrupt is enabled).

24.6.11 EFFECTS OF A RESET
A Reset disables the MSSPx module and terminates the current transfer.

24.6.12 MULTI-MASTER MODE
In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSPx module is disabled. Control of the \( \text{I}^2\text{C} \) bus may be taken when the P bit of the SSPxSTAT register is set, or the bus is idle, with both the S and P bits clear. When the bus is busy, enabling the SSPx interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDAx line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLxIF bit.

The states where arbitration can be lost are:
- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

24.6.13 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDAx pin, arbitration takes place when the master outputs a ‘1’ on SDAx, by letting SDAx float high and another master asserts a ‘0’. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a ‘1’ and the data sampled on the SDAx pin is ‘0’, then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF and reset the \( \text{I}^2\text{C} \) port to its Idle state (Figure 24-31).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDAx and SCLx lines are deasserted and the SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the \( \text{I}^2\text{C} \) bus is free, the user can resume communication by asserting a Start condition.

If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDAx and SCLx lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the \( \text{I}^2\text{C} \) bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDAx and SCLx pins. If a Stop condition occurs, the SSPxIF bit will be set. A write to the SSPxBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the \( \text{I}^2\text{C} \) bus can be taken when the P bit is set in the SSPxSTAT register, or the bus is idle and the S and P bits are cleared.

Note: TBRG = one Baud Rate Generator period.
FIGURE 24-32: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE

Data changes while SCLx = 0  \rightarrow  SDAx line pulled low by another source  \rightarrow  SDAx released by master  \rightarrow  Sample SDAx. While SCLx is high, data does not match what is driven by the master, Bus collision has occurred.

SDAx

SCLx

BCLxIF

Set bus collision interrupt (BCLxIF)
24.6.13.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDAx or SCLx are sampled low at the beginning of the Start condition (Figure 24-32).
b) SCLx is sampled low before SDAx is asserted low (Figure 24-33).

During a Start condition, both the SDAx and the SCLx pins are monitored. If the SDAx pin is already low, or the SCLx pin is already low, then all of the following occur:

• the Start condition is aborted,
• the BCLxIF flag is set and
• the MSSPx module is reset to its Idle state (Figure 24-32).

The Start condition begins with the SDAx and SCLx pins deasserted. When the SDAx pin is sampled high, the Baud Rate Generator is loaded and counts down. If the SCLx pin is sampled low while SDAx is high, a bus collision occurs because it is assumed that another master is attempting to drive a data ‘1’ during the Start condition.

If the SDAx pin is sampled low during this count, the BRG is reset and the SDAx line is asserted early (Figure 24-34). If, however, a ‘1’ is sampled on the SDAx pin, the SDAx pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to zero; if the SCLx pin is sampled as ‘0’ during this time, a bus collision does not occur. At the end of the BRG count, the SCLx pin is asserted low.

Note: The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDAx before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

FIGURE 24-33: BUS COLLISION DURING START CONDITION (SDAx ONLY)
**FIGURE 24-34: BUS COLLISION DURING START CONDITION (SCLx = 0)**

SDAx = 0, SCLx = 1

SDAx

SCLx

Set SEN, enable Start sequence if SDAx = 1, SCLx = 1

SEN

SCLx = 0 before SDAx = 0, bus collision occurs. Set BCLxIF.

BCLxIF

Interrupt cleared by software

S

'S' '0'

SSPxIF

'S' '0'

**FIGURE 24-35: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION**

SDAx = 0, SCLx = 1

SDAx pulled low by other master. Reset BRG and assert SDAx.

SDAx

SCLx

S

SEN

Set SEN, enable Start sequence if SDAx = 1, SCLx = 1

'B' '0'

BCLxIF

Interrupts cleared by software

SSPxIF

SDAx = 0, SCLx = 1

set SSPxIF

Interrupts cleared by software
24.6.13.2  Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDAx when SCLx goes from low level to high level.

b) SCLx goes low before SDAx is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user releases SDAx and the pin is allowed to float high, the BRG is loaded with SSPxADD and counts down to zero. The SCLx pin is then deasserted and when sampled high, the SDAx pin is sampled.

If SDAx is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', Figure 24-35). If SDAx is sampled high, the BRG is reloaded and begins counting. If SDAx goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDAx at exactly the same time.

If SCLx goes from high-to-low before the BRG times out and SDAx has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition, see Figure 24-36.

If, at the end of the BRG time-out, both SCLx and SDAx are still high, the SDAx pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCLx pin, the SCLx pin is driven low and the Repeated Start condition is complete.

FIGURE 24-36: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

FIGURE 24-37: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)
24.6.13.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDAx pin has been deasserted and allowed to float high, SDAx is sampled low after the BRG has timed out.

b) After the SCLx pin is deasserted, SCLx is sampled low before SDAx goes high.

The Stop condition begins with SDAx asserted low. When SDAx is sampled low, the SCLx pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPxADD and counts down to 0. After the BRG times out, SDAx is sampled. If SDAx is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data ‘0’ (Figure 24-37). If the SCLx pin is sampled low before SDAx is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data ‘0’ (Figure 24-38).

**FIGURE 24-38: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

<table>
<thead>
<tr>
<th>TBRG</th>
<th>TBRG</th>
<th>TBRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDAx asserted low</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCLx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCLxIF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P '0'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPxIF '0'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 24-39: BUS COLLISION DURING A STOP CONDITION (CASE 2)**

<table>
<thead>
<tr>
<th>TBRG</th>
<th>TBRG</th>
<th>TBRG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assert SDAx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCLx goes low before SDAx goes high, set BCLxIF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PEN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCLxIF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P '0'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPxIF '0'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**TABLE 24-3: 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>TMR0IF</td>
<td>INTF</td>
<td>IOCIF</td>
<td>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSFIE</td>
<td>C2IE</td>
<td>C1IE</td>
<td>EEIE</td>
<td>BCL1IE</td>
<td>—</td>
<td>—</td>
<td>CCP2IE(^{(1)})</td>
<td>91</td>
</tr>
<tr>
<td>PIE4(^{(1)})</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BCL2IE</td>
<td>SSP2IE</td>
<td>93</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMRI1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSFIF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>EEIF</td>
<td>BCL1IF</td>
<td>—</td>
<td>—</td>
<td>CCP2IF(^{(1)})</td>
<td>95</td>
</tr>
<tr>
<td>PIR4(^{(1)})</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>BCL2IF</td>
<td>SSP2IF</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>120</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
<tr>
<td>SSPxADD</td>
<td>ADD7</td>
<td>ADD6</td>
<td>ADD5</td>
<td>ADD4</td>
<td>ADD3</td>
<td>ADD2</td>
<td>ADD1</td>
<td>ADD0</td>
<td>285</td>
</tr>
<tr>
<td>SSPxBUF</td>
<td>MSSPx Receive Buffer/Transmit Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>237*</td>
</tr>
<tr>
<td>SSPxCON1</td>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPEN</td>
<td>CKP</td>
<td>SSPM3</td>
<td>SSPM2</td>
<td>SSPM1</td>
<td>SSPM0</td>
<td>282</td>
</tr>
<tr>
<td>SSPxCON2</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>283</td>
</tr>
<tr>
<td>SSPxCON3</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>284</td>
</tr>
<tr>
<td>SSPxMSK</td>
<td>MSK7</td>
<td>MSK6</td>
<td>MSK5</td>
<td>MSK4</td>
<td>MSK3</td>
<td>MSK2</td>
<td>MSK1</td>
<td>MSK0</td>
<td>285</td>
</tr>
<tr>
<td>SSPxSTAT</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>281</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used by the MSSP module in I²C™ mode.

*Page provides register information.

**Note 1:** PIC16F/LF1827 only.
24.7 BAUD RATE GENERATOR

The MSSPx module has a Baud Rate Generator available for clock generation in both I^2C and SPI Master modes. The Baud Rate Generator (BRG) reload value is placed in the SSPxADD register (Register 24-6). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting down.

Once the given operation is complete, the internal clock will automatically stop counting and the clock pin will remain in its last state.

An internal signal “Reload” in Figure 24-39 triggers the value from SSPxADD to be loaded into the BRG counter. This occurs twice for each oscillation of the module clock line. The logic dictating when the reload signal is asserted depends on the mode the MSSPx is being operated in.

Table 24-4 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD.

**EQUATION 24-1:**

\[
F_{\text{clock}} = \frac{F_{\text{osc}}}{(SPSxADD + 1)(4)}
\]

**FIGURE 24-40: BAUD RATE GENERATOR BLOCK DIAGRAM**

- **TABLE 24-4: MSSPX 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:** The I^2C interface does not conform to the 400 kHz I^2C 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.

**Note 2:** SPI mode only.
REGISTER 24-1: SSPXSTAT: SSPX 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:
  SMP 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 SM bus specification
  0 = Disable SM bus 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 MSSPx module is disabled, SSPxEN is cleared.)
  1 = Indicates that a Stop bit has been detected last (this bit is ‘0’ on Reset)
  0 = Stop bit was not detected last

bit 3  S: Start bit
- (I²C mode only. This bit is cleared when the MSSPx module is disabled, SSPxEN is cleared.)
  1 = Indicates that a Start bit has been detected last (this bit is ‘0’ on Reset)
  0 = Start bit was not detected last

bit 2  R/W: Read/Write bit information (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 MSSPx 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 SSPxADD register
- 0 = Address does not need to be updated

bit 0  BF: Buffer Full Status bit
- Receive (SPI and I²C modes):
  1 = Receive complete, SSPxBUF is full
  0 = Receive not complete, SSPxBUF is empty
- Transmit (I²C mode only):
  1 = Data transmit in progress (does not include the ACK and Stop bits), SSPxBUF is full
  0 = Data transmit complete (does not include the ACK and Stop bits), SSPxBUF is empty
**REGISTER 24-2: SSPXCON1: SSPX CONTROL REGISTER 1**

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCOL</td>
<td>SSPxOV</td>
<td>SSPxEN</td>
<td>CKP</td>
<td>SSPxM3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCOL: Write Collision Detect bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Master mode:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = A write to the SSPxBUF register was attempted while the \textit{i}^{2}\text{C} conditions were not valid for a transmission to be started</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = No collision</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Slave mode:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = No collision</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6</th>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSPxOV: Receive Overflow Indicator bit(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In SPI mode:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = A new byte is received while the SSPxBUF register is still holding the previous data. In case of overflow, the data in SSPxSR is lost.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overflow can only occur in Slave mode. In Slave mode, the user must read the SSPxBUF, even if only transmitting data, to avoid setting overflow. In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register (must be cleared in software).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = No overflow</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>In \textit{i}^{2}\text{C} mode:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = A byte is received while the SSPxBUF register is still holding the previous byte. SSPxOV is a “don't care” in Transmit mode (must be cleared in software).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = No overflow</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5</th>
<th>bit 4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSPxEN: Synchronous Serial Port Enable bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In both modes, when enabled, these pins must be properly configured as input or output</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In SPI mode:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = Enables serial port and configures SCKx, SDOx, SDIx and SSx as the source of the serial port pins(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = Disables serial port and configures these pins as I/O port pins</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In \textit{i}^{2}\text{C} mode:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = Enables the serial port and configures the SDAx and SCLx pins as the source of the serial port pins(3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 = Disables serial port and configures these pins as I/O port pins</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 4</th>
<th>bit 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKP: Clock Polarity Select bit</td>
<td></td>
</tr>
<tr>
<td>In SPI mode:</td>
<td></td>
</tr>
<tr>
<td>1 = Idle state for clock is a high level</td>
<td></td>
</tr>
<tr>
<td>0 = Idle state for clock is a low level</td>
<td></td>
</tr>
<tr>
<td>In \textit{i}^{2}\text{C} Slave mode:</td>
<td></td>
</tr>
<tr>
<td>SClx release control</td>
<td></td>
</tr>
<tr>
<td>1 = Enable clock</td>
<td></td>
</tr>
<tr>
<td>0 = Holds clock low (clock stretch). (Used to ensure data setup time.)</td>
<td></td>
</tr>
<tr>
<td>In \textit{i}^{2}\text{C} Master mode:</td>
<td></td>
</tr>
<tr>
<td>Unused in this mode</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 3-0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSPxM&lt;3:0&gt;: Synchronous Serial Port Mode Select bits</td>
<td></td>
</tr>
<tr>
<td>0000 = SPI Master mode, clock = Fosc/4</td>
<td></td>
</tr>
<tr>
<td>0001 = SPI Master mode, clock = Fosc/16</td>
<td></td>
</tr>
<tr>
<td>0010 = SPI Master mode, clock = Fosc/64</td>
<td></td>
</tr>
<tr>
<td>0011 = SPI Master mode, clock = TMR2 output/2</td>
<td></td>
</tr>
<tr>
<td>0100 = SPI Slave mode, clock = SCKx pin, SSx pin control enabled</td>
<td></td>
</tr>
<tr>
<td>0101 = SPI Slave mode, clock = SCKx pin, SSx pin control disabled, SSx can be used as I/O pin</td>
<td></td>
</tr>
<tr>
<td>0110 = \textit{i}^{2}\text{C} Slave mode, 7-bit address</td>
<td></td>
</tr>
<tr>
<td>0111 = \textit{i}^{2}\text{C} Slave mode, 10-bit address</td>
<td></td>
</tr>
<tr>
<td>1000 = \textit{i}^{2}\text{C} Master mode, clock = Fosc / (4 * (SSPxADD+1))(4)</td>
<td></td>
</tr>
<tr>
<td>1001 = Reserved</td>
<td></td>
</tr>
<tr>
<td>1010 = SPI Master mode, clock = Fosc / (4 * (SSPxADD+1))</td>
<td></td>
</tr>
<tr>
<td>1011 = \textit{i}^{2}\text{C} firmware controlled Master mode (slave idle)</td>
<td></td>
</tr>
<tr>
<td>1100 = Reserved</td>
<td></td>
</tr>
<tr>
<td>1101 = Reserved</td>
<td></td>
</tr>
<tr>
<td>1110 = \textit{i}^{2}\text{C} Slave mode, 7-bit address with Start and Stop bit interrupts enabled</td>
<td></td>
</tr>
<tr>
<td>1111 = \textit{i}^{2}\text{C} Slave mode, 10-bit address with Start and Stop bit interrupts enabled</td>
<td></td>
</tr>
</tbody>
</table>

**Note**:

1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register.
2: When enabled, these pins must be properly configured as input or output.
3: When enabled, the SDAx and SCLx pins must be configured as inputs.
4: SSPxADD values of 0, 1 or 2 are not supported for \textit{i}^{2}\text{C} Mode.
REGISTER 24-3:  SSPXCON2: SSPX 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/HC-0/0</th>
<th>R/S/HC-0/0</th>
<th>R/S/HC-0/0</th>
<th>R/S/HC-0/0</th>
<th>R/W/HC-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>

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

bit 7  **GCEN**: General Call Enable bit (in I²C Slave mode only)
- 1 = Enable interrupt when a general call address (0x00 or 00h) is received in the SSPxSR
- 0 = General call address disabled

bit 6  **ACKSTAT**: Acknowledge Status bit (in I²C mode only)
- 1 = Acknowledge was not received
- 0 = Acknowledge was received

bit 5  **ACKDT**: Acknowledge Data bit (in I²C mode only)
- **In Receive mode:**
  - Value transmitted when the user initiates an Acknowledge sequence at the end of a receive
- 1 = Not Acknowledge
- 0 = Acknowledge

bit 4  **ACKEN**: Acknowledge Sequence Enable bit (in I²C Master mode only)
- **In Master Receive mode:**
  - 1 = Initiate Acknowledge sequence on SDAx and SCLx pins, and transmit ACKDT data bit.
  - Automatically cleared by hardware.
- 0 = Acknowledge sequence idle

bit 3  **RCEN**: Receive Enable bit (in I²C Master mode only)
- 1 = Enables Receive mode for I²C
- 0 = Receive idle

bit 2  **PEN**: Stop Condition Enable bit (in I²C Master mode only)
- **SCxK Release Control:**
  - 1 = Initiate Stop condition on SDAx and SCLx pins. Automatically cleared by hardware.
  - 0 = Stop condition Idle

bit 1  **RSEN**: Repeated Start Condition Enabled bit (in I²C Master mode only)
- 1 = Initiate Repeated Start condition on SDAx and SCLx pins. Automatically cleared by hardware.
- 0 = Repeated Start condition Idle

bit 0  **SEN**: Start Condition Enabled bit (in I²C Master mode only)
- **In Master mode:**
  - 1 = Initiate Start condition on SDAx and SCLx pins. Automatically cleared by hardware.
  - 0 = Start condition Idle
- **In Slave mode:**
  - 1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
  - 0 = Clock stretching is disabled

**Note 1:** For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I²C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).
REGISTER 24-4:  SSPXCON3: SSPX 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

**Legend:**
- **ACKTIM**: Acknowledge Time Status bit (I2C mode only)
  - 1 = Indicates the I2C bus is in an Acknowledge sequence, set on 8TH falling edge of SCLx clock
  - 0 = Not an Acknowledge sequence, cleared on 9TH rising edge of SCLx clock

**Legend:**
- **PCIE**: Stop Condition Interrupt Enable bit (I2C mode only)
  - 1 = Enable interrupt on detection of Stop condition
  - 0 = Stop detection interrupts are disabled

**Legend:**
- **SCIE**: Start Condition Interrupt Enable bit (I2C mode only)
  - 1 = Enable interrupt on detection of Start or Restart conditions
  - 0 = Start detection interrupts are disabled

**Legend:**
- **BOEN**: Buffer Overwrite Enable bit
  - In SPI Slave mode:
    - 1 = SSPxBUF updates every time that a new data byte is shifted in ignoring the BF bit
    - 0 = If new byte is received with BF bit of the SSPxSTAT register already set, SSPxOV bit of the SSPxCON1 register is set, and the buffer is not updated
  - In I2C Master mode:
    - This bit is ignored.
  - In I2C Slave mode:
    - 1 = SSPxBUF is updated and ACK is generated for a received address/data byte, ignoring the state of the SSPxOV bit only if the BF bit = 0.
    - 0 = SSPxBUF is only updated when SSPxOV is clear

**Legend:**
- **SDAHT**: SDAx Hold Time Selection bit (I2C mode only)
  - 1 = Minimum of 300 ns hold time on SDAx after the falling edge of SCLx
  - 0 = Minimum of 100 ns hold time on SDAx after the falling edge of SCLx

**Legend:**
- **SBCDE**: Slave Mode Bus Collision Detect Enable bit (I2C Slave mode only)
  - If on the rising edge of SCLx, SDAx is sampled low when the module is outputting a high state, the BCLx1F bit of the PIR2 register is set, and bus goes idle
    - 1 = Enable slave bus collision interrupts
    - 0 = Slave bus collision interrupts are disabled

**Legend:**
- **AHEN**: Address Hold Enable bit (I2C Slave mode only)
  - 1 = Following the 8th falling edge of SCLx for a matching received address byte; CKP bit of the SSPxCON1 register will be cleared and the SCLx will be held low.
  - 0 = Address holding is disabled

**Legend:**
- **DHEN**: Data Hold Enable bit (I2C Slave mode only)
  - 1 = Following the 8th falling edge of SCLx for a received data byte; slave hardware clears the CKP bit of the SSPxCON1 register and SCLx is held low.
  - 0 = Data holding is disabled

**Note 1:** For daisy-chained SPI operation; allows the user to ignore all but the last received byte. SSPxOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPxBUF.

**Note 2:** This bit has no effect in Slave modes that Start and Stop condition detection is explicitly listed as enabled.
REGISTER 24-5: SSPXMSK: SSPX 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>MSK7</td>
<td>MSK6</td>
<td>MSK5</td>
<td>MSK4</td>
<td>MSK3</td>
<td>MSK2</td>
<td>MSK1</td>
<td>MSK0</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 7-1 MSK<7:1>: Mask bits
  - 1 = The received address bit n is compared to SSPxADD<n> to detect I2C address match
  - 0 = The received address bit n is not used to detect I2C address match

- bit 0 MSK<0>: Mask bit for I2C Slave mode, 10-bit Address
  - I2C Slave mode, 10-bit address (SSPxM<3:0> = 0111 or 1111):
    - 1 = The received address bit 0 is compared to SSPxADD<0> to detect I2C address match
    - 0 = The received address bit 0 is not used to detect I2C address match
  - I2C Slave mode, 7-bit address, the bit is ignored

REGISTER 24-6: SSPXADD: MSSPX ADDRESS AND BAUD RATE REGISTER (I2C 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>ADD7</td>
<td>ADD6</td>
<td>ADD5</td>
<td>ADD4</td>
<td>ADD3</td>
<td>ADD2</td>
<td>ADD1</td>
<td>ADD0</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

Master mode:

- bit 7-0 ADD<7:0>: Baud Rate Clock Divider bits
  SCLx pin clock period = ((ADD<7:0> + 1) * 4)/FOSC

10-Bit Slave mode — Most Significant Address byte:

- bit 7-3 Not used: Unused for Most Significant Address byte. Bit state of this register is a “don’t care”. Bit pattern sent by master is fixed by I2C specification and must be equal to ‘11110’. However, those bits are compared by hardware and are not affected by the value in this register.

- bit 2-1 ADD<2:1>: Two Most Significant bits of 10-bit address

- bit 0 Not used: Unused in this mode. Bit state is a “don’t care”.

10-Bit Slave mode — Least Significant Address byte:

- bit 7-0 ADD<7:0>: Eight Least Significant bits of 10-bit address

7-Bit Slave mode:

- bit 7-1 ADD<7:1>: 7-bit address

- bit 0 Not used: Unused in this mode. Bit state is a “don’t care”.
25.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer independent of device program execution. The EUSART, also known as a Serial Communications Interface (SCI), can be configured as a full-duplex asynchronous system or half-duplex synchronous system. Full-Duplex mode is useful for communications with peripheral systems, such as CRT terminals and personal computers. Half-Duplex Synchronous mode is intended for communications with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs or other microcontrollers. These devices typically do not have internal clocks for baud rate generation and require the external clock signal provided by a master synchronous device.

The EUSART module includes the following capabilities:
- Full-duplex asynchronous transmit and receive
- Two-character input buffer
- One-character output buffer
- Programmable 8-bit or 9-bit character length
- Address detection in 9-bit mode
- Input buffer overrun error detection
- Received character framing error detection
- Half-duplex synchronous master
- Half-duplex synchronous slave
- Programmable clock polarity in synchronous modes
- Sleep operation

The EUSART module implements the following additional features, making it ideally suited for use in Local Interconnect Network (LIN) bus systems:
- Automatic detection and calibration of the baud rate
- Wake-up on Break reception
- 13-bit Break character transmit

Block diagrams of the EUSART transmitter and receiver are shown in Figure 25-1 and Figure 25-2.

FIGURE 25-1: EUSART TRANSMIT BLOCK DIAGRAM

[Block Diagram Image]

- TXREG Register
- TXIE Interrupt
- MSb LSb
- TXEN
- Multiplier x4 x16 x64
- BRG16
- SPBRGH SPBRGL
- FOSC
- n
- TRMT SPEN
- TX9D
- TX9
- TXIF
- TXCK pin
- Data Bus
- Pin Buffer and Control

---

© 2009 Microchip Technology Inc. Preliminary DS41391B-page 287
The operation of the EUSART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)

These registers are detailed in Register 25-1, Register 25-2 and Register 25-3, respectively.

When the receiver or transmitter section is not enabled then the corresponding RX or TX pin may be used for general purpose input and output.
25.1 EUSART Asynchronous Mode

The EUSART transmits and receives data using the standard non-return-to-zero (NRZ) format. NRZ is implemented with two levels: a VOH mark state which represents a ‘1’ data bit, and a VOL space state which represents a ‘0’ data bit. NRZ refers to the fact that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port idles in the mark state. Each character transmission consists of one Start bit followed by eight or nine data bits and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is 8 bits. Each transmitted bit persists for a period of 1/(Baud Rate). An on-chip dedicated 8-bit/16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See Table 25-5 for examples of baud rate configurations.

The EUSART transmits and receives the LSB first. The EUSART’s transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is not supported by the hardware, but can be implemented in software and stored as the ninth data bit.

25.1.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 25-1. The heart of the transmitter is the serial Transmit Shift Register (TSR), which is not directly accessible by software. The TSR obtains its data from the transmit buffer, which is the TXREG register.

25.1.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- TXEN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the TXEN bit of the TXSTA register enables the transmitter circuitry of the EUSART. Clearing the SYNC bit of the TXSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTA register enables the EUSART and automatically configures the TX/CK I/O pin as an output.

Note 1: The TXIF Transmitter Interrupt flag is set when the TXEN enable bit is set.

25.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREG is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the TXREG until the Stop bit of the previous character has been transmitted. The pending character in the TXREG is then transferred to the TSR in one Tcy immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXREG.

25.1.3 Transmit Interrupt Flag

The TXIF interrupt flag bit of the PIR1 register is set whenever the EUSART transmitter is enabled and no character is being held for transmission in the TXREG. In other words, the TXIF bit is only clear when the TSR is busy with a character and a new character has been queued for transmission in the TXREG. The TXIF flag bit is not cleared immediately upon writing TXREG. TXIF becomes valid in the second instruction cycle following the write execution. Polling TXIF immediately following the TXREG write will return invalid results. The TXIF bit is read-only, it cannot be set or cleared by software.

The TXIF interrupt can be enabled by setting the TXIE interrupt enable bit of the PIE1 register. However, the TXIF flag bit will be set whenever the TXREG is empty, regardless of the state of TXIE enable bit.

To use interrupts when transmitting data, set the TXIE bit only when there is more data to send. Clear the TXIE interrupt enable bit upon writing the last character of the transmission to the TXREG.
25.1.1.4 TSR Status

The TRMT bit of the TXSTA register indicates the status of the TSR register. This is a read-only bit. The TRMT bit is set when the TSR register is empty and is cleared when a character is transferred to the TSR register from the TXREG. The TRMT bit remains clear until all bits have been shifted out of the TSR register. No interrupt logic is tied to this bit, so the user has to poll this bit to determine the TSR status.

Note: The TSR register is not mapped in data memory, so it is not available to the user.

25.1.1.5 Transmitting 9-Bit Characters

The EUSART supports 9-bit character transmissions. When the TX9 bit of the TXSTA register is set the EUSART will shift 9 bits out for each character transmitted. The TX9D bit of the TXSTA register is the ninth, and Most Significant, data bit. When transmitting 9-bit data, the TX9D data bit must be written before writing the 8 Least Significant bits into the TXREG. All nine bits of data will be transferred to the TSR shift register immediately after the TXREG is written.

A special 9-bit Address mode is available for use with multiple receivers. See Section 25.1.2.7 “Address Detection” for more information on the address mode.

FIGURE 25-3: ASYNCHRONOUS TRANSMISSION

FIGURE 25-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)

25.1.1.6 Asynchronous Transmission Set-up:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see Section 25.3 “EUSART Baud Rate Generator (BRG)”).
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If 9-bit transmission is desired, set the TX9 control bit. A set ninth data bit will indicate that the 8 Least Significant data bits are an address when the receiver is set for address detection.
4. Enable the transmission by setting the TXEN control bit. This will cause the TXIF interrupt bit to be set.
5. If interrupts are desired, set the TXIE interrupt enable bit of the PIE1 register. An interrupt will occur immediately provided that the GIE and PEIE bits of the INTCON register are also set.
6. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
7. Load 8-bit data into the TXREG register. This will start the transmission.

Note: This timing diagram shows two consecutive transmissions.
### TABLE 25-1: SUMMARY OF REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

<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>BAUDCON</td>
<td>ABDOVF</td>
<td>RCIDL</td>
<td>—</td>
<td>SCKP</td>
<td>BRG16</td>
<td>—</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</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>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>297</td>
</tr>
<tr>
<td>SPBRG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>299*</td>
</tr>
<tr>
<td>TXREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>289*</td>
</tr>
<tr>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>296</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented read as ‘0’. Shaded cells are not used for Asynchronous Transmission.

* Page provides register information.
25.1.2 EUSART ASYNCHRONOUS RECEIVER

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in Figure 25-2. The data is received on the RX/DT pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all 8 or 9 bits of the character have been shifted in, they are immediately transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must start servicing the EUSART receiver. The FIFO and RSR registers are not directly accessible by software. Access to the received data is via the RCREG register.

25.1.2.1 Enabling the Receiver

The EUSART receiver is enabled for asynchronous operation by configuring the following three control bits:
- CREN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the CREN bit of the RCSTA register enables the receiver circuitry of the EUSART. Clearing the SYNC bit of the TXSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCSTA register enables the EUSART and automatically configures the RX/DT I/O pin as an input.

Note: When the SPEN bit is set the TX/CK I/O pin is automatically configured as an output, regardless of the state of the corresponding TRIS bit and whether or not the EUSART transmitter is enabled. The PORT latch is disconnected from the output driver so it is not possible to use the TX/CK pin as a general purpose output.

25.1.2.2 Receiving Data

The receiver data recovery circuit initiates character reception on the falling edge of the first bit. The first bit, also known as the Start bit, is always a zero. The data recovery circuit counts one-half bit time to the center of the Start bit and verifies that the bit is still a zero. If it is not a zero then the data recovery circuit aborts character reception, without generating an error, and resumes looking for the falling edge of the Start bit. If the Start bit zero verification succeeds then the data recovery circuit counts a full bit time to the center of the next bit. The bit is then sampled by a majority detect circuit and the resulting ‘0’ or ‘1’ is shifted into the RSR. This repeats until all data bits have been sampled and shifted into the RSR. One final bit time is measured and the level sampled. This is the Stop bit, which is always a ‘1’. If the data recovery circuit samples a ‘0’ in the Stop bit position then a framing error is set for this character, otherwise the framing error is cleared for this character. See Section 25.1.2.4 “Receive Framing Error” for more information on framing errors.

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the EUSART receive FIFO and the RCIF interrupt flag bit of the PIR1 register is set. The top character in the FIFO is transferred out of the FIFO by reading the RCREG register.

Note: If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See Section 25.1.2.5 “Receive Overrun Error” for more information on overrun errors.

25.1.2.3 Receive Interrupts

The RCIF interrupt flag bit of the PIR1 register is set whenever the EUSART receiver is enabled and there is an unread character in the receive FIFO. The RCIF interrupt flag bit is read-only, it cannot be set or cleared by software.

RCIF interrupts are enabled by setting all of the following bits:
- RCIE interrupt enable bit of the PIE1 register
- PEIE peripheral interrupt enable bit of the INTCON register
- GIE global interrupt enable bit of the INTCON register

The RCIF interrupt flag bit will be set when there is an unread character in the FIFO, regardless of the state of interrupt enable bits.
25.1.2.4 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error Status bit. A framing error indicates that a Stop bit was not seen at the expected time. The framing error status is accessed via the FERR bit of the RCSTA register. The FERR bit represents the status of the top unread character in the receive FIFO. Therefore, the FERR bit must be read before reading the RCREG.

The FERR bit is read-only and only applies to the top unread character in the receive FIFO. A framing error (FERR = 1) does not preclude reception of additional characters. It is not necessary to clear the FERR bit. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERR bit can be forced clear by clearing the SPEN bit of the RCSTA register which resets the EUSART. Clearing the CREN bit of the RCSTA register does not affect the FERR bit. A framing error by itself does not generate an interrupt.

Note: If all receive characters in the receive FIFO have framing errors, repeated reads of the RCREG will not clear the FERR bit.

25.1.2.5 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before the FIFO is accessed. When this happens the OERR bit of the RCSTA register is set. The characters already in the FIFO buffer can be read but no additional characters will be received until the error is cleared. The error must be cleared by either clearing the CREN bit of the RCSTA register or by resetting the EUSART by clearing the SPEN bit of the RCSTA register.

25.1.2.6 Receiving 9-bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTA register is set the EUSART will shift 9 bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth and Most Significant data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the 8 Least Significant bits from the RCREG.

25.1.2.7 Address Detection

A special Address Detection mode is available for use when multiple receivers share the same transmission line, such as in RS-485 systems. Address detection is enabled by setting the ADDEN bit of the RCSTA register.

Address detection requires 9-bit character reception. When address detection is enabled, only characters with the ninth data bit set will be transferred to the receive FIFO buffer, thereby setting the RCIF interrupt bit. All other characters will be ignored.

Upon receiving an address character, user software determines if the address matches its own. Upon address match, user software must disable address detection by clearing the ADDEN bit before the next Stop bit occurs. When user software detects the end of the message, determined by the message protocol used, software places the receiver back into the Address Detection mode by setting the ADDEN bit.
25.1.2.8 Asynchronous Reception Set-up:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see Section 25.3 “EUSART Baud Rate Generator (BRG)”).
2. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
3. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Enable reception by setting the CREN bit.
6. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
7. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
8. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register.
9. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

25.1.2.9 9-bit Address Detection Mode Set-up

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see Section 25.3 “EUSART Baud Rate Generator (BRG)”).
2. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
3. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
4. Enable 9-bit reception by setting the RX9 bit.
5. Enable address detection by setting the ADDEN bit.
6. Enable reception by setting the CREN bit.
7. The RCIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
8. Read the RCSTA register to get the error flags. The ninth data bit will always be set.
9. Get the received 8 Least Significant data bits from the receive buffer by reading the RCREG register. Software determines if this is the device’s address.
10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

FIGURE 25-5: ASYNCHRONOUS RECEPTION

Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word, causing the OERR (overrun) bit to be set.
### TABLE 25-2: SUMMARY OF REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

<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>BAUDCON</td>
<td>ABDVF</td>
<td>BDVF</td>
<td>—</td>
<td>SCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>298</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>—</td>
<td>TMR0IE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>RCREG</td>
<td>EUSARTReceive Data Register</td>
<td>292*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>297</td>
</tr>
<tr>
<td>SPCR</td>
<td>BSF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SPCR</td>
<td>BSF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
<tr>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>296</td>
</tr>
</tbody>
</table>

**Legend:**
- — = unimplemented read as '0'. Shaded cells are not used for Asynchronous Reception.
- * Page provides register information.
25.2 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (INTOSC). However, the INTOSC frequency may drift as VDD or temperature changes, and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the INTOSC output. Adjusting the value in the OSCTUNE register allows for fine resolution changes to the system clock source. See Section 5.2.2 "Internal Clock Sources" for more information.

The other method adjusts the value in the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see Section 25.3.1 "Auto-Baud Detect"). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change in the peripheral clock frequency.

REGISTER 25-1: TXSTA: TRANSMIT STATUS AND CONTROL 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-1/1</th>
<th>R/W-0/0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN(1)</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
</tr>
</tbody>
</table>

bit 7   CSRC: Clock Source Select bit
        Asynchronous mode:
        Don't care
        Synchronous mode:
        1 = Master mode (clock generated internally from BRG)
        0 = Slave mode (clock from external source)

bit 6   TX9: 9-bit Transmit Enable bit
        1 = Selects 9-bit transmission
        0 = Selects 8-bit transmission

bit 5   TXEN: Transmit Enable bit(1)
        1 = Transmit enabled
        0 = Transmit disabled

bit 4   SYNC: EUSART Mode Select bit
        1 = Synchronous mode
        0 = Asynchronous mode

bit 3   SENDB: Send Break Character bit
        Asynchronous mode:
        1 = Send Sync Break on next transmission (cleared by hardware upon completion)
        0 = Sync Break transmission completed
        Synchronous mode:
        Don't care

bit 2   BRGH: High Baud Rate Select bit
        Asynchronous mode:
        1 = High speed
        0 = Low speed
        Synchronous mode:
        Unused in this mode

bit 1   TRMT: Transmit Shift Register Status bit
        1 = TSR empty
        0 = TSR full

bit 0   TX9D: Ninth bit of Transmit Data
        Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode.
REGISTER 25-2:  RCSTA: RECEIVE STATUS AND CONTROL REGISTER(1)

<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-0/0</th>
<th>R-0/0</th>
<th>R-x/x</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
</tr>
</tbody>
</table>

bit 7  SPEN: Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6  RX9: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5  SREN: Single Receive Enable bit
Asynchronous mode:
Don’t care
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave
Don’t care

bit 4  CREN: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3  ADDEN: Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
Don’t care

bit 2  FERR: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1  OERR: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0  RX9D: Ninth bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.
REGISTER 25-3: BAUDCON: BAUD RATE CONTROL REGISTER

<table>
<thead>
<tr>
<th>bit 7</th>
<th>R-0/0</th>
<th>R-1/1</th>
<th>U-0</th>
<th>R/W-0/0</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>ABDOVF</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ABDEN</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
- **x** = Bit is unknown
- **1** = Bit is set
- **0** = Bit is cleared

**bit 7**
ABDOVF: Auto-Baud Detect Overflow bit
- **Asynchronous mode:**
  - 1 = Auto-baud timer overflowed
  - 0 = Auto-baud timer did not overflow
- **Synchronous mode:**
  - Don’t care

**bit 6**
RCIDL: Receive Idle Flag bit
- **Asynchronous mode:**
  - 1 = Receiver is idle
  - 0 = Start bit has been received and the receiver is receiving
- **Synchronous mode:**
  - Don’t care

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

**bit 4**
SCKP: Synchronous Clock Polarity Select bit
- **Asynchronous mode:**
  - 1 = Transmit inverted data to the RB7/TX/CK pin
  - 0 = Transmit non-inverted data to the RB7/TX/CK pin
- **Synchronous mode:**
  - 1 = Data is clocked on rising edge of the clock
  - 0 = Data is clocked on falling edge of the clock

**bit 3**
BRG16: 16-bit Baud Rate Generator bit
- 1 = 16-bit Baud Rate Generator is used
- 0 = 8-bit Baud Rate Generator is used

**bit 2**
Unimplemented: Read as ‘0’

**bit 1**
WUE: Wake-up Enable bit
- **Asynchronous mode:**
  - 1 = Receiver is waiting for a falling edge. No character will be received, byte RCIF will be set. WUE will automatically clear after RCIF is set.
  - 0 = Receiver is operating normally
- **Synchronous mode:**
  - Don’t care

**bit 0**
ABDEN: Auto-Baud Detect Enable bit
- **Asynchronous mode:**
  - 1 = Auto-Baud Detect mode is enabled (clears when auto-baud is complete)
  - 0 = Auto-Baud Detect mode is disabled
- **Synchronous mode:**
  - Don’t care
25.3 EUSART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is an 8-bit or 16-bit timer that is dedicated to the support of both the asynchronous and synchronous EUSART operation. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit of the BAUDCON register selects 16-bit mode.

The SPBRGH, SPBRGL register pair determines the period of the free running baud rate timer. In Asynchronous mode the multiplier of the baud rate period is determined by both the BRGH bit of the TXSTA register and the BRG16 bit of the BAUDCON register. In Synchronous mode, the BRGH bit is ignored.

Table 25-3 contains the formulas for determining the baud rate. Example 25-1 provides a sample calculation for determining the baud rate and baud rate error.

Typical baud rates and error values for various asynchronous modes have been computed for your convenience and are shown in Table 25-3. It may be advantageous to use the high baud rate (BRGH = 1), or the 16-bit BRG (BRG16 = 1) to reduce the baud rate error. The 16-bit BRG mode is used to achieve slow baud rates for fast oscillator frequencies.

Writing a new value to the SPBRGH, SPBRGL register pair causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RCIDL bit to make sure that the receive operation is idle before changing the system clock.

**EXAMPLE 25-1: CALCULATING BAUD RATE ERROR**

For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

\[
\text{Desired Baud Rate} = \frac{Fosc}{64(\text{SPBRGH:SPBRGL} + 1)}
\]

Solving for SPBRGH:SPBRGL:

\[
X = \frac{Fosc}{\text{Desired Baud Rate}} - 1
\]

\[
= \frac{16000000}{9600} - 1
\]

\[
= 25.042
\]

\[
\text{Calculated Baud Rate} = \frac{16000000}{64(25 + 1)}
\]

\[
= 9615
\]

\[
\text{Error} = \frac{\text{Calculated Baud Rate} - \text{Desired Baud Rate}}{\text{Desired Baud Rate}}
\]

\[
= \frac{9615 - 9600}{9600} = 0.16%
\]
### TABLE 25-3: BAUD RATE FORMULAS

<table>
<thead>
<tr>
<th>Configuration Bits</th>
<th>BRG/EUSART Mode</th>
<th>Baud Rate Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>8-bit/Asynchronous</td>
<td>Fosc/[64 (n+1)]</td>
</tr>
<tr>
<td>0 0 1</td>
<td>8-bit/Asynchronous</td>
<td>Fosc/[16 (n+1)]</td>
</tr>
<tr>
<td>0 1 0</td>
<td>16-bit/Asynchronous</td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>16-bit/Asynchronous</td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>8-bit/Synchronous</td>
<td>Fosc/[4 (n+1)]</td>
</tr>
<tr>
<td>1 1 0</td>
<td>16-bit/Synchronous</td>
<td></td>
</tr>
</tbody>
</table>

Legend:  
$x$ = Don’t care, $n$ = value of SPBRGH, SPBRGL register pair

### TABLE 25-4: SUMMARY OF REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR

<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>BAUDCON</td>
<td></td>
<td></td>
<td></td>
<td>SCKP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>298</td>
</tr>
<tr>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>297</td>
</tr>
<tr>
<td>SPBRGL</td>
<td>Baud Rate Generator Data Register Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>299*</td>
</tr>
<tr>
<td>SPBREGH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>299*</td>
</tr>
<tr>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>296</td>
</tr>
</tbody>
</table>

Legend:  
— = unimplemented read as ‘0’. Shaded cells are not used for the Baud Rate Generator.

* Page provides register information.
**TABLE 25-5: BAUD RATES FOR ASYNCHRONOUS MODES**

<table>
<thead>
<tr>
<th>BAUD RATE</th>
<th>Fosc = 32.000 MHz</th>
<th>Fosc = 20.000 MHz</th>
<th>Fosc = 18.432 MHz</th>
<th>Fosc = 11.0592 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate</td>
<td>% Error</td>
<td>SPBRG value</td>
<td>Actual Rate</td>
</tr>
<tr>
<td>300</td>
<td>1221</td>
<td>1.73</td>
<td>255</td>
<td>1200</td>
</tr>
<tr>
<td>1200</td>
<td>47</td>
<td>0.16</td>
<td>207</td>
<td>2404</td>
</tr>
<tr>
<td>2400</td>
<td>9470</td>
<td>-1.36</td>
<td>32</td>
<td>10417</td>
</tr>
<tr>
<td>9600</td>
<td>19.23k</td>
<td>0.16</td>
<td>25</td>
<td>19.53k</td>
</tr>
<tr>
<td>10417</td>
<td>57.6k</td>
<td>-3.55</td>
<td>3</td>
<td>57.60k</td>
</tr>
<tr>
<td>19.2k</td>
<td>55.55k</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57.6k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115.2k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BAUD RATE</th>
<th>Fosc = 8.000 MHz</th>
<th>Fosc = 4.000 MHz</th>
<th>Fosc = 3.6864 MHz</th>
<th>Fosc = 1.000 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate</td>
<td>% Error</td>
<td>SPBRG value</td>
<td>Actual Rate</td>
</tr>
<tr>
<td>300</td>
<td>300</td>
<td>0.16</td>
<td>207</td>
<td>300</td>
</tr>
<tr>
<td>1200</td>
<td>1202</td>
<td>0.16</td>
<td>103</td>
<td>1202</td>
</tr>
<tr>
<td>2400</td>
<td>2404</td>
<td>0.16</td>
<td>51</td>
<td>2404</td>
</tr>
<tr>
<td>9600</td>
<td>9615</td>
<td>0.16</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>10417</td>
<td>10417</td>
<td>0.00</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>19.2k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57.6k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115.2k</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BAUD RATE</th>
<th>Fosc = 32.000 MHz</th>
<th>Fosc = 20.000 MHz</th>
<th>Fosc = 18.432 MHz</th>
<th>Fosc = 11.0592 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate</td>
<td>% Error</td>
<td>SPBRG value</td>
<td>Actual Rate</td>
</tr>
<tr>
<td>300</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1200</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2400</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9600</td>
<td>9615</td>
<td>0.16</td>
<td>207</td>
<td>9615</td>
</tr>
<tr>
<td>10417</td>
<td>10417</td>
<td>0.00</td>
<td>191</td>
<td>10417</td>
</tr>
<tr>
<td>19.2k</td>
<td>19.23k</td>
<td>0.16</td>
<td>103</td>
<td>19.23k</td>
</tr>
<tr>
<td>57.6k</td>
<td>57.14k</td>
<td>-0.79</td>
<td>34</td>
<td>56.82k</td>
</tr>
<tr>
<td>115.2k</td>
<td>117.64k</td>
<td>2.12</td>
<td>16</td>
<td>113.64k</td>
</tr>
</tbody>
</table>
### TABLE 25-5: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

#### SYNC = 0, BRGH = 1, BRG16 = 0

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Fosc = 8.000 MHz</th>
<th>Fosc = 4.000 MHz</th>
<th>Fosc = 3.6864 MHz</th>
<th>Fosc = 1.000 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate</td>
<td>% Error</td>
<td>SPBRG value</td>
<td>Actual Rate</td>
</tr>
<tr>
<td>300</td>
<td>299.9 -0.02</td>
<td>1666</td>
<td>6666</td>
<td>300.0 0.00</td>
</tr>
<tr>
<td>1200</td>
<td>1200 -0.02</td>
<td>3332</td>
<td>1200</td>
<td>1200 -0.03</td>
</tr>
<tr>
<td>2400</td>
<td>2404 0.16</td>
<td>207</td>
<td>2404</td>
<td>2399 -0.03</td>
</tr>
<tr>
<td>9600</td>
<td>9615 0.16</td>
<td>51</td>
<td>9615</td>
<td>9615 0.16</td>
</tr>
<tr>
<td>10417</td>
<td>10417 0.00</td>
<td>47</td>
<td>10417</td>
<td>10417 0.00</td>
</tr>
<tr>
<td>19.2k</td>
<td>19.23k 0.16</td>
<td>12</td>
<td>19.23k</td>
<td>19.23k 0.16</td>
</tr>
<tr>
<td>57.6k</td>
<td>57.14k -0.79</td>
<td>34</td>
<td>56.818</td>
<td>56.818 -1.36</td>
</tr>
<tr>
<td>115.2k</td>
<td>117.6k 2.12</td>
<td>16</td>
<td>113.636</td>
<td>113.636 -1.36</td>
</tr>
</tbody>
</table>

#### SYNC = 0, BRGH = 0, BRG16 = 1

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Fosc = 32.000 MHz</th>
<th>Fosc = 20.000 MHz</th>
<th>Fosc = 18.432 MHz</th>
<th>Fosc = 11.0592 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate</td>
<td>% Error</td>
<td>SPBRG value</td>
<td>Actual Rate</td>
</tr>
<tr>
<td>300</td>
<td>300.0 0.00</td>
<td>6666</td>
<td>300.0 0.00</td>
<td>4166</td>
</tr>
<tr>
<td>1200</td>
<td>1200 -0.03</td>
<td>3332</td>
<td>1200 -0.03</td>
<td>1041</td>
</tr>
<tr>
<td>2400</td>
<td>2401 -0.04</td>
<td>832</td>
<td>2399 -0.03</td>
<td>520</td>
</tr>
<tr>
<td>9600</td>
<td>9615 0.16</td>
<td>207</td>
<td>9615 0.16</td>
<td>129</td>
</tr>
<tr>
<td>10417</td>
<td>10417 0.00</td>
<td>47</td>
<td>10417 0.00</td>
<td>119</td>
</tr>
<tr>
<td>19.2k</td>
<td>19.23k 0.16</td>
<td>103</td>
<td>19.23k 0.16</td>
<td>64</td>
</tr>
<tr>
<td>57.6k</td>
<td>57.14k -0.79</td>
<td>34</td>
<td>56.818 -1.36</td>
<td>21</td>
</tr>
<tr>
<td>115.2k</td>
<td>117.6k 2.12</td>
<td>16</td>
<td>113.636 -1.36</td>
<td>10</td>
</tr>
</tbody>
</table>

#### SYNC = 0, BRGH = 0, BRG16 = 1

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Fosc = 8.000 MHz</th>
<th>Fosc = 4.000 MHz</th>
<th>Fosc = 3.6864 MHz</th>
<th>Fosc = 1.000 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate</td>
<td>% Error</td>
<td>SPBRG value</td>
<td>Actual Rate</td>
</tr>
<tr>
<td>300</td>
<td>300.0 0.00</td>
<td>1666</td>
<td>300.0 0.00</td>
<td>1041</td>
</tr>
<tr>
<td>1200</td>
<td>1200 -0.03</td>
<td>3332</td>
<td>1200 -0.03</td>
<td>1041</td>
</tr>
<tr>
<td>2400</td>
<td>2401 -0.04</td>
<td>832</td>
<td>2399 -0.03</td>
<td>520</td>
</tr>
<tr>
<td>9600</td>
<td>9615 0.16</td>
<td>207</td>
<td>9615 0.16</td>
<td>129</td>
</tr>
<tr>
<td>10417</td>
<td>10417 0.00</td>
<td>47</td>
<td>10417 0.00</td>
<td>119</td>
</tr>
<tr>
<td>19.2k</td>
<td>19.23k 0.16</td>
<td>103</td>
<td>19.23k 0.16</td>
<td>64</td>
</tr>
<tr>
<td>57.6k</td>
<td>57.14k -0.79</td>
<td>34</td>
<td>56.818 -1.36</td>
<td>21</td>
</tr>
<tr>
<td>115.2k</td>
<td>117.6k 2.12</td>
<td>16</td>
<td>113.636 -1.36</td>
<td>10</td>
</tr>
<tr>
<td>BAUD RATE</td>
<td>Fosc = 32.000 MHz</td>
<td>Fosc = 20.000 MHz</td>
<td>Fosc = 18.432 MHz</td>
<td>Fosc = 11.0592 MHz</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
<td>------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td>Actual Rate</td>
<td>% Error</td>
<td>SPBRG value (decimal)</td>
<td>Actual Rate</td>
</tr>
<tr>
<td>300</td>
<td>300.0</td>
<td>0.00</td>
<td>26666</td>
<td>300.0</td>
</tr>
<tr>
<td>1200</td>
<td>1200.0</td>
<td>0.00</td>
<td>6666</td>
<td>1200.0</td>
</tr>
<tr>
<td>2400</td>
<td>2400.0</td>
<td>0.01</td>
<td>3332</td>
<td>2400.0</td>
</tr>
<tr>
<td>9600</td>
<td>9604.0</td>
<td>0.04</td>
<td>832</td>
<td>9597.0</td>
</tr>
<tr>
<td>10417</td>
<td>10417.0</td>
<td>0.00</td>
<td>767</td>
<td>10417.0</td>
</tr>
<tr>
<td>19.2k</td>
<td>19.18k</td>
<td>-0.08</td>
<td>416</td>
<td>19.23k</td>
</tr>
<tr>
<td>57.6k</td>
<td>57.55k</td>
<td>-0.08</td>
<td>138</td>
<td>57.47k</td>
</tr>
<tr>
<td>115.2k</td>
<td>115.9k</td>
<td>0.64</td>
<td>68</td>
<td>116.3k</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BAUD RATE</th>
<th>Fosc = 8.000 MHz</th>
<th>Fosc = 4.000 MHz</th>
<th>Fosc = 3.6864 MHz</th>
<th>Fosc = 1.000 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate</td>
<td>% Error</td>
<td>SPBRG value (decimal)</td>
<td>Actual Rate</td>
</tr>
<tr>
<td>300</td>
<td>300.0</td>
<td>0.00</td>
<td>6666</td>
<td>300.0</td>
</tr>
<tr>
<td>1200</td>
<td>1200.0</td>
<td>-0.02</td>
<td>1666</td>
<td>1200.0</td>
</tr>
<tr>
<td>2400</td>
<td>2401.0</td>
<td>0.04</td>
<td>832</td>
<td>2398.0</td>
</tr>
<tr>
<td>9600</td>
<td>9615.0</td>
<td>0.16</td>
<td>207</td>
<td>9615.0</td>
</tr>
<tr>
<td>10417</td>
<td>10417.0</td>
<td>0.00</td>
<td>191</td>
<td>10417.0</td>
</tr>
<tr>
<td>19.2k</td>
<td>19.23k</td>
<td>0.16</td>
<td>103</td>
<td>19.23k</td>
</tr>
<tr>
<td>57.6k</td>
<td>57.14k</td>
<td>-0.79</td>
<td>34</td>
<td>58.82k</td>
</tr>
<tr>
<td>115.2k</td>
<td>117.6k</td>
<td>2.12</td>
<td>16</td>
<td>111.1k</td>
</tr>
</tbody>
</table>
25.3.1 AUTO-BAUD DETECT

The EUSART module supports automatic detection and calibration of the baud rate.

In the Auto-Baud Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. The Baud Rate Generator is used to time the period of a received 55h (ASCII "U") which is the Sync character for the LIN bus. The unique feature of this character is that it has five rising edges including the Stop bit edge.

Setting the ABDEN bit of the BAUDCON register starts the auto-baud calibration sequence (Figure 25-6). While the ABD sequence takes place, the EUSART state machine is held in Idle. On the first rising edge of the receive line, after the Start bit, the SPBRG begins counting up using the BRG counter clock as shown in Table 25-6. The fifth rising edge will occur on the RX pin at the end of the eighth bit period. At that time, an accumulated value totaling the proper BRG period is left in the SPBRGH, SPBRGL register pair, the ABDEN bit is automatically cleared and the RCIF interrupt flag is set. The value in the RCREG needs to be read to clear the RCIF interrupt. RCREG content should be discarded. When calibrating for modes that do not use the SPBRGH register, the user can verify that the SPBRGL register did not overflow by checking for 00h in the SPBRGH register.

The BRG auto-baud clock is determined by the BRG16 and BRGH bits as shown in Table 25-6. During ABD, both the SPBRGH and SPBRGL registers are used as a 16-bit counter, independent of the BRG16 bit setting. While calibrating the baud rate period, the SPBRGH and SPBRGL registers are clocked at 1/8th the BRG base clock rate. The resulting byte measurement is the average bit time when clocked at full speed.

**TABLE 25-6: BRG COUNTER CLOCK RATES**

<table>
<thead>
<tr>
<th>BRG16</th>
<th>BRGH</th>
<th>BRG Base Clock</th>
<th>BRG ABD Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Fosc/64</td>
<td>Fosc/512</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Fosc/16</td>
<td>Fosc/128</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Fosc/16</td>
<td>Fosc/128</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Fosc/4</td>
<td>Fosc/32</td>
</tr>
</tbody>
</table>

**Note:** During the ABD sequence, SPBRGH and SPBRGL registers are both used as a 16-bit counter, independent of BRG16 setting.

**FIGURE 25-6: AUTOMATIC BAUD RATE CALIBRATION**

<table>
<thead>
<tr>
<th>BRG Value</th>
<th>0000h</th>
<th>001Ch</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX pin</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRG Clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABDEN bit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCIDL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCIF bit (Interrupt)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read RCREG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBRGL</td>
<td>XXh</td>
<td></td>
</tr>
<tr>
<td>SPBRGH</td>
<td></td>
<td>00h</td>
</tr>
</tbody>
</table>

**Note 1:** The ABD sequence requires the EUSART module to be configured in Asynchronous mode.
25.3.2 AUTO-BAUD OVERFLOW

During the course of automatic baud detection, the ABDOVF bit of the BAUDCON register will be set if the baud rate counter overflows before the fifth rising edge is detected on the RX pin. The ABDOVF bit indicates that the counter has exceeded the maximum count that can fit in the 16 bits of the SPBRGH:SPBRGL register pair. After the ABDOVF has been set, the counter continues to count until the fifth rising edge is detected on the RX pin. Upon detecting the fifth RX edge, the hardware will set the RCIF interrupt flag and clear the ABDEN bit of the BAUDCON register. The RCIF flag can be subsequently cleared by reading the RCREG register. The ABDOVF flag of the BAUDCON register can be cleared by software directly.

To terminate the auto-baud process before the RCIF flag is set, clear the ABDEN bit then clear the ABDOVF bit of the BAUDCON register. The ABDOVF bit will remain set if the ABDEN bit is not cleared first.

25.3.3 AUTO-WAKE-UP ON BREAK

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the Baud Rate Generator is inactive and a proper character reception cannot be performed. The Auto-Wake-up feature allows the controller to wake-up due to activity on the RX/DT line. This feature is available only in Asynchronous mode.

The Auto-Wake-up feature is enabled by setting the WUE bit of the BAUDCON register. Once set, the normal receive sequence on RX/DT is disabled, and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a Sync Break or a wake-up signal character for the LIN protocol.)

The EUSART module generates an RCIF interrupt coincident with the wake-up event. The interrupt is generated synchronously to the Q clocks in normal CPU operating modes (Figure 25-7), and asynchronously if the device is in Sleep mode (Figure 25-8). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared by the low-to-high transition on the RX line at the end of the Break. This signals to the user that the Break event is over. At this point, the EUSART module is in Idle mode waiting to receive the next character.

25.3.3.1 Special Considerations

Break Character

To avoid character errors or character fragments during a wake-up event, the wake-up character must be all zeros.

When the wake-up is enabled the function works independent of the low time on the data stream. If the WUE bit is set and a valid non-zero character is received, the low time from the Start bit to the first rising edge will be interpreted as the wake-up event. The remaining bits in the character will be received as a fragmented character and subsequent characters can result in framing or overrun errors.

Therefore, the initial character in the transmission must be all ‘0’s. This must be 10 or more bit times, 13-bit times recommended for LIN bus, or any number of bit times for standard RS-232 devices.

Oscillator Start-up Time

Oscillator start-up time must be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL mode). The Sync Break (or wake-up signal) character must be of sufficient length, and be followed by a sufficient interval, to allow enough time for the selected oscillator to start and provide proper initialization of the EUSART.

WUE Bit

The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared in hardware by a rising edge on RX/DT. The interrupt condition is then cleared in software by reading the RCREG register and discarding its contents.

To ensure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process before setting the WUE bit. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.
FIGURE 25-7: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION

Note 1: The EUSART remains in Idle while the WUE bit is set.

FIGURE 25-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP

Note 1: If the wake-up event requires long oscillator warm-up time, the automatic clearing of the WUE bit can occur while the stoposc signal is still active. This sequence should not depend on the presence of Q clocks.

2: The EUSART remains in Idle while the WUE bit is set.
25.3.4 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. A Break character consists of a Start bit, followed by 12 '0' bits and a Stop bit.

To send a Break character, set the SENDB and TXEN bits of the TXSTA register. The Break character transmission is then initiated by a write to the TXREG. The value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

The TRMT bit of the TXSTA register indicates when the transmit operation is active or idle, just as it does during normal transmission. See Figure 25-9 for the timing of the Break character sequence.

25.3.4.1 Break and Sync Transmit Sequence

The following sequence will start a message frame header made up of a Break, followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to enable the Break sequence.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware and the Sync character is then transmitted.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

25.3.5 RECEIVING A BREAK CHARACTER

The Enhanced EUSART module can receive a Break character in two ways.

The first method to detect a Break character uses the FERR bit of the RCSTA register and the Received data as indicated by RCREG. The Baud Rate Generator is assumed to have been initialized to the expected baud rate.

A Break character has been received when:
- RCIF bit is set
- FERR bit is set
- RCREG = 00h

The second method uses the Auto-Wake-up feature described in Section 25.3.3 “Auto-Wake-up on Break”. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt, and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Detect feature. For both methods, the user can set the ABDEN bit of the BAUDCON register before placing the EUSART in Sleep mode.

FIGURE 25-9: SEND BREAK CHARACTER SEQUENCE
25.4 EUSART Synchronous Mode

Synchronous serial communications are typically used in systems with a single master and one or more slaves. The master device contains the necessary circuitry for baud rate generation and supplies the clock for all devices in the system. Slave devices can take advantage of the master clock by eliminating the internal clock generation circuitry.

There are two signal lines in Synchronous mode: a bidirectional data line and a clock line. Slaves use the external clock supplied by the master to shift the serial data into and out of their respective receive and transmit shift registers. Since the data line is bidirectional, synchronous operation is half-duplex only. Half-duplex refers to the fact that master and slave devices can receive and transmit data but not both simultaneously. The EUSART can operate as either a master or slave device.

Start and Stop bits are not used in synchronous transmissions.

25.4.1 SYNCHRONOUS MASTER MODE

The following bits are used to configure the EUSART for Synchronous Master operation:

- \( \text{SYNC} = 1 \)
- \( \text{CSRC} = 1 \)
- \( \text{SREN} = 0 \) (for transmit); \( \text{SREN} = 1 \) (for receive)
- \( \text{CREN} = 0 \) (for transmit); \( \text{CREN} = 1 \) (for receive)
- \( \text{SPEN} = 1 \)

Setting the \( \text{SYNC} \) bit of the TXSTA register configures the device for synchronous operation. Setting the \( \text{CSRC} \) bit of the TXSTA register configures the device as a master. Clearing the \( \text{SREN} \) and \( \text{CREN} \) bits of the RCSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the \( \text{SPEN} \) bit of the RCSTA register enables the EUSART.

25.4.1.1 Master Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a master transmits the clock on the TX/CK line. The TX/CK pin output driver is automatically enabled when the EUSART is configured for synchronous transmit or receive operation. Serial data bits change on the leading edge to ensure they are valid at the trailing edge of each clock. One clock cycle is generated for each data bit. Only as many clock cycles are generated as there are data bits.

25.4.1.2 Clock Polarity

A clock polarity option is provided for Microwire compatibility. Clock polarity is selected with the \( \text{SCKP} \) bit of the BAUDCON register. Setting the \( \text{SCKP} \) bit sets the clock Idle state as high. When the \( \text{SCKP} \) bit is set, the data changes on the falling edge of each clock. Clearing the \( \text{SCKP} \) bit sets the Idle state as low. When the \( \text{SCKP} \) bit is cleared, the data changes on the rising edge of each clock.

25.4.1.3 Synchronous Master Transmission

Data is transferred out of the device on the RX/DT pin. The RX/DT and TX/CK pin output drivers are automatically enabled when the EUSART is configured for synchronous master transmit operation.

A transmission is initiated by writing a character to the TXREG register. If the TSR still contains all or part of a previous character the new character data is held in the TXREG until the last bit of the previous character has been transmitted. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXREG is immediately transferred to the TSR. The transmission of the character commences immediately following the transfer of the data to the TSR from the TXREG.

Each data bit changes on the leading edge of the master clock and remains valid until the subsequent leading clock edge.

Note: The TSR register is not mapped in data memory, so it is not available to the user.

25.4.1.4 Synchronous Master Transmission

Set-up:

1. Initialize the SPBRGH, SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see Section 25.3 “EUSART Baud Rate Generator (BRG)”).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Disable Receive mode by clearing bits SREN and CREN.
4. Enable Transmit mode by setting the TXEN bit.
5. If 9-bit transmission is desired, set the TX9 bit.
6. If interrupts are desired, set the TXIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
7. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
8. Start transmission by loading data to the TXREG register.
FIGURE 25-10: SYNCHRONOUS TRANSMISSION

<table>
<thead>
<tr>
<th>RX/DT pin</th>
<th>TX/CK pin (SCKP = 0)</th>
<th>TX/CK pin (SCKP = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1</td>
<td>Write to TXREG reg</td>
<td>Write Word 1</td>
</tr>
<tr>
<td>Word 2</td>
<td>Write Word 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TXIF bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRMT bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TXEN bit</td>
<td>Sync Master mode, SPBRG = 0, continuous transmission of two 8-bit words.</td>
</tr>
</tbody>
</table>

FIGURE 25-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

<table>
<thead>
<tr>
<th>RX/DT pin</th>
<th>TX/CK pin</th>
<th>Write to TXREG reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word 1</td>
<td></td>
<td>Write Word 1</td>
</tr>
<tr>
<td>Word 2</td>
<td></td>
<td>Write Word 2</td>
</tr>
<tr>
<td></td>
<td>TXIF bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRMT bit</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TXEN bit</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 25-7: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

<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>BAUDCON</td>
<td>ABDOVF</td>
<td>RCIDL</td>
<td>—</td>
<td>SCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>298</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>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>297</td>
</tr>
<tr>
<td>SPBRGL</td>
<td>Baud Rate Generator Data Register Low</td>
<td>299*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBRGH</td>
<td>Baud Rate Generator Data Register High</td>
<td>299*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXREG</td>
<td>EUSART Transmit Data Register</td>
<td>289*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>296</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented read as ‘0’. Shaded cells are not used for Synchronous Master Transmission.
* Page provides register information.
25.4.1.5 Synchronous Master Reception

Data is received at the RX/DT pin. The RX/DT pin output driver is automatically disabled when the EUSART is configured for synchronous master receive operation.

In Synchronous mode, reception is enabled by setting either the Single Receive Enable bit (SREN of the RCSTA register) or the Continuous Receive Enable bit (CREN of the RCSTA register).

When SREN is set and CREN is clear, only as many clock cycles are generated as there are data bits in a single character. The SREN bit is automatically cleared at the completion of one character. When CREN is set, clocks are continuously generated until CREN is cleared. If CREN is cleared in the middle of a character the CK clock stops immediately and the partial character is discarded. If SREN and CREN are both set, then SREN is cleared at the completion of the first character and CREN takes precedence.

To initiate reception, set either SREN or CREN. Data is sampled at the RX/DT pin on the trailing edge of the TX/CK clock pin and is shifted into the Receive Shift Register (RSR). When a complete character is received into the RSR, the RCIF bit is set and the character is automatically transferred to the two character receive FIFO. The Least Significant eight bits of the top character in the receive FIFO are available in RCREG. The RCIF bit remains set as long as there are unread characters in the receive FIFO.

25.4.1.6 Slave Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a slave receives the clock on the TX/CK line. The TX/CK pin output driver is automatically disabled when the device is configured for synchronous slave transmit or receive operation. Serial data bits change on the leading edge to ensure they are valid at the trailing edge of each clock. One data bit is transferred for each clock cycle. Only as many clock cycles should be received as there are data bits.

25.4.1.7 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before RCREG is read to access the FIFO. When this happens the OERR bit of the RCSTA register is set. Previous data in the FIFO will not be overwritten. The two characters in the FIFO buffer can be read, however, no additional characters will be received until the error is cleared. The OERR bit can only be cleared by clearing the overrun condition. If the overrun error occurred when the SREN bit is set and CREN is clear then the error is cleared by reading RCREG. If the overrun occurred when the CREN bit is set then the error condition is cleared by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

25.4.1.8 Receiving 9-bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCSTA register is set the EUSART will shift 9-bits into the RSR for each character received. The RX9D bit of the RCSTA register is the ninth, and Most Significant, data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the 8 Least Significant bits from the RCREG.

25.4.1.9 Synchronous Master Reception

Set-up:

1. Initialize the SPBRGH, SPBRGL register pair for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Ensure bits CREN and SREN are clear.
4. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. If 9-bit reception is desired, set bit RX9.
6. Start reception by setting the SREN bit or for continuous reception, set the CREN bit.
7. Interrupt flag bit RCIF will be set when reception of a character is complete. An interrupt will be generated if the enable bit RCIE was set.
8. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREG register.
10. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.
FIGURE 25-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)

TABLE 25-8: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

<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>BAUDCON</td>
<td>ABDVF</td>
<td>RCIDL</td>
<td>—</td>
<td>SCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>298</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>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>90</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>94</td>
</tr>
<tr>
<td>RCREG</td>
<td>EUSART Receive Data Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>292*</td>
</tr>
<tr>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>297</td>
</tr>
<tr>
<td>SPBRG</td>
<td>Baud Rate Generator Data Register Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>299*</td>
</tr>
<tr>
<td>SPBRGH</td>
<td>Baud Rate Generator Data Register High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>299*</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
<tr>
<td>TXSTA</td>
<td>CSRRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>296</td>
</tr>
</tbody>
</table>

Legend:  — = unimplemented read as '0'. Shaded cells are not used for Synchronous Master Reception.

Note: Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0.
25.4.2 SYNCHRONOUS SLAVE MODE

The following bits are used to configure the EUSART for synchronous slave operation:

- SYNC = 1
- CSRC = 0
- SREN = 0 (for transmit); SREN = 1 (for receive)
- CREN = 0 (for transmit); CREN = 1 (for receive)
- SPEN = 1

Setting the SYNC bit of the TXSTA register configures the device for synchronous operation. Clearing the CSRC bit of the TXSTA register configures the device as a slave. Clearing the SREN and CREN bits of the RCSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCSTA register enables the EUSART.

25.4.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see Section 25.4.1.3 “Synchronous Master Transmission”), except in the case of the Sleep mode.

If two words are written to the TXREG and then the SLEEP instruction is executed, the following will occur:

1. The first character will immediately transfer to the TSR register and transmit.
2. The second word will remain in TXREG register.
3. The TXIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXREG register will transfer the second character to the TSR and the TXIF bit will now be set.
5. If the PEIE and TXIE bits are set, the interrupt will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will call the Interrupt Service Routine.

25.4.2.2 Synchronous Slave Transmission

Set-up:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the CREN and SREN bits.
3. If interrupts are desired, set the TXIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
4. If 9-bit transmission is desired, set the TX9 bit.
5. Enable transmission by setting the TXEN bit.
6. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
7. Start transmission by writing the Least Significant 8 bits to the TXREG register.

TABLE 25-9: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

<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>BAUDCON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SCKP</td>
<td>—</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABREN 298</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>PEIE</td>
<td>TMROIE</td>
<td>INTE</td>
<td>IOCIE</td>
<td>TMR0IF</td>
<td>INTF</td>
<td>IOCIF 89</td>
<td></td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE 90</td>
<td></td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF 94</td>
<td></td>
</tr>
<tr>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D 297</td>
<td></td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0 126</td>
<td></td>
</tr>
<tr>
<td>TXREG</td>
<td>EUSART Transmit Data Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SSEND</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D 296</td>
<td></td>
</tr>
</tbody>
</table>

Legend: — = unimplemented read as ‘0’. Shaded cells are not used for synchronous slave transmission.

* Page provides register information.
25.4.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (Section 25.4.1.5 “Synchronous Master Reception”), with the following exceptions:

- Sleep
- CREN bit is always set, therefore the receiver is never idle
- SREN bit, which is a “don’t care” in Slave mode

A character may be received while in Sleep mode by setting the CREN bit prior to entering Sleep. Once the word is received, the RSR register will transfer the data to the RCREG register. If the RCIE enable bit is set, the interrupt generated will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will branch to the interrupt vector.

25.4.2.4 Synchronous Slave Reception

Set-up:
1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
3. If 9-bit reception is desired, set the RX9 bit.
4. Set the CREN bit to enable reception.
5. The RCIF bit will be set when reception is complete. An interrupt will be generated if the RCIE bit was set.
6. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCSTA register.
7. Retrieve the 8 Least Significant bits from the receive FIFO by reading the RCREG register.
8. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCSTA register or by clearing the SPEN bit which resets the EUSART.

TABLE 25-10: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

<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>BAUDCON</td>
<td>ABDOVF</td>
<td>RCIDL</td>
<td>—</td>
<td>SCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>298</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>89</td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1E</td>
<td>90</td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1F</td>
<td>94</td>
</tr>
<tr>
<td>RCREG</td>
<td>EUSART</td>
<td>Receive Data Register</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>292*</td>
</tr>
<tr>
<td>RCSTA</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADDEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>297</td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td>126</td>
</tr>
<tr>
<td>TXSTA</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>296</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented read as ‘0’. Shaded cells are not used for synchronous slave reception.
* Page provides register information.
# 25.5 EUSART Operation During Sleep

The EUSART will remain active during Sleep only in the Synchronous Slave mode. All other modes require the system clock and therefore cannot generate the necessary signals to run the Transmit or Receive Shift registers during Sleep.

Synchronous Slave mode uses an externally generated clock to run the Transmit and Receive Shift registers.

## 25.5.1 SYNCHRONOUS RECEIVE DURING SLEEP

To receive during Sleep, all the following conditions must be met before entering Sleep mode:

- RCSTA and TXSTA Control registers must be configured for synchronous slave reception (see Section 25.4.2.4 “Synchronous Slave Reception Set-up:").
- If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
- The RCIF interrupt flag must be cleared by reading RCREG to unload any pending characters in the receive buffer.

Upon entering Sleep mode, the device will be ready to accept data and clocks on the RX/DT pins when the data word in the TSR has been completely clocked in by the external device, the RCIF interrupt flag bit of the PIR1 register will be set. Thereby, waking the processor from Sleep.

Upon waking from Sleep, the instruction following the SLEEP instruction will be executed. If the GIE global interrupt enable bit of the INTCON register is also set, then the Interrupt Service Routine at address 004h will be called.

## 25.5.2 SYNCHRONOUS TRANSMIT DURING SLEEP

To transmit during Sleep, all the following conditions must be met before entering Sleep mode:

- RCSTA and TXSTA Control registers must be configured for synchronous slave transmission (see Section 25.4.2.2 “Synchronous Slave Transmission Set-up:").
- If interrupts are desired, set the TXIE bit of the PIE1 register and the PEIE bit of the INTCON register.
- Interrupt enable bits TXIE of the PIE1 register and PEIE of the INTCON register must set.

Upon entering Sleep mode, the device will be ready to accept clocks on TX/CK pin and transmit data on the RX/DT pin. When the data word in the TSREG has been completely clocked out by the external device, the pending byte in the TXREG will transfer to the TSR and the TXIF flag will be set. Thereby, waking the processor from Sleep. At this point, the TXREG is available to accept another character for transmission, which will clear the TXIF flag.

Upon waking from Sleep, the instruction following the SLEEP instruction will be executed. If the Global Interrupt Enable (GIE) bit is also set then the Interrupt Service Routine at address 004h will be called.
26.0 CAPACITIVE SENSING MODULE

The capacitive sensing module allows for an interaction with an end user without a mechanical interface. In a typical application, the capacitive sensing module is attached to a pad on a Printed Circuit Board (PCB), which is electrically isolated from the end user. When the end user places their finger over the PCB pad, a capacitive load is added, causing a frequency shift in the capacitive sensing module. The capacitive sensing module requires software and at least one timer resource to determine the change in frequency. Key features of this module include:

- Analog MUX for monitoring multiple inputs
- Capacitive sensing oscillator
- Multiple timer resources
- Software control
- Operation during Sleep

FIGURE 26-1: CAPACITIVE SENSING BLOCK DIAGRAM

Note 1: If CPSON = 0, disabling capacitive sensing, no channel is selected.
26.1 Analog MUX

The capacitive sensing module can monitor up to 12 inputs. The capacitive sensing inputs are defined as CPS<11:0>. To determine if a frequency change has occurred the user must:

- Select the appropriate CPS pin by setting the CPSCH<3:0> bits of the CPSCON1 register
- Set the corresponding ANSEL bit
- Set the corresponding TRIS bit
- Run the software algorithm

Selection of the CPSx pin while the module is enabled will cause the capacitive sensing oscillator to be on the CPSx pin. Failure to set the corresponding ANSEL and TRIS bits can cause the capacitive sensing oscillator to stop, leading to false frequency readings.

26.2 Capacitive Sensing Oscillator

The capacitive sensing oscillator consists of a constant current source and a constant current sink, to produce a triangle waveform. The CPSOUT bit of the CPSCON0 register shows the status of the capacitive sensing oscillator, whether it is a sinking or sourcing current. The oscillator is designed to drive a capacitive load (single PCB pad) and at the same time, be a clock source to either Timer0 or Timer1. The oscillator has three different current settings as defined by CPSRNG<1:0> of the CPSCON0 register. The different current settings for the oscillator serve two purposes:

- Maximize the number of counts in a timer for a fixed time base
- Maximize the count differential in the timer during a change in frequency

26.3 Timer resources

To measure the change in frequency of the capacitive sensing oscillator, a fixed time base is required. For the period of the fixed time base, the capacitive sensing oscillator is used to clock either Timer0 or Timer1. The frequency of the capacitive sensing oscillator is equal to the number of counts in the timer divided by the period of the fixed time base.

26.4 Fixed Time Base

To measure the frequency of the capacitive sensing oscillator, a fixed time base is required. Any timer resource or software loop can be used to establish the fixed time base. It is up to the end user to determine the method in which the fixed time base is generated.

| Note: | The fixed time base can not be generated by the timer resource that the capacitive sensing oscillator is clocking. |

26.4.1 TIMER0

To select Timer0 as the timer resource for the capacitive sensing module:

- Set the T0XCS bit of the CPSCON0 register
- Clear the TMR0CS bit of the OPTION register

When Timer0 is chosen as the timer resource, the capacitive sensing oscillator will be the clock source for Timer0. Refer to Section 19.0 “Timer0 Module” for additional information.

26.4.2 TIMER1

To select Timer1 as the timer resource for the capacitive sensing module, set the TMR1CS<1:0> of the T1CON register to '11'. When Timer1 is chosen as the timer resource, the capacitive sensing oscillator will be the clock source for Timer1. Because the Timer1 module has a gate control, developing a time base for the frequency measurement can be simplified by using the Timer0 overflow flag.

It is recommend that the Timer0 overflow flag, in conjunction with the Toggle mode of the Timer1 Gate, be used to develop the fixed time base required by the software portion of the capacitive sensing module. Refer to Section 20.6.3 “Timer1 Gate Toggle Mode” for additional information.

| TABLE 26-1: TIMER1 ENABLE FUNCTION |
|-------------------------------|----------------------|----------------------|
| TMR1ON | TMR1GE | Timer1 Operation |
| 0 | 0 | Off |
| 0 | 1 | Off |
| 1 | 0 | On |
| 1 | 1 | Count Enabled by input |
26.5 Software Control

The software portion of the capacitive sensing module is required to determine the change in frequency of the capacitive sensing oscillator. This is accomplished by the following:

- Setting a fixed time base to acquire counts on Timer0 or Timer1
- Establishing the nominal frequency for the capacitive sensing oscillator
- Establishing the reduced frequency for the capacitive sensing oscillator due to an additional capacitive load
- Set the frequency threshold

26.5.1 NOMINAL FREQUENCY
(NO CAPACITIVE LOAD)

To determine the nominal frequency of the capacitive sensing oscillator:

- Remove any extra capacitive load on the selected CPSx pin
- At the start of the fixed time base, clear the timer resource
- At the end of the fixed time base save the value in the timer resource

The value of the timer resource is the number of oscillations of the capacitive sensing oscillator for the given time base. The frequency of the capacitive sensing oscillator is equal to the number of counts on in the timer divided by the period of the fixed time base.

26.5.2 REDUCED FREQUENCY
(ADDITIONAL CAPACITIVE LOAD)

The extra capacitive load will cause the frequency of the capacitive sensing oscillator to decrease. To determine the reduced frequency of the capacitive sensing oscillator:

- Add a typical capacitive load on the selected CPSx pin
- Use the same fixed time base as the nominal frequency measurement
- At the start of the fixed time base, clear the timer resource
- At the end of the fixed time base save the value in the timer resource

The value of the timer resource is the number of oscillations of the capacitive sensing oscillator with an additional capacitive load. The frequency of the capacitive sensing oscillator is equal to the number of counts on in the timer divided by the period of the fixed time base. This frequency should be less than the value obtained during the nominal frequency measurement.

26.5.3 FREQUENCY THRESHOLD

The frequency threshold should be placed midway between the value of nominal frequency and the reduced frequency of the capacitive sensing oscillator. Refer to Application Note AN1103, "Software Handling for Capacitive Sensing" (DS01103) for more detailed information on the software required for capacitive sensing module.

Note: For more information on general capacitive sensing refer to Application Notes:

- AN1101, "Introduction to Capacitive Sensing" (DS01101)
- AN1102, "Layout and Physical Design Guidelines for Capacitive Sensing" (DS01102)

26.6 Operation during Sleep

The capacitive sensing oscillator will continue to run as long as the module is enabled, independent of the part being in Sleep. In order for the software to determine if a frequency change has occurred, the part must be awake. However, the part does not have to be awake when the timer resource is acquiring counts.

Note: Timer0 does not operate when in Sleep, and therefore cannot be used for capacitive sense measurements in Sleep.
## REGISTER 26-1: CPSCON0: CAPACITIVE SENSING CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td><strong>CPSON</strong>: Capacitive Sensing Module Enable bit</td>
<td>1 = Enable, 0 = Disable</td>
</tr>
<tr>
<td>bit 6-4</td>
<td>Unimplemented</td>
<td>Read as '0'</td>
</tr>
<tr>
<td>bit 3-2</td>
<td><strong>CPSRNG&lt;1:0&gt;</strong>: Capacitive Sensing Oscillator Range bits</td>
<td>00 = Off, 01 = Low, 10 = Medium, 11 = High</td>
</tr>
<tr>
<td>bit 1</td>
<td><strong>CPSOUT</strong>: Capacitive Sensing Oscillator Status bit</td>
<td>1 = Sourcing, 0 = Sinking</td>
</tr>
<tr>
<td>bit 0</td>
<td><strong>T0XCS</strong>: Timer0 External Clock Source Select bit</td>
<td>1 = Capacitive sensing oscillator, 0 = TOCKI pin</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

---

- **CPSON**: Capacitive Sensing Module Enable bit
  - 1 = Capacitive sensing module is enabled
  - 0 = Capacitive sensing module is disabled

- **CPSRNG<1:0>**: Capacitive Sensing Oscillator Range bits
  - 00 = Oscillator is off
  - 01 = Oscillator is in low range. Charge/discharge current is nominally 0.1 μA.
  - 10 = Oscillator is in medium range. Charge/discharge current is nominally 1.2 μA.
  - 11 = Oscillator is in high range. Charge/discharge current is nominally 18 μA.

- **CPSOUT**: Capacitive Sensing Oscillator Status bit
  - 1 = Oscillator is sourcing current (Current flowing out the pin)
  - 0 = Oscillator is sinking current (Current flowing into the pin)

- **T0XCS**: Timer0 External Clock Source Select bit
  - If **TMR0CS = 1**
    - The T0XCS bit controls which clock external to the core/Timer0 module supplies Timer0:
      - 1 = Timer0 clock source is the capacitive sensing oscillator
      - 0 = Timer0 clock source is the TOCKI pin
  - If **TMR0CS = 0**
    - Timer0 clock source is controlled by the core/Timer0 module and is Fosc/4
TABLE 26-2: SUMMARY OF REGISTERS ASSOCIATED WITH CAPACITIVE SENSING

<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>—</td>
<td>—</td>
<td>—</td>
<td>ANSA4</td>
<td>ANSA3</td>
<td>ANSA2</td>
<td>ANSA1</td>
</tr>
<tr>
<td>ANSELB</td>
<td>ANSB7</td>
<td>ANSB6</td>
<td>ANSB5</td>
<td>ANSB4</td>
<td>ANSB3</td>
<td>ANSB2</td>
<td>ANSB1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPSCON0</td>
<td>CPSON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPSRNG1</td>
<td>CPSRNG0</td>
<td>CPSOUT</td>
<td></td>
</tr>
<tr>
<td>CPSCON1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CPSCH3</td>
<td>CPSCH2</td>
<td>CPSCH1</td>
<td>CPSCH0</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></td>
</tr>
<tr>
<td>OPTION_REG</td>
<td>WPUEN</td>
<td>INTEDG</td>
<td>TMR0CS</td>
<td>TMR0SE</td>
<td>PSA</td>
<td>PS2</td>
<td>PS1</td>
<td>PS0</td>
<td></td>
</tr>
<tr>
<td>PIE1</td>
<td>TMR1GIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td></td>
</tr>
<tr>
<td>PIR1</td>
<td>TMR1GIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td></td>
</tr>
<tr>
<td>T1CON</td>
<td>TMR1CS1</td>
<td>TMR1CS0</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>T1SYNC</td>
<td>—</td>
<td>TMR1ON</td>
<td></td>
</tr>
<tr>
<td>TxCON</td>
<td>—</td>
<td>TxOUTPS3</td>
<td>TxOUTPS2</td>
<td>TxOUTPS1</td>
<td>TxOUTPS0</td>
<td>TMRxON</td>
<td>TxCKPS1</td>
<td>TxCKPS0</td>
<td></td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td></td>
</tr>
<tr>
<td>TRISB</td>
<td>TRISB7</td>
<td>TRISB6</td>
<td>TRISB5</td>
<td>TRISB4</td>
<td>TRISB3</td>
<td>TRISB2</td>
<td>TRISB1</td>
<td>TRISB0</td>
<td></td>
</tr>
</tbody>
</table>

Legend: — = Unimplemented locations, read as ‘0’. Shaded cells are not used by the capacitive sensing module.
27.0 IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

In-Circuit Serial 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 “PIC16F/LF1826/27 Memory Programming Specification” (DS41390).

27.1 High-voltage Programming Mode

The device is placed into high-voltage Program/Verify mode by holding the ICSPCLK and ICSPDAT pins low then raising the voltage on MCLR/VPP to Vih.

27.2 Low-Voltage Programming Mode

The Low-Voltage Programming mode allows the PIC16F/LF1826/27 devices to be programmed using VDD only, without high voltage. When the LVP bit of the Configuration Word 2 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 ICSP Program/Verify modes 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.

Note: The ICD 2 produces a VPP voltage greater than the maximum VPP specification of the PIC16F/LF1826/27. When using this programmer, an external circuit is required to keep the VPP voltage within the device specifications.

FIGURE 27-1: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING

External Programming Signals

VDD
VPP
VSS
Data
Clock

Device to be Programmed

VDD
MCLR/VPP
VSS
ICSPDAT
ICSPCLK

10k

To Normal Connections

* Isolation devices (as required).
28.0 INSTRUCTION SET SUMMARY

Each PIC16 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 28-3 lists the instructions recognized by the MPASM™ assembler.

All instructions are executed within a single instruction cycle, with the following exceptions, which may take two or three cycles:

- Subroutine takes two cycles (CALL, CALLW)
- Returns from interrupts or subroutines take two cycles (RETURN, RETLW, RETFIE)
- Program branching takes two cycles (GOTO, BRA, BRW, BTFSS, BTFSC, DECFSZ, 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.

28.1 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>
<thead>
<tr>
<th>TABLE 28-1: OPCODE FIELD DESCRIPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>f</td>
</tr>
<tr>
<td>W</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>k</td>
</tr>
<tr>
<td>x</td>
</tr>
<tr>
<td>d</td>
</tr>
<tr>
<td>n</td>
</tr>
<tr>
<td>mm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 28-2: ABBREVIATION DESCRIPTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>PC</td>
</tr>
<tr>
<td>TO</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>DC</td>
</tr>
<tr>
<td>Z</td>
</tr>
<tr>
<td>PD</td>
</tr>
</tbody>
</table>
FIGURE 28-1: GENERAL FORMAT FOR INSTRUCTIONS

<table>
<thead>
<tr>
<th>Byte-oriented file register operations</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13 8 7 6 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPCODE</td>
<td>d</td>
<td>f (FILE #)</td>
</tr>
<tr>
<td>d = 0 for destination W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>d = 1 for destination f</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f = 7-bit file register address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit-oriented file register operations</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13 10 9 7 6 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPCODE</td>
<td>b (BIT #)</td>
<td>f (FILE #)</td>
</tr>
<tr>
<td>b = 3-bit bit address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f = 7-bit file register address</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Literal and control operations</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 8 7 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
</tr>
<tr>
<td>k = 8-bit immediate value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CALL and GOTO instructions only</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13 11 10 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
</tr>
<tr>
<td>k = 11-bit immediate value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MOVLP instruction only</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13 7 6 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
</tr>
<tr>
<td>k = 7-bit immediate value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MOVLB instruction only</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13 5 4 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
</tr>
<tr>
<td>k = 5-bit immediate value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BRA instruction only</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13 9 8 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
</tr>
<tr>
<td>k = 9-bit immediate value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FSR Offset instructions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13 7 6 5 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPCODE</td>
<td>n</td>
<td>k (literal)</td>
</tr>
<tr>
<td>n = appropriate FSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k = 6-bit immediate value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FSR Increment instructions</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13 3 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPCODE</td>
<td>n m (mode)</td>
<td></td>
</tr>
<tr>
<td>n = appropriate FSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 2-bit mode value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OPCODE only</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>13 0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 28-3: PIC16F/LF1826/27 ENHANCED 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 0111 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>CLRF f</td>
<td>Clear f</td>
<td>1</td>
<td>00 0001 ffff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>CLRW – f</td>
<td>Clear W</td>
<td>1</td>
<td>00 0001 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>DECF 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>INCF 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>MOV 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</td>
<td>Move W to f</td>
<td>1</td>
<td>00 0000 0000 0000</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>RLF 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>RRF 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 0100 dfff ffff</td>
<td>C, DC, Z</td>
<td>2</td>
</tr>
<tr>
<td>SUBWFf d</td>
<td>Subtract with Borrow W from f</td>
<td>1</td>
<td>11 1111 dfff ffff</td>
<td>C, DC, Z</td>
<td>2</td>
</tr>
<tr>
<td>SWAPF 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 1011 dfff ffff</td>
<td></td>
<td>1, 2</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>1, 2</td>
</tr>
<tr>
<td><strong>BIT-ORIENTED FILE 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 000b bfff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>BSF f, b</td>
<td>Bit Set f</td>
<td>1</td>
<td>01 01bb bfff ffff</td>
<td>Z</td>
<td>2</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>1, 2</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>1, 2</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 1101 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 1100 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>MOVLB k</td>
<td>Move literal to PCLATH</td>
<td>1</td>
<td>11 0001 1kkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVLW 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.
### TABLE 28-3: PIC16F/LF1826/27 ENHANCED 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></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRA k</td>
<td>Relative Branch</td>
<td>2</td>
<td>11 000k 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 1010</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>CLRWDT –</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 0011</td>
<td></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 to FSRn</td>
<td>1</td>
<td>11 0001 0nkk kkkk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVIW n mm</td>
<td>Move INDFn to W, with pre/post inc/dec</td>
<td>1</td>
<td>00 0000 0001 0nmm Z 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k[n]</td>
<td>Move INDFn to W, Indexed Indirect.</td>
<td>1</td>
<td>11 1111 0nkk kkkk Z 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVWI n mm</td>
<td>Move INDFn, with pre/post inc/dec</td>
<td>1</td>
<td>00 0000 0001 1nmm Z 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k[n]</td>
<td>Move W to INDFn, Indexed Indirect.</td>
<td>1</td>
<td>11 1111 1nkk kkkk Z 2</td>
<td></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.
### 28.2 Instruction Descriptions

#### ADDFSR  Add Literal to FSRn

**Syntax:** 
\[
[label] \text{ADDFSR } \text{FSR}_n, k
\]

**Operands:**
\[-32 \leq k \leq 31 \]
\[n \in [0, 1]\]

**Operation:**
FSR(n) + k → FSR(n)

**Status Affected:** None

**Description:**
The signed 6-bit literal ‘k’ is added to the contents of the FSRH:FSRL register pair.
FSRn is limited to the range 0000h - FFFFh. Moving beyond these bounds will cause the FSR to wrap around.

#### ADDLW  Add literal and W

**Syntax:** 
\[
[label] \text{ADDLW } k
\]

**Operands:**
\[0 \leq k \leq 255\]

**Operation:**
(W) + k → (W)

**Status Affected:** C, DC, Z

**Description:**
The contents of the W register are added to the eight-bit literal ‘k’ and the result is placed in the W register.

#### ADDWF  Add W and f

**Syntax:** 
\[
[label] \text{ADDWF } f, d
\]

**Operands:**
\[0 \leq f \leq 127 \]
\[d \in [0, 1]\]

**Operation:**
(W) + (f) → (destination)

**Status Affected:** C, DC, Z

**Description:**
Add the contents of the W register with register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

#### ADDWFC  ADD W and CARRY bit to f

**Syntax:** 
\[
[label] \text{ADDWFC } f, d
\]

**Operands:**
\[0 \leq f \leq 127 \]
\[d \in [0, 1]\]

**Operation:**
(W) + (f) + (C) → dest

**Status Affected:** C, DC, Z

**Description:**
Add W, the Carry flag and data memory location ‘f’. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed in data memory location ‘f’.

#### ANDLW  AND literal with W

**Syntax:** 
\[
[label] \text{ANDLW } k
\]

**Operands:**
\[0 \leq k \leq 255\]

**Operation:**
(W) .AND. (k) → (W)

**Status Affected:** Z

**Description:**
The contents of W register are AND’ed with the eight-bit literal ‘k’. The result is placed in the W register.

#### ANDWF  AND W with f

**Syntax:** 
\[
[label] \text{ANDWF } f, d
\]

**Operands:**
\[0 \leq f \leq 127\]
\[d \in [0, 1]\]

**Operation:**
(W) .AND. (f) → (destination)

**Status Affected:** Z

**Description:**
AND the W register with register ‘f’. If ‘d’ is ‘0’, the result is placed in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

#### ASRF  Arithmetic Right Shift

**Syntax:** 
\[
[label] \text{ASRF } f, d
\]

**Operands:**
\[0 \leq f \leq 127\]
\[d \in [0, 1]\]

**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’.

[Diagram showing register f and C]
BCF  Bit Clear f
Syntax:  \[ label \] BCF     f,b
Operands:  0 ≤ f ≤ 127
          0 ≤ b ≤ 7
Operation:  0 → (f<b>)
Status Affected: None
Description: Bit 'b' in register 'f' is cleared.

BRA  Relative Branch
Syntax:  \[ label \] BRA   label
         \[ label \] BRA   $+k
Operands:  -256 ≤ label - PC + 1 ≤ 255
          -256 ≤ k ≤ 255
Operation:  (PC) + 1 + k → PC
Status Affected: None
Description: Add the signed 9-bit literal 'k' to the
PC. Since the PC will have incremented to fetch the next instruction,
the new address will be PC + 1 + k. This instruction is a two-cycle instruction.
This branch has a limited range.

BRW  Relative Branch with W
Syntax:  \[ label \] BRW
Operands: None
Operation:  (PC) + (W) → PC
Status Affected: None
Description: Add the contents of W (unsigned) to
the PC. Since the PC will have incremented to fetch the next instruction,
the new address will be PC + 1 + (W). This instruction is a two-cycle instruction.

BSF  Bit Set f
Syntax:  \[ label \] BSF     f,b
Operands:  0 ≤ f ≤ 127
          0 ≤ b ≤ 7
Operation:  1 → (f<b>)
Status Affected: None
Description: Bit 'b' in register 'f' is set.

BTFSC  Bit Test f, Skip if Clear
Syntax:  \[ label \] BTFSC   f,b
Operands:  0 ≤ f ≤ 127
          0 ≤ b ≤ 7
Operation:  skip if (f<b>) = 0
Status Affected: None
Description: If bit 'b' in register 'f' is '1', the next
instruction is executed. 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.

BTSS  Bit Test f, Skip if Set
Syntax:  \[ label \] BTSS   f,b
Operands:  0 ≤ f ≤ 127
          0 ≤ b ≤ 7
Operation:  skip if (f<b>) = 1
Status Affected: None
Description: If bit 'b' in register 'f' is '0', the next
instruction is executed. 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 Syntax: \[ \text{label} \] CALL \ k
Operands: \( 0 \leq k \leq 2047 \)
Operation: \((\text{PC}) + 1 \rightarrow \text{TOS}, \ k \rightarrow \text{PC}<10:0>, (\text{PCLATH}<4:3>) \rightarrow \text{PC}<12:11>\)
Status Affected: None
Description: Call Subroutine. First, return address (PC + 1) is pushed onto the stack. The eleven-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

### CALLW Syntax: \[ \text{label} \] CALLW
Operands: None
Operation: \((\text{PC}) + 1 \rightarrow \text{TOS}, (\text{W}) \rightarrow \text{PC}<7:0>, (\text{PCLATH}<6:0>) \rightarrow \text{PC}<14:8>\)
Status Affected: None
Description: Subroutine call with W. First, the return address (PC + 1) is pushed onto the return stack. Then, the contents of W is loaded into PC<7:0>, and the contents of PCLATH into PC<14:8>. CALLW is a two-cycle instruction.

### CLRWF Syntax: \[ \text{label} \] CLRWF \ 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’.

### COMF Syntax: \[ \text{label} \] COMF
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 Syntax: \[ \text{label} \] DECF
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’.

### CLRWD Syntax: \[ \text{label} \] CLRWD
Operands: None
Operation: \(00h \rightarrow \text{WDT}\)
\(0 \rightarrow \text{WDT prescaler}, 1 \rightarrow \text{T0}, 1 \rightarrow \text{PD}\)
Status Affected: \(\text{T0}, \text{PD}\)
Description: CLRWD instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are set.

### CLRF Syntax: \[ \text{label} \] CLRF
Operands: \( 0 \leq f \leq 127 \)
Operation: \((f) \rightarrow (\text{Z})\)
Status Affected: Z
Description: The contents of register ‘f’ are cleared and the Z bit is set.

### CLRW Syntax: \[ \text{label} \] CLRW
Operands: None
Operation: \(00h \rightarrow (\text{W})\)
\(1 \rightarrow (\text{Z})\)
Status Affected: Z
Description: W register is cleared. Zero bit (Z) is set.
<table>
<thead>
<tr>
<th><strong>DECFSZ</strong></th>
<th>Decrement f, Skip if 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[ label ] DECFSZ f,d</td>
</tr>
<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) - 1 → (destination); skip if result = 0</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
</tbody>
</table>
| Description: | The contents of register 'f' are decre-
|               | mented. If 'd' is '0', the result is placed
|               | in the W register. If 'd' is '1', the result
|               | is placed back in register 'f'.
|               | If the result is '1', the next instruction is
|               | executed. If the result is '0', then a
|               | NOP is executed instead, making it a
|               | 2-cycle instruction. |

<table>
<thead>
<tr>
<th><strong>INCFSZ</strong></th>
<th>Increment f, Skip if 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[ label ] INCFSZ f,d</td>
</tr>
<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) + 1 → (destination), skip if result = 0</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
</tbody>
</table>
| Description: | The contents of register 'f' are incre-
|               | mented. If 'd' is '0', the result is placed
|               | in the W register. If 'd' is '1', the result
|               | is placed back in register 'f'.
|               | If the result is '1', the next instruction is
|               | executed. If the result is '0', a
|               | NOP is executed instead, making it a
|               | 2-cycle instruction. |

<table>
<thead>
<tr>
<th><strong>GOTO</strong></th>
<th>Unconditional Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[ label ] GOTO k</td>
</tr>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 2047</td>
</tr>
<tr>
<td>Operation:</td>
<td>k → PC&lt;10:0&gt;</td>
</tr>
<tr>
<td></td>
<td>PCLATH&lt;4:3&gt; → PC&lt;12:11&gt;</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
</tbody>
</table>
| Description: | GOTO is an unconditional branch. The
|               | eleven-bit immediate value is loaded
|               | into PC bits <10:0>. The upper bits of
|               | PC are loaded from PCLATH<4:3>.
|               | GOTO is a two-cycle instruction. |

<table>
<thead>
<tr>
<th><strong>IORLW</strong></th>
<th>Inclusive OR literal with W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[ label ] IORLW k</td>
</tr>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 255</td>
</tr>
<tr>
<td>Operation:</td>
<td>(W) .OR. k → (W)</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>Z</td>
</tr>
</tbody>
</table>
| Description: | The contents of the W register are
|               | OR'ed with the eight-bit literal 'k'. The
|               | result is placed in the W register. |

<table>
<thead>
<tr>
<th><strong>INCF</strong></th>
<th>Increment f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[ label ] INCF f,d</td>
</tr>
<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) + 1 → (destination)</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>Z</td>
</tr>
</tbody>
</table>
| Description: | The contents of register 'f' are incre-
|               | mented. If 'd' is '0', the result is placed
|               | in the W register. If 'd' is '1', the result
|               | is placed back in register 'f'. |

<table>
<thead>
<tr>
<th><strong>IORWF</strong></th>
<th>Inclusive OR W with f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>[ label ] IORWF f,d</td>
</tr>
<tr>
<td>Operands:</td>
<td>0 ≤ f ≤ 127</td>
</tr>
<tr>
<td></td>
<td>d ∈ [0,1]</td>
</tr>
<tr>
<td>Operation:</td>
<td>(W) .OR. (f) → (destination)</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>Z</td>
</tr>
</tbody>
</table>
| Description: | Inclusive OR the W register with regist-
|               | er 'f'. If 'd' is '0', the result is placed
|               | in the W register. If 'd' is '1', the result
|               | is placed back in register 'f'. |
**LSLF**  Logical Left Shift

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[ label ] 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>(&lt;7&gt;) → C</td>
</tr>
<tr>
<td></td>
<td>(&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>

**LSRF**  Logical Right Shift

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[ label ] 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>(&lt;7:1&gt;) → dest&lt;6:0&gt;</td>
</tr>
<tr>
<td></td>
<td>(&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>

**MOVF**  Move f

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>[ label ] 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>

Words: 1  
Cycles: 1  
Example: MOVF FSR, 0  
After Instruction  
W = value in FSR register  
Z = 1
MOVIW  Move INDFn to W
Syntax:  
[ label ] MOVIW ++FSRn  
[ label ] MOVIW --FSRn  
[ label ] MOVIW FSRn++  
[ label ] MOVIW FSRn--  
[ label ] MOVIW k[FSRn]  
[ label ] MOVIW FSRn
Operands:  
n ∈ [0, 1]  
mm ∈ [00, 01, 10, 11].  
-32 ≤ k ≤ 31  
If not present, k = 0.
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

<table>
<thead>
<tr>
<th>mm</th>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Preincrement</td>
<td>++FSRn</td>
</tr>
<tr>
<td>01</td>
<td>Prededcrement</td>
<td>--FSRn</td>
</tr>
<tr>
<td>10</td>
<td>Postincrement</td>
<td>FSRn++</td>
</tr>
<tr>
<td>11</td>
<td>Postdecrement</td>
<td>FSRn--</td>
</tr>
</tbody>
</table>

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.

MOVLB  Move literal to BSR
Syntax:  
[ label ] MOVLB  k
Operands:  
0 ≤ k ≤ 15
Operation:  
k → BSR
Status Affected:  
None
Description:  
The five-bit literal ‘k’ is loaded into the Bank Select Register (BSR).

MOVLW  Move literal to W
Syntax:  
[ label ] MOVLW  k
Operands:  
0 ≤ k ≤ 255
Operation:  
k → (W)
Status Affected:  
None
Description:  
The eight-bit literal ‘k’ is loaded into W register. The “don’t cares” will assemble as ‘0’s.
Words:  
1
Cycles:  
1
Example:  
MOVLW 0x5A
After Instruction  
W = 0x5A

MOVLW  Move literal to PCLATH
Syntax:  
[ label ] MOVLW  k
Operands:  
0 ≤ k ≤ 127
Operation:  
k → PCLATH
Status Affected:  
None
Description:  
The seven-bit literal ‘k’ is loaded into the PCLATH register.

MOVLW  Move literal to PCLATH
Syntax:  
[ label ] MOVLW  k
Operands:  
0 ≤ k ≤ 255
Operation:  
k → (W)
Status Affected:  
None
Description:  
The eight-bit literal ‘k’ is loaded into W register. The “don’t cares” will assemble as ‘0’s.
Words:  
1
Cycles:  
1
Example:  
MOVLW 0x5A
After Instruction  
W = 0x5A

MOVWF  Move W to f
Syntax:  
[ label ] MOVWF  f
Operands:  
0 ≤ f ≤ 127
Operation:  
(W) → (f)
Status Affected:  
None
Description:  
Move data from W register to register ‘f’.
Words:  
1
Cycles:  
1
Example:  
MOVWF OPTION
Before Instruction  
OPTION = 0xFF  
W = 0x4F
After Instruction  
OPTION = 0x4F  
W = 0x4F
MOVWI  Move W to INDFn

Syntax:

\[
[\text{label}] \text{MOVWI ++FSRn} \\
[\text{label}] \text{MOVWI --FSRn} \\
[\text{label}] \text{MOVWI FSRn++} \\
[\text{label}] \text{MOVWI FSRn--} \\
[\text{label}] \text{MOVWI k[FSRn]} \\
[\text{label}] \text{MOVWI FSRn}
\]

Operands:

\[n \in [0, 1]\]
\[m \in \{00, 01, 10, 11\}\]
\[-32 \leq k \leq 31\]

If not present, \(k = 0\).

Operation:

\[W \rightarrow \text{INDFn}\]

Effective address is determined by

- \(\text{FSR} + 1\) (preincrement)
- \(\text{FSR} - 1\) (predecrement)
- \(\text{FSR} + k\) (relative offset)

After the Move, the FSR value will be either:

- \(\text{FSR} + 1\) (all increments)
- \(\text{FSR} - 1\) (all decrements)
- Unchanged

Status Affected: None

<table>
<thead>
<tr>
<th>mm</th>
<th>Mode</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Preincrement</td>
<td>++FSRn</td>
</tr>
<tr>
<td>01</td>
<td>Predecrement</td>
<td>--FSRn</td>
</tr>
<tr>
<td>10</td>
<td>Postincrement</td>
<td>FSRn++</td>
</tr>
<tr>
<td>11</td>
<td>Postdecrement</td>
<td>FSRn--</td>
</tr>
</tbody>
</table>

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:

\[[\text{label}] \text{NOP}\]

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:

\[[\text{label}] \text{OPTION}\]

Operands: None

Operation: \((W) \rightarrow \text{OPTION_REG}\)

Status Affected: None

Description: Move data from W register to OPTION_REG register.

RESET  Software Reset

Syntax:

\[[\text{label}] \text{RESET}\]

Operands: None

Operation: Execute a device Reset. Resets the nRI flag of the PCON register.

Status Affected: None

Description: This instruction provides a way to execute a hardware Reset by software.
### RETFIE
**Return from Interrupt**

**Syntax:**

```
[ label ] RETFIE
```

**Operands:**

None

**Operation:**

```
TOS → PC,
1 → GIE
```

**Status Affected:**

None

**Description:**

Return from Interrupt. Stack is POPed and Top-of-Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two-cycle instruction.

**Words:**

1

**Cycles:**

2

**Example:**

```
After Interrupt
PC = TOS
GIE = 1
```

### RETLW
**Return with literal in W**

**Syntax:**

```
[ label ] RETLW k
```

**Operands:**

```
0 ≤ k ≤ 255
```

**Operation:**

```
k → (W);
TOS → PC
```

**Status Affected:**

None

**Description:**

The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.

**Words:**

1

**Cycles:**

2

**Example:**

```
CALL TABLE;W contains table
;offset value
• ;W now has table value

TABLE

ADDWF PC ;W = offset
RETLW k1 ;Begin table
RETLW k2 ;
•
•
•
RETLW kn ; End of table
```

### RETURN
**Return from Subroutine**

**Syntax:**

```
[ label ] RETURN
```

**Operands:**

None

**Operation:**

```
TOS → PC
```

**Status Affected:**

None

**Description:**

Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.

**Words:**

1

**Cycles:**

2

### RLF
**Rotate Left f through Carry**

**Syntax:**

```
[ label ] RLF f,d
```

**Operands:**

```
0 ≤ f ≤ 127
0 ≤ d ≤ 1
```

**Operation:**

See description below

**Status Affected:**

C

**Description:**

The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is stored back in register 'f'.

**Words:**

1

**Cycles:**

1

**Example:**

```
RLF REG1,0
```

Before Instruction

```
REG1 = 1110 0110
C = 0
```

After Instruction

```
REG1 = 1110 0110
W = 1100 1100
C = 1
```

---

**Before Instruction**

```
W = 0x07
```

**After Instruction**

```
W = value of k8
```
### RRF Rotate Right f through Carry

**Syntax:**

\[ \text{[ label ]} \quad \text{RRF} \quad f, d \]

**Operands:**

\[ 0 \leq f \leq 127 \]
\[ d \in [0, 1] \]

**Operation:** See description below

**Status Affected:**

C

**Description:** The contents of register ‘f’ are rotated one bit to the right through the Carry flag. If ‘d’ is ‘0’, the result is placed in the W register. If ‘d’ is ‘1’, the result is placed back in register ‘f’.

### SLEEP Enter Sleep mode

**Syntax:**

\[ \text{[ label ]} \quad \text{SLEEP} \]

**Operands:** None

**Operation:**

\[ 00h \rightarrow \text{WDT}, \]
\[ 0 \rightarrow \text{WDT prescaler}, \]
\[ 1 \rightarrow \text{TO}, \]
\[ 0 \rightarrow \text{PD} \]

**Status Affected:**

TO, PD

**Description:** 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.

### SUBLW Subtract W from literal

**Syntax:**

\[ \text{[ label ]} \quad \text{SUBLW} \quad k \]

**Operands:**

\[ 0 \leq k \leq 255 \]

**Operation:**

\[ k - (W) \rightarrow (W) \]

**Status Affected:**

C, DC, Z

**Description:** The W register is subtracted (2’s complement method) from the eight-bit literal ‘k’. The result is placed in the W register.

<table>
<thead>
<tr>
<th>C = 0</th>
<th>W &gt; k</th>
</tr>
</thead>
<tbody>
<tr>
<td>C = 1</td>
<td>W ≤ k</td>
</tr>
<tr>
<td>DC = 0</td>
<td>W&lt;3:0&gt; &gt; k&lt;3:0&gt;</td>
</tr>
<tr>
<td>DC = 1</td>
<td>W&lt;3:0&gt; ≤ k&lt;3:0&gt;</td>
</tr>
</tbody>
</table>

### SUBWF Subtract W from f

**Syntax:**

\[ \text{[ label ]} \quad \text{SUBWF} \quad f, d \]

**Operands:**

\[ 0 \leq f \leq 127 \]
\[ d \in [0, 1] \]

**Operation:**

\[ (f) - (W) \rightarrow (destination) \]

**Status Affected:**

C, DC, Z

**Description:** Subtract (2’s complement method) W register from register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

<table>
<thead>
<tr>
<th>C = 0</th>
<th>W &gt; f</th>
</tr>
</thead>
<tbody>
<tr>
<td>C = 1</td>
<td>W ≤ f</td>
</tr>
<tr>
<td>DC = 0</td>
<td>W&lt;3:0&gt; &gt; f&lt;3:0&gt;</td>
</tr>
<tr>
<td>DC = 1</td>
<td>W&lt;3:0&gt; ≤ f&lt;3:0&gt;</td>
</tr>
</tbody>
</table>

### SUBWFB Subtract W from f with Borrow

**Syntax:**

\[ \text{SUBWFB} \quad f \{,d\} \]

**Operands:**

\[ 0 \leq f \leq 127 \]
\[ d \in [0, 1] \]

**Operation:**

\[ (f) - (W) - (B) \rightarrow \text{dest} \]

**Status Affected:**

C, DC, Z

**Description:** Subtract W and the BORROW flag (CARRY) from register ‘f’ (2’s complement method). If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

<table>
<thead>
<tr>
<th>C = 0</th>
<th>W &gt; f</th>
</tr>
</thead>
<tbody>
<tr>
<td>C = 1</td>
<td>W ≤ f</td>
</tr>
<tr>
<td>DC = 0</td>
<td>W&lt;3:0&gt; &gt; f&lt;3:0&gt;</td>
</tr>
<tr>
<td>DC = 1</td>
<td>W&lt;3:0&gt; ≤ f&lt;3:0&gt;</td>
</tr>
</tbody>
</table>
### SWAPF
#### Swap Nibbles in f

<table>
<thead>
<tr>
<th>Syntax</th>
<th>[ label ] SWAPF 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;3:0&gt;) → (destination&lt;7:4&gt;), (f&lt;7:4&gt;) → (destination&lt;3:0&gt;)</td>
</tr>
<tr>
<td>Status Affected</td>
<td>None</td>
</tr>
<tr>
<td>Description</td>
<td>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'.</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 ≤ f ≤ 7</td>
</tr>
<tr>
<td>Operation</td>
<td>(W) → TRIS register 'f'</td>
</tr>
<tr>
<td>Status Affected</td>
<td>None</td>
</tr>
<tr>
<td>Description</td>
<td>Move data from W register to TRIS register. When 'f' = 5, TRISA is loaded. When 'f' = 6, TRISB is loaded.</td>
</tr>
</tbody>
</table>

### 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 ≤ k ≤ 255</td>
</tr>
<tr>
<td>Operation</td>
<td>(W) .XOR. k → (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 eight-bit literal 'k'. The result is placed in the W register.</td>
</tr>
</tbody>
</table>

### XORWF
#### Exclusive OR W with f

<table>
<thead>
<tr>
<th>Syntax</th>
<th>[ label ] XORWF 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>(W) .XOR. (f) → (destination)</td>
</tr>
<tr>
<td>Status Affected</td>
<td>Z</td>
</tr>
<tr>
<td>Description</td>
<td>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'.</td>
</tr>
</tbody>
</table>
29.0 ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings(†)

Ambient temperature under bias ....................................................................................................... -40°C to +125°C
Storage temperature ........................................................................................................................ -65°C to +150°C
Voltage on VDD with respect to VSS, PIC16F1826/27 ........................................................................ -0.3V to +6.5V
Voltage on VDD with respect to VSS, PIC16LF1826/27 ...................................................................... -0.3V to +4.0V
Voltage on MCLR with respect to Vss ..................................................................................................... -0.3V to +9.0V
Voltage on all other pins with respect to VSS ........................................................................... -0.3V to (V DD + 0.3V)
Total power dissipation(1) ........................................................................................................................... 800 mW
Maximum current out of Vss pin ...................................................................................................................... 95 mA
Maximum current into VDD pin ......................................................................................................................... 70 mA
Clamp current, IK (VPIN < 0 or VPIN > VDD) ................................................................................................ ±20 mA
Maximum output current sunk by any I/O pin .................................................................................................... 25 mA
Maximum output current sourced by any I/O pin .............................................................................................. 25 mA
Maximum current sunk by all ports(2), -40°C ≤ TA ≤ +85°C for industrial ................................................ 200 mA
Maximum current sunk by all ports(2), -40°C ≤ TA ≤ +125°C for extended ................................................. 90 mA
Maximum current sourced by all ports(2), 40°C ≤ TA ≤ +85°C for industrial ............................................. 140 mA
Maximum current sourced by all ports(2), -40°C ≤ TA ≤ +125°C for extended ........................................... 65 mA

Note 1: Power dissipation is calculated as follows: PDIS = VDD x {IDD – ∑ IOH} + ∑ ((VDD – VOH) x IOH) + ∑ (VOL x IOL).

† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure above maximum rating conditions for extended periods may affect device reliability.
FIGURE 29-1: PIC16F1826/27 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 29-1 for each Oscillator mode’s supported frequencies.

FIGURE 29-2: PIC16LF1826/27 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 29-1 for each Oscillator mode’s supported frequencies.
FIGURE 29-3: HFINTOSC FREQUENCY ACCURACY OVER DEVICE VDD AND TEMPERATURE

![Graph showing HFINTOSC frequency accuracy over device VDD and temperature. The graph has a grid with VDD (V) on the x-axis ranging from 1.8 to 5.5 and Temperature (°C) on the y-axis ranging from -40 to 125. The graph includes regions marked with +5%, ±2%, and +5% accuracy.]
## 29.1 DC Characteristics: PIC16F/LF1826/27-I/E (Industrial, Extended)

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>PIC16LF1826/27</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</td>
<td>PIC16LF1826/27</td>
<td>1.8</td>
<td>—</td>
<td>3.6</td>
<td>V</td>
<td>Fosc ≤ 16 MHz;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.3</td>
<td>—</td>
<td>3.6</td>
<td>V</td>
<td>Fosc ≤ 32 MHz (NOTE 2)</td>
</tr>
<tr>
<td>D001</td>
<td>VDD</td>
<td>Supply Voltage</td>
<td>PIC16F1826/27</td>
<td>1.8</td>
<td>—</td>
<td>5.5</td>
<td>V</td>
<td>Fosc ≤ 16 MHz;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.3</td>
<td>—</td>
<td>5.5</td>
<td>V</td>
<td>Fosc ≤ 32 MHz (NOTE 2)</td>
</tr>
<tr>
<td>D002*</td>
<td>VDR</td>
<td>RAM Data Retention Voltage(1)</td>
<td>PIC16LF1826/27</td>
<td>1.5</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>Device in Sleep mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PIC16LF1826/27</td>
<td>1.7</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>Device in Sleep mode</td>
</tr>
<tr>
<td>VPOR*</td>
<td>Power-on Reset Release Voltage</td>
<td>—</td>
<td>1.6</td>
<td>—</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPORR*</td>
<td>Power-on Reset Rearm Voltage</td>
<td>PIC16LF1826/27</td>
<td>—</td>
<td>0.8</td>
<td>—</td>
<td>V</td>
<td>Device in Sleep mode</td>
<td></td>
</tr>
<tr>
<td>D004*</td>
<td>SVDD</td>
<td>Vdd Rise Rate to ensure internal Power-on Reset signal</td>
<td>PIC16LF1826/27</td>
<td>0.05</td>
<td>—</td>
<td>—</td>
<td>V/ms</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:** This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.

**Note 2:** PLL required for 32 MHz operation.
FIGURE 29-4: POR AND POR REARM WITH SLOW RISING VDD

Note 1: When NPOR is low, the device is held in Reset.
2: TPOR 1 μs typical.
3: TVLOW 2.7 μs typical.
### 29.2 DC Characteristics: PIC16F/LF1826/27-I/E (Industrial, Extended)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device Characteristics</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Standard Operating Conditions (unless otherwise stated)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Operating temperature</td>
<td>-40°C ≤ TA ≤ +85°C for industrial</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<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>Supply Current (I\text{DD})</td>
<td>—</td>
<td>7.0</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 32 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9.0</td>
<td>μA</td>
<td>3.0</td>
<td>LP Oscillator mode</td>
</tr>
<tr>
<td>D010</td>
<td></td>
<td></td>
<td>9.5</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 32 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>12.5</td>
<td>μA</td>
<td>3.0</td>
<td>LP Oscillator mode</td>
</tr>
<tr>
<td>D011</td>
<td></td>
<td></td>
<td>13.5</td>
<td>μA</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>D011</td>
<td></td>
<td></td>
<td>110</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 1 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>220</td>
<td>μA</td>
<td>3.0</td>
<td>XT Oscillator mode</td>
</tr>
<tr>
<td>D011</td>
<td></td>
<td></td>
<td>130</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 1 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>240</td>
<td>μA</td>
<td>3.0</td>
<td>XT Oscillator mode</td>
</tr>
<tr>
<td>D012</td>
<td></td>
<td></td>
<td>290</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 4 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>520</td>
<td>μA</td>
<td>3.0</td>
<td>XT Oscillator mode</td>
</tr>
<tr>
<td>D012</td>
<td></td>
<td></td>
<td>300</td>
<td>μA</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>D012</td>
<td></td>
<td></td>
<td>300</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 4 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>550</td>
<td>μA</td>
<td>3.0</td>
<td>XT Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>670</td>
<td>μA</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>D013</td>
<td></td>
<td></td>
<td>80</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 1 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>130</td>
<td>μA</td>
<td>3.0</td>
<td>EC Oscillator mode, Medium-power mode</td>
</tr>
<tr>
<td>D013</td>
<td></td>
<td></td>
<td>100</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 1 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>150</td>
<td>μA</td>
<td>3.0</td>
<td>EC Oscillator mode, Medium-power mode</td>
</tr>
<tr>
<td>D014</td>
<td></td>
<td></td>
<td>260</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 4 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>450</td>
<td>μA</td>
<td>3.0</td>
<td>EC Oscillator mode, Medium-power mode</td>
</tr>
<tr>
<td>D014</td>
<td></td>
<td></td>
<td>280</td>
<td>μA</td>
<td>1.8</td>
<td>Fosc = 4 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>480</td>
<td>μA</td>
<td>3.0</td>
<td>EC Oscillator mode, Medium-power mode</td>
</tr>
<tr>
<td>D014</td>
<td></td>
<td></td>
<td>560</td>
<td>μA</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.

**Note**: The test conditions for all I\text{DD} measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V\text{DD}; MCLR = V\text{DD}; WDT disabled.

1. 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.
2. 8 MHz internal RC oscillator with 4x PLL enabled.
3. 8 MHz crystal oscillator with 4x PLL enabled.
4. For RC oscillator configurations, current through R\text{EXT} is not included. The current through the resistor can be extended by the formula I\text{R} = V\text{DD}/2R\text{EXT} (mA) with R\text{EXT} in kΩ.
## DC Characteristics: PIC16F/LF1826/27-I/E (Industrial, Extended) (Continued)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device Characteristics</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D015</td>
<td>Supply Current (Idd)1,2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>3.6 — 4.0</td>
</tr>
<tr>
<td>D015</td>
<td></td>
<td>—</td>
<td>—</td>
<td>5.5</td>
<td>μA</td>
<td>1.8</td>
</tr>
<tr>
<td>D015</td>
<td></td>
<td>—</td>
<td>—</td>
<td>8.5</td>
<td>μA</td>
<td>3.0</td>
</tr>
<tr>
<td>D015</td>
<td></td>
<td>—</td>
<td>—</td>
<td>9.7</td>
<td>μA</td>
<td>5.0</td>
</tr>
<tr>
<td>D016</td>
<td></td>
<td>—</td>
<td>—</td>
<td>110</td>
<td>μA</td>
<td>1.8</td>
</tr>
<tr>
<td>D016</td>
<td></td>
<td>—</td>
<td>—</td>
<td>150</td>
<td>μA</td>
<td>3.0</td>
</tr>
<tr>
<td>D016</td>
<td></td>
<td>—</td>
<td>—</td>
<td>120</td>
<td>μA</td>
<td>1.8</td>
</tr>
<tr>
<td>D016</td>
<td></td>
<td>—</td>
<td>—</td>
<td>160</td>
<td>μA</td>
<td>3.0</td>
</tr>
<tr>
<td>D016</td>
<td></td>
<td>—</td>
<td>—</td>
<td>190</td>
<td>μA</td>
<td>5.0</td>
</tr>
<tr>
<td>D017*</td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.8</td>
<td>mA</td>
<td>1.8</td>
</tr>
<tr>
<td>D017*</td>
<td></td>
<td>—</td>
<td>—</td>
<td>1.3</td>
<td>mA</td>
<td>3.0</td>
</tr>
<tr>
<td>D017*</td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.8</td>
<td>mA</td>
<td>1.8</td>
</tr>
<tr>
<td>D017*</td>
<td></td>
<td>—</td>
<td>—</td>
<td>1.3</td>
<td>mA</td>
<td>3.0</td>
</tr>
<tr>
<td>D017*</td>
<td></td>
<td>—</td>
<td>—</td>
<td>1.4</td>
<td>mA</td>
<td>5.0</td>
</tr>
<tr>
<td>D018</td>
<td></td>
<td>—</td>
<td>—</td>
<td>1.2</td>
<td>mA</td>
<td>1.8</td>
</tr>
<tr>
<td>D018</td>
<td></td>
<td>—</td>
<td>—</td>
<td>2.0</td>
<td>mA</td>
<td>3.0</td>
</tr>
<tr>
<td>D018</td>
<td></td>
<td>—</td>
<td>—</td>
<td>1.2</td>
<td>mA</td>
<td>1.8</td>
</tr>
<tr>
<td>D018</td>
<td></td>
<td>—</td>
<td>—</td>
<td>2.0</td>
<td>mA</td>
<td>3.0</td>
</tr>
<tr>
<td>D019</td>
<td></td>
<td>—</td>
<td>—</td>
<td>4.0</td>
<td>mA</td>
<td>3.0</td>
</tr>
<tr>
<td>D019</td>
<td></td>
<td>—</td>
<td>—</td>
<td>4.4</td>
<td>mA</td>
<td>3.6</td>
</tr>
<tr>
<td>D019</td>
<td></td>
<td>—</td>
<td>—</td>
<td>3.9</td>
<td>mA</td>
<td>3.0</td>
</tr>
<tr>
<td>D019</td>
<td></td>
<td>—</td>
<td>—</td>
<td>4.2</td>
<td>mA</td>
<td>5.0</td>
</tr>
</tbody>
</table>

1. These parameters are characterized but not tested.

**Note 1:** The test conditions for all Idd measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD; MCLR = VDD; WDT disabled.

**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.

**Note 3:** 8 MHz crystal oscillator with 4x PLL enabled.

**Note 4:** 8 MHz crystal oscillator with 4x PLL enabled.

**Note 5:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be extended by the formula I = VDD/2REXT (mA) with REXT in kΩ.
## 29.2 DC Characteristics: PIC16F/LF1826/27-I/E (Industrial, Extended) (Continued)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device Characteristics</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>VDD</td>
</tr>
<tr>
<td>D020</td>
<td>Supply Current (Idd)(1, 2)</td>
<td></td>
<td></td>
<td></td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.3</td>
<td></td>
<td>3.0</td>
<td>3.6</td>
<td>Fosc = 32 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.3</td>
<td></td>
<td>3.0</td>
<td>5.0</td>
<td>Fosc = 32 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>410</td>
<td></td>
<td>1.8</td>
<td>3.0</td>
<td>Fosc = 4 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>430</td>
<td></td>
<td>1.8</td>
<td>5.0</td>
<td>Fosc = 4 MHz</td>
</tr>
</tbody>
</table>

Note 1: These parameters are characterized but not tested.

Note: The test conditions for all I/O measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD; MCLR = VDD; WDT disabled.

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.

Note 3: 8 MHz internal RC oscillator with 4x PLL enabled.

Note 4: 8 MHz crystal oscillator with 4x PLL enabled.

Note 5: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be extended by the formula IR = VDD/2REXT (mA) with REXT in kΩ.
# 29.3 DC Characteristics: PIC16F/LF1826/27-I/E (Power-Down)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device Characteristics</th>
<th>Min.</th>
<th>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><strong>Power-down Base Current (IPD)</strong>[2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>µA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D022</td>
<td>—</td>
<td>0.02</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>WDT, BOR, FVR, and T1OSC disabled, all Peripherals Inactive</td>
<td></td>
</tr>
<tr>
<td>D022</td>
<td>—</td>
<td>0.03</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D023</td>
<td>—</td>
<td>1.6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>WDT, BOR, FVR, and T1OSC disabled, all Peripherals Inactive</td>
<td></td>
</tr>
<tr>
<td>D023</td>
<td>—</td>
<td>1.9</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D023</td>
<td>—</td>
<td>2.0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>5.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D023</td>
<td>—</td>
<td>0.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>LPWDT Current (Note 1)</td>
<td></td>
</tr>
<tr>
<td>D023</td>
<td>—</td>
<td>0.8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D023A</td>
<td>—</td>
<td>1.9</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>LPWDT Current (Note 1)</td>
<td></td>
</tr>
<tr>
<td>D023A</td>
<td>—</td>
<td>2.3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D023A</td>
<td>—</td>
<td>2.4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>5.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D024</td>
<td>—</td>
<td>8.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>FVR current (Note 1)</td>
<td></td>
</tr>
<tr>
<td>D024</td>
<td>—</td>
<td>8.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D025</td>
<td>—</td>
<td>32</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>FVR current (Note 1)</td>
<td></td>
</tr>
<tr>
<td>D025</td>
<td>—</td>
<td>38</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D025</td>
<td>—</td>
<td>68</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>5.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D024</td>
<td>—</td>
<td>8.1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td>BOR Current (Note 1)</td>
<td></td>
</tr>
<tr>
<td>D024</td>
<td>—</td>
<td>18</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td>BOR Current (Note 1)</td>
<td></td>
</tr>
<tr>
<td>D025</td>
<td>—</td>
<td>0.6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>T1OSC Current (Note 1)</td>
<td></td>
</tr>
<tr>
<td>D025</td>
<td>—</td>
<td>0.8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D026</td>
<td>—</td>
<td>3.0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>T1OSC Current (Note 1)</td>
<td></td>
</tr>
<tr>
<td>D026</td>
<td>—</td>
<td>3.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D026</td>
<td>—</td>
<td>4.0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>5.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D026</td>
<td>—</td>
<td>0.1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>A/D Current (Note 1, Note 3), no conversion in progress</td>
<td></td>
</tr>
<tr>
<td>D026</td>
<td>—</td>
<td>0.1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D026</td>
<td>—</td>
<td>18</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1.8</td>
<td>A/D Current (Note 1, Note 3), no conversion in progress</td>
<td></td>
</tr>
<tr>
<td>D026</td>
<td>—</td>
<td>21</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D026</td>
<td>—</td>
<td>25</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>5.0</td>
<td></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.

**Legend:**
- TBD = To Be Determined
- Note 1: The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.
- 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: A/D oscillator source is FRC.
## 29.3 DC Characteristics: PIC16F/LF1826/27-I/E (Power-Down) (Continued)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device Characteristics</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max. +85°C</th>
<th>Max. +125°C</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D026A*</td>
<td></td>
<td>250</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>A/D Current (Note 1, Note 3), conversion in progress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>250</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D026A*</td>
<td></td>
<td>280</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>A/D Current (Note 3), conversion in progress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>280</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D027</td>
<td></td>
<td>3.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Cap Sense Low Power Oscillator mode (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D027</td>
<td></td>
<td>3.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Cap Sense Low Power Oscillator mode (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D027A</td>
<td></td>
<td>4.2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Cap Sense Medium Power Oscillator mode (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D027A</td>
<td></td>
<td>8.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Cap Sense Medium Power Oscillator mode (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D027B</td>
<td></td>
<td>12</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Cap Sense High Power Oscillator mode (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D027B</td>
<td></td>
<td>16</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Cap Sense High Power Oscillator mode (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>36</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D028</td>
<td></td>
<td>4.9</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Comparator Current, Low Power mode, one comparator enabled (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7.0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D028</td>
<td></td>
<td>9.0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Comparator Current, Low Power mode, one comparator enabled (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.3</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D028A</td>
<td></td>
<td>6.6</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Comparator Current, Low Power mode, two comparators enabled (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>D028A</td>
<td></td>
<td>8.5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td>Comparator Current, Low Power mode, two comparators enabled (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8.8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.2</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>μA</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.

**Legend:**
- TBD = To Be Determined

**Note:**
1. The peripheral current is the sum of the base I DD or IPD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base I DD or IPD current from this limit. Max values should be used when calculating total current consumption.
2. The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD.
3. A/D oscillator source is FRC.
### 29.3 DC Characteristics: PIC16F/LF1826/27-I/E (Power-Down) (Continued)

<table>
<thead>
<tr>
<th>Device</th>
<th>Characteristics</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max. +85°C</th>
<th>Max. +125°C</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D028B</td>
<td>Power-down Base Current (IPD)</td>
<td>24</td>
<td>25</td>
<td>μA</td>
<td>1.8</td>
<td>Comparator Current, High Power mode, one comparator enabled (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27</td>
<td>28</td>
<td>μA</td>
<td>1.8</td>
<td>Comparator Current, High Power mode, one comparator enabled (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29</td>
<td>30</td>
<td>μA</td>
<td>3.0</td>
<td>Comparator Current, High Power mode, two comparators enabled (Note 1)</td>
</tr>
<tr>
<td>D028C</td>
<td></td>
<td>40</td>
<td>41</td>
<td>μA</td>
<td>4.8</td>
<td>Comparator Current, High Power mode, two comparators enabled (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>43</td>
<td>44</td>
<td>μA</td>
<td>9.0</td>
<td>Comparator Current, High Power mode, two comparators enabled (Note 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>45</td>
<td>46</td>
<td>μA</td>
<td>5.0</td>
<td>Comparator Current, High Power mode, two comparators enabled (Note 1)</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.

Legend: TBD = To Be Determined

**Note 1:** The peripheral current is the sum of the base IDD or IPD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max values should be used when calculating total current consumption.

**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:** A/D oscillator source is FRC.
### 29.4 DC Characteristics: PIC16F/LF1826/27-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>
<tbody>
<tr>
<td>D030</td>
<td>VIL</td>
<td>Input Low Voltage</td>
<td>—</td>
<td>—</td>
<td>0.8</td>
<td>V</td>
<td>4.5V ≤ Vdd ≤ 5.5V</td>
</tr>
<tr>
<td>D030A</td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.15</td>
<td>V</td>
<td>1.8V ≤ Vdd ≤ 4.5V</td>
</tr>
<tr>
<td>D031</td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.2</td>
<td>V</td>
<td>2.0V ≤ Vdd ≤ 5.5V</td>
</tr>
<tr>
<td>D032</td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.3</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D033</td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.8</td>
<td>V</td>
<td>2.7V ≤ Vdd ≤ 5.5V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.3</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D040</td>
<td>VIH</td>
<td>Input High Voltage</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>4.5V ≤ Vdd ≤ 5.5V</td>
</tr>
<tr>
<td>D040A</td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.25</td>
<td>V</td>
<td>1.8V ≤ Vdd ≤ 4.5V</td>
</tr>
<tr>
<td>D041</td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.8</td>
<td>V</td>
<td>2.0V ≤ Vdd ≤ 5.5V</td>
</tr>
<tr>
<td>D042</td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.7</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D043A</td>
<td></td>
<td></td>
<td>0.8</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>2.7V ≤ Vdd ≤ 5.5V</td>
</tr>
<tr>
<td>D043B</td>
<td></td>
<td></td>
<td>0.7</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.9</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D060</td>
<td>IIL</td>
<td>Input Leakage Current(2)</td>
<td>—</td>
<td>± 5</td>
<td>± 125</td>
<td>nA</td>
<td>Vss ≤ VPIN ≤ Vdd, Pin at high-impedance at 85°C</td>
</tr>
<tr>
<td>D061</td>
<td></td>
<td></td>
<td>—</td>
<td>± 5</td>
<td>±1000</td>
<td>nA</td>
<td>125°C</td>
</tr>
<tr>
<td>D070</td>
<td>IPUR</td>
<td>Weak Pull-up Current</td>
<td>25</td>
<td>100</td>
<td>200</td>
<td>μA</td>
<td>Vdd = 3.3V, VPIN = Vss</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>25</td>
<td>140</td>
<td>300</td>
<td>μA</td>
<td>Vdd = 5.0V, VPIN = Vss</td>
</tr>
<tr>
<td>D080</td>
<td>VOL</td>
<td>Output Low Voltage</td>
<td>—</td>
<td>—</td>
<td>0.6</td>
<td>V</td>
<td>IOL = 8mA, VDD = 5V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.6</td>
<td>V</td>
<td>IOL = 6mA, VDD = 3.3V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>—</td>
<td>0.6</td>
<td>V</td>
<td>IOL = 1.8mA, VDD = 1.8V</td>
</tr>
</tbody>
</table>

#### Legend:
- TBD = To Be Determined
- * These parameters are characterized but not tested.
- † Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended to use an external clock in RC mode.

**Note 2:** Negative current is defined as current sourced by the pin.

**Note 3:** The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**Note 4:** Including OSC2 in CLKOUT mode.
## 29.4 DC Characteristics: PIC16F/LF1826/27-I/E (Continued)

<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>D090</td>
<td>VoH</td>
<td>Output High Voltage(4)</td>
<td></td>
<td></td>
<td></td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>I/O ports</td>
<td></td>
<td>VDD - 0.7</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td>IOH = 3.5mA, VDD = 5V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IOH = 3mA, VDD = 3.6V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IOH = 1mA, VDD = 1.8V</td>
</tr>
<tr>
<td>D101*</td>
<td>COSC2</td>
<td>Capacitive Loading Specs on Output Pins</td>
<td>—</td>
<td>—</td>
<td>15</td>
<td>pF</td>
<td>In XT, HS and LP modes when external clock is used to drive OSC1</td>
</tr>
<tr>
<td>D101A*</td>
<td>Cio</td>
<td>All I/O pins</td>
<td>—</td>
<td>—</td>
<td>50</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- * These parameters are characterized but not tested.
- † Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended to use an external clock in RC mode.

**Note 2:** Negative current is defined as current sourced by the pin.

**Note 3:** The leakage current on the MCLR pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

**Note 4:** Including OSC2 in CLKOUT mode.
## 29.5 Memory Programming Requirements

<table>
<thead>
<tr>
<th>DC CHARACTERISTICS</th>
<th>Standard Operating Conditions (unless otherwise stated)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Operating temperature $-40°C \leq TA \leq +125°C$</td>
</tr>
</tbody>
</table>

<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/RA5 pin</td>
<td>8.0</td>
<td>—</td>
<td>9.0</td>
<td>V</td>
<td>(Note 3, Note 4)</td>
</tr>
<tr>
<td>D111</td>
<td>IDD</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>VDD</td>
<td>VDD for Bulk Erase</td>
<td>2.7</td>
<td>—</td>
<td>VDD</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D113</td>
<td>VPEW</td>
<td>VDD for Write or Row Erase</td>
<td>VDD min.</td>
<td>—</td>
<td>VDD max.</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>D116</td>
<td>E</td>
<td>Byte Endurance</td>
<td>—</td>
<td>100K</td>
<td>—</td>
<td>E/W</td>
<td>$-40°C \text{ to } +85°C$</td>
</tr>
<tr>
<td>D117</td>
<td>VDRW</td>
<td>VDD for Read/Write</td>
<td>VDD min.</td>
<td>—</td>
<td>VDD max.</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D118</td>
<td>TDEW</td>
<td>Erase/Write Cycle Time</td>
<td>—</td>
<td>4.0</td>
<td>5.0</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>D119</td>
<td>TRETD</td>
<td>Characteristic Retention</td>
<td>40</td>
<td>—</td>
<td>—</td>
<td>Year</td>
<td>Provided no other specifications are violated</td>
</tr>
<tr>
<td>D120</td>
<td>TREF</td>
<td>Number of Total Erase/Write Cycles before Refresh(2)</td>
<td>1M</td>
<td>—</td>
<td>10M</td>
<td>—</td>
<td>E/W</td>
</tr>
<tr>
<td>D121</td>
<td>EP</td>
<td>Cell Endurance</td>
<td>—</td>
<td>10K</td>
<td>—</td>
<td>E/W</td>
<td>$-40°C \text{ to } +85°C \ (Note 1)$</td>
</tr>
<tr>
<td>D122</td>
<td>VPR</td>
<td>VDD for Read</td>
<td>VDD min.</td>
<td>—</td>
<td>VDD max.</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>Provided no other specifications are violated</td>
</tr>
<tr>
<td>D124</td>
<td>TRETD</td>
<td>Characteristic Retention</td>
<td>40</td>
<td>—</td>
<td>—</td>
<td>Year</td>
<td></td>
</tr>
</tbody>
</table>

† Data in “Typ” column is at 3.3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Self-write and Block Erase.

**Note 2:** Refer to Section 11.5.1 “Using the Data EEPROM” for a more detailed discussion on data EEPROM endurance.

**Note 3:** Required only if single-supply programming is disabled.

**Note 4:** The MPLAB ICD 2 does not support variable VPP output. Circuitry to limit the ICD 2 VPP voltage must be placed between the ICD 2 and target system when programming or debugging with the ICD 2.
## 29.6 Thermal Considerations

### 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>Typ.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TH01</td>
<td>θJA</td>
<td>Thermal Resistance Junction to Ambient</td>
<td>TBD</td>
<td>°C/W</td>
<td>18-pin PDIP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>°C/W</td>
<td>18-pin SOIC package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>°C/W</td>
<td>20-pin SSOP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>°C/W</td>
<td>28-pin UQFN 4x4mm package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>°C/W</td>
<td>28-pin QFN 6x6mm package</td>
</tr>
<tr>
<td>TH02</td>
<td>θJC</td>
<td>Thermal Resistance Junction to Case</td>
<td>TBD</td>
<td>°C/W</td>
<td>18-pin SPDIP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>°C/W</td>
<td>18-pin SOIC package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>°C/W</td>
<td>20-pin SSOP package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>°C/W</td>
<td>28-pin UQFN 4x4mm package</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>°C/W</td>
<td>28-pin QFN 6x6mm package</td>
</tr>
<tr>
<td>TH03</td>
<td>TJMAX</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 = PINTERNAL + PI/O</td>
</tr>
<tr>
<td>TH05</td>
<td>PINTERNAL</td>
<td>Internal Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>PINTERNAL = IDD x VDD(1)</td>
</tr>
<tr>
<td>TH06</td>
<td>PI/O</td>
<td>I/O Power Dissipation</td>
<td>—</td>
<td>W</td>
<td>PI/O = ∑ (IOL * VOL) + ∑ (IOH * (VDD - VOH))</td>
</tr>
<tr>
<td>TH07</td>
<td>PDER</td>
<td>Derated Power</td>
<td>—</td>
<td>W</td>
<td>PDER = PDMAX (TJ - TA)/θJA(2)</td>
</tr>
</tbody>
</table>

**Legend:**

- TBD = To Be Determined

**Note:**

1. IDD is current to run the chip alone without driving any load on the output pins.
2. TA = Ambient Temperature
3. TJ = Junction Temperature
29.7 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>F Frequency</th>
<th>T</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>pp</td>
<td>cc CCP1</td>
<td>osc OSC1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ck CLKOUT</td>
<td>rd RD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>cs CS</td>
<td>rw RD or WR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>di SDIx</td>
<td>sc SCKx</td>
<td></td>
</tr>
<tr>
<td></td>
<td>do SDO</td>
<td>ss SS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dt Data in</td>
<td>t0 T0CKI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>io I/O PORT</td>
<td>t1 T1CKI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mc MCLR</td>
<td>wr WR</td>
<td></td>
</tr>
</tbody>
</table>

Lowercase letters (pp) and their meanings:
- cc: CCP1
- ck: CLKOUT
- cs: CS
- di: SDIx
- do: SDO
- dt: Data in
- io: I/O PORT
- mc: MCLR

Uppercase letters and their meanings:
- S: Fall
- H: High
- I: Invalid (High-impedance)
- L: Low
- P: Period
- R: Rise
- V: Valid
- Z: High-impedance

Legend: CL = 50 pF for all pins, 15 pF for OSC2 output.
29.8 AC Characteristics: PIC16F/LF1826/27-I/E

**FIGURE 29-6: CLOCK TIMING**

![Clock Timing Diagram]

**TABLE 29-1: CLOCK OSCILLATOR TIMING REQUIREMENTS**

<table>
<thead>
<tr>
<th>Standard Operating Conditions (unless otherwise stated)</th>
<th>Operating temperature $-40°C \leq TA \leq +125°C$</th>
</tr>
</thead>
</table>

<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 CLIN Frequency†</td>
<td>DC</td>
<td>0.5</td>
<td>MHz</td>
<td></td>
<td>EC Oscillator mode (low)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DC</td>
<td>4</td>
<td>MHz</td>
<td></td>
<td></td>
<td>EC Oscillator mode (medium)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DC</td>
<td>32</td>
<td>MHz</td>
<td></td>
<td></td>
<td>EC Oscillator mode (high)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oscillator Frequency†</td>
<td>DC</td>
<td>32.768</td>
<td>kHz</td>
<td></td>
<td>LP Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>XT</td>
<td>4</td>
<td>MHz</td>
<td></td>
<td></td>
<td>XT Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HS</td>
<td>4</td>
<td>MHz</td>
<td></td>
<td></td>
<td>HS Oscillator mode, VDD \leq 2.3V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EC</td>
<td>20</td>
<td>MHz</td>
<td></td>
<td></td>
<td>HS Oscillator mode, VDD &gt; 2.3V</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RC</td>
<td>DC</td>
<td>4</td>
<td>MHz</td>
<td></td>
<td>RC Oscillator mode</td>
</tr>
<tr>
<td>OS02</td>
<td>Tosc</td>
<td>External CLIN Period†</td>
<td>DC</td>
<td>27</td>
<td>10,000</td>
<td>μs</td>
<td>LP Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DC</td>
<td>250</td>
<td>1000</td>
<td>ns</td>
<td>XT Oscillator mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>1000</td>
<td>ns</td>
<td></td>
<td>HS Oscillator mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>31.25</td>
<td>1000</td>
<td>ns</td>
<td></td>
<td>EC Oscillator mode</td>
<td></td>
</tr>
<tr>
<td>OS03</td>
<td>Tcy</td>
<td>Instruction Cycle Time†</td>
<td>DC</td>
<td>125</td>
<td>ns</td>
<td>Tcy = Fosc/4</td>
<td></td>
</tr>
<tr>
<td>OS04*</td>
<td>TosH</td>
<td>External CLIN High, External CLIN Low</td>
<td>2</td>
<td>—</td>
<td>μs</td>
<td>LP oscillator</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>XT oscillator</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>HS oscillator</td>
<td></td>
</tr>
<tr>
<td>OS05*</td>
<td>TosR</td>
<td>External CLIN Rise, External CLIN Fall</td>
<td>0</td>
<td>—</td>
<td>ns</td>
<td>LP oscillator</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>XT oscillator</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>HS oscillator</td>
<td></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: Instruction cycle period (Tcy) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min” values with an external clock applied to OSC1 pin. When an external clock input is used, the “max” cycle time limit is “DC” (no clock) for all devices.
## TABLE 29-2: OSCILLATOR PARAMETERS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristic</th>
<th>Freq. Tolerance</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(^2)</td>
<td>±2%</td>
<td>—</td>
<td>16.0</td>
<td>—</td>
<td>MHz</td>
<td>0°C ≤ TA ≤ +85°C, VDD ≥ 2.5V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>±5%</td>
<td>—</td>
<td>16.0</td>
<td>—</td>
<td>MHz</td>
<td>-40°C ≤ TA ≤ +125°C</td>
</tr>
<tr>
<td>OS08A</td>
<td>MFosc</td>
<td>Internal Calibrated MFINTOSC Frequency(^2)</td>
<td>±2%</td>
<td>—</td>
<td>500</td>
<td>—</td>
<td>kHz</td>
<td>0°C ≤ TA ≤ +85°C, VDD ≥ 2.5V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>±5%</td>
<td>—</td>
<td>500</td>
<td>—</td>
<td>kHz</td>
<td>-40°C ≤ TA ≤ +125°C</td>
</tr>
<tr>
<td>OS10(^*)</td>
<td>TiOsc</td>
<td>ST HFINTOSC Wake-up from Sleep Start-up Time</td>
<td>—</td>
<td>—</td>
<td>5</td>
<td>8</td>
<td>μs</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MFINTOSC Wake-up from Sleep Start-up Time</td>
<td>—</td>
<td>—</td>
<td>20</td>
<td>30</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:
Instruction cycle period (Tcy) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min" values with an external clock applied to the OSC1 pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

2: To ensure these oscillator frequency tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

3: By design.

## TABLE 29-3: PLL CLOCK TIMING SPECIFICATIONS (VDD = 2.7V TO 5.5V)

<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>F10</td>
<td>Fosc</td>
<td>Oscillator Frequency Range</td>
<td>4</td>
<td>—</td>
<td>8</td>
<td>MHz</td>
<td>—</td>
</tr>
<tr>
<td>F11</td>
<td>Fsys</td>
<td>On-Chip VCO System Frequency</td>
<td>16</td>
<td>—</td>
<td>32</td>
<td>MHz</td>
<td>—</td>
</tr>
<tr>
<td>F12</td>
<td>TRC</td>
<td>PLL Start-up Time (Lock Time)</td>
<td>—</td>
<td>—</td>
<td>2</td>
<td>ms</td>
<td>—</td>
</tr>
<tr>
<td>F13(^*)</td>
<td>Δclk</td>
<td>CLKOUT Stability (Jitter)</td>
<td>—</td>
<td>—</td>
<td>+0.25%</td>
<td>%</td>
<td>—</td>
</tr>
</tbody>
</table>

\(^*\) These parameters are characterized but not tested.
† Data in "Typ" column is at 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.
FIGURE 29-7: CLKOUT AND I/O TIMING

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Write</th>
<th>Fetch</th>
<th>Read</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q4</td>
<td>OS11</td>
<td>OS19</td>
<td>OS13</td>
<td>OS15</td>
</tr>
<tr>
<td>Q1</td>
<td>OS20</td>
<td>OS21</td>
<td>OS16</td>
<td>OS17</td>
</tr>
<tr>
<td>Q2</td>
<td>OS18</td>
<td></td>
<td>OS14</td>
<td></td>
</tr>
<tr>
<td>Q3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

I/O pin (Input) | Old Value | I/O pin (Output) | New Value

OS18, OS19
TABLE 29-4: 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</td>
<td>ns</td>
<td>VDD = 3.3-5.0V</td>
</tr>
<tr>
<td>OS12</td>
<td>TosH2ckH</td>
<td>Fosc↑ to CLKOUT↑ (1)</td>
<td>—</td>
<td>—</td>
<td>72</td>
<td>ns</td>
<td>VDD = 3.3-5.0V</td>
</tr>
<tr>
<td>OS13</td>
<td>TckL2ioV</td>
<td>CLKOUT↓ to Port out valid (1)</td>
<td>—</td>
<td>—</td>
<td>20</td>
<td>ns</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*</td>
<td>ns</td>
<td>VDD = 3.3-5.0V</td>
</tr>
<tr>
<td>OS16</td>
<td>TosH2ioI</td>
<td>Fosc↑ (Q2 cycle) to Port input invalid (I/O in hold time)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>VDD = 3.3-5.0V</td>
</tr>
<tr>
<td>OS17</td>
<td>TioV2osH</td>
<td>Port input valid to Fosc↑ (Q2 cycle) (I/O in setup time)</td>
<td>—</td>
<td>20</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>40</td>
<td>72</td>
<td>ns</td>
<td>VDD = 1.8V</td>
</tr>
<tr>
<td>OS19</td>
<td>TioF</td>
<td>Port output fall time (2)</td>
<td>—</td>
<td>28</td>
<td>55</td>
<td>ns</td>
<td>VDD = 1.8V</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 RC mode where CLKOUT output is 4 x Tosc.
2: Includes OSC2 in CLKOUT mode.

FIGURE 29-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

Note 1: Asserted low.
FIGURE 29-9: BROWN-OUT RESET TIMING AND CHARACTERISTICS

Note 1: 64 ms delay only if PWRTE bit in the Configuration Word 1 is programmed to '0'.
2 ms delay if $PWRTE = 0$ and $VREGEN = 1$. 

(Device in Brown-out Reset) 

(VDD) 

(VBOR) 

(37) 

(33) 

(due to BOR) 

(Device not in Brown-out Reset) 

VBOR and VHYST
TABLE 29-5:  RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET PARAMETERS

Standard Operating Conditions (unless otherwise stated)
Operating Temperature -40°C ≤ TA ≤ +125°C

<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>T MCL</td>
<td>MCLR Pulse Width (low)</td>
<td>2</td>
<td>—</td>
<td>—</td>
<td>μs</td>
<td>VDD = 3.3-5V, -40°C to +85°C</td>
</tr>
<tr>
<td>31</td>
<td>T WDTLP</td>
<td>Low-Power Watchdog Timer</td>
<td>10</td>
<td>18</td>
<td>27</td>
<td>ms</td>
<td>VDD = 3.3-5V</td>
</tr>
<tr>
<td>32</td>
<td>T TOST</td>
<td>Oscillator Start-up Timer Period</td>
<td>—</td>
<td>1024</td>
<td>—</td>
<td>—</td>
<td>Tosc (Note 3)</td>
</tr>
<tr>
<td>33*</td>
<td>T 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>T 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>V VBOR</td>
<td>Brown-out Reset Voltage</td>
<td>2.38</td>
<td>2.5</td>
<td>2.65</td>
<td>V</td>
<td>BORV = 2.6V</td>
</tr>
<tr>
<td>36*</td>
<td>V VHyst</td>
<td>Brown-out Reset Hysteresis</td>
<td>0</td>
<td>25</td>
<td>50</td>
<td>mV</td>
<td>—1.9V</td>
</tr>
<tr>
<td>37*</td>
<td>T TBORDC</td>
<td>Brown-out Reset DC Response Time</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>μs</td>
<td>VDD ≤ VBOR</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 the OSC1 pin. When an external clock input is used, the “max” cycle time limit is “DC” (no clock) for all devices.

2: By design.
3: Period of the slower clock.
4: To ensure these voltage tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 μF and 0.01 μF values in parallel are recommended.

FIGURE 29-10:  TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS
TABLE 29-6: TIMER0 AND TIMER1 EXTERNAL CLOCK 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>No Prescaler</th>
<th>With Prescaler</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>0.5 TCY + 20</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41*</td>
<td>T0L</td>
<td>T0CKI Low Pulse Width</td>
<td>0.5 TCY + 20</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42*</td>
<td>T0P</td>
<td>T0CKI Period</td>
<td>Greater of: 20 or TCY + 40 N</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>N = prescale value (2, 4, ..., 256)</td>
<td></td>
</tr>
<tr>
<td>45*</td>
<td>T1H</td>
<td>T1CKI High Time</td>
<td>Synchronous, No Prescaler 0.5 TCY + 20</td>
<td>15</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46*</td>
<td>T1L</td>
<td>T1CKI Low Time</td>
<td>Synchronous, No Prescaler 0.5 TCY + 20</td>
<td>15</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>47*</td>
<td>T1P</td>
<td>T1CKI Input Period</td>
<td>Synchronous Greater of: 30 or TCY + 40 N</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>N = prescale value (1, 2, 4, 8)</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>T1</td>
<td>Timer1 Oscillator Input Frequency Range</td>
<td>32.4 to 32.768 kHz</td>
<td>33.1 kHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>49*</td>
<td>TCPEZ</td>
<td>Delay from External Clock Edge to Timer Increment</td>
<td>2 Tosc</td>
<td>7 Tosc</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>Timers in Sync mode</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.

FIGURE 29-11: CAPTURE/COMPARE/PWM TIMINGS (CCP)

![Diagram showing CCP timings]

Note: Refer to Figure 29.5 for load conditions.

TABLE 29-7: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)

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>No Prescaler</th>
<th>With Prescaler</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC01</td>
<td>TccL</td>
<td>CCPx Input Low Time</td>
<td>0.5 TCY + 20</td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC02</td>
<td>TccH</td>
<td>CCPx Input High Time</td>
<td>0.5 TCY + 20</td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC03*</td>
<td>TccP</td>
<td>CCPx Input Period</td>
<td>3 TCY + 40 N</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>N = prescale value (1, 4 or 16)</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.
### TABLE 29-8: PIC16F/LF1826/27 A/D CONVERTER (ADC) CHARACTERISTICS:

**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>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>—</td>
<td>±1.7</td>
<td>LSB</td>
<td>VREF = 3.0V</td>
</tr>
<tr>
<td>AD03</td>
<td>EDL</td>
<td>Differential Error</td>
<td>—</td>
<td>—</td>
<td>±1</td>
<td>LSB</td>
<td>No missing codes</td>
</tr>
<tr>
<td>AD04</td>
<td>EOFF</td>
<td>Offset Error</td>
<td>—</td>
<td>—</td>
<td>±2</td>
<td>LSB</td>
<td>VREF = 3.0V</td>
</tr>
<tr>
<td>AD05</td>
<td>EGN</td>
<td>Gain Error</td>
<td>—</td>
<td>—</td>
<td>±1.5</td>
<td>LSB</td>
<td>VREF = 3.0V</td>
</tr>
<tr>
<td>AD06</td>
<td>VREF</td>
<td>Reference Voltage[3]</td>
<td>1.8</td>
<td></td>
<td></td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>AD07</td>
<td>VAIN</td>
<td>Full-Scale Range</td>
<td>—</td>
<td>—</td>
<td>50</td>
<td>kΩ</td>
<td>Can go higher if external 0.01 μF capacitor is present on input pin.</td>
</tr>
<tr>
<td>AD08</td>
<td>ZAIN</td>
<td>Recommended Impedance of Analog Voltage Source</td>
<td>—</td>
<td>—</td>
<td>50</td>
<td>kΩ</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:** Total Absolute Error includes integral, differential, offset and gain errors.

2: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

3: ADC VREF is from external VREF, VDD pin or FVREF, whichever is selected as reference input.

4: When ADC is off, it will not consume any current other than leakage current. The power-down current specification includes any such leakage from the ADC module.

### TABLE 29-9: PIC16F/LF1826/27 A/D 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>A/D Clock Period</td>
<td>1.0</td>
<td></td>
<td>9.0</td>
<td>μs</td>
<td>Tosc-based</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A/D Internal RC Oscillator</td>
<td>1.0</td>
<td>1.6</td>
<td>6.0</td>
<td>μs</td>
<td>ADCS&lt;1:0&gt; = 11 (ADRC mode)</td>
</tr>
<tr>
<td>AD131</td>
<td>TCVN</td>
<td>Conversion Time (not including Acquisition Time)[1]</td>
<td>11</td>
<td></td>
<td>TAD</td>
<td></td>
<td>Set GO/DONE bit to conversion complete</td>
</tr>
<tr>
<td>AD132*</td>
<td>TAQ</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 29-12: PIC16F/LF1826/27 A/D CONVERSION TIMING (NORMAL MODE)

Note 1: If the A/D clock source is selected as RC, a time of Tcy is added before the A/D clock starts. This allows the SLEEP instruction to be executed.

FIGURE 29-13: PIC16F/LF1826/27 A/D CONVERSION TIMING (SLEEP MODE)

Note 1: If the A/D clock source is selected as RC, a time of Tcy is added before the A/D clock starts. This allows the SLEEP instruction to be executed.
TABLE 29-10: COMPARATOR SPECIFICATIONS

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CM01</td>
<td>VIOFF</td>
<td>Input Offset Voltage</td>
<td></td>
<td>±7.5</td>
<td>±60</td>
<td>mV</td>
<td></td>
</tr>
<tr>
<td>CM02</td>
<td>VICM</td>
<td>Input Common Mode Voltage</td>
<td>0</td>
<td>—</td>
<td>VDD</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>CM03</td>
<td>CMRR</td>
<td>Common Mode Rejection Ratio</td>
<td>50</td>
<td>—</td>
<td>—</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>CM04</td>
<td>TRESP</td>
<td>Response Time</td>
<td></td>
<td>150</td>
<td>400</td>
<td>ns</td>
<td>Note 1</td>
</tr>
<tr>
<td>CM05</td>
<td>TMC2OV</td>
<td>Comparator Mode Change to Output Valid*</td>
<td>—</td>
<td>—</td>
<td>10</td>
<td>μs</td>
<td></td>
</tr>
<tr>
<td>CM06</td>
<td>CHYSTER</td>
<td>Comparator Hysterisis</td>
<td>65</td>
<td>—</td>
<td>—</td>
<td>mV</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: These parameters are characterized but not tested.

* These parameters are characterized but not tested.

Operating Conditions: 1.8V < VDD < 5.5V, -40°C < TA < +125°C (unless otherwise stated).

TABLE 29-11: DIGITAL-TO-ANALOG CONVERTER (DAC) SPECIFICATIONS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym.</th>
<th>Characteristics</th>
<th>Min.</th>
<th>Typ</th>
<th>Max.</th>
<th>Units</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC01*</td>
<td>CLSB</td>
<td>Step Size(^{(2)})</td>
<td>—</td>
<td>Vdd/32</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>DAC02*</td>
<td>CACC</td>
<td>Absolute Accuracy</td>
<td>—</td>
<td>±1/2</td>
<td>LSb</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAC03*</td>
<td>CR</td>
<td>Unit Resistor Value (R)</td>
<td>—</td>
<td>TBD</td>
<td>—</td>
<td>Ω</td>
<td></td>
</tr>
<tr>
<td>DAC04*</td>
<td>CST</td>
<td>Settling Time(^{(1)})</td>
<td>10</td>
<td>—</td>
<td>—</td>
<td>μs</td>
<td></td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.

Legend: TBD = To Be Determined

Note 1: Settling time measured while DACR<4:0> transitions from '0000' to '1111'.

TABLE 29-12: PIC16F/LF1826/27 LOW DROPOUT (LDO) REGULATOR 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>LD001</td>
<td>LDO</td>
<td>Regulation Voltage</td>
<td></td>
<td>3.2</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>LD002</td>
<td>LDO</td>
<td>External Capacitor</td>
<td>0.1</td>
<td>—</td>
<td>1</td>
<td>μF</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.
FIGURE 29-14: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

Note: Refer to Figure 29-5 for load conditions.

TABLE 29-13: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)
Operating Temperature: -40°C ≤ TA ≤ 125°C

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>US120</td>
<td>TckH2DTV</td>
<td>SYNC XMIT (Master and Slave)</td>
<td>3.0-5.5V</td>
<td>80</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clock high to data-out valid</td>
<td>1.8-5.5V</td>
<td>100</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>US121</td>
<td>TckRF</td>
<td>Clock out rise time and fall time (Master mode)</td>
<td>3.0-5.5V</td>
<td>45</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>US122</td>
<td>TdTRF</td>
<td>Data-out rise time and fall time</td>
<td>3.0-5.5V</td>
<td>45</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.8-5.5V</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

Note: Refer to Figure 29-5 for load conditions.

FIGURE 29-15: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

Note: Refer to Figure 29-5 for load conditions.

TABLE 29-14: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)
Operating Temperature: -40°C ≤ TA ≤ 125°C

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>US125</td>
<td>TckL2CKL</td>
<td>SYNC RCV (Master and Slave)</td>
<td>10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Data-hold before CK ↓ (DT hold time)</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>US126</td>
<td>TckL2DTL</td>
<td>Data-hold after CK ↓ (DT hold time)</td>
<td>15</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 29-16: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)

Note: Refer to Figure 29-5 for load conditions.

FIGURE 29-17: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)

Note: Refer to Figure 29-5 for load conditions.
FIGURE 29-18: SPI SLAVE MODE TIMING (CKE = 0)

Note: Refer to Figure 29-5 for load conditions.

FIGURE 29-19: SPI SLAVE MODE TIMING (CKE = 1)

Note: Refer to Figure 29-5 for load conditions.
### TABLE 29-15: SPI MODE REQUIREMENTS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP70*</td>
<td>TssL2scH, TssL2scL</td>
<td>SSt↓ to SCKx↓ or SCKx↑ input</td>
<td></td>
<td>Tcy</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP71*</td>
<td>TscH</td>
<td>SCKx input high time (Slave mode)</td>
<td>Tcy + 20</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP72*</td>
<td>TscL</td>
<td>SCKx input low time (Slave mode)</td>
<td>Tcy + 20</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP73*</td>
<td>TovV2scH, TovV2scL</td>
<td>Setup time of SDIx data input to SCKx edge</td>
<td>100</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP74*</td>
<td>Tsch2dIL, TscL2dIL</td>
<td>Hold time of SDIx data input to SCKx edge</td>
<td>100</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP75*</td>
<td>TdOR</td>
<td>SDO data output rise time</td>
<td></td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.8-5.5V</td>
<td>25</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP76*</td>
<td>TdOF</td>
<td>SDOx data output fall time</td>
<td></td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP77*</td>
<td>TsshH2doZ</td>
<td>SSt↑ to SDOx output high-impedance</td>
<td>10</td>
<td></td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP78*</td>
<td>TscR</td>
<td>SCKx output rise time (Master mode)</td>
<td></td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.8-5.5V</td>
<td>25</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP79*</td>
<td>TscF</td>
<td>SCKx output fall time (Master mode)</td>
<td></td>
<td>10</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP80*</td>
<td>TshH2doV, TscL2doV</td>
<td>SDOx data output valid after SCKx edge</td>
<td></td>
<td></td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1.8-5.5V</td>
<td></td>
<td>145</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP81*</td>
<td>TovV2scH, TovV2scL</td>
<td>SDOx data output setup to SCKx edge</td>
<td>Tcy</td>
<td></td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP82*</td>
<td>TssL2doV</td>
<td>SDOx data output valid after SS↓ edge</td>
<td></td>
<td></td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>SP83*</td>
<td>TshH2ssH, TscL2ssH</td>
<td>SSt↑ after SCKx edge</td>
<td>1.5Tcy + 40</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. These parameters are for design guidance only and are not tested.

### FIGURE 29-20: \(^2C\) BUS START/STOP BITS TIMING

Note: Refer to Figure 29-5 for load conditions.
FIGURE 29-21: \( \text{I}^2\text{C}^{\text{TM}} \text{ BUS DATA TIMING} \)

Note: Refer to Figure 29-5 for load conditions.
<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP100*</td>
<td>THIGH</td>
<td>Clock high time</td>
<td>100 kHz mode</td>
<td>4.0</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>0.6</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SSPx module</td>
<td>1.5TY</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SP101*</td>
<td>TLOW</td>
<td>Clock low time</td>
<td>100 kHz mode</td>
<td>4.7</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>1.3</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SSPx module</td>
<td>1.5TY</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SP102*</td>
<td>TR</td>
<td>SDAx and SCLx rise time</td>
<td>100 kHz mode</td>
<td>—</td>
<td>1000</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>20 + 0.1C_B</td>
<td>300</td>
<td>ns</td>
</tr>
<tr>
<td>SP103*</td>
<td>TF</td>
<td>SDAx and SCLx fall time</td>
<td>100 kHz mode</td>
<td>—</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>20 + 0.1C_B</td>
<td>250</td>
<td>ns</td>
</tr>
<tr>
<td>SP106*</td>
<td>THD:DAT</td>
<td>Data input hold time</td>
<td>100 kHz mode</td>
<td>0</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>0</td>
<td>0.9</td>
<td>μs</td>
</tr>
<tr>
<td>SP107*</td>
<td>TSU:DAT</td>
<td>Data input setup time</td>
<td>100 kHz mode</td>
<td>250</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>100</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>SP109*</td>
<td>TAA</td>
<td>Output valid from clock</td>
<td>100 kHz mode</td>
<td>—</td>
<td>3500</td>
<td>ns</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>SP110*</td>
<td>TBUF</td>
<td>Bus free time</td>
<td>100 kHz mode</td>
<td>4.7</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>1.3</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>SP111</td>
<td>C_B</td>
<td>Bus capacitive loading</td>
<td>—</td>
<td>—</td>
<td>400</td>
<td>pF</td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested.

**Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCLx to avoid unintended generation of Start or Stop conditions.

**Note 2:** A Fast mode (400 kHz) I²C™ bus device can be used in a Standard mode (100 kHz) I²C bus system, but the requirement \(TSU:DAT \geq 250\) ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCLx signal. If such a device does stretch the low period of the SCLx signal, it must output the next data bit to the SDAx line \(TR_{max.} + TSU:DAT = 1000 + 250 = 1250\) ns (according to the Standard mode I²C bus specification), before the SCLx line is released.
### TABLE 29-17: CAP SENSE OSCILLATOR SPECIFICATIONS

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min.</th>
<th>Typ†</th>
<th>Max.</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS01</td>
<td>ISRC</td>
<td>Current Source</td>
<td>High</td>
<td>-3</td>
<td>-8</td>
<td>-15 μA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
<td>-0.8</td>
<td>-1.5</td>
<td>-3  μA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Low</td>
<td>-0.1</td>
<td>-0.3</td>
<td>-0.4 μA</td>
<td></td>
</tr>
<tr>
<td>CS02</td>
<td>ISNK</td>
<td>Current Sink</td>
<td>High</td>
<td>2.5</td>
<td>7.5</td>
<td>14 μA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
<td>0.6</td>
<td>1.5</td>
<td>2.9 μA</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Low</td>
<td>0.1</td>
<td>0.25</td>
<td>0.6 μA</td>
<td></td>
</tr>
<tr>
<td>CS03</td>
<td>VCTH</td>
<td>Cap Threshold</td>
<td>—</td>
<td>0.8</td>
<td>—</td>
<td>mV</td>
<td></td>
</tr>
<tr>
<td>CS04</td>
<td>VCTL</td>
<td>Cap Threshold</td>
<td>—</td>
<td>0.4</td>
<td>—</td>
<td>mV</td>
<td></td>
</tr>
<tr>
<td>CS05</td>
<td>VCHYST</td>
<td>CAP HYSTERESIS (VCTH - VCTL)</td>
<td>High</td>
<td>350</td>
<td>525</td>
<td>725 mV</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Medium</td>
<td>250</td>
<td>375</td>
<td>500 mV</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Low</td>
<td>175</td>
<td>300</td>
<td>425 mV</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.

### FIGURE 29-22: CAP SENSE OSCILLATOR

![Figure 29-22: CAP SENSE OSCILLATOR](image-url)
30.0  DC AND AC
CHARACTERISTICS GRAPHS
AND CHARTS

Graphs and charts are not available at this time.
31.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C18 and MPLAB C30 C Compilers
  - MPLINK™ Object Linker/
    MPLIB™ Object Librarian
  - MPLAB ASM30 Assembler/Linker/Library
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD 2
- Device Programmers
  - PICSTART® Plus Development Programmer
  - MPLAB PM3 Device Programmer
  - PICkit™ 2 Development Programmer
- Low-Cost Demonstration and Development Boards and Evaluation Kits

31.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
  - Simulator
  - Programmer (sold separately)
  - Emulator (sold separately)
  - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Visual device initializer for easy register initialization
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as HI-TECH Software C Compilers and IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either assembly or C)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (assembly or C)
  - Mixed assembly and C
  - Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.
31.2 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for all PIC MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:
- Integration into MPLAB IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

31.3 MPLAB C18 and MPLAB C30 C Compilers

The MPLAB C18 and MPLAB C30 Code Development Systems are complete ANSI C compilers for Microchip's PIC18 and PIC24 families of microcontrollers and the dsPIC30 and dsPIC33 family of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

31.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:
- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

31.5 MPLAB ASM30 Assembler, Linker and Librarian

MPLAB ASM30 Assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:
- Support for the entire dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

31.6 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C18 and MPLAB C30 C Compilers, and the MPASM and MPLAB ASM30 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.
31.7 MPLAB ICE 2000
High-Performance
In-Circuit Emulator

The MPLAB ICE 2000 In-Circuit Emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers. Software control of the MPLAB ICE 2000 In-Circuit Emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The architecture of the MPLAB ICE 2000 In-Circuit Emulator allows expansion to support new PIC microcontrollers.

The MPLAB ICE 2000 In-Circuit Emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft® Windows® 32-bit operating system were chosen to best make these features available in a simple, unified application.

31.8 MPLAB REAL ICE In-Circuit Emulator System

MPLAB REAL ICE In-Circuit Emulator System is Microchip’s next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs PIC® Flash MCUs and dsPIC® Flash DSCs with the easy-to-use, powerful graphical user interface of the MPLAB Integrated Development Environment (IDE), included with each kit.

The MPLAB REAL ICE probe 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 the popular MPLAB ICD 2 system (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

MPLAB REAL ICE is field upgradeable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added, such as software breakpoints and assembly code trace. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, real-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

31.9 MPLAB ICD 2 In-Circuit Debugger

Microchip’s In-Circuit Debugger, MPLAB ICD 2, is a powerful, low-cost, run-time development tool, connecting to the host PC via an RS-232 or high-speed USB interface. This tool is based on the Flash PIC MCUs and can be used to develop for these and other PIC MCUs and dsPIC DSCs. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the Flash devices. This feature, along with Microchip’s In-Circuit Serial Programming™ (ICSP™) protocol, offers cost-effective, in-circuit Flash debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single stepping and watching variables, and CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real time. MPLAB ICD 2 also serves as a development programmer for selected PIC devices.

31.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages and a modular, detachable socket assembly to support various package types. The ICSP™ cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices and incorporates an SD/MMC card for file storage and secure data applications.
31.11 PICSTART Plus Development Programmer

The PICSTART Plus Development Programmer is an easy-to-use, low-cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. The PICSTART Plus Development Programmer supports most PIC devices in DIP packages up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus Development Programmer is CE compliant.

31.12 PICKit 2 Development Programmer

The PICKit™ 2 Development Programmer is a low-cost programmer and selected Flash device debugger with an easy-to-use interface for programming many of Microchip’s baseline, mid-range and PIC18F families of Flash memory microcontrollers. The PICKit 2 Starter Kit includes a prototyping development board, twelve sequential lessons, software and HI-TECH’s PICC™ Lite C compiler, and is designed to help get up to speed quickly using PIC® microcontrollers. The kit provides everything needed to program, evaluate and develop applications using Microchip’s powerful, mid-range Flash memory family of microcontrollers.

31.13 Demonstration, Development and Evaluation Boards

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, Keeloo® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.
32.0 PACKAGING INFORMATION

32.1 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.

* Standard PICmicro® device marking consists of Microchip part number, year code, week code and traceability code. For PICmicro device marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.
### 32.2 Package Details

The following sections give the technical details of the packages.

#### 18-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](http://www.microchip.com/packaging)

![Package Diagram]

<table>
<thead>
<tr>
<th>Units</th>
<th>INCHES</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>Top to Seating Plane</td>
<td>A</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
</tr>
<tr>
<td>Base to Seating Plane</td>
<td>A1</td>
</tr>
<tr>
<td>Shoulder to Shoulder Width</td>
<td>E</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Tip to Seating Plane</td>
<td>L</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>Upper Lead Width</td>
<td>b1</td>
</tr>
<tr>
<td>Lower Lead Width</td>
<td>b</td>
</tr>
<tr>
<td>Overall Row Spacing §</td>
<td>eB</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.

---

Microchip Technology Drawing C04-007B
18-Lead Plastic Small Outline (SO) – Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packages

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Pins</td>
<td>N 18</td>
</tr>
<tr>
<td>Pitch</td>
<td>e 1.27 BSC</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A 2.65</td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2 2.05</td>
</tr>
<tr>
<td>Standoff §</td>
<td>A1 0.10</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E 10.30 BSC</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1 7.50 BSC</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D 11.55 BSC</td>
</tr>
<tr>
<td>Chamfer (optional)</td>
<td>h 0.25</td>
</tr>
<tr>
<td>Foot Length</td>
<td>L 1.27</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1 1.40 REF</td>
</tr>
<tr>
<td>Foot Angle</td>
<td>(\phi) 0°</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c 0.20</td>
</tr>
<tr>
<td>Lead Width</td>
<td>b 0.31</td>
</tr>
<tr>
<td>Mold Draft Angle Top</td>
<td>(\alpha) 5°</td>
</tr>
<tr>
<td>Mold Draft Angle Bottom</td>
<td>(\beta) 5°</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 0.15 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-051B
18-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

RECOMMENDED LAND PATTERN

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dimension Limits</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</td>
<td>X</td>
</tr>
<tr>
<td>Contact Pad Length</td>
<td>Y</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>Gx</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. C04-2051A
20-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at [http://www.microchip.com/packaging](http://www.microchip.com/packaging)

### Dimensions Table

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dimension Limits</strong></td>
<td><strong>MIN</strong></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</td>
<td>A1</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</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>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>Foot Angle</td>
<td>ψ</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 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.
   - BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   - REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-072B
28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

---

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

**Microchip Technology Drawing C04-105B**
28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com-packaging

RECOMMENDED LAND PATTERN

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits (MIN)</td>
<td>NOM</td>
</tr>
<tr>
<td>Contact Pitch E</td>
<td>0.65 BSC</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 C1</td>
<td>5.70</td>
</tr>
<tr>
<td>Contact Pad Spacing C2</td>
<td>5.70</td>
</tr>
<tr>
<td>Contact Pad Width (X28) X1</td>
<td>0.37</td>
</tr>
<tr>
<td>Contact Pad Length (X28) Y1</td>
<td>1.00</td>
</tr>
<tr>
<td>Distance Between Pads G</td>
<td>0.20</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. C04-2105A
28-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) – 4x4x0.5 mm Body [UQFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging
28-Lead Plastic Ultra Thin Quad Flat, No Lead Package (MV) – 4x4x0.5 mm Body [UQFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th></th>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dimension</td>
<td>MIN</td>
</tr>
<tr>
<td>Number of Pins</td>
<td>N</td>
<td>28</td>
</tr>
<tr>
<td>Pitch</td>
<td>e</td>
<td>0.40 BSC</td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
<td>0.45</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
<td>0.00</td>
</tr>
<tr>
<td>Contact Thickness</td>
<td>A3</td>
<td></td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>Exposed Pad Width</td>
<td>E2</td>
<td>2.55</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>Exposed Pad Length</td>
<td>D2</td>
<td>2.55</td>
</tr>
<tr>
<td>Contact Width</td>
<td>b</td>
<td>0.15</td>
</tr>
<tr>
<td>Contact Length</td>
<td>L</td>
<td>0.30</td>
</tr>
<tr>
<td>Contact-to-Exposed Pad</td>
<td>K</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.
APPENDIX A: DATA SHEET
REVISION HISTORY

Revision A
Original release (06/2009)

Revision B (08/09)
Revised Tables 5-3, 6-2, 12-2, 12-3; Updated Electrical Specifications; Added UQFN Package; Added SOIC and QFN Land Patterns; Updated Product ID section.

APPENDIX B: MIGRATING FROM OTHER PIC® DEVICES

This section provides comparisons when migrating from other similar PIC® devices to the PIC16F/LF1826/27 family of devices.

B.1 PIC16F648A to PIC16F/LF1827

TABLE B-1: FEATURE COMPARISON

<table>
<thead>
<tr>
<th>Feature</th>
<th>PIC16F648A</th>
<th>PIC16F/LF1827</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. Operating Speed</td>
<td>20 MHz</td>
<td>32 MHz</td>
</tr>
<tr>
<td>Max. Program Memory</td>
<td>4K</td>
<td>4K</td>
</tr>
<tr>
<td>(Words)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max. SRAM (Bytes)</td>
<td>256</td>
<td>384</td>
</tr>
<tr>
<td>Max. EEPROM (Bytes)</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>A/D Resolution</td>
<td>10-bit</td>
<td>10-bit</td>
</tr>
<tr>
<td>Timers (8/16-bit)</td>
<td>2/1</td>
<td>4/1</td>
</tr>
<tr>
<td>Brown-out Reset</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Internal Pull-ups</td>
<td>RB&lt;7:0&gt;</td>
<td>RB&lt;7:0&gt;, RA5</td>
</tr>
<tr>
<td>Interrupt-on-change</td>
<td>RB&lt;7:4&gt;</td>
<td>RB&lt;7:0&gt;, Edge Selectable</td>
</tr>
<tr>
<td>Comparator</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>AUSART/EUSART</td>
<td>1/0</td>
<td>0/2</td>
</tr>
<tr>
<td>Extended WDT</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Software Control Option</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>of WDT/BOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTOSC Frequencies</td>
<td>48 kHz or 4 MHz</td>
<td>31 kHz - 32 MHz</td>
</tr>
<tr>
<td>Clock Switching</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Capacitive Sensing</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>CCP/ECCP</td>
<td>2/0</td>
<td>2/2</td>
</tr>
<tr>
<td>Enhanced PIC16 CPU</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>MSSPx/SSPx</td>
<td>0</td>
<td>2/0</td>
</tr>
<tr>
<td>Reference Clock</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Data Signal Modulator</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>SR Latch</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Voltage Reference</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>DAC</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
INDEX

A
A/D
Specifications .......................................................... 358
Absolute Maximum Ratings ........................................... 335
AC Characteristics
Industrial and Extended ........................................... 351
Load Conditions ....................................................... 350
ACKSTAT .............................................................. 266
ACKSTAT Status Flag ................................................. 266
ADC ......................................................................... 135
Acquisition Requirements ........................................... 145
Associated registers .................................................. 147
Block Diagram .......................................................... 135
Calculating Acquisition Time ....................................... 145
Channel Selection ..................................................... 136
Configuration .......................................................... 136
Configuring Interrupt ................................................ 140
Conversion Clock ...................................................... 136
Conversion Procedure ................................................ 140
Internal Sampling Switch (RSS) IMPEDANCE ............... 145
Interrupts ............................................................... 138
Operation ............................................................. 139
Operation During Sleep ............................................... 139
Port Configuration .................................................... 136
Reference Voltage (VREF) .......................................... 136
Source Impedance .................................................... 145
Special Event Trigger ................................................ 139
Starting an A/D Conversion ........................................ 138
ADCON0 Register ..................................................... 29, 141
ADCON1 Register ..................................................... 29, 142
ADDFSR ................................................................. 325
ADDFWC ................................................................. 325
ADRESH Register ....................................................... 29
ADRESH Register (ADFM = 0) .................................... 143
ADRESH Register (ADFM = 1) .................................... 144
ADRESL Register (ADFM = 0) .................................... 143
ADRESL Register (ADFM = 1) .................................... 144
Alternate Pin Function ............................................... 115
Analog-to-Digital Converter. See ADC
ANSEL Register ........................................................ 120
ANSELB Register ..................................................... 126
APFCON0 Register ................................................... 116
APFCON1 Register ................................................... 116
Assembler
MPASM Assembler .................................................. 372
B
BAUDCON Register ................................................... 296
BF ................................................................. 266, 268
BF Status Flag ......................................................... 266, 268
Block Diagram
Capacitive Sensing .................................................. 313
Block Diagrams
(CCP) Capture Mode Operation .................................. 205
ADC ................................................................. 135
ADC Transfer Function ............................................ 146
Analog Input Model .................................................. 146, 160
CCPx PWM .......................................................... 209
Clock Source ........................................................ 53
Comparator .......................................................... 156
Compare ........................................................... 207
Crystal Operation ..................................................... 55
Digital-to-Analog Converter (DAC) .................. 150
EUSART Receive .................................................. 286
EUSART Transmit .................................................. 285
External RC Mode .................................................. 56
Fail-Safe Clock Monitor (FSCM) ............................... 63
Generic I/O Port .................................................... 115
Interrupt Logic ....................................................... 81
On-Chip Reset Circuit .............................................. 73
Peripheral Interrupt Logic ......................................... 82
PIC16F/LF1826/27 .................................................. 10, 16
PWM (Enhanced) ................................................... 213
Resonator Operation ............................................... 55
Timer0 ............................................................... 171
Timer1 ............................................................... 175
Timer1 Gate ......................................................... 180, 181, 182
Timer2/4/6 .......................................................... 187
Voltage Reference .................................................. 133
Voltage Reference Output Buffer Example .................... 150
BORCON Register .................................................. 75
BRA ................................................................. 326
Break Character (12-bit) Transmit and Receive ............. 305
Brown-out Reset (BOR) ............................................ 75
Specifications ....................................................... 356
Timing and Characteristics ....................................... 355
C
C Compilers
MPLAB C18 .......................................................... 372
MPLAB C30 .......................................................... 372
CALL ................................................................. 327
CALLW ............................................................... 327
Capacitive Sensing .................................................. 313
Associated registers w/ Capacitive Sensing ............... 317
Specifications ........................................................ 367
Capture Module. See Enhanced Capture/Compare/ PWM (ECCP)
Capture/Compare/PWM ............................................ 201
Capture/Compare/PWM (CCP) ................................. 204
Associated Registers w/ Capture ............................... 206
Associated Registers w/ Compare ............................. 208
Associated Registers w/ PWM ................................. 229
Capture Mode ....................................................... 205
CCPx Pin Configuration ............................................ 205
Clock Selection ...................................................... 204
Compare Mode ..................................................... 207
CCPx Pin Configuration ............................................ 207
Software Interrupt Mode .......................................... 205, 207
Special Event Trigger ............................................. 207
Timer1 Mode Selection ............................................ 205, 207
Prescaler ............................................................ 205
PWM Mode ........................................................ 209
Duty Cycle .......................................................... 210
Effects of Reset ..................................................... 212
Example PWM Frequencies and Resolutions, 20 MHZ .......... 211
Example PWM Frequencies and Resolutions, 32 MHZ .......... 211
Example PWM Frequencies and Resolutions, 8 MHZ .......... 211
Operation in Sleep Mode .......................................... 212
Resolution .......................................................... 211
Setup for Operation ............................................... 212
System Clock Frequency Changes ............................ 212
PWM Period ......................................................... 210
Setup for PWM Operation ........................................ 212

© 2009 Microchip Technology Inc.

Preliminary

DS41391B-page 389
Device Configuration .......................................................... 47
Code Protection ........................................................................ 51
Configuration Word ............................................................. 47
User ID ........................................................................... 51, 52
Device Overview .................................................................. 9, 99
Digital-to-Analog Converter (DAC) .............................................. 149
Associated Registers .......................................................... 153
Effects of a Reset .................................................................. 150
Operation During Sleep .......................................................... 149
Specifications .................................................................. 360

E

ECCP/ICP. See Enhanced Capture/Compare/PWM
EEADR Registers .......................................................... 103
EEADRH Registers .......................................................... 103
EEADRL Register .......................................................... 104
EEADRL Registers .......................................................... 103
EECON1 Register .......................................................... 103, 105
EECON2 Register .......................................................... 103, 106
EEETH Register .......................................................... 104
EEETATL Register .......................................................... 104
EEPROM Data Memory
Avoiding Spurious Write .................................................. 114
Write Verify .................................................................. 114
Effects of Reset
PDM mode .................................................................. 212
Electrical Specifications .................................................. 335
Enhanced Capture/Compare/PWM
Timer Resources .......................................................... 202
Enhanced Capture/Compare/PWM (ECCP) ...................... 202
Enhanced PDM Mode .................................................. 213
Auto-Restart .................................................................. 223
Auto-shutdown .............................................................. 221
Direction Change in Full-Bridge Output Mode ......... 219
Full-Bridge Application .................................................. 217
Full-Bridge Mode .......................................................... 217
Half-Bridge Application .................................................. 216
Half-Bridge Application Examples .................................. 224
Half-Bridge Mode .......................................................... 216
Output Relationships (Active-High and
Active-Low) .............................................................. 214
Output Relationships Diagram ..................................... 215
Programmable Dead Band Delay ...................... 224
Shoot-through Current .................................................. 224
Start-up Considerations .................................................. 221
Specifications .................................................................. 357
Enhanced Mid-range CPU .................................................. 15
Enhanced Universal Synchronous Asynchronous
Receiver Transmitter (EUSART) .................................. 285
Errata ........................................................................... 7
EUSART ............................................................................. 285
Associated Registers
Baud Rate Generator ................................................... 298
Asynchronous Mode ................................................... 287
12-bit Break Transmit and Receive ...................... 305
Associated Registers
Receive .................................................................. 293
Transmit................................................................. 289
Auto-Wake-up on Break .............................................. 303
Baud Rate Generator (BRG) .............................................. 297
Clock Accuracy ............................................................. 284
Receiver ................................................................. 289
Setting up 9-bit Mode with Address Detect .......... 292
Transmitter ............................................................. 287
Baud Rate Generator (BRG)
Auto Baud Rate Detect .............................................. 302
Baud Rate Error, Calculating ................................. 297
Baud Rates, Asynchronous Modes ......................... 299
Formulas .................................................................. 298
High Baud Rate Select (BRGH Bit) .......................... 297
Synchronous Master Mode ........................................ 306, 310
Associated Registers
Receive ................................................................. 309
Transmit ............................................................... 307
Reception ............................................................. 308
Transmission ......................................................... 306
Synchronous Slave Mode
Associated Registers
Receive ................................................................. 311
Transmit ............................................................... 310
Reception ............................................................. 311
Transmission ......................................................... 310
Extended Instruction Set
ADDFSR ................................................................ 325

F
Fail-Safe Clock Monitor ............................................. 63
Fail-Safe Condition Clearing ...................................... 63
Fail-Safe Detection ................................................... 63
Fail-Safe Operation .................................................. 63
Reset or Wake-up from Sleep .................................... 63
Firmware Instructions ................................................ 321
Fixed Voltage Reference (FVR) ................................. 133
Associated Registers ................................................. 134
Flash Program Memory ............................................. 103
Erasing .................................................................... 110
Writing ................................................................... 110
FSR Register ............................................................ 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38
FVRCON (Fixed Voltage Reference Control) Register ... 134

I
I²C Mode (MSSPx)
Acknowledge Sequence Timing .................................. 270
Bus Collision
During a Repeated Start Condition ......................... 275
During a Stop Condition ......................................... 276
Effects of a Reset ................................................... 271
I²C Clock Rate w/BRG ................................................. 278
Master Mode
Operation ............................................................... 262
Reception .............................................................. 268
Start Condition Timing ............................................. 264, 265
Transmission .......................................................... 266
Multi-Master Communication, Bus Collision and Arbitra-
tion ................................................................. 271
Multi-Master Mode .................................................... 271
Read/Write Bit Information (R/W Bit) ......................... 247
Slave Mode
Transmission .......................................................... 252
Sleep Operation ....................................................... 271
Stop Condition Timing .............................................. 270
INDF Register .......................................................... 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38
Indirect Addressing .................................................. 42
Instruction Format .................................................... 322
Instruction Set .......................................................... 321
ADDLW .................................................................. 325
ADDFW .................................................................. 325
ADDFWC ............................................................... 325
ANDLW .................................................................. 325
ANDWF ................................................................. 325
BRA ................................................................. 326
CALL ................................................................. 327
CALLW .............................................................. 327
LSLF .................................................................... 329
LSRF .................................................................... 329
MOVF ................................................................. 329
MOVIW ............................................................... 330
MOVWI ............................................................... 330
OPTION .............................................................. 331
RETURN ............................................................ 332
RETURN ............................................................. 332
RLF ................................................................. 332
RPL ................................................................. 332
SLEEP .............................................................. 333
SUBLW ............................................................... 333
SUBWF ............................................................... 333
SWAPF .............................................................. 334
XORLW .............................................................. 334
XORWF .............................................................. 334
INTCON Register ..................................................... 87
Internal Oscillator Block
INTOSC
Specifications ......................................................... 352
Internal Sampling Switch (RSS) IMPEDANCE ............ 145
Internet Address ...................................................... 394
Interrupt-On-Change ............................................... 129
Associated Registers ................................................. 131
Interrupts .............................................................. 81
ADC .................................................................... 140
Associated registers w/ Interrupts ......................... 95
Configuration Word w/ Clock Sources ..................... 67
Configuration Word w/ PORTA ......................... 122
Configuration Word w/ Reference Clock Sources .... 71
TMR1 ................................................................. 179
INTOSC Specifications .............................................. 352
IOCBF Register ....................................................... 130
IOCBF Register ....................................................... 130
IOCBP Register ....................................................... 130

© 2009 Microchip Technology Inc.

PRELIMINARY
ADCON1 (ADC Control 1) ................................. 142  
ADRESH (ADC Result High) with ADFM = 0) .............. 143  
ADRESH (ADC Result High) with ADFM = 1) .............. 144  
ADRESL (ADC Result Low) with ADFM = 0) ............... 143  
ADRESL (ADC Result Low) with ADFM = 1) ............... 144  
ANSELA (PORTA Analog Select) ............................. 120  
ANSELB (PORTB Analog Select) ............................. 126  
APFCON0 (Alternate Pin Function Control 0) ............. 116  
APFCON1 (Alternate Pin Function Control 1) ............. 116  
BAUDCON (Baud Rate Control) .............................. 296  
BORCON Brown-out Reset Control) .......................... 75  
CCPTMRS (CCP Timers Control) ........................... 204  
CCPxCON (CCPx Auto-Shutdown Control) .................. 222  
CCPxCON (CCPx Control) .................................... 202  
CLKRCON (Reference Clock Control) ...................... 70  
CMOUT (Comparator Output) .................................. 162  
CMxCON0 (Cx Control) ....................................... 161  
CMxCON1 (Cx Control 1) ..................................... 162  
Configuration Word 1 ........................................... 48  
Configuration Word 2 .......................................... 50  
CPSCON0 (Capacitive Sensing Control Register 0) ........ 316  
CPSCON1 (Capacitive Sensing Control Register 1) ........ 317  
DACCON0 ............................................................. 152  
DACCON1 ............................................................. 152  
EEADRL (EEPROM Address) .................................. 104  
EECON1 (EEPROM Control 1) ................................ 105  
EECON2 (EEPROM Control 2) ................................ 106  
EEDATH (EEPROM Data) ....................................... 104  
EEDATL (EEPROM Data) ....................................... 104  
FVRCON .............................................................. 134  
INTCON (Interrupt Control) .................................... 87  
ICBN (Interrupt-on-Change Negative Edge) ............... 130  
ICOBF (Interrupt-on-Change Flag) ........................... 130  
IOCNP (Interrupt-on-Change Positive Edge) ............... 130  
LATA (Data Latch PORTA) .................................... 118  
LATB (Data Latch PORTB) .................................... 124  
MDCARH (Modulation High Carrier Control Register) .... 198  
MDCARL (Modulation Low Carrier Control Register) ..... 199  
MDCON (Modulation Control Register) ..................... 196  
MDSRC (Modulation Source Control Register) ............. 197  
OPTION_REG (OPTION) ........................................ 173  
OSCICON (Oscillator Control) ................................ 65  
OSCTUNE (Oscillator Tuning) .................................. 67  
PCON (Power Control Register) ............................... 79  
PCON (Power Control) .......................................... 79  
PFA1 (Peripheral Interrupt Enable 1) ....................... 88  
PFA2 (Peripheral Interrupt Enable 2) ....................... 89  
PFA3 (Peripheral Interrupt Enable 3) ....................... 90  
PFA4 (Peripheral Interrupt Enable 4) ....................... 91  
PFA1 (Peripheral Interrupt Register 1) ..................... 92  
PFA2 (Peripheral Interrupt Register 2) ..................... 93  
PFA3 (Peripheral Interrupt Request 3) ..................... 94  
PFA4 (Peripheral Interrupt Request 4) ..................... 95  
PFTA ................................................................. 117  
PFTB ................................................................. 124  
PSTRxCON (PWM Steering Control) ....................... 226  
PWRxCON (Enhanced PWM Control) ......................... 225  
RCSTA (Receive Status and Control) ....................... 295  
Special Function, Summary .................................. 28  
SRCON0 (SR Latch Control 0) ................................ 167  
SRCON1 (SR Latch Control 1) ................................ 168  
SSPxADD ( MSSPx Address and Baud Rate, I²C Mode) .... 283  
SSPxCON1 ( MSSPx Control 1) ............................... 280  
SSPxCON2 (SSPx Control 2) ................................. 281  
SSPxCON3 (SSPx Control 3) .................................. 282  
SSPxMSK (SSPx Mask) .......................................... 283  
SPPXSTAT (SSPx Status) ....................................... 279  
STATUS .............................................................. 21  
T1CON (Timer1 Control) ....................................... 183  
T1GCON (Timer1 Gate Control) ............................... 184  
TRIS (Tri-State PORTA) ........................................ 118  
TRISB (Tri-State PORTB) ....................................... 124  
TXCON ............................................................... 189  
TXSTA (Transmit Status and Control) ....................... 294  
WDTCON (Watchdog Timer Control) ......................... 101  
WPUB (Weak Pull-up PORTB) .................................. 119, 125  
Reset ................................................................. 73, 331  
Brown-Out Reset (BOR) ........................................ 74  
MCLR ................................................................. 76  
Power-on Reset (POR) .......................................... 74  
Stack Overflow/Underflow ..................................... 76  
Watchdog Timer (WDT) Reset .................................. 76  
Reset Instruction .................................................. 76  
Resets ................................................................. 76  
Associated Registers ........................................... 80  
Revision History .................................................. 385  
S  
Shoot-through Current .......................................... 224  
Software Simulator (MPLAB SIM) ........................... 372  
SPBRG ............................................................... 297  
SPBRG Register .................................................... 31  
SPBRGH .............................................................. 297  
Special Event Trigger ............................................ 139  
Special Function Registers (SFRs) ............................ 28  
SPI Mode (MSSPx) .................................................. 241  
Associated Registers ........................................... 241  
SPI Clock ............................................................ 237  
SR Latch ............................................................. 165  
Associated registers w/ SR Latch ............................ 169  
SRCON0 Register .................................................. 167  
SRCON1 Register .................................................. 168  
SSP1ADD Register ................................................ 32  
SSP1BUF Register ................................................ 32  
SSP1CON Register ................................................ 32  
SSP1CON2 Register ............................................... 32  
SSP1CON3 Register ............................................... 32  
SSP1MSK Register ................................................ 32  
SSP1STAT Register ................................................ 32  
SSP2ADD Register ................................................ 32  
SSP2BUF Register ................................................ 32  
SSP2CON1 Register ............................................... 32  
SSP2CON2 Register ............................................... 32  
SSP2CON3 Register ............................................... 32  
SSP2MSK Register ................................................ 32  
SSP2STAT Register ................................................ 32  
SSPADD Register ................................................... 283  
SSPCON1 Register ................................................ 280  
SSPCON2 Register ................................................ 281  
SSPCON3 Register ................................................ 282  
SSPMASK Register ................................................ 283  
SSPOV ............................................................... 268  
SSPOV Status Flag ............................................... 268  
SSPSTAT Register ................................................ 279  
R/W Bit ............................................................. 247  
Stack ................................................................. 40  

© 2009 Microchip Technology Inc.  
Preliminary  
DS41391B-page 393
V
VREF, SEE ADC Reference Voltage

W
Wake-up on Break ...................................................... 303
Wake-up Using Interrupts ............................................. 98
Watchdog Timer (WDT) .................................................. 76
  Modes ......................................................................... 100
  Specifications .......................................................... 356
WCOL ........................................................................ 263, 266, 268, 270
WCOL Status Flag ......................................................... 263, 266, 268, 270
WDTCON Register ........................................................ 101
WPUB Register ............................................................ 119, 125
Write Protection ............................................................. 51
WWW Address ............................................................ 394
WWW, On-Line Support ............................................... 7
THE MICROCHIP WEB SITE

Microchip provides online support via our WWW site at www.microchip.com. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user’s guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at www.microchip.com, click on Customer Change Notification and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:
- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support
- Development Systems Information Line

Customers should contact their distributor, representative or field application engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://support.microchip.com
READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

To: Technical Publications Manager

RE: Reader Response

From: Name ____________________________________________

Company _____________________________________________

Address ______________________________________________

City / State / ZIP / Country ______________________________

Telephone: (______) _________ - _________

FAX: (_____ ) _________ - _________

Application (optional):

Would you like a reply? ___ Y ____ N

Device: PIC16F/LF1826/27 Literature Number: DS41391B

Questions:

1. What are the best features of this document?

______________________________________________________________________________

2. How does this document meet your hardware and software development needs?

______________________________________________________________________________

3. Do you find the organization of this document easy to follow? If not, why?

______________________________________________________________________________

4. What additions to the document do you think would enhance the structure and subject?

______________________________________________________________________________

5. What deletions from the document could be made without affecting the overall usefulness?

______________________________________________________________________________

6. Is there any incorrect or misleading information (what and where)?

______________________________________________________________________________

7. How would you improve this document?

______________________________________________________________________________
PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

<table>
<thead>
<tr>
<th>PART NO.</th>
<th>X</th>
<th>/XX</th>
<th>XXX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>Temperature</td>
<td>Package</td>
<td>Pattern</td>
</tr>
</tbody>
</table>

Device: PIC16F1826(1), PIC16F1827(1), PIC16F1826T(2), PIC16F1827T(2), Vcc range 1.8V to 5.5V
PIC16LF1826(1), PIC16LF1827(1), PIC16LF1826T(2), PIC16LF1827T(2), Vcc range 1.8V to 3.6V

Temperature Range:
- I = -40°C to +85°C (Industrial)
- E = -40°C to +125°C (Extended)

Package:
- ML = Micro Lead Frame (QFN) 6x6
- MV = Micro Lead Frame (UQFN) 4x4
- P = Plastic DIP
- SO = SOIC
- SS = SSOP

Pattern: QTP, SQTP, Code or Special Requirements (blank otherwise)

Examples:

- PIC16F1826 - I/ML 301 = Industrial temp., QFN package, Extended Vcc limits, QTP pattern #301.
- PIC16F1826 - I/P = Industrial temp., PDIP package, Extended Vcc limits.
- PIC16F1827 - E/SS = Extended temp., SSOP package, normal Vcc limits.

Note 1: F = Wide Voltage Range
- LF = Standard Voltage Range
2: T = in tape and reel SOIC, SSOP, and QFN/UQFN packages only.