PIC18F87J11 Family Data Sheet

64/80-Pin High-Performance, 1-Mbit Flash Microcontrollers with nanoWatt 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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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, Accuron, dsPIC, KeeLog, KeeLog logo, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, rPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AmpLab, FilterLab, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL, SmartSensor 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, In-Circuit Serial Programming, ICSP, ICEPIC, Mindi, MiliWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, PICkit, PICDEM, PICDEM.net, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rLAB, Select Mode, Total Endurance, UNI/O, 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.

© 2008, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.
Flexible Oscillator Structure:
- Four Crystal modes, including High-Precision PLL
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
  - Provides 8 user-selectable frequencies from 31 kHz to 8 MHz
  - Provides a complete range of clock speeds, from 31 kHz to 32 MHz when used with PLL
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:
- High-Current Sink/Source 25 mA/25mA on PORTB and PORTC
- Four Programmable External Interrupts
- Four Input Change Interrupts
- One 8/16-Bit Timer/Counter
- Two 8-Bit Timers/Counters
- Two 16-Bit Timers/Counters
- Two Capture/Compare/PWM (CCP) modules
- Three Enhanced Capture/Compare/PWM (ECCP) modules:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Two Master Synchronous Serial Port (MSSP) modules supporting 3-Wire SPI (all 4 modes) and \( \text{I}^2\text{C} \)™ Master and Slave modes
- Two Enhanced USART modules:
  - Supports RS-485, RS-232 and LIN 1.2
  - Auto-wake-up on Start bit
  - Auto-Baud Detect

Peripheral Highlights (continued):
- 8-Bit Parallel Master Port/Enhanced Parallel Slave Port (P/M/EPSP) with 16 Address Lines
- Dual Analog Comparators with Input Multiplexing
- 10-Bit, up to 15-Channel Analog-to-Digital Converter module (A/D):
  - Auto-acquisition capability
  - Conversion available during Sleep

External Memory Bus
(80-pin devices only):
- Address Capability of up to 2 Mbytes
- 8-Bit or 16-Bit Interface
- 12-Bit, 16-Bit and 20-Bit Addressing modes

Special Microcontroller Features:
- Low-Power, High-Speed CMOS Flash Technology
- C Compiler Optimized Architecture for Re-Entrant Code
- Power Management Features:
  - Run: CPU on, peripherals on
  - Idle: CPU off, peripherals on
  - Sleep: CPU off, peripherals off
- Priority Levels for Interrupts
- Self-Programmable under Software Control
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-Supply In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) with 3 Breakpoints via Two Pins
- Operating Voltage Range of 2.0V to 3.6V
- 5.5V Tolerant Inputs (digital only pins)
- On-Chip 2.5V Regulator
- Flash Program Memory of 10000 Erase/Write Cycles and 20-Year Data Retention

<table>
<thead>
<tr>
<th>Device</th>
<th>Flash Program Memory (bytes)</th>
<th>SRAM Data Memory (bytes)</th>
<th>I/O</th>
<th>10-Bit A/D (ch)</th>
<th>CCP/ECCP (PWM)</th>
<th>MSSP</th>
<th>SPI</th>
<th>Master I²C™</th>
<th>EUSART</th>
<th>Comparators</th>
<th>Timers 8/16-Bit</th>
<th>External Bus</th>
<th>PMP/EPSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC18F66J11</td>
<td>64 kB</td>
<td>3930</td>
<td>52</td>
<td>11</td>
<td>2/3</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2</td>
<td>2/3</td>
<td>N</td>
</tr>
<tr>
<td>PIC18F66J16</td>
<td>96 kB</td>
<td>3930</td>
<td>52</td>
<td>11</td>
<td>2/3</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2</td>
<td>2/3</td>
<td>N</td>
</tr>
<tr>
<td>PIC18F67J11</td>
<td>128 kB</td>
<td>3930</td>
<td>52</td>
<td>11</td>
<td>2/3</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2</td>
<td>2/3</td>
<td>N</td>
</tr>
<tr>
<td>PIC18F68J11</td>
<td>64 kB</td>
<td>3930</td>
<td>68</td>
<td>15</td>
<td>2/3</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2</td>
<td>2/3</td>
<td>Y</td>
</tr>
<tr>
<td>PIC18F68J16</td>
<td>96 kB</td>
<td>3930</td>
<td>68</td>
<td>15</td>
<td>2/3</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2</td>
<td>2/3</td>
<td>Y</td>
</tr>
<tr>
<td>PIC18F87J11</td>
<td>128 kB</td>
<td>3930</td>
<td>68</td>
<td>15</td>
<td>2/3</td>
<td>2</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>2</td>
<td>2</td>
<td>2/3</td>
<td>Y</td>
</tr>
</tbody>
</table>
Note 1: The ECCP2/P2A pin placement depends on the CCP2MX Configuration bit setting.
Note 1: The ECCP2/P2A pin placement depends on the CCP2MX Configuration bit and Processor mode settings.
Note 2: P1B, P1C, P3B, and P3C pin placement depends on the ECCPMX Configuration bit setting.
Note 3: PMP pin placement depends on the PMPMX Configuration bit setting.
# Table of Contents

1.0 Device Overview ........................................................................................................... 7
2.0 Oscillator Configurations ............................................................................................... 31
3.0 Power-Managed Modes ................................................................................................. 41
4.0 Reset ........................................................................................................................... 49
5.0 Memory Organization .................................................................................................... 61
6.0 Flash Program Memory ................................................................................................ 87
7.0 External Memory Bus .................................................................................................... 97
8.0 8 x 8 Hardware Multiplier ............................................................................................ 109
9.0 Interrupts ...................................................................................................................... 111
10.0 I/O Ports ....................................................................................................................... 127
11.0 Parallel Master Port .................................................................................................... 151
12.0 Timer0 Module ............................................................................................................ 177
13.0 Timer1 Module ............................................................................................................ 181
14.0 Timer2 Module ............................................................................................................ 187
15.0 Timer3 Module ............................................................................................................ 189
16.0 Timer4 Module ............................................................................................................ 193
17.0 Capture/Compare/PWM (CCP) Modules ..................................................................... 195
18.0 Enhanced Capture/Compare/PWM (ECCP) Module ................................................... 203
19.0 Master Synchronous Serial Port (MSSP) Module ....................................................... 221
20.0 Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) .......... 269
22.0 Comparator Module ..................................................................................................... 301
23.0 Comparator Voltage Reference Module ...................................................................... 309
24.0 Special Features of the CPU ....................................................................................... 313
25.0 Instruction Set Summary ............................................................................................. 329
26.0 Development Support ................................................................................................. 379
27.0 Electrical Characteristics ............................................................................................. 383
28.0 Packaging Information ............................................................................................... 423

Appendix A: Revision History .......................................................................................... 429
Appendix B: Device Differences ....................................................................................... 429
The Microchip Web Site .................................................................................................. 431
Customer Change Notification Service ........................................................................... 431
Customer Support .......................................................................................................... 431
Reader Response ............................................................................................................. 432
Product Identification System ......................................................................................... 445
TO OUR VALUED CUSTOMERS

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

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

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

Customer Notification System

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

This document contains device-specific information for the following devices:

- PIC18F66J11
- PIC18F66J16
- PIC18F67J11
- PIC18F86J11
- PIC18F86J16
- PIC18F87J11
- PIC18F87J16

This family introduces a line of low-voltage, general purpose microcontrollers with the main traditional advantage of all PIC18 microcontrollers – namely, high computational performance and a rich feature set – at an extremely competitive price point. These features make the PIC18F87J11 Family a logical choice for many high-performance applications, where an extended peripheral feature set is required, and cost is a primary consideration.

1.1 Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F87J11 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal RC oscillator, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-Fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application’s software design.

1.1.2 OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F87J11 Family offer four different oscillator options, allowing users a range of choices in developing application hardware. These include:

- Two Crystal modes, using crystals or ceramic resonators.
- Two External Clock modes, offering the option of a divide-by-4 clock output.
- An internal oscillator block which provides an 8 MHz clock and an INTRC source (approximately 31 kHz, stable over temperature and VDD), as well as a range of 6 user-selectable clock frequencies, between 125 kHz to 4 MHz, for a total of 8 clock frequencies. This option frees an oscillator pin for use as an additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to all of the oscillator modes, which allows a wide range of clock speeds from 16 MHz to 40 MHz.

The internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.1.3 EXPANDED MEMORY

The PIC18F87J11 family provides ample room for application code, from 64 Kbytes to 128 Kbytes of code space. The Flash cells for program memory are rated to last up to 10,000 erase/write cycles. Data retention without refresh is conservatively estimated to be greater than 20 years.

The Flash program memory is readable, writable, and during normal operation, the PIC18F87J11 Family also provides plenty of room for dynamic application data with up to 3930 bytes of data RAM.

1.1.4 EXTERNAL MEMORY BUS

In the event that 128 Kbytes of memory are inadequate for an application, the 80-pin members of the PIC18F87J11 Family also implement an External Memory Bus (EMB). This allows the controller’s internal program counter to address a memory space of up to 2 Mbytes, permitting a level of data access that few 8-bit devices can claim. This allows additional memory options, including:

- Using combinations of on-chip and external memory up to the 2-Mbyte limit
- Using external Flash memory for reprogrammable application code or large data tables
- Using external RAM devices for storing large amounts of variable data

1.1.5 EXTENDED INSTRUCTION SET

The PIC18F87J11 Family implements the optional extension to the PIC18 instruction set, adding 8 new instructions and an Indexed Addressing mode. Enabled as a device configuration option, the extension has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as ‘C’.
1.1.6 EASY MIGRATION

Regardless of the memory size, all devices share the same rich set of peripherals, allowing for a smooth migration path as applications grow and evolve.

The consistent pinout scheme used throughout the entire family also aids in migrating to the next larger device. This is true when moving between the 64-pin members, between the 80-pin members, or even jumping from 64-pin to 80-pin devices.

The PIC18F87J11 Family is also pin compatible with other PIC18 families, such as the PIC18F87J10, PIC18F85J11, PIC18F8720 and PIC18F8722. This allows a new dimension to the evolution of applications, allowing developers to select different price points within Microchip’s PIC18 portfolio, while maintaining the same feature set.

1.2 Other Special Features

• Communications: The PIC18F87J11 Family incorporates a range of serial and parallel communication peripherals. These devices all include 2 independent Enhanced USARTs and 2 Master SSP modules, capable of both SPI and I²C™ (Master and Slave) modes of operation. The devices also have a parallel port and can be configured to function as either a Parallel Master Port or as a Parallel Slave Port.

• CCP Modules: All devices in the family incorporate two Capture/Compare/PWM (CCP) modules and three Enhanced CCP (ECCP) modules to maximize flexibility in control applications. Up to four different time bases may be used to perform several different operations at once. Each of the three ECCP modules offers up to four PWM outputs, allowing for a total of 12 PWMs. The ECCPs also offer many beneficial features, including polarity selection, programmable dead time, auto-shutdown and restart, and Half-Bridge and Full-Bridge Output modes.

• 10-Bit A/D Converter: This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period, and thus, reducing code overhead.

• Extended Watchdog Timer (WDT): This enhanced version incorporates a 16-bit prescaler, allowing an extended time-out range that is stable across operating voltage and temperature. See Section 27.0 “Electrical Characteristics” for time-out periods.

1.3 Details on Individual Family Members

Devices in the PIC18F87J11 Family are available in 64-pin and 80-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2. The devices are differentiated from each other in three ways:

1. Flash program memory (three sizes, ranging from 64 Kbytes for PIC18FX6J11 devices to 128 Kbytes for PIC18FX7J11 devices).
2. I/O ports (7 bidirectional ports on 64-pin devices, 9 bidirectional ports on 80-pin devices).
3. A/D input channels (11 on 64-pin devices, 15 on 80-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1 and Table 1-2.

The pinouts for all devices are listed in Table 1-3 and Table 1-4.
### TABLE 1-1: DEVICE FEATURES FOR THE PIC18F6XJ1X (64-PIN DEVICES)

<table>
<thead>
<tr>
<th>Features</th>
<th>PIC18F66J11</th>
<th>PIC18F66J16</th>
<th>PIC18F67J11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Frequency</td>
<td>DC – 48 MHz</td>
<td>DC – 48 MHz</td>
<td>DC – 48 MHz</td>
</tr>
<tr>
<td>Program Memory (Bytes)</td>
<td>64K</td>
<td>96K</td>
<td>128K</td>
</tr>
<tr>
<td>Program Memory (Instructions)</td>
<td>32768</td>
<td>49152</td>
<td>65536</td>
</tr>
<tr>
<td>Data Memory (Bytes)</td>
<td>3930</td>
<td>3930</td>
<td>3930</td>
</tr>
<tr>
<td>Interrupt Sources</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O Ports</td>
<td>Ports A, B, C, D, E, F, G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timers</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capture/Compare/PWM Modules</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enhanced Capture/Compare/PWM Modules</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial Communications</td>
<td>MSSP (2), Enhanced USART (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Communications (PMP)</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-Bit Analog-to-Digital Module</td>
<td>11 Input Channels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resets (and Delays)</td>
<td>POR, BOR, RESET Instruction, Stack Full, Stack Underflow, MCLR, WDT (PWRT, OST)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction Set</td>
<td>75 Instructions, 83 with Extended Instruction Set Enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packages</td>
<td>64-Pin TQFP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 1-2: DEVICE FEATURES FOR THE PIC18F8XJ1X (80-PIN DEVICES)

<table>
<thead>
<tr>
<th>Features</th>
<th>PIC18F86J11</th>
<th>PIC18F86J16</th>
<th>PIC18F87J11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Frequency</td>
<td>DC – 48 MHz</td>
<td>DC – 48 MHz</td>
<td>DC – 48 MHz</td>
</tr>
<tr>
<td>Program Memory (Bytes)</td>
<td>64K</td>
<td>96K</td>
<td>128K</td>
</tr>
<tr>
<td>Program Memory (Instructions)</td>
<td>32768</td>
<td>49152</td>
<td>65536</td>
</tr>
<tr>
<td>Data Memory (Bytes)</td>
<td>3930</td>
<td>3930</td>
<td>3930</td>
</tr>
<tr>
<td>Interrupt Sources</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I/O Ports</td>
<td>Ports A, B, C, D, E, F, G, H, J</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timers</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capture/Compare/PWM Modules</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enhanced Capture/Compare/PWM Modules</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial Communications</td>
<td>MSSP (2), Enhanced USART (2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Communications (PMP)</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-Bit Analog-to-Digital Module</td>
<td>15 Input Channels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resets (and Delays)</td>
<td>POR, BOR, RESET Instruction, Stack Full, Stack Underflow, MCLR, WDT (PWRT, OST)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instruction Set</td>
<td>75 Instructions, 83 with Extended Instruction Set Enabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packages</td>
<td>80-Pin TQFP</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 1-1: PIC18F6XJ1X (64-PIN) BLOCK DIAGRAM

Note 1: See Table 1-3 for I/O port pin descriptions.
2: BOR functionality is provided when the on-board voltage regulator is enabled.
FIGURE 1-2: PIC18F8XJ1X (80-PIN) BLOCK DIAGRAM

Note 1: See Table 1-4 for I/O port pin descriptions.
2: BOR functionality is provided when the on-board voltage regulator is enabled.
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCLR</td>
<td>7</td>
<td>I</td>
<td>ST</td>
<td>Master Clear (Reset) input. This pin is an active-low Reset to the device.</td>
</tr>
<tr>
<td>OSC1/CLKI/RA7</td>
<td>39</td>
<td>I</td>
<td>ST</td>
<td>Oscillator crystal or external clock input. Available only in external oscillator modes (EC/ECPLL and HS/HSPLL).</td>
</tr>
<tr>
<td>OSC1</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>Main oscillator input connection.</td>
</tr>
<tr>
<td>CLKI</td>
<td></td>
<td>I</td>
<td>CMOS</td>
<td>Main clock input connection.</td>
</tr>
<tr>
<td>RA7</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>General purpose I/O pin. Available only in INTIO2 and INTPLL2 Oscillator modes.</td>
</tr>
<tr>
<td>OSC2/CLKO/RA6</td>
<td>40</td>
<td>O</td>
<td>—</td>
<td>Oscillator crystal or clock output. Available only in external oscillator modes (EC/ECPLL and HS/HSPLL).</td>
</tr>
<tr>
<td>OSC2</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Main oscillator feedback output connection.</td>
</tr>
<tr>
<td>CLKO</td>
<td></td>
<td>O</td>
<td>—</td>
<td>System cycle clock output (Fosc/4).</td>
</tr>
<tr>
<td>RA6</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>General purpose I/O pin. Available only in INTIO1 and INTPLL1 Oscillator modes.</td>
</tr>
</tbody>
</table>

Legend:
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

Note 1: Default assignment for ECCP2/P2A when Configuration bit, CCP2MX, is set.
Note 2: Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared.
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA is a bidirectional I/O port.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA0/AN0</td>
<td>24</td>
<td>I/O TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA0</td>
<td></td>
<td>I Analog</td>
<td>Analog input 0.</td>
</tr>
<tr>
<td>AN0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA1/AN1</td>
<td>23</td>
<td>I/O TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA1</td>
<td></td>
<td>I Analog</td>
<td>Analog input 1.</td>
</tr>
<tr>
<td>AN1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA2/AN2/VREF-</td>
<td>22</td>
<td>I/O TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA2</td>
<td></td>
<td>I Analog</td>
<td>Analog input 2.</td>
</tr>
<tr>
<td>AN2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF-</td>
<td></td>
<td>I Analog</td>
<td>A/D reference voltage (low) input.</td>
</tr>
<tr>
<td>RA3/AN3/VREF+</td>
<td>21</td>
<td>I/O TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA3</td>
<td></td>
<td>I Analog</td>
<td>Analog input 3.</td>
</tr>
<tr>
<td>AN3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF+</td>
<td></td>
<td>I Analog</td>
<td>A/D reference voltage (high) input.</td>
</tr>
<tr>
<td>RA4/T0CKI</td>
<td>28</td>
<td>I/O ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA4</td>
<td></td>
<td>I ST</td>
<td>Timer0 external clock input.</td>
</tr>
<tr>
<td>T0CKI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA5/AN4</td>
<td>27</td>
<td>I/O TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA5</td>
<td></td>
<td>I Analog</td>
<td>Analog input 4.</td>
</tr>
<tr>
<td>AN4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA6</td>
<td></td>
<td></td>
<td>See the OSC2/CLKO/RA6 pin.</td>
</tr>
<tr>
<td>RA7</td>
<td></td>
<td></td>
<td>See the OSC1/CLKI/RA7 pin.</td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note:**
1. Default assignment for ECCP2/P2A when Configuration bit, CCP2MX, is set.
2. Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared.
PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB0/FLT0/INT0</td>
<td>48</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB0</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>ECCP1/2/3 Fault input.</td>
</tr>
<tr>
<td>FLT0</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>External interrupt 0.</td>
</tr>
<tr>
<td>INT0</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>External interrupt 1.</td>
</tr>
<tr>
<td>PMA4</td>
<td>O</td>
<td>—</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB1/INT1/PMA4</td>
<td>47</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB1</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>External interrupt 1.</td>
</tr>
<tr>
<td>INT1</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA4</td>
<td>O</td>
<td>—</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB2/INT2/PMA3</td>
<td>46</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB2</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>External interrupt 2.</td>
</tr>
<tr>
<td>INT2</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA3</td>
<td>O</td>
<td>—</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB3/INT3/PMA2</td>
<td>45</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB3</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>External interrupt 3.</td>
</tr>
<tr>
<td>INT3</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA2</td>
<td>O</td>
<td>—</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB4/KBI0/PMA1</td>
<td>44</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB4</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>Interrupt-on-change pin.</td>
</tr>
<tr>
<td>KBI0</td>
<td>I</td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA1</td>
<td>I</td>
<td>O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB5/KBI1/PMA0</td>
<td>43</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB5</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>Interrupt-on-change pin.</td>
</tr>
<tr>
<td>KBI1</td>
<td>I</td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA0</td>
<td>I</td>
<td>O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB6/KBI2/PGC</td>
<td>42</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB6</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>Interrupt-on-change pin.</td>
</tr>
<tr>
<td>KBI2</td>
<td>I</td>
<td>TTL</td>
<td></td>
<td>In-Circuit Debugger and ICSP™ programming clock pin.</td>
</tr>
<tr>
<td>PGC</td>
<td>I/O</td>
<td>ST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB7/KBI3/PGD</td>
<td>37</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB7</td>
<td>I</td>
<td>ST</td>
<td></td>
<td>Interrupt-on-change pin.</td>
</tr>
<tr>
<td>KBI3</td>
<td>I</td>
<td>TTL</td>
<td></td>
<td>In-Circuit Debugger and ICSP programming data pin.</td>
</tr>
<tr>
<td>PGD</td>
<td>I/O</td>
<td>ST</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note:**
1. Default assignment for ECCP2/P2A when Configuration bit, CCP2MX, is set.
2. Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared.
### TABLE 1-3: PIC18F6XJ1X PINOUT I/O DESCRIPTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC0/T1OSO/T13CKI</td>
<td>30</td>
<td>I/O</td>
<td>ST</td>
<td>PORTC is a bidirectional I/O port.</td>
</tr>
<tr>
<td>RC0</td>
<td></td>
<td>I</td>
<td>—</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>T1OSO</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Timer1 oscillator output.</td>
</tr>
<tr>
<td>T13CKI</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>Timer1/Timer3 external clock input.</td>
</tr>
<tr>
<td>RC1/T1OSI/ECCP2/P2A</td>
<td>29</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RC1</td>
<td></td>
<td>I</td>
<td>—</td>
<td>Timer1 oscillator input.</td>
</tr>
<tr>
<td>T1OSI</td>
<td></td>
<td>I</td>
<td>CMOS</td>
<td>Timer1 oscillator input.</td>
</tr>
<tr>
<td>ECCP2(t)</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Capture 2 input/Compare 2 output/PWM2 output.</td>
</tr>
<tr>
<td>P2A(t)</td>
<td></td>
<td>O</td>
<td>—</td>
<td>ECCP2 PWM output A.</td>
</tr>
<tr>
<td>RC2/ECCP1/P2A</td>
<td>33</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RC2</td>
<td></td>
<td>I</td>
<td>—</td>
<td>Capture 1 input/Compare 1 output/PWM1 output.</td>
</tr>
<tr>
<td>ECCP1</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Capture 1 input/Compare 1 output/PWM1 output.</td>
</tr>
<tr>
<td>P2A</td>
<td></td>
<td>O</td>
<td>—</td>
<td>ECCP1 PWM output A.</td>
</tr>
<tr>
<td>RC3/SCK1/SCL1</td>
<td>34</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RC3</td>
<td></td>
<td>I</td>
<td>—</td>
<td>Synchronous serial clock input/output for SPI mode.</td>
</tr>
<tr>
<td>SCK1</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Synchronous serial clock input/output for SPI mode.</td>
</tr>
<tr>
<td>SCL1</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Synchronous serial clock input/output for (^{12}C)™ mode.</td>
</tr>
<tr>
<td>RC4/SDI1/SDA1</td>
<td>35</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RC4</td>
<td></td>
<td>I</td>
<td>—</td>
<td>SPI data in.</td>
</tr>
<tr>
<td>SDI1</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>SPI data in.</td>
</tr>
<tr>
<td>SDA1</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>(^{12}C) data I/O.</td>
</tr>
<tr>
<td>RC5/SDO1</td>
<td>36</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RC5</td>
<td></td>
<td>I</td>
<td>—</td>
<td>SPI data out.</td>
</tr>
<tr>
<td>SDO1</td>
<td></td>
<td>O</td>
<td>—</td>
<td>SPI data out.</td>
</tr>
<tr>
<td>RC6/TX1/CK1</td>
<td>31</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RC6</td>
<td></td>
<td>I</td>
<td>—</td>
<td>EUSART1 asynchronous transmit.</td>
</tr>
<tr>
<td>TX1</td>
<td></td>
<td>O</td>
<td>—</td>
<td>EUSART1 synchronous clock (see related RX1/DT1).</td>
</tr>
<tr>
<td>CK1</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>EUSART1 synchronous clock (see related RX1/DT1).</td>
</tr>
<tr>
<td>RC7/RX1/DT1</td>
<td>32</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RC7</td>
<td></td>
<td>I</td>
<td>—</td>
<td>EUSART1 asynchronous receive.</td>
</tr>
<tr>
<td>RX1</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>EUSART1 asynchronous data (see related TX1/CK1).</td>
</tr>
<tr>
<td>DT1</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>EUSART1 asynchronous data (see related TX1/CK1).</td>
</tr>
</tbody>
</table>

**Legend:**  
- TTL = TTL compatible input  
- CMOS = CMOS compatible input or output  
- ST = Schmitt Trigger input with CMOS levels  
- Analog = Analog input  
- I = Input  
- O = Output  
- P = Power  
- OD = Open-Drain (no P diode to VDD)

**Note 1:** Default assignment for ECCP2/P2A when Configuration bit, CCP2MX, is set.  
**Note 2:** Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared.
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD0/PMD0</td>
<td>58</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD0</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>PMD0</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>RD1/PMD1</td>
<td>55</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD1</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>PMD1</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>RD2/PMD2</td>
<td>54</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD2</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>PMD2</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>RD3/PMD3</td>
<td>53</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD3</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>PMD3</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>RD4/PMD4/SDO2</td>
<td>52</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD4</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>PMD4</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>SDO2</td>
<td></td>
<td>O</td>
<td>—</td>
<td>SPI data out.</td>
</tr>
<tr>
<td>RD5/PMD5/SDI2/SDA2</td>
<td>51</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD5</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>PMD5</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>SDI2</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>SPI data in.</td>
</tr>
<tr>
<td>SDA2</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>I²C™ data I/O.</td>
</tr>
<tr>
<td>RD6/PMD6/SCK2/SCL2</td>
<td>50</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RD6</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>PMD6</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
</tbody>
</table>
| SCK2         |            | I/O      | ST          | Synchronous serial clock input/output for SPI mode.
| SCL2         |            | I/O      | ST          | Synchronous serial clock input/output for I²C mode.|
| RD7/PMD7/SS2 | 49         | I/O      | ST          | Digital I/O.                                      |
| RD7          |            | I/O      | TTL         | Parallel Master Port data.                       |
| PMD7         |            | I/O      | TTL         | Parallel Master Port data.                       |
| SS2          |            | I        | TTL         | SPI slave select input.                          |

Legend:
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

Note 1: Default assignment for ECCP2/P2A when Configuration bit, CCP2MX, is set.
Note 2: Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared.
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTE</td>
<td>2</td>
<td>I/O ST</td>
<td></td>
<td>PORTE is a bidirectional I/O port.</td>
</tr>
<tr>
<td>RE0/PMRD/P2D</td>
<td>1</td>
<td>I/O ST</td>
<td></td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>PMRD</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port read strobe.</td>
</tr>
<tr>
<td>P2D</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP2 PWM output D.</td>
</tr>
<tr>
<td>RE1/PMWR/P2C</td>
<td>64</td>
<td>I/O ST</td>
<td></td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE1</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port write strobe.</td>
</tr>
<tr>
<td>PMWR</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP2 PWM output C.</td>
</tr>
<tr>
<td>P2C</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE2/PMBE/P2B</td>
<td>64</td>
<td>I/O ST</td>
<td></td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE2</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port byte enable</td>
</tr>
<tr>
<td>PMBE</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP2 PWM output B.</td>
</tr>
<tr>
<td>P2B</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE3/PMA13/P3C/REFO</td>
<td>63</td>
<td>I/O ST</td>
<td></td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE3</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA13</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP3 PWM output C.</td>
</tr>
<tr>
<td>P3C</td>
<td></td>
<td>O</td>
<td></td>
<td>Reference clock out.</td>
</tr>
<tr>
<td>REFO</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE4/PMA12/P3B</td>
<td>62</td>
<td>I/O ST</td>
<td></td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE4</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA12</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP3 PWM output B.</td>
</tr>
<tr>
<td>P3B</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE5/PMA11/P1C</td>
<td>61</td>
<td>I/O ST</td>
<td></td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE5</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA11</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP1 PWM output C.</td>
</tr>
<tr>
<td>P1C</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE6/PMA10/P1B</td>
<td>60</td>
<td>I/O ST</td>
<td></td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE6</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA10</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP1 PWM output B.</td>
</tr>
<tr>
<td>P1B</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE7/PMA9/ECCP2/P2A</td>
<td>59</td>
<td>I/O ST</td>
<td></td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE7</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA9</td>
<td></td>
<td>O</td>
<td></td>
<td>Capture 2 input/Compare 2 output/PWM2 output.</td>
</tr>
<tr>
<td>ECCP2(2)</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP2 PWM output A.</td>
</tr>
<tr>
<td>P2A(2)</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note 1:** Default assignment for ECCP2/P2A when Configuration bit, CCP2MX, is set.
**Note 2:** Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared.
PORTF is a bidirectional I/O port.

**TABLE 1-3: PIC18F6XJ1X PINOUT I/O DESCRIPTIONS (CONTINUED)**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF1/AN6/C2OUT</td>
<td>17</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RF1</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 6.</td>
</tr>
<tr>
<td>AN6</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Comparator 2 output.</td>
</tr>
<tr>
<td>C2OUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF2/PMA5/AN7/C1OUT</td>
<td>16</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RF2</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 7.</td>
</tr>
<tr>
<td>PMA5</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Comparator 1 output.</td>
</tr>
<tr>
<td>AN7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C1OUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF3/AN8/C2INB</td>
<td>15</td>
<td>I</td>
<td>ST</td>
<td>Digital input.</td>
</tr>
<tr>
<td>RF3</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 8.</td>
</tr>
<tr>
<td>AN8</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input B.</td>
</tr>
<tr>
<td>C2INB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF4/AN9/C2INA</td>
<td>14</td>
<td>I</td>
<td>ST</td>
<td>Digital input.</td>
</tr>
<tr>
<td>RF4</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 8.</td>
</tr>
<tr>
<td>AN9</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input A.</td>
</tr>
<tr>
<td>C2INA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF5/AN10/C1INB/CVREF</td>
<td>13</td>
<td>I</td>
<td>ST</td>
<td>Digital input.</td>
</tr>
<tr>
<td>RF5</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 10.</td>
</tr>
<tr>
<td>AN10</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input B.</td>
</tr>
<tr>
<td>C1INB</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator reference voltage output.</td>
</tr>
<tr>
<td>CVREF</td>
<td></td>
<td>O</td>
<td>Analog</td>
<td></td>
</tr>
<tr>
<td>RF6/AN11/C1INA</td>
<td>12</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RF6</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 11.</td>
</tr>
<tr>
<td>AN11</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input A.</td>
</tr>
<tr>
<td>C1INA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RF7/SS1</td>
<td>11</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RF7</td>
<td></td>
<td>I</td>
<td>TTL</td>
<td>SPI slave select input.</td>
</tr>
<tr>
<td>SS1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note 1:** Default assignment for ECCP2/P2A when Configuration bit, CCP2MX, is set.

**Note 2:** Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared.
### PORTG

PORTG is a bidirectional I/O port.

#### RG0/PMA8/ECCP3/P3A

- **RG0**: Digital I/O.
- **PMA8**: Parallel Master Port address.
- **ECCP3**: Capture 3 input/Compare 3 output/PWM3 output.
- **P3A**: ECCP3 PWM output A.

#### RG1/PMA7/TX2/CK2

- **RG1**: Digital I/O.
- **PMA7**: Parallel Master Port address.
- **TX2**: EUSART2 asynchronous transmit.
- **CK2**: EUSART2 synchronous clock (see related RX2/DT2).

#### RG2/PMA6/RX2/DT2

- **RG2**: Digital I/O.
- **PMA6**: Parallel Master Port address.
- **RX2**: EUSART2 asynchronous receive.
- **DT2**: EUSART2 synchronous data (see related TX2/CK2).

#### RG3/PMCS1/CCP4/P3D

- **RG3**: Digital I/O.
- **PMCS1**: Parallel Master Port chip select 1.
- **CCP4**: Capture 4 input/Compare 4 output/PWM4 output.
- **P3D**: ECCP3 PWM output D.

#### RG4/PMCS2/CCP5/P1D

- **RG4**: Digital I/O.
- **PMCS2**: Parallel Master Port chip select 2.
- **CCP5**: Capture 5 input/Compare 5 output/PWM5 output.
- **P1D**: ECCP1 PWM output D.

### VSS, VDD, AVSS, AVDD, ENVREG

- **VSS**: Ground reference for logic and I/O pins.
- **VDD**: Positive supply for peripheral digital logic and I/O pins.
- **AVSS**: Ground reference for analog modules.
- **AVDD**: Positive supply for analog modules.
- **ENVREG**: Enable for on-chip voltage regulator.

### VDDCORE/VCAP

- **VDDCORE**: Core logic power or external filter capacitor connection.
- **VCAP**: External filter capacitor connection (regulator enabled).

### Legend

- **TTL** = TTL compatible input
- **CMOS** = CMOS compatible input or output
- **ST** = Schmitt Trigger input with CMOS levels
- **Analog** = Analog input
- **I** = Input
- **O** = Output
- **P** = Power
- **OD** = Open-Drain (no P diode to VDD)

### Note

1. Default assignment for ECCP2/P2A when Configuration bit, CCP2MX, is set.
2. Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared.
**TABLE 1-4: PIC18F8XJ1X PINOUT I/O DESCRIPTIONS**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCLR</td>
<td>9</td>
<td>I</td>
<td>ST</td>
<td>Master Clear (Reset) input. This pin is an active-low Reset to the device.</td>
</tr>
<tr>
<td>OSC1/CLKI/RA7</td>
<td>49</td>
<td>I</td>
<td>ST</td>
<td>Oscillator crystal or external clock input. Available only in external oscillator modes (EC/ECPLL and HS/HSPLL). Main oscillator input connection.</td>
</tr>
<tr>
<td>OSC1</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>Main oscillator input connection. Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; CMOS otherwise.</td>
</tr>
<tr>
<td>CLKI</td>
<td></td>
<td>I</td>
<td>CMOS</td>
<td>Main clock input connection. External clock source input. Always associated with pin function OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)</td>
</tr>
<tr>
<td>RA7</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>General purpose I/O pin. Available only in INTIO2 and INTPLL2 Oscillator modes.</td>
</tr>
<tr>
<td>OSC2/CLKO/RA6</td>
<td>50</td>
<td>O</td>
<td>—</td>
<td>Oscillator crystal or clock output. Available only in external oscillator modes (EC/ECPLL and HS/HSPLL). Main oscillator feedback output connection.</td>
</tr>
<tr>
<td>OSC2</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.</td>
</tr>
<tr>
<td>CLKO</td>
<td></td>
<td>O</td>
<td>—</td>
<td>System cycle clock output (Fosc/4). In EC, ECPLL, INTIO1 and INTPLL1 Oscillator modes, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.</td>
</tr>
<tr>
<td>RA6</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>General purpose I/O pin. Available only in INTIO and INTPLL Oscillator modes.</td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- ST = Schmitt Trigger input with CMOS levels
- I = Input
- CMOS = CMOS compatible input or output
- O = Output
- Analog = Analog input
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note 1:**
- Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).
- Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).
- Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).
- Alternate assignment for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).
- Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).
- Default assignment for PMP data and control pins when PMPMX Configuration bit is set.
- Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
TABLE 1-4: PIC18F8XJ1X PINOUT I/O DESCRIPTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA is a bidirectional I/O port.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA0/AN0</td>
<td>30</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA0</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 0.</td>
</tr>
<tr>
<td>AN0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA1/AN1</td>
<td>29</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA1</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 1.</td>
</tr>
<tr>
<td>AN1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RA2/AN2/VREF-</td>
<td>28</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA2</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 2.</td>
</tr>
<tr>
<td>AN2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF-</td>
<td></td>
<td>I</td>
<td></td>
<td>A/D reference voltage (low) input.</td>
</tr>
<tr>
<td>RA3/AN3/VREF+</td>
<td>27</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA3</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 3.</td>
</tr>
<tr>
<td>AN3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF+</td>
<td></td>
<td>I</td>
<td></td>
<td>A/D reference voltage (high) input.</td>
</tr>
<tr>
<td>RA4/PMD5/T0CKI</td>
<td>34</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA4</td>
<td></td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMD5(7)</td>
<td></td>
<td></td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>T0CKI</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>Timer0 external clock input.</td>
</tr>
<tr>
<td>RA5/PMD4/AN4</td>
<td>33</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RA5</td>
<td></td>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMD4(7)</td>
<td></td>
<td></td>
<td>TTL</td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>AN4</td>
<td></td>
<td>I</td>
<td></td>
<td>Analog input 4.</td>
</tr>
<tr>
<td>RA6</td>
<td></td>
<td></td>
<td></td>
<td>See the OSC2/CLKO/RA6 pin.</td>
</tr>
<tr>
<td>RA7</td>
<td></td>
<td></td>
<td></td>
<td>See the OSC1/CLKI/RA7 pin.</td>
</tr>
</tbody>
</table>

Legend:  
TTL = TTL compatible input  
CMOS = CMOS compatible input or output  
ST = Schmitt Trigger input with CMOS levels  
Analog = Analog input  
I = Input  
O = Output  
P = Power  
OD = Open-Drain (no P diode to VDD)

Note 1: Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).
2: Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).
3: Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).
4: Alternate assignment for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).
5: Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).
6: Default assignment for PMP data and control pins when PMPMX Configuration bit is set.
7: Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB0/FLT0/INT0</td>
<td>58</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB0</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>ECCP1/2/3 Fault input.</td>
</tr>
<tr>
<td>FLT0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INT0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RB1/INT1/PMA4</td>
<td>57</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB1</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>External interrupt 0.</td>
</tr>
<tr>
<td>INT1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMA4</td>
<td></td>
<td>O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB2/INT2/PMA3</td>
<td>56</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB2</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>External interrupt 2.</td>
</tr>
<tr>
<td>INT2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMA3</td>
<td></td>
<td>O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB3/INT3/PMA2/ECCP2/P2A</td>
<td>55</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB3</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>External interrupt 3.</td>
</tr>
<tr>
<td>INT3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMA2</td>
<td></td>
<td>O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>ECCP2(1)</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Capture 2 input/Compare 2 output/PWM2 output.</td>
</tr>
<tr>
<td>P2A(1)</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP2 PWM output A.</td>
</tr>
<tr>
<td>RB4/KBI0/PMA1</td>
<td>54</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB4</td>
<td></td>
<td>I</td>
<td>TTL</td>
<td>Interrupt-on-change pin.</td>
</tr>
<tr>
<td>KBI0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMA1</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB5/KBI1/PMA0</td>
<td>53</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB5</td>
<td></td>
<td>I</td>
<td>TTL</td>
<td>Interrupt-on-change pin.</td>
</tr>
<tr>
<td>KBI1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMA0</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RB6/KBI2/PGC</td>
<td>52</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB6</td>
<td></td>
<td>I</td>
<td>TTL</td>
<td>Interrupt-on-change pin.</td>
</tr>
<tr>
<td>KBI2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGC</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>In-Circuit Debugger and ICSP™ programming clock pin.</td>
</tr>
<tr>
<td>RB7/KBI3/PGD</td>
<td>47</td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RB7</td>
<td></td>
<td>I</td>
<td>TTL</td>
<td>Interrupt-on-change pin.</td>
</tr>
<tr>
<td>KBI3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PGD</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>In-Circuit Debugger and ICSP programming data pin.</td>
</tr>
</tbody>
</table>

Legend:  
TTL = TTL compatible input  
CMOS = CMOS compatible input or output  
ST = Schmitt Trigger input with CMOS levels  
Analog = Analog input  
I = Input  
O = Output  
P = Power  
OD = Open-Drain (no P diode to VDD)  

Note:  
1: Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).  
2: Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).  
3: Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).  
4: Alternate assignment for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).  
5: Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).  
6: Default assignment for PMP data and control pins when PMPMX Configuration bit is set.  
7: Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
PORTC is a bidirectional I/O port.

**RC0/T1OSO/T13CKI**
- **RC0**: I/O, ST
- **T1OSO**: O
- **T13CKI**: I
  - Digital I/O.
  - Timer1 oscillator output.

**RC1/T1OSI/ECCP2/P2A**
- **RC1**: I/O, ST
- **T1OSI**: I, CMOS
- **ECCP2** (2): I/O, ST
  - Capture 2 input/Compare 2 output/PWM2 output.
  - ECCP2 PWM output A.
- **P2A** (2): O
  - Capture 2 input/Compare 2 output/PWM2 output.

**RC2/ECCP1/P1A**
- **RC2**: I/O, ST
- **ECCP1**: I/O, ST
  - Capture 1 input/Compare 1 output/PWM1 output.
  - ECCP1 PWM output A.
- **P1A**: O
  - Energize 1 input/Compare 1 output/PWM1 output.

**RC3/SCK1/SCL1**
- **RC3**: I/O, ST
- **SCK1**: I/O, ST
- **SCL1**: I/O, ST
  - Synchronous serial clock input/output for SPI mode.
  - Synchronous serial clock input/output for I²C™ mode.

**RC4/SDI1/SDA1**
- **RC4**: I/O, ST
- **SDI1**: O
- **SDA1**: I/O, ST
  - SPI data in.
  - I²C data I/O.

**RC5/SDO1**
- **RC5**: I/O, ST
- **SDO1**: O
  - SPI data out.

**RC6/TX1/CK1**
- **RC6**: I/O, ST
- **TX1**: O
- **CK1**: I/O, ST
  - EUSART1 asynchronous transmit.
  - EUSART1 synchronous clock (see related RX1/DT1).

**RC7/RX1/DT1**
- **RC7**: I/O, ST
- **RX1**: I
- **DT1**: I/O, ST
  - EUSART1 asynchronous receive.
  - EUSART1 synchronous data (see related TX1/CK1).

**Legend:**
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note:**
1: Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).
2: Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).
3: Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).
4: Alternate assignment for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).
5: Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).
6: Default assignment for PMP data and control pins when PMPMX Configuration bit is set.
7: Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
### TABLE 1-4: PIC18F8XJ1X PINOUT I/O DESCRIPTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD0/AD0/PMD0</td>
<td>72</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AD0</td>
<td></td>
<td>TTL</td>
<td></td>
<td>External memory address/data 0.</td>
</tr>
<tr>
<td>PMD0(6)</td>
<td></td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>RD1/AD1/PMD1</td>
<td>69</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AD1</td>
<td></td>
<td>TTL</td>
<td></td>
<td>External memory address/data 1.</td>
</tr>
<tr>
<td>PMD1(6)</td>
<td></td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>RD2/AD2/PMD2</td>
<td>68</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AD2</td>
<td></td>
<td>TTL</td>
<td></td>
<td>External memory address/data 2.</td>
</tr>
<tr>
<td>PMD2(6)</td>
<td></td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>RD3/AD3/PMD3</td>
<td>67</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AD3</td>
<td></td>
<td>TTL</td>
<td></td>
<td>External memory address/data 3.</td>
</tr>
<tr>
<td>PMD3(6)</td>
<td></td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>AD4</td>
<td></td>
<td>TTL</td>
<td></td>
<td>External memory address/data 4.</td>
</tr>
<tr>
<td>PMD4(6)</td>
<td></td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>SDO2</td>
<td></td>
<td>O</td>
<td>—</td>
<td>SPI data out.</td>
</tr>
<tr>
<td>RD5/AD5/PMD5/SDI2/SDA2</td>
<td>65</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AD5</td>
<td></td>
<td>TTL</td>
<td></td>
<td>External memory address/data 5.</td>
</tr>
<tr>
<td>PMD5(6)</td>
<td></td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>SDI2</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>SPI data in.</td>
</tr>
<tr>
<td>SDA2</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>I2C™ data I/O.</td>
</tr>
<tr>
<td>AD6</td>
<td></td>
<td>TTL</td>
<td></td>
<td>External memory address/data 6.</td>
</tr>
<tr>
<td>PMD6(6)</td>
<td></td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>SCK2</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Synchronous serial clock input/output for SPI mode.</td>
</tr>
<tr>
<td>SCL2</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Synchronous serial clock input/output for I2C mode.</td>
</tr>
<tr>
<td>RD7/AD7/PMD7/SS2</td>
<td>63</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AD7</td>
<td></td>
<td>TTL</td>
<td></td>
<td>External memory address/data 7.</td>
</tr>
<tr>
<td>PMD7(6)</td>
<td></td>
<td>TTL</td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>SS2</td>
<td></td>
<td>I</td>
<td>TTL</td>
<td>SPI slave select input.</td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note 1:** Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).

**Note 2:** Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).

**Note 3:** Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).

**Note 4:** Alternate assignment for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).

**Note 5:** Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).

**Note 6:** Default assignment for PMP data and control pins when PMPMX Configuration bit is set.

**Note 7:** Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE0/AD8/PMRD/P2D</td>
<td>4</td>
<td>I/O</td>
<td>ST</td>
<td>PORTE is a bidirectional I/O port.</td>
</tr>
<tr>
<td>RE0</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AD8</td>
<td></td>
<td>I/O</td>
<td></td>
<td>External memory address/data 8.</td>
</tr>
<tr>
<td>PMRD&lt;sup&gt;(6)&lt;/sup&gt;</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port read strobe.</td>
</tr>
<tr>
<td>P2D</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP2 PWM output D.</td>
</tr>
<tr>
<td>RE1/AD9/PMWR/P2C</td>
<td>3</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE1</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>External memory address/data 9.</td>
</tr>
<tr>
<td>AD9</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port write strobe.</td>
</tr>
<tr>
<td>PMWR&lt;sup&gt;(6)&lt;/sup&gt;</td>
<td></td>
<td>I/O</td>
<td></td>
<td>ECCP2 PWM output C.</td>
</tr>
<tr>
<td>P2C</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE2/AD10/PMBE/P2B</td>
<td>78</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE2</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>External memory address/data 10.</td>
</tr>
<tr>
<td>AD10</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port byte enable.</td>
</tr>
<tr>
<td>PMBE&lt;sup&gt;(6)&lt;/sup&gt;</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP2 PWM output B.</td>
</tr>
<tr>
<td>P2B</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE3/AD11/PMA13/P3C/REFO</td>
<td>77</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE3</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>External memory address/data 11.</td>
</tr>
<tr>
<td>AD11</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA13</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP3 PWM output C.</td>
</tr>
<tr>
<td>P3C&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REFO</td>
<td></td>
<td>O</td>
<td></td>
<td>Reference clock out.</td>
</tr>
<tr>
<td>RE4/AD12/PMA12/P3B</td>
<td>76</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE4</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>External memory address/data 12.</td>
</tr>
<tr>
<td>AD12</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA12</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP3 PWM output B.</td>
</tr>
<tr>
<td>P3B&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE5/AD13/PMA11/P1C</td>
<td>75</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE5</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>External memory address/data 13.</td>
</tr>
<tr>
<td>AD13</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA11</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP1 PWM output C.</td>
</tr>
<tr>
<td>P1C&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE6/AD14/PMA10/P1B</td>
<td>74</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE6</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>External memory address/data 14.</td>
</tr>
<tr>
<td>AD14</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA10</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP1 PWM output B.</td>
</tr>
<tr>
<td>P1B&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RE7/AD15/PMA9/ECCP2/P2A</td>
<td>73</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RE7</td>
<td></td>
<td>I/O</td>
<td>TTL</td>
<td>External memory address/data 15.</td>
</tr>
<tr>
<td>AD15</td>
<td></td>
<td>I/O</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMA9</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP2 PWM output A.</td>
</tr>
<tr>
<td>ECCP2&lt;sup&gt;(4)&lt;/sup&gt;</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Capture 2 input/Compare 2 output/PWM2 output.</td>
</tr>
<tr>
<td>P2A&lt;sup&gt;(4)&lt;/sup&gt;</td>
<td></td>
<td>O</td>
<td></td>
<td>ECCP2 PWM output A.</td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note:**
1: Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).
2: Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).
3: Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).
4: Alternate assignments for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).
5: Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).
6: Default assignment for PMP data and control pins when PMPMX Configuration bit is set.
7: Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF1/AN6/C2OUT</td>
<td>23</td>
<td>I/O</td>
<td>ST</td>
<td>PORTF is a bidirectional I/O port.</td>
</tr>
<tr>
<td>RF1</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>AN6</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 6.</td>
</tr>
<tr>
<td>C2OUT</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Comparator 2 output.</td>
</tr>
<tr>
<td>RF2/PMA5/AN7/C1OUT</td>
<td>18</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RF2</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 7.</td>
</tr>
<tr>
<td>PMA5</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Comparator 1 output.</td>
</tr>
<tr>
<td>AN7</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input B.</td>
</tr>
<tr>
<td>C1OUT</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Comparator 1 input B.</td>
</tr>
<tr>
<td>RF3/AN8/C2INB</td>
<td>17</td>
<td>I</td>
<td>ST</td>
<td>Digital input.</td>
</tr>
<tr>
<td>RF3</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 8.</td>
</tr>
<tr>
<td>AN8</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input B.</td>
</tr>
<tr>
<td>C2INB</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input B.</td>
</tr>
<tr>
<td>RF4/AN9/C2INA</td>
<td>16</td>
<td>I</td>
<td>ST</td>
<td>Digital input.</td>
</tr>
<tr>
<td>RF4</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input A.</td>
</tr>
<tr>
<td>AN9</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input B.</td>
</tr>
<tr>
<td>C2INA</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input A.</td>
</tr>
<tr>
<td>RF5/PMD2/AN10/C1INB/CVREF</td>
<td>15</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RF5</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 10.</td>
</tr>
<tr>
<td>PMD2(7)</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input B.</td>
</tr>
<tr>
<td>AN10</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator reference voltage output.</td>
</tr>
<tr>
<td>C1INB</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input B.</td>
</tr>
<tr>
<td>CVREF</td>
<td></td>
<td>O</td>
<td>Analog</td>
<td>Comparator reference voltage output.</td>
</tr>
<tr>
<td>RF6/PMD1/AN11/C1INA</td>
<td>14</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RF6</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Analog input 11.</td>
</tr>
<tr>
<td>PMD1(7)</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input A.</td>
</tr>
<tr>
<td>AN11</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input A.</td>
</tr>
<tr>
<td>C1INA</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input A.</td>
</tr>
<tr>
<td>RF7/PMD0/SS1</td>
<td>13</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RF7</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>PMD0(7)</td>
<td></td>
<td>I</td>
<td>TTL</td>
<td>SPI slave select input.</td>
</tr>
<tr>
<td>SS1</td>
<td></td>
<td>I</td>
<td>TTL</td>
<td>SPI slave select input.</td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note:**
1. Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).
2. Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).
3. Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).
4. Alternate assignment for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).
5. Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).
6. Default assignment for PMP data and control pins when PMPMX Configuration bit is set.
7. Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
**TABLE 1-4: PIC18F8XJ1X PINOUT I/O DESCRIPTIONS (CONTINUED)**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RG0/PMA8/ECCP3/P3A</td>
<td>5</td>
<td>I/O</td>
<td>ST</td>
<td>PORTG is a bidirectional I/O port.</td>
</tr>
<tr>
<td>RG0</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>PMA8</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>ECCP3</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Capture 3 input/Compare 3 output/PWM3 output.</td>
</tr>
<tr>
<td>P3A</td>
<td></td>
<td>O</td>
<td>—</td>
<td>ECCP3 PWM output A.</td>
</tr>
<tr>
<td>RG1/PMA7/TX2/CK2</td>
<td>6</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RG1</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>PMA7</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>TX2</td>
<td></td>
<td>O</td>
<td>—</td>
<td>EUSART2 asynchronous transmit.</td>
</tr>
<tr>
<td>CK2</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>EUSART2 synchronous clock (see related RX2/DT2).</td>
</tr>
<tr>
<td>RG2/PMA6/RX2/DT2</td>
<td>7</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RG2</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>PMA6</td>
<td></td>
<td>I/O</td>
<td>—</td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RX2</td>
<td></td>
<td>I</td>
<td>ST</td>
<td>EUSART2 asynchronous receive.</td>
</tr>
<tr>
<td>DT2</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>EUSART2 synchronous data (see related TX2/CK2).</td>
</tr>
<tr>
<td>RG3/PMCS1/CCP4/P3D</td>
<td>8</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RG3</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>PMCS1</td>
<td></td>
<td>I/O</td>
<td>—</td>
<td>Parallel Master Port chip select 1.</td>
</tr>
<tr>
<td>CCP4</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Capture 4 input/Compare 4 output/PWM4 output.</td>
</tr>
<tr>
<td>P3D</td>
<td></td>
<td>O</td>
<td>—</td>
<td>ECCP3 PWM output D.</td>
</tr>
<tr>
<td>RG4/PMCS2/CCP5/P1D</td>
<td>10</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RG4</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>PMCS2</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Parallel Master Port chip select 2.</td>
</tr>
<tr>
<td>CCP5</td>
<td></td>
<td>I/O</td>
<td>ST</td>
<td>Capture 5 input/Compare 5 output/PWM5 output.</td>
</tr>
<tr>
<td>P1D</td>
<td></td>
<td>O</td>
<td>—</td>
<td>ECCP1 PWM output D.</td>
</tr>
</tbody>
</table>

**Legend:**
- **TTL** = TTL compatible input
- **CMOS** = CMOS compatible input or output
- **ST** = Schmitt Trigger input with CMOS levels
- **Analog** = Analog input
- **I** = Input
- **O** = Output
- **P** = Power
- **OD** = Open-Drain (no P diode to VDD)

**Note 1:** Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).
**Note 2:** Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).
**Note 3:** Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).
**Note 4:** Alternate assignment for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).
**Note 5:** Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).
**Note 6:** Default assignment for PMP data and control pins when PMPMX Configuration bit is set.
**Note 7:** Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
## TABLE 1-4: PIC18F8XJ1X PINOUT I/O DESCRIPTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RH0/A16</td>
<td>79</td>
<td>I/O</td>
<td>ST</td>
<td>PORTH is a bidirectional I/O port.</td>
</tr>
<tr>
<td>RH0</td>
<td></td>
<td>O</td>
<td>TTL</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>A16</td>
<td></td>
<td></td>
<td></td>
<td>External memory address/data 16.</td>
</tr>
<tr>
<td>RH1/A17</td>
<td>80</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RH1</td>
<td></td>
<td>O</td>
<td>TTL</td>
<td>External memory address/data 17.</td>
</tr>
<tr>
<td>A17</td>
<td></td>
<td></td>
<td></td>
<td>External memory address/data 17.</td>
</tr>
<tr>
<td>RH2/A18/PMD7</td>
<td>1</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>A18</td>
<td></td>
<td>O</td>
<td>TTL</td>
<td>External memory address/data 18.</td>
</tr>
<tr>
<td>PMD7</td>
<td>(7)</td>
<td></td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>RH3/A19/PMD6</td>
<td>2</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>A19</td>
<td></td>
<td>O</td>
<td>TTL</td>
<td>External memory address/data 19.</td>
</tr>
<tr>
<td>PMD6</td>
<td>(7)</td>
<td></td>
<td></td>
<td>Parallel Master Port data.</td>
</tr>
<tr>
<td>RH4/PMD3/AN12/P3C/C2INC</td>
<td>22</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>P3C/C2INC</td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>RH4</td>
<td></td>
<td>O</td>
<td>TTL</td>
<td>Analog input 12.</td>
</tr>
<tr>
<td>PMD3</td>
<td>(7)</td>
<td></td>
<td></td>
<td>ECCP3 PWM output C.</td>
</tr>
<tr>
<td>AN12</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input C.</td>
</tr>
<tr>
<td>P3C</td>
<td>(5)</td>
<td>O</td>
<td>—</td>
<td>ECCP3 PWM output B.</td>
</tr>
<tr>
<td>C2INC</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input D.</td>
</tr>
<tr>
<td>RH5/PMBE/AN13/P3B/C2IND</td>
<td>21</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>P3B/C2IND</td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port byte enable.</td>
</tr>
<tr>
<td>RH5</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Analog input 13.</td>
</tr>
<tr>
<td>PMBE</td>
<td>(7)</td>
<td></td>
<td></td>
<td>ECCP3 PWM output B.</td>
</tr>
<tr>
<td>AN13</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input D.</td>
</tr>
<tr>
<td>P3B</td>
<td>(6)</td>
<td>O</td>
<td>—</td>
<td>ECCP3 PWM output B.</td>
</tr>
<tr>
<td>C2IND</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 2 input D.</td>
</tr>
<tr>
<td>RH6/PMRD/AN14/P1C/C1INC</td>
<td>20</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>P1C/C1INC</td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port read strobe.</td>
</tr>
<tr>
<td>RH6</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Analog input 14.</td>
</tr>
<tr>
<td>PMRD</td>
<td>(7)</td>
<td></td>
<td></td>
<td>ECCP1 PWM output C.</td>
</tr>
<tr>
<td>AN14</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input C.</td>
</tr>
<tr>
<td>P1C</td>
<td>(5)</td>
<td>O</td>
<td>—</td>
<td>ECCP1 PWM output C.</td>
</tr>
<tr>
<td>C1INC</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>Comparator 1 input C.</td>
</tr>
<tr>
<td>RH7/PMWR/AN15/P1B</td>
<td>19</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>P1B</td>
<td>(6)</td>
<td>O</td>
<td>—</td>
<td>ECCP1 PWM output B.</td>
</tr>
<tr>
<td>RH7</td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port write strobe.</td>
</tr>
<tr>
<td>PMWR</td>
<td>(7)</td>
<td></td>
<td></td>
<td>Analog input 15.</td>
</tr>
<tr>
<td>AN15</td>
<td></td>
<td>I</td>
<td>Analog</td>
<td>ECCP1 PWM output B.</td>
</tr>
</tbody>
</table>

**Legend:**
- TTL = TTL compatible input
- CMOS = CMOS compatible input or output
- ST = Schmitt Trigger input with CMOS levels
- Analog = Analog input
- I = Input
- O = Output
- P = Power
- OD = Open-Drain (no P diode to VDD)

**Note:**
1. Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).
2. Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).
3. Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).
4. Alternate assignment for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).
5. Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).
6. Default assignment for PMP data and control pins when PMPMX Configuration bit is set.
7. Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
TABLE 1-4: PIC18F8XJ1X PINOUT I/O DESCRIPTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Pin Number</th>
<th>Pin Type</th>
<th>Buffer Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RJ0/ALE</td>
<td>62</td>
<td>I/O</td>
<td>ST</td>
<td>PORTJ is a bidirectional I/O port.</td>
</tr>
<tr>
<td>RJ0/ALE</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RJ1/Œ</td>
<td>61</td>
<td>I/O</td>
<td>ST</td>
<td>External memory address latch enable.</td>
</tr>
<tr>
<td>RJ1/Œ</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RJ2/WRL</td>
<td>60</td>
<td>I/O</td>
<td>ST</td>
<td>External memory output enable.</td>
</tr>
<tr>
<td>RJ2/WRL</td>
<td></td>
<td>O</td>
<td>—</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RJ3/WRH</td>
<td>59</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RJ3/WRH</td>
<td></td>
<td>O</td>
<td>—</td>
<td>External memory write high control.</td>
</tr>
<tr>
<td>RJ4/BA0</td>
<td>39</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RJ4/BA0</td>
<td></td>
<td>O</td>
<td>—</td>
<td>External memory byte address 0 control.</td>
</tr>
<tr>
<td>RJ5/Œ</td>
<td>40</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RJ5/Œ</td>
<td></td>
<td>O</td>
<td>—</td>
<td>External memory chip enable control.</td>
</tr>
<tr>
<td>RJ6/LB</td>
<td>41</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RJ6/LB</td>
<td></td>
<td>O</td>
<td>—</td>
<td>External memory low byte control.</td>
</tr>
<tr>
<td>RJ7/UB</td>
<td>42</td>
<td>I/O</td>
<td>ST</td>
<td>Digital I/O.</td>
</tr>
<tr>
<td>RJ7/UB</td>
<td></td>
<td>O</td>
<td>—</td>
<td>External memory high byte control.</td>
</tr>
<tr>
<td>VSS</td>
<td>11, 31, 51, 70</td>
<td>P</td>
<td>—</td>
<td>Ground reference for logic and I/O pins.</td>
</tr>
<tr>
<td>VDD</td>
<td>32, 48, 71</td>
<td>P</td>
<td>—</td>
<td>Positive supply for peripheral digital logic and I/O pins.</td>
</tr>
<tr>
<td>AVss</td>
<td>26</td>
<td>P</td>
<td>—</td>
<td>Ground reference for analog modules.</td>
</tr>
<tr>
<td>AVDD</td>
<td>25</td>
<td>P</td>
<td>—</td>
<td>Positive supply for analog modules.</td>
</tr>
<tr>
<td>ENVREG</td>
<td>24</td>
<td>I</td>
<td>ST</td>
<td>Enable for on-chip voltage regulator.</td>
</tr>
<tr>
<td>VDDCORE/VCAP</td>
<td>12</td>
<td>P</td>
<td>—</td>
<td>Core logic power or external filter capacitor connection.</td>
</tr>
<tr>
<td>VDDCORE</td>
<td></td>
<td>P</td>
<td>—</td>
<td>Positive supply for microcontroller core logic (regulator disabled).</td>
</tr>
<tr>
<td>VCAP</td>
<td></td>
<td>P</td>
<td>—</td>
<td>External filter capacitor connection (regulator enabled).</td>
</tr>
</tbody>
</table>

Legend: TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
Analog = Analog input
I = Input
O = Output
P = Power
OD = Open-Drain (no P diode to Vdd)

Note 1: Alternate assignment for ECCP2/P2A when Configuration bit, CCP2MX, is cleared (Extended Microcontroller mode).

2: Default assignment for ECCP2/P2A for all devices in all operating modes (CCP2MX is set).
3: Default assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is set).
4: Alternate assignment for ECCP2/P2A when CCP2MX is cleared (Microcontroller mode).
5: Alternate assignments for P1B/P1C/P3B/P3C (ECCPMX Configuration bit is cleared).
6: Default assignment for PMP data and control pins when PMPMX Configuration bit is set.
7: Alternate assignment for PMP data and control pins when PMPMX Configuration bit is cleared (programmed).
2.0 OSCILLATOR CONFIGURATIONS

2.1 Oscillator Types

The PIC18F87J11 family of devices can be operated in eight different oscillator modes:

1. HS  High-Speed Crystal/Resonator
2. HSPLL  High-Speed Crystal/Resonator with Software PLL Control
3. EC  External Clock with FOSC/4 Output
4. ECPLL  External Clock with Software PLL Control
5. INTIO1  Internal Oscillator Block with FOSC/4 Output on RA6 and I/O on RA7
6. INTIO2  Internal Oscillator Block with I/O on RA6 and RA7
7. INTPLL1  Internal Oscillator Block with Software PLL Control, FOSC/4 Output on RA6 and I/O on RA7
8. INTPLL2  Internal Oscillator Block with Software PLL Control and I/O on RA6 and RA7

All of these modes are selected by the user by programming the FOSC2:FOSC0 Configuration bits.

In addition, PIC18F87J11 Family devices can switch between different clock sources, either under software control or automatically under certain conditions. This allows for additional power savings by managing device clock speed in real time without resetting the application.

The clock sources for the PIC18F87J11 family of devices are shown in Figure 2-1.

**FIGURE 2-1: PIC18F87J11 FAMILY CLOCK DIAGRAM**
2.2 Control Registers

The OSCCON register (Register 2-1) controls the main aspects of the device clock's operation. It selects the oscillator type to be used, which of the power-managed modes to invoke and the output frequency of the INTOSC source. It also provides status on the oscillators.

The OSCTUNE register (Register 2-2) controls the tuning and operation of the internal oscillator block. It also implements the PLLEN bits which control the operation of the Phase Locked Loop (PLL) (see Section 2.4.3 “PLL Frequency Multiplier”).

### REGISTER 2-1: OSCCON: OSCILLATOR CONTROL REGISTER(1)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-0</th>
<th>(2)</th>
<th>(2)</th>
<th>(2)</th>
<th>(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLEN</td>
<td>IRCF2(3)</td>
<td>IRCF1(3)</td>
<td>IRCF0(3)</td>
<td>OSTS</td>
<td>—</td>
<td>SCS1(5)</td>
<td>SCS0(5)</td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 6-4</td>
<td>bit 3</td>
<td>bit 2</td>
<td>bit 1-0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Legend:

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

#### bit 7: IDLEN

- **IDLEN**: Idle Enable bit
- 1 = Device enters an Idle mode when a SLEEP instruction is executed
- 0 = Device enters Sleep mode when a SLEEP instruction is executed

#### bit 6-4: IRCF2:IRCF0

- **IRCF2:IRCF0**: INTOSC Source Frequency Select bits(3)
- 111 = 8 MHz (INTOSC drives clock directly)
- 110 = 4 MHz (default)
- 101 = 2 MHz
- 100 = 1 MHz
- 011 = 500 kHz
- 010 = 250 kHz
- 001 = 125 kHz
- 000 = 31 kHz (from either INTOSC/256 or INTRC)(4)

#### bit 3: OSTS

- **OSTS**: Oscillator Start-up Timer Time-out Status bit(2)
- 1 = Oscillator Start-up Timer (OST) time-out has expired; primary oscillator is running
- 0 = Oscillator Start-up Timer (OST) time-out is running; primary oscillator is not ready

#### bit 2: Unimplemented

- Read as ‘1’

#### bit 1-0: SCS1:SCS0

- **SCS1:SCS0**: System Clock Select bits(6)
- 11 = Internal oscillator block
- 10 = Primary oscillator
- 01 = Timer1 oscillator
- 00 = Default primary oscillator (as defined by FOSC2:FOSC0 Configuration bits)

#### Note 1:

- Default (legacy) SFR at this address, available when WDTCON<4> = 0.
- 2: Reset state depends on state of the IESO Configuration bit.
- 3: Modifying these bits will cause an immediate clock frequency switch if the internal oscillator is providing the device clocks.
- 4: Source selected by the INTSRC bit (OSCTUNE<7>), see text.
- 5: Modifying these bits will cause an immediate clock source switch.
2.3 Clock Sources and Oscillator Switching

Essentially, PIC18F87J11 Family devices have three independent clock sources:
• Primary oscillators
• Secondary oscillators
• Internal oscillator

The primary oscillators can be thought of as the main device oscillators. These are any external oscillators connected to the OSC1 and OSC2 pins, and include the External Crystal and Resonator modes and the External Clock modes. If selected by the FOSC2:FOSC0 Configuration bits, the internal oscillator block (either the 31 kHz INTRC or the 8 MHz INTOSC source) may be considered a primary oscillator. The particular mode is defined by the FOSC Configuration bits. The details of these modes are covered in Section 2.4 “External Oscillator Modes”.

The secondary oscillators are external clock sources that are not connected to the OSC1 or OSC2 pins. These sources may continue to operate even after the controller is placed in a power-managed mode. PIC18F87J11 Family devices offer the Timer1 oscillator as a secondary oscillator source. This oscillator, in all power-managed modes, is often the time base for functions such as a Real-Time Clock (RTC). The Timer1 oscillator is discussed in greater detail in Section 13.0 “Timer1 Module”.

In addition to being a primary clock source in some circumstances, the internal oscillator is available as a power-managed mode clock source. The INTRC source is also used as the clock source for several special features, such as the WDT and Fail-Safe Clock Monitor. The internal oscillator block is discussed in more detail in Section 2.5 “Internal Oscillator Block”.

The PIC18F87J11 Family includes features that allow the device clock source to be switched from the main oscillator, chosen by device configuration, to one of the alternate clock sources. When an alternate clock source is enabled, various power-managed operating modes are available.
2.3.1 CLOCK SOURCE SELECTION

The System Clock Select bits, SCS1:SCS0 (OSCCON<1:0>), select the clock source. The available clock sources are the primary clock defined by the FOSC2:FOSC0 Configuration bits, the secondary clock (Timer1 oscillator) and the internal oscillator. The clock source changes after one or more of the bits are written to, following a brief clock transition interval.

The OSTS (OSCCON<3>) and T1RUN (T1CON<6>) bits indicate which clock source is currently providing the device clock. The OSTS bit indicates that the Oscillator Start-up Timer (OST) has timed out and the primary clock is providing the device clock in primary clock modes. The T1RUN bit indicates when the Timer1 oscillator is providing the device clock in secondary clock modes. In power-managed modes, only one of these bits will be set at any time. If neither of these bits is set, the INTRC is providing the clock, or the internal oscillator has just started and is not yet stable.

The IDLEN bit determines if the device goes into Sleep mode or one of the Idle modes when the SLEEP instruction is executed.

The use of the flag and control bits in the OSCCON register is discussed in more detail in Section 3.0 “Power-Managed Modes”.

Note 1: The Timer1 oscillator must be enabled to select the secondary clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 Control register (T1CON<3>). If the Timer1 oscillator is not enabled, then any attempt to select a secondary clock source when executing a SLEEP instruction will be ignored.

2: It is recommended that the Timer1 oscillator be operating and stable before executing the SLEEP instruction or a very long delay may occur while the Timer1 oscillator starts.

2.3.1.1 System Clock Selection and Device Resets

Since the SCS bits are cleared on all forms of Reset, this means the primary oscillator defined by the FOSC2:FOSC0 Configuration bits is used as the primary clock source on device Resets. This could either be the internal oscillator block by itself, or one of the other primary clock source (HS, EC, HSPLL, ECPLL1/2 or INTPLL1/2).

In those cases when the internal oscillator block, without PLL, is the default clock on Reset, the Fast RC oscillator (INTOSC) will be used as the device clock source. It will initially start at 4 MHz; the postscaler selection that corresponds to the Reset value of the IRCF2:IRCF0 bits (110).

Regardless of which primary oscillator is selected, INTRC will always be enabled on device power-up. It serves as the clock source until the device has loaded its configuration values from memory. It is at this point that the FOSC Configuration bits are read and the oscillator selection of the operational mode is made.

Note that either the primary clock source, or the internal oscillator, will have two bit setting options for the possible values of the SCS1:SCS0 bits at any given time.

2.3.2 OSCILLATOR TRANSITIONS

PIC18F87J11 family devices contain circuitry to prevent clock “glitches” when switching between clock sources. A short pause in the device clock occurs during the clock switch. The length of this pause is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Clock transitions are discussed in greater detail in Section 3.1.2 “Entering Power-Managed Modes”. 
2.4 External Oscillator Modes

2.4.1 CRYSTAL OSCILLATOR/CERAMIC RESONATORS (HS MODES)

In HS or HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-2 shows the pin connections.

The oscillator design requires the use of a crystal rated for parallel resonant operation.

Note: Use of a crystal rated for series resonant operation may give a frequency out of the crystal manufacturer’s specifications.

TABLE 2-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

<table>
<thead>
<tr>
<th>Mode</th>
<th>Freq.</th>
<th>OSC1</th>
<th>OSC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>HS</td>
<td>8.0 MHz</td>
<td>27 pF</td>
<td>27 pF</td>
</tr>
<tr>
<td></td>
<td>16.0 MHz</td>
<td>22 pF</td>
<td>22 pF</td>
</tr>
</tbody>
</table>

Capacitor values are for design guidance only.

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application. Refer to the following application notes for oscillator specific information:

- AN588, “PIC® Microcontroller Oscillator Design Guide”
- AN826, “Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices”
- AN849, “Basic PIC® Oscillator Design”
- AN943, “Practical PIC® Oscillator Analysis and Design”
- AN949, “Making Your Oscillator Work”

See the notes following Table 2-2 for additional information.

TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

<table>
<thead>
<tr>
<th>Osc Type</th>
<th>Crystal Freq.</th>
<th>Typical Capacitor Values Tested:</th>
</tr>
</thead>
<tbody>
<tr>
<td>HS</td>
<td>4 MHz</td>
<td>C1 = 27 pF, C2 = 27 pF</td>
</tr>
<tr>
<td></td>
<td>8 MHz</td>
<td>C1 = 22 pF, C2 = 22 pF</td>
</tr>
<tr>
<td></td>
<td>20 MHz</td>
<td>C1 = 15 pF, C2 = 15 pF</td>
</tr>
</tbody>
</table>

Capacitor values are for design guidance only.

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

Refer to the Microchip application notes cited in Table 2-1 for oscillator specific information. Also see the notes following this table for additional information.

Note 1: Higher capacitance increases the stability of oscillator but also increases the start-up time.

2: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

3: Rs may be required to avoid overdriving crystals with low drive level specification.

4: Always verify oscillator performance over the VDD and temperature range that is expected for the application.

FIGURE 2-2: CRYSTAL/CERAMIC RESONATOR OPERATION (HS OR HSPLL CONFIGURATION)

Note 1: See Table 2-1 and Table 2-2 for initial values of C1 and C2.

2: A series resistor (Rs) may be required for AT strip cut crystals.

3: RF varies with the oscillator mode chosen.
2.4.2 EXTERNAL CLOCK INPUT
(EC Mdes)

The EC and ECPLL Oscillator modes require an external clock source to be connected to the OSC1 pin. There is no oscillator start-up time required after a Power-on Reset or after an exit from Sleep mode.

In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-3 shows the pin connections for the EC Oscillator mode.

Figure 2-3: EXTERNAL CLOCK INPUT OPERATION (EC CONFIGURATION)

An external clock source may also be connected to the OSC1 pin in the HS mode, as shown in Figure 2-4. In this configuration, the divide-by-4 output on OSC2 is not available. Current consumption in this configuration will be somewhat higher than EC mode, as the internal oscillator’s feedback circuitry will be enabled (in EC mode, the feedback circuit is disabled).

Figure 2-4: EXTERNAL CLOCK INPUT OPERATION (HS OSC CONFIGURATION)

2.4.3 PLL FREQUENCY MULTIPLIER

A Phase Locked Loop (PLL) circuit is provided as an option for users who want to use a lower frequency oscillator circuit, or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals, or users who require higher clock speeds from an internal oscillator.

2.4.3.1 HSPLL and ECPLL Modes

The HSPLL and ECPLL modes provide the ability to selectively run the device at 4 times the external oscillating source to produce frequencies up to 40 MHz.

The PLL is enabled by programming the FOSC2:FOSC0 Configuration bits to either ‘111’ (for ECPLL) or ‘101’ (for HSPLL). In addition, the PLLEN bit (OSCTUNE<6>) must also be set. Clearing PLLEN disables the PLL, regardless of the chosen oscillator configuration. It also allows additional flexibility for controlling the application’s clock speed in software.

Figure 2-5: PLL BLOCK DIAGRAM

2.4.3.2 PLL and INTOSC

The PLL is also available to the internal oscillator block when the internal oscillator block is configured as the primary clock source. In this configuration, the PLL is enabled in software and generates a clock output of up to 32 MHz. The operation of INTOSC with the PLL is described in Section 2.5.2 “INTPLL Modes”.
2.5 Internal Oscillator Block

The PIC18F87J11 Family of devices includes an internal oscillator block which generates two different clock signals; either can be used as the microcontroller’s clock source. This may eliminate the need for an external oscillator circuit on the OSC1 and/or OSC2 pins.

The main output is the Fast RC oscillator, or INTOSC, an 8 MHz clock source which can be used to directly drive the device clock. It also drives a postscaler, which can provide a range of clock frequencies from 31 kHz to 4 MHz. INTOSC is enabled when a clock frequency from 125 kHz to 8 MHz is selected. The INTOSC output can also be enabled when 31 kHz is selected, depending on the INTSRC bit (OSCTUNE<7>).

The other clock source is the internal RC oscillator (INTRC), which provides a nominal 31 kHz output. INTRC is enabled if it is selected as the device clock source; it is also enabled automatically when any of the following are enabled:
- Power-up Timer
- Fail-Safe Clock Monitor
- Watchdog Timer
- Two-Speed Start-up

These features are discussed in greater detail in Section 24.0 “Special Features of the CPU”.

The clock source frequency (INTOSC direct, INTOSC with postscaler or INTRC direct) is selected by configuring the IRCF bits of the OSCCON register. The default frequency on device Resets is 4 MHz.

2.5.1 INTIO Modes

Using the internal oscillator as the clock source eliminates the need for up to two external oscillator pins, which can then be used for digital I/O. Two distinct oscillator configurations, which are determined by the FOSC Configuration bits, are available:
- In INTIO1 mode, the OSC2 pin outputs Fosc/4, while OSC1 functions as RA7 (see Figure 2-6) for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6 (see Figure 2-7), both for digital input and output.

2.5.2 INTPLL Modes

The 4x Phase Locked Loop (PLL) can be used with the internal oscillator block to produce faster device clock speeds than are normally possible with the internal oscillator sources. When enabled, the PLL produces a clock speed of 16 MHz or 32 MHz.

PLL operation is controlled through software. The control bit, PLLEN (OSCTUNE<6>), is used to enable or disable its operation. The PLL is available only to INTOSC when the device is configured to use one of the INTPLL modes as the primary clock source (FOSC2:FOSC0 = 011 or 010). Additionally, the PLL will only function when the selected output frequency is either 4 MHz or 8 MHz (OSCCON<6:4> = 111 or 110).

Like the INTIO modes, there are two distinct INTPLL modes available:
- In INTPLL1 mode, the OSC2 pin outputs Fosc/4, while OSC1 functions as RA7 for digital input and output. Externally, this is identical in appearance to INTIO1 (Figure 2-6).
- In INTPLL2 mode, OSC1 functions as RA7 and OSC2 functions as RA6, both for digital input and output. Externally, this is identical to INTIO2 (Figure 2-7).
2.5.3 INTERNAL OSCILLATOR OUTPUT FREQUENCY AND TUNING

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 8 MHz. It can be adjusted in the user’s application by writing to TUN5:TUN0 (OSCTUNE<5:0>) in the OSCTUNE register (Register 2-2).

When the OSCTUNE register is modified, the INTOSC frequency will begin shifting to the new frequency. The oscillator will stabilize within 1 ms. Code execution continues during this shift and there is no indication that the shift has occurred.

The INTRC oscillator operates independently of the INTOSC source. Any changes in INTOSC across voltage and temperature are not necessarily reflected by changes in INTRC or vice versa. The frequency of INTRC is not affected by OSCTUNE.

2.5.4 INTOSC FREQUENCY DRIFT

The INTOSC frequency may drift as VDD or temperature changes, and can affect the controller operation in a variety of ways. It is possible to adjust the INTOSC frequency by modifying the value in the OSCTUNE register. Depending on the device, this may have no effect on the INTRC clock source frequency.

Tuning INTOSC requires knowing when to make the adjustment, in which direction it should be made, and in some cases, how large a change is needed. Three compensation techniques are shown here.

2.5.4.1 Compensating with the EUSART

An adjustment may be required when the EUSART begins to generate framing errors or receives data with errors while in Asynchronous mode. Framing errors indicate that the device clock frequency is too high. To adjust for this, decrement the value in OSCTUNE to reduce the clock frequency. On the other hand, errors in data may suggest that the clock speed is too low. To compensate, increment OSCTUNE to increase the clock frequency.

2.5.4.2 Compensating with the Timers

This technique compares device clock speed to some reference clock. Two timers may be used; one timer is clocked by the peripheral clock, while the other is clocked by a fixed reference source, such as the Timer1 oscillator.

Both timers are cleared, but the timer clocked by the reference generates interrupts. When an interrupt occurs, the internally clocked timer is read and both timers are cleared. If the internally clocked timer value is much greater than expected, then the internal oscillator block is running too fast. To adjust for this, decrement the OSCTUNE register.

2.5.4.3 Compensating with the CCP Module in Capture Mode

A CCP module can use free-running Timer1 (or Timer3), clocked by the internal oscillator block and an external event with a known period (i.e., AC power frequency). The time of the first event is captured in the CCPRxH:CCPRxL registers and is recorded for use later. When the second event causes a capture, the time of the first event is subtracted from the time of the second event. Since the period of the external event is known, the time difference between events can be calculated.

If the measured time is much greater than the calculated time, the internal oscillator block is running too fast. To compensate, decrement the OSCTUNE register. If the measured time is much less than the calculated time, the internal oscillator block is running too slow. To compensate, increment the OSCTUNE register.

2.6 Reference Clock Output

In addition to the Fosc/4 clock output in certain oscillator modes, the device clock in the PIC18F87J11 family can also be configured to provide a reference clock output signal to a port pin. This feature is available in all oscillator configurations and allows the user to select a greater range of clock sub-multiples to drive external devices in the application.

This reference clock output is controlled by the REFOCON register (Register 2-3). Setting the ROON bit (REFOCON<7>) makes the clock signal available on the REFO (RE3) pin. The RODIV3:RODIV0 bits enable the selection of 16 different clock divider options.

The ROSSLP and ROSEL bits (REFOCON<5:4>) control the availability of the reference output during Sleep mode. The ROSEL bit determines if the oscillator on OSC1 and OSC2, or the current system clock source, is used for the reference clock output. The ROSSLP bit determines if the reference source is available on RE3 when the device is in Sleep mode.

To use the reference clock output in Sleep mode, both the ROSSLP and ROSEL bits must be set. The device clock must also be configured for an EC or HS mode; otherwise, the oscillator on OSC1 and OSC2 will be powered down when the device enters Sleep mode.

Clearing the ROSEL bit allows the reference output frequency to change as the system clock changes during any clock switches.

The REFOCON register is an alternate SFR, and shares the same memory address as the OSCCON register. It is accessed by setting the ADSHR bit in the WDTCON register (WDTCON<4>).
REGISTER 2-3: REFOCON: REFERENCE OSCILLATOR CONTROL REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROON</td>
<td>—</td>
<td>ROSSL</td>
<td>ROSEL(1)</td>
<td>RODIV3</td>
<td>RODIV2</td>
<td>RODIV1</td>
<td>RODIV0</td>
</tr>
</tbody>
</table>

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

bit 7  
ROON: Reference Oscillator Output Enable bit  
1 = Reference oscillator output available on REFO pin  
0 = Reference oscillator output disabled

bit 6  
Unimplemented: Read as '0'

bit 5  
ROSSL: Reference Oscillator Output Stop in Sleep bit  
1 = Reference oscillator continues to run in Sleep  
0 = Reference oscillator is disabled in Sleep

bit 4  
ROSEL: Reference Oscillator Source Select bit(1)  
1 = Primary oscillator (EC or HS) used as the base clock  
0 = System clock used as the base clock; base clock reflects any clock switching of the device

bit 3-0  
RODIV3:RODIV0: Reference Oscillator Divisor Select bits  
1111 = Base clock value divided by 32,768  
1110 = Base clock value divided by 16,384  
1101 = Base clock value divided by 8,192  
1100 = Base clock value divided by 4,096  
1011 = Base clock value divided by 2,048  
1010 = Base clock value divided by 1,024  
1001 = Base clock value divided by 512  
1000 = Base clock value divided by 256  
0111 = Base clock value divided by 128  
0110 = Base clock value divided by 64  
0101 = Base clock value divided by 32  
0100 = Base clock value divided by 16  
0011 = Base clock value divided by 8  
0010 = Base clock value divided by 4  
0001 = Base clock value divided by 2  
0000 = Base clock value

Note 1: If ROSEL = 1, an EC or HS oscillator must be configured as the default oscillator with the FOSC Configuration bits to maintain clock output during Sleep mode.
2.7 Effects of Power-Managed Modes on the Various Clock Sources

When PRI_IDLE mode is selected, the designated primary oscillator continues to run without interruption. For all other power-managed modes, the oscillator using the OSC1 pin is disabled. The OSC1 pin (and OSC2 pin if used by the oscillator) will stop oscillating.

In secondary clock modes (SEC_RUN and SEC_IDLE), the Timer1 oscillator is operating and providing the device clock. The Timer1 oscillator may also run in all power-managed modes if required to clock Timer1 or Timer3.

In RC_RUN and RC_IDLE modes, the internal oscillator provides the device clock source. The 31 kHz INTRC output can be used directly to provide the clock and may be enabled to support various special features, regardless of the power-managed mode (see Section 24.2 “Watchdog Timer (WDT)” through Section 24.5 “Fail-Safe Clock Monitor” for more information on WDT, Fail-Safe Clock Monitor and Two-Speed Start-up).

If the Sleep mode is selected, all clock sources are stopped. Since all the transistor switching currents have been stopped, Sleep mode achieves the lowest current consumption of the device (only leakage currents).

Enabling any on-chip feature that will operate during Sleep will increase the current consumed during Sleep. The INTRC is required to support WDT operation. The Timer1 oscillator may be operating to support a Real-Time Clock (RTC). Other features may be operating that do not require a device clock source (i.e., MSSP slave, PSP, INTx pins and others). Peripherals that may add significant current consumption are listed in Section 27.2 “DC Characteristics: Power-Down and Supply Current”.

2.8 Power-up Delays

Power-up delays are controlled by two timers, so that no external Reset circuitry is required for most applications. The delays ensure that the device is kept in Reset until the device power supply is stable under normal circumstances and the primary clock is operating and stable. For additional information on power-up delays, see Section 4.6 “Power-up Timer (PWRT)”.

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up (parameter 33, Table 27-12); it is always enabled.

The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable (HS modes). The OST does this by counting 1024 oscillator cycles before allowing the oscillator to clock the device.

There is a delay of interval TCSD (parameter 38, Table 27-12), following POR, while the controller becomes ready to execute instructions.

<table>
<thead>
<tr>
<th>Oscillator Mode</th>
<th>OSC1 Pin</th>
<th>OSC2 Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC, ECPLL</td>
<td>Floating, pulled by external clock</td>
<td>At logic low (clock/4 output)</td>
</tr>
<tr>
<td>HS, HSPLL</td>
<td>Feedback inverter disabled at quiescent voltage level</td>
<td>Feedback inverter disabled at quiescent voltage level</td>
</tr>
<tr>
<td>INTOSC, INTPLL1/2</td>
<td>I/O pin RA6, direction controlled by TRISA&lt;6&gt;</td>
<td>I/O pin RA6, direction controlled by TRISA&lt;7&gt;</td>
</tr>
</tbody>
</table>

Note: See Section 4.0 “Reset” for time-outs due to Sleep and MCLR Reset.
3.0 POWER-MANAGED MODES

The PIC18F87J11 Family of devices provides the ability to manage power consumption by simply managing clocking to the CPU and the peripherals. In general, a lower clock frequency and a reduction in the number of circuits being clocked constitutes lower consumed power. For the sake of managing power in an application, there are three primary modes of operation:

- Run mode
- Idle mode
- Sleep mode

These modes define which portions of the device are clocked and at what speed. The Run and Idle modes may use any of the three available clock sources (primary, secondary or internal oscillator block); the Sleep mode does not use a clock source.

The power-managed modes include several power-saving features offered on previous devices. One is the clock switching feature, offered in other PIC18 devices, allowing the controller to use the Timer1 oscillator in place of the primary oscillator. Also included is the Sleep mode, offered by all PIC® devices, where all device clocks are stopped.

3.1 Selecting Power-Managed Modes

Selecting a power-managed mode requires two decisions: if the CPU is to be clocked or not and which clock source is to be used. The IDLEN bit (OSCCON<7>) controls CPU clocking, while the SCS1:SCS0 bits (OSCCON<1:0>) select the clock source. The individual modes, bit settings, clock sources and affected modules are summarized in Table 3-1.

3.1.1 CLOCK SOURCES

The SCS1:SCS0 bits allow the selection of one of three clock sources for power-managed modes. They are:

- the primary clock, as defined by the FOSC2:FOSC0 Configuration bits
- the secondary clock (Timer1 oscillator)
- the internal oscillator

3.1.2 ENTERING POWER-MANAGED MODES

Switching from one power-managed mode to another begins by loading the OSCCON register. The SCS1:SCS0 bits select the clock source and determine which Run or Idle mode is to be used. Changing these bits causes an immediate switch to the new clock source, assuming that it is running. The switch may also be subject to clock transition delays. These are discussed in Section 3.1.3 “Clock Transitions and Status Indicators” and subsequent sections.

Entry to the power-managed Idle or Sleep modes is triggered by the execution of a SLEEP instruction. The actual mode that results depends on the status of the IDLEN bit.

Depending on the current mode and the mode being switched to, a change to a power-managed mode does not always require setting all of these bits. Many transitions may be done by changing the oscillator select bits, or changing the IDLEN bit, prior to issuing a SLEEP instruction. If the IDLEN bit is already configured correctly, it may only be necessary to perform a SLEEP instruction to switch to the desired mode.

### Table 3-1: POWER-MANAGED MODES

<table>
<thead>
<tr>
<th>Mode</th>
<th>OSCCON&lt;7,1:0&gt;</th>
<th>Module Clocking</th>
<th>Available Clock and Oscillator Source</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IDLEN(1)</td>
<td>SCS1:SCS0</td>
<td>CPU</td>
</tr>
<tr>
<td>Sleep</td>
<td>0</td>
<td>N/A</td>
<td>Off</td>
</tr>
<tr>
<td>PRI_RUN</td>
<td>N/A</td>
<td>10</td>
<td>Clocked</td>
</tr>
<tr>
<td>SEC_RUN</td>
<td>N/A</td>
<td>01</td>
<td>Clocked</td>
</tr>
<tr>
<td>RC_RUN</td>
<td>N/A</td>
<td>11</td>
<td>Clocked</td>
</tr>
<tr>
<td>PRI_IDLE</td>
<td>1</td>
<td>10</td>
<td>Off</td>
</tr>
<tr>
<td>SEC_IDLE</td>
<td>1</td>
<td>01</td>
<td>Off</td>
</tr>
<tr>
<td>RC_IDLE</td>
<td>1</td>
<td>11</td>
<td>Off</td>
</tr>
</tbody>
</table>

Note 1: IDLEN reflects its value when the SLEEP instruction is executed.

2: Includes INTRC and INTOSC postcaler (internal oscillator block).
3.1.3 CLOCK TRANSITIONS AND STATUS INDICATORS

The length of the transition between clock sources is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Two bits indicate the current clock source and its status: OSTS (OSCCON<3>) and T1RUN (T1CON<6>). In general, only one of these bits will be set while in a given power-managed mode. When the OSTS bit is set, the primary clock is providing the device clock. When the T1RUN bit is set, the Timer1 oscillator is providing the clock. If neither of these bits is set, INTRC is clocking the device.

3.1.4 MULTIPLE SLEEP COMMANDS

The power-managed mode that is invoked with the SLEEP instruction is determined by the setting of the IDLEN bit at the time the instruction is executed. If another SLEEP instruction is executed, the device will enter the power-managed mode specified by IDLEN at that time. If IDLEN has changed, the device will enter the new power-managed mode specified by the new setting.

Note: Executing a SLEEP instruction does not necessarily place the device into Sleep mode. It acts as the trigger to place the controller into either the Sleep mode, or one of the Idle modes, depending on the setting of the IDLEN bit.

3.2 Run Modes

In the Run modes, clocks to both the core and peripherals are active. The difference between these modes is the clock source.

3.2.1 PRI_RUN MODE

The PRI_RUN mode is the normal, full-power execution mode of the microcontroller. This is also the default mode upon a device Reset unless Two-Speed Start-up is enabled (see Section 24.4 “Two-Speed Start-up” for details). In this mode, the OSTS bit is set. (see Section 2.2 “Control Registers”).

3.2.2 SEC_RUN MODE

The SEC_RUN mode is the compatible mode to the “clock switching” feature offered in other PIC18 devices. In this mode, the CPU and peripherals are clocked from the Timer1 oscillator. This gives users the option of lower power consumption while still using a high-accuracy clock source.

SEC_RUN mode is entered by setting the SCS1:SCS0 bits to '01'. The device clock source is switched to the Timer1 oscillator (see Figure 3-1), the primary oscillator is shut down, the T1RUN bit (T1CON<6>) is set and the OSTS bit is cleared.

Note: Executing a SLEEP instruction does not necessarily place the device into Sleep mode. It acts as the trigger to place the controller into either the Sleep mode, or one of the Idle modes, depending on the setting of the IDLEN bit.
On transitions from SEC_RUN mode to PRI_RUN mode, the peripherals and CPU continue to be clocked from the Timer1 oscillator while the primary clock is started. When the primary clock becomes ready, a clock switch back to the primary clock occurs (see Figure 3-2). When the clock switch is complete, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the clock. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run.

**Note:** The Timer1 oscillator should already be running prior to entering SEC_RUN mode. If the T1OSCEN bit is not set when the SCS1:SCS0 bits are set to '01', entry to SEC_RUN mode will not occur. If the Timer1 oscillator is enabled, but not yet running, device clocks will be delayed until the oscillator has started. In such situations, initial oscillator operation is far from stable and unpredictable operation may result.

**FIGURE 3-1: TRANSITION TIMING FOR ENTRY TO SEC_RUN MODE**

**FIGURE 3-2: TRANSITION TIMING FROM SEC_RUN MODE TO PRI_RUN MODE (HSPLL)**

**Note 1:** TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.
3.2.3 RC_RUN MODE

In RC_RUN mode, the CPU and peripherals are clocked from the internal oscillator; the primary clock is shut down. This mode provides the best power conservation of all the Run modes while still executing code. It works well for user applications which are not highly timing sensitive or do not require high-speed clocks at all times.

This mode is entered by setting SCS<1:0> to '11'. When the clock source is switched to the internal oscillator block (see Figure 3-3), the primary oscillator is shut down and the OSTS bit is cleared.

On transitions from RC_RUN mode to PRI_RUN mode, the device continues to be clocked from the INTOSC block while the primary clock is started. When the primary clock becomes ready, a clock switch to the primary clock occurs (see Figure 3-4). When the clock switch is complete, the OSTS bit is set and the primary clock is providing the device clock. The IDLEN and SCS bits are not affected by the switch. The INTRC block source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

**FIGURE 3-3: TRANSITION TIMING TO RC_RUN MODE**

**FIGURE 3-4: TRANSITION TIMING FROM RC_RUN MODE TO PRI_RUN MODE**

*Note 1: TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.*
3.3 Sleep Mode

The power-managed Sleep mode is identical to the legacy Sleep mode offered in all other PIC devices. It is entered by clearing the IDLEN bit (the default state on device Reset) and executing the SLEEP instruction. This shuts down the selected oscillator (Figure 3-5). All clock source status bits are cleared.

Entering the Sleep mode from any other mode does not require a clock switch. This is because no clocks are needed once the controller has entered Sleep. If the WDT is selected, the INTRC source will continue to operate. If the Timer1 oscillator is enabled, it will also continue to run.

When a wake event occurs in Sleep mode (by interrupt, Reset or WDT time-out), the device will not be clocked at all until the clock source selected by the SCS1:SCS0 bits becomes ready (see Figure 3-6), or it will be clocked from the internal oscillator if either the Two-Speed Start-up or the Fail-Safe Clock Monitor are enabled (see Section 24.0 “Special Features of the CPU”). In either case, the OSTS bit is set when the primary clock is providing the device clocks. The IDLEN and SCS bits are not affected by the wake-up.

3.4 Idle Modes

The Idle modes allow the controller’s CPU to be selectively shut down while the peripherals continue to operate. Selecting a particular Idle mode allows users to further manage power consumption.

If the IDLEN bit is set to ‘1’ when a SLEEP instruction is executed, the peripherals will be clocked from the clock source selected using the SCS1:SCS0 bits; however, the CPU will not be clocked. The clock source status bits are not affected. Setting IDLEN and executing a SLEEP instruction provides a quick method of switching from a given Run mode to its corresponding Idle mode.

If the WDT is selected, the INTRC source will continue to operate. If the Timer1 oscillator is enabled, it will also continue to run.

Since the CPU is not executing instructions, the only exits from any of the Idle modes are by interrupt, WDT time-out or a Reset. When a wake event occurs, CPU execution is delayed by an interval of TCSD (parameter 38, Table 27-12) while it becomes ready to execute code. When the CPU begins executing code, it resumes with the same clock source for the current Idle mode. For example, when waking from RC_IDLE mode, the internal oscillator block will clock the CPU and peripherals (in other words, RC_RUN mode). The IDLEN and SCS bits are not affected by the wake-up.

While in any Idle mode or the Sleep mode, a WDT time-out will result in a WDT wake-up to the Run mode currently specified by the SCS1:SCS0 bits.
3.4.1 PRI_IDLE MODE

This mode is unique among the three low-power Idle modes, in that it does not disable the primary device clock. For timing sensitive applications, this allows for the fastest resumption of device operation with its more accurate primary clock source, since the clock source does not have to “warm up” or transition from another oscillator.

PRI_IDLE mode is entered from PRI_RUN mode by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set IDLEN first, then set the SCS bits to '10' and execute SLEEP. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified by the FOSC1:FOSC0 Configuration bits. The OSTS bit remains set (see Figure 3-7).

When a wake event occurs, the CPU is clocked from the primary clock source. A delay of interval TCSD is required between the wake event and when code execution starts. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTS bit remains set. The IDLEN and SCS bits are not affected by the wake-up (see Figure 3-8).

3.4.2 SEC_IDLE MODE

In SEC_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the Timer1 oscillator. This mode is entered from SEC_RUN by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, set IDLEN first, then set SCS1:SCS0 to '01' and execute SLEEP. When the clock source is switched to the Timer1 oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the T1RUN bit is set.

When a wake event occurs, the peripherals continue to be clocked from the Timer1 oscillator. After an interval of TCSD following the wake event, the CPU begins executing code being clocked by the Timer1 oscillator. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run (see Figure 3-8).

Note: The Timer1 oscillator should already be running prior to entering SEC_IDLE mode. If the T1OSCEN bit is not set when the SLEEP instruction is executed, the SLEEP instruction will be ignored and entry to SEC_IDLE mode will not occur. If the Timer1 oscillator is enabled, but not yet running, peripheral clocks will be delayed until the oscillator has started. In such situations, initial oscillator operation is far from stable and unpredictable operation may result.

FIGURE 3-7: TRANSITION TIMING FOR ENTRY TO IDLE MODE

FIGURE 3-8: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE
3.4.3 RC_IDLE MODE
In RC_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the internal oscillator block. This mode allows for controllable power conservation during Idle periods.

From RC_RUN, this mode is entered by setting the IDLEN bit and executing a SLEEP instruction. If the device is in another Run mode, first set IDLEN, then clear the SCS bits and execute SLEEP. When the clock source is switched to the INTRC block, the primary oscillator is shut down and the OSTS bit is cleared.

When a wake event occurs, the peripherals continue to be clocked from the internal oscillator block. After a delay of TcSd following the wake event, the CPU begins executing code being clocked by the INTRC. The IDLEN and SCS bits are not affected by the wake-up. The INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

3.5 Exiting Idle and Sleep Modes
An exit from Sleep mode, or any of the Idle modes, is triggered by an interrupt, a Reset or a WDT time-out. This section discusses the triggers that cause exits from power-managed modes. The clocking subsystem actions are discussed in each of the power-managed modes sections (see Section 3.2 “Run Modes”, Section 3.3 “Sleep Mode” and Section 3.4 “Idle Modes”).

3.5.1 EXIT BY INTERRUPT
Any of the available interrupt sources can cause the device to exit from an Idle mode, or the Sleep mode, to a Run mode. To enable this functionality, an interrupt source must be enabled by setting its enable bit in one of the INTCON or PIE registers. The exit sequence is initiated when the corresponding interrupt flag bit is set.

On all exits from Idle or Sleep modes by interrupt, code execution branches to the interrupt vector if the GIE/GIEH bit (INTCON<7>) is set. Otherwise, code execution continues or resumes without branching (see Section 9.0 “Interrupts”).

A fixed delay of interval, TcSd, following the wake event is required when leaving Sleep and Idle modes. This delay is required for the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

3.5.2 EXIT BY WDT TIME-OUT
A WDT time-out will cause different actions depending on which power-managed mode the device is in when the time-out occurs.

If the device is not executing code (all Idle modes and Sleep mode), the time-out will result in an exit from the power-managed mode (see Section 3.2 “Run Modes” and Section 3.3 “Sleep Mode”). If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see Section 24.2 “Watchdog Timer (WDT)”).

The Watchdog Timer and postscaler are cleared by one of the following events:
- Executing a SLEEP or CLRWD T instruction
- The loss of a currently selected clock source (if the Fail-Safe Clock Monitor is enabled)

3.5.3 EXIT BY RESET
Exiting an Idle or Sleep mode by Reset automatically forces the device to run from the INTRC.

3.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY
Certain exits from power-managed modes do not invoke the OST at all. There are two cases:
- PRI_IDLE mode, where the primary clock source is not stopped; and
- The primary clock source is either the EC or ECPLL mode.

In these instances, the primary clock source either does not require an oscillator start-up delay, since it is already running (PRI_IDLE), or normally does not require an oscillator start-up delay (EC). However, a fixed delay of interval, TcSd, following the wake event is still required when leaving Sleep and Idle modes to allow the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.
4.0 RESET

The PIC18F87J11 Family of devices differentiate between various kinds of Reset:

a) Power-on Reset (POR)
b) MCLR Reset during normal operation
c) MCLR Reset during power-managed modes
d) Watchdog Timer (WDT) Reset (during execution)
e) Configuration Mismatch (CM)
f) Brown-out Reset (BOR)
g) RESET Instruction
h) Stack Full Reset
i) Stack Underflow Reset

This section discusses Resets generated by MCLR, POR and BOR and covers the operation of the various start-up timers. Stack Reset events are covered in Section 5.1.6.4 “Stack Full and Underflow Resets”. WDT Resets are covered in Section 24.2 “Watchdog Timer (WDT)”.

A simplified block diagram of the on-chip Reset circuit is shown in Figure 4-1.

4.1 RCON Register

Device Reset events are tracked through the RCON register (Register 4-1). The lower five bits of the register indicate that a specific Reset event has occurred. In most cases, these bits can only be set by the event and must be cleared by the application after the event. The state of these flag bits, taken together, can be read to indicate the type of Reset that just occurred. This is described in more detail in Section 4.7 “Reset State of Registers”.

The RCON register also has a control bit for setting interrupt priority (IPEN). Interrupt priority is discussed in Section 9.0 “Interrupts”.

FIGURE 4-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT

Note 1: The ENVREG pin must be tied high to enable Brown-out Reset. The Brown-out Reset is provided by the on-chip voltage regulator when there is insufficient source voltage to maintain regulation.
### REGISTER 4-1: RCON: RESET CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R-1</th>
<th>R-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPEN</td>
<td>—</td>
<td>CM</td>
<td>RI</td>
<td>TO</td>
<td>PD</td>
<td>POR</td>
<td>BOR</td>
</tr>
</tbody>
</table>

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

- **IPEN**: Interrupt Priority Enable bit
  - 1 = Enable priority levels on interrupts
  - 0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

- **CM**: Configuration Mismatch Flag bit
  - 1 = A Configuration Mismatch Reset has not occurred
  - 0 = A Configuration Mismatch Reset has occurred (must be set in software after a Configuration Mismatch Reset occurs)

- **RI**: RESET Instruction Flag bit
  - 1 = The RESET instruction was not executed (set by firmware only)
  - 0 = The RESET instruction was executed causing a device Reset (must be set in software after a Brown-out Reset occurs)

- **TO**: Watchdog Time-out Flag bit
  - 1 = Set by power-up, CLRWDT instruction or SLEEP instruction
  - 0 = A WDT time-out occurred

- **PD**: Power-Down Detection Flag bit
  - 1 = Set by power-up or by the CLRWDT instruction
  - 0 = Set by execution of the SLEEP instruction

- **POR**: Power-on Reset Status bit
  - 1 = A Power-on Reset has not occurred (set by firmware only)
  - 0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

- **BOR**: Brown-out Reset Status bit
  - 1 = A Brown-out Reset has not occurred (set by firmware only)
  - 0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

**Note 1:** It is recommended that the POR bit be set after a Power-on Reset has been detected, so that subsequent Power-on Resets may be detected.

2: If the on-chip voltage regulator is disabled, BOR remains ‘0’ at all times. See Section 4.4.1 “Detecting BOR” for more information.

3: Brown-out Reset is said to have occurred when BOR is ‘0’ and POR is ‘1’ (assuming that POR was set to ‘1’ by software immediately after a Power-on Reset).
4.2 Master Clear (MCLR)

The MCLR pin provides a method for triggering a hard external Reset of the device. A Reset is generated by holding the pin low. PIC18 extended microcontroller devices have a noise filter in the MCLR Reset path which detects and ignores small pulses.

The MCLR pin is not driven low by any internal Resets, including the WDT.

4.3 Power-on Reset (POR)

A Power-on Reset condition is generated on-chip whenever VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.

To take advantage of the POR circuitry, tie the MCLR pin through a resistor (1 kΩ to 10 kΩ) to VDD. This will eliminate external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for VDD is specified (parameter D004). For a slow rise time, see Figure 4-2.

When the device starts normal operation (i.e., exits the Reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in Reset until the operating conditions are met.

Power-on Reset events are captured by the POR bit (RCON<1>). The state of the bit is set to ‘0’ whenever a Power-on Reset occurs; it does not change for any other Reset event. POR is not reset to ‘1’ by any hardware event. To capture multiple events, the user manually resets the bit to ‘1’ in software following any Power-on Reset.

4.4 Brown-out Reset (BOR)

The PIC18F87J11 family of devices incorporates a simple Brown-out Reset function when the internal regulator is enabled (ENVREG pin is tied to VDD). Any drop of VDD below VBOR (parameter D005) for greater than time TBOR (parameter 35) will reset the device. A Reset may or may not occur if VDD falls below VBOR for less than TBOR. The chip will remain in Brown-out Reset until VDD rises above VBOR.

Once a Brown-out Reset has occurred, the Power-up Timer will keep the chip in Reset for TPWRT (parameter 33). If VDD drops below VBOR while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be initialized. Once VDD rises above VBOR, the Power-up Timer will execute the additional time delay.

FIGURE 4-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)

Note 1: External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device’s electrical specification.

3: R1 ≥ 1 kΩ will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

4.4.1 DETECTING BOR

The BOR bit always resets to ‘0’ on any Brown-out Reset or Power-on Reset event. This makes it difficult to determine if a Brown-out Reset event has occurred just by reading the state of BOR alone. A more reliable method is to simultaneously check the state of both POR and BOR. This assumes that the POR bit is reset to ‘1’ in software immediately after any Power-on Reset event. If BOR is ‘0’ while POR is ‘1’, it can be reliably assumed that a Brown-out Reset event has occurred.

If the voltage regulator is disabled, Brown-out Reset functionality is disabled. In this case, the BOR bit cannot be used to determine a Brown-out Reset event. The BOR bit is still cleared by a Power-on Reset event.

4.5 Configuration Mismatch (CM)

The Configuration Mismatch (CM) Reset is designed to detect and attempt to recover from random, memory corrupting events. These include Electrostatic Discharge (ESD) events, which can cause widespread, single-bit changes throughout the device and result in catastrophic failure.

In PIC18FXXJ Flash devices, the device Configuration registers (located in the configuration memory space) are continuously monitored during operation by comparing their values to complimentary shadow registers. If a mismatch is detected between the two sets of registers, a CM Reset automatically occurs. These events are captured by the CM bit (RCON<5>). The state of the bit is set to ‘0’ whenever a CM event occurs; it does not change for any other Reset event.
A CM Reset behaves similarly to a Master Clear Reset, \texttt{RESET} instruction, WDT time-out or Stack Event Resets. As with all hard and power Reset events, the device Configuration Words are reloaded from the Flash Configuration Words in program memory as the device restarts.

### 4.6 Power-up Timer (PWRT)

PIC18F87J11 Family devices incorporate an on-chip Power-up Timer (PWRT) to help regulate the Power-on Reset process. The PWRT is always enabled. The main function is to ensure that the device voltage is stable before code is executed.

The Power-up Timer (PWRT) of the PIC18F87J11 Family devices is an 11-bit counter which uses the INTRC source as the clock input. This yields an approximate time interval of $2048 \times 32 \mu s = 66 \text{ ms}$. While the PWRT is counting, the device is held in Reset.

The power-up time delay depends on the INTRC clock and will vary from chip-to-chip due to temperature and process variation. See DC parameter 33 for details.

#### 4.6.1 TIME-OUT SEQUENCE

If enabled, the PWRT time-out is invoked after the POR pulse has cleared. The total time-out will vary based on the status of the PWRT. Figure 4-3, Figure 4-4, Figure 4-5 and Figure 4-6 all depict time-out sequences on power-up with the Power-up Timer enabled.

Since the time-outs occur from the POR pulse, if MCLR is kept low long enough, the PWRT will expire. Bringing MCLR high will begin execution immediately (Figure 4-5). This is useful for testing purposes, or to synchronize more than one PIC18FXXXX device operating in parallel.
FIGURE 4-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2

FIGURE 4-6: SLOW RISE TIME (MCLR TIED TO VDD, VDD RISE > TPWRT)
4.7 Reset State of Registers

Most registers are unaffected by a Reset. Their status is unknown on POR and unchanged by all other Resets. The other registers are forced to a “Reset state” depending on the type of Reset that occurred.

Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register (CM, RI, TO, PD, POR and BOR) are set or cleared differently in different Reset situations, as indicated in Table 4-1. These bits are used in software to determine the nature of the Reset.

Table 4-2 describes the Reset states for all of the Special Function Registers. These are categorized by Power-on and Brown-out Resets, Master Clear and WDT Resets and WDT wake-ups.

### Table 4-1: Status Bits, Their Significance and the Initialization Condition for RCON Register

<table>
<thead>
<tr>
<th>Condition</th>
<th>Program Counter (1)</th>
<th>RCON Register</th>
<th>STKPTR Register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CM</td>
<td>RI</td>
<td>TO</td>
</tr>
<tr>
<td>Power-on Reset</td>
<td>0000h</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RESET instruction</td>
<td>0000h</td>
<td>u</td>
<td>0</td>
</tr>
<tr>
<td>Brown-out Reset</td>
<td>0000h</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Configuration Mismatch Reset</td>
<td>0000h</td>
<td>0</td>
<td>u</td>
</tr>
<tr>
<td>MCLR Reset during power-managed Run modes</td>
<td>0000h</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>MCLR Reset during power-managed Idle modes and Sleep mode</td>
<td>0000h</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>MCLR Reset during full-power execution</td>
<td>0000h</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>Stack Full Reset (STVREN = 1)</td>
<td>0000h</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>Stack Underflow Reset (STVREN = 1)</td>
<td>0000h</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>Stack Underflow Error (not an actual Reset, STVREN = 0)</td>
<td>0000h</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>WDT time-out during full-power or power-managed Run modes</td>
<td>0000h</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>WDT time-out during power-managed Idle or Sleep modes</td>
<td>PC + 2</td>
<td>u</td>
<td>u</td>
</tr>
<tr>
<td>Interrupt exit from power-managed modes</td>
<td>PC + 2</td>
<td>u</td>
<td>u</td>
</tr>
</tbody>
</table>

Legend:  u = unchanged

Note 1: When the wake-up is due to an interrupt and the GIEH or GIEL bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).
### TABLE 4-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS

<table>
<thead>
<tr>
<th>Register</th>
<th>Applicable Devices</th>
<th>Power-on Reset, Brown-out Reset</th>
<th>MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets</th>
<th>Wake-up via WDT or Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOSU</td>
<td>PIC18F6XJ1X</td>
<td>0 0000</td>
<td>0 0000</td>
<td>---0 uuuu(1)</td>
</tr>
<tr>
<td>TOSH</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu(1)</td>
</tr>
<tr>
<td>TOSL</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu(1)</td>
</tr>
<tr>
<td>STKPTR</td>
<td>PIC18F6XJ1X</td>
<td>00 0 0000</td>
<td>uu-0 0000</td>
<td>uu-0 uuuu(1)</td>
</tr>
<tr>
<td>PCLATU</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PCLATH</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>TOSU</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PCL</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>TBLPTRU</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>TBLPTRH</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>TABLAT</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PRODH</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PRODL</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>INTCON</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>INTCON2</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>INTCON3</td>
<td>PIC18F6XJ1X</td>
<td>00 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>INDF0</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POSTINC0</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POSTDEC0</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>PREINC0</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>PLUSW0</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FSR0H</td>
<td>PIC18F6XJ1X</td>
<td>---- xxxxx</td>
<td>---- uuuu</td>
<td>---- uuuu</td>
</tr>
<tr>
<td>FSR0L</td>
<td>PIC18F6XJ1X</td>
<td>---- xxxxx</td>
<td>---- uuuu</td>
<td>---- uuuu</td>
</tr>
<tr>
<td>WREG</td>
<td>PIC18F6XJ1X</td>
<td>---- xxxxx</td>
<td>---- uuuu</td>
<td>---- uuuu</td>
</tr>
<tr>
<td>INDF1</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POSTINC1</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POSTDEC1</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>PREINC1</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>PLUSW1</td>
<td>PIC18F6XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FSR1H</td>
<td>PIC18F6XJ1X</td>
<td>---- xxxxx</td>
<td>---- uuuu</td>
<td>---- uuuu</td>
</tr>
<tr>
<td>FSR1L</td>
<td>PIC18F6XJ1X</td>
<td>---- xxxxx</td>
<td>---- uuuu</td>
<td>---- uuuu</td>
</tr>
<tr>
<td>BSR</td>
<td>PIC18F6XJ1X</td>
<td>---- xxxxx</td>
<td>---- uuuu</td>
<td>---- uuuu</td>
</tr>
</tbody>
</table>

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

**Note 1:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

4: See Table 4-1 for Reset value for specific condition.
### TABLE 4-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

<table>
<thead>
<tr>
<th>Register</th>
<th>Applicable Devices</th>
<th>Power-on Reset, Brown-out Reset</th>
<th>MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets</th>
<th>Wake-up via WDT or Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>INDF2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POSTINC2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>POSTDEC2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>PREINC2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>PLUSW2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>FSR2H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>xxxx uuuu</td>
<td>xxxx uuuu</td>
</tr>
<tr>
<td>FSR2L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>xxxx uuuu xxxx uuuu</td>
<td>xxxx uuuu xxxx uuuu</td>
</tr>
<tr>
<td>STATUS</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>xxxx uuuu xxxx uuuu</td>
<td>xxxx uuuu xxxx uuuu</td>
</tr>
<tr>
<td>TMR0H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000 0000 0000</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>TMR0L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>xxxx uuuu xxxx uuuu</td>
<td>xxxx uuuu xxxx uuuu</td>
</tr>
<tr>
<td>T0CON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111 1111 1111</td>
<td>1111 1111 1111 1111</td>
</tr>
<tr>
<td>OSCCON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1110 1110</td>
<td>0110 1110 0110 1110</td>
<td>0110 1110 0110 1110</td>
</tr>
<tr>
<td>REFOCON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0-00 0000</td>
<td>u-uu uuuu u-uu uuuu</td>
<td>u-uu uuuu u-uu uuuu</td>
</tr>
<tr>
<td>CM1CON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0001 1111</td>
<td>uuuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>CM2CON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0001 1111</td>
<td>uuuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>RCON(4)</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0-11 1100</td>
<td>0-qq qquu uuuu uuuu</td>
<td>uuuu uuuu uuuu uuuu</td>
</tr>
<tr>
<td>TMR1H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>xxuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>ODCON1</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>---0 0000</td>
<td>---u uuuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>TMR1L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uuuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>ODCON2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uuuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>T1CON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>uuuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>ODCON3</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>---- ---0</td>
<td>---- ---u uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>TMR2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uuuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>PADCFG1</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111 1111 1111</td>
<td>1111 1111 1111 1111</td>
</tr>
<tr>
<td>PR2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0-00 ---0</td>
<td>0-00 ---0 uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>MEMCON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0-00 ---0</td>
<td>0-00 ---0 uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>T2CON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>-000 0000</td>
<td>-000 0000 uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>SSP1BUF</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uuuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>SSP1ADD</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000 uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>SSP1MSK</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>uuuu uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>SSP1STAT</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000 uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>SSP1CON1</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000 uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
<tr>
<td>SSP1CON2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000 uuuu uuuu</td>
<td>uuuu uuuu uuuu</td>
</tr>
</tbody>
</table>

Legend:  
- u = unchanged, x = unknown, - = unimplemented bit, read as ‘0’, q = value depends on condition.  
Shaded cells indicate conditions do not apply for the designated device.  

Note 1: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

4: See Table 4-1 for Reset value for specific condition.
<table>
<thead>
<tr>
<th>Register</th>
<th>Applicable Devices</th>
<th>Power-on Reset, Brown-out Reset</th>
<th>MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets</th>
<th>Wake-up via WDT or Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRESH</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>ADRESL</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>ADCN0</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>ADCN1</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>ANCON0</td>
<td>PIC18F6XJ1X</td>
<td>00-0 0000</td>
<td>uu-u uu-u</td>
<td>uu-u uu-u</td>
</tr>
<tr>
<td>ANCON1</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>uu-u uu-u</td>
<td>uu-u uu-u</td>
</tr>
<tr>
<td>WDTCON</td>
<td>PIC18F6XJ1X</td>
<td>0x-0 ----0</td>
<td>0x-u ----0</td>
<td>ux-u ---u</td>
</tr>
<tr>
<td>ECCP1AS</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>ECCP1DEL</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>CCPR1H</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uu-u uu-u</td>
<td>uu-u uu-u</td>
</tr>
<tr>
<td>CCPR1L</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uu-u uu-u</td>
<td>uu-u uu-u</td>
</tr>
<tr>
<td>CCP1CON</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>ECCP2AS</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>ECCP2DEL</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>CCPR2H</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uu-u uu-u</td>
<td>uu-u uu-u</td>
</tr>
<tr>
<td>CCPR2L</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uu-u uu-u</td>
<td>uu-u uu-u</td>
</tr>
<tr>
<td>CCP2CON</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>ECCP3AS</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>ECCP3DEL</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>CCPR3H</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uu-u uu-u</td>
<td>uu-u uu-u</td>
</tr>
<tr>
<td>CCPR3L</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uu-u uu-u</td>
<td>uu-u uu-u</td>
</tr>
<tr>
<td>CCP3CON</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>SPBRG1</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>RCREG1</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>TXREG1</td>
<td>PIC18F6XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uu-u uu-u</td>
<td>uu-u uu-u</td>
</tr>
<tr>
<td>TXSTA1</td>
<td>PIC18F6XJ1X</td>
<td>0000 010</td>
<td>0000 010</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>RCSTA1</td>
<td>PIC18F6XJ1X</td>
<td>0000 00x</td>
<td>0000 00x</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>SPBRG2</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>RCREG2</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>TXREG2</td>
<td>PIC18F6XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>TXSTA2</td>
<td>PIC18F6XJ1X</td>
<td>0000 010</td>
<td>0000 010</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>EECON2</td>
<td>PIC18F6XJ1X</td>
<td>----- -----</td>
<td>----- -----</td>
<td>----- -----</td>
</tr>
<tr>
<td>EECON1</td>
<td>PIC18F6XJ1X</td>
<td>---0 x00-</td>
<td>---0 u00-</td>
<td>---0 u00-</td>
</tr>
</tbody>
</table>

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

Shaded cells indicate conditions do not apply for the designated device.

**Note 1:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

4: See Table 4-1 for Reset value for specific condition.
### TABLE 4-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

<table>
<thead>
<tr>
<th>Register</th>
<th>Applicable Devices</th>
<th>Power-on Reset, Brown-out Reset</th>
<th>MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets</th>
<th>Wake-up via WDT or Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPR3</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu</td>
</tr>
<tr>
<td>PIR3</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>PIE3</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>IPR2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>111- 1111</td>
<td>111- 1111</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>PIR2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>000- 0000</td>
<td>000- 0000</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>PIE2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>000- 0000</td>
<td>000- 0000</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>IPR1</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>PIR1</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uu uu uu uu uu uu uu</td>
</tr>
<tr>
<td>PIE1</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>RCSTA2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 000x</td>
<td>0000 000x</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>OSCTUNE</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>TRISJ</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>TRISH</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu uu</td>
</tr>
<tr>
<td>TRISG</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>---1 1111</td>
<td>---1 1111</td>
<td>---uu uu uu</td>
</tr>
<tr>
<td>TRISF</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu uu uu uu uu</td>
</tr>
<tr>
<td>TRISE</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu uu uu uu uu</td>
</tr>
<tr>
<td>TRISD</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu uu uu uu uu</td>
</tr>
<tr>
<td>TRISC</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu uu uu uu uu</td>
</tr>
<tr>
<td>TRISB</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu uu uu uu uu</td>
</tr>
<tr>
<td>TRISA</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>uu uu uu uu uu uu uu</td>
</tr>
<tr>
<td>LATJ</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uu uu uu uu uu uu uu</td>
<td></td>
</tr>
<tr>
<td>LATH</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uu uu uu uu uu uu uu</td>
<td></td>
</tr>
<tr>
<td>LATG</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uu uu uu uu uu uu uu</td>
<td></td>
</tr>
<tr>
<td>LATF</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uu uu uu uu uu uu uu</td>
<td></td>
</tr>
<tr>
<td>LATE</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uu uu uu uu uu uu uu</td>
<td></td>
</tr>
<tr>
<td>LATD</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uu uu uu uu uu uu uu</td>
<td></td>
</tr>
<tr>
<td>LATC</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uu uu uu uu uu uu uu</td>
<td></td>
</tr>
<tr>
<td>LATB</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uu uu uu uu uu uu uu</td>
<td></td>
</tr>
<tr>
<td>LATA</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxx xxxx</td>
<td>uu uu uu uu uu uu uu</td>
<td></td>
</tr>
</tbody>
</table>

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

**Note 1:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

**Note 2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

**Note 3:** One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

**Note 4:** See Table 4-1 for Reset value for specific condition.
### TABLE 4-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

<table>
<thead>
<tr>
<th>Register</th>
<th>Applicable Devices</th>
<th>Power-on Reset, Brown-out Reset</th>
<th>MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets</th>
<th>Wake-up via WDT or Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTJ</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PORTH</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PORTG</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>000x xxxxx</td>
<td>000u uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PORTF</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>x001 100-</td>
<td>xuuu uuu-</td>
<td>xuuu uuu-</td>
</tr>
<tr>
<td>PORTE</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PORTD</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PORTC</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PORTB</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PORTA</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>000x 0000</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>SPBRGH1</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>BAUDCON1</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0100 0-0-0</td>
<td>0100 0-0-0</td>
<td>uuuu u-uu</td>
</tr>
<tr>
<td>SPBRGH2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>BAUDCON2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0100 0-0-0</td>
<td>0100 0-0-0</td>
<td>uuuu u-uu</td>
</tr>
<tr>
<td>TMR3H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>TMR3L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>T3CON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>TMR4</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PR4</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>1111 1111</td>
<td>1111 1111</td>
<td>1111 1111</td>
</tr>
<tr>
<td>CVRCON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>T4CON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0-00 0000</td>
<td>0-00 0000</td>
<td>-uu uuuu</td>
</tr>
<tr>
<td>CCRP4H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>CCRP4L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>CCP4CON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>--0 0000</td>
<td>--0 0000</td>
<td>--uu uuuu</td>
</tr>
<tr>
<td>CCP5H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>CCP5L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>CCP5CON</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>--0 0000</td>
<td>--0 0000</td>
<td>--uu uuuu</td>
</tr>
<tr>
<td>SSP2BUF</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>xxxxx xxxxx</td>
<td>uuuu uuuu</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>SSP2ADD</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>SSP2MSK</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>SSP2ADD</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>SSP2CON2</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>CMSTAT</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>----- --11</td>
<td>----- --11</td>
<td>----- --uuu</td>
</tr>
</tbody>
</table>

**Legend:**  
- = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition.  
Shaded cells indicate conditions do not apply for the designated device.

**Note 1:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

**Note 2:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

**Note 3:** One or more bits in the INTCNx or PIRx registers will be affected (to cause wake-up).

**Note 4:** See Table 4-1 for Reset value for specific condition.
### TABLE 4-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

<table>
<thead>
<tr>
<th>Register</th>
<th>Applicable Devices</th>
<th>Power-on Reset, Brown-out Reset</th>
<th>MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets</th>
<th>Wake-up via WDT or Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMADDRH</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMDOUT1H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMADDRL</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMDOUT1L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMDIN1H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMDIN1L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMCONH</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0-00 0000</td>
<td>0-00 0000</td>
<td>u-uu uuuu</td>
</tr>
<tr>
<td>PMCONL</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMMODEH</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMMODEL</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMDOUT2H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMDOUT2L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMDIN2H</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMDIN2L</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMEH</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMEL</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>uuuu uuuu</td>
</tr>
<tr>
<td>PMSTATH</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>00-- 0000</td>
<td>00-- 0000</td>
<td>uu-- uuuu</td>
</tr>
<tr>
<td>PMSTATL</td>
<td>PIC18F6XJ1X PIC18F8XJ1X</td>
<td>10-- 1111</td>
<td>10-- 1111</td>
<td>uu-- uuuu</td>
</tr>
</tbody>
</table>

**Legend:**  
- u = unchanged, x = unknown, - = unimplemented bit, read as ‘0’, q = value depends on condition.  
- Shaded cells indicate conditions do not apply for the designated device.  

**Note 1:** When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.  
  2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).  
  3: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).  
  4: See Table 4-1 for Reset value for specific condition.
5.0 MEMORY ORGANIZATION

There are two types of memory in PIC18 Flash microcontroller devices:
• Program Memory
• Data RAM

As Harvard architecture devices, the data and program memories use separate busses; this allows for concurrent access of the two memory spaces.

Additional detailed information on the operation of the Flash program memory is provided in Section 6.0 “Flash Program Memory”.

5.1 Program Memory Organization

PIC18 microcontrollers implement a 21-bit program counter which is capable of addressing a 2-Mbyte program memory space. Accessing a location between the upper boundary of the physically implemented memory and the 2-Mbyte address will return all ’0’s (a NOP instruction).

The entire PIC18F87J11 Family of devices offers three different on-chip Flash program memory sizes, from 64 Kbytes (up to 16,384 single-word instructions) to 128 Kbytes (65,536 single-word instructions). The program memory maps for individual family members are shown in Figure 5-3.

FIGURE 5-1: MEMORY MAPS FOR PIC18F87J11 FAMILY DEVICES

Note: Sizes of memory areas are not to scale. Sizes of program memory areas are enhanced to show detail.
5.1.1 HARD MEMORY VECTORS

All PIC18 devices have a total of three hard-coded return vectors in their program memory space. The Reset vector address is the default value to which the program counter returns on all device Resets; it is located at 0000h.

PIC18 devices also have two interrupt vector addresses for the handling of high-priority and low-priority interrupts. The high-priority interrupt vector is located at 0008h and the low-priority interrupt vector is at 0018h. Their locations in relation to the program memory map are shown in Figure 5-2.

FIGURE 5-2: HARD VECTOR AND CONFIGURATION WORD LOCATIONS FOR PIC18F87J11 FAMILY DEVICES

| reset Vector | 0000h |
| High-Priority Interrupt Vector | 0008h |
| Low-Priority Interrupt Vector | 0018h |
| On-Chip Program Memory | |
| Flash Configuration Words | (Top of Memory-7) (Top of Memory) |
| Read as '0' | |
| 1FFFFFh |

Legend: (Top of Memory) represents upper boundary of on-chip program memory space (see Figure 5-1 for device-specific values). Shaded area represents unimplemented memory. Areas are not shown to scale.

5.1.2 FLASH CONFIGURATION WORDS

Because PIC18F87J11 Family devices do not have persistent configuration memory, the top four words of on-chip program memory are reserved for configuration information. On Reset, the configuration information is copied into the Configuration registers.

The Configuration Words are stored in their program memory location in numerical order, starting with the lower byte of CONFIG1 at the lowest address and ending with the upper byte of CONFIG4. For these devices, only Configuration Words, CONFIG1 through CONFIG3, are used; CONFIG4 is reserved. The actual addresses of the Flash Configuration Word for devices in the PIC18F87J11 Family are shown in Table 5-1. Their location in the memory map is shown with the other memory vectors in Figure 5-2.

Additional details on the device Configuration Words are provided in Section 24.1 “Configuration Bits”.

TABLE 5-1: FLASH CONFIGURATION WORD FOR PIC18F87J11 FAMILY DEVICES

<table>
<thead>
<tr>
<th>Device</th>
<th>Program Memory (Kbytes)</th>
<th>Configuration Word Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC18F66J11</td>
<td>64</td>
<td>FFF8h to FFFFh</td>
</tr>
<tr>
<td>PIC18F86J11</td>
<td>96</td>
<td>17FF8h to 17FFFh</td>
</tr>
<tr>
<td>PIC18F66J16</td>
<td>128</td>
<td>1FFFF8h to 1FFFFFh</td>
</tr>
<tr>
<td>PIC18F86J16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIC18F67J11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIC18F87J11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.1.3 PIC18F8XJ11/8XJ16 PROGRAM MEMORY MODES

The 80-pin devices in this family can address up to a total of 2 Mbytes of program memory. This is achieved through the external memory bus. There are two distinct operating modes available to the controllers:

- Microcontroller (MC)
- Extended Microcontroller (EMC)

The program memory mode is determined by setting the EMB Configuration bits (CONFIG3L<5:4>), as shown in Register 5-1. (See also Section 24.1 “Configuration Bits” for additional details on the device Configuration bits.)

The program memory modes operate as follows:

- The Microcontroller Mode accesses only on-chip Flash memory. Attempts to read above the top of on-chip memory causes a read of all ‘0’s (a NOP instruction).

The Microcontroller mode is also the only operating mode available to 64-pin devices.

- The Extended Microcontroller Mode allows access to both internal and external program memories as a single block. The device can access its entire on-chip program memory; above this, the device accesses external program memory up to the 2-Mbyte program space limit. Execution automatically switches between the two memories as required.

The setting of the EMB Configuration bits also controls the address bus width of the external memory bus. This is covered in more detail in Section 7.0 “External Memory Bus”.

In all modes, the microcontroller has complete access to data RAM.

Figure 5-3 compares the memory maps of the different program memory modes. The differences between on-chip and external memory access limitations are more fully explained in Table 5-2.

REGISTER 5-1: CONFIG3L: CONFIGURATION REGISTER 3 LOW

<table>
<thead>
<tr>
<th>R/WO-1</th>
<th>R/WO-1</th>
<th>R/WO-1</th>
<th>R/WO-1</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAIT(1)</td>
<td>BW(1)</td>
<td>EMB1(1)</td>
<td>EMB0(1)</td>
<td>EASHFT(1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 7                  bit 0

Legend:

- WO = Write-Once bit
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 7 \(\text{WAIT: External Bus Wait Enable bit}^{(1)}\)
\(1 = \text{Wait states on the external bus are disabled}\)
\(0 = \text{Wait states on the external bus are enabled and selected by MEMCON}<5:4>\)

bit 6 \(\text{BW: Data Bus Width Select bit}^{(1)}\)
\(1 = \text{16-Bit Data Width modes}\)
\(0 = \text{8-Bit Data Width modes}\)

bit 5-4 \(\text{EMB1:EMB0: External Memory Bus Configuration bits}^{(1)}\)
\(11 = \text{Microcontroller mode, external bus disabled}\)
\(10 = \text{Extended Microcontroller mode, 12-bit address width for external bus}\)
\(01 = \text{Extended Microcontroller mode, 16-bit address width for external bus}\)
\(00 = \text{Extended Microcontroller mode, 20-bit address width for external bus}\)

bit 3 \(\text{EASHFT: External Address Bus Shift Enable bit}^{(1)}\)
\(1 = \text{Address shifting enabled – external address bus is shifted to start at 000000h}\)
\(0 = \text{Address shifting disabled – external address bus reflects the PC value}\)

bit 2-0 

\text{Unimplemented: Read as ‘0’}

\text{Note 1: } \text{Implemented only on 80-pin devices.}
5.1.4 EXTENDED MICROCONTROLLER MODE AND ADDRESS SHIFTING

By default, devices in Extended Microcontroller mode directly present the program counter value on the external address bus for those addresses in the range of the external memory space. In practical terms, this means addresses in the external memory device below the top of on-chip memory are unavailable.

To avoid this, the Extended Microcontroller mode implements an address shifting option to enable automatic address translation. In this mode, addresses presented on the external bus are shifted down by the size of the on-chip program memory and are remapped to start at 0000h. This allows the complete use of the external memory device’s memory space as an extension of the device’s on-chip program memory.

FIGURE 5-3: MEMORY MAPS FOR PIC18F87J11 FAMILY PROGRAM MEMORY MODES

<table>
<thead>
<tr>
<th>Microcontroller Mode&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>Extended Microcontroller Mode&lt;sup&gt;(2)&lt;/sup&gt;</th>
<th>Extended Microcontroller Mode with Address Shifting&lt;sup&gt;(2)&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-Chip Memory Space</td>
<td>External Memory Space</td>
<td>On-Chip Memory Space</td>
</tr>
<tr>
<td>000000h</td>
<td>On-Chip Program Memory</td>
<td>-</td>
</tr>
<tr>
<td>(Top of Memory) + 1</td>
<td>Mapped to External Memory Space</td>
<td>(Top of Memory) + 1</td>
</tr>
<tr>
<td>Reads as ‘0’s</td>
<td>External Memory</td>
<td>-</td>
</tr>
<tr>
<td>1FFFFFFh</td>
<td></td>
<td>1FFFFFFh - (Top of Memory) + 1&lt;sup&gt;(3)&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

Legend: (Top of Memory) represents upper boundary of on-chip program memory space (see Figure 5-1 for device-specific values). Shaded areas represent unimplemented, or inaccessible areas, depending on the mode.

Note 1: This mode is the only available mode on 64-pin devices and the default on 80-pin devices.

2: These modes are only available on 80-pin devices.

3: Addresses starting at the top of the program memory are translated to start at 0000h of the external device whenever the EASHIFT Configuration bit is set.

TABLE 5-2: MEMORY ACCESS FOR PIC18F8X11/8616 PROGRAM MEMORY MODES

<table>
<thead>
<tr>
<th>Operating Mode</th>
<th>Internal Program Memory</th>
<th>External Program Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Execution From</td>
<td>Table Read From</td>
</tr>
<tr>
<td>Microcontroller</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Extended Microcontroller</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
5.1.5 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see Section 5.1.8.1 “Computed GOTO”).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of ‘0’. The PC increments by 2 to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

5.1.6 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a CALL or RCALL instruction is executed, or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction (and on ADDULNK and SUBULNK instructions if the extended instruction set is enabled). PCLATU and PCLATH are not transferred to the program counter.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, STKPTR. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack Special Function Registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL type instruction causes a push onto the stack. The Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack. The contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

The Stack Pointer is initialized to ‘00000’ after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of ‘00000’; this is only a Reset value. Status bits indicate if the stack is full, has overflowed or has underflowed.

5.1.6.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 5-4). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt (and ADDULNK and SUBULNK instructions if the extended instruction set is enabled), the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the global interrupt enable bits while accessing the stack to prevent inadvertent stack corruption.

![FIGURE 5-4: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS](image-url)
5.1.6.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 5-2) contains the Stack Pointer value, the STKFUL (Stack Full) status bit and the STKUNF (Stack Underflow) status bits. The value of the Stack Pointer can be 0 through 31. The Stack Pointer increments before values are pushed onto the stack and decrements after values are popped off the stack. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The Stack Pointer will remain at zero.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and set the STKUNF bit, while the Stack Pointer remains at zero. The STKUNF bit will remain set until cleared by software or until a POR occurs.

Note: Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken. This is not the same as a Reset, as the contents of the SFRs are not affected.

5.1.6.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off the stack, without disturbing normal program execution, is a desirable feature. The PIC18 instruction set includes two instructions, PUSH and POP, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The PUSH instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The POP instruction discards the current TOS by decrementing the Stack Pointer and loads the current PC value onto the stack.

The previous value pushed onto the stack then becomes the TOS value.

REGISTER 5-2: STKPTR: STACK POINTER REGISTER

<table>
<thead>
<tr>
<th>R/C-0</th>
<th>R/C-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>STKFUL</td>
<td>STKUNF</td>
<td>—</td>
<td>SP4</td>
<td>SP3</td>
<td>SP2</td>
<td>SP1</td>
<td>SP0</td>
</tr>
</tbody>
</table>

Legend:

- C = Clearable-only bit
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 7    STKFUL: Stack Full Flag bit(1)
          1 = Stack became full or overflowed
          0 = Stack has not become full or overflowed

bit 6    STKUNF: Stack Underflow Flag bit(1)
          1 = Stack underflow occurred
          0 = Stack underflow did not occur

bit 5    Unimplemented: Read as ‘0’

bit 4-0  SP4:SP0: Stack Pointer Location bits

Note 1: Bit 7 and bit 6 are cleared by user software or by a POR.
5.1.6.4  Stack Full and Underflow Resets
Device Resets on stack overflow and stack underflow conditions are enabled by setting the STVREN bit in Configuration Register 1L. When STVREN is set, a full or underflow condition will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. When STVREN is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit, but not cause a device Reset. The STKFUL or STKUNF bits are cleared by the user software or a Power-on Reset.

5.1.7  FAST REGISTER STACK
A Fast Register Stack is provided for the STATUS, WREG and BSR registers to provide a “fast return” option for interrupts. This stack is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the Stack registers. The values in the registers are then loaded back into the working registers if the RETFIE, FAST instruction is used to return from the interrupt.

If both low and high-priority interrupts are enabled, the Stack registers cannot be used reliably to return from low-priority interrupts. If a high-priority interrupt occurs while servicing a low-priority interrupt, the Stack register values stored by the low-priority interrupt will be overwritten. In these cases, users must save the key registers in software during a low-priority interrupt.

If interrupt priority is not used, all interrupts may use the Fast Register Stack for returns from interrupt. If no interrupts are used, the Fast Register Stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the Fast Register Stack for a subroutine call, a CALL label, FAST instruction must be executed to save the STATUS, WREG and BSR registers to the Fast Register Stack. A RET, FAST instruction is then executed to restore these registers from the Fast Register Stack.

Example 5-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

EXAMPLE 5-1:  FAST REGISTER STACK
CODE EXAMPLE

```
CALL SUB1, FAST ;STATUS, WREG, BSR
;SAVED IN FAST REGISTER
;STACK

SUB1

RETURN FAST ;RESTORE VALUES SAVED
;IN FAST REGISTER STACK
```

5.1.8  LOOK-UP TABLES IN PROGRAM MEMORY
There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed GOTO
- Table Reads

5.1.8.1  Computed GOTO
A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 5-2.

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW nn instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW nn instructions that returns the value ‘nn’ to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of 2 (LSb = 0).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

**EXAMPLE 5-2:  COMPUTED GOTO USING AN OFFSET VALUE**

```
MOVF OFFSET, W
CALL TABLE
ORG nn00h
TABLE ADDWF PCL
RET L W nn
RET L W nn
RET L W nn
.
.
.
```

5.1.8.2  Table Reads
A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word while programming. The Table Pointer (TBLPTR) specifies the byte address and the Table Latch (TABLAT) contains the data that is read from the program memory. Data is transferred from program memory one byte at a time.

Table read operation is discussed further in Section 6.1 “Table Reads and Table Writes”.

© 2008 Microchip Technology Inc.

Preliminary

DS39778C-page 67
5.2 PIC18 Instruction Cycle

5.2.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the program counter is incremented on every Q1; the instruction is fetched from the program memory and latched into the Instruction Register (IR) during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 5-5.

5.2.2 INSTRUCTION FLOW/PIPELINING

An “Instruction Cycle” consists of four Q cycles, Q1 through Q4. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), then two cycles are required to complete the instruction (Example 5-3).

A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 5-5: CLOCK/INSTRUCTION CYCLE

EXAMPLE 5-3: INSTRUCTION PIPELINE FLOW

<table>
<thead>
<tr>
<th>TCY0</th>
<th>TCY1</th>
<th>TCY2</th>
<th>TCY3</th>
<th>TCY4</th>
<th>TCY5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch 1</td>
<td>Execute 1</td>
<td>Fetch 2</td>
<td>Execute 2</td>
<td>Fetch 3</td>
<td>Execute 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fetch 4</td>
<td>Flush (NOP)</td>
<td>Fetch SUB_1</td>
<td>Execute SUB_1</td>
</tr>
</tbody>
</table>

All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.
5.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSB = 0). To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see Section 5.1.5 “Program Counter”).

Figure 5-6 shows an example of how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1> which accesses the desired byte address in program memory. Instruction #2 in Figure 5-6 shows how the instruction, GOTO 0006h, is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. Section 25.0 “Instruction Set Summary” provides further details of the instruction set.

FIGURE 5-6: INSTRUCTIONS IN PROGRAM MEMORY

<table>
<thead>
<tr>
<th>Program Memory Byte Locations →</th>
<th>LSB = 1 Word Address</th>
<th>LSB = 0 ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction 1: MOVLW 055h</td>
<td>000000h</td>
<td></td>
</tr>
<tr>
<td>Instruction 2: GOTO 0006h</td>
<td>000002h</td>
<td>000004h</td>
</tr>
<tr>
<td>Instruction 3: MOVFF 123h, 456h</td>
<td>000006h</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000008h</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000Ah</td>
<td>000010h</td>
</tr>
<tr>
<td></td>
<td>000012h</td>
<td></td>
</tr>
<tr>
<td></td>
<td>000014h</td>
<td></td>
</tr>
</tbody>
</table>

5.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: CALL, MOVFF, GOTO and LSFR. In all cases, the second word of the instructions always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the 4 MSBs of an instruction specifies a special form of NOP. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a NOP is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 5-4 shows how this works.

Note: See Section 5.5 “Program Memory and the Extended Instruction Set” for information on two-word instructions in the extended instruction set.

EXAMPLE 5-4: TWO-WORD INSTRUCTIONS

<table>
<thead>
<tr>
<th>CASE 1:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object Code</strong></td>
<td><strong>Source Code</strong></td>
</tr>
<tr>
<td>0110 0110 0000 0000</td>
<td>TSTFSZ REG1 ; is RAM location 0?</td>
</tr>
<tr>
<td>1100 0001 0010 0011</td>
<td>MOVFF REG1, REG2 ; No, skip this word</td>
</tr>
<tr>
<td>1111 0100 0101 0110</td>
<td>ADDWF REG3 ; continue code</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CASE 2:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object Code</strong></td>
<td><strong>Source Code</strong></td>
</tr>
<tr>
<td>0110 0110 0000 0000</td>
<td>TSTFSZ REG1 ; is RAM location 0?</td>
</tr>
<tr>
<td>1100 0001 0010 0011</td>
<td>MOVFF REG1, REG2 ; Yes, execute this word</td>
</tr>
<tr>
<td>1111 0100 0101 0110</td>
<td>ADDWF REG3 ; 2nd word of instruction</td>
</tr>
<tr>
<td>0010 0100 0000 0000</td>
<td>ADDWF REG3 ; continue code</td>
</tr>
</tbody>
</table>
5.3 Data Memory Organization

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. The PIC18F87J11 family implements all available banks and provide 3936 bytes of data memory available to the user. Figure 5-7 shows the data memory organization for the devices.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user’s application. Any read of an unimplemented location will read as '0's.

The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this section.

To ensure that commonly used registers (select SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to select SFRs and the lower portion of GPR Bank 0 without using the BSR. Section 5.3.2 “Access Bank” provides a detailed description of the Access RAM.

5.3.1 BANK SELECT REGISTER

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 16 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 12-bit address, or an 8-bit low-order address and a 4-bit Bank Pointer.

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the 4 Most Significant bits of a location’s address; the instruction itself includes the 8 Least Significant bits. Only the four lower bits of the BSR are implemented (BSR3:BSR0). The upper four bits are unused; they will always read '0' and cannot be written to. The BSR can be loaded directly by using the MOVLB instruction.

The value of the BSR indicates the bank in data memory. The 8 bits in the instruction show the location in the bank and can be thought of as an offset from the bank’s lower boundary. The relationship between the BSR’s value and the bank division in data memory is shown in Figure 5-8.

Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 0Fh, will end up resetting the program counter.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory map in Figure 5-7 indicates which banks are implemented.

In the core PIC18 instruction set, only the MOVFF instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the Access Bank to locate their target registers.
**FIGURE 5-7: DATA MEMORY MAP FOR PIC18F87J11 FAMILY DEVICES**

<table>
<thead>
<tr>
<th>Bank</th>
<th>Data Memory Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank 0</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 1</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 2</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 3</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 4</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 5</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 6</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 7</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 8</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 9</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 10</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 11</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 12</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 13</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 14</td>
<td>FFh 00h</td>
</tr>
<tr>
<td>Bank 15</td>
<td>FFh 00h</td>
</tr>
</tbody>
</table>

**Note 1:** Addresses F5Ah through F5Fh are also used by SFRs, but are not part of the Access RAM. Users must always use the complete address, or load the proper BSR value, to access these registers.

When \( a = 0 \):
- The BSR is ignored and the Access Bank is used.
- The first 96 bytes are general purpose RAM (from Bank 0).
- The remaining 160 bytes are Special Function Registers (from Bank 15).

When \( a = 1 \):
- The BSR specifies the bank used by the instruction.

**Access Bank**

- **Access RAM Low (SFRs)**
- **Access RAM High**

---

Note 1: Addresses F5Ah through F5Fh are also used by SFRs, but are not part of the Access RAM. Users must always use the complete address, or load the proper BSR value, to access these registers.
FIGURE 5-8: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)

5.3.2 ACCESS BANK

While the use of the BSR with an embedded 8-bit address allows users to address the entire range of data memory, it also means that the user must always ensure that the correct bank is selected. Otherwise, data may be read from or written to the wrong location. This can be disastrous if a GPR is the intended target of an operation, but an SFR is written to instead. Verifying and/or changing the BSR for each read or write to data memory can become very inefficient.

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Bank 15. The lower half is known as the “Access RAM” and is composed of GPRs. The upper half is where the device’s SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figure 5-7).

The Access Bank is used by core PIC18 instructions that include the Access RAM bit (the ‘a’ parameter in the instruction). When ‘a’ is equal to ‘1’, the instruction uses the BSR and the 8-bit address included in the opcode for the data memory address. When ‘a’ is ‘0’, however, the instruction is forced to use the Access Bank address map; the current value of the BSR is ignored entirely.

Using this “forced” addressing allows the instruction to operate on a data address in a single cycle without updating the BSR first. For 8-bit addresses of 60h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 60h is a good place for data values that the user might need to access rapidly, such as immediate computational results or common program variables. Access RAM also allows for faster and more code efficient context saving and switching of variables.

The mapping of the Access Bank is slightly different when the extended instruction set is enabled (XINST Configuration bit = 1). This is discussed in more detail in Section 5.6.3 “Mapping the Access Bank in Indexed Literal Offset Mode”.

5.3.3 GENERAL PURPOSE REGISTER FILE

PIC18 devices may have banked memory in the GPR area. This is data RAM which is available for use by all instructions. GPRs start at the bottom of Bank 0 (address 000h) and grow upwards towards the bottom of the SFR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.
5.3.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy more than the top half of Bank 15 (F5Ah to FFFh). A list of these registers is given in Table 5-3, Table 5-4 and Table 5-5.

The SFRs can be classified into two sets: those associated with the "core" device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The Reset and interrupt registers are described in their respective chapters, while the ALU's STATUS register is described later in this section. Registers related to the operation of the peripheral features are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as '0's.

Note: Addresses, F5Ah through F5Fh, are not part of the Access Bank. These registers must always be accessed using the Bank Select Register.

### TABLE 5-3: SPECIAL FUNCTION REGISTER MAP FOR PIC18F87J11 FAMILY DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Address</th>
<th>Name</th>
<th>Address</th>
<th>Name</th>
<th>Address</th>
<th>Name</th>
<th>Address</th>
<th>Name</th>
<th>Address</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFFh</td>
<td>TOSU</td>
<td>FDeh</td>
<td>INDF2(1)</td>
<td>F8Fh</td>
<td>ECCP1AS</td>
<td>F7Fh</td>
<td>IPR1</td>
<td>F7Fh</td>
<td>SPBRGH1</td>
<td>F5Fh</td>
<td>PMDIN2H</td>
</tr>
<tr>
<td>FFEh</td>
<td>TOSH</td>
<td>FDeh</td>
<td>POSTINC2(2)</td>
<td>F8Eh</td>
<td>ECCP1DEL</td>
<td>F6Eh</td>
<td>PIR1</td>
<td>F6Eh</td>
<td>BAUDCON1</td>
<td>F5Eh</td>
<td>PMDIN2L</td>
</tr>
<tr>
<td>FFDeh</td>
<td>TOSL</td>
<td>FDeh</td>
<td>POSTDEC2(2)</td>
<td>F8Dh</td>
<td>CCPR1H</td>
<td>F5Dh</td>
<td>PIE1</td>
<td>F5Dh</td>
<td>SPBRGH2</td>
<td>F5Dh</td>
<td>PMEH</td>
</tr>
<tr>
<td>FFCh</td>
<td>STKPTR</td>
<td>FDeh</td>
<td>PREINC(2)</td>
<td>F8Ch</td>
<td>CCPR1L</td>
<td>F4Ch</td>
<td>RCSTA2</td>
<td>F4Ch</td>
<td>BAUDCON2</td>
<td>F4Ch</td>
<td>PMEL</td>
</tr>
<tr>
<td>FFb</td>
<td>PCLATU</td>
<td>FDeh</td>
<td>PLUSW2(3)</td>
<td>F8b</td>
<td>CCP1CON</td>
<td>F3b</td>
<td>OSCUTE</td>
<td>F3b</td>
<td>TMR3H</td>
<td>F2b</td>
<td>PMSTATH</td>
</tr>
<tr>
<td>FFb</td>
<td>PCLATH</td>
<td>FDeh</td>
<td>FSR2H</td>
<td>F8b</td>
<td>ECCP2AS</td>
<td>F2b</td>
<td>TRIS(4)</td>
<td>F2b</td>
<td>TMR3L</td>
<td>F2b</td>
<td>PMSTATL</td>
</tr>
<tr>
<td>FFb</td>
<td>PCL</td>
<td>FDeh</td>
<td>FSR2L</td>
<td>F8b</td>
<td>ECCP2DEL</td>
<td>F2b</td>
<td>TRIS(4)</td>
<td>F2b</td>
<td>TMR3L</td>
<td>F2b</td>
<td>PMSTATL</td>
</tr>
<tr>
<td>FFb</td>
<td>TBLPTRU</td>
<td>FDeh</td>
<td>STATUS</td>
<td>F8b</td>
<td>CCPR2H</td>
<td>F2b</td>
<td>TRIS</td>
<td>F2b</td>
<td>TMR4</td>
<td>F2b</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>TBLPTRH</td>
<td>FDeh</td>
<td>TMROH</td>
<td>F8b</td>
<td>CCPR2L</td>
<td>F2b</td>
<td>TRIS</td>
<td>F2b</td>
<td>TMR5</td>
<td>F2b</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>TBLPTRL</td>
<td>FDeh</td>
<td>TMROL</td>
<td>F8b</td>
<td>CCP2CON</td>
<td>F2b</td>
<td>TRIS</td>
<td>F2b</td>
<td>T4CON</td>
<td>F2b</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>TABLAT</td>
<td>FDeh</td>
<td>TOCON</td>
<td>F8b</td>
<td>ECCP3AS</td>
<td>F2b</td>
<td>TRISD</td>
<td>F2b</td>
<td>CCP4H</td>
<td>F2b</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>PRODH</td>
<td>FDeh</td>
<td>—</td>
<td>F8b</td>
<td>ECCP3DEL</td>
<td>F2b</td>
<td>TRIS</td>
<td>F2b</td>
<td>CCP4L</td>
<td>F2b</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>PRODL</td>
<td>FDeh</td>
<td>OSCCON(3)</td>
<td>F8b</td>
<td>CCPR3H</td>
<td>F1b</td>
<td>TRISB</td>
<td>F1b</td>
<td>CCP4CON</td>
<td>F1b</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>INTCON</td>
<td>FDeh</td>
<td>CM1CON</td>
<td>F8b</td>
<td>CCP3RL</td>
<td>F1b</td>
<td>TRISA</td>
<td>F1b</td>
<td>CCP5H</td>
<td>F1b</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>INTCON2</td>
<td>FDeh</td>
<td>CM2CON</td>
<td>F8b</td>
<td>CCP3CON</td>
<td>F1b</td>
<td>TRISA</td>
<td>F1b</td>
<td>CCP5H</td>
<td>F1b</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>INTCON3</td>
<td>FDeh</td>
<td>RCON</td>
<td>F8b</td>
<td>SPBRG1</td>
<td>F1b</td>
<td>LAT(3)</td>
<td>F1b</td>
<td>CCP5L</td>
<td>F1b</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>INDF0</td>
<td>FDeh</td>
<td>TMR1(3)</td>
<td>FAFh</td>
<td>RCREG1</td>
<td>F0h</td>
<td>LATG</td>
<td>F0h</td>
<td>CCP6CON</td>
<td>F0h</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>INDF1</td>
<td>FDeh</td>
<td>TMR1H(3)</td>
<td>FAFh</td>
<td>RCREG1</td>
<td>F0h</td>
<td>LATG</td>
<td>F0h</td>
<td>CCP6CON</td>
<td>F0h</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>POSTINC(3)</td>
<td>FDeh</td>
<td>TMRL1(3)</td>
<td>FAEl</td>
<td>TXREG1</td>
<td>F0h</td>
<td>LATF</td>
<td>F0h</td>
<td>SSP2BUF</td>
<td>F0h</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>POSTDEC0(1)</td>
<td>FDeh</td>
<td>T1CON(3)</td>
<td>FAEl</td>
<td>TXREG1</td>
<td>F0h</td>
<td>LATF</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>POSTDEC1(1)</td>
<td>FDeh</td>
<td>T2CON(3)</td>
<td>FAEl</td>
<td>TXREG1</td>
<td>F0h</td>
<td>LATF</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>PREINC0(3)</td>
<td>FDeh</td>
<td>TMR2(3)</td>
<td>FAEl</td>
<td>TXREG1</td>
<td>F0h</td>
<td>LATF</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>PLUSW0(3)</td>
<td>FDeh</td>
<td>PR2(3)</td>
<td>FAEl</td>
<td>TXREG1</td>
<td>F0h</td>
<td>LATF</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>FSROH</td>
<td>FCah</td>
<td>T2CON</td>
<td>FAc</td>
<td>RCREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>CMSTAT</td>
<td>F0h</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>FSROL</td>
<td>FCah</td>
<td>SSP1BUFF</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>CMSTAT</td>
<td>F0h</td>
<td>—</td>
</tr>
<tr>
<td>FFb</td>
<td>WREG</td>
<td>FCah</td>
<td>SSP1ADD</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>WREG</td>
<td>FCah</td>
<td>SSP1ADD</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>WREG</td>
<td>FCah</td>
<td>SSP1ADD</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>WREG</td>
<td>FCah</td>
<td>SSP1ADD</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>INDF1(1)</td>
<td>FCCh</td>
<td>TSP1STAT</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>POSTINC(1)</td>
<td>FCCh</td>
<td>TSP1ADD</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>POSTDEC0(1)</td>
<td>FCCh</td>
<td>TSP1ADD</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>PREINC(1)</td>
<td>FCCh</td>
<td>TSP1ADD</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>PLUSW(1)</td>
<td>FCCh</td>
<td>ADRESL</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>FSROH</td>
<td>FCb</td>
<td>ADRES</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>FSROL</td>
<td>FCb</td>
<td>ADRESL</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>FFb</td>
<td>WREG</td>
<td>FCb</td>
<td>ADRES</td>
<td>FAc</td>
<td>TXREG2</td>
<td>F0h</td>
<td>LATB</td>
<td>F0h</td>
<td>—</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: This is not a physical register.
2: This register is not available on 64-pin devices.
3: This register shares the same address with another register (see Table 5-4 for alternate register).
4: The PMADDRH/L and PMDOUT1H/L register pairs share the same address. PMADDR is used in Master modes and PMDOUT1 is used in Slave modes.
5.3.4.1 Shared Address SFRs

In several locations in the SFR bank, a single address is used to access two different hardware registers. In these cases, a “legacy” register of the standard PIC18 SFR set (such as OSCCON, T1CON, etc.) shares its address with an alternate register. These alternate registers are associated with enhanced configuration options for peripherals, or with new device features not included in the standard PIC18 SFR map. A complete list of shared register addresses and the registers associated with them is provided in Table 5-4.

Access to the alternate registers is enabled in software by setting the ADSHR bit in the WDTCON register (Register 5-3). ADSHR must be manually set or cleared to access the alternate or legacy registers, as required. Since the bit remains in a given state until changed, users should always verify the state of ADSHR before writing to any of the shared SFR addresses.

5.3.4.2 Context Defined SFRs

In addition to the shared address SFRs, there are several registers that share the same address in the SFR space, but are not accessed with the ADSHR bit. Instead, the register’s definition and use depends on the operating mode of its associated peripheral. These registers are:

- SSPxADD and SSPxMSK: These are two separate hardware registers, accessed through a single SFR address. The operating mode of the MSSP module determines which register is being accessed. See Section 19.4.3.4 “7-Bit Address Masking Mode” for additional details.
- PMADDRH/L and PMDOUT2H/L: In this case, these named buffer pairs are actually the same physical registers. The PMP module’s operating mode determines what function the registers take on. See Section 11.1.2 “Data Registers” for additional details.

### TABLE 5-4: SHARED SFR ADDRESSES FOR PIC18F87J11 FAMILY DEVICES

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Address</th>
<th>Name</th>
<th>Address</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>FD3h (D)</td>
<td>OSCCON</td>
<td>FCCh (D)</td>
<td>TMR2</td>
<td>FC1h (D)</td>
<td>ADCON1</td>
</tr>
<tr>
<td>(A)</td>
<td>REFOCON</td>
<td>(A)</td>
<td>ODCON1</td>
<td>(A)</td>
<td>ADCON1</td>
</tr>
<tr>
<td>FCFh (D)</td>
<td>TMR1H</td>
<td>FCCh (D)</td>
<td>TMR2</td>
<td>FC1h (D)</td>
<td>ADCON1</td>
</tr>
<tr>
<td>(A)</td>
<td>ODCON1</td>
<td>(A)</td>
<td>PADCFG1</td>
<td>(A)</td>
<td>ANCON0</td>
</tr>
<tr>
<td>FCEh (D)</td>
<td>TMR1L</td>
<td>FCBh (D)</td>
<td>PR2</td>
<td>F77h (D)</td>
<td>PR4</td>
</tr>
<tr>
<td>(A)</td>
<td>ODCON2</td>
<td>(A)</td>
<td>MEMCON</td>
<td>(A)</td>
<td>CVRCON</td>
</tr>
</tbody>
</table>

Legend:
- (D) = Default SFR, accessible only when ADSHR = 0; (A) = Alternate SFR, accessible only when ADSHR = 1.
- Note 1: Implemented in 80-pin devices only.

### REGISTER 5-3: WDTCON: WATCHDOG TIMER CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R-x</th>
<th>U-0</th>
<th>R/W-0</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>REGSLP</td>
<td>LVDSTAT</td>
<td>—</td>
<td>ADSHR</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SWDTEN</td>
</tr>
</tbody>
</table>

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

bit 7  REGSLP: Voltage Regulator Low-Power Operation Enable bit
       For details of bit operation, see Register 24-9.

bit 6  LVDSTAT: LVD Status bit
       1 = VDDCORE > 2.45 V
       0 = VDDCORE < 2.45 V

bit 5  Unimplemented: Read as ‘0’

bit 4  ADSHR: Shared Address SFR Select bit
       1 = Alternate SFR is selected
       0 = Default (Legacy) SFR is selected

bit 3-1 Unimplemented: Read as ‘0’

bit 0  SWDTEN: Software Controlled Watchdog Timer Enable bit
       For details of bit operation, see Register 24-9.
### TABLE 5-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY)

<table>
<thead>
<tr>
<th>File 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>
<th>Value on POR, BOR</th>
<th>Details on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOSU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>---0</td>
<td>0000 0000</td>
<td>55, 65</td>
</tr>
<tr>
<td>TOSH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>55, 65</td>
</tr>
<tr>
<td>TOSL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>0000 0000</td>
<td>55, 65</td>
</tr>
<tr>
<td>STKPTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STKFUL</td>
<td>STKUNF</td>
<td></td>
</tr>
<tr>
<td>PCLATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 21(1)</td>
<td>---0 0000</td>
<td>55, 65</td>
</tr>
<tr>
<td>TCLATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>---00</td>
<td>0000 0000</td>
<td>55, 96</td>
</tr>
<tr>
<td>TBLPTRU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit 21</td>
<td>Program Memory Table Pointer Upper Byte (TBLPTR&lt;20:16&gt;)</td>
<td>55, 96</td>
</tr>
<tr>
<td>TBLPTRH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TBLPTRH</td>
<td>Program Memory Table Pointer High Byte (TBLPTR&lt;15:8&gt;)</td>
<td>55, 96</td>
</tr>
<tr>
<td>TBLPRTR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TBLPRTR</td>
<td>Program Memory Table Pointer Low Byte (TBLPTR&lt;7:0&gt;)</td>
<td>55, 96</td>
</tr>
<tr>
<td>TABLAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>---00-0</td>
<td>Program Memory Table Latch</td>
<td>55, 96</td>
</tr>
<tr>
<td>PRODH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PRODH</td>
<td>Product Register High Byte</td>
<td>55, 109</td>
</tr>
<tr>
<td>PRODL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PRODL</td>
<td>Product Register Low Byte</td>
<td>55, 109</td>
</tr>
<tr>
<td>INTCN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INTCN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INTCN2</td>
<td>RBPU, INTEDG0, INTEDG1, INTEDG2, INTEDG3, TMR0IP, INT3IP, RBIP</td>
<td>55, 113</td>
</tr>
<tr>
<td>INTCON3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INTCON3</td>
<td>INT2IP, INT1IP, INT3IE, INT2IE, INT1IE, INT3IF, INT2IF, INT1IF</td>
<td>55, 113</td>
</tr>
<tr>
<td>INDF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INDF0</td>
<td>Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)</td>
<td>N/A 55, 82</td>
</tr>
<tr>
<td>POSTINC0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POSTINC0</td>
<td>Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>POSTDEC0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POSTDEC0</td>
<td>Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>PREINC0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PREINC0</td>
<td>Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>PLUSW0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PLUSW0</td>
<td>Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>FSR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FSR0H</td>
<td>Indirect Data Memory Address Pointer 0 High Byte</td>
<td>----- xxxx</td>
</tr>
<tr>
<td>FSR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FSR0L</td>
<td>Working Register</td>
<td>XXXX XXXX</td>
</tr>
<tr>
<td>WREG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WREG</td>
<td>XXXX XXXX</td>
<td>55, 67</td>
</tr>
<tr>
<td>INDF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INDF1</td>
<td>Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)</td>
<td>N/A 55, 82</td>
</tr>
<tr>
<td>POSTINC1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POSTINC1</td>
<td>Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>POSTDEC1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POSTDEC1</td>
<td>Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>PREINC1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PREINC1</td>
<td>Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>PLUSW1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PLUSW1</td>
<td>Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>FSR1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FSR1H</td>
<td>Indirect Data Memory Address Pointer 1 High Byte</td>
<td>----- xxxx</td>
</tr>
<tr>
<td>FSR1L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FSR1L</td>
<td>Indirect Data Memory Address Pointer 1 Low Byte</td>
<td>----- xxxx</td>
</tr>
<tr>
<td>BSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BSR</td>
<td>Bank Select Register</td>
<td>----- 0000</td>
</tr>
<tr>
<td>INDF2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INDF2</td>
<td>Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)</td>
<td>N/A 55, 82</td>
</tr>
<tr>
<td>POSTINC2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POSTINC2</td>
<td>Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>POSTDEC2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>POSTDEC2</td>
<td>Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>PREINC2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PREINC2</td>
<td>Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)</td>
<td>N/A 55, 83</td>
</tr>
<tr>
<td>PLUSW2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PLUSW2</td>
<td>Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W</td>
<td>N/A 55, 83</td>
</tr>
</tbody>
</table>

Legend:  
- x = unknown, u = unchanged, - = unimplemented, q = value depends on condition. **Bold** indicates shared access SFRs.  

**Note**  
1: Bit 21 of the PIC is only available in Serial Programming modes.  
2: Default (legacy) SFR at this address, available when WDTCON<4> = 0.  
3: Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.  
4: Reset value is ‘0’ when Two-Speed Start-up is enabled and ‘1’ if disabled.  
5: The SSPxMSK registers are only accessible when SSPxCON2<3:0> = 1001.  
6: Alternate names and definitions for these bits when the MSSP modules are operating in I²C™ Slave mode. See Section 19.4.3.2 “Address Masking Modes” for details.  
7: These bits and/or registers are only available in 80-pin devices; otherwise, they are unimplemented and read as ‘0’. Reset values are shown for 80-pin devices.  
8: These bits are only available in select oscillator modes (FOSC2 Configuration bit = 0); otherwise, they are unimplemented.  
9: The PMADDRH/PMDOUT1H and PMADDRL/PMDOUT1L register pairs share the physical registers and addresses, but have different functions determined by the module’s operating mode. See Section 11.1.2 “Data Registers” for more information.
TABLE 5-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY) (CONTINUED)

<table>
<thead>
<tr>
<th>File 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>Details on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSR2H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address Pointer 2 High Byte</td>
<td>56, 82</td>
</tr>
<tr>
<td>FSR2L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indirect Data Memory Address Pointer 2 Low Byte</td>
<td>56, 82</td>
</tr>
<tr>
<td>STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>N, OV, Z, DC, C</td>
<td>56, 80</td>
</tr>
<tr>
<td>TMR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer0 Register High Byte</td>
<td>56, 179</td>
</tr>
<tr>
<td>TMR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer0 Register Low Byte</td>
<td>56, 179</td>
</tr>
<tr>
<td>T0CON</td>
<td>TMR0ON</td>
<td>T0BBIT</td>
<td>T0CS</td>
<td>T0SE</td>
<td>PSA</td>
<td>T0PS2</td>
<td>T0PS1</td>
<td>T0PS0</td>
<td>1111 1111</td>
<td>56, 178</td>
</tr>
<tr>
<td>OSCCON(2)</td>
<td>IDLEN</td>
<td>IRCF2</td>
<td>IRCF1</td>
<td>IRCF0</td>
<td>OSTS(4)</td>
<td></td>
<td></td>
<td></td>
<td>0110 q100</td>
<td>56, 32</td>
</tr>
<tr>
<td>REFCON(5)</td>
<td>ROON</td>
<td>ROSSL</td>
<td>ROSEL</td>
<td>RODIV3</td>
<td>RODIV2</td>
<td>RODIV1</td>
<td>RODIV0</td>
<td>0-00 0000</td>
<td>56, 39</td>
<td></td>
</tr>
<tr>
<td>CM1CON</td>
<td>CON</td>
<td>COE</td>
<td>CPOL</td>
<td>EVPOL1</td>
<td>EVPOL0</td>
<td>CREF</td>
<td>CCH1</td>
<td>CCH0</td>
<td>0001 1111</td>
<td>56, 302</td>
</tr>
<tr>
<td>CM2CON</td>
<td>CON</td>
<td>COE</td>
<td>CPOL</td>
<td>EVPOL1</td>
<td>EVPOL0</td>
<td>CREF</td>
<td>CCH1</td>
<td>CCH0</td>
<td>0001 1111</td>
<td>56, 302</td>
</tr>
<tr>
<td>RCON</td>
<td>IPEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CM, R, TO, PD, POR, BOR</td>
<td>0-11 1100, 54, 6, 125</td>
</tr>
<tr>
<td>TMR1H(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer1 Register High Byte</td>
<td>56, 182</td>
</tr>
<tr>
<td>ODCON1(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCP5OD</td>
<td>CCP4OD</td>
<td>ECCP3OD</td>
<td>ECCP2OD</td>
<td>0-00 0000</td>
<td>56, 129</td>
</tr>
<tr>
<td>TMR1L(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer1 Register Low Byte</td>
<td>56, 182</td>
</tr>
<tr>
<td>ODCON2(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U2OD</td>
<td>U1OD</td>
<td>U0OD</td>
<td></td>
<td>0-00 0000</td>
<td>56, 129</td>
</tr>
<tr>
<td>T1CON(2)</td>
<td>RD16</td>
<td>T1RUN</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>TTSYNC</td>
<td>TMR1CS</td>
<td>TMR1ON</td>
<td>0000 0000</td>
<td>56, 182</td>
</tr>
<tr>
<td>ODCON3(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPI2OD</td>
<td>SPI1OD</td>
<td></td>
<td></td>
<td>0-00 0000</td>
<td>56, 129</td>
</tr>
<tr>
<td>TMR2(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer2 Register</td>
<td>56, 187</td>
</tr>
<tr>
<td>PADCFG1(3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>56, 187</td>
</tr>
<tr>
<td>PR2(2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer2 Period Register</td>
<td>56, 130</td>
</tr>
<tr>
<td>MEMCON(3,7)</td>
<td>EDBIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MSSP1 Receive Buffer/Transmit Register</td>
<td>56, 222, 231</td>
</tr>
<tr>
<td>SSP1BUF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MSSP1 Address Register (I²C™ Slave mode), MSSP1 Baud Rate Reload Register (I²C Master mode)</td>
<td>56, 223, 231</td>
</tr>
<tr>
<td>SSP1ADD/</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000 0000</td>
<td>56, 223</td>
</tr>
<tr>
<td>SSP1MSK(6)</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>0000 0000</td>
<td>56, 238</td>
</tr>
<tr>
<td>SSP1STAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SMP, CKE, D/A, P, S, R/W, UA, BF</td>
<td>56, 222, 232</td>
</tr>
<tr>
<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</td>
<td>56, 223, 233</td>
</tr>
<tr>
<td>SSP1CON2</td>
<td>GCEC</td>
<td>ACKSTAT</td>
<td>ACKDT</td>
<td>ACKEN</td>
<td>RCEN</td>
<td>PEN</td>
<td>RSEN/ SEN</td>
<td>GCEC</td>
<td>ACKSTAT</td>
<td>ADMSK(4)</td>
</tr>
<tr>
<td>ADRESH</td>
<td>A/D Result Register High Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 291</td>
<td></td>
</tr>
<tr>
<td>ADRESL</td>
<td>A/D Result Register Low Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 291</td>
<td></td>
</tr>
<tr>
<td>ADCON0(2)</td>
<td>VCFG1</td>
<td>VCFG0</td>
<td>CHS3</td>
<td>CHS2</td>
<td>CHS1</td>
<td>CHS0</td>
<td>GO/DONE</td>
<td>ADON</td>
<td>0000 0000</td>
<td>57, 291</td>
</tr>
<tr>
<td>ADCON1(3)</td>
<td>PCFG15</td>
<td>PCFG14</td>
<td>PCFG13</td>
<td>PCFG12</td>
<td>PCFG11</td>
<td>PCFG10</td>
<td>PCFG9</td>
<td>PCFG8</td>
<td>0000 0000</td>
<td>57, 293</td>
</tr>
<tr>
<td>ADCON1(4)</td>
<td>ADFM</td>
<td>ADCA</td>
<td>ACQ2</td>
<td>ACQ1</td>
<td>ACQ0</td>
<td>ADCS2</td>
<td>ADCS1</td>
<td>ADCS0</td>
<td>0000 0000</td>
<td>57, 292</td>
</tr>
<tr>
<td>ADCON0(3)</td>
<td>PCFG7</td>
<td>PCFG6</td>
<td></td>
<td>PCFG4</td>
<td>PCFG3</td>
<td>PCFG2</td>
<td>PCFG1</td>
<td>PCFG0</td>
<td>0-00 0000</td>
<td>57, 293</td>
</tr>
<tr>
<td>WDTCON</td>
<td>REGSLP</td>
<td>LVDSTAP</td>
<td></td>
<td></td>
<td>ADHNR</td>
<td></td>
<td></td>
<td></td>
<td>0x-0 0000</td>
<td>57, 321</td>
</tr>
</tbody>
</table>

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

Bold indicates shared access SFRs.

Note:
1. Bit 21 of the PC is only available in Serial Programming modes.
2. Default (legacy) SFR at this address, available when WDTCON<4> = 0.
3. Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.
4. Reset value is ‘0’ when Two-Speed Start-up is enabled and ‘1’ if disabled.
5. The SSPxMSK registers are only accessible when SSPxCON2<3:0> = 1001.
6. Alternate names and definitions for these bits when the MSSP modules are operating in I²C™ Slave mode. See Section 19.4.3.2 “Address Masking Modes” for details.
7. These bits and/or registers are only available in 80-pin devices; otherwise, they are unimplemented and read as ‘0’.
8. These bits are only available in select oscillator modes (FOSC2 Configuration bit = 0); otherwise, they are unimplemented.
9. The PMADDRH/PMOUTH and PMADDRL/PMOUTL register pairs share the physical registers and addresses, but have different functions determined by the module’s operating mode. See Section 11.1.2 “Data Registers” for more information.
### TABLE 5-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY) (CONTINUED)

<table>
<thead>
<tr>
<th>File 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>Details on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECCP1AS</td>
<td>ECCP1ASE</td>
<td>ECCP1AS2</td>
<td>ECCP1AS1</td>
<td>ECCP1AS0</td>
<td>PSS1AC1</td>
<td>PSS1AC0</td>
<td>PSS1BD1</td>
<td>PSS1BD0</td>
<td>0000 0000</td>
<td>57, 219</td>
</tr>
<tr>
<td>ECCP1DEL</td>
<td>P1RSEN</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</td>
<td>57, 219</td>
</tr>
<tr>
<td>CCPR1H</td>
<td>Capture/Compare/PWM Register 1 High Byte</td>
<td>xxxxx</td>
<td>xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 219</td>
</tr>
<tr>
<td>CCPR1L</td>
<td>Capture/Compare/PWM Register 1 Low Byte</td>
<td>xxxxx</td>
<td>xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 219</td>
</tr>
<tr>
<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</td>
<td>57, 219</td>
</tr>
<tr>
<td>EPPAS</td>
<td>EPPASE</td>
<td>EPPAS2</td>
<td>EPPAS1</td>
<td>EPPAS0</td>
<td>PSS1AC1</td>
<td>PSS1AC0</td>
<td>PSS1BD1</td>
<td>PSS1BD0</td>
<td>0000 0000</td>
<td>57, 219</td>
</tr>
<tr>
<td>EPPDEL</td>
<td>P2RSEN</td>
<td>P2DC6</td>
<td>P2DC5</td>
<td>P2DC4</td>
<td>P2DC3</td>
<td>P2DC2</td>
<td>P2DC1</td>
<td>P2DC0</td>
<td>0000 0000</td>
<td>57, 219</td>
</tr>
<tr>
<td>CCPR2H</td>
<td>Capture/Compare/PWM Register 2 HIgh Byte</td>
<td>xxxxx</td>
<td>xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 219</td>
</tr>
<tr>
<td>CCPR2L</td>
<td>Capture/Compare/PWM Register 2 Low Byte</td>
<td>xxxxx</td>
<td>xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 219</td>
</tr>
<tr>
<td>CCP2CON</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</td>
<td>57, 219</td>
</tr>
<tr>
<td>EPPAS</td>
<td>EPPASE</td>
<td>EPPAS2</td>
<td>EPPAS1</td>
<td>EPPAS0</td>
<td>PSS1AC1</td>
<td>PSS1AC0</td>
<td>PSS1BD1</td>
<td>PSS1BD0</td>
<td>0000 0000</td>
<td>57, 219</td>
</tr>
<tr>
<td>EPPDEL</td>
<td>P2RSEN</td>
<td>P2DC6</td>
<td>P2DC5</td>
<td>P2DC4</td>
<td>P2DC3</td>
<td>P2DC2</td>
<td>P2DC1</td>
<td>P2DC0</td>
<td>0000 0000</td>
<td>57, 219</td>
</tr>
<tr>
<td>CCPR3H</td>
<td>Capture/Compare/PWM Register 3 HIgh Byte</td>
<td>xxxxx</td>
<td>xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 219</td>
</tr>
<tr>
<td>CCPR3L</td>
<td>Capture/Compare/PWM Register 3 Low Byte</td>
<td>xxxxx</td>
<td>xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 219</td>
</tr>
<tr>
<td>CCP3CON</td>
<td>P3M1</td>
<td>P3M0</td>
<td>DC3B1</td>
<td>DC3B0</td>
<td>CCP3M3</td>
<td>CCP3M2</td>
<td>CCP3M1</td>
<td>CCP3M0</td>
<td>0000 0000</td>
<td>57, 219</td>
</tr>
<tr>
<td>SPBRG1</td>
<td>EUSART1 Baud Rate Generator Register Low Byte</td>
<td>0000</td>
<td>0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 273</td>
</tr>
<tr>
<td>RCREG1</td>
<td>EUSART1 Receive Register</td>
<td>0000</td>
<td>0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 281, 282</td>
</tr>
<tr>
<td>TXREG1</td>
<td>EUSART1 Transmit Register</td>
<td>xxxxx</td>
<td>xxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 279, 280</td>
</tr>
<tr>
<td>TXSTA1</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SNCDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>0000 0010</td>
<td>57, 279</td>
</tr>
<tr>
<td>RSTCA1</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</td>
<td>57, 281</td>
</tr>
<tr>
<td>SPBRG2</td>
<td>EUSART2 Baud Rate Generator Register Low Byte</td>
<td>0000</td>
<td>0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 273</td>
</tr>
<tr>
<td>RCREG2</td>
<td>EUSART2 Receive Register</td>
<td>0000</td>
<td>0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 281, 282</td>
</tr>
<tr>
<td>TXREG2</td>
<td>EUSART2 Transmit Register</td>
<td>0000</td>
<td>0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57, 279, 280</td>
</tr>
<tr>
<td>TXSTA2</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYNC</td>
<td>SNCDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>0000 0010</td>
<td>57, 279</td>
</tr>
<tr>
<td>EECON1</td>
<td>Program Memory Control Register 2 (not a physical register)</td>
<td>-----</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-----</td>
<td>57, 88</td>
</tr>
<tr>
<td>EECON2</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-----</td>
<td>57, 88</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>---</td>
<td>57, 88</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TM R4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>1111 1111</td>
<td>58, 122</td>
</tr>
<tr>
<td>PR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TM R4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>0000 0000</td>
<td>58, 116</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TM R4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>0000 0000</td>
<td>58, 119</td>
</tr>
<tr>
<td>IPR2</td>
<td>OSCFIP</td>
<td>CM2IP</td>
<td>CM1IP</td>
<td>—</td>
<td>BCL1IP</td>
<td>LVDF1</td>
<td>TMR3IP</td>
<td>CCP2IP</td>
<td>111 - 1111</td>
<td>58, 122</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSCFIF</td>
<td>CM2IF</td>
<td>CM1IF</td>
<td>—</td>
<td>BCL1IF</td>
<td>LVDF1</td>
<td>TMR3IF</td>
<td>CCP2IF</td>
<td>0000 - 0000</td>
<td>58, 116</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSCFIE</td>
<td>CM2IE</td>
<td>CM1IE</td>
<td>—</td>
<td>BCL1IE</td>
<td>LVDF1</td>
<td>TMR3IE</td>
<td>CCP2IE</td>
<td>0000 - 0000</td>
<td>58, 119</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMP1IP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TM R2IP</td>
<td>TM R1IP</td>
<td>1111 1111</td>
<td>58, 122</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMP1IF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TM R2IF</td>
<td>TM R1IF</td>
<td>0000 0000</td>
<td>58, 116</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TM R2IE</td>
<td>TM R1IE</td>
<td>0000 0000</td>
<td>58, 119</td>
</tr>
<tr>
<td>RCSTA2</td>
<td>SPISA</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</td>
<td>58, 281</td>
</tr>
<tr>
<td>OSC2FEN</td>
<td>INTSRC</td>
<td>PLLEN</td>
<td>TUN5</td>
<td>TUN4</td>
<td>TUN3</td>
<td>TUN2</td>
<td>TUN1</td>
<td>TUN0</td>
<td>0000 0000</td>
<td>58, 33</td>
</tr>
</tbody>
</table>

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

**Bold** indicates shared access SFRs.

**Note:**
1. Bit 21 of the PC is only available in Serial Programming modes.
2. Default (legacy) SFR at this address, available when WDTCON<4> = 0.
3. Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.
4. Reset value is '0' when Two-Speed Start-up is enabled and '1' if disabled.
5. The SSPxMSK registers are only accessible when SSPxCON2<3:0> = 0001.
6. Alternate names and definitions for these bits when the MSSP modules are operating in i²C™ Slave mode. See Section 19.4.3.2 “Address Masking Modes” for details.
7. These bits and/or registers are only available in 80-pin devices; otherwise, they are unimplemented and read as '0'. Reset values are shown for 80-pin devices.
8. These bits are only available in select oscillator modes (FOSC2 Configuration bit = 0); otherwise, they are unimplemented.
9. The PMADDRH/PMOUT1H and PMADDR/L/PMOUT1L register pairs share the physical registers and addresses, but have different functions determined by the module’s operating mode. See Section 11.1.2 “Data Registers” for more information.
### TABLE 5-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY) (CONTINUED)

<table>
<thead>
<tr>
<th>File 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>Details on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRISJ</td>
<td>TRISJ7</td>
<td>TRISJ6</td>
<td>TRISJ5</td>
<td>TRISJ4</td>
<td>TRISJ3</td>
<td>TRISJ2</td>
<td>TRISJ1</td>
<td>TRISJ0</td>
<td>1111 1111</td>
<td>58, 150</td>
</tr>
<tr>
<td>TRISH</td>
<td>TRISH7</td>
<td>TRISH6</td>
<td>TRISH5</td>
<td>TRISH4</td>
<td>TRISH3</td>
<td>TRISH2</td>
<td>TRISH1</td>
<td>TRISH0</td>
<td>1111 1111</td>
<td>58, 148</td>
</tr>
<tr>
<td>TRISG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRISG</td>
<td>—</td>
<td>1111 1111</td>
</tr>
<tr>
<td>TRISF</td>
<td>TRISF7</td>
<td>TRISF6</td>
<td>TRISF5</td>
<td>TRISF4</td>
<td>TRISF3</td>
<td>TRISF2</td>
<td>—</td>
<td>TRISF1</td>
<td>—</td>
<td>1111 1111</td>
</tr>
<tr>
<td>TRISE</td>
<td>TRISE7</td>
<td>TRISE6</td>
<td>TRISE5</td>
<td>TRISE4</td>
<td>TRISE3</td>
<td>TRISE2</td>
<td>TRISE1</td>
<td>TRISE0</td>
<td>1111 1111</td>
<td>58, 144</td>
</tr>
<tr>
<td>TRISD</td>
<td>TRISD7</td>
<td>TRISD6</td>
<td>TRISD5</td>
<td>TRISD4</td>
<td>TRISD3</td>
<td>TRISD2</td>
<td>TRISD1</td>
<td>TRISD0</td>
<td>1111 1111</td>
<td>58, 138</td>
</tr>
<tr>
<td>TRISC</td>
<td>TRISC7</td>
<td>TRISC6</td>
<td>TRISC5</td>
<td>TRISC4</td>
<td>TRISC3</td>
<td>TRISC2</td>
<td>TRISC1</td>
<td>TRISC0</td>
<td>1111 1111</td>
<td>58, 136</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>1111 1111</td>
<td>58, 134</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>1111 1111</td>
<td>58, 132</td>
</tr>
<tr>
<td>LATJ(7)</td>
<td>LATJ7</td>
<td>LATJ6</td>
<td>LATJ5</td>
<td>LATJ4</td>
<td>LATJ3</td>
<td>LATJ2</td>
<td>LATJ1</td>
<td>LATJ0</td>
<td>xxxx xxxx</td>
<td>58, 150</td>
</tr>
<tr>
<td>LATH(7)</td>
<td>LATH7</td>
<td>LATH6</td>
<td>LATH5</td>
<td>LATH4</td>
<td>LATH3</td>
<td>LATH2</td>
<td>LATH1</td>
<td>LATH0</td>
<td>xxxx xxxx</td>
<td>58, 148</td>
</tr>
<tr>
<td>LATG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>LATG4</td>
<td>LATG3</td>
<td>LATG2</td>
<td>LATG1</td>
<td>—</td>
<td>1111 1111</td>
</tr>
<tr>
<td>LATF</td>
<td>LATF7</td>
<td>LATF6</td>
<td>LATF5</td>
<td>LATF4</td>
<td>LATF3</td>
<td>LATF2</td>
<td>—</td>
<td>LATF1</td>
<td>—</td>
<td>1111 1111</td>
</tr>
<tr>
<td>LATE</td>
<td>LATE7</td>
<td>LATE6</td>
<td>LATE5</td>
<td>LATE4</td>
<td>LATE3</td>
<td>LATE2</td>
<td>LATE1</td>
<td>LATE0</td>
<td>1111 1111</td>
<td>58, 141</td>
</tr>
<tr>
<td>LATD</td>
<td>LATD7</td>
<td>LATD6</td>
<td>LATD5</td>
<td>LATD4</td>
<td>LATD3</td>
<td>LATD2</td>
<td>LATD1</td>
<td>LATD0</td>
<td>1111 1111</td>
<td>58, 138</td>
</tr>
<tr>
<td>LATC</td>
<td>LATC7</td>
<td>LATC6</td>
<td>LATC5</td>
<td>LATC4</td>
<td>LATC3</td>
<td>LATC2</td>
<td>LATC1</td>
<td>LATC0</td>
<td>1111 1111</td>
<td>58, 136</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>1111 1111</td>
<td>58, 134</td>
</tr>
<tr>
<td>LATA</td>
<td>LATA7</td>
<td>LATA6</td>
<td>LATA5</td>
<td>LATA4</td>
<td>LATA3</td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
<td>1111 1111</td>
<td>58, 132</td>
</tr>
<tr>
<td>PORTJ</td>
<td>RJ7</td>
<td>RJ6</td>
<td>RJ5</td>
<td>RJ4</td>
<td>RJ3</td>
<td>RJ2</td>
<td>RJ1</td>
<td>RJ0</td>
<td>—</td>
<td>1111 1111</td>
</tr>
<tr>
<td>PORTH</td>
<td>RH7</td>
<td>RH6</td>
<td>RH5</td>
<td>RH4</td>
<td>RH3</td>
<td>RH2</td>
<td>RH1</td>
<td>RH0</td>
<td>0000 xxxx</td>
<td>58, 148</td>
</tr>
<tr>
<td>PORTG</td>
<td>RDPU</td>
<td>REP</td>
<td>RJP(J7)</td>
<td>RG4</td>
<td>RG3</td>
<td>RG2</td>
<td>RG1</td>
<td>RG0</td>
<td>0000 xxxx</td>
<td>58, 146</td>
</tr>
<tr>
<td>PORTF</td>
<td>RF7</td>
<td>RF6</td>
<td>RF5</td>
<td>RF4</td>
<td>RF3</td>
<td>RF2</td>
<td>RF1</td>
<td>—</td>
<td>x000 0000-2</td>
<td>58, 144</td>
</tr>
<tr>
<td>PORTE</td>
<td>RE7</td>
<td>RE6</td>
<td>RE5</td>
<td>RE4</td>
<td>RE3</td>
<td>RE2</td>
<td>RE1</td>
<td>RE0</td>
<td>—</td>
<td>1111 1111</td>
</tr>
<tr>
<td>PORTD</td>
<td>RD7</td>
<td>RD6</td>
<td>RD5</td>
<td>RD4</td>
<td>RD3</td>
<td>RD2</td>
<td>RD1</td>
<td>RD0</td>
<td>—</td>
<td>1111 1111</td>
</tr>
<tr>
<td>PORTE</td>
<td>RC7</td>
<td>RC6</td>
<td>RC5</td>
<td>RC4</td>
<td>RC3</td>
<td>RC2</td>
<td>RC1</td>
<td>RC0</td>
<td>—</td>
<td>1111 1111</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>—</td>
<td>1111 1111</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>0000 0000</td>
<td>58, 132</td>
</tr>
<tr>
<td>SPBRGH1</td>
<td>EUSART1 Baud Rate Generator Register High Byte</td>
<td>0000 0000</td>
<td>58, 273</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAUDCON1</td>
<td>ABDVF</td>
<td>RCIDL</td>
<td>RXDTP</td>
<td>TXCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>0100 0-00</td>
<td>58, 273</td>
</tr>
<tr>
<td>SPBRGH2</td>
<td>EUSART2 Baud Rate Generator Register High Byte</td>
<td>0000 0000</td>
<td>58, 273</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BAUDCON2</td>
<td>ABDOVF</td>
<td>RCIDL</td>
<td>RXDTP</td>
<td>TXCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>0100 0-00</td>
<td>58, 273</td>
</tr>
<tr>
<td>TMR3H</td>
<td>Timer3 Register High Byte</td>
<td>xxxx xxxx</td>
<td>58, 194</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR3L</td>
<td>Timer3 Register Low Byte</td>
<td>xxxx xxxx</td>
<td>58, 194</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3CON</td>
<td>RD16</td>
<td>T3CCP2</td>
<td>T3CKPS1</td>
<td>T3CKPS0</td>
<td>T3CCP1</td>
<td>T3SYNC</td>
<td>TMR3CS</td>
<td>TMR3ON</td>
<td>0000 0000</td>
<td>58, 194</td>
</tr>
<tr>
<td>T4CON</td>
<td>T4CON</td>
<td>—</td>
<td>T4OUTPS3</td>
<td>T4OUTP3</td>
<td>T4OUTPS3</td>
<td>T4OUTP3</td>
<td>T4OUTP3</td>
<td>T4OUTPS3</td>
<td>T4OUTPS3</td>
<td>0000 0000</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Note</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bit 21 of the PC is only available in Serial Programming modes.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Default (legacy) SFR at this address, available when WDTCON&lt;4&gt; = 0.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Configuration SFR, overlaps with default SFR at this address; available only when WDTCON&lt;4&gt; = 1.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reset value is ‘0’ when Two-Speed Start-up is enabled and ‘1’ if disabled.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>The SSPxMSK registers are only accessible when SSPxCON2&lt;3:0&gt; = 1001.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Alternate names and definitions for these bits when the MSSP modules are operating in i2C™ Slave mode. See Section 19.4.3.2 “Address Masking Modes” for details.</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>These bits and/or registers are only available in 80-pin devices; otherwise, they are unimplemented and read as ‘0’. Reset values are shown for 80-pin devices.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>These bits are only available in select oscillator modes (FOSC2 Configuration bit = 0); otherwise, they are unimplemented.</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>The PMADDRI/PMOUTI1H and PMADDRI/PMOUTI1L register pairs share the physical registers and addresses, but have different functions determined by the module’s operating mode. See Section 11.1.2 “Data Registers” for more information.</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 5-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY) (CONTINUED)

<table>
<thead>
<tr>
<th>File 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>Details on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCPR4H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x x x x x x</td>
<td>59, 196</td>
</tr>
<tr>
<td>CCPR4L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x x x x x x</td>
<td>59, 196</td>
</tr>
<tr>
<td>CCP4CON</td>
<td></td>
<td></td>
<td>DC4B1</td>
<td>DC4B0</td>
<td>CCP4M3</td>
<td>CCP4M2</td>
<td>CCP4M1</td>
<td>CCP4M0</td>
<td>-- -- 00 0000</td>
<td>59, 196</td>
</tr>
<tr>
<td>CCP5H</td>
<td></td>
<td></td>
<td>DC5B1</td>
<td>DC5B0</td>
<td>CCP5M3</td>
<td>CCP5M2</td>
<td>CCP5M1</td>
<td>CCP5M0</td>
<td>-- -- 00 0000</td>
<td>59, 196</td>
</tr>
<tr>
<td>CCP5L</td>
<td></td>
<td></td>
<td>DC5B1</td>
<td>DC5B0</td>
<td>CCP5M3</td>
<td>CCP5M2</td>
<td>CCP5M1</td>
<td>CCP5M0</td>
<td>-- -- 00 0000</td>
<td>59, 196</td>
</tr>
<tr>
<td>SSP2BUF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x x x x x x</td>
<td>59, 222, 231</td>
</tr>
<tr>
<td>SSP2ADD/</td>
<td>MSPI2 Address Register (I2C™ Slave mode), MSPI Baud Rate Reload Register (I2C Master mode)</td>
<td>0000 0000</td>
<td>59, 231</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSP2MSK(5)</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>0000 0000</td>
<td>59, 238</td>
</tr>
<tr>
<td>SSP2STAT</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>0000 0000</td>
<td>59, 222, 232</td>
</tr>
<tr>
<td>SSP2CON1</td>
<td>WCEN</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</td>
<td>59, 238</td>
</tr>
<tr>
<td>SSP2CON2</td>
<td>GCEN</td>
<td>ACKST</td>
<td>ACKDT</td>
<td>ACKEN</td>
<td>RCEN</td>
<td>PEN</td>
<td>RSEN/</td>
<td>SEN</td>
<td>0000 0000</td>
<td>59, 234, 268</td>
</tr>
<tr>
<td>CMSTAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>----- 11 11</td>
<td>59, 303</td>
</tr>
<tr>
<td>PMADDRH /</td>
<td>CS2</td>
<td>CS1</td>
<td>Parallel Master Port Address High Byte</td>
<td>0000 0000</td>
<td>60, 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMDOUT1H(9)</td>
<td></td>
<td></td>
<td>Parallel Port Out Data High Byte (Buffer 1)</td>
<td>0000 0000</td>
<td>60, 161</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMADDR/L</td>
<td>Parallel Master Port Address Low Byte</td>
<td>0000 0000</td>
<td>60, 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMDOUT1L(9)</td>
<td></td>
<td></td>
<td>Parallel Port Out Data Low Byte (Buffer 0)</td>
<td>0000 0000</td>
<td>60, 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMDIN1H</td>
<td>Parallel Port In Data High Byte (Buffer 1)</td>
<td>0000 0000</td>
<td>60, 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMDIN1L</td>
<td>Parallel Port In Data Low Byte (Buffer 0)</td>
<td>0000 0000</td>
<td>60, 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMCONH</td>
<td>PMPEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 0</td>
<td>60, 165</td>
</tr>
<tr>
<td>PMCONL</td>
<td>CSF1</td>
<td>CSF0</td>
<td>ALP</td>
<td>CS2P</td>
<td>CS1P</td>
<td>BEP</td>
<td>WRSP</td>
<td>RDSP</td>
<td>0000 0000</td>
<td>60, 153</td>
</tr>
<tr>
<td>PMPMODEH</td>
<td>BUSY</td>
<td>IROM1</td>
<td>IROM0</td>
<td>INC1</td>
<td>INC0</td>
<td>MODE16</td>
<td>MODE1</td>
<td>MODE0</td>
<td>0000 0000</td>
<td>60, 154</td>
</tr>
<tr>
<td>PMPMODEL</td>
<td>WAITB1</td>
<td>WAITB0</td>
<td>WAITM3</td>
<td>WAITM2</td>
<td>WAITM1</td>
<td>WAITM0</td>
<td>WAITE1</td>
<td>WAITE0</td>
<td>0000 0000</td>
<td>60, 155</td>
</tr>
<tr>
<td>PMDOUT2H</td>
<td>Parallel Port Out Data High Byte (Buffer 3)</td>
<td>0000 0000</td>
<td>60, 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMDOUT2L</td>
<td>Parallel Port Out Data Low Byte (Buffer 2)</td>
<td>0000 0000</td>
<td>60, 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMDIN2H</td>
<td>Parallel Port In Data High Byte (Buffer 3)</td>
<td>0000 0000</td>
<td>60, 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMDIN2L</td>
<td>Parallel Port In Data Low Byte (Buffer 2)</td>
<td>0000 0000</td>
<td>60, 158</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMEIH</td>
<td>PTEN15</td>
<td>PTEN14</td>
<td>PTEN13</td>
<td>PTEN12</td>
<td>PTEN11</td>
<td>PTEN10</td>
<td>PTEN9</td>
<td>PTEN8</td>
<td>0000 0000</td>
<td>60, 165</td>
</tr>
<tr>
<td>PMEL</td>
<td>PTEN7</td>
<td>PTEN6</td>
<td>PTEN5</td>
<td>PTEN4</td>
<td>PTEN3</td>
<td>PTEN2</td>
<td>PTEN1</td>
<td>PTEN0</td>
<td>0000 0000</td>
<td>60, 165</td>
</tr>
<tr>
<td>PMSTATH</td>
<td>IBF</td>
<td>IBOV</td>
<td></td>
<td></td>
<td>IB3F</td>
<td>IB2F</td>
<td>IB1F</td>
<td>IB0F</td>
<td>0000 0000</td>
<td>60, 156</td>
</tr>
<tr>
<td>PMSTATH</td>
<td>OBE</td>
<td>OBUF</td>
<td></td>
<td></td>
<td>OBE3</td>
<td>OBE2</td>
<td>OBE1</td>
<td>OBE0</td>
<td>0000 0000</td>
<td>60, 157</td>
</tr>
</tbody>
</table>

**Legend:**
- x = unknown, u = unchanged, - = unimplemented, q = value depends on condition. **Bold** indicates shared access SFRs.

**Note:**
1: Bit 21 of the PC is only available in Serial Programming modes.
2: Default (legacy) SFR at this address, available when WDTCON<4> = 0.
3: Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.
4: Reset value is '0' when Two-Speed Start-up is enabled and '1' if disabled.
5: The SSPxMSK registers are only accessible when SSPxCON2<3:0> = 1001.
6: Alternate names and definitions for these bits when the MSSP modules are operating in I²C™ Slave mode. See Section 19.4.3.2 "Address Masking Modes" for details.
7: These bits and/or registers are only available in 80-pin devices; otherwise, they are unimplemented and read as '0'. Reset values are shown for 80-pin devices.
8: These bits are only available in select oscillator modes (FOSC2 Configuration bit = 0); otherwise, they are unimplemented.
9: The PMADDRH/PMDOUT1H and PMADDR/L/PMDOUT1L register pairs share the physical registers and addresses, but have different functions determined by the module’s operating mode. See Section 11.1.2 "Data Registers" for more information.
5.3.5 STATUS REGISTER

The STATUS register, shown in Register 5-4, contains the arithmetic status of the ALU. The STATUS register can be the operand for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, then the write to these five bits is disabled.

These bits are set or cleared according to the device logic. Therefore, the result of an instruction with the STATUS register as destination may be different than intended. For example, CLRF STATUS will set the Z bit but leave the other bits unchanged. The STATUS register then reads back as '0000 u uuu1'. It is recommended, therefore, that only BCF, BSF, SWAPF, MOVFF and MOVWF instructions are used to alter the STATUS register because these instructions do not affect the Z, C, DC, OV or N bits in the STATUS register.

For other instructions not affecting any Status bits, see the instruction set summaries in Table 25-2 and Table 25-3.

Note: The C and DC bits operate as a borrow and digit borrow bit respectively, in subtraction.

REGISTER 5-4: STATUS REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-x</th>
<th>R/W-x</th>
<th>R/W-x</th>
<th>R/W-x</th>
<th>R/W-x</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>N</td>
<td>OV</td>
<td>Z</td>
<td>DC(1)</td>
<td>C(2)</td>
</tr>
</tbody>
</table>

Legend:

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

bit 7-5 Unimplemented: Read as ‘0’

bit 4 N: Negative bit
This bit is used for signed arithmetic (2’s complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 OV: Overflow bit
This bit is used for signed arithmetic (2’s complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

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/borrow bit(1)
For ADDWF, ADDLW, SUBLW and SUBWF instructions:
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(2)
For ADDWF, ADDLW, SUBLW and SUBWF instructions:
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 2’s complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either bit 4 or bit 3 of the source register.

2: For borrow, the polarity is reversed. A subtraction is executed by adding the 2’s complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low-order bit of the source register.
5.4 Data Addressing Modes

| Note: | The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See Section 5.6 “Data Memory and the Extended Instruction Set” for more information. |

While the program memory can be addressed in only one way – through the program counter – information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:
- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in Section 5.6.1 “Indexed Addressing with Literal Offset”.

5.4.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device, or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include SLEEP, RESET and DAW.

Other instructions work in a similar way, but require an additional explicit argument in the opcode. This is known as Literal Addressing mode, because they require some literal value as an argument. Examples include ADDLW and MOVLW, which respectively, add or move a literal value to the W register. Other examples include CALL and GOTO, which include a 20-bit program memory address.

5.4.2 DIRECT ADDRESSING

Direct Addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit Literal Address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (Section 5.3.3 “General Purpose Register File”), or a location in the Access Bank (Section 5.3.2 “Access Bank”) as the data source for the instruction.

The Access RAM bit ‘a’ determines how the address is interpreted. When ‘a’ is ‘1’, the contents of the BSR (Section 5.3.1 “Bank Select Register”) are used with the address to determine the complete 12-bit address of the register. When ‘a’ is ‘0’, the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as MOVFF, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation’s results is determined by the destination bit ‘d’. When ‘d’ is ‘1’, the results are stored back in the source register, overwriting its original contents. When ‘d’ is ‘0’, the results are stored in the W register. Instructions without the ‘d’ argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

5.4.3 INDIRECT ADDRESSING

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations to be read or written to. Since the FSRs are themselves located in RAM as Special Function Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures such as tables and arrays in data memory.

The registers for Indirect Addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code using loops, such as the example of clearing an entire RAM bank in Example 5-5. It also enables users to perform Indexed Addressing and other Stack Pointer operations for program memory in data memory.

EXAMPLE 5-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

| LFSR FSR0, 100h ; |
| NEXT CLRF POSTINC0 ; Clear INDF |
| BTFSS FSR0H, 1 ; All done with Bank1? |
| BRA NEXT ; NO, clear next |
| CONTINUE ; YES, continue |
5.4.3.1 FSR Registers and the INDF Operand

At the core of Indirect Addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers, FSRnH and FSRnL. The four upper bits of the FSRnH register are not used, so each FSR pair holds a 12-bit value. This represents a value that can address the entire range of the data memory in a linear fashion. The FSR register pairs, then, serve as pointers to data memory locations.

Indirect Addressing is accomplished with a set of Indirect File Operands, INDF0 through INDF2. These can be thought of as "virtual" registers: they are mapped in the SFR space but are not physically implemented. Reading or writing to a particular INDF register actually accesses its corresponding FSR register pair. A read from INDF1, for example, reads the data at the address indicated by FSR1H:FSR1L. Instructions that use the INDF registers as operands actually use the contents of their corresponding FSR as a pointer to the instruction's target. The INDF operand is just a convenient way of using the pointer.

Because Indirect Addressing uses a full 12-bit address, data RAM banking is not necessary. Thus, the current contents of the BSR and the Access RAM bit have no effect on determining the target address.

FIGURE 5-9: INDIRECT ADDRESSING

Using an instruction with one of the Indirect Addressing registers as the operand....

...uses the 12-bit address stored in the FSR pair associated with that register....

...to determine the data memory location to be used in that operation.

In this case, the FSR1 pair contains FCCh. This means the contents of location FCCh will be added to that of the W register and stored back in FCCh.
5.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are "virtual" registers that cannot be indirectly read or written to. Accessing these registers actually accesses the associated FSR register pair, but also performs a specific action on its stored value. They are:

- **POSTDEC**: accesses the FSR value, then automatically decrements it by '1' afterwards
- **POSTINC**: accesses the FSR value, then automatically increments it by '1' afterwards
- **PREINC**: increments the FSR value by '1', then uses it in the operation
- **PLUSW**: adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the new value in the operation

In this context, accessing an INDF register uses the value in the FSR registers without changing them. Similarly, accessing a PLUSW register gives the FSR value offset by the value in the W register; neither value is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR registers.

Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, rollovers of the FSRnL register from FFh to 00h carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (e.g., Z, N, OV, etc.).

The PLUSW register can be used to implement a form of Indexed Addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

5.4.3.3 Operations by FSRs on FSRs

Indirect Addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specific case, assume that FSR0H:FSR0L contains FE7h, the address of INDF1. Attempts to read the value of the INDF1, using INDF0 as an operand, will return 00h. Attempts to write to INDF1, using INDF0 as the operand, will result in a **NOP**.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to INDF2 or POSTDEC2 will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, particularly if their code uses Indirect Addressing.

Similarly, operations by Indirect Addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.
5.5 Program Memory and the Extended Instruction Set

The operation of program memory is unaffected by the use of the extended instruction set. Enabling the extended instruction set adds five additional two-word commands to the existing PIC18 instruction set: ADDFSR, CALLW, MOVSF, MOVSS and SUBFSR. These instructions are executed as described in Section 5.2.4 “Two-Word Instructions”.

5.6 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different. This is due to the introduction of a new addressing mode for the data memory space. This mode also alters the behavior of Indirect Addressing using FSR2 and its associated operands. What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect Addressing with FSR0 and FSR1 also remains unchanged.

5.6.1 Indexed Addressing with Literal Offset

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair and its associated file operands. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of Indexed Addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset, or Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- The use of the Access Bank is forced ('a' = 0);
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in Direct Addressing) or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

5.6.2 Instructions Affected by Indexed Literal Offset Mode

Any of the core PIC18 instructions that can use Direct Addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they use the Access Bank (Access RAM bit is ’1’) or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in Figure 5-10.

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in Section 25.2.1 “Extended Instruction Syntax”.
FIGURE 5-10: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

EXAMPLE INSTRUCTION: ADDWF, f, d, a (Opcode: 0010 01da ffff ffff)

When a = 0 and f ≥ 60h:
The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and FFFh. This is the same as locations F60h to FFFh (Bank 15) of data memory. Locations below 060h are not available in this addressing mode.

When a = 0 and f ≤ 5Fh:
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now: ADDWF [k], d where 'k' is the same as 'f'.

When a = 1 (all values of f):
The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 16 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.
5.6.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

The use of Indexed Literal Offset Addressing mode effectively changes how the lower part of Access RAM (00h to 5Fh) is mapped. Rather than containing just the contents of the bottom part of Bank 0, this mode maps the contents from Bank 0 and a user-defined “window” that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see Section 5.3.2 “Access Bank”). An example of Access Bank remapping in this addressing mode is shown in Figure 5-11.

Remapping of the Access Bank applies only to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit is ‘1’) will continue to use Direct Addressing as before. Any Indirect or Indexed Addressing operation that explicitly uses any of the indirect file operands (including FSR2) will continue to operate as standard Indirect Addressing. Any instruction that uses the Access Bank, but includes a register address of greater than 05Fh, will use Direct Addressing and the normal Access Bank map.

5.6.4 BSR IN INDEXED LITERAL OFFSET MODE

Although the Access Bank is remapped when the extended instruction set is enabled, the operation of the BSR remains unchanged. Direct Addressing, using the BSR to select the data memory bank, operates in the same manner as previously described.

FIGURE 5-11: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING

Example Situation:
ADDWF f, d, a
FSR2H:FSR2L = 120h

Locations in the region from the FSR2 Pointer (120h) to the pointer plus 05Fh (17Fh) are mapped to the bottom of the Access RAM (000h-05Fh).

Special Function Registers at F60h through FFFh are mapped to 60h through FFh, as usual.

Bank 0 addresses below 5Fh are not available in this mode. They can still be addressed by using the BSR.
6.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed on one byte at a time. A write to program memory is executed on blocks of 64 bytes at a time or two bytes at a time. Program memory is erased in blocks of 1024 bytes at a time. A bulk erase operation may not be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

6.1 Table Reads and Table Writes

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- **Table Read (TBLRD)**
- **Table Write (TBLWT)**

The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 6-1 shows the operation of a table read with program memory and data RAM.

Table write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in Section 6.5 “Writing to Flash Program Memory”. Figure 6-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word-aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word-aligned.

**FIGURE 6-1: TABLE READ OPERATION**

- **Table Pointer**
  - TBLPTRU
  - TBLPTRH
  - TBLPTRL
- **Program Memory**
- **Table Latch (8-bit)**
  - TABLAT

**Note 1:** Table Pointer register points to a byte in program memory.
6.2 Control Registers

Several control registers are used in conjunction with the TBLRD and TBLWT instructions. These include the:

- EECON1 register
- EECON2 register
- TABLAT register
- TBLPTR registers

6.2.1 EECON1 AND EECON2 REGISTERS

The EECON1 register (Register 6-1) is the control register for memory accesses. The EECON2 register is not a physical register; it is used exclusively in the memory write and erase sequences. Reading EECON2 will read all ‘0’s.

The WPROG bit, when set, allows the user to program a single word (two bytes) upon the execution of the WR command. If this bit is cleared, the WR command programs a block of 64 bytes.

The FREE bit, when set, will allow a program memory erase operation. When FREE is set, the erase operation is initiated on the next WR command. When FREE is clear, only writes are enabled.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set in hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

Note: During normal operation, the WRERR is read as ‘1’. This can indicate that a write operation was prematurely terminated by a Reset, or a write operation was attempted improperly.

The WR control bit initiates write operations. The bit cannot be cleared, only set, in software. It is cleared in hardware at the completion of the write operation.

Table Pointer actually points to one of 64 holding registers, the address of which is determined by TBLPTRL<5:0>. The process for physically writing data to the program memory array is discussed in Section 6.5 “Writing to Flash Program Memory”.

Note 1: Table Pointer actually points to one of 64 holding registers, the address of which is determined by TBLPTRL<5:0>. The process for physically writing data to the program memory array is discussed in Section 6.5 “Writing to Flash Program Memory”.

Instruction: TBLWT*

Table Pointer(1)

TBLPTRU TBLPTRH TBLPTRL

Program Memory (TBLPTR)

Table Latch (8-bit)

Note: The FREE bit, when set, will allow a program memory erase operation. When FREE is set, the erase operation is initiated on the next WR command. When FREE is clear, only writes are enabled.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set in hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

Note: During normal operation, the WRERR is read as ‘1’. This can indicate that a write operation was prematurely terminated by a Reset, or a write operation was attempted improperly.

The WR control bit initiates write operations. The bit cannot be cleared, only set, in software. It is cleared in hardware at the completion of the write operation.
## REGISTER 6-1: EECON1: EEPROM CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>S = Set-only bit (cannot be cleared in software)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 5</td>
<td>W = Writable bit</td>
</tr>
<tr>
<td>WPROG:</td>
<td>Unimplemented bit, read as ‘0’</td>
</tr>
<tr>
<td>bit 4</td>
<td>U = Unimplemented bit, read as ‘0’</td>
</tr>
<tr>
<td>FREE:</td>
<td>bit 5-6 = Value at POR</td>
</tr>
<tr>
<td>bit 3</td>
<td>bit 5-6 = Value at POR</td>
</tr>
<tr>
<td>WRERR:</td>
<td>bit 5-6 = Value at POR</td>
</tr>
<tr>
<td>bit 2</td>
<td>bit 5-6 = Value at POR</td>
</tr>
<tr>
<td>WREN:</td>
<td>bit 5-6 = Value at POR</td>
</tr>
<tr>
<td>bit 1</td>
<td>bit 5-6 = Value at POR</td>
</tr>
<tr>
<td>WR:</td>
<td>bit 5-6 = Value at POR</td>
</tr>
<tr>
<td>bit 0</td>
<td>bit 5-6 = Value at POR</td>
</tr>
</tbody>
</table>

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

#### bit 5
**WPROG:** One Word-Wide Program bit
  - **1** = Program 2 bytes on the next WR command
  - **0** = Program 64 bytes on the next WR command

#### bit 4
**FREE:** Flash Row Erase Enable bit
  - **1** = Erase the program memory row addressed by TBLPTR on the next WR command
    (cleared by completion of erase operation)
  - **0** = Perform write only

#### bit 3
**WRERR:** Flash Program Error Flag bit
  - **1** = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation, or an improper write attempt)
  - **0** = The write operation completed

#### bit 2
**WREN:** Flash Program Write Enable bit
  - **1** = Allows write cycles to Flash program memory
  - **0** = Inhibits write cycles to Flash program memory

#### bit 1
**WR:** Write Control bit
  - **1** = Initiates a program memory erase cycle or write cycle
    (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
  - **0** = Write cycle is complete

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

### Note 1:
When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.
6.2.2 TABLE LATCH REGISTER (TABLAT)
The Table Latch (TABLAT) is an 8-bit register mapped into the SFR space. The Table Latch register is used to hold 8-bit data during data transfers between program memory and data RAM.

6.2.3 TABLE POINTER REGISTER (TBLPTR)
The Table Pointer (TBLPTR) register addresses a byte within the program memory. The TBLPTR is comprised of three SFR registers: Table Pointer Upper Byte, Table Pointer High Byte and Table Pointer Low Byte (TBLPTRU:TBLPTRH:TBLPTRL). These three registers join to form a 22-bit wide pointer. The low-order 21 bits allow the device to address up to 2 Mbytes of program memory space. The 22nd bit allows access to the device ID, the user ID and the Configuration bits.

The Table Pointer register, TBLPTR, is used by the TBLRD and TBLWT instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations are shown in Table 6-1. These operations on the TBLPTR only affect the low-order 21 bits.

6.2.4 TABLE POINTER BOUNDARIES
TBLPTR is used in reads, writes and erases of the Flash program memory.

When a TBLRD is executed, all 22 bits of the TBLPTR determine which byte is read from program memory into TABLAT.

When a TBLWT is executed, the seven LSbs of the Table Pointer register (TBLPTR<6:0>) determine which of the 64 program memory holding registers is written to. When the timed write to program memory begins (via the WR bit), the 12 MSbs of the TBLPTR (TBLPTR<21:10>) determine which program memory block of 1024 bytes is written to. For more detail, see Section 6.5 “Writing to Flash Program Memory”.

When an erase of program memory is executed, the 12 MSbs of the Table Pointer register point to the 1024-byte block that will be erased. The Least Significant bits are ignored.

Figure 6-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

### TABLE 6-1: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS

<table>
<thead>
<tr>
<th>Example</th>
<th>Operation on Table Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBLRD*</td>
<td>TBLPTR is not modified</td>
</tr>
<tr>
<td>TBLWT*</td>
<td>TBLPTR is not modified</td>
</tr>
<tr>
<td>TBLRD*+</td>
<td>TBLPTR is incremented after the read/write</td>
</tr>
<tr>
<td>TBLWT*+</td>
<td>TBLPTR is incremented after the read/write</td>
</tr>
<tr>
<td>TBLRD*--</td>
<td>TBLPTR is decremented after the read/write</td>
</tr>
<tr>
<td>TBLWT*--</td>
<td>TBLPTR is decremented after the read/write</td>
</tr>
<tr>
<td>TBLRD++</td>
<td>TBLPTR is incremented before the read/write</td>
</tr>
<tr>
<td>TBLWT++</td>
<td>TBLPTR is incremented before the read/write</td>
</tr>
</tbody>
</table>

### FIGURE 6-3: TABLE POINTER BOUNDARIES BASED ON OPERATION

[Diagram showing the boundaries of TBLPTR based on program memory operations]
6.3 Reading the Flash Program Memory

The TBLRD instruction is used to retrieve data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

TBLPTR points to a byte address in program space. Executing TBLRD places the byte pointed to into TABLAT. In addition, TBLPTR can be modified automatically for the next table read operation.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word. Figure 6-4 shows the interface between the internal program memory and the TABLAT.

FIGURE 6-4: READS FROM FLASH PROGRAM MEMORY

EXAMPLE 6-1: READING A FLASH PROGRAM MEMORY WORD

```
MOVLW CODE_ADDR_UPPER ; Load TBLPTR with the base address of the word
MOVWF TBLPTRU
MOVLW CODE_ADDR_HIGH
MOVWF TBLPTRH
MOVLW CODE_ADDR_LOW
MOVWF TBLPTRL

READ_WORD
    TBLRD + ; read into TABLAT and increment
    MOVF TABLAT, W ; get data
    MOVWF WORD_EVEN
    TBLRD + ; read into TABLAT and increment
    MOVF TABLAT, W ; get data
    MOVWF WORD_ODD
```
6.4 Erasing Flash Program Memory

The minimum erase block is 512 words or 1024 bytes. Only through the use of an external programmer, or through ICSP control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

When initiating an erase sequence from the microcontroller itself, a block of 1024 bytes of program memory is erased. The Most Significant 12 bits of the TBLPTR<21:10> point to the block being erased. TBLPTR<9:0> are ignored.

The EECON1 register commands the erase operation. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation. For protection, the write initiate sequence for EECON2 must be used.

A long write is necessary for erasing the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

6.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory location is:

1. Load Table Pointer register with address of row being erased.
2. Set the WREN and FREE bits (EECON1<2,4>) to enable the erase operation.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit. This will begin the row erase cycle.
7. The CPU will stall for duration of the erase for Tiw (see parameter D133A).
8. Re-enable interrupts.

**EXAMPLE 6-2: ERASING A FLASH PROGRAM MEMORY ROW**

```
MOVLW CODE_ADDR_UPPER ; load TBLPTR with the base
MOVWF TBLPTRU
MOVLW CODE_ADDR_HIGH
MOVWF TBLPTRH
MOVLW CODE_ADDR_LOW
MOVWF TBLPTRL

ERASE_ROW

BSF EECON1, FREE ; enable Row Erase operation
BCF INTCON, GIE ; disable interrupts

Required Sequence

MOVLW 55h
MOVWF EECON2 ; write 55h
MOVLW 0AAh
MOVWF EECON2 ; write 0AAh
BSF EECON1, WR ; start erase (CPU stall)
BSF INTCON, GIE ; re-enable interrupts
```
6.5 Writing to Flash Program Memory

The programming block is 32 words or 64 bytes. Programming one word or two bytes at a time is also supported.

Table writes are used internally to load the holding registers needed to program the Flash memory. There are 64 holding registers used by the table writes for programming.

Since the Table Latch (TABLAT) is only a single byte, the TBLWT instruction may need to be executed 64 times for each programming operation (if WPROG = 0). All of the table write operations will essentially be short writes because only the holding registers are written. At the end of updating the 64 holding registers, the EECON1 register must be written to in order to start the programming operation with a long write.

The long write is necessary for programming the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

FIGURE 6-5: TABLE WRITES TO FLASH PROGRAM MEMORY

6.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

The sequence of events for programming an internal program memory location should be:

1. Read 1024 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load Table Pointer register with address being erased.
4. Execute the row erase procedure.
5. Load Table Pointer register with address of first byte being written, minus 1.
6. Write the 64 bytes into the holding registers with auto-increment.
7. Set the WREN bit (EECON1<2>) to enable byte writes.
8. Disable interrupts.
9. Write 55h to EECON2.
10. Write 0AAh to EECON2.
11. Set the WR bit. This will begin the write cycle.
12. The CPU will stall for duration of the write for Tiw (parameter D133A).
13. Re-enable interrupts.
14. Repeat steps 6 through 13 until all 1024 bytes are written to program memory.
15. Verify the memory (table read).

An example of the required code is shown in Example 6-3 on the following page.

Note: Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the 64 bytes in the holding register.

Note 1: Unlike previous PIC18 Flash devices, members of the PIC18F87J11 family do not reset the holding registers after a write occurs. The holding registers must be cleared or overwritten before a programming sequence.

Note 2: To maintain the endurance of the program memory cells, each Flash byte should not be programmed more than one time between erase operations. Before attempting to modify the contents of the target cell a second time, a row erase of the target row, or a bulk erase of the entire memory, must be performed.
EXAMPLE 6-3: WRITING TO FLASH PROGRAM MEMORY

```assembly
MOVLW CODE_ADDR_UPPER ; Load TBLPTR with the base address of the memory block, minus 1
MOVWF TBLPTRU
MOVLW CODE_ADDR_HIGH
MOVWF TBLPTRH
MOVLW CODE_ADDR_LOW
MOVWF TBLPTRL

ERASE_BLOCK
BSF EECON1, WREN ; enable write to memory
BSF EECON1, FREE ; enable Row Erase operation
BCF INTCON, GIE ; disable interrupts
MOVLW 55h
MOVWF EECON2 ; write 55h
MOVLW 0Ah
MOVWF EECON2 ; write 0Ah
BSF EECON1, WR ; start erase (CPU stall)
BSF INTCON, GIE ; re-enable interrupts
MOVLW D'16'
MOVWF WRITE_COUNTER ; Need to write 16 blocks of 64 to write one erase block of 1024

RESTART_BUFFER
MOVLW D'64'
MOVWF COUNTER
MOVLW BUFFER_ADDR_HIGH ; point to buffer
MOVWF FSR0H
MOVLW BUFFER_ADDR_LOW
MOVWF FSR0L

FILL_BUFFER
... ; read the new data from I2C, SPI, PSP, USART, etc.

WRITE_BUFFER
MOVLW D'64' ; number of bytes in holding register
MOVWF COUNTER

WRITE_BYTE_TO_HREGS
MOVFF POSTINC0, WREG ; get low byte of buffer data
MOVWF TABLAT ; present data to table latch
TBLWT+* ; write data, perform a short write to internal TBLWT holding register.
DECFSZ COUNTER ; loop until buffers are full
BRA WRITE_BYTE_TO_HREGS

PROGRAM_MEMORY
BSF EECON1, WREN ; enable write to memory
BCF INTCON, GIE ; disable interrupts
MOVLW 55h
MOVWF EECON2 ; write 55h
MOVLW 0Ah
MOVWF EECON2 ; write 0Ah
BSF EECON1, WR ; start program (CPU stall)
BSF INTCON, GIE ; re-enable interrupts
BCF EECON1, WREN ; disable write to memory
DECFSZ WRITE_COUNTER ; done with one write cycle
BRA RESTART_BUFFER ; if not done replacing the erase block
```

Required Sequence

<table>
<thead>
<tr>
<th>Required Sequence</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVLW 55h</td>
<td>write 55h</td>
</tr>
<tr>
<td>MOVWF EECON2</td>
<td>write 0Ah</td>
</tr>
<tr>
<td>MOVLW 0Ah</td>
<td>start program (CPU stall)</td>
</tr>
<tr>
<td>BSF EECON1, WREN</td>
<td>re-enable interrupts</td>
</tr>
<tr>
<td>BSF INTCON, GIE</td>
<td>disable write to memory</td>
</tr>
<tr>
<td>DECFSZ WRITE_COUNTER</td>
<td>done with one write cycle</td>
</tr>
<tr>
<td>BRA RESTART_BUFFER</td>
<td>if not done replacing the erase block</td>
</tr>
</tbody>
</table>
6.5.2  FLASH PROGRAM MEMORY WRITE SEQUENCE (WORD PROGRAMMING).

The PIC18F87J11 Family of devices have a feature that allows programming a single word (two bytes). This feature is enable when the WPROG bit is set. If the memory location is already erased, the following sequence is required to enable this feature:

1. Load the Table Pointer register with the address of the data to be written
2. Write the 2 bytes into the holding registers and perform a table write
3. Set the WREN bit (EECON1<2>) to enable byte writes.
4. Disable interrupts.
5. Write 55h to EECON2.
6. Write 0AAh to EECON2.
7. Set the WR bit. This will begin the write cycle.
8. The CPU will stall for duration of the write for Tw (see parameter D133A).
9. Re-enable interrupts.

EXAMPLE 6-4: SINGLE-WORD WRITE TO FLASH PROGRAM MEMORY

```
MOVLW CODE_ADDR_UPPER ; Load TBLPTR with the base address
MOVWF TBLPTRU
MOVLW CODE_ADDR_HIGH
MOVWF TBLPTRH
MOVLW CODE_ADDR_LOW
MOVWF TBLPTRL

MOVLW DATA0
MOVWF TABLAT
TBLWT*

MOVLW DATA1
MOVWF TABLAT
TBLWT*

PROGRAM_MEMORY

BSF EECON1, WPROG ; enable single word write
BSF EECON1, WREN ; enable write to memory
BCF INTCON, GIE ; disable interrupts

Required Sequence

MOVLW 55h
MOVWF EECON2 ; write 55h
MOVLW 0AAh
MOVWF EECON2 ; write 0AAh

BSF EECON1, WR ; start program (CPU stall)
BSF INTCON, GIE ; re-enable interrupts
BCF EECON1, WPROG ; disable single word write
BCF EECON1, WREN ; disable write to memory
```
6.5.3 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

6.5.4 UNEXPECTED TERMINATION OF WRITE OPERATION

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. If the write operation is interrupted by a MCLR Reset or a WDT time-out Reset during normal operation, the user can check the WRERR bit and rewrite the location(s) as needed.

6.6 Flash Program Operation During Code Protection

See Section 24.6 “Program Verification and Code Protection” for details on code protection of Flash program memory.

### TABLE 6-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBLPTRU</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>bit 21</td>
<td>Program Memory Table Pointer Upper Byte (TBLPTR&lt;20:16&gt;)</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBPLTRH</td>
<td>Program Memory Table Pointer High Byte (TBLPTR&lt;15:8&gt;)</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLPTRL</td>
<td>Program Memory Table Pointer Low Byte (TBLPTR&lt;7:0&gt;)</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TABLAT</td>
<td>Program Memory Table Latch</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INTOIE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INTOIF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>EECON2</td>
<td>Program Memory Control Register 2 (not a physical register)</td>
<td>57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EECON1</td>
<td>—</td>
<td>—</td>
<td>WPROG</td>
<td>FREE</td>
<td>WRERR</td>
<td>WREN</td>
<td>WR</td>
<td>—</td>
<td>57</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used during Flash program memory access.
7.0 EXTERNAL MEMORY BUS

The External Memory Bus (EMB) allows the device to access external memory devices (such as Flash, EPROM, SRAM, etc.) as program or data memory. It supports both 8 and 16-Bit Data Width modes and three address widths of up to 20 bits.

The bus is implemented with 28 pins, multiplexed across four I/O ports. Three ports (PORTD, PORTE and PORTH) are multiplexed with the address/data bus for a total of 20 available lines, while PORTJ is multiplexed with the bus control signals.

A list of the pins and their functions is provided in Table 7-1.

<table>
<thead>
<tr>
<th>Name</th>
<th>Port</th>
<th>Bit</th>
<th>External Memory Bus Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD0/AD0</td>
<td>PORTD</td>
<td>0</td>
<td>Address bit 0 or Data bit 0</td>
</tr>
<tr>
<td>RD1/AD1</td>
<td>PORTD</td>
<td>1</td>
<td>Address bit 1 or Data bit 1</td>
</tr>
<tr>
<td>RD2/AD2</td>
<td>PORTD</td>
<td>2</td>
<td>Address bit 2 or Data bit 2</td>
</tr>
<tr>
<td>RD3/AD3</td>
<td>PORTD</td>
<td>3</td>
<td>Address bit 3 or Data bit 3</td>
</tr>
<tr>
<td>RD4/AD4</td>
<td>PORTD</td>
<td>4</td>
<td>Address bit 4 or Data bit 4</td>
</tr>
<tr>
<td>RD5/AD5</td>
<td>PORTD</td>
<td>5</td>
<td>Address bit 5 or Data bit 5</td>
</tr>
<tr>
<td>RD6/AD6</td>
<td>PORTD</td>
<td>6</td>
<td>Address bit 6 or Data bit 6</td>
</tr>
<tr>
<td>RD7/AD7</td>
<td>PORTD</td>
<td>7</td>
<td>Address bit 7 or Data bit 7</td>
</tr>
<tr>
<td>RE0/AD8</td>
<td>PORTE</td>
<td>0</td>
<td>Address bit 8 or Data bit 8</td>
</tr>
<tr>
<td>RE1/AD9</td>
<td>PORTE</td>
<td>1</td>
<td>Address bit 9 or Data bit 9</td>
</tr>
<tr>
<td>RE2/AD10</td>
<td>PORTE</td>
<td>2</td>
<td>Address bit 10 or Data bit 10</td>
</tr>
<tr>
<td>RE3/AD11</td>
<td>PORTE</td>
<td>3</td>
<td>Address bit 11 or Data bit 11</td>
</tr>
<tr>
<td>RE4/AD12</td>
<td>PORTE</td>
<td>4</td>
<td>Address bit 12 or Data bit 12</td>
</tr>
<tr>
<td>RE5/AD13</td>
<td>PORTE</td>
<td>5</td>
<td>Address bit 13 or Data bit 13</td>
</tr>
<tr>
<td>RE6/AD14</td>
<td>PORTE</td>
<td>6</td>
<td>Address bit 14 or Data bit 14</td>
</tr>
<tr>
<td>RE7/AD15</td>
<td>PORTE</td>
<td>7</td>
<td>Address bit 15 or Data bit 15</td>
</tr>
<tr>
<td>RH0/A16</td>
<td>PORTH</td>
<td>0</td>
<td>Address bit 16</td>
</tr>
<tr>
<td>RH1/A17</td>
<td>PORTH</td>
<td>1</td>
<td>Address bit 17</td>
</tr>
<tr>
<td>RH2/A18</td>
<td>PORTH</td>
<td>2</td>
<td>Address bit 18</td>
</tr>
<tr>
<td>RH3/A19</td>
<td>PORTH</td>
<td>3</td>
<td>Address bit 19</td>
</tr>
<tr>
<td>RJ0/ALE</td>
<td>PORTJ</td>
<td>0</td>
<td>Address Latch Enable (ALE) Control pin</td>
</tr>
<tr>
<td>RJ1/OE</td>
<td>PORTJ</td>
<td>1</td>
<td>Output Enable (OE) Control pin</td>
</tr>
<tr>
<td>RJ2/WRL</td>
<td>PORTJ</td>
<td>2</td>
<td>Write Low (WRL) Control pin</td>
</tr>
<tr>
<td>RJ3/WRH</td>
<td>PORTJ</td>
<td>3</td>
<td>Write High (WRH) Control pin</td>
</tr>
<tr>
<td>RJ4/BA0</td>
<td>PORTJ</td>
<td>4</td>
<td>Byte Address bit 0 (BA0)</td>
</tr>
<tr>
<td>RJ5/CE</td>
<td>PORTJ</td>
<td>5</td>
<td>Chip Enable (CE) Control pin</td>
</tr>
<tr>
<td>RJ6/LB</td>
<td>PORTJ</td>
<td>6</td>
<td>Lower Byte Enable (LB) Control pin</td>
</tr>
<tr>
<td>RJ7/UB</td>
<td>PORTJ</td>
<td>7</td>
<td>Upper Byte Enable (UB) Control pin</td>
</tr>
</tbody>
</table>

For the sake of clarity, only I/O port and external bus assignments are shown here. One or more additional multiplexed features may be available on some pins.
7.1 External Memory Bus Control

The operation of the interface is controlled by the MEMCON register (Register 7-1). This register is available in all program memory operating modes except Microcontroller mode. In this mode, the register is disabled and cannot be written to.

The EBDIS bit (MEMCON<7>) controls the operation of the bus and related port functions. Clearing EBDIS enables the interface and disables the I/O functions of the ports, as well as any other functions multiplexed to those pins. Setting the bit enables the I/O ports and other functions, but allows the interface to override everything else on the pins when an external memory operation is required. By default, the external bus is always enabled and disables all other I/O.

The operation of the EBDIS bit is also influenced by the program memory mode being used. This is discussed in more detail in Section 7.5 “Program Memory Modes and the External Memory Bus”.

The WAIT bits allow for the addition of wait states to external memory operations. The use of these bits is discussed in Section 7.3 “Wait States”.

The WM bits select the particular operating mode used when the bus is operating in 16-Bit Data Width mode. These are discussed in more detail in Section 7.6 “16-Bit Data Width Modes”. These bits have no effect when an 8-bit Data Width mode is selected.

The MEMCON register (see Register 7-1) shares the same memory space as the PR2 register and can be alternately selected based on the designation of the ADSHR bit in the WDTCON register (see Register 24-9).

**REGISTER 7-1: MEMCON: EXTERNAL MEMORY BUS CONTROL REGISTER**

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>WM1</th>
<th>WM0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBDIS</td>
<td>---</td>
<td>WAIT1</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

Legend:

<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td>R = Readable bit</td>
<td>W = Writable bit</td>
<td>U = Unimplemented bit, read as ‘0’</td>
</tr>
<tr>
<td>-n = Value at POR</td>
<td>‘1’ = Bit is set</td>
<td>‘0’ = Bit is cleared</td>
</tr>
</tbody>
</table>

**bit 7**

**EBDIS**: External Bus Disable bit

1 = External bus enabled when microcontroller accesses external memory; otherwise, all external bus drivers are mapped as I/O ports

0 = External bus always enabled, I/O ports are disabled

**bit 6**

**Unimplemented**: Read as ‘0’

**bit 5-4**

**WAIT1:WAIT0**: Table Reads and Writes Bus Cycle Wait Count bits

11 = Table reads and writes will wait 0 TCY

10 = Table reads and writes will wait 1 TCY

01 = Table reads and writes will wait 2 TCY

00 = Table reads and writes will wait 3 TCY

**bit 3-2**

**Unimplemented**: Read as ‘0’

**bit 1-0**

**WM1:WM0**: TBLWT Operation with 16-Bit Data Bus Width Select bits

1x = Word Write mode: TABLAT word output, WRH active when TABLAT written

01 = Byte Select mode: TABLAT data copied on both MSB and LSB, WRH and (UB or LB) will activate

00 = Byte Write mode: TABLAT data copied on both MSB and LSB, WRH or WRL will activate
7.2 Address and Data Width

The PIC18F87J11 Family of devices can be independently configured for different address and data widths on the same memory bus. Both address and data width are set by Configuration bits in the CONFIG3L register. As Configuration bits, this means that these options can only be configured by programming the device and are not controllable in software.

The BW bit selects an 8-bit or 16-bit data bus width. Setting this bit (default) selects a data width of 16 bits.

The EMB1:EMB0 bits determine both the program memory operating mode and the address bus width. The available options are 20-bit, 16-bit and 12-bit, as well as Microcontroller mode (external bus disabled). Selecting a 16-bit or 12-bit width makes a corresponding number of high-order lines available for I/O functions. These pins are no longer affected by the setting of the EBDIS bit. For example, selecting a 16-Bit Addressing mode (EMB1:EMB0 = 01) disables A19:A16 and allows PORTH<3:0> to function without interruptions from the bus. Using the smaller address widths allows users to tailor the memory bus to the size of the external memory space for a particular design while freeing up pins for dedicated I/O operation.

Because the EMB bits have the effect of disabling pins for memory bus operations, it is important to always select an address width at least equal to the data width. If a 12-bit address width is used with a 16-bit data width, the upper four bits of data will not be available on the bus.

All combinations of address and data widths require multiplexing of address and data information on the same lines. The address and data multiplexing, as well as I/O ports made available by the use of smaller address widths, are summarized in Table 7-2.

### 7.2.1 ADDRESS SHIFTING ON THE EXTERNAL BUS

By default, the address presented on the external bus is the value of the PC. In practical terms, this means that addresses in the external memory device below the top of on-chip memory are unavailable to the microcontroller. To access these physical locations, the glue logic between the microcontroller and the external memory must somehow translate addresses.

To simplify the interface, the external bus offers an extension of Extended Microcontroller mode that automatically performs address shifting. This feature is controlled by the EASHFT Configuration bit. Setting this bit offsets addresses on the bus by the size of the microcontroller’s on-chip program memory and sets the bottom address at 0000h. This allows the device to use the entire range of physical addresses of the external memory.

### 7.2.2 21-BIT ADDRESSING

As an extension of 20-bit address width operation, the external memory bus can also fully address a 2-Mbyte memory space. This is done by using the Bus Address bit 0 (BA0) control line as the Least Significant bit of the address. The UB and LB control signals may also be used with certain memory devices to select the upper and lower bytes within a 16-bit wide data word.

This addressing mode is available in both 8-bit and certain 16-Bit Data Width modes. Additional details are provided in Section 7.6.3 “16-Bit Byte Select Mode” and Section 7.7 “8-Bit Data Width Mode”.

---

### TABLE 7-2: ADDRESS AND DATA LINES FOR DIFFERENT ADDRESS AND DATA WIDTHS

<table>
<thead>
<tr>
<th>Data Width</th>
<th>Address Width</th>
<th>Multiplexed Data and Address Lines (and Corresponding Ports)</th>
<th>Address Only Lines (and Corresponding Ports)</th>
<th>Ports Available for I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit</td>
<td>12-bit</td>
<td>AD7:AD0 (PORTD&lt;7:0&gt;)</td>
<td>AD11:AD8 (PORTE&lt;3:0&gt;)</td>
<td>PORTE&lt;7:4&gt;, All of PORTH</td>
</tr>
<tr>
<td>16-bit</td>
<td>16-bit</td>
<td>AD15:AD8 (PORTE&lt;7:0&gt;)</td>
<td></td>
<td>All of PORTH</td>
</tr>
<tr>
<td>20-bit</td>
<td>20-bit</td>
<td>A19:A16, AD15:AD8 (PORTH&lt;3:0&gt;, PORTE&lt;7:0&gt;)</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>16-bit</td>
<td>16-bit</td>
<td>AD15:AD0 (PORTD&lt;7:0&gt;, PORTE&lt;7:0&gt;)</td>
<td></td>
<td>—</td>
</tr>
<tr>
<td>20-bit</td>
<td>20-bit</td>
<td>A19:A16 (PORTE&lt;3:0&gt;)</td>
<td></td>
<td>—</td>
</tr>
</tbody>
</table>
7.3 Wait States

While it may be assumed that external memory devices will operate at the microcontroller clock rate, this is often not the case. In fact, many devices require longer times to write or retrieve data than the time allowed by the execution of table read or table write operations. To compensate for this, the external memory bus can be configured to add a fixed delay to each table operation using the bus. Wait states are enabled by setting the WAIT Configuration bit. When enabled, the amount of delay is set by the WAIT1:WAIT0 bits (MEMCON<5:4>). The delay is based on multiples of microcontroller instruction cycle time and are added following the instruction cycle when the table operation is executed. The range is from no delay to 3 TCY (default value).

7.4 Port Pin Weak Pull-ups

With the exception of the upper address lines, A19:A16, the pins associated with the external memory bus are equipped with weak pull-ups. The pull-ups are controlled by the upper three bits of the PORTG register (PORTG<7:5>). They are named RDPU, REP and RJPU and control pull-ups on PORTD, PORTE and PORTJ, respectively. Setting one of these bits enables the corresponding pull-ups for that port. All pull-ups are disabled by default on all device Resets.

In Extended Microcontroller mode, the port pull-ups can be useful in preserving the memory state on the external bus while the bus is temporarily disabled (EBDIS = ‘1’).

7.5 Program Memory Modes and the External Memory Bus

The PIC18F87J11 Family of devices is capable of operating in one of two program memory modes, using combinations of on-chip and external program memory. The functions of the multiplexed port pins depend on the program memory mode selected, as well as the setting of the EBDIS bit.

In Microcontroller Mode, the bus is not active and the pins have their port functions only. Writes to the MEMCOM register are not permitted. The Reset value of EBDIS (‘0’) is ignored and EMB pins behave as I/O ports.

In Extended Microcontroller Mode, the external program memory bus shares I/O port functions on the pins. When the device is fetching or doing table read/table write operations on the external program memory space, the pins will have the external bus function.

If the device is fetching and accessing internal program memory locations only, the EBDIS control bit will change the pins from external memory to I/O port functions. When EBDIS = 0, the pins function as the external bus. When EBDIS = 1, the pins function as I/O ports.

If the device fetches or accesses external memory while EBDIS = 1, the pins will switch to external bus. If the EBDIS bit is set by a program executing from external memory, the action of setting the bit will be delayed until the program branches into the internal memory. At that time, the pins will change from external bus to I/O ports.

If the device is executing out of internal memory when EBDIS = 0, the memory bus address/data and control pins will not be active. They will go to a state where the active address/data pins are tri-state; the CE, OE, WRH, WRL, UB and LB signals are ‘1’ and ALE and BA0 are ‘0’. Note that only those pins associated with the current address width are forced to tri-state; the other pins continue to function as I/O. In the case of 16-bit address width, for example, only AD<15:0> (PORTD and PORTE) are affected; A19:A16 (PORTH<3:0>) continue to function as I/O.

In all external memory modes, the bus takes priority over any other peripherals that may share pins with it. This includes the Parallel Master Port and serial communication modules which would otherwise take priority over the I/O port.

7.6 16-Bit Data Width Modes

In 16-Bit Data Width mode, the external memory interface can be connected to external memories in three different configurations:

- 16-Bit Byte Write
- 16-Bit Word Write
- 16-Bit Byte Select

The configuration to be used is determined by the WM1:WM0 bits in the MEMCON register (MEMCON<1:0>). These three different configurations allow the designer maximum flexibility in using both 8-bit and 16-bit devices with 16-bit data.

For all 16-bit modes, the AddressLatch Enable (ALE) pin indicates that the address bits, AD<15:0>, are available on the external memory interface bus. Following the address latch, the Output Enable signal (OE) will enable both bytes of program memory at once to form a 16-bit instruction word. The Chip Enable signal (CE) is active at any time that the microcontroller accesses external memory, whether reading or writing; it is inactive (asserted high) whenever the device is in Sleep mode.

In Byte Select mode, JEDEC standard Flash memories will require BA0 for the byte address line and one I/O line to select between Byte and Word mode. The other 16-bit modes do not need BA0. JEDEC standard static RAM memories will use the UB or LB signals for byte selection.
7.6.1 16-BIT BYTE WRITE MODE

Figure 7-1 shows an example of 16-Bit Byte Write mode for PIC18F87J11 Family devices. This mode is used for two separate 8-bit memories connected for 16-bit operation. This generally includes basic EPROM and Flash devices. It allows table writes to byte-wide external memories.

During a TBLWT instruction cycle, the TABLAT data is presented on the upper and lower bytes of the AD15:AD0 bus. The appropriate WRH or WRL control line is strobed on the LSb of the TBLPTR.

**FIGURE 7-1: 16-BIT BYTE WRITE MODE EXAMPLE**

**Note 1:** Upper order address lines are used only for 20-bit address widths.
**2:** This signal only applies to table writes. See Section 6.1 “Table Reads and Table Writes”.

© 2008 Microchip Technology Inc. Preliminary DS39778C-page 101
7.6.2 16-BIT WORD WRITE MODE

Figure 7-2 shows an example of 16-Bit Word Write mode for PIC18F87J11 Family devices. This mode is used for word-wide memories which include some of the EPROM and Flash-type memories. This mode allows opcode fetches and table reads from all forms of 16-bit memory and table writes to any type of word-wide external memories. This method makes a distinction between TBLWT cycles to even or odd addresses.

During a TBLWT cycle to an even address (TBLPTR<0> = 0), the TABLAT data is transferred to a holding latch and the external address data bus is tri-stated for the data portion of the bus cycle. No write signals are activated.

During a TBLWT cycle to an odd address (TBLPTR<0> = 1), the TABLAT data is presented on the upper byte of the AD15:AD0 bus. The contents of the holding latch are presented on the lower byte of the AD15:AD0 bus.

The WRH signal is strobed for each write cycle; the WRL pin is unused. The signal on the BA0 pin indicates the LSb of the TBLPTR, but it is left unconnected. Instead, the UB and LB signals are active to select both bytes. The obvious limitation to this method is that the table write must be done in pairs on a specific word boundary to correctly write a word location.

**FIGURE 7-2: 16-BIT WORD WRITE MODE EXAMPLE**

```
Note 1: Upper order address lines are used only for 20-bit address widths.
2: This signal only applies to table writes. See Section 6.1 “Table Reads and Table Writes”.
```
7.6.3  16-BIT BYTE SELECT MODE

Figure 7-3 shows an example of 16-Bit Byte Select mode. This mode allows table write operations to word-wide external memories with byte selection capability. This generally includes both word-wide Flash and SRAM devices.

During a TBLWT cycle, the TABLAT data is presented on the upper and lower byte of the AD15:AD0 bus. The WRH signal is strobed for each write cycle; the WRL pin is not used. The BA0 or UB/LB signals are used to select the byte to be written, based on the Least Significant bit of the TBLPTR register.

Flash and SRAM devices use different control signal combinations to implement Byte Select mode. JEDEC standard Flash memories require that a controller I/O port pin be connected to the memory’s BYTE/WORD pin to provide the select signal. They also use the BA0 signal from the controller as a byte address. JEDEC standard static RAM memories, on the other hand, use the UB or LB signals to select the byte.

**FIGURE 7-3: 16-BIT BYTE SELECT MODE EXAMPLE**
7.6.4 16-BIT MODE TIMING

The presentation of control signals on the external memory bus is different for the various operating modes. Typical signal timing diagrams are shown in Figure 7-4 and Figure 7-5.

**FIGURE 7-4: EXTERNAL MEMORY BUS TIMING FOR TBLRD (EXTENDED MICROCONTROLLER MODE)**

<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>
</tr>
</thead>
<tbody>
<tr>
<td>A&lt;19:16&gt;</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>AD&lt;15:0&gt;</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>CE</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>ALE</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>OE</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>

Memory Cycle: Opcode Fetch Opcode Fetch Opcode Fetch TBLRD 92h Opcode Fetch
- TBLRD * from 000100h
- MOV LW 55h from 000102h
- TBLRD 92h from 199E67h
- Opcode Fetch from 000104h

Instruction Execution: INST(PC – 2) TBLRD Cycle 1 TBLRD Cycle 2 MOV LW

**FIGURE 7-5: EXTERNAL MEMORY BUS TIMING FOR SLEEP (EXTENDED MICROCONTROLLER MODE)**

<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>
</tr>
</thead>
<tbody>
<tr>
<td>A&lt;19:16&gt;</td>
<td>00h</td>
<td>00h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD&lt;15:0&gt;</td>
<td>AAAAh</td>
<td>0003h</td>
<td>3AABh</td>
<td>0E55h</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memory Cycle: Opcode Fetch Opcode Fetch Sleep Mode, Bus Inactive
- SLEEP from 007554h
- MOV LW 55h from 007556h

Instruction Execution: INST(PC – 2) SLEEP
7.7 8-Bit Data Width Mode

In 8-Bit Data Width mode, the external memory bus operates only in Multiplexed mode; that is, data shares the 8 Least Significant bits of the address bus.

Figure 7-6 shows an example of 8-Bit Multiplexed mode for 80-pin devices. This mode is used for a single 8-bit memory connected for 16-bit operation. The instructions will be fetched as two 8-bit bytes on a shared data/address bus. The two bytes are sequentially fetched within one instruction cycle (T_CY). Therefore, the designer must choose external memory devices according to timing calculations based on 1/2 T_CY (2 times the instruction rate). For proper memory speed selection, glue logic propagation delay times must be considered, along with setup and hold times.

The Address Latch Enable (ALE) pin indicates that the address bits, AD<15:0>, are available on the external memory interface bus. The Output Enable signal (OE) will enable one byte of program memory for a portion of the instruction cycle, then BA0 will change and the second byte will be enabled to form the 16-bit instruction word. The Least Significant bit of the address, BA0, must be connected to the memory devices in this mode. The Chip Enable signal (CE) is active at any time that the microcontroller accesses external memory, whether reading or writing. It is inactive (asserted high) whenever the device is in Sleep mode.

This generally includes basic EPROM and Flash devices. It allows table writes to byte-wide external memories.

During a TBLWT instruction cycle, the TABLAT data is presented on the upper and lower bytes of the AD15:AD0 bus. The appropriate level of the BA0 control line is strobed on the LSb of the TBLPTR.

FIGURE 7-6: 8-BIT MULTIPLEXED MODE EXAMPLE

![Diagram of 8-Bit Multiplexed Mode Example]

Note 1: Upper order address bits are only used for 20-bit address width. The upper AD byte is used for all address widths except 8-bit.

2: This signal only applies to table writes. See Section 6.1 “Table Reads and Table Writes”.

© 2008 Microchip Technology Inc.  Preliminary  DS39778C-page 105
7.7.1 8-BIT MODE TIMING

The presentation of control signals on the external memory bus is different for the various operating modes. Typical signal timing diagrams are shown in Figure 7-7 and Figure 7-8.

**FIGURE 7-7:** EXTERNAL MEMORY BUS TIMING FOR TBLRD (EXTENDED MICROCONTROLLER MODE)

<table>
<thead>
<tr>
<th>Memory Cycle</th>
<th>Instruction Execution</th>
<th>Opcode Fetch</th>
<th>Opcode Fetch</th>
<th>Opcode Fetch</th>
<th>Opcode Fetch</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&lt;19:16&gt;</td>
<td></td>
<td></td>
<td>TBLRD 92h</td>
<td></td>
<td>ADDLW 55h</td>
</tr>
<tr>
<td>AD&lt;15:8&gt;</td>
<td></td>
<td>0Ch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD&lt;7:0&gt;</td>
<td></td>
<td>CFh</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opcode Fetch</td>
<td></td>
<td></td>
<td>TBLRD 92h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opcode Fetch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opcode Fetch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opcode Fetch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 7-8:** EXTERNAL MEMORY BUS TIMING FOR SLEEP (EXTENDED MICROCONTROLLER MODE)

<table>
<thead>
<tr>
<th>Memory Cycle</th>
<th>Instruction Execution</th>
<th>Opcode Fetch</th>
<th>Opcode Fetch</th>
<th>Opcode Fetch</th>
<th>Opcode Fetch</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&lt;19:16&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AD&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BA0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opcode Fetch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opcode Fetch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opcode Fetch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opcode Fetch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Memory Cycle: SLEEP from 007554h

Instruction Execution: INST(PC – 2) SLEEP
7.8 Operation in Power-Managed Modes

In alternate, power-managed Run modes, the external bus continues to operate normally. If a clock source with a lower speed is selected, bus operations will run at that speed. In these cases, excessive access times for the external memory may result if wait states have been enabled and added to external memory operations. If operations in a lower power Run mode are anticipated, users should provide in their applications for adjusting memory access times at the lower clock speeds.

In Sleep and Idle modes, the microcontroller core does not need to access data; bus operations are suspended. The state of the external bus is frozen, with the address/data pins and most of the control pins holding at the same state they were in when the mode was invoked. The only potential changes are the CE, LB and UB pins, which are held at logic high.
8.0 8 x 8 HARDWARE MULTIPLIER

8.1 Introduction

All PIC18 devices include an 8 x 8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. The multiplier’s operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 8-1.

8.2 Operation

Example 8-1 shows the instruction sequence for an 8 x 8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 8-2 shows the sequence to do an 8 x 8 signed multiplication. To account for the sign bits of the arguments, each argument’s Most Significant bit (MSb) is tested and the appropriate subtractions are done.

### TABLE 8-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

<table>
<thead>
<tr>
<th>Routine</th>
<th>Multiply Method</th>
<th>Program Memory (Words)</th>
<th>Cycles (Max)</th>
<th>Time @ 48 MHz</th>
<th>Time @ 10 MHz</th>
<th>Time @ 4 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 x 8 unsigned</td>
<td>Without hardware multiply</td>
<td>13</td>
<td>69</td>
<td>5.7 μs</td>
<td>27.6 μs</td>
<td>69 μs</td>
</tr>
<tr>
<td>8 x 8 unsigned</td>
<td>Hardware multiply</td>
<td>1</td>
<td>1</td>
<td>83.3 ns</td>
<td>400 ns</td>
<td>1 μs</td>
</tr>
<tr>
<td>8 x 8 signed</td>
<td>Without hardware multiply</td>
<td>33</td>
<td>91</td>
<td>7.5 μs</td>
<td>36.4 μs</td>
<td>91 μs</td>
</tr>
<tr>
<td>8 x 8 signed</td>
<td>Hardware multiply</td>
<td>6</td>
<td>6</td>
<td>500 ns</td>
<td>2.4 μs</td>
<td>6 μs</td>
</tr>
<tr>
<td>16 x 16 unsigned</td>
<td>Without hardware multiply</td>
<td>21</td>
<td>242</td>
<td>20.1 μs</td>
<td>96.8 μs</td>
<td>242 μs</td>
</tr>
<tr>
<td>16 x 16 unsigned</td>
<td>Hardware multiply</td>
<td>28</td>
<td>28</td>
<td>2.3 μs</td>
<td>11.2 μs</td>
<td>28 μs</td>
</tr>
<tr>
<td>16 x 16 signed</td>
<td>Without hardware multiply</td>
<td>52</td>
<td>254</td>
<td>21.6 μs</td>
<td>102.6 μs</td>
<td>254 μs</td>
</tr>
<tr>
<td>16 x 16 signed</td>
<td>Hardware multiply</td>
<td>35</td>
<td>40</td>
<td>3.3 μs</td>
<td>16.0 μs</td>
<td>40 μs</td>
</tr>
</tbody>
</table>

**EXAMPLE 8-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE**

MOVF ARG1, W ;
MULWF ARG2 ; ARG1 * ARG2 ->
            ; PRODH:PRODL

**EXAMPLE 8-2: 8 x 8 SIGNED MULTIPLY ROUTINE**

MOVF ARG1, W
MULWF ARG2 ; ARG1 * ARG2 ->
            ; PRODH:PRODL
BTFSC ARG2, SB ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH
            ; - ARG1
MOVF ARG2, W
BTFSC ARG1, SB ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH
            ; - ARG2
Example 8-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 8-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES3:RES0).

**EQUATION 8-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM**

\[
\text{RES3:RES0} = \text{ARG1H:ARG1L} \cdot \text{ARG2H:ARG2L} = \text{ARG1H} \cdot \text{ARG2H} \cdot 2^{16} + \text{ARG1H} \cdot \text{ARG2L} \cdot 2^8 + \text{ARG1L} \cdot \text{ARG2H} \cdot 2^8 + \text{ARG1L} \cdot \text{ARG2L}
\]

**EXAMPLE 8-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE**

```
MOVF ARG1L, W
MULWF ARG2L ; ARG1L * ARG2L-> PRODH:PRODL
MOVFF PRODH, RES1 ;
MOVFF PRODL, RES0 ;
;
MOVF ARG1H, W
MULWF ARG2H ; ARG1H * ARG2H-> PRODH:PRODL
MOVFF PRODH, RES3 ;
MOVFF PRODL, RES2 ;
;
MOVF ARG1L, W
MULWF ARG2H ; ARG1L * ARG2H-> PRODH:PRODL
MOVFF PRODH, RES1 ;
MOVFF PRODL, RES0 ;
;
MOVF ARG1H, W
MULWF ARG2L ; ARG1H * ARG2L-> PRODH:PRODL
MOVFF PRODH, RES3 ;
MOVFF PRODL, RES2 ;
;
BTFSS ARG2H, 7 ; ARG2H:ARG2L neg?
BRA SIGN_ARG1 ; no, check ARG1
MOVF ARG2L, W ;
ADDWF RES1, F ; Add cross
MOVF PRODH, W ; products
ADDWFC RES2, F ;
CLRF WREG ;
ADDWFC RES3, F ;
;
MOVF ARG1H, W ;
MULWF ARG2L ; ARG1H * ARG2L-> PRODH:PRODL
MOVFF PRODH, RES3 ;
MOVFF PRODL, RES2 ;
;
BTFSS ARG2H, 7 ; ARG2H:ARG2L neg?
BRA SIGN_ARG1 ; no, check ARG1
MOVF ARG1L, W ;
SUBWF RES3 ;
MOVF ARG1H, W ;
SUBWFB RES3 ;
;
SIGN_ARG1
BTFSS ARG1H, 7 ; ARG1H:ARG1L neg?
BRA CONT_CODE ; no, done
MOVF ARG2L, W ;
SUBWF RES2 ;
MOVF ARG2H, W ;
SUBWFB RES3 ;
;
CONT_CODE ;
```

Example 8-4 shows the sequence to do a 16 x 16 signed multiply. Equation 8-2 shows the algorithm used. The 32-bit result is stored in four registers (RES3:RES0). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

**EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM**

\[
\text{RES3:RES0} = \text{ARG1H:ARG1L} \cdot \text{ARG2H:ARG2L} = (\text{ARG1H} \cdot \text{ARG2H} \cdot 2^{16}) + (\text{ARG1H} \cdot \text{ARG2L} \cdot 2^8) + (\text{ARG1L} \cdot \text{ARG2H} \cdot 2^8) + (\text{ARG1L} \cdot \text{ARG2L}) + (-1 \cdot \text{ARG2H}<7>) \cdot \text{ARG1H:ARG1L} \cdot 2^{16}) + (-1 \cdot \text{ARG1H}<7>) \cdot \text{ARG2H:ARG2L} \cdot 2^{16})
\]

**EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE**

```
MOVF ARG1L, W
MULWF ARG2L ; ARG1L * ARG2L-> PRODH:PRODL
MOVFF PRODH, RES1 ;
MOVFF PRODL, RES0 ;
;
MOVF ARG1H, W
MULWF ARG2H ; ARG1H * ARG2H-> PRODH:PRODL
MOVFF PRODH, RES3 ;
MOVFF PRODL, RES2 ;
;
MOVF ARG1L, W
MULWF ARG2H ; ARG1L * ARG2H-> PRODH:PRODL
MOVFF PRODH, RES1 ;
MOVFF PRODL, RES0 ;
;
MOVF ARG1H, W
MULWF ARG2L ; ARG1H * ARG2L-> PRODH:PRODL
MOVFF PRODH, RES3 ;
MOVFF PRODL, RES2 ;
;
BTFSS ARG2H, 7 ; ARG2H:ARG2L neg?
BRA SIGN_ARG1 ; no, check ARG1
MOVF ARG1L, W ;
SUBWF RES3 ;
MOVF ARG1H, W ;
SUBWFB RES3 ;
;
SIGN_ARG1
BTFSS ARG1H, 7 ; ARG1H:ARG1L neg?
BRA CONT_CODE ; no, done
MOVF ARG2L, W ;
SUBWF RES2 ;
MOVF ARG2H, W ;
SUBWFB RES3 ;
;
CONT_CODE ;
```
9.0 INTERRUPTS

Members of the PIC18F87J11 Family of devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high-priority level or a low-priority level. The high-priority interrupt vector is at 0008h and the low-priority interrupt vector is at 0018h. High-priority interrupt events will interrupt any low-priority interrupts that may be in progress.

There are thirteen registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

In general, interrupt sources have three bits to control their operation. They are:

- **Flag bit** to indicate that an interrupt event occurred
- **Enable bit** that allows program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** to select high-priority or low-priority

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 0008h or 0018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC16 mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit which enables/disables all interrupt sources. All interrupts branch to address 0008h in Compatibility mode.

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High-priority interrupt sources can interrupt a low-priority interrupt. Low-priority interrupts are not processed while high-priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The "return from interrupt" instruction, RETFI, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used) which re-enables interrupts.

For external interrupt events, such as the INTx pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set regardless of the status of their corresponding enable bit or the GIE bit.

**Note:** Do not use the MOVFF instruction to modify any of the interrupt control registers while any interrupt is enabled. Doing so may cause erratic microcontroller behavior.
FIGURE 9-1: PIC18F87J11 FAMILY INTERRUPT LOGIC

High-Priority Interrupt Generation

Low-Priority Interrupt Generation

Wake-up if in Idle or Sleep modes

Interrupt to CPU Vector to Location 0008h

Interrupt to CPU Vector to Location 0018h

IPEN

PIR1<7:0>
PIE1<7:0>
IPR1<7:0>

PIR2<7:5, 3:0>
PIE2<7:5, 3:0>
IPR2<7:5, 3:0>

PIR3<7, 0>
PIE3<7, 0>
IPR3<7, 0>

INT0IF
INT0IE
INT0IP

INT1IF
INT1IE
INT1IP

INT2IF
INT2IE
INT2IP

INT3IF
INT3IE
INT3IP

TMR0IF
TMR0IE
TMR0IP

RBIF
RBIE
RBIP

INT1F
INT1E
INT1P

INT2F
INT2E
INT2P

INT3F
INT3E
INT3P

IPEN

GIE/GIEH

PEIE/GIEL

IPEN

IPEN
### 9.1 INTCON Registers

The INTCON registers are readable and writable registers which contain various enable, priority and flag bits.

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

#### REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>Value at POR</th>
<th>Value after Reset</th>
<th>Change</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>GIE/GIEH</td>
<td>Global Interrupt Enable bit</td>
<td>1</td>
<td>0</td>
<td>R = Readable</td>
<td><em>GIE/GIEH</em> serves as a global enable bit. When IPEN = 0:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W = Writable</td>
<td>1 = Enables all unmasked interrupts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U = Unimplemented bit, read as ‘0’</td>
<td>0 = Disables all interrupts</td>
</tr>
<tr>
<td>6</td>
<td>PEIE/GIEL</td>
<td>Peripheral Interrupt Enable bit</td>
<td>1</td>
<td>0</td>
<td>R = Readable</td>
<td><em>PEIE/GIEL</em> serves as a low-priority peripheral enable bit. When IPEN = 0:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W = Writable</td>
<td>1 = Enables all low-priority peripheral interrupts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U = Unimplemented bit, read as ‘0’</td>
<td>0 = Disables all low-priority peripheral interrupts</td>
</tr>
<tr>
<td>5</td>
<td>TMR0IE</td>
<td>TMR0 Overflow Interrupt Enable bit</td>
<td>1</td>
<td>0</td>
<td>R = Readable</td>
<td>Enables the TMR0 overflow interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W = Writable</td>
<td>Disables the TMR0 overflow interrupt</td>
</tr>
<tr>
<td>4</td>
<td>INT0IE</td>
<td>INT0 External Interrupt Enable bit</td>
<td>1</td>
<td>0</td>
<td>R = Readable</td>
<td>Enables the INT0 external interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W = Writable</td>
<td>Disables the INT0 external interrupt</td>
</tr>
<tr>
<td>3</td>
<td>RBIE</td>
<td>RB Port Change Interrupt Enable bit</td>
<td>1</td>
<td>0</td>
<td>R = Readable</td>
<td>Enables the RB port change interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W = Writable</td>
<td>Disables the RB port change interrupt</td>
</tr>
<tr>
<td>2</td>
<td>TMR0IF</td>
<td>TMR0 Overflow Interrupt Flag bit</td>
<td>1</td>
<td>0</td>
<td>R = Readable</td>
<td>TMR0 register has overflowed (must be cleared in software)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W = Writable</td>
<td>TMR0 register did not overflow</td>
</tr>
<tr>
<td>1</td>
<td>INT0IF</td>
<td>INT0 External Interrupt Flag bit</td>
<td>1</td>
<td>0</td>
<td>R = Readable</td>
<td>The INT0 external interrupt occurred (must be cleared in software)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W = Writable</td>
<td>The INT0 external interrupt did not occur</td>
</tr>
<tr>
<td>0</td>
<td>RBIF</td>
<td>RB Port Change Interrupt Flag bit</td>
<td>1</td>
<td>0</td>
<td>R = Readable</td>
<td>At least one of the RB7:RB4 pins changed state (must be cleared in software)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W = Writable</td>
<td>None of the RB7:RB4 pins have changed state</td>
</tr>
</tbody>
</table>

**Note 1:** A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.
### REGISTER 9-2: INTCON2: INTERRUPT CONTROL REGISTER 2

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBPU</td>
<td>INTEDG0</td>
<td>INTEDG1</td>
<td>INTEDG2</td>
<td>INTEDG3</td>
<td>TMR0IP</td>
<td>INT3IP</td>
<td>RBIP</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>
<td>bit 0</td>
</tr>
</tbody>
</table>

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

- **bit 7**: **RBPU**: PORTB Pull-up Enable bit
  - 1 = All PORTB pull-ups are disabled
  - 0 = PORTB pull-ups are enabled by individual port latch values

- **bit 6**: **INTEDG0**: External Interrupt 0 Edge Select bit
  - 1 = Interrupt on rising edge
  - 0 = Interrupt on falling edge

- **bit 5**: **INTEDG1**: External Interrupt 1 Edge Select bit
  - 1 = Interrupt on rising edge
  - 0 = Interrupt on falling edge

- **bit 4**: **INTEDG2**: External Interrupt 2 Edge Select bit
  - 1 = Interrupt on rising edge
  - 0 = Interrupt on falling edge

- **bit 3**: **INTEDG3**: External Interrupt 3 Edge Select bit
  - 1 = Interrupt on rising edge
  - 0 = Interrupt on falling edge

- **bit 2**: **TMR0IP**: TMR0 Overflow Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority

- **bit 1**: **INT3IP**: INT3 External Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority

- **bit 0**: **RBIP**: RB Port Change Interrupt Priority bit
  - 1 = High priority
  - 0 = Low priority

#### Note:
Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global interrupt enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.
REGISTER 9-3: INTCON3: INTERRUPT CONTROL REGISTER 3

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT2IP</td>
<td>INT1IP</td>
<td>INT3IE</td>
<td>INT2IE</td>
<td>INT1IE</td>
<td>INT3IF</td>
<td>INT2IF</td>
<td>INT1IF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 7</th>
<th>INT2IP: INT2 External Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6</th>
<th>INT1IP: INT1 External Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5</th>
<th>INT3IE: INT3 External Interrupt Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enables the INT3 external interrupt</td>
</tr>
<tr>
<td>0</td>
<td>Disables the INT3 external interrupt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 4</th>
<th>INT2IE: INT2 External Interrupt Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enables the INT2 external interrupt</td>
</tr>
<tr>
<td>0</td>
<td>Disables the INT2 external interrupt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 3</th>
<th>INT1IE: INT1 External Interrupt Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enables the INT1 external interrupt</td>
</tr>
<tr>
<td>0</td>
<td>Disables the INT1 external interrupt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 2</th>
<th>INT3IF: INT3 External Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The INT3 external interrupt occurred (must be cleared in software)</td>
</tr>
<tr>
<td>0</td>
<td>The INT3 external interrupt did not occur</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1</th>
<th>INT2IF: INT2 External Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The INT2 external interrupt occurred (must be cleared in software)</td>
</tr>
<tr>
<td>0</td>
<td>The INT2 external interrupt did not occur</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 0</th>
<th>INT1IF: INT1 External Interrupt Flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The INT1 external interrupt occurred (must be cleared in software)</td>
</tr>
<tr>
<td>0</td>
<td>The INT1 external interrupt did not occur</td>
</tr>
</tbody>
</table>

**Legend:**

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

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

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Request (Flag) registers (PIR1, PIR2, PIR3).

Note 1: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE (INTCON<7>).

2: User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
</tr>
</tbody>
</table>

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

bit 7 PMPIF: Parallel Master Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 ADIF: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 RC1IF: EUSART1 Receive Interrupt Flag bit
1 = The EUSART1 receive buffer, RCReg1, is full (cleared when RCReg1 is read)
0 = The EUSART1 receive buffer is empty

bit 4 TX1IF: EUSART1 Transmit Interrupt Flag bit
1 = The EUSART1 transmit buffer, TXReg1, is empty (cleared when TXReg1 is written)
0 = The EUSART1 transmit buffer is full

bit 3 SSP1IF: MSSP1 Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

bit 2 CCP1IF: ECCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1/TMR3 register capture occurred (must be cleared in software)
0 = No TMR1/TMR3 register capture occurred

Compare mode:
1 = A TMR1/TMR3 register compare match occurred (must be cleared in software)
0 = No TMR1/TMR3 register compare match occurred

PWM mode:
Unused in this mode.

bit 1 TMR2IF: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 TMR1IF: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow
## REGISTER 9-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) 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</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSCF1F</td>
<td>CM2IF</td>
<td>CM1IF</td>
<td>—</td>
<td>BCL1IF</td>
<td>LVDIF</td>
<td>TMR3IF</td>
<td>CCP2IF</td>
</tr>
</tbody>
</table>

**Legend:**

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

- **bit 7**
  - **OSCF1F**: Oscillator Fail Interrupt Flag bit
    - 1 = Device oscillator failed, clock input has changed to INTOSC (must be cleared in software)
    - 0 = Device clock operating

- **bit 6**
  - **CM2IF**: Comparator 2 Interrupt Flag bit
    - 1 = Comparator input has changed (must be cleared in software)
    - 0 = Comparator input has not changed

- **bit 5**
  - **CM1IF**: Comparator 1 Interrupt Flag bit
    - 1 = Comparator input has changed (must be cleared in software)
    - 0 = Comparator input has not changed

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

- **bit 3**
  - **BCL1IF**: Bus Collision Interrupt Flag bit (MSSP1 module)
    - 1 = A bus collision occurred (must be cleared in software)
    - 0 = No bus collision occurred

- **bit 2**
  - **LVDIF**: Low-Voltage Detect Interrupt Flag bit
    - 1 = A low-voltage condition occurred (must be cleared in software)
    - 0 = VDDCORE has not fallen below the low-voltage trip point (about 2.45V)

- **bit 1**
  - **TMR3IF**: TMR3 Overflow Interrupt Flag bit
    - 1 = TMR3 register overflowed (must be cleared in software)
    - 0 = TMR3 register did not overflow

- **bit 0**
  - **CCP2IF**: ECCP2 Interrupt Flag bit
    - **Capture mode**:
      - 1 = A TMR1/TMR3 register capture occurred (must be cleared in software)
      - 0 = No TMR1/TMR3 register capture occurred
    - **Compare mode**:
      - 1 = A TMR1/TMR3 register compare match occurred (must be cleared in software)
      - 0 = No TMR1/TMR3 register compare match occurred
    - **PWM mode**:
      - Unused in this mode.
### REGISTER 9-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3

<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>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
</tr>
</tbody>
</table>

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

- **bit 7 SSP2IF**: MSSP2 Interrupt Flag bit
  - **1** = The transmission/reception is complete (must be cleared in software)
  - **0** = Waiting to transmit/receive
- **bit 6 BCL2IF**: Bus Collision Interrupt Flag bit (MSSP2 module)
  - **1** = A bus collision occurred (must be cleared in software)
  - **0** = No bus collision occurred
- **bit 5 RC2IF**: EUSART2 Receive Interrupt Flag bit
  - **1** = The EUSART2 receive buffer, RCREG2, is full (cleared when RCREG2 is read)
  - **0** = The EUSART2 receive buffer is empty
- **bit 4 TX2IF**: EUSART2 Transmit Interrupt Flag bit
  - **1** = The EUSART2 transmit buffer, TXREG2, is empty (cleared when TXREG2 is written)
  - **0** = The EUSART2 transmit buffer is full
- **bit 3 TMR4IF**: TMR4 to PR4 Match Interrupt Flag bit
  - **1** = TMR4 to PR4 match occurred (must be cleared in software)
  - **0** = No TMR4 to PR4 match occurred
- **bit 2 CCP5IF**: CCP5 Interrupt Flag bit
  - **Capture mode**:
    - **1** = A TMR1/TMR3 register capture occurred (must be cleared in software)
    - **0** = No TMR1/TMR3 register capture occurred
  - **Compare mode**:
    - **1** = A TMR1/TMR3 register compare match occurred (must be cleared in software)
    - **0** = No TMR1/TMR3 register compare match occurred
  - **PWM mode**:
    - Unused in this mode.
- **bit 1 CCP4IF**: CCP4 Interrupt Flag bit
  - **Capture mode**:
    - **1** = A TMR1/TMR3 register capture occurred (must be cleared in software)
    - **0** = No TMR1/TMR3 register capture occurred
  - **Compare mode**:
    - **1** = A TMR1/TMR3 register compare match occurred (must be cleared in software)
    - **0** = No TMR1/TMR3 register compare match occurred
  - **PWM mode**:
    - Unused in this mode.
- **bit 0 CCP3IF**: ECCP3 Interrupt Flag bit
  - **Capture mode**:
    - **1** = A TMR1/TMR3 register capture occurred (must be cleared in software)
    - **0** = No TMR1/TMR3 register capture occurred
  - **Compare mode**:
    - **1** = A TMR1/TMR3 register compare match occurred (must be cleared in software)
    - **0** = No TMR1/TMR3 register compare match occurred
  - **PWM mode**:
    - Unused in this mode.
9.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Enable registers (PIE1, PIE2, PIE3). When \( IPEN = 0 \), the PEIE bit must be set to enable any of these peripheral interrupts.

REGISTER 9-7: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PMPIE: Parallel Master Port Read/Write Interrupt Enable bit</td>
<td>1</td>
<td>Enables the PM read/write interrupt</td>
</tr>
<tr>
<td>6</td>
<td>ADIE: A/D Converter Interrupt Enable bit</td>
<td>0</td>
<td>Disables the A/D interrupt</td>
</tr>
<tr>
<td>5</td>
<td>RC1IE: EUSART1 Receive Interrupt Enable bit</td>
<td>0</td>
<td>Disables the EUSART1 receive interrupt</td>
</tr>
<tr>
<td>4</td>
<td>TX1IE: EUSART1 Transmit Interrupt Enable bit</td>
<td>0</td>
<td>Disables the EUSART1 transmit interrupt</td>
</tr>
<tr>
<td>3</td>
<td>SSP1IE: MSSP1 Interrupt Enable bit</td>
<td>0</td>
<td>Disables the MSSP1 interrupt</td>
</tr>
<tr>
<td>2</td>
<td>CCP1IE: ECCP1 Interrupt Enable bit</td>
<td>0</td>
<td>Disables the ECCP1 interrupt</td>
</tr>
<tr>
<td>1</td>
<td>TMR2IE: TMR2 to PR2 Match Interrupt Enable bit</td>
<td>0</td>
<td>Disables the TMR2 to PR2 match interrupt</td>
</tr>
<tr>
<td>0</td>
<td>TMR1IE: TMR1 Overflow Interrupt Enable bit</td>
<td>0</td>
<td>Disables the TMR1 overflow interrupt</td>
</tr>
</tbody>
</table>

Legend:
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown
### REGISTER 9-8: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSCFIE</td>
<td>CM2IE</td>
<td>CM1IE</td>
<td>—</td>
<td>BCL1IE</td>
<td>LVDIE</td>
<td>TMR3IE</td>
<td>CCP2IE</td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

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

**bit 7**  
**OSCFIE**: Oscillator Fail Interrupt Enable bit  
1 = Enabled  
0 = Disabled

**bit 6**  
**CM2IE**: Comparator 2 Interrupt Enable bit  
1 = Enabled  
0 = Disabled

**bit 5**  
**CM1IE**: Comparator 1 Interrupt Enable bit  
1 = Enabled  
0 = Disabled

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

**bit 3**  
**BCL1IE**: Bus Collision Interrupt Enable bit (MSSP1 module)  
1 = Enabled  
0 = Disabled

**bit 2**  
**LVDIE**: Low-Voltage Detect Interrupt Enable bit  
1 = Enabled  
0 = Disabled

**bit 1**  
**TMR3IE**: TMR3 Overflow Interrupt Enable bit  
1 = Enabled  
0 = Disabled

**bit 0**  
**CCP2IE**: ECCP2 Interrupt Enable bit  
1 = Enabled  
0 = Disabled
**REGISTER 9-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR</th>
<th>Enabled</th>
<th>Disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>SSP2IE: MSSP2 Interrupt Enable bit</td>
<td>1</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>6</td>
<td>BCL2IE: Bus Collision Interrupt Enable bit (MSSP2 module)</td>
<td>1</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>5</td>
<td>RC2IE: EUSART2 Receive Interrupt Enable bit</td>
<td>1</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>4</td>
<td>TX2IE: EUSART2 Transmit Interrupt Enable bit</td>
<td>1</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>3</td>
<td>TMR4IE: TMR4 to PR4 Match Interrupt Enable bit</td>
<td>1</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>2</td>
<td>CCP5IE: CCP5 Interrupt Enable bit</td>
<td>1</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td>CCP4IE: CCP4 Interrupt Enable bit</td>
<td>1</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>0</td>
<td>CCP3IE: ECCP3 Interrupt Enable bit</td>
<td>1</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
</tbody>
</table>
9.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Priority registers (IPR1, IPR2, IPR3). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

REGISTER 9-10:  IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
</tr>
</tbody>
</table>

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

bit 7  PMPIP: Parallel Master Port Read/Write Interrupt Priority bit
       1 = High priority
       0 = Low priority

bit 6  ADIP: A/D Converter Interrupt Priority bit
       1 = High priority
       0 = Low priority

bit 5  RC1IP: EUSART1 Receive Interrupt Priority bit
       1 = High priority
       0 = Low priority

bit 4  TX1IP: EUSART1 Transmit Interrupt Priority bit
       1 = High priority
       0 = Low priority

bit 3  SSP1IP: MSSP1 Interrupt Priority bit
       1 = High priority
       0 = Low priority

bit 2  CCP1IP: ECCP1 Interrupt Priority bit
       1 = High priority
       0 = Low priority

bit 1  TMR2IP: TMR2 to PR2 Match Interrupt Priority bit
       1 = High priority
       0 = Low priority

bit 0  TMR1IP: TMR1 Overflow Interrupt Priority bit
       1 = High priority
       0 = Low priority
<table>
<thead>
<tr>
<th>bit 7</th>
<th>OSCFIP: Oscillator Fail Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6</th>
<th>CM2IP: Comparator 2 Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5</th>
<th>C12IP: Comparator 1 Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 4</th>
<th>Unimplemented: Read as ‘0’</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 3</th>
<th>BCL1IP: Bus Collision Interrupt Priority bit (MSSP1 module)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 2</th>
<th>LVDIP: Low-Voltage Detect Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1</th>
<th>TMR3IP: TMR3 Overflow Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 0</th>
<th>CCP2IP: ECCP2 Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>
### REGISTER 9-12: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
</tr>
</tbody>
</table>

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

**Legend:**

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

**Legend:**

<table>
<thead>
<tr>
<th>bit 7</th>
<th>SSP2IP: MSSP2 Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

**Legend:**

<table>
<thead>
<tr>
<th>bit 6</th>
<th>BCL2IP: Bus Collision Interrupt Priority bit (MSSP2 module)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

**Legend:**

<table>
<thead>
<tr>
<th>bit 5</th>
<th>RC2IP: EUSART2 Receive Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

**Legend:**

<table>
<thead>
<tr>
<th>bit 4</th>
<th>TX2IP: EUSART2 Transmit Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

**Legend:**

<table>
<thead>
<tr>
<th>bit 3</th>
<th>TMR4IE: TMR4 to PR4 Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

**Legend:**

<table>
<thead>
<tr>
<th>bit 2</th>
<th>CCP5IP: CCP5 Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

**Legend:**

<table>
<thead>
<tr>
<th>bit 1</th>
<th>CCP4IP: CCP4 Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>

**Legend:**

<table>
<thead>
<tr>
<th>bit 0</th>
<th>CCP3IP: ECCP3 Interrupt Priority bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High priority</td>
</tr>
<tr>
<td>0</td>
<td>Low priority</td>
</tr>
</tbody>
</table>
9.5 RCON Register

The RCON register contains bits used to determine the cause of the last Reset or wake-up from Idle or Sleep modes. RCON also contains the bit that enables interrupt priorities (IPEN).

REGISTER 9-13: RCON: RESET CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R-1</th>
<th>R-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPEN</td>
<td>—</td>
<td>CM</td>
<td>Ri</td>
<td>TO</td>
<td>PD</td>
<td>POR</td>
<td>BOR</td>
</tr>
</tbody>
</table>

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

bit 7  **IPEN**: Interrupt Priority Enable bit
   1 = Enable priority levels on interrupts
   0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)
bit 6  **Unimplemented**: Read as ‘0’
bit 5  **CM**: Configuration Mismatch Flag bit
   For details of bit operation, see Register 4-1.
bit 4  **RI**: RESET Instruction Flag bit
   For details of bit operation, see Register 4-1.
bit 3  **TO**: Watchdog Timer Time-out Flag bit
   For details of bit operation, see Register 4-1.
bit 2  **PD**: Power-Down Detection Flag bit
   For details of bit operation, see Register 4-1.
bit 1  **POR**: Power-on Reset Status bit
   For details of bit operation, see Register 4-1.
bit 0  **BOR**: Brown-out Reset Status bit
   For details of bit operation, see Register 4-1.
9.6 INTx Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1, RB2/INT2 and RB3/INT3 pins are edge-triggered. If the corresponding INTEDGx bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. When a valid edge appears on the RBx/INTx pin, the corresponding flag bit, INTxIF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxIE. Flag bit, INTxIF, must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1, INT2 and INT3) can wake-up the processor from the power-managed modes if bit INTxIE was set prior to going into the power-managed modes. If the Global Interrupt Enable bit, GIE, is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority for INT1, INT2 and INT3 is determined by the value contained in the interrupt priority bits, INT1IP (INTCON3<6>), INT2IP (INTCON3<7>) and INT3IP (INTCON2<1>). There is no priority bit associated with INT0. It is always a high-priority interrupt source.

9.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit, TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set TMROIF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMROIE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMROIP (INTCON2<2>). See Section 12.0 “Timer0 Module” for further details on the Timer0 module.

9.8 PORTB Interrupt-on-Change

An input change on PORTB<7:4> sets flag bit, RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE (INTCON<3>). Interrupt priority for PORTB interrupt-on-change is determined by the value contained in the interrupt priority bit, RBIP (INTCON2<0>).

9.9 Context Saving During Interrupts

During interrupts, the return PC address is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the Fast Return Stack. If a fast return from interrupt is not used (see Section 5.3 “Data Memory Organization”), the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user’s application, other registers may also need to be saved. Example 9-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

EXAMPLE 9-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

```asm
MOVF W_TEMP ; W_TEMP is in virtual bank
MOVF STATUS, STATUS_TEMP ; STATUS_TEMP located anywhere
MOVF BSR, BSR_TEMP ; BSR_TEMP located anywhere
;
; USER ISR CODE
;
MOVF BSR_TEMP, BSR ; Restore BSR
MOVF W_TEMP, W ; Restore WREG
MOVFF STATUS_TEMP, STATUS ; Restore STATUS
```
10.0 I/O PORTS

Depending on the device selected and features enabled, there are up to nine ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Each port has three memory-mapped registers for its operation:
- TRIS register (Data Direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (Output Latch register)

Reading the PORT register reads the current status of the pins, whereas writing to the PORT register writes to the Output Latch (LAT) register.

Setting a TRIS bit (\( = 1 \)) makes the corresponding PORT pin an input (i.e., puts the corresponding output driver in a high-impedance mode). Clearing a TRIS bit (\( = 0 \)) makes the corresponding PORT pin an output (i.e., puts the contents of the corresponding LAT bit on the selected pin).

The Output Latch (LAT register) is useful for read-modify-write operations on the value that the I/O pins are driving. Read-modify-write operations on the LAT register read and write the latched output value for the PORT register.

A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in Figure 10-1.

10.1 I/O Port Pin Capabilities

When developing an application, the capabilities of the port pins must be considered. Outputs on some pins have higher output drive strength than others. Similarly, some pins can tolerate higher than \( V_{DD} \) input levels.

10.1.1 INPUT PINS AND VOLTAGE CONSIDERATIONS

The voltage tolerance of pins used as device inputs is dependent on the pin's input function. Pins that are used as digital only inputs are able to handle DC voltages up to 5.5V, a level typical for digital logic circuits. In contrast, pins that also have analog input functions of any kind (such as A/D and comparator inputs) can only tolerate voltages up to \( V_{DD} \). Voltage excursions beyond \( V_{DD} \) on these pins should be avoided.

Table 10-1 summarizes the input capabilities. Refer to Section 27.0 “Electrical Characteristics” for more details.

<table>
<thead>
<tr>
<th>Port or Pin</th>
<th>Tolerated Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA&lt;7:0&gt;</td>
<td>( V_{DD} )</td>
<td>Only ( V_{DD} ) input levels tolerated.</td>
</tr>
<tr>
<td>PORTC&lt;1:0&gt;</td>
<td>5.5V</td>
<td>Tolerates input levels above ( V_{DD} ), useful for most standard logic.</td>
</tr>
<tr>
<td>PORTF&lt;6:1&gt;</td>
<td>( V_{DD} )</td>
<td>Tolerates input levels above ( V_{DD} ), useful for most standard logic.</td>
</tr>
<tr>
<td>PORTH&lt;7:4&gt;</td>
<td>( V_{DD} )</td>
<td>Tolerates input levels above ( V_{DD} ), useful for most standard logic.</td>
</tr>
<tr>
<td>PORTB&lt;7:0&gt;</td>
<td>( V_{DD} )</td>
<td>Only ( V_{DD} ) input levels tolerated.</td>
</tr>
<tr>
<td>PORTC&lt;7:2&gt;</td>
<td>( V_{DD} )</td>
<td>Only ( V_{DD} ) input levels tolerated.</td>
</tr>
<tr>
<td>PORTD&lt;7:0&gt;</td>
<td>( V_{DD} )</td>
<td>Only ( V_{DD} ) input levels tolerated.</td>
</tr>
<tr>
<td>PORTE&lt;7:0&gt;</td>
<td>( V_{DD} )</td>
<td>Only ( V_{DD} ) input levels tolerated.</td>
</tr>
<tr>
<td>PORTF&lt;7&gt;</td>
<td>( V_{DD} )</td>
<td>Only ( V_{DD} ) input levels tolerated.</td>
</tr>
<tr>
<td>PORTG&lt;4:0&gt;</td>
<td>( V_{DD} )</td>
<td>Only ( V_{DD} ) input levels tolerated.</td>
</tr>
<tr>
<td>PORTH&lt;3:0&gt;</td>
<td>( V_{DD} )</td>
<td>Only ( V_{DD} ) input levels tolerated.</td>
</tr>
<tr>
<td>PORTJ&lt;7:0&gt;</td>
<td>( V_{DD} )</td>
<td>Only ( V_{DD} ) input levels tolerated.</td>
</tr>
</tbody>
</table>

Note 1: These ports are not available on 64-pin devices.

10.1.2 PIN OUTPUT DRIVE

When used as digital I/O, the output pin drive strengths vary for groups of pins intended to meet the needs for a variety of applications. In general, there are three classes of output pins in terms of drive capability.

PORTB and PORTC, as well as PORTA<7:6>, are designed to drive higher current loads, such as LEDs. PORTD, PORTE and PORTJ are capable of driving digital circuits associated with external memory devices; they can also drive LEDs, but only those with smaller current requirements. PORTF, PORTG and PORTH, along with PORTA<5:0>, have the lowest drive level, but are capable of driving normal digital circuit loads with a high input impedance.
Table 10-2 summarizes the output capabilities of the ports. Refer to the “Absolute Maximum Ratings” in Section 27.0 “Electrical Characteristics” for more details.

### TABLE 10-2: OUTPUT DRIVE LEVELS

<table>
<thead>
<tr>
<th>Port</th>
<th>Drive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA</td>
<td>Minimum</td>
<td>Intended for indication.</td>
</tr>
<tr>
<td>PORTF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORTG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORTH(^{(1)})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORTD</td>
<td>Medium</td>
<td>Sufficient drive levels for external memory interfacing as well as indication.</td>
</tr>
<tr>
<td>PORTE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORTJ(^{(1)})</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORTB</td>
<td>High</td>
<td>Suitable for direct LED drive levels.</td>
</tr>
<tr>
<td>PORTC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** These ports are not available on 64-pin devices.

10.1.3 PULL-UP CONFIGURATION

Four of the I/O ports (PORTB, PORTD, PORTE and PORTJ) implement configurable weak pull-ups on all pins. These are internal pull-ups that allow floating digital input signals to be pulled to a consistent level, without the use of external resistors.

The pull-ups are enabled with a single bit for each of the ports: RBPU (INTCON2<7>) for PORTB, and RDPU, RPJU and RJPU (PORTG<7:5>) for the other ports.

10.1.4 OPEN-DRAIN OUTPUTS

The output pins for several peripherals are also equipped with a configurable, open-drain output option. This allows the peripherals to communicate with external digital logic operating at a higher voltage level, without the use of level translators.

The open-drain option is implemented on port pins specifically associated with the data and clock outputs of the EUSARTs, the MSSP modules (in SPI mode) and the CCP and ECCP modules. It is selectively enabled by setting the open-drain control bit for the corresponding module in the ODCON registers (Register 10-1, Register 10-2 and Register 10-3). Their configuration is discussed in more detail with the individual port where these peripherals are multiplexed.

The ODCON registers all reside in the SFR configuration space and share the same SFR addresses as the Timer1 registers (see Section 5.3.4.1 “Shared Address SFRs” for more details). The ODCON registers are accessed by setting the ADSHR bit (WDTCN<4>).

When the open-drain option is required, the output pin must also be tied through an external pull-up resistor provided by the user to a higher voltage level, up to 5V on digital only pins (Figure 10-2). When a digital logic high signal is output, it is pulled up to the higher voltage level.

**FIGURE 10-2: USING THE OPEN-DRAIN OUTPUT (EUSARTx SHOWN AS EXAMPLE)**

10.1.5 TTL INPUT BUFFER OPTION

Many of the digital I/O ports use Schmitt Trigger (ST) input buffers. While this form of buffering works well with many types of input, some applications may require TTL-level signals to interface with external logic devices. This is particularly true with the EMB and the Parallel Master Port (PMP), which are particularly likely to be interfaced to TTL-level logic or memory devices.

The inputs for the PMP can be optionally configured for TTL buffers with the PMPTTL bit in the PADCFG1 register (Register 10-4). Setting this bit configures all data and control input pins for the PMP to use TTL buffers. By default, these PMP inputs use the port’s ST buffers.

As with the ODCON registers, the PADCFG1 register resides in the SFR configuration space; it shares the same memory address as the TMR2 register. PADCFG1 is accessed by setting the ADSHR bit (WDTCN<4>).

Note: These ports are not available on 64-pin devices.
REGISTER 10-1:  ODCON1: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 1

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCP5OD</td>
<td>CCP4OD</td>
<td>ECCP3OD</td>
<td>ECCP2OD</td>
</tr>
</tbody>
</table>

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

bit 7-5 Unimplemented: Read as '0'
bit 4-3 CCP5OD:CCP4OD: CCPx Open-Drain Output Enable bits
0 = Open-drain output disabled
1 = Open-drain output on CCPx pin (Capture/PWM modes) enabled

bit 2-0 ECCP3OD:ECCP1OD: ECCPx Open-Drain Output Enable bits
0 = Open-drain output disabled
1 = Open-drain output on ECCPx pin (Capture mode) enabled

REGISTER 10-2:  ODCON2: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 2

<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</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U2OD</td>
<td>U1OD</td>
</tr>
</tbody>
</table>

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

bit 7-2 Unimplemented: Read as '0'
bit 1-0 U2OD:U1OD: EUSARTx Open-Drain Output Enable bits
0 = Open-drain output disabled
1 = Open-drain output on TXx pin enabled

REGISTER 10-3:  ODCON3: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 3

<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</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPI2OD</td>
<td>SPI1OD</td>
</tr>
</tbody>
</table>

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

bit 7-2 Unimplemented: Read as '0'
bit 1-0 SPI2OD:SPI1OD: SPI Open-Drain Output Enable bits
0 = Open-drain output disabled
1 = Open-drain output on SDOx pin enabled
PORTA, TRISA and LATA Registers

PORTA is an 8-bit wide, bidirectional port. It may function as a 6-bit or 7-bit port, depending on the oscillator mode selected. The corresponding Data Direction and Output Latch registers are TRISA and LATA.

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin; it is also multiplexed as the Parallel Master Port data pin (in 80-pin devices). The other PORTA pins are multiplexed with the analog VREF+ and VREF- inputs. The operation of pins, RA<5,3:0>, as A/D Converter inputs is selected by clearing or setting the appropriate PCFG control bits in the ANCON0 register.

The RA4/T0CKI pin is a Schmitt Trigger input. All other PORTA pins have TTL input levels and full CMOS output drivers.

The TRISA register controls the direction of the PORTA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

Oscillator Mode (FOSC2:FOSC0 Configuration) | RA6 | RA7
---|---|---
INTPLL1 (011) | CLKO | I/O
INTPLL2 (010) | I/O | I/O
INTIO1 (001) | CLKO | I/O
INTIO2 (000) | I/O | I/O

Legend: CLKO = Fosc/4 clock output; I/O = digital port.

EXAMPLE 10-1: INITIALIZING PORTA

```assembly
CLRF PORTA ; Initialize PORTA by clearing output
CLRF LATA ; Alternate method to clear data latches
BSF WDTCON,ADSHR ; Enable write/read to the shared SFR
MOVLW 1Fh ; Configure A/D
MOVF WDTCON,ADSHR ; for digital inputs
BSF WDTCON,ADSHR ; Disable write/read to the shared SFR
MOVLW 0CFh ; Value used to initialize
BSF WDTCON,ADSHR ; data direction
MOVF TRISA ; Set RA<3:0> as inputs,
```

For INTIO and INTPLL Oscillator modes (FOSC2 Configuration bit is '0'), either RA7 or both RA6 and RA7 automatically become available as digital I/O, depending on the oscillator mode selected. When RA6 is not configured as a digital I/O, in these cases, it provides a clock output at Fosc/4. A list of the possible configurations for RA6 and RA7, based on oscillator mode, is provided in Table 10-3. For these pins, the corresponding PORTA, TRISA and LATA bits are only defined when the pins are configured as I/O.

### Table 10-3: Function of RA7:RA6 in INTIO and INTPLL Modes

<table>
<thead>
<tr>
<th>Oscillator Mode</th>
<th>RA6</th>
<th>RA7</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTPLL1 (011)</td>
<td>CLKO</td>
<td>I/O</td>
</tr>
<tr>
<td>INTPLL2 (010)</td>
<td>I/O</td>
<td>I/O</td>
</tr>
<tr>
<td>INTIO1 (001)</td>
<td>CLKO</td>
<td>I/O</td>
</tr>
<tr>
<td>INTIO2 (000)</td>
<td>I/O</td>
<td>I/O</td>
</tr>
</tbody>
</table>

Legend: CLKO = Fosc/4 clock output; I/O = digital port.
## TABLE 10-4: PORTA FUNCTIONS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA0/AN0</td>
<td>RA0</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATA&lt;0&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTA&lt;0&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>AN0</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 0. Default input configuration on POR; does not affect digital output.</td>
</tr>
<tr>
<td>RA1/AN1</td>
<td>RA1</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATA&lt;1&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTA&lt;1&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>AN1</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 1. Default input configuration on POR; does not affect digital output.</td>
</tr>
<tr>
<td>RA2/AN2/VREF-</td>
<td>RA2</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATA&lt;2&gt; data output; not affected by analog input. Disabled when CVREF output enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTA&lt;2&gt; data input. Disabled when analog functions enabled; disabled when CVREF output enabled.</td>
</tr>
<tr>
<td>AN2</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 2. Default input configuration on POR; does not affected by analog output.</td>
</tr>
<tr>
<td>VREF-</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D low reference voltage input.</td>
</tr>
<tr>
<td>RA3/AN3/VREF+</td>
<td>RA3</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATA&lt;3&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTA&lt;3&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>AN3</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 3. Default input configuration on POR.</td>
</tr>
<tr>
<td>VREF+</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D high reference voltage input.</td>
</tr>
<tr>
<td>RA4/PMDS5/T0CKI</td>
<td>RA4</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATA&lt;4&gt; data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTA&lt;4&gt; data input; default configuration on POR.</td>
</tr>
<tr>
<td>PMDS5(f)</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Parallel Master Port data output.</td>
</tr>
<tr>
<td>T0CKI</td>
<td></td>
<td>x</td>
<td>I</td>
<td>TTL</td>
<td>Parallel Master Port data output.</td>
</tr>
<tr>
<td>RA5/PMDS4/AN4</td>
<td>RA5</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATA&lt;5&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTA&lt;5&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>PMDS4(f)</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Parallel Master Port data output.</td>
</tr>
<tr>
<td>AN4</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 4. Default configuration on POR.</td>
</tr>
<tr>
<td>OSC2/CLKO/RA6</td>
<td>OSC2</td>
<td>x</td>
<td>O</td>
<td>ANA</td>
<td>Main oscillator feedback output connection (HS and HSPLL modes).</td>
</tr>
<tr>
<td>CLKO</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>System cycle clock output, Fosc/4 (EC, ECPLL, INTIO1 and INTPLL1 modes).</td>
</tr>
<tr>
<td>RA6</td>
<td></td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATA&lt;6&gt; data output; disabled when FOSC2 Configuration bit is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTA&lt;6&gt; data input; disabled when FOSC2 Configuration bit is set.</td>
</tr>
<tr>
<td>OSC1/CLKI/RA7</td>
<td>OSC1</td>
<td>x</td>
<td>I</td>
<td>ANA</td>
<td>Main oscillator input connection (HS and HSPLL modes).</td>
</tr>
<tr>
<td>CLKI</td>
<td></td>
<td>x</td>
<td>I</td>
<td>ANA</td>
<td>Main external clock source input (EC and ECPLL modes).</td>
</tr>
<tr>
<td>RA7</td>
<td></td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATA&lt;7&gt; data output; disabled when FOSC2 Configuration bit is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTA&lt;7&gt; data input; disabled when FOSC2 Configuration bit is set.</td>
</tr>
</tbody>
</table>

**Legend:**
- **O** = Output, **I** = Input, **ANA** = Analog Signal, **DIG** = Digital Output, **ST** = Schmitt Buffer Input, **TTL** = TTL Buffer Input, **x** = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Alternate PMP configuration when the PMPMX Configuration bit is "0"; available on 80-pin devices only.
### 10.3 PORTB, TRISB and LATB Registers

PORTB is an 8-bit wide, bidirectional port. The corresponding Data Direction register is TRISB. All pins on PORTB are digital only and tolerate voltages up to 5.5V.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit, RBPU (INTCON2<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Four of the PORTB pins (RB7:RB4) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are ORed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

This interrupt can wake the device from power-managed modes. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB (except with the MOVFF (ANY), PORTB instruction). This will end the mismatch condition.

b) Clear flag bit, RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB will end the mismatch condition and allow flag bit, RBIF, to be cleared.

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

For 80-pin devices, RB3 can be configured as the alternate peripheral pin for the ECCP2 module and Enhanced PWM output 2A by clearing the CCP2MX Configuration bit. This applies only to 80-pin devices operating in Extended Microcontroller mode. If the device is in Microcontroller mode, the alternate assignment for ECCP2 is RE7. As with other ECCP2 configurations, the user must ensure that the TRISB<3> bit is set appropriately for the intended operation. Ports, RB1, RB2, RB3, RB4 and RB5, are multiplexed with the Parallel Master Port address.

#### EXAMPLE 10-2: INITIALIZING PORTB

```assembly
CLRF PORTB ; Initialize PORTB by clearing output
          ; data latches
CLRF LATB ; Alternate method to clear output data latches
MOV LW 0CFh ; Value used to initialize data direction
MOV WF TRISB ; Set RB<3:0> as inputs
            ; RB<5:4> as outputs
            ; RB<7:6> as inputs
```

---

**TABLE 10-5: 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>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PORTA</td>
<td>RA7(1)</td>
<td>RA6(1)</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
<td>59</td>
</tr>
<tr>
<td>LATA</td>
<td>LATA7(1)</td>
<td>LATA6(1)</td>
<td>LATA5</td>
<td>LATA4</td>
<td>LATA3</td>
<td>LATA2</td>
<td>LATA1</td>
<td>LATA0</td>
<td>58</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7(1)</td>
<td>TRISA6(1)</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>58</td>
</tr>
<tr>
<td>ANCON0</td>
<td>PCFG7</td>
<td>PCFG6</td>
<td>—</td>
<td>PCFG4</td>
<td>PCFG3</td>
<td>PCFG2</td>
<td>PCFG1</td>
<td>PCFG0</td>
<td>57</td>
</tr>
</tbody>
</table>

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

Note 1: Implemented only in specific oscillator modes (FOSC2 Configuration bit = 0); otherwise read as '0'.

2: Configuration SFR, overlaps with default SFR at this address; available only when WDTCN<4> = 1.
### TABLE 10-6: PORTB FUNCTIONS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB0/INT0/FLT0</td>
<td>RB0</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATB&lt;0&gt; data output.</td>
</tr>
<tr>
<td>INT0</td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTB&lt;0&gt; data input; weak pull-up when RBPU bit is cleared.</td>
</tr>
<tr>
<td>FLT0</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>External interrupt 0 input.</td>
</tr>
<tr>
<td>RB1/INT1/PMA4</td>
<td>RB1</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATB&lt;1&gt; data output.</td>
</tr>
<tr>
<td>INT1</td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTB&lt;1&gt; data input; weak pull-up when RBPU bit is cleared.</td>
</tr>
<tr>
<td>PMA4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port address out.</td>
</tr>
<tr>
<td>RB2/INT2/PMA3</td>
<td>RB2</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATB&lt;2&gt; data output.</td>
</tr>
<tr>
<td>INT2</td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTB&lt;2&gt; data input; weak pull-up when RBPU bit is cleared.</td>
</tr>
<tr>
<td>PMA3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port address out.</td>
</tr>
<tr>
<td>RB3/INT3/PMA2/ECCP2/P2A</td>
<td>RB3</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATB&lt;3&gt; data output.</td>
</tr>
<tr>
<td>INT3</td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTB&lt;3&gt; data input; weak pull-up when RBPU bit is cleared.</td>
</tr>
<tr>
<td>PMA2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port address out.</td>
</tr>
<tr>
<td>ECCP2</td>
<td></td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>ECCP2 compare output and CCP2 PWM output; takes priority over port data.</td>
</tr>
<tr>
<td>P2A</td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>ECCP2 capture input.</td>
</tr>
<tr>
<td>RB4/KBI0/PMA1</td>
<td>RB4</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATB&lt;4&gt; data output.</td>
</tr>
<tr>
<td>KBI0</td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTB&lt;4&gt; data input; weak pull-up when RBPU bit is cleared.</td>
</tr>
<tr>
<td>PMA1</td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>Parallel Master Port address out.</td>
</tr>
<tr>
<td>RB5/KBI1/PMA0</td>
<td>RB5</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATB&lt;5&gt; data output.</td>
</tr>
<tr>
<td>KBI1</td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTB&lt;5&gt; data input; weak pull-up when RBPU bit is cleared.</td>
</tr>
<tr>
<td>PMA0</td>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>Parallel Master Port address out.</td>
</tr>
<tr>
<td>RB6/KBI2/PGC</td>
<td>RB6</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATB&lt;6&gt; data output.</td>
</tr>
<tr>
<td>KBI2</td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTB&lt;6&gt; data input; weak pull-up when RBPU bit is cleared.</td>
</tr>
<tr>
<td>PGC</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>Interrupt-on-pin change.</td>
</tr>
<tr>
<td>RB7/KBI3/PGD</td>
<td>RB7</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATB&lt;7&gt; data output.</td>
</tr>
<tr>
<td>KBI3</td>
<td></td>
<td>1</td>
<td>I</td>
<td>TTL</td>
<td>PORTB&lt;7&gt; data input; weak pull-up when RBPU bit is cleared.</td>
</tr>
<tr>
<td>PGD</td>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>Interrupt-on-pin change.</td>
</tr>
</tbody>
</table>

**Legend:**
- **O** = Output, **I** = Input, **DIG** = Digital Output, **ST** = Schmitt Buffer Input, **TTL** = TTL Buffer Input,
- **x** = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Alternate assignment for ECCP2/P2A when the CCP2MX Configuration bit is cleared (Extended Microcontroller mode, 80-pin devices only). Default assignment is RC1.

**Note 2:** All other pin functions are disabled when ICSP™ or ICD is enabled.
TABLE 10-7: 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>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<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>59</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>58</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>58</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMROIE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>INTCON2</td>
<td>RBPU</td>
<td>INTEDG0</td>
<td>INTEDG1</td>
<td>INTEDG2</td>
<td>INTEDG3</td>
<td>TMR0IP</td>
<td>INT3IP</td>
<td>RBIP</td>
<td>55</td>
</tr>
<tr>
<td>INTCON3</td>
<td>INT2IP</td>
<td>INT1IP</td>
<td>INT3IE</td>
<td>INT2IE</td>
<td>INT1IE</td>
<td>INT3IF</td>
<td>INT2IF</td>
<td>INT1IF</td>
<td>55</td>
</tr>
</tbody>
</table>

Legend: Shaded cells are not used by PORTB.

10.4 PORTC, TRISC and LATC Registers

PORTC is an 8-bit wide, bidirectional port. Only PORTC pins, RC2 through RC7, are digital only pins and can tolerate input voltages up to 5.5V.

PORTC is multiplexed with ECCP, MSSP and EUSART peripheral functions (Table 10-8). The pins have Schmitt Trigger input buffers. The pins for ECCP, SPI and EUSART are also configurable for open-drain output whenever these functions are active. Open-drain configuration is selected by setting the SPIxOD, ECCPxOD, and UxOD control bits in the ODCON registers (see Section 10.1.3 “Pull-up Configuration” for more information).

RC1 is normally configured as the default peripheral pin for the ECCP2 module. Assignment of ECCP2 is controlled by Configuration bit, CCP2MX (default state, CCP2MX = 1).

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

Note: These pins are configured as digital inputs on any device Reset.

The contents of the TRISC register are affected by peripheral overrides. Reading TRISC always returns the current contents, even though a peripheral device may be overriding one or more of the pins.

EXAMPLE 10-3: INITIALIZING PORTC

```plaintext
CLRF PORTC ; Initialize PORTC by clearing output data latches
CLRF LATC ; Alternate method to clear output data latches
MOVLW 0CFh ; Value used to initialize data direction
MOVWF TRISC ; Set RC<3:0> as inputs
             ; RC<5:4> as outputs
             ; RC<7:6> as inputs
```
## TABLE 10-8: PORTC FUNCTIONS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O Type</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC0/T1OSO/ T13CKI</td>
<td>RC0</td>
<td>0</td>
<td>O DIG</td>
<td>LATC&lt;0&gt;</td>
<td>data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTC&lt;0&gt;</td>
<td>data input.</td>
</tr>
<tr>
<td>T1OSO</td>
<td>x</td>
<td>O ANA</td>
<td></td>
<td></td>
<td>Timer1 oscillator output; enabled when Timer1 oscillator enabled. Disables digital I/O.</td>
</tr>
<tr>
<td>T13CKI</td>
<td>1</td>
<td>I ST</td>
<td></td>
<td></td>
<td>Timer1/Timer3 counter input.</td>
</tr>
<tr>
<td>RC1/T1OSI/ ECCP2/P2A</td>
<td>RC1</td>
<td>0</td>
<td>O DIG</td>
<td>LATC&lt;1&gt;</td>
<td>data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTC&lt;1&gt;</td>
<td>data input.</td>
</tr>
<tr>
<td>T1OSI</td>
<td>x</td>
<td>I ANA</td>
<td></td>
<td></td>
<td>Timer1 oscillator input; enabled when Timer1 oscillator enabled. Disables digital I/O.</td>
</tr>
<tr>
<td>ECCP2(1)</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>ECCP2 compare output and ECCP2 PWM output; takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>I ST</td>
<td></td>
<td></td>
<td>ECCP2 capture input.</td>
</tr>
<tr>
<td>P2A(1)</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>ECCP2 Enhanced PWM output, channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data.</td>
</tr>
<tr>
<td>RC2/ECCP1/ P1A</td>
<td>RC2</td>
<td>0</td>
<td>O DIG</td>
<td>LATC&lt;2&gt;</td>
<td>data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTC&lt;2&gt;</td>
<td>data input.</td>
</tr>
<tr>
<td>ECCP1</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>ECCP1 compare output and ECCP1 PWM output; takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>I ST</td>
<td></td>
<td></td>
<td>ECCP1 capture input.</td>
</tr>
<tr>
<td>P1A</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>ECCP1 Enhanced PWM output, channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data.</td>
</tr>
<tr>
<td>RC3/SCK1/ SCL1</td>
<td>RC3</td>
<td>0</td>
<td>O DIG</td>
<td>LATC&lt;3&gt;</td>
<td>data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTC&lt;3&gt;</td>
<td>data input.</td>
</tr>
<tr>
<td>SCK1</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>SPI clock output (MSSP1 module); takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>I ST</td>
<td></td>
<td></td>
<td>SPI clock input (MSSP1 module).</td>
</tr>
<tr>
<td>SCL1</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>I²C™ clock output (MSSP1 module); takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>I ST</td>
<td></td>
<td></td>
<td>I²C clock input (MSSP1 module); input type depends on module setting.</td>
</tr>
<tr>
<td>RC4/SDI1/ SDA1</td>
<td>RC4</td>
<td>0</td>
<td>O DIG</td>
<td>LATC&lt;4&gt;</td>
<td>data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTC&lt;4&gt;</td>
<td>data input.</td>
</tr>
<tr>
<td>SDI1</td>
<td>1</td>
<td>I ST</td>
<td></td>
<td></td>
<td>SPI data input (MSSP1 module).</td>
</tr>
<tr>
<td>SDA1</td>
<td>1</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>I²C data output (MSSP1 module); takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td></td>
<td>I²C data input (MSSP1 module); input type depends on module setting.</td>
</tr>
<tr>
<td>RC5/SDO1</td>
<td>RC5</td>
<td>0</td>
<td>O DIG</td>
<td>LATC&lt;5&gt;</td>
<td>data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTC&lt;5&gt;</td>
<td>data input.</td>
</tr>
<tr>
<td>SDO1</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>SPI data output (MSSP1 module); takes priority over port data.</td>
</tr>
<tr>
<td>RC6/TX1/CK1</td>
<td>RC6</td>
<td>0</td>
<td>O DIG</td>
<td>LATC&lt;6&gt;</td>
<td>data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTC&lt;6&gt;</td>
<td>data input.</td>
</tr>
<tr>
<td>TX1</td>
<td>1</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>Synchronous serial data output (EUSART1 module); takes priority over port data.</td>
</tr>
<tr>
<td>CK1</td>
<td>1</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>Synchronous serial data input (EUSART1 module). User must configure as an input.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td></td>
<td>Synchronous serial clock input (EUSART1 module).</td>
</tr>
<tr>
<td>RC7/RX1/DT1</td>
<td>RC7</td>
<td>0</td>
<td>O DIG</td>
<td>LATC&lt;7&gt;</td>
<td>data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTC&lt;7&gt;</td>
<td>data input.</td>
</tr>
<tr>
<td>RX1</td>
<td>1</td>
<td>I ST</td>
<td></td>
<td></td>
<td>Asynchronous serial receive data input (EUSART1 module).</td>
</tr>
<tr>
<td>DT1</td>
<td>1</td>
<td>O DIG</td>
<td></td>
<td></td>
<td>Synchronous serial data output (EUSART1 module); takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td></td>
<td>Synchronous serial data input (EUSART1 module). User must configure as an input.</td>
</tr>
</tbody>
</table>

**Legend:**
- O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input,
- TTL = TTL Buffer Input, x = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Default assignment for ECCP2/P2A when CCP2MX Configuration bit is set.
TABLE 10-9: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

<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>PORTC</td>
<td>RC7</td>
<td>RC6</td>
<td>RC5</td>
<td>RC4</td>
<td>RC3</td>
<td>RC2</td>
<td>RC1</td>
<td>RC0</td>
<td>59</td>
</tr>
<tr>
<td>LATC</td>
<td>LATC7</td>
<td>LATBC6</td>
<td>LATC5</td>
<td>LATCB4</td>
<td>LATC3</td>
<td>LATC2</td>
<td>LATC1</td>
<td>LATC0</td>
<td>58</td>
</tr>
<tr>
<td>TRISC</td>
<td>TRISC7</td>
<td>TRISC6</td>
<td>TRISC5</td>
<td>TRISC4</td>
<td>TRISC3</td>
<td>TRISC2</td>
<td>TRISC1</td>
<td>TRISC0</td>
<td>58</td>
</tr>
</tbody>
</table>

10.5 PORTD, TRISD and LATD Registers

PORTD is an 8-bit wide, bidirectional port. All pins on PORTD are digital only and tolerate voltages up to 5.5V.

All pins on PORTD are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

**Note:** These pins are configured as digital inputs on any device Reset.

On 80-pin devices, PORTD is multiplexed with the system bus as part of the external memory interface. I/O port and other functions are only available when the interface is disabled by setting the EBDIS bit (MEMCON<7>). When the interface is enabled, PORTD is the low-order byte of the multiplexed address/data bus (AD7:AD0). The TRISD bits are also overridden.

PORTD is also multiplexed with the data functions of the Parallel Master Port data. In this mode, Parallel Master Port takes priority over the other digital I/O (but not the external memory bus). This multiplexing is available when PMPMX = 1. When the Parallel Master Port is active, the input buffers are TTL. For more information, refer to Section 11.0 “Parallel Master Port”.

Each of the PORTD pins has a weak internal pull-up. This is performed by clearing bit RDPU (PORTG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on all device Resets.

**EXAMPLE 10-4: INITIALIZING PORTD**

```asm
CLRF PORTD ; Initialize PORTD by clearing output data latches
CLRF LATD ; Alternate method to clear output data latches
MOVLW 0CFh ; Value used to initialize data direction
MOVWF TRISD ; Set RD<3:0> as inputs
            ; RD<5:4> as outputs
            ; RD<7:6> as inputs
```
## TABLE 10-10: PORTD FUNCTIONS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD0/AD0/ PMD0</td>
<td>RD0</td>
<td>0</td>
<td>O DIG</td>
<td>LATD&lt;0&gt; data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTD&lt;0&gt; data input.</td>
</tr>
<tr>
<td></td>
<td>AD0&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>External memory interface, address/data bit 0 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 0 input.</td>
</tr>
<tr>
<td></td>
<td>PMD0&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td>RD1/AD1/ PMD1</td>
<td>RD1</td>
<td>0</td>
<td>O DIG</td>
<td>LATD&lt;1&gt; data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTD&lt;1&gt; data input.</td>
</tr>
<tr>
<td></td>
<td>AD1&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>External memory interface, address/data bit 1 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 1 input.</td>
</tr>
<tr>
<td></td>
<td>PMD1&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td>RD2/AD2/ PMD2</td>
<td>RD2</td>
<td>0</td>
<td>O DIG</td>
<td>LATD&lt;2&gt; data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTD&lt;2&gt; data input.</td>
</tr>
<tr>
<td></td>
<td>AD2&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>External memory interface, address/data bit 2 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 2 input.</td>
</tr>
<tr>
<td></td>
<td>PMD2&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td>RD3/AD3/ PMD3</td>
<td>RD3</td>
<td>0</td>
<td>O DIG</td>
<td>LATD&lt;3&gt; data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTD&lt;3&gt; data input.</td>
</tr>
<tr>
<td></td>
<td>AD3&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>External memory interface, address/data bit 3 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 3 input.</td>
</tr>
<tr>
<td></td>
<td>PMD3&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td>RD4/AD4/ PMD4/SDO2</td>
<td>RD4</td>
<td>0</td>
<td>O DIG</td>
<td>LATD&lt;4&gt; data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTD&lt;4&gt; data input.</td>
</tr>
<tr>
<td></td>
<td>AD4&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>External memory interface, address/data bit 4 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 4 input.</td>
</tr>
<tr>
<td></td>
<td>PMD4&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td></td>
<td>SDO2</td>
<td>0</td>
<td>O DIG</td>
<td>SPI data output (MSSP2 module); takes priority over port data.</td>
</tr>
<tr>
<td>RD5/AD5/ PMD5/SDI2/ SDA2</td>
<td>RD5</td>
<td>0</td>
<td>O DIG</td>
<td>LATD&lt;5&gt; data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTD&lt;5&gt; data input.</td>
</tr>
<tr>
<td></td>
<td>AD5&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>External memory interface, address/data bit 5 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 5 input.</td>
</tr>
<tr>
<td></td>
<td>PMD5&lt;0&gt;</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td></td>
<td>SDI2</td>
<td>1</td>
<td>I ST</td>
<td>SPI data output (MSSP2 module).</td>
</tr>
<tr>
<td></td>
<td>SDA2</td>
<td>1</td>
<td>O DIG</td>
<td>I²C™ data output (MSSP2 module); takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>I²C data input (MSSP2 module); input type depends on module setting.</td>
</tr>
</tbody>
</table>

**Legend:**
- O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input,
- x = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** External memory interface I/O takes priority over all other digital and PMP I/O.
**Note 2:** Available on 80-pin devices only.
**Note 3:** Default configuration for PMP (PMPMX Configuration bit = 1).
### TABLE 10-10: PORTD FUNCTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD6/AD6/</td>
<td>RD6</td>
<td>0</td>
<td>O DIG</td>
<td>LATD&lt;6&gt; data output.</td>
</tr>
<tr>
<td>PMD6/SCK2/</td>
<td>AD6(2)</td>
<td>x</td>
<td>O DIG-3</td>
<td>External memory interface, address/data bit 6 output. (^{(1)})</td>
</tr>
<tr>
<td>SCL2</td>
<td>SCK2</td>
<td>0</td>
<td>O DIG</td>
<td>SPI clock output (MSSP2 module); takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>ST</td>
<td>SPI clock input (MSSP2 module).</td>
</tr>
<tr>
<td>RD7/AD7/</td>
<td>RD7</td>
<td>0</td>
<td>O DIG</td>
<td>LATD&lt;7&gt; data output.</td>
</tr>
<tr>
<td>PMD7/SS2</td>
<td>AD7(2)</td>
<td>x</td>
<td>O DIG</td>
<td>External memory interface, address/data bit 7 output. (^{(1)})</td>
</tr>
<tr>
<td></td>
<td>PMD7(3)</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td></td>
<td>SS2</td>
<td>x</td>
<td>ST</td>
<td>Slave select input for MSSP2 module.</td>
</tr>
</tbody>
</table>

**Legend:**
- **O** = Output, **I** = Input, **DIG** = Digital Output, **ST** = Schmitt Buffer Input, **TTL** = TTL Buffer Input,
- **x** = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** External memory interface I/O takes priority over all other digital and PMP I/O.

**Note 2:** Available on 80-pin devices only.

**Note 3:** Default configuration for PMP (PMPMX Configuration bit = 1).

### TABLE 10-11: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

<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>PORTD</td>
<td>RD7</td>
<td>RD6</td>
<td>RD5</td>
<td>RD4</td>
<td>RD3</td>
<td>RD2</td>
<td>RD1</td>
<td>RD0</td>
<td>59</td>
</tr>
<tr>
<td>LATD</td>
<td>LATD7</td>
<td>LATD6</td>
<td>LATD5</td>
<td>LATD4</td>
<td>LATD3</td>
<td>LATD2</td>
<td>LATD1</td>
<td>LATD0</td>
<td>58</td>
</tr>
<tr>
<td>TRISD</td>
<td>TRISD7</td>
<td>TRISD6</td>
<td>TRISD5</td>
<td>TRISD4</td>
<td>TRISD3</td>
<td>TRISD2</td>
<td>TRISD1</td>
<td>TRISD0</td>
<td>58</td>
</tr>
<tr>
<td>PORTG</td>
<td>REPu</td>
<td>REPU</td>
<td>RJPU(4)</td>
<td>RG4</td>
<td>RG3</td>
<td>RG2</td>
<td>RG1</td>
<td>RG0</td>
<td>59</td>
</tr>
</tbody>
</table>

**Legend:** Shaded cells are not used by PORTD.

**Note 1:** Unimplemented on 64-pin devices, read as ‘0’.
10.6 PORTE, TRISE and LATE Registers

PORTE is an 8-bit wide, bidirectional port. All pins on PORTE are digital only and tolerate voltages up to 5.5V.

All pins on PORTE are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Note: These pins are configured as digital inputs on any device Reset.

On 80-pin devices, PORTE is multiplexed with the system bus as part of the external memory interface. I/O port and other functions are only available when the interface is disabled, by setting the EBDIS bit (MEMCON<7>). When the interface is enabled, PORTE is the high-order byte of the multiplexed address/data bus (AD15:AD8). The TRISE bits are also overridden.

Each of the PORTE pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is performed by clearing bit REPU (PORTG<6>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on any device Reset.

PORTE is also multiplexed with Enhanced PWM outputs B and C for ECCP1 and ECCP3 and outputs B, C and D for ECCP2. For all devices, their default assignments are on PORTE<6:0>. On 80-pin devices, the multiplexing for the outputs of ECCP1 and ECCP3 is controlled by the ECCPMX Configuration bit. Clearing this bit reassigns the P1B/P1C and P3B/P3C outputs to PORTH.

For devices operating in Microcontroller mode, pin RE7 can be configured as the alternate peripheral pin for the ECCP2 module and Enhanced PWM output 2A. This is done by clearing the CCP2MX Configuration bit.

PORTE is also multiplexed with the Parallel Master Port address lines. When PMPMX = 0, RE1 and RE0 are multiplexed with the control signals PMWR and PMRD.

RE3 can also be configured as the Reference Clock Output (REFO) from the system clock. For further details, refer to Section 2.6 “Reference Clock Output”.

EXAMPLE 10-5: INITIALIZING PORTE

Note: These pins are configured as digital inputs on any device Reset.

| CLRF PORTE ; Initialize PORTE by clearing output |
| CLRF LATE ; Alternate method to clear output data latches |
| MOVWF TRISE ; Set RE<1:0> as inputs RE<7:2> as outputs |

CLRF PORTE ; Initialize PORTE by clearing output
CLRF LATE ; Alternate method to clear output data latches
MOVLW 03h ; Value used to initialize data direction
MOVWF TRISE ; Set RE<1:0> as inputs RE<7:2> as outputs
### TABLE 10-12: PORTE FUNCTIONS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE0/AD8/</td>
<td>RE0</td>
<td>0</td>
<td>O DIG</td>
<td>LATE&lt;0&gt; data output.</td>
</tr>
<tr>
<td>PMRD/P2D</td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTE&lt;0&gt; data input.</td>
</tr>
<tr>
<td>AD8</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>External memory interface, address/data bit 8 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 8 input.</td>
</tr>
<tr>
<td>PMRD</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>Parallel Master Port read strobe pin.</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>I TTL</td>
<td></td>
<td>Parallel Master Port read pin.</td>
</tr>
<tr>
<td>P2D</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td>ECCP2 Enhanced PWM output, channel D; takes priority over port and PMP data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>May be configured for tri-state during Enhanced PWM shutdown events.</td>
</tr>
<tr>
<td>RE1/AD9/</td>
<td>RE1</td>
<td>0</td>
<td>O DIG</td>
<td>LATE&lt;1&gt; data output.</td>
</tr>
<tr>
<td>PMWR/P2C</td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTE&lt;1&gt; data input.</td>
</tr>
<tr>
<td>AD9</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>External memory interface, address/data bit 9 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 9 input.</td>
</tr>
<tr>
<td>PMWR</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>Parallel Master Port write strobe pin.</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>I TTL</td>
<td></td>
<td>Parallel Master Port write pin.</td>
</tr>
<tr>
<td>P2C</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td>ECCP2 Enhanced PWM output, channel C; takes priority over port and PMP data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>May be configured for tri-state during Enhanced PWM shutdown events.</td>
</tr>
<tr>
<td>RE2/AD10/</td>
<td>RE2</td>
<td>0</td>
<td>O DIG</td>
<td>LATE&lt;2&gt; data output.</td>
</tr>
<tr>
<td>PMBE/P2B</td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTE&lt;2&gt; data input.</td>
</tr>
<tr>
<td>AD10</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>External memory interface, address/data bit 10 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 10 input.</td>
</tr>
<tr>
<td>PMBE</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>Parallel Master Port byte enable.</td>
</tr>
<tr>
<td>P2B</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td>ECCP2 Enhanced PWM output, channel B; takes priority over port and PMP data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>May be configured for tri-state during Enhanced PWM shutdown events.</td>
</tr>
<tr>
<td>RE3/AD11/</td>
<td>RE3</td>
<td>0</td>
<td>O DIG</td>
<td>LATE&lt;3&gt; data output.</td>
</tr>
<tr>
<td>PMA13/P3C/REFO</td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTE&lt;3&gt; data input.</td>
</tr>
<tr>
<td>AD11</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>External memory interface, address/data bit 11 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 11 input.</td>
</tr>
<tr>
<td>PMA13</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>P3C</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td>ECCP3 Enhanced PWM output, channel C; takes priority over port and PMP data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>May be configured for tri-state during Enhanced PWM shutdown events.</td>
</tr>
<tr>
<td>REFO</td>
<td></td>
<td>O DIG</td>
<td></td>
<td>Reference output clock.</td>
</tr>
<tr>
<td>RE4/AD12/</td>
<td>RE4</td>
<td>0</td>
<td>O DIG</td>
<td>LATE&lt;4&gt; data output.</td>
</tr>
<tr>
<td>PMA12/P3B</td>
<td></td>
<td>1</td>
<td>I ST</td>
<td>PORTE&lt;4&gt; data input.</td>
</tr>
<tr>
<td>AD12</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>External memory interface, address/data bit 12 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I TTL</td>
<td>External memory interface, data bit 12 input.</td>
</tr>
<tr>
<td>PMA12</td>
<td>x</td>
<td>O DIG</td>
<td></td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>P3B</td>
<td>0</td>
<td>O DIG</td>
<td></td>
<td>ECCP3 Enhanced PWM output, channel B; takes priority over port and PMP data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>May be configured for tri-state during Enhanced PWM shutdown events.</td>
</tr>
</tbody>
</table>

**Legend:**
- **O** = Output, **I** = Input, **DIG** = Digital Output, **ST** = Schmitt Buffer Input, **TTL** = TTL Buffer Input,
- **x** = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

**Note**
1: Default assignments for P1B/P1C and P3B/P3C when ECCPMX Configuration bit is set (80-pin devices only).
2: External memory interface I/O takes priority over all other digital and PMP I/O.
3: Available on 80-pin devices only.
4: Alternate assignment for ECCP2/P2A when ECCP2MX Configuration bit is cleared (all devices in Microcontroller mode).
5: Default configuration for PMP (PMPMX Configuration bit = 1).
<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE5/AD13/</td>
<td>RE5</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATE&lt;5&gt; data output.</td>
</tr>
<tr>
<td>PMA11/P1C</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTE&lt;5&gt; data input.</td>
</tr>
<tr>
<td>AD13</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>External memory interface, address/data bit 13 output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>I</td>
<td>TTL</td>
<td>External memory interface, data bit 13 input.</td>
</tr>
<tr>
<td>PMA11</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>P1C</td>
<td></td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>ECCP1 Enhanced PWM output, channel C; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events.</td>
</tr>
</tbody>
</table>

| RE6/AD14/       | RE6      | 0            | O   | DIG     | LATE<6> data output.                                                                                                                            |
| PMA10/P1B       |          | 1            | I   | ST      | PORTE<6> data input.                                                                                                                            |
| AD14            |          | x            | O   | DIG     | External memory interface, address/data bit 14 output.                                                                                         |
|                 |          | x            | I   | TTL     | External memory interface, data bit 14 input.                                                                                                 |
| PMA10           |          | x            | O   | DIG     | Parallel Master Port address.                                                                                                                   |
| P1B             |          | 0            | O   | DIG     | ECCP1 Enhanced PWM output, channel B; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |

| RE7/AD15/       | RE7      | 0            | O   | DIG     | LATE<7> data output.                                                                                                                            |
| PMA9/ECCP2/     |          | 1            | I   | ST      | PORTE<7> data input.                                                                                                                            |
| P2A             |          | x            | O   | DIG     | ECCP2 compare output and ECCP2 PWM output; takes priority over port data.                                                                   |
| P2A             |          | 1            | I   | ST      | ECCP2 capture input.                                                                                                                           |

Legend: O = Output, I = Input, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

Note 1: Default assignments for P1B/P1C and P3B/P3C when ECCPMX Configuration bit is set (80-pin devices only).
Note 2: External memory interface I/O takes priority over all other digital and PMP I/O.
Note 3: Available on 80-pin devices only.
Note 4: Alternate assignment for ECCP2/P2A when ECCP2MX Configuration bit is cleared (all devices in Microcontroller mode).
Note 5: Default configuration for PMP (PMPMX Configuration bit = 1).

<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>PORTE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>LATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>TRISE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>PORTG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
</tbody>
</table>

Legend: Shaded cells are not used by PORTE.

Note 1: Unimplemented on 64-pin devices, read as ‘0’.
10.7 PORTF, LATF and TRISF Registers

PORTF is a 7-bit wide, bidirectional port. Only pin 7 of PORTF has no analog input; it is the only pin that can tolerate voltages up to 5.5V.

All pins on PORTF are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

PORTF is multiplexed with analog peripheral functions. RF1 through RF6 may also be used as analog input channels for the A/D Converter. All pins may be used as comparator inputs or outputs by setting the appropriate bits in the CMCON register. To use RF<6:3> as digital inputs, it is also necessary to turn off the comparators.

Note 1: On device Resets, pins RF6:RF1 are configured as analog inputs and are read as ’0’.

2: To configure PORTF as digital I/O, set the corresponding bits in ANCON0 and ANCON1.

EXAMPLE 10-6: INITIALIZING PORTF

| CLRF PORTF | ; Initialize PORTF by clearing output data latches |
| CLRF LATF | ; Alternate method to clear output latches |
| BSF WDTCON,ADSHR | ; Enable write/read to the shared SFR |
| MOVWF ANCON0 | ; make RF1:RF2 digital |
| MOVWF ANCON1 | ; make RF<6:3> digital |
| BCF WDTCON,ADSHR | ; Disable write/read to the shared SFR |
| MOVWF CEh | |
| MOVWF TRISF | ; Set RF5:RF4 as outputs, RF<7:6>,<3:1> as inputs |
### TABLE 10-14: PORTF FUNCTIONS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF1/AN6/</td>
<td>RF1</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATF&lt;1&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td>C2OUT</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTF&lt;1&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>AN6</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 6. Default configuration on POR.</td>
</tr>
<tr>
<td>C2OUT</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Comparator 2 output.</td>
</tr>
<tr>
<td>RF2/PMA5/</td>
<td>RF2</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATF&lt;2&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td>AN7/C1OUT</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTF&lt;2&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>PMA5</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td>AN7</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 7. Default configuration on POR.</td>
</tr>
<tr>
<td>C1OUT</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Comparator 1 output.</td>
</tr>
<tr>
<td>RF3/AN8/</td>
<td>RF3</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATF&lt;3&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td>C2INB</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTF&lt;3&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>AN8</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 8. Default configuration on POR.</td>
</tr>
<tr>
<td>C2INB</td>
<td></td>
<td>x</td>
<td>I</td>
<td>ANA</td>
<td>Comparator 2 input B.</td>
</tr>
<tr>
<td>RF4/AN9/</td>
<td>RF4</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATF&lt;4&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td>C2INA</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTF&lt;4&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>AN9</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 9. Default configuration on POR.</td>
</tr>
<tr>
<td>C2INA</td>
<td></td>
<td>x</td>
<td>I</td>
<td>ANA</td>
<td>Comparator 2 input A.</td>
</tr>
<tr>
<td>RF5/PMD2/</td>
<td>RF5</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATF&lt;5&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td>AN10/C1INB/</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTF&lt;5&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>CVREF</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td>PMD2(1)</td>
<td></td>
<td>x</td>
<td>I</td>
<td>TTL</td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td>AN10</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 10 and Comparator C1+ input. Default input</td>
</tr>
<tr>
<td>C1INB</td>
<td></td>
<td>x</td>
<td>I</td>
<td>ANA</td>
<td>Comparator 1 input B.</td>
</tr>
<tr>
<td>CVREF</td>
<td></td>
<td>x</td>
<td>O</td>
<td>ANA</td>
<td>Comparator voltage reference output. Enabling this feature</td>
</tr>
<tr>
<td>CVREF</td>
<td></td>
<td>x</td>
<td>O</td>
<td>ANA</td>
<td>Comparator voltage reference output. Enabling this feature</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>affects digital I/O.</td>
</tr>
<tr>
<td>RF6/PMD1/</td>
<td>RF6</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATF&lt;6&gt; data output; not affected by analog input.</td>
</tr>
<tr>
<td>AN11/C1INA/</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTF&lt;6&gt; data input; disabled when analog input enabled.</td>
</tr>
<tr>
<td>PMD1(1)</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td>AN11</td>
<td></td>
<td>x</td>
<td>I</td>
<td>TTL</td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td>C1INA</td>
<td></td>
<td>x</td>
<td>I</td>
<td>ANA</td>
<td>Comparator 1 input A.</td>
</tr>
<tr>
<td>RF7/PMD0/</td>
<td>RF7</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATF&lt;7&gt; data output.</td>
</tr>
<tr>
<td>SS1</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTF&lt;7&gt; data input.</td>
</tr>
<tr>
<td>PMD0(1)</td>
<td></td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td>SS1</td>
<td></td>
<td>x</td>
<td>I</td>
<td>TTL</td>
<td>Parallel Master Port data input.</td>
</tr>
</tbody>
</table>

**Legend:**  
- **O** = Output, **I** = Input, **ANA** = Analog Signal, **DIG** = Digital Output, **ST** = Schmitt Buffer Input, **TTL** = TTL Buffer Input, **x** = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Alternate PMP configuration when the PMPMX Configuration bit = 0; available on 80-pin devices only.
TABLE 10-15: SUMMARY OF REGISTERS ASSOCIATED WITH PORTF

<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>PORTF</td>
<td>RF7</td>
<td>RF6</td>
<td>RF5</td>
<td>RF4</td>
<td>RF3</td>
<td>RF2</td>
<td>RF1</td>
<td>—</td>
<td>59</td>
</tr>
<tr>
<td>LATF</td>
<td>LATF7</td>
<td>LATF6</td>
<td>LATF5</td>
<td>LATF4</td>
<td>LATF3</td>
<td>LATF2</td>
<td>LATF1</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>TRISF</td>
<td>TRISF7</td>
<td>TRISF6</td>
<td>TRISF5</td>
<td>TRISF4</td>
<td>TRISF3</td>
<td>TRISF2</td>
<td>TRISF1</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>ANCON0(1)</td>
<td>PCFG7</td>
<td>PCFG6</td>
<td>—</td>
<td>PCFG4</td>
<td>PCFG3</td>
<td>PCFG2</td>
<td>PCFG1</td>
<td>PCFG0</td>
<td>57</td>
</tr>
<tr>
<td>ANCON1(1)</td>
<td>PCFG15</td>
<td>PCFG14</td>
<td>PCFG13</td>
<td>PCFG12</td>
<td>PCFG11</td>
<td>PCFG10</td>
<td>PCFG9</td>
<td>PCFG8</td>
<td>57</td>
</tr>
</tbody>
</table>

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

Note 1: Configuration SFR, overlaps with default SFR at this address; available only when WDTCN<4> = 1.

10.8 PORTG, TRISG and LATG Registers

PORTG is a 5-bit wide, bidirectional port. All pins on PORTG are digital only and tolerate voltages up to 5.5V.

PORTG is multiplexed with EUSART2 functions (Table 10-16). PORTG pins have Schmitt Trigger input buffers. PORTG is also multiplexed with address and control functions of the Parallel Master Port.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTG pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings. The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register without concern due to peripheral overrides.

Although the port itself is only five bits wide, PORTG<7:5> bits are still implemented. These are used to control the weak pull-ups on the I/O ports associated with the external memory bus (PORTD, PORTE and PORTJ). Setting these bits enables the pull-ups. Since these are control bits and are not associated with port I/O, the corresponding TRISG and LATG bits are not implemented.

EXAMPLE 10-7: INITIALIZING PORTG

```mlist
CLRF PORTG ; Initialize PORTG by clearing output ; data latches
CLRF LATG ; Alternate method to clear output data latches
MOVLW 04h ; Value used to initialize data direction
MOVWF TRISG ; Set RG1:RG0 as outputs ; RG2 as input ; RG4:RG3 as outputs
```
### TABLE 10-16: PORTG FUNCTIONS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RG0/PMA8/ECCP3/P3A</td>
<td>RG0</td>
<td>0</td>
<td>O DIG</td>
<td>LATG&lt;0&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>PMA8</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td></td>
<td>ECCP3</td>
<td>O</td>
<td>DIG</td>
<td>ECCP3 compare and PWM output; takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td>P3A</td>
<td>0</td>
<td>O DIG</td>
<td>ECCP3 Enhanced PWM output, channel A; takes priority over port and PMP data.</td>
</tr>
<tr>
<td>RG1/PMA7/TX2/CK2</td>
<td>RG1</td>
<td>0</td>
<td>O DIG</td>
<td>LATG&lt;1&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>PMA7</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td></td>
<td>TX2</td>
<td>1</td>
<td>O DIG</td>
<td>Synchronous serial data output (EUSART2 module); takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td>CK2</td>
<td>1</td>
<td>O DIG</td>
<td>Synchronous serial data input (EUSART2 module). User must configure as an input.</td>
</tr>
<tr>
<td>RG2/PMA6/RX2/DT2</td>
<td>RG2</td>
<td>0</td>
<td>O DIG</td>
<td>LATG&lt;2&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>PMA6</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port address.</td>
</tr>
<tr>
<td></td>
<td>RX2</td>
<td>1</td>
<td>I ST</td>
<td>Asynchronous serial receive data input (EUSART2 module).</td>
</tr>
<tr>
<td></td>
<td>DT2</td>
<td>1</td>
<td>O DIG</td>
<td>Synchronous serial data output (EUSART2 module); takes priority over port data.</td>
</tr>
<tr>
<td>RG3/PMCS1/CCP4/P3D</td>
<td>RG3</td>
<td>0</td>
<td>O DIG</td>
<td>LATG&lt;3&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>PMCS1</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port address chip select 1</td>
</tr>
<tr>
<td></td>
<td>P3D</td>
<td>0</td>
<td>O DIG</td>
<td>ECCP3 Enhanced PWM output, channel D; takes priority over port and PMP data.</td>
</tr>
<tr>
<td>RG4/PMCS2/CCP5/P1D</td>
<td>RG4</td>
<td>0</td>
<td>O DIG</td>
<td>LATG&lt;4&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>PMCS2</td>
<td>x</td>
<td>O DIG</td>
<td>Parallel Master Port address chip select 2</td>
</tr>
<tr>
<td></td>
<td>CCP5</td>
<td>0</td>
<td>O DIG</td>
<td>CCP5 compare output and CCP5 PWM output; takes priority over port data.</td>
</tr>
<tr>
<td></td>
<td>P1D</td>
<td>0</td>
<td>O DIG</td>
<td>ECCP1 Enhanced PWM output, channel D; takes priority over port and PMP data.</td>
</tr>
</tbody>
</table>

**Legend:**
- **O** = Output, **I** = Input, **DIG** = Digital Output, **ST** = Schmitt Buffer Input, **TTL** = TTL Buffer Input,
- **x** = Don't care (TRIS bit does not affect port direction or is overridden for this option).
10.9 PORTH, LATH and TRISH Registers

PORTH is an 8-bit wide, bidirectional I/O port. PORTH pins <3:0> are digital only and tolerate voltages up to 5.5V.

All pins on PORTH are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

When the external memory interface is enabled, four of the PORTH pins function as the high-order address lines for the interface. The address output from the interface takes priority over other digital I/O. The corresponding TRISH bits are also overridden. PORTH pins, RH4 through RH7, are multiplexed with analog converter inputs. The operation of these pins as analog inputs is selected by clearing or setting the corresponding bits in the ANCON1 register. RH2 to RH6 are multiplexed with the Parallel Master Port and RH4 to RH6 are multiplexed as comparator inputs.

PORTH can also be configured as the alternate Enhanced PWM output channels B and C for the ECCP1 and ECCP3 modules. This is done by clearing the ECCPMX Configuration bit.

EXAMPLE 10-8: INITIALIZING PORTH

<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>PORTG</td>
<td>RDPU</td>
<td>REPU</td>
<td>RJPU(1)</td>
<td>RG4</td>
<td>RG3</td>
<td>RG2</td>
<td>RG1</td>
<td>RG0</td>
<td>59</td>
</tr>
<tr>
<td>LATG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>LATG4</td>
<td>LATG3</td>
<td>LATG2</td>
<td>LATG1</td>
<td>LATG0</td>
<td>58</td>
</tr>
<tr>
<td>TRISG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRISG4</td>
<td>TRISG3</td>
<td>TRISG2</td>
<td>TRISG1</td>
<td>TRISG0</td>
<td>58</td>
</tr>
</tbody>
</table>

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

Note 1: Unimplemented on 64-pin devices, read as '0'.

Note: PORTH is available only on 80-pin devices.
## TABLE 10-18: PORHRESULTS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RH0/A16</td>
<td>RH0</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATH&lt;0&gt; data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTH&lt;0&gt; data input.</td>
</tr>
<tr>
<td>A16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>External memory interface, address line 16. Takes priority over port data.</td>
</tr>
<tr>
<td>RH1/A17</td>
<td>RH1</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATH&lt;1&gt; data output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTH&lt;1&gt; data input.</td>
</tr>
<tr>
<td>A17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>External memory interface, address line 17. Takes priority over port data.</td>
</tr>
<tr>
<td>RH2/A18/</td>
<td>RH2</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATH&lt;2&gt; data output.</td>
</tr>
<tr>
<td>PMD7</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTH&lt;2&gt; data input.</td>
</tr>
<tr>
<td>A18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>External memory interface, address line 18. Takes priority over port data.</td>
</tr>
<tr>
<td>PMD7&lt;2&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td>A19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td>RH3/A19/</td>
<td>RH3</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATH&lt;3&gt; data output.</td>
</tr>
<tr>
<td>PMD6</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTH&lt;3&gt; data input.</td>
</tr>
<tr>
<td>A19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>External memory interface, address line 19. Takes priority over port data.</td>
</tr>
<tr>
<td>PMD6&lt;2&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port data out.</td>
</tr>
<tr>
<td>A19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel Master Port data input.</td>
</tr>
<tr>
<td>RH4/PMD3/</td>
<td>RH4</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATH&lt;4&gt; data output.</td>
</tr>
<tr>
<td>AN12/P3C</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTH&lt;4&gt; data input.</td>
</tr>
<tr>
<td>AN12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A/D input channel 12. Default input configuration on POR; does not affect</td>
</tr>
<tr>
<td>P3C&lt;1&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Digital output.</td>
</tr>
<tr>
<td>C2INC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Comparator 2 input C</td>
</tr>
<tr>
<td>RH5/PMRD/</td>
<td>RH5</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATH&lt;5&gt; data output.</td>
</tr>
<tr>
<td>AN13/P3B</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTH&lt;5&gt; data input.</td>
</tr>
<tr>
<td>P3B&lt;1&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A/D input channel 13. Default input configuration on POR; does not affect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Digital output.</td>
</tr>
<tr>
<td>C2IND</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Comparator 2 input D</td>
</tr>
<tr>
<td>RH6/PMRD/</td>
<td>RH6</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATH&lt;6&gt; data output.</td>
</tr>
<tr>
<td>AN14/P1C</td>
<td></td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTH&lt;6&gt; data input.</td>
</tr>
<tr>
<td>P1C&lt;1&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A/D input channel 14. Default input configuration on POR; does not affect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Digital output.</td>
</tr>
<tr>
<td>C1INC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Comparator 1 input C</td>
</tr>
</tbody>
</table>

**Legend:**
- O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input,
  TTL = TTL Buffer Input, x = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Alternate assignments for P1B/P1C and P3B/P3C when the ECCP[MX Configuration bit is cleared. Default assignments are PORTE<6:3>.

**Note 2:** Alternate PMP configuration when the PMPMX Configuration bit = 0; available on 80-pin devices only.
### TABLE 10-18: PORTH FUNCTIONS (CONTINUED)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RH7/PMWR/AN15/P1B</td>
<td>RH7</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATH&lt;7&gt; data output.</td>
</tr>
<tr>
<td>PMWR(2)</td>
<td>x</td>
<td>O</td>
<td>DIG</td>
<td>Parallel Master Port write strobe.</td>
<td></td>
</tr>
<tr>
<td>AN15</td>
<td>x</td>
<td>I</td>
<td>TTL</td>
<td>Parallel Master Port write in.</td>
<td></td>
</tr>
<tr>
<td>P1B(2)</td>
<td>x</td>
<td>I</td>
<td>ANA</td>
<td>A/D input channel 15. Default input configuration on POR; does not affect digital output.</td>
<td></td>
</tr>
<tr>
<td>P1B(2)</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>ECCP1 Enhanced PWM output, channel B; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events.</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** Alternate assignments for P1B/P1C and P3B/P3C when the ECCPMX Configuration bit is cleared. Default assignments are PORTE<6:3>.

**Note 2:** Alternate PMP configuration when the PMPMX Configuration bit = 0; available on 80-pin devices only.

### TABLE 10-19: SUMMARY OF REGISTERS ASSOCIATED WITH PORTH

<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>PORTH(1)</td>
<td>RH7</td>
<td>RH6</td>
<td>RH5</td>
<td>RH4</td>
<td>RH3</td>
<td>RH2</td>
<td>RH1</td>
<td>RH0</td>
<td>58</td>
</tr>
<tr>
<td>LATH(1)</td>
<td>LATH7</td>
<td>LATH6</td>
<td>LATH5</td>
<td>LATH4</td>
<td>LATH3</td>
<td>LATH2</td>
<td>LATH1</td>
<td>LATH0</td>
<td>59</td>
</tr>
<tr>
<td>TRISH(1)</td>
<td>TRISH7</td>
<td>TRISH6</td>
<td>TRISH5</td>
<td>TRISH4</td>
<td>TRISH3</td>
<td>TRISH2</td>
<td>TRISH1</td>
<td>TRISH0</td>
<td>58</td>
</tr>
<tr>
<td>ANCON1(2)</td>
<td>PCFG15</td>
<td>PCFG14</td>
<td>PCFG13</td>
<td>PCFG12</td>
<td>PCFG11</td>
<td>PCFG10</td>
<td>PCFG9</td>
<td>PCFG8</td>
<td>57</td>
</tr>
</tbody>
</table>

**Legend:** Shaded cells are not used by PORTH.

**Note 1:** Unimplemented on 64-pin devices, read as ‘0’.

**Note 2:** Configuration SFR, overlaps with default SFR at this address; available only when WDTCONG<4> = 1.
10.10 PORTJ, TRISJ and LATJ Registers

PORTJ is an 8-bit wide, bidirectional port. All pins on PORTJ are digital only and tolerate voltages up to 5.5V. All pins on PORTJ are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

When the external memory interface is enabled, all of the PORTJ pins function as control outputs for the interface. This occurs automatically when the interface is enabled by clearing the EBDIS control bit (MEMCON<7>). The TRISJ bits are also overridden.

Each of the PORTJ pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is performed by clearing bit RJPU (PORTG<5>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on any device Reset.

**EXAMPLE 10-9: INITIALIZING PORTJ**

Note: PORTJ is available only on 80-pin devices.

Note: These pins are configured as digital inputs on any device Reset.

Note: PORTJ is available only on 80-pin devices.

Note: These pins are configured as digital inputs on any device Reset.

| CLRF PORTJ ; Initialize PORTG by clearing output data latches |
| CLRF LATJ ; Alternate method to clear output data latches |
| MOVFW OCFh ; Value used to initialize data direction |
| MOVWF TRISJ ; Set RJ3:RJ0 as inputs RJ5:RJ4 as output RJ7:RJ6 as inputs |
# TABLE 10-20: PORTJ FUNCTIONS

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Function</th>
<th>TRIS Setting</th>
<th>I/O Type</th>
<th>I/O Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RJ0/ALE</td>
<td>RJ0</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATJ&lt;0&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>ALE</td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTJ&lt;0&gt; data input.</td>
</tr>
<tr>
<td>RJ1/Œ</td>
<td>RJ1</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATJ&lt;1&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>Œ</td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTJ&lt;1&gt; data input.</td>
</tr>
<tr>
<td>RJ2/WRL</td>
<td>RJ2</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATJ&lt;2&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>WRL</td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTJ&lt;2&gt; data input.</td>
</tr>
<tr>
<td>RJ3/WRH</td>
<td>RJ3</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATJ&lt;3&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>WRH</td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTJ&lt;3&gt; data input.</td>
</tr>
<tr>
<td>RJ4/BA0</td>
<td>RJ4</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATJ&lt;4&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>BA0</td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTJ&lt;4&gt; data input.</td>
</tr>
<tr>
<td>RJ5/Œ</td>
<td>RJ5</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATJ&lt;5&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>Œ</td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTJ&lt;5&gt; data input.</td>
</tr>
<tr>
<td>RJ6/LB</td>
<td>RJ6</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATJ&lt;6&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>LB</td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTJ&lt;6&gt; data input.</td>
</tr>
<tr>
<td>RJ7/UB</td>
<td>RJ7</td>
<td>0</td>
<td>O</td>
<td>DIG</td>
<td>LATJ&lt;7&gt; data output.</td>
</tr>
<tr>
<td></td>
<td>UB</td>
<td>1</td>
<td>I</td>
<td>ST</td>
<td>PORTJ&lt;7&gt; data input.</td>
</tr>
</tbody>
</table>

**Legend:**
- **O** = Output
- **I** = Input
- **DIG** = Digital Output
- **ST** = Schmitt Buffer Input
- **x** = Don’t care (TRIS bit does not affect port direction or is overridden for this option).

# TABLE 10-21: SUMMARY OF REGISTERS ASSOCIATED WITH PORTJ

<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>PORTJ&lt;1&gt;</td>
<td>RJ7</td>
<td>RJ6</td>
<td>RJ5</td>
<td>RJ4</td>
<td>RJ3</td>
<td>RJ2</td>
<td>RJ1</td>
<td>RJ0</td>
<td>59</td>
</tr>
<tr>
<td>LATJ&lt;1&gt;</td>
<td>LATJ7</td>
<td>LATJ6</td>
<td>LATJ5</td>
<td>LATJ4</td>
<td>LATJ3</td>
<td>LATJ2</td>
<td>LATJ1</td>
<td>LATJ0</td>
<td>58</td>
</tr>
<tr>
<td>TRISJ&lt;1&gt;</td>
<td>TRISJ7</td>
<td>TRISJ6</td>
<td>TRISJ5</td>
<td>TRISJ4</td>
<td>TRISJ3</td>
<td>TRISJ2</td>
<td>TRISJ1</td>
<td>TRISJ0</td>
<td>58</td>
</tr>
<tr>
<td>PORTG</td>
<td>RDPU</td>
<td>REPJ</td>
<td>RJPJ</td>
<td>RG4</td>
<td>RG3</td>
<td>RG2</td>
<td>RG1</td>
<td>RG0</td>
<td>59</td>
</tr>
</tbody>
</table>

**Legend:** Shaded cells are not used by PORTJ.

**Note 1:** Unimplemented on 64-pin devices, read as ‘0’.
11.0 PARALLEL MASTER PORT

The Parallel Master Port module (PMP) is a parallel, 8-bit I/O module, specifically designed to communicate with a wide variety of parallel devices, such as communication peripherals, LCDs, external memory devices and microcontrollers. Because the interface to parallel peripherals varies significantly, the PMP is highly configurable. The PMP module can be configured to serve as either a Parallel Master Port or as a Parallel Slave Port.

Key features of the PMP module include:
- Up to 16 Programmable Address Lines
- Up to Two Chip Select Lines
- Programmable Strobe Options
  - Individual Read and Write Strobes or;
  - Read/Write Strobe with Enable Strobe
- Address Auto-Increment/Auto-Decrement
- Programmable Address/Data Multiplexing
- Programmable Polarity on Control Signals
- Legacy Parallel Slave Port Support
- Enhanced Parallel Slave Support
  - Address Support
  - 4-Byte Deep, Auto-Incrementing Buffer
- Programmable Wait States
- Selectable Input Voltage Levels

FIGURE 11-1: PMP MODULE OVERVIEW
11.1 Module Registers

The PMP module has a total of 14 Special Function Registers for its operation, plus one additional register to set configuration options. Of these, 8 registers are used for control and 6 are used for PMP data transfer.

11.1.1 CONTROL REGISTERS

The eight PMP Control registers are:
- PMCONH and PMCONL
- PMMODEH and PMMODEL
- PMSTATL and PMSTATH
- PMEH and PMEL

The PMCON registers (Register 11-1 and Register 11-2) control basic module operations, including turning the module on or off. They also configure address multiplexing and control strobe configuration.

The PMMODE registers (Register 11-3 and Register 11-4) configure the various Master and Slave Operating modes, the data width and interrupt generation.

The PMEH and PMEL registers (Register 11-5 and Register 11-6) configure the module’s operation at the hardware (I/O pin) level.

The PMSTAT registers (Register 11-7 and Register 11-8) provide status flags for the module’s input and output buffers, depending on the operating mode.

REGISTER 11-1: PMCONH: PARALLEL PORT CONTROL HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th></th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMPEN</td>
<td></td>
<td></td>
<td>PSIDL</td>
<td>ADRMUX1</td>
<td>ADRMUX0</td>
<td>PTBEEN</td>
<td>PTWREN</td>
<td>PTRDEN</td>
</tr>
<tr>
<td>bit 7</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’
-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

bit 7  **PMPEN**: Parallel Master Port Enable bit
1 = PMP enabled  0 = PMP disabled, no off-chip access performed

bit 6  **Unimplemented**: Read as ‘0’

bit 5  **PSIDL**: Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode  0 = Continue module operation in Idle mode

bit 4-3  **ADRMUX1**: **ADRMUX0**: Address/Data Multiplexing Selection bits
11 = Reserved  10 = All 16 bits of address are multiplexed on PMD<7:0> pins  01 = Lower 8 bits of address are multiplexed on PMD<7:0> pins, upper 8 bits are on PMA<15:8>  00 = Address and data appear on separate pins

bit 2  **PTBEEN**: Byte Enable Port Enable bit (16-bit Master mode)
1 = PMBE port enabled  0 = PMBE port disabled

bit 1  **PTWREN**: Write Enable Strobe Port Enable bit
1 = PMWR/PMENB port enabled  0 = PMWR/PMENB port disabled

bit 0  **PTRDEN**: Read/Write Strobe Port Enable bit
1 = PMRD/PMWR port enabled  0 = PMRD/PMWR port disabled
REGISTER 11-2:  PMCONL: PARALLEL PORT CONTROL LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0(1)</th>
<th>R/W-0(1)</th>
<th>R/W-0(1)</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSF1</td>
<td>CSF0</td>
<td>ALP</td>
<td>CS2P</td>
<td>CS1P</td>
<td>BEP</td>
<td>WRSP</td>
<td>RDSP</td>
<td></td>
</tr>
</tbody>
</table>

Legend:

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

**bit 7-6**  
CSF1:CSF0: Chip Select Function bits
- 11 = Reserved
- 10 = PMCS1 and PMCS2 function as chip select
- 01 = PMCS2 functions as chip select, PMCS1 used as address bit 14 (PMADDRH address bit 6)
- 00 = PMCS2 and PMCS1 used as address bits 15 and 14 (PMADDRH address bits 7 and 6)

**bit 5**  
ALP: Address Latch Polarity bit(1)
- 1 = Active-high (PMALL and PMALH)
- 0 = Active-low (PMALL and PMALH)

**bit 4**  
CS2P: Chip Select 2 Polarity bit(1)
- 1 = Active-high (PMCS2)
- 0 = Active-low (PMCS2)

**bit 3**  
CS1P: Chip Select 1 Polarity bit(1)
- 1 = Active-high (PMCS1/PMCS)
- 0 = Active-low (PMCS1/PMCS)

**bit 2**  
BEP: Byte Enable Polarity bit
- 1 = Byte enable active-high (PMBE)
- 0 = Byte enable active-low (PMBE)

**bit 1**  
WRSP: Write Strobe Polarity bit
For Slave modes and Master mode 2 (PMODEH<1:0> = 00, 01, 10):
- 1 = Write strobe active-high (PMWR)
- 0 = Write strobe active-low (PMWR)
For Master mode 1 (PMODEH<1:0> = 11):
- 1 = Enable strobe active-high (PMENB)
- 0 = Enable strobe active-low (PMENB)

**bit 0**  
RDSP: Read Strobe Polarity bit
For Slave modes and Master mode 2 (PMODEH<1:0> = 00, 01, 10):
- 1 = Read strobe active-high (PMRD)
- 0 = Read strobe active-low (PMRD)
For Master mode 1 (PMODEH<1:0> = 11):
- 1 = Read/write strobe active-high (PMRD/PMWR)
- 0 = Read/write strobe active-low (PMRD/PMWR)

**Note 1:** These bits have no effect when their corresponding pins are used as address lines.
REGISTER 11-3: PMMODEH: PARALLEL PORT MODE HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>R-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>IRQM1</td>
<td>IRQM0</td>
<td>INCM1</td>
<td>INCM0</td>
<td>MODE16</td>
<td>MODE1</td>
<td>MODE0</td>
</tr>
</tbody>
</table>

bit 7  
BUSY: Busy bit (Master mode only)  
1 = Port is busy  
0 = Port is not busy

bit 6-5  
IRQM1:IRQM0: Interrupt Request Mode bits  
11 = Interrupt generated when Read Buffer 3 is read or Write Buffer 3 is written (Buffered PSP mode) or on a read or write operation when PMA<1:0> = 11 (Addressable PSP mode only)  
10 = No interrupt generated, processor stall activated  
01 = Interrupt generated at the end of the read/write cycle  
00 = No interrupt generated

bit 4-3  
INCM1:INCM0: Increment Mode bits  
11 = PSP read and write buffers auto-increment (Legacy PSP mode only)  
10 = Decrement ADDR<15,13:0> by 1 every read/write cycle  
01 = Increment ADDR<15,13:0> by 1 every read/write cycle  
00 = No increment or decrement of address

bit 2  
MODE16: 8/16-Bit Mode bit  
1 = 16-Bit mode: data register is 16 bits, a read or write to the data register invokes two 8-bit transfers  
0 = 8-Bit mode: data register is 8 bits, a read or write to the data register invokes one 8-bit transfer

bit 1-0  
MODE1:MODE0: Parallel Port Mode Select bits  
11 = Master mode 1 (PMCSx, PMRD/PMWR, PMENB, PMBE, PMA<x:0> and PMD<7:0>)  
10 = Master mode 2 (PMCSx, PMRD, PMWR, PMBE, PMA<x:0> and PMD<7:0>)  
01 = Enhanced PSP, control signals (PMRD, PMWR, PMCS, PMD<7:0> and PMA<1:0>)  
00 = Legacy Parallel Slave Port, control signals (PMRD, PMWR, PMCS and PMD<7:0>)

Legend:  
R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’  
-n = Value at POR  
‘1’ = Bit is set  
‘0’ = Bit is cleared  
x = Bit is unknown
REGISTER 11-4:  PMMODEL: PARALLEL PORT MODE LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-6</td>
<td>WAITB1:WAITB0: Data Setup to Read/Write Wait State Configuration bits</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WAITM3:WAITM2: Data Hold After Strobe Wait State Configuration bits</td>
<td></td>
</tr>
<tr>
<td>5-2</td>
<td>WAITM3:WAITM0: Read to Byte Enable Strobe Wait State Configuration bits</td>
<td></td>
</tr>
<tr>
<td>1-0</td>
<td>WAITE1:WAITE0: Data Hold After Strobe Wait State Configuration bits</td>
<td></td>
</tr>
</tbody>
</table>

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

Note 1: WAITB and WAITE bits are ignored whenever WAITM3:WAITM0 = 0000.

REGISTER 11-5:  PMEH: PARALLEL PORT ENABLE HIGH BYTE REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-6</td>
<td>PTEN15:PTEN14: PMCSx Strobe Enable bits</td>
<td></td>
</tr>
<tr>
<td>5-0</td>
<td>PTEN13:PTEN8: PMP Address Port Enable bits</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown
**REGISTER 11-6: PMEL: PARALLEL PORT ENABLE LOW BYTE REGISTER**

<table>
<thead>
<tr>
<th></th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTEN7</td>
<td></td>
<td>PTEN6</td>
<td>PTEN5</td>
<td>PTEN4</td>
<td>PTEN3</td>
<td>PTEN2</td>
<td>PTEN1</td>
<td>PTEN0</td>
</tr>
</tbody>
</table>

Legend:

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

Bit 7-2

**PTEN7:** PMP Address Port Enable bits

- **1** = PMA<7:2> function as PMP address lines
- **0** = PMA<7:2> function as port I/O

Bit 1-0

**PTEN1:** PMALH/PMALL Strobe Enable bits

- **1** = PMA1 and PMA0 function as either PMA<1:0> or PMALH and PMALL
- **0** = PMA1 and PMA0 pads functions as port I/O

---

**REGISTER 11-7: PMSTATH: PARALLEL PORT STATUS HIGH BYTE REGISTER**

<table>
<thead>
<tr>
<th></th>
<th>R-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBF</td>
<td></td>
<td>IBOV</td>
<td>—</td>
<td>—</td>
<td>IB3F</td>
<td>IB2F</td>
<td>IB1F</td>
<td>IB0F</td>
</tr>
</tbody>
</table>

Legend:

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

Bit 7

**IBF:** Input Buffer Full Status bit

- **1** = All writable input buffer registers are full
- **0** = Some or all of the writable input buffer registers are empty

Bit 6

**IBOV:** Input Buffer Overflow Status bit

- **1** = A write attempt to a full input byte register occurred (must be cleared in software)
- **0** = No overflow occurred

Bit 5-4

**Unimplemented:** Read as ‘0’

Bit 3-0

**IB3F:IB0F:** Input Buffer Status Full bits

- **1** = Input buffer contains data that has not been read (reading buffer will clear this bit)
- **0** = Input buffer does not contain any unread data
### REGISTER 11-8: PMSTATL: PARALLEL PORT STATUS LOW BYTE REGISTER

<table>
<thead>
<tr>
<th>R-1</th>
<th>R/W-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R-1</th>
<th>R-1</th>
<th>R-1</th>
<th>R-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBE</td>
<td>OBUF</td>
<td>—</td>
<td>—</td>
<td>OB3E</td>
<td>OB2E</td>
<td>OB1E</td>
<td>OB0E</td>
</tr>
</tbody>
</table>

#### Legend:

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

**bit 7**  
**OBE**: Output Buffer Empty Status bit  
1 = All readable output buffer registers are empty  
0 = Some or all of the readable output buffer registers are full

**bit 6**  
**OBUF**: Output Buffer Underflow Status bit  
1 = A read occurred from an empty output byte register (must be cleared in software)  
0 = No underflow occurred

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

**bit 3-0**  
**OBnE**: Output Buffer n Status Empty bit  
1 = Output buffer is empty (writing data to the buffer will clear this bit)  
0 = Output buffer contains data that has not been transmitted
11.1.2 DATA REGISTERS

The PMP module uses 6 registers for transferring data into and out of the microcontroller. They are arranged as three pairs to allow the option of 16-bit data operations:

- PMDIN1H and PMDIN1L
- PMDIN2H and PMDIN2L
- PMADDRH/PMOUT1H and PMADDRL/PMOUT1L
- PMDOUT2H and PMDOUT2L

The PMDIN1 register is used for incoming data in Slave modes, and both input and output data in Master modes. The PMDIN2 register is used for buffering input data in select Slave modes.

The PMADDRx/PMOUT1x registers are actually a single register pair; the name and function is dictated by the module’s operating mode. In Master modes, the registers functions as the PMADDRH and PMADDRL registers, and contain the address of any incoming or outgoing data. In Slave modes, the registers function as PMOUT1H and PMOUT1L and are used for outgoing data.

PMADDRH differs from PMADDRL in that it can also have limited PMP control functions. When the module is operating in select Master mode configurations, the upper two bits of the register can be used to determine the operation of chip select signals. If chip select signals are not used, PMADDR simply functions to hold the upper 8 bits of the address. The function of the individual bits in PMADDRH is shown in Register 11-9.

The PMDOUT2H and PMDOUT2L registers are only used in buffered Slave modes and serve as a buffer for outgoing data.

11.1.3 PAD CONFIGURATION CONTROL REGISTER

In addition to the module level configuration options, the PMP module can also be configured at the I/O pin for electrical operation. This option allows users to select either the normal Schmitt Trigger input buffer on digital I/O pins shared with the PMP, or use TTL level compatible buffers instead. Buffer configuration is controlled by the PMPTTL bit in the PADCFS1 register. The PADCFS1 register is one of the shared address SFRs, and has the same address as the TMR2 register. PADCFS1 is accessed by setting the ADSHR bit (WDTCON<4>). Refer to Section 5.3.4.1 “Shared Address SFRs” for more information.

REGISTER 11-9: PMADDR: PARALLEL PORT ADDRESS REGISTER, HIGH BYTE
(MASTER MODES ONLY)(1)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS2</td>
<td>CS1</td>
<td>ADDR&lt;13:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as '0'

-? = Value at Reset  
1 = bit is set  
0 = bit is cleared  
x = bit is unknown

- bit 7 CS2: Chip Select 2 bit
  
  if PMCON<7:6> = 10 or 01:  
  1 = Chip Select 2 is active  
  0 = Chip Select 2 is inactive

- bit 6 CS1: Chip Select 1 bit
  
  if PMCON<7:6> = 10:  
  1 = Chip Select 1 is active  
  0 = Chip Select 1 is inactive

- bit 5-0 ADDR13:ADDR0: Destination Address bits

Note 1: In Enhanced Slave mode, PMADDR functions as PMOUT1H, one of the Output Data Buffer registers.
11.1.4 PMP MULTIPLEXING OPTIONS (80-PIN DEVICES)

By default, the PMP and the external memory bus multiplex some of their signals to the same I/O pins on PORTD and PORTE. It is possible that some applications may require the PMP signals to be located elsewhere. For these instances, the 80-pin devices can be configured to multiplex the PMP to different I/O ports. PMP configuration is determined by the PMPMX Configuration bit setting; by default, the PMP and EMB modules share PORTD and PORTE. The optional pin configuration is shown in Table 11-1.

TABLE 11-1: PMP PIN MULTIPLEXING FOR 80-PIN DEVICES

<table>
<thead>
<tr>
<th>PMP Function</th>
<th>Pin Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PMPMX = 1</td>
</tr>
<tr>
<td>PMDO</td>
<td>PORTD&lt;0&gt;</td>
</tr>
<tr>
<td>PMD1</td>
<td>PORTD&lt;1&gt;</td>
</tr>
<tr>
<td>PMD2</td>
<td>PORTD&lt;2&gt;</td>
</tr>
<tr>
<td>PMD3</td>
<td>PORTD&lt;3&gt;</td>
</tr>
<tr>
<td>PMD4</td>
<td>PORTD&lt;4&gt;</td>
</tr>
<tr>
<td>PMD5</td>
<td>PORTD&lt;5&gt;</td>
</tr>
<tr>
<td>PMD6</td>
<td>PORTD&lt;6&gt;</td>
</tr>
<tr>
<td>PMD7</td>
<td>PORTD&lt;7&gt;</td>
</tr>
<tr>
<td>PMBE</td>
<td>PORTE&lt;2&gt;</td>
</tr>
<tr>
<td>PMWR</td>
<td>PORTE&lt;1&gt;</td>
</tr>
<tr>
<td>PMRD</td>
<td>PORTE&lt;0&gt;</td>
</tr>
</tbody>
</table>

11.2 Slave Port Modes

The primary mode of operation for the module is configured using the MODE1:MODE0 bits in the PMMODEH register. The setting affects whether the module acts as a slave or a master and it determines the usage of the control pins.

11.2.1 LEGACY MODE (PSP)

In Legacy mode (PMMODEH<1:0> = 00 and PMPEN = 1), the module is configured as a Parallel Slave Port with the associated enabled module pins dedicated to the module. In this mode, an external device, such as another microcontroller or microprocessor, can asynchronously read and write data using the 8-bit data bus (PMD<7:0>), the read (PMRD), write (PMWR) and chip select (PMCS1) inputs. It acts as a slave on the bus and responds to the read/write control signals.

Figure 11-2 shows the connection of the Parallel Slave Port. When chip select is active and a write strobe occurs (PMCS = 1 and PMWR = 1), the data from PMD<7:0> is captured into the PMDIN1L register.
11.2.1.1 WRITE TO SLAVE PORT

When chip select is active and a write strobe occurs (PMCS = 1 and PMWR = 1), the data from PMD<7:0> is captured into the PMDIN1L register. The PMPIF and IBF flag bits are set when the write ends. The timing for the control signals in Write mode is shown in Figure 11-3. The polarity of the control signals are configurable.

11.2.1.2 READ FROM SLAVE PORT

When chip select is active and a read strobe occurs (PMCS = 1 and PMRD = 1), the data from the PMDOTUL1 register (PMDOUTL1<7:0>) is presented onto PMD<7:0>. The timing for the control signals in Read mode is shown in Figure 11-4.

FIGURE 11-3: PARALLEL SLAVE PORT WRITE WAVEFORMS

FIGURE 11-4: PARALLEL SLAVE PORT READ WAVEFORMS
11.2.2 BUFFERED PARALLEL SLAVE PORT MODE

Buffered Parallel Slave Port mode is functionally identical to the Legacy Parallel Slave Port mode with one exception: the implementation of 4-level read and write buffers. Buffered PSP mode is enabled by setting the INCM bits in the PMMODE register. If the INCM<1:0> bits are set to '11', the PMP module will act as the Buffered Parallel Slave Port.

When the Buffered mode is active, the PMDIN1L, PMDIN1H, PMDIN2L and PMDIN2H registers become the write buffers and the PMDOUT1L, PMDOUT1H, PMDOUT2L and PMDOUT2H registers become the read buffers. Buffers are numbered 0 through 3, starting with the lower byte of PMDIN1L to PMDIN2H as the read buffers, and PMDOUT1L to PMDOUT2H as the write buffers.

11.2.2.1 READ FROM SLAVE PORT

For read operations, the bytes will be sent out sequentially, starting with Buffer 0 (PMDOUT1L<7:0>) and ending with Buffer 3 (PMDOUT2H<7:0>) for every read strobe. The module maintains an internal pointer to keep track of which buffer is to be read. Each of the buffers has a corresponding read status bit, OBxE, in the PMSTATL register. This bit is cleared when a buffer contains data that has not been written to the bus, and is set when data is written to the bus. If the current buffer location being read from is empty, a buffer underflow is generated, and the Buffer Overflow flag bit OBUF is set. If all 4 OBxE status bits are set, then the Output Buffer Empty flag (OBE) will also be set.

11.2.2.2 WRITE TO SLAVE PORT

For write operations, the data is be stored sequentially, starting with Buffer 0 (PMDIN1L<7:0>) and ending with Buffer 3 (PMDIN2H<7:0>). As with read operations, the module maintains an internal pointer to the buffer that is to be written next.

The input buffers have their own write status bits, IBxF in the PMSTATH register. The bit is set when the buffer contains unread incoming data, and cleared when the data has been read. The flag bit is set on the write strobe. If a write occurs on a buffer when its associated IBxF bit is set, the Buffer Overflow flag, IBOV, is set; any incoming data in the buffer will be lost. If all 4 IBxF flags are set, the Input Buffer Full Flag (IBF) is set.

In Buffered Slave mode, the module can be configured to generate an interrupt on every read or write strobe (IRQM1:IRQM0 = 01). It can be configured to generate an interrupt on a read from Read Buffer 3 or a write to Write Buffer 3, which is essentially an interrupt every fourth read or write strobe (RQM1:IRQM0 = 11). When interrupting every fourth byte for input data, all input buffer registers should be read to clear the IBxF flags. If these flags are not cleared, then there is a risk of hitting an overflow condition.

---

**FIGURE 11-5: PARALLEL MASTER/SLAVE CONNECTION BUFFERED EXAMPLE**
11.2.3 ADDRESSABLE PARALLEL SLAVE PORT MODE

In the Addressable Parallel Slave Port mode (PMMODEH<1:0> = 01), the module is configured with two extra inputs, PMA<1:0>, which are the address lines 1 and 0. This makes the 4-byte buffer space directly addressable as fixed pairs of read and write buffers. As with Buffered Legacy mode, data is output from PMDOUT1L, PMDOUT1H, PMDOUT2L and PMDOUT2H, and is read in PMDIN1L, PMDIN1H, PMDIN2L and PMDIN2H. Table 11-2 shows the buffer addressing for the incoming address to the input and output registers.

**TABLE 11-2: SLAVE MODE BUFFER ADDRESSING**

<table>
<thead>
<tr>
<th>PMADDR&lt;1:0&gt;</th>
<th>Output Register (Buffer)</th>
<th>Input Register (Buffer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>PMDOUT1L (0)</td>
<td>PMDIN1L (0)</td>
</tr>
<tr>
<td>01</td>
<td>PMDOUT1H (1)</td>
<td>PMDIN1H (1)</td>
</tr>
<tr>
<td>10</td>
<td>PMDOUT2L (2)</td>
<td>PMDIN2L (2)</td>
</tr>
<tr>
<td>11</td>
<td>PMDOUT2H (3)</td>
<td>PMDIN2H (3)</td>
</tr>
</tbody>
</table>

**FIGURE 11-6: PARALLEL MASTER/SLAVE CONNECTION ADDRESSED BUFFER EXAMPLE**
11.2.3.1 READ FROM SLAVE PORT

When chip select is active and a read strobe occurs (PMCS = 1 and PMRD = 1), the data from one of the four output bytes is presented onto PMD<7:0>. Which byte is read depends on the 2-bit address placed on ADDR<1:0>. Table 11-2 shows the corresponding output registers and their associated address.

When an output buffer is read, the corresponding OBxE bit is set. The OBE flag bit is set when all the buffers are empty. If any buffer is already empty (OBxE = 1), the next read to that buffer will generate an OBUF event.

FIGURE 11-7: PARALLEL SLAVE PORT READ WAVEFORMS

11.2.3.2 WRITE TO SLAVE PORT

When chip select is active and a write strobe occurs (PMCS = 1 and PMWR = 1), the data from PMD<7:0> is captured into one of the four input buffer bytes. Which byte is written depends on the 2-bit address placed on ADDRL<1:0>. Table 11-2 shows the corresponding input registers and their associated address.

When an input buffer is written, the corresponding IBxF bit is set. The IBF flag bit is set when all the buffers are written. If any buffer is already written (IBxF = 1), the next write strobe to that buffer will generate an OBUF event and the byte will be discarded.

FIGURE 11-8: PARALLEL SLAVE PORT WRITE WAVEFORMS
11.3 Master Port Modes

In its Master modes, the PMP module provides an 8-bit data bus, up to 16 bits of address, and all the necessary control signals to operate a variety of external parallel devices, such as memory devices, peripherals and slave microcontrollers. To use the PMP as a master, the module must be enabled (PMPEN = 1) and the mode must be set to one of the two possible Master modes (PMMODEH<1:0> = 10 or 11).

Because there are a number of parallel devices with a variety of control methods, the PMP module is designed to be extremely flexible to accommodate a range of configurations. Some of these features include:

- 8 and 16-Bit Data modes on an 8-bit data bus
- Configurable address/data multiplexing
- Up to two chip select lines
- Up to 16 selectable address lines
- Address auto-increment and auto-decrement
- Selectable polarity on all control lines
- Configurable wait states at different stages of the read/write cycle

11.3.1 PMP AND I/O PIN CONTROL

Multiple control bits are used to configure the presence or absence of control and address signals in the module. These bits are PTBEEN, PTWREN, PTREDN, and PTEN<15:0>. They give the user the ability to conserve pins for other functions and allow flexibility to control the external address. When any one of these bits is set, the associated function is present on its associated pin; when clear, the associated pin reverts to its defined I/O port function.

Setting a PTEN bit will enable the associated pin as an address pin and drive the corresponding data contained in the PMADDR register. Clearing the PTENx bit will force the pin to revert to its original I/O function.

For the pins configured as chip select (PMCS1 or PMCS2) with the corresponding PTENx bit set, chip select pins drive inactive data (with polarity defined by the CS1P and CS2P bits) when a read or write operation is not being performed. The PTEN0 and PTEN1 bits also control the PMALL and PMALH signals. When multiplexing is used, the associated address latch signals should be enabled.

11.3.2 READ/WRITE CONTROL

The PMP module supports two distinct read/write signaling methods. In Master mode 1, read and write strobes are combined into a single control line, PMRD/PMWR. A second control line, PMENB, determines when a read or write action is to be taken. In Master mode 2, separate read and write strobes (PMRD and PMWR) are supplied on separate pins.

All control signals (PMRD, PMWR, PMBE, PMENB, PMAL and PMCSx) can be individually configured as either positive or negative polarity. Configuration is controlled by separate bits in the PMCONL register. Note that the polarity of control signals that share the same output pin (for example, PMWR and PMENB) are controlled by the same bit; the configuration depends on which Master Port mode is being used.

11.3.3 DATA WIDTH

The PMP supports data widths of both 8 and 16 bits. The data width is selected by the MODE16 bit (PMMODEH<2>). Because the data path into and out of the module is only 8 bits wide, 16-bit operations are always handled in a multiplexed fashion, with the Least Significant Byte of data being presented first. To differentiate data bytes, the Byte Enable (PMBE) control strobe is used to signal when the Most Significant Byte of data is being presented on the data lines.

11.3.4 ADDRESS MULTIPLEXING

In either of the Master modes (PMMODEH<1:0> = 1x), the user can configure the address bus to be multiplexed together with the data bus. This is accomplished using the ADRMUX1:ADRMUX0 bits (PMCONH<4:3>). There are three address multiplexing modes available; typical pinout configurations for these modes are shown in Figure 11-9, Figure 11-10 and Figure 11-11.

In Demultiplexed mode (PMCONH<4:3> = 00), data and address information are completely separated. Data bits are presented on PMD<7:0>, and address bits are presented on PMADDRH<7:0> and PMADDR<7:0>.

In Partially Multiplexed mode (PMCONH<4:3> = 01), the lower eight bits of the address are multiplexed with the data pins on PMD<7:0>. The upper eight bits of address are unaffected and are presented on PMADDRH<7:0>. The PMA0 pin is used as an address latch and presents the Address Latch Low (PMALL) enable strobe. The read and write sequences are extended by a complete CPU cycle during which the address is presented on the PMD<7:0> pins.

In Fully Multiplexed mode (PMCONH<4:3> = 10), the entire 16 bits of the address are multiplexed with the data pins on PMD<7:0>. The PMA0 and PMA1 pins are used to present Address Latch Low (PMALL) enable and Address Latch High (PMALH) enable strobes, respectively. The read and write sequences are extended by two complete CPU cycles. During the first cycle, the lower eight bits of the address are presented on the PMD<7:0> pins with the PMALL strobe active. During the second cycle, the upper eight bits of the address are presented on the PMD<7:0> pins with the PMALH strobe active. In the event the upper address bits are configured as chip select pins, the corresponding address bits are automatically forced to '0'.
FIGURE 11-9: DEMULTIPLEXED ADDRESSING MODE (SEPARATE READ AND WRITE STROBES, TWO CHIP SELECTS)

PIC18F

PMRD
PMWR
PMD<7:0>
PMCS1
PMCS2
PMRD
PMWR

Address Bus
Data Bus
Control Lines

FIGURE 11-10: PARTIALLY MULTIPLEXED ADDRESSING MODE (SEPARATE READ AND WRITE STROBES, TWO CHIP SELECTS)

PIC18F

PMRD
PMWR
PMD<7:0>
PMA<7:0>
PMCS1
PMCS2
PMALL
PMRD
PMWR

Address Bus
Multiplexed Data and Address Bus
Control Lines

FIGURE 11-11: FULLY MULTIPLEXED ADDRESSING MODE (SEPARATE READ AND WRITE STROBES, TWO CHIP SELECTS)

PIC18F

PMRD
PMWR
PMD<7:0>
PMA<13:8>
PMCS1
PMCS2
PMALL
PMALH
PMRD
PMWR

Multiplexed Data and Address Bus
Control Lines
11.3.5 CHIP SELECT FEATURES

Up to two chip select lines, PMCS1 and PMCS2, are available for the Master modes of the PMP. The two chip select lines are multiplexed with the Most Significant bits of the address bus (PMADDRH<6> and PMADDRH<7>). When a pin is configured as a chip select, it is not included in any address auto-increment/decrement. The function of the chip select signals is configured using the chip select function bits (PMCONL <7:6>).

11.3.6 AUTO-INCREMENT/DECREMENT

While the module is operating in one of the Master modes, the INCM bits (PMMODEH<3:4>) control the behavior of the address value. The address can be made to automatically increment or decrement after each read and write operation. The address increments once each operation is completed and the BUSY bit goes to ‘0’. If the chip select signals are disabled and configured as address bits, the bits will participate in the increment and decrement operations; otherwise, the CS2 and CS1 bit values will be unaffected.

11.3.7 WAIT STATES

In Master mode, the user has control over the duration of the read, write and address cycles by configuring the module wait states. Three portions of the cycle, the beginning, middle, and end, are configured using the corresponding WAITBx, WAITMx and WAITEx bits in the PMMODEL register.

The WAITB1:WAITB0 bits (PMMODEL<7:6>) set the number of wait cycles for the data setup prior to the PMRD/PMWT strobe in Mode 10, or prior to the PMENB strobe in Mode 11. The WAITM3:WAITM0 bits (PMMODEL<5:2>) set the number of wait cycles for the PMRD/PMWT strobe in Mode 10, or for the PMENB strobe in Mode 11. When this wait state setting is 0, then WAITB and WAITE have no effect. The WAITE1:WAITE0 bits (PMMODEL<1:0>) define the number of wait cycles for the data hold time after the PMRD/PMWT strobe in Mode 10, or after the PMENB strobe in Mode 11.

11.3.8 READ OPERATION

To perform a read on the Parallel Master Port, the user reads the PMDIN1L register. This causes the PMP to output the desired values on the chip select lines and the address bus. Then the read line (PMRD) is strobed. The read data is placed into the PMDIN1L register.

If the 16-bit mode is enabled (MODE16 = 1), the read of the low byte of the PMDIN1L register will initiate two bus reads. The first read data byte is placed into the PMDIN1L register, and the second read data is placed into the PMDIN1H.

Note that the read data obtained from the PMDIN1L register is actually the read value from the previous read operation. Hence, the first user read will be a dummy read to initiate the first bus read and fill the read register. Also, the requested read value will not be ready until after the BUSY bit is observed low. Thus, in a back-to-back read operation, the data read from the register will be the same for both reads. The next read of the register will yield the new value.

11.3.9 WRITE OPERATION

To perform a write onto the parallel bus, the user writes to the PMDIN1L register. This causes the module to first output the desired values on the chip select lines and the address bus. The write data from the PMDIN1L register is placed onto the PMD<7:0> data bus. Then the write line (PMWR) is strobed. If the 16-bit mode is enabled (MODE16 = 1), the write to the PMDIN1L register will initiate two bus writes. First write will consist of the data contained in PMDIN1L and the second write will contain the PMDIN1H.

11.3.10 PARALLEL MASTER PORT STATUS

11.3.10.1 The BUSY Bit

In addition to the PMP interrupt, a BUSY bit is provided to indicate the status of the module. This bit is only used in Master mode. While any read or write operation is in progress, the BUSY bit is set for all but the very last CPU cycle of the operation. In effect, if a single-cycle read or write operation is requested, the BUSY bit will never be active. This allows back-to-back transfers. While the bit is set, any request by the user to initiate a new operation will be ignored (i.e., writing or reading the lower byte of the PMDIN1L register will not initiate either a read nor a write).

11.3.10.2 INTERRUPTS

When the PMP module interrupt is enabled for Master mode, the module will interrupt on every completed read or write cycle; otherwise, the BUSY bit is available to query the status of the module.
11.3.11 MASTER MODE TIMING

This section contains a number of timing examples that represent the common Master mode configuration options. These options vary from 8-bit to 16-bit data, fully demultiplexed to fully multiplexed address, as well as wait states.

FIGURE 11-12: READ AND WRITE TIMING, 8-BIT DATA, DEMULTIPLEXED ADDRESS

FIGURE 11-13: READ TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS
FIGURE 11-14: READ TIMING, 8-BIT DATA, WAIT STATES ENABLED, PARTIALLY MULTIPLEXED ADDRESS

FIGURE 11-15: WRITE TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS

FIGURE 11-16: WRITE TIMING, 8-BIT DATA, WAIT STATES ENABLED, PARTIALLY MULTIPLEXED ADDRESS
FIGURE 11-17: READ TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS, ENABLE STROBE

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PMCS2</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>PMCS1</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>PMD&lt;7:0&gt;</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>PMA&lt;13:8&gt;</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>PMRD/PMWR</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>PMENB</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>PMALL</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>PMPIF</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>BUSY</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>

FIGURE 11-18: WRITE TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS, ENABLE STROBE

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PMCS2</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>PMCS1</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>PMD&lt;7:0&gt;</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>PMA&lt;13:8&gt;</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>PMRD/PMWR</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>PMENB</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>PMALL</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>PMPIF</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>BUSY</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>

FIGURE 11-19: READ TIMING, 8-BIT DATA, FULLY MULTIPLEXED 16-BIT ADDRESS

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PMCS2</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>PMCS1</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>PMD&lt;7:0&gt;</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>PMWR</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>PMRD</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>PMALL</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>PMALH</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>PMPIF</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>BUSY</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>
FIGURE 11-20: WRITE TIMING, 8-BIT DATA, FULLY MULTIPLEXED 16-BIT ADDRESS

FIGURE 11-21: READ TIMING, 16-BIT DATA, DEMULTIPLEXED ADDRESS

FIGURE 11-22: WRITE TIMING, 16-BIT DATA, DEMULTIPLEXED ADDRESS
FIGURE 11-23: READ TIMING, 16-BIT MULTIPLEXED DATA, PARTIALLY MULTIPLEXED ADDRESS

FIGURE 11-24: WRITE TIMING, 16-BIT MULTIPLEXED DATA, PARTIALLY MULTIPLEXED ADDRESS
FIGURE 11-25: READ TIMING, 16-BIT MULTIPLEXED DATA,
FULLY MULTIPLEXED 16-BIT ADDRESS

FIGURE 11-26: WRITE TIMING, 16-BIT MULTIPLEXED DATA,
FULLY MULTIPLEXED 16-BIT ADDRESS
11.4 Application Examples

This section introduces some potential applications for the PMP module.

11.4.1 MULTIPLEXED MEMORY OR PERIPHERAL

Figure 11-27 demonstrates the hookup of a memory or other addressable peripheral in Full Multiplex mode. Consequently, this mode achieves the best pin saving from the microcontroller perspective. However, for this configuration, there needs to be some external latches to maintain the address.

**FIGURE 11-27: EXAMPLE OF A MULTIPLEXED ADDRESSING APPLICATION**

Partial multiplexing implies using more pins; however, for a few extra pins, some extra performance can be achieved. Figure 11-28 shows an example of a memory or peripheral that is partially multiplexed with an external latch. If the peripheral has internal latches as shown in Figure 11-29, then no extra circuitry is required except for the peripheral itself.

**FIGURE 11-28: EXAMPLE OF A PARTIALLY MULTIPLEXED ADDRESSING APPLICATION**

**FIGURE 11-29: EXAMPLE OF AN 8-BIT MULTIPLEXED ADDRESS AND DATA APPLICATION**
11.4.3 PARALLEL EEPROM EXAMPLE

Figure 11-30 shows an example connecting parallel EEPROM to the PMP. Figure 11-31 shows a slight variation to this, configuring the connection for 16-bit data from a single EEPROM.

FIGURE 11-30: PARALLEL EEPROM EXAMPLE (UP TO 15-BIT ADDRESS, 8-BIT DATA)

FIGURE 11-31: PARALLEL EEPROM EXAMPLE (UP TO 15-BIT ADDRESS, 16-BIT DATA)

11.4.4 LCD CONTROLLER EXAMPLE

The PMP module can be configured to connect to a typical LCD controller interface, as shown in Figure 11-32. In this case, the PMP module is configured for active-high control signals since common LCD displays require active-high control.

FIGURE 11-32: LCD CONTROL EXAMPLE (BYTE MODE OPERATION)
### TABLE 11-3: REGISTERS ASSOCIATED WITH PMP 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>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPRI</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>PMCONH</td>
<td>PMPEN</td>
<td>—</td>
<td>PSIDL</td>
<td>ADRMUX1</td>
<td>ADRMUX0</td>
<td>PTBEN</td>
<td>PTWREN</td>
<td>PTRDEN</td>
<td>60</td>
</tr>
<tr>
<td>PMCONL</td>
<td>CSF1</td>
<td>CSF0</td>
<td>ALP</td>
<td>CS2P</td>
<td>CS1P</td>
<td>BEP</td>
<td>WRSP</td>
<td>RDSP</td>
<td>60</td>
</tr>
<tr>
<td>PMADDRH/PMOUT1H&lt;br&gt;(1)</td>
<td>CS2</td>
<td>CS1</td>
<td>Parallel Master Port Address High Byte</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMADDRL/PMOUT1L&lt;br&gt;(1)</td>
<td>Parallel Master Port Address Low Byte</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMOUT2H</td>
<td>Parallel Port Out Data High Byte (Buffer 0)</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMOUT2L</td>
<td>Parallel Port Out Data Low Byte (Buffer 0)</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMIDIN1H</td>
<td>Parallel Port In Data High Byte (Buffer 1)</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMIDIN1L</td>
<td>Parallel Port In Data Low Byte (Buffer 0)</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMIDIN2H</td>
<td>Parallel Port In Data High Byte (Buffer 3)</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMIDIN2L</td>
<td>Parallel Port In Data Low Byte (Buffer 2)</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMMODEH</td>
<td>BUSY</td>
<td>IRQM1</td>
<td>IRQM0</td>
<td>INXM1</td>
<td>INCM0</td>
<td>MODE16</td>
<td>MODE1</td>
<td>MODE0</td>
<td>60</td>
</tr>
<tr>
<td>PMMODEL</td>
<td>WAITB1</td>
<td>WAITB0</td>
<td>WAITM3</td>
<td>WAITM2</td>
<td>WAITM1</td>
<td>WAITM0</td>
<td>WAITE1</td>
<td>WAITE0</td>
<td>60</td>
</tr>
<tr>
<td>PMEH</td>
<td>PTEN15</td>
<td>PTEN14</td>
<td>PTEN13</td>
<td>PTEN12</td>
<td>PTEN11</td>
<td>PTEN10</td>
<td>PTEN9</td>
<td>PTEN8</td>
<td>60</td>
</tr>
<tr>
<td>PMEL</td>
<td>PTEN7</td>
<td>PTEN6</td>
<td>PTEN5</td>
<td>PTEN4</td>
<td>PTEN3</td>
<td>PTEN2</td>
<td>PTEN1</td>
<td>PTEN0</td>
<td>60</td>
</tr>
<tr>
<td>PMSTATH</td>
<td>IBF</td>
<td>IBOV</td>
<td>—</td>
<td>—</td>
<td>IB3F</td>
<td>IB2F</td>
<td>IB1F</td>
<td>IB0F</td>
<td>60</td>
</tr>
<tr>
<td>PMSTATL</td>
<td>OBE</td>
<td>OBUF</td>
<td>—</td>
<td>—</td>
<td>OB3E</td>
<td>OB2E</td>
<td>OB1E</td>
<td>OB0E</td>
<td>60</td>
</tr>
<tr>
<td>PADCFG1&lt;br&gt;(2)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>PMPTTL</td>
<td>56</td>
</tr>
</tbody>
</table>

**Legend:** — unimplemented, read as ‘0’. Shaded cells are not used during PMP operation.

**Note 1:** The PMADDRH/PMOUT1H and PMADDRL/PMOUT1L register pairs share the physical registers and addresses, but have different functions determined by the module’s operating mode.

**Note 2:** Configuration SFR, overlaps with default SFR at this address; available only when WDTCN<4> = 1.
## 12.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

### REGISTER 12-1:  T0CON: TIMER0 CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>TMR0ON</td>
<td>Timer0 On/Off Control bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Enables Timer0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Stops Timer0</td>
</tr>
<tr>
<td>6</td>
<td>T08BIT</td>
<td>Timer0 8-Bit/16-Bit Control bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Timer0 is configured as an 8-bit timer/counter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Timer0 is configured as a 16-bit timer/counter</td>
</tr>
<tr>
<td>5</td>
<td>T0CS</td>
<td>Timer0 Clock Source Select bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Transition on T0CKI pin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Internal instruction cycle clock (CLKO)</td>
</tr>
<tr>
<td>4</td>
<td>T0SE</td>
<td>Timer0 Source Edge Select bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Increment on high-to-low transition on T0CKI pin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Increment on low-to-high transition on T0CKI pin</td>
</tr>
<tr>
<td>3</td>
<td>PSA</td>
<td>Timer0 Prescaler Assignment bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Timer0 prescaler is not assigned. Timer0 clock input bypasses prescaler.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.</td>
</tr>
<tr>
<td>2-0</td>
<td>T0PS2:T0PS0</td>
<td>Timer0 Prescaler Select bits</td>
</tr>
<tr>
<td>111</td>
<td></td>
<td>1:256 Prescale value</td>
</tr>
<tr>
<td>110</td>
<td></td>
<td>1:128 Prescale value</td>
</tr>
<tr>
<td>101</td>
<td></td>
<td>1:64 Prescale value</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>1:32 Prescale value</td>
</tr>
<tr>
<td>011</td>
<td></td>
<td>1:16 Prescale value</td>
</tr>
<tr>
<td>010</td>
<td></td>
<td>1:8 Prescale value</td>
</tr>
<tr>
<td>001</td>
<td></td>
<td>1:4 Prescale value</td>
</tr>
<tr>
<td>000</td>
<td></td>
<td>1:2 Prescale value</td>
</tr>
</tbody>
</table>

The T0CON register (Register 12-1) controls all aspects of the module's operation, including the prescaler selection. It is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 12-1. Figure 12-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.
12.1 Timer0 Operation

Timer0 can operate as either a timer or a counter. The mode is selected with the T0CS bit (T0CON<5>). In Timer mode (T0CS = 0), the module increments on every clock by default unless a different prescaler value is selected (see Section 12.3 “Prescaler”). If the TMR0 register is written to, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

The Counter mode is selected by setting the T0CS bit (= 1). In this mode, Timer0 increments either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (T0CON<4>); clearing this bit selects the rising edge. Restrictions on the external clock input are discussed below.

An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (Tosc). There is a delay between synchronization and the onset of incrementing the timer/counter.

12.2 Timer0 Reads and Writes in 16-Bit Mode

TMR0H is not the actual high byte of Timer0 in 16-bit mode. It is actually a buffered version of the real high byte of Timer0 which is not directly readable nor writable (refer to Figure 12-2). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte were valid, due to a rollover between successive reads of the high and low byte.

Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

![FIGURE 12-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)](image1)

Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

![FIGURE 12-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)](image2)

Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.
12.3 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not directly readable or writable. Its value is set by the PSA and T0PS2:T0PS0 bits (T0CON<3:0>) which determine the prescaler assignment and prescale ratio.

Clearing the PSA bit assigns the prescaler to the Timer0 module. When it is assigned, prescale values from 1:2 through 1:256 in power-of-2 increments are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., CLRF TMR0, MOVWF TMR0, BSF TMR0, etc.) clear the prescaler count.

Note: Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count but will not change the prescaler assignment.

12.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control and can be changed “on-the-fly” during program execution.

12.4 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or from FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF flag bit. The interrupt can be masked by clearing the TMROIE bit (INTCON<5>). Before re-enabling the interrupt, the TMROIF bit must be cleared in software by the Interrupt Service Routine.

Since Timer0 is shut down in Sleep mode, the TMR0 interrupt cannot awaken the processor from Sleep.

TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER0

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMR0L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>TMR0H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INTOIE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INTOIF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>T0CON</td>
<td>TMR0ON</td>
<td>T08BIT</td>
<td>T0CS</td>
<td>T0SE</td>
<td>PSA</td>
<td>T0PS2</td>
<td>T0PS1</td>
<td>T0PS0</td>
<td>56</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7&lt;1&gt;</td>
<td>TRISA6&lt;1&gt;</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>58</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as '0'. Shaded cells are not used by Timer0.

Note 1: These bits are only available in select oscillator modes (FOSC2 Configuration bit = 0); otherwise, they are unimplemented.
13.0 TIMER1 MODULE

The Timer1 timer/counter module incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR1H and TMR1L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt on overflow
- Reset on ECCPx Special Event Trigger
- Device clock status flag (T1RUN)

A simplified block diagram of the Timer1 module is shown in Figure 13-1. A block diagram of the module’s operation in Read/Write mode is shown in Figure 13-2.

The module incorporates its own low-power oscillator to provide an additional clocking option. The Timer1 oscillator can also be used as a low-power clock source for the microcontroller in power-managed operation.

Timer1 can also be used to provide Real-Time Clock (RTC) functionality to applications with only a minimal addition of external components and code overhead.

Timer1 is controlled through the T1CON Control register (Register 13-1). It also contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

REGISTER 13-1: T1CON: TIMER1 CONTROL REGISTER(1)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD16</td>
<td>T1RUN</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>T1SYNC</td>
<td>TMR1CS</td>
<td>TMR1ON</td>
</tr>
</tbody>
</table>

Legend:

R = Readable bit  
W = Writable bit  
U = Unimplemented bit, read as ‘0’

-7 = Value at POR  
‘1’ = Bit is set  
‘0’ = Bit is cleared  
x = Bit is unknown

<table>
<thead>
<tr>
<th>bit 7</th>
<th>RD16: 16-Bit Read/Write Mode Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enables register read/write of Timer1 in one 16-bit operation</td>
</tr>
<tr>
<td>0</td>
<td>Enables register read/write of Timer1 in two 8-bit operations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6</th>
<th>T1RUN: Timer1 System Clock Status bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Device clock is derived from Timer1 oscillator</td>
</tr>
<tr>
<td>0</td>
<td>Device clock is derived from another source</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5-4</th>
<th>T1CKPS1:T1CKPS0: Timer1 Input Clock Prescale Select bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>1:8 Prescale value</td>
</tr>
<tr>
<td>10</td>
<td>1:4 Prescale value</td>
</tr>
<tr>
<td>01</td>
<td>1:2 Prescale value</td>
</tr>
<tr>
<td>00</td>
<td>1:1 Prescale value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 3</th>
<th>T1OSCEN: Timer1 Oscillator Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Timer1 oscillator is enabled</td>
</tr>
<tr>
<td>0</td>
<td>Timer1 oscillator is shut off</td>
</tr>
</tbody>
</table>

The oscillator inverter and feedback resistor are turned off to eliminate power drain.

<table>
<thead>
<tr>
<th>bit 2</th>
<th>T1SYNC: Timer1 External Clock Input Synchronization Select bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>When TMR1CS = 1:</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Do not synchronize external clock input</td>
</tr>
<tr>
<td>0</td>
<td>Synchronize external clock input</td>
</tr>
<tr>
<td>When TMR1CS = 0:</td>
<td></td>
</tr>
<tr>
<td>This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1</th>
<th>TMR1CS: Timer1 Clock Source Select bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>External clock from pin RC0/T1OSO/T13CKI (on the rising edge)</td>
</tr>
<tr>
<td>0</td>
<td>Internal clock (FOSC/4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 0</th>
<th>TMR1ON: Timer1 On bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enables Timer1</td>
</tr>
<tr>
<td>0</td>
<td>Stops Timer1</td>
</tr>
</tbody>
</table>

Note 1:  Default (legacy) SFR at this address, available when WDTCN<4> = 0.
13.1 Timer1 Operation

Timer1 can operate in one of these modes:
• Timer
• Synchronous Counter
• Asynchronous Counter

The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>). When TMR1CS is cleared (= 0), Timer1 increments on every internal instruction cycle (Fosc/4). When the bit is set, Timer1 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

When Timer1 is enabled, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

Note 1: When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

FIGURE 13-1: TIMER1 BLOCK DIAGRAM

FIGURE 13-2: TIMER1 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)
13.2 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 13-2). When the RD16 control bit, T1CON<7>, is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 High Byte Buffer register. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, has become invalid due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H Buffer register. The Timer1 high byte is updated with the contents of TMR1H when a write occurs to TMR1L. This allows a user to write all 16 bits to both the high and low bytes of Timer1 at once.

The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

13.3 Timer1 Oscillator

An on-chip crystal oscillator circuit is incorporated between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting the Timer1 Oscillator Enable bit, T1OSCEN (T1CON<3>). The oscillator is a low-power circuit rated for 32 kHz crystals. It will continue to run during all power-managed modes. The circuit for a typical LP oscillator is shown in Figure 13-3. Table 13-1 shows the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

FIGURE 13-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR

![Diagram of Timer1 LP Oscillator]

TABLE 13-1: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR

<table>
<thead>
<tr>
<th>Oscillator Type</th>
<th>Freq.</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>32 kHz</td>
<td>27 pF(1)</td>
<td>27 pF(1)</td>
</tr>
</tbody>
</table>

Note 1: Microchip suggests these values as a starting point in validating the oscillator circuit.

2: Higher capacitance increases the stability of the oscillator but also increases the start-up time.

3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

4: Capacitor values are for design guidance only.

13.3.1 USING TIMER1 AS A CLOCK SOURCE

The Timer1 oscillator is also available as a clock source in power-managed modes. By setting the clock select bits, SCS1:SCS0 (OSCCON<1:0>), to '01', the device switches to SEC_RUN mode; both the CPU and peripherals are clocked from the Timer1 oscillator. If the IDLEN bit (OSCCON<7>) is cleared and a SLEEP instruction is executed, the device enters SEC_IDLE mode. Additional details are available in Section 3.0 “Power-Managed Modes”.

Whenever the Timer1 oscillator is providing the clock source, the Timer1 system clock status flag, T1RUN (T1CON<6>), is set. This can be used to determine the controller’s current clocking mode. It can also indicate the clock source being currently used by the Fail-Safe Clock Monitor. If the Clock Monitor is enabled and the Timer1 oscillator fails while providing the clock, polling the T1RUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

13.3.2 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

The Timer1 oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity.

The oscillator circuit, shown in Figure 13-3, should be located as close as possible to the microcontroller. There should be no circuits passing within the oscillator circuit boundaries other than VSS or VDD.
If a high-speed circuit must be located near the oscillator (such as the ECCP1 pin in Output Compare or PWM mode, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as shown in Figure 13-4, may be helpful when used on a single-sided PCB or in addition to a ground plane.

**FIGURE 13-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING**

13.4 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The Timer1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled or disabled by setting or clearing the Timer1 Interrupt Enable bit, TMR1IE (PIE1<0>).

13.5 Resetting Timer1 Using the ECCPx Special Event Trigger

If ECCP1 or ECCP2 is configured to use Timer1 and to generate a Special Event Trigger in Compare mode (CCPxM3:CCPxM0 = 1011), this signal will reset Timer3. The trigger from ECCP2 will also start an A/D conversion if the A/D module is enabled (see Section 18.2.1 “Special Event Trigger” for more information).

The module must be configured as either a timer or a synchronous counter to take advantage of this feature. When used this way, the CCPRxH:CCPRxl register pair effectively becomes a period register for Timer1.

If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a Special Event Trigger, the write operation will take precedence.

**Note:** The Special Event Triggers from the ECCPx module will not set the TMR1IF interrupt flag bit (PIR1<0>).

13.6 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in Section 13.3 “Timer1 Oscillator”) gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or supercapacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, RTCisr, shown in Example 13-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow triggers the interrupt and calls the routine which increments the seconds counter by one. Additional counters for minutes and hours are incremented as the previous counter overflows.

Since the register pair is 16 bits wide, counting up to overflow the register directly from a 32.768 kHz clock would take 2 seconds. To force the overflow at the required one-second intervals, it is necessary to preload it. The simplest method is to set the MSb of TMR1H with a BSF instruction. Note that the TMR1L register is never preloaded or altered; doing so may introduce cumulative error over many cycles.

For this method to be accurate, Timer1 must operate in Asynchronous mode and the Timer1 overflow interrupt must be enabled (PIE1<0> = 1), as shown in the routine, RTCinit. The Timer1 oscillator must also be enabled and running at all times.
13.7 Considerations in Asynchronous Counter Mode

Following a Timer1 interrupt and an update to the TMR1 registers, the Timer1 module uses a falling edge on its clock source to trigger the next register update on the rising edge. If the update is completed after the clock input has fallen, the next rising edge will not be counted.

If the application can reliably update TMR1 before the timer input goes low, no additional action is needed. Otherwise, an adjusted update can be performed following a later Timer1 increment. This can be done by monitoring TMR1L within the interrupt routine until it increments, and then updating the TMR1H:TMR1L register pair while the clock is low, or one-half of the period of the clock source. Assuming that Timer1 is being used as a Real-Time Clock, the clock source is a 32.768 kHz crystal oscillator. In this case, one-half period of the clock is 15.25 μs.

The Real-Time Clock application code in Example 13-1 shows a typical ISR for Timer1, as well as the optional code required if the update cannot be done reliably within the required interval.

**EXAMPLE 13-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE**

```
RTCinit
MOVLW 80h ; Preload TMR1 register pair
MOVWF TMR1H ; for 1 second overflow
CLRF TMR1L
MOVLW b'00001111' ; Configure for external clock,
MOVWF T1CON ; Asynchronous operation, external oscillator
CLRF secs ; Initialize timekeeping registers
CLRF mins ;
MOVLW .12
MOVF W hours
BSF PIE1, TMR1IE ; Enable Timer1 interrupt
RETURN

RTCisr
; Insert the next 4 lines of code when TMR1
; can not be reliably updated before clock pulse goes low
BTFSC TMR1L, 0 ; wait for TMR1L to become clear
BRA $-2 ; (may already be clear)
BTFSS TMR1L, 0 ; wait for TMR1L to become set
BRA $-2 ; TMR1 has just incremented
; If TMR1 update can be completed before clock pulse goes low
; Start ISR here
BSF TMR1H, 7 ; Preload for 1 sec overflow
BCF PIR1, TMR1IF ; Clear interrupt flag
INCF secs, F ; Increment seconds
MOVLW .59 ; 60 seconds elapsed?
CPFSGT secs
RETURN ; No, done
CLRF secs ; Clear seconds
INCF mins, F ; Increment minutes
MOVLW .59 ; 60 minutes elapsed?
CPFSGT mins
RETURN ; No, done
CLRF mins ; clear minutes
INCF hours, F ; Increment hours
MOVLW .23 ; 24 hours elapsed?
CPFSGT hours
RETURN ; No, done
CLRF hours ; Reset hours
RETURN ; Done
```
## TABLE 13-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

<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/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>TMR1L&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>Timer1 Register Low Byte</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR1H&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>Timer1 Register High Byte</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1CON&lt;sup&gt;(3)&lt;/sup&gt;</td>
<td>RD16</td>
<td>T1RUN</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>T1SYNC</td>
<td>TMR1CS</td>
<td>TMR1ON</td>
<td>56</td>
</tr>
</tbody>
</table>

**Legend:** Shaded cells are not used by the Timer1 module.

**Note 1:** Default (legacy) SFR at this address, available when WDTCN<4> = 0.
14.0 TIMER2 MODULE

The Timer2 module incorporates the following features:

- 8-Bit Timer and Period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler
  (1:1, 1:4 and 1:16)
- Software programmable postscaler
  (1:1 through 1:16)
- Interrupt on TMR2 to PR2 match
- Optional use as the shift clock for the MSSP modules

The module is controlled through the T2CON register (Register 14-1) which enables or disables the timer and configures the prescaler and postscaler. Timer2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

A simplified block diagram of the module is shown in Figure 14-1.

14.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (FOSC/4). A 4-bit counter/prescaler on the clock input gives direct input, divide-by-4 and divide-by-16 prescale options. These are selected by the prescaler control bits, T2CKPS1:T2CKPS0 (T2CON<1:0>). The value of TMR2 is compared to that of the Period register, PR2, on each clock cycle. When the two values match, the comparator generates a match signal as the timer output. This signal also resets the value of TMR2 to 00h on the next cycle and drives the output counter/postscaler (see Section 14.2 “Timer2 Interrupt”).

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, while the PR2 register initializes at FFh. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (Power-on Reset, MCLR Reset, Watchdog Timer Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

REGISTER 14-1: T2CON: TIMER2 CONTROL REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>T2OUTPS3</td>
<td>T2OUTPS2</td>
<td>T2OUTPS1</td>
<td>T2OUTPS0</td>
<td>TMR2ON</td>
<td>T2CKPS1</td>
<td>T2CKPS0</td>
</tr>
</tbody>
</table>

bit 7: Unimplemented: Read as ‘0’

bit 6-3

T2OUTPS3:T2OUTPS0: Timer2 Output Postscale Select bits

| 0000 | 1:1 Postscale |
| 0001 | 1:2 Postscale |
| 0011 | 1:4 Postscale |
| 0111 | 1:8 Postscale |
| 1111 | 1:16 Postscale |

bit 2

TMR2ON: Timer2 On bit

1 = Timer2 is on
0 = Timer2 is off

bit 1-0

T2CKPS1:T2CKPS0: Timer2 Clock Prescale Select bits

| 00 | Prescaler is 1 |
| 01 | Prescaler is 4 |
| 1x | Prescaler is 16 |

Legend:
R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’
- = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown
14.2 Timer2 Interrupt

Timer2 can also generate an optional device interrupt. The Timer2 output signal (TMR2 to PR2 match) provides the input for the 4-bit output counter/postscaler. This counter generates the TMR2 match interrupt flag which is latched in TMR2IF (PIR1<1>). The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE (PIE1<1>).

A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS3:T2OUTPS0 (T2CON<6:3>).

14.3 Timer2 Output

The unscaled output of TMR2 is available primarily to the ECCPx/CCPx modules, where it is used as a time base for operations in PWM mode.

Timer2 can be optionally used as the shift clock source for the MSSP modules operating in SPI mode. Additional information is provided in Section 19.0 “Master Synchronous Serial Port (MSSP) Module”.

![Timer2 Block Diagram](image)

**TABLE 14-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

| Name          | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page:
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE</td>
<td>GIEH</td>
<td>PEIE</td>
<td>GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>TMR2(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>56</td>
</tr>
<tr>
<td>T2CON</td>
<td>—</td>
<td>T2OUTPS3</td>
<td>T2OUTPS2</td>
<td>T2OUTPS1</td>
<td>T2OUTPS0</td>
<td>TMR2ON</td>
<td>T2CKPS1</td>
<td>T2CKPS0</td>
<td>56</td>
</tr>
<tr>
<td>PR2(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>56</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the Timer2 module.

**Note 1:** Default (legacy) SFR at this address, available when WDTCN<4> = 0.
15.0 TIMER3 MODULE

The Timer3 timer/counter module incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR3H and TMR3L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Module Reset on ECCPx Special Event Trigger

A simplified block diagram of the Timer3 module is shown in Figure 15-1. A block diagram of the module’s operation in Read/Write mode is shown in Figure 15-2.

The Timer3 module is controlled through the T3CON register (Register 15-1). It also selects the clock source options for the CCP and ECCP modules; see Section 17.1.1 “CCP Modules and Timer Resources” for more information.

REGISTER 15-1: T3CON: TIMER3 CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD16</td>
<td>T3CCP2</td>
<td>T3CKPS1</td>
<td>T3CKPS0</td>
<td>T3CCP1</td>
<td>T3SYNC</td>
<td>TMR3CS</td>
<td>TMR3ON</td>
</tr>
</tbody>
</table>

Legend:

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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>RD16: 16-Bit Read/Write Mode Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enables register read/write of Timer3 in one 16-bit operation</td>
</tr>
<tr>
<td>0</td>
<td>Enables register read/write of Timer3 in two 8-bit operations</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6,3</th>
<th>T3CCP2:T3CCP1: Timer3 and Timer1 to ECCPx/CCPx Enable bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Timer3 and Timer4 are the clock sources for all ECCPx/CCPx modules</td>
</tr>
<tr>
<td>10</td>
<td>Timer3 and Timer4 are the clock sources for ECCP3, CCP4 and CCP5; Timer1 and Timer2 are the clock sources for ECCP1 and ECCP2</td>
</tr>
<tr>
<td>01</td>
<td>Timer3 and Timer4 are the clock sources for ECCP2, ECCP3, CCP4 and CCP5; Timer1 and Timer2 are the clock sources for ECCP1</td>
</tr>
<tr>
<td>00</td>
<td>Timer3 and Timer4 are the clock sources for all ECCPx modules</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5-4</th>
<th>T3CKPS1:T3CKPS0: Timer3 Input Clock Prescale Select bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>1:8 Prescale value</td>
</tr>
<tr>
<td>10</td>
<td>1:4 Prescale value</td>
</tr>
<tr>
<td>01</td>
<td>1:2 Prescale value</td>
</tr>
<tr>
<td>00</td>
<td>1:1 Prescale value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 2</th>
<th>T3SYNC: Timer3 External Clock Input Synchronization Control bit (Not usable if the device clock comes from Timer1/Timer3.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>When TMR3CS = 1:</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Do not synchronize external clock input</td>
</tr>
<tr>
<td>0</td>
<td>Synchronize external clock input</td>
</tr>
<tr>
<td>When TMR3CS = 0:</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1</th>
<th>TMR3CS: Timer3 Clock Source Select bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>External clock input from Timer1 oscillator or T13CKI (on the rising edge after the first falling edge)</td>
</tr>
<tr>
<td>0</td>
<td>Internal clock (Fosc/4)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 0</th>
<th>TMR3ON: Timer3 On bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enables Timer3</td>
</tr>
<tr>
<td>0</td>
<td>Stops Timer3</td>
</tr>
</tbody>
</table>
15.1 Timer3 Operation

Timer3 can operate in one of three modes:
- Timer
- Synchronous Counter
- Asynchronous Counter

The operating mode is determined by the clock select bit, TMR3CS (T3CON<1>). When TMR3CS is cleared (= 0), Timer3 increments on every internal instruction cycle (Fosc/4). When the bit is set, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator, if enabled.

As with Timer1, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs when the Timer1 oscillator is enabled. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

**FIGURE 15-1: TIMER3 BLOCK DIAGRAM**

**FIGURE 15-2: TIMER3 BLOCK DIAGRAM (16-BIT READ/WRITE MODE)**

Note 1: When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.
15.2 Timer3 16-Bit Read/Write Mode

Timer3 can be configured for 16-bit reads and writes (see Figure 15-2). When the RD16 control bit (T3CON<7>) is set, the address for TMR3H is mapped to a buffer register for the high byte of Timer3. A read from TMR3L will load the contents of the high byte of Timer3 into the Timer3 High Byte Buffer register. This provides the user with the ability to accurately read all 16 bits of Timer3 without having to determine whether a read of the high byte, followed by a read of the low byte, has become invalid due to a rollover between reads.

A write to the high byte of Timer3 must also take place through the TMR3H Buffer register. The Timer3 high byte is updated with the contents of TMR3H when a write occurs to TMR3L. This allows a user to write all 16 bits to both the high and low bytes of Timer3 at once. The high byte of Timer3 is not directly readable or writable in this mode. All reads and writes must take place through the Timer3 High Byte Buffer register.

Writes to TMR3H do not clear the Timer3 prescaler. The prescaler is only cleared on writes to TMR3L.

15.3 Using the Timer1 Oscillator as the Timer3 Clock Source

The Timer1 internal oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN (T1CON<3>) bit. To use it as the Timer3 clock source, the TMR1CS bit must also be set. As previously noted, this also configures Timer3 to increment on every rising edge of the oscillator source.

The Timer1 oscillator is described in Section 13.0 “Timer1 Module”.

15.4 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and overflows to 0000h. The Timer3 interrupt, if enabled, is generated on overflow and is latched in interrupt flag bit, TMR3IF (PIR2<1>). This interrupt can be enabled or disabled by setting or clearing the Timer3 Interrupt Enable bit, TMR3IE (PIE2<1>).

15.5 Resetting Timer3 Using the ECCPx Special Event Trigger

If ECCP1 or ECCP2 is configured to use Timer3 and to generate a Special Event Trigger in Compare mode (CCPxM3:CCPxM0 = 1011), this signal will reset Timer3. The trigger from ECCP2 will also start an A/D conversion if the A/D module is enabled (see Section 18.2.1 “Special Event Trigger” for more information).

The module must be configured as either a timer or synchronous counter to take advantage of this feature. When used this way, the CCPxH:CCPxL register pair effectively becomes a period register for Timer3. If Timer3 is running in Asynchronous Counter mode, the Reset operation may not work.

In the event that a write to Timer3 coincides with a Special Event Trigger from an ECCPx module, the write will take precedence.

Note: The Special Event Triggers from the ECCPx module will not set the TMR3IF interrupt flag bit (PIR1<0>).

### TABLE 15-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER

<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/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSCFIF</td>
<td>CM2IF</td>
<td>CM1IF</td>
<td>—</td>
<td>BCL1IF</td>
<td>LVDIF</td>
<td>TMR3IF</td>
<td>CCP2IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSCFIE</td>
<td>CM2IE</td>
<td>CM1IE</td>
<td>—</td>
<td>BCL1IE</td>
<td>LVDIE</td>
<td>TMR3IE</td>
<td>CCP2IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR2</td>
<td>OSCFIP</td>
<td>CM2IP</td>
<td>CM1IP</td>
<td>—</td>
<td>BCL1IP</td>
<td>LVDIP</td>
<td>TMR3IP</td>
<td>CCP2IP</td>
<td>58</td>
</tr>
<tr>
<td>TMR3L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer3 Register Low Byte</td>
<td>59</td>
</tr>
<tr>
<td>TMR3H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer3 Register High Byte</td>
<td>59</td>
</tr>
<tr>
<td>T1CON(1)</td>
<td>RD16</td>
<td>T1RUN</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>T1SYNC</td>
<td>TMR1CS</td>
<td>TMR1ON</td>
<td>56</td>
</tr>
<tr>
<td>T3CON</td>
<td>RD16</td>
<td>T3CP2</td>
<td>T3CKPS1</td>
<td>T3CKPS0</td>
<td>T3CP1</td>
<td>T3SYNC</td>
<td>TMR3CS</td>
<td>TMR3ON</td>
<td>59</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the Timer3 module.

Note 1: Default (legacy) SFR at this address, available when WDTCN<4> = 0.
16.0 TIMER4 MODULE

The Timer4 timer module has the following features:

- 8-bit timer register (TMR4)
- 8-bit period register (PR4)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMR4 match of PR4

Timer4 has a control register shown in Register 16-1. Timer4 can be shut off by clearing control bit, TMR4ON (T4CON<2>), to minimize power consumption. The prescaler and postscaler selection of Timer4 are also controlled by this register. Figure 16-1 is a simplified block diagram of the Timer4 module.

16.1 Timer4 Operation

Timer4 can be used as the PWM time base for the PWM mode of the ECCPxx/CCPx modules. The TMR4 register is readable and writable and is cleared on any device Reset. The input clock (Fosc/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits T4CKPS1:T4CKPS0 (T4CON<1:0>). The match output of TMR4 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR4 interrupt, latched in flag bit, TMR4IF (PIR3<3>).

The prescaler and postscaler counters are cleared when any of the following occurs:

- a write to the TMR4 register
- a write to the T4CON register
- any device Reset (Power-on Reset, MCLR Reset, Watchdog Timer Reset or Brown-out Reset)

TMR4 is not cleared when T4CON is written.

REGISTER 16-1: T4CON: TIMER4 CONTROL REGISTER

<table>
<thead>
<tr>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T4OUTPS3</td>
<td>T4OUTPS2</td>
<td>T4OUTPS1</td>
<td>T4OUTPS0</td>
<td>TMR4ON</td>
<td>T4CKPS1</td>
<td>T4CKPS0</td>
</tr>
<tr>
<td>bit 7</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'
- -n = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- x = Bit is unknown

bit 7 Unimplemented: Read as '0'

bit 6-3 T4OUTPS3:T4OUTPS0: Timer4 Output Postscale Select bits

0000 = 1:1 Postscale
0001 = 1:2 Postscale

bit 2 TMR4ON: Timer4 On bit

1 = Timer4 is on
0 = Timer4 is off

bit 1-0 T4CKPS1:T4CKPS0: Timer4 Clock Prescale Select bits

00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16
16.2 Timer4 Interrupt
The Timer4 module has an 8-bit period register, PR4, which is both readable and writable. Timer4 increments from 00h until it matches PR4 and then resets to 00h on the next increment cycle. The PR4 register is initialized to FFh upon Reset.

16.3 Output of TMR4
The output of TMR4 (before the postscaler) is used only as a PWM time base for the ECCPx/CCPx modules. It is not used as a baud rate clock for the MSSP modules as is the Timer2 output.

FIGURE 16-1: TIMER4 BLOCK DIAGRAM

TABLE 16-1: REGISTERS ASSOCIATED WITH TIMER4 AS A TIMER/COUNTER

<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>GIEH</td>
<td>PEIE</td>
<td>GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td></td>
</tr>
<tr>
<td>PIR3</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td></td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMR4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td></td>
</tr>
<tr>
<td>TMR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timer4 Register</td>
</tr>
<tr>
<td>T4CON</td>
<td>—</td>
<td>T4OUTPS3</td>
<td>T4OUTPS2</td>
<td>T4OUTPS1</td>
<td>T4OUTPS0</td>
<td>TMR4ON</td>
<td>T4CKPS1</td>
<td>T4CKPS0</td>
<td></td>
</tr>
<tr>
<td>PR4</td>
<td>Timer4 Period Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the Timer4 module.
17.0 CAPTURE/COMPARE/PWM (CCP) MODULES

Members of the PIC18F87J11 family of devices all have a total of five CCP (Capture/Compare/PWM) modules. Two of these (CCP4 and CCP5) implement standard Capture, Compare and Pulse-Width Modulation (PWM) modes and are discussed in this section. The other three modules (ECCP1, ECCP2, ECCP3) implement standard Capture and Compare modes, as well as Enhanced PWM modes. These are discussed in Section 18.0 “Enhanced Capture/Compare/PWM (ECCP) Module”.

Each CCP/ECCP module contains a 16-bit register which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register. For the sake of clarity, all CCP module operation in the following sections is described with respect to CCP4, but is equally applicable to CCP5.

REGISTER 17-1: CCPxCON: CCPx CONTROL REGISTER (CCP4 MODULE, CCP5 MODULE)

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
<td>—</td>
<td>CCPxX</td>
<td>CCPxY</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’
- = Value at POR
’1’ = Bit is set
’0’ = Bit is cleared
x = Bit is unknown

bit 7-6 Unimplemented: Read as ‘0’
bit 5-4 CCPx<X:Y>: PWM Duty Cycle bit 1 and bit 0 for CCPx Module
Capture mode:
Unused.
Compare mode:
Unused.
PWM mode:
These bits are the two Least Significant bits (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight Most Significant bits (DCx9:DCx2) of the duty cycle are found in CCPRxL.

bit 3-0 CCPxM3:CCPxM0: CCPx Module Mode Select bits
0000 = Capture/Compare/PWM disabled (resets CCPx module)
0001 = Reserved
0010 = Compare mode, toggle output on match (CCPxIF bit is set)
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 CCPx pin low; on compare match, force CCPx pin high (CCPxIF bit is set)
1001 = Compare mode: initialize CCPx pin high; on compare match, force CCPx pin low (CCPxIF bit is set)
1010 = Compare mode: generate software interrupt on compare match (CCPxIF bit is set, CCPx pin reflects I/O state)
1011 = Compare mode: trigger special event, reset timer, start A/D conversion on CCPx match (CCPxIF bit is set)(f)
11xx = PWM mode

Note: Throughout this section and Section 18.0 “Enhanced Capture/Compare/PWM (ECCP) Module”, references to register and bit names that may be associated with a specific CCP module are referred to generically by the use of ‘x’ or ‘y’ in place of the specific module number. Thus, “CCPxCON” might refer to the control register for ECCP1, ECCP2, ECCP3, CCP4 or CCP5.


17.1 CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (generically, CCPxCON) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte). All registers are both readable and writable.

17.1.1 CCP MODULES AND TIMER RESOURCES

The ECCP/CCP modules utilize Timers 1, 2, 3 or 4, depending on the mode selected. Timer1 and Timer3 are available to modules in Capture or Compare modes, while Timer2 and Timer4 are available for modules in PWM mode.

### TABLE 17-1: CCP MODE – TIMER RESOURCE

<table>
<thead>
<tr>
<th>CCP Mode</th>
<th>Timer Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture</td>
<td>Timer1 or Timer3</td>
</tr>
<tr>
<td>Compare</td>
<td>Timer1 or Timer3</td>
</tr>
<tr>
<td>PWM</td>
<td>Timer2 or Timer4</td>
</tr>
</tbody>
</table>

17.1.2 OPEN-DRAIN OUTPUT OPTION

When operating in Output mode (i.e., in Compare or PWM modes), the drivers for the CCP pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor, and allows the output to communicate with external circuits without the need for additional level shifters. For more information, see Section 10.1.4 “Open-Drain Outputs”.

The open-drain output option is controlled by the bits in the ODCON1 register. Setting the appropriate bit configures the pin for the corresponding module for open-drain operation. The ODCON1 memory shares the same address space as TMR1H. The ODCON1 register can be accessed by setting the ADSHR bit in the WDTCON register (WDTCON<4>).

FIGURE 17-1: ECCPx/CCPx AND TIMER INTERCONNECT CONFIGURATIONS

Timer1 is used for all Capture and Compare operations for all CCP modules. Timer2 is used for PWM operations for all CCP modules. Modules may share either timer resource as a common time base. Timer3 and Timer4 are not available.

Timer1 and Timer2 are used for Capture and Compare or PWM operations for ECCP1 only (depending on selected mode).

All other modules use either Timer3 or Timer4. Modules may share either timer resource as a common time base if they are in Capture/Compare or PWM modes.

The other modules use either Timer3 or Timer4. Modules may share either timer resource as a common time base if they are in Capture/Compare or PWM modes.
17.2 Capture Mode

In Capture mode, the CCPRxH:CCPRxL register pair captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on the corresponding CCP pin. An event is defined as one of the following:

- every falling edge
- every rising edge
- every 4th rising edge
- every 16th rising edge

The event is selected by the mode select bits, CCPxM3:CCPxM0 (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit, CCPxIF, is set; it must be cleared in software. If another capture occurs before the value in register CCPRx is read, the old captured value is overwritten by the new captured value.

17.2.1 CCP PIN CONFIGURATION

In Capture mode, the appropriate CCP pin should be configured as an input by setting the corresponding TRIS direction bit.

**Note:** If RG4/CCP5 is configured as an output, a write to the port can cause a capture condition.

17.2.2 TIMER1/TIMER3 MODE SELECTION

The timers that are to be used with the capture feature (Timer1 and/or Timer3) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation will not work. The timer to be used with each CCP module is selected in the T3CON register (see Section 17.1.1 “CCP Modules and Timer Resources”).

17.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE interrupt enable bit clear to avoid false interrupts. The interrupt flag bit, CCPxIF, should also be cleared following any such change in operating mode.

17.2.4 CCP PRESCALER

There are four prescaler settings in Capture mode. They are specified as part of the operating mode selected by the mode select bits (CCPxM3:CCPxM0). Whenever the CCP module is turned off or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared; therefore, the first capture may be from a non-zero prescaler. Example 17-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

**EXAMPLE 17-1: CHANGING BETWEEN CAPTURE PRESCALERS (CCP5 SHOWN)**

```plaintext
CLRF CCP5CON ; Turn CCP module off
MOVLW NEW_CAPT_PS ; Load WREG with the ; new prescaler mode ; value and CCP ON
MOVWF CCP5CON ; Load CCP5CON with ; this value
```

---

**FIGURE 17-2: CAPTURE MODE OPERATION BLOCK DIAGRAM**
17.3 Compare Mode

In Compare mode, the 16-bit CCPRx register value is constantly compared against either the TMR1 or TMR3 register pair value. When a match occurs, the CCP pin can be:

- driven high
- driven low
- toggled (high-to-low or low-to-high)
- remains unchanged (that is, reflects the state of the I/O latch)

The action on the pin is based on the value of the mode select bits (CCPxM3:CCPxM0). At the same time, the interrupt flag bit, CCPxIF, is set.

17.3.1 CCP PIN CONFIGURATION

The user must configure the CCP pin as an output by clearing the appropriate TRIS bit.

17.3.2 TIMER1/TIMER3 MODE SELECTION

Timer1 and/or Timer3 must be running in Timer mode or Synchronized Counter mode if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

17.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCPxM3:CCPxM0 = 1010), the corresponding CCP pin is not affected. Only a CCP interrupt is generated, if enabled, and the CCPxIE bit is set.
### TABLE 17-2: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

<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/GIEH</td>
<td>PEI/PEIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>RCON</td>
<td>IPEN</td>
<td>—</td>
<td>CM</td>
<td>RI</td>
<td>TO</td>
<td>PD</td>
<td>POR</td>
<td>BOR</td>
<td>56</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSCF1F</td>
<td>CM2IF</td>
<td>CM1IF</td>
<td>—</td>
<td>BCL1IF</td>
<td>LVDIF</td>
<td>TMR3IF</td>
<td>CCP2IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSCFIE</td>
<td>CM2IE</td>
<td>CM1IE</td>
<td>—</td>
<td>BCL1IE</td>
<td>LVIDE</td>
<td>TMR3IE</td>
<td>CCP2IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR2</td>
<td>OSCFIP</td>
<td>CM2IP</td>
<td>CM1IP</td>
<td>—</td>
<td>BCL1IP</td>
<td>LVIDP</td>
<td>TMR3IP</td>
<td>CCP2IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR3</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMR4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>58</td>
</tr>
<tr>
<td>TRISG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRISG4</td>
<td>TRISG3</td>
<td>TRISG2</td>
<td>TRISG1</td>
<td>TRISG0 58</td>
</tr>
<tr>
<td>TMR1L&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>Timer1 Register Low Byte</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR1H&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>Timer1 Register High Byte</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODCON&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP5OD</td>
<td>CCP4OD</td>
<td>ECCP3OD</td>
<td>ECCP2OD</td>
<td>ECCP1OD</td>
<td>56</td>
</tr>
<tr>
<td>T1CON&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>RD16</td>
<td>T1RUN</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>T1SYNC</td>
<td>TMR1CS</td>
<td>TMR1ON</td>
<td>56</td>
</tr>
<tr>
<td>TMR3H</td>
<td>Timer3 Register High Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR3L</td>
<td>Timer3 Register Low Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3CON</td>
<td>RD16</td>
<td>T3CCP2</td>
<td>T3CKPS1</td>
<td>T3CKPS0</td>
<td>T3CCP1</td>
<td>T3SYNC</td>
<td>TMR3CS</td>
<td>TMR3ON</td>
<td>59</td>
</tr>
<tr>
<td>CCPR4L</td>
<td>Capture/Compare/PWM Register 4 Low Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCPR4H</td>
<td>Capture/Compare/PWM Register 4 High Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCPR5L</td>
<td>Capture/Compare/PWM Register 5 Low Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCPR5H</td>
<td>Capture/Compare/PWM Register 5 High Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP4CON</td>
<td>—</td>
<td>—</td>
<td>DC4B1</td>
<td>DC4B0</td>
<td>CCP4M3</td>
<td>CCP4M2</td>
<td>CCP4M1</td>
<td>CCP4M0</td>
<td>59</td>
</tr>
<tr>
<td>CCP5CON</td>
<td>—</td>
<td>—</td>
<td>DC5B1</td>
<td>DC5B0</td>
<td>CCP5M3</td>
<td>CCP5M2</td>
<td>CCP5M1</td>
<td>CCP5M0</td>
<td>59</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used by Capture/Compare, Timer1 or Timer3.

**Note 1:** Default (legacy) SFR at this address, available when WDTCON<4> = 0.

**Note 2:** Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.
17.4 PWM Mode

In Pulse-Width Modulation (PWM) mode, the CCP pin produces up to a 10-bit resolution PWM output. Since the CCP4 and CCP5 pins are multiplexed with a PORTG data latch, the appropriate TRISG bit must be cleared to make the CCP4 or CCP5 pin an output.

**Note:** Clearing the CCP4CON or CCP5CON register will force the RG3 or RG4 output latch (depending on device configuration) to the default low level. This is not the PORTG I/O data latch.

Figure 17-4 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to set up a CCP module for PWM operation, see **Section 17.4.3 “Setup for PWM Operation”**.

**FIGURE 17-4: SIMPLIFIED PWM BLOCK DIAGRAM**

A PWM output (Figure 17-5) has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

**FIGURE 17-5: PWM OUTPUT**

17.4.1 PWM Period

The PWM period is specified by writing to the PR2 (PR4) register. The PWM period can be calculated using Equation 17-1:

**EQUATION 17-1:**

\[
\text{PWM Period} = \frac{\text{PR2} + 1}{4 \times \text{Tosc} \times \text{TMR2 Prescale Value}}
\]

PWM frequency is defined as 1/[PWM period].

When TMR2 (TMR4) is equal to PR2 (PR4), the following three events occur on the next increment cycle:

- TMR2 (TMR4) is cleared
- The CCP pin is set (exception: if PWM duty cycle = 0%, the CCP pin will not be set)
- The PWM duty cycle is latched from CCPRxL into CCPRxH

**Note:** The Timer2 and Timer 4 postscalers (see **Section 14.0 “Timer2 Module”** and **Section 16.0 “Timer4 Module”**) are not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

17.4.2 PWM Duty Cycle

The PWM duty cycle is specified by writing to the CCPRxL register and to the CCPxCON<5:4> bits. Up to 10-bit resolution is available. The CCPRxL contains the eight MSbs and the CCPxCON<5:4> contains the two LSbs. This 10-bit value is represented by CCPRxL:CCPxCON<5:4>. Equation 17-2 is used to calculate the PWM duty cycle in time.

**EQUATION 17-2:**

\[
\text{PWM Duty Cycle} = \frac{(CCPRxL:CCPxCON<5:4>) \times \text{Tosc} \times \text{TMR2 Prescale Value}}{2 \times \text{CCPxCON<5:4>}}
\]

CCPRxL and CCPxCON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPRxH until after a match between PR2 (PR4) and TMR2 (TMR4) occurs (i.e., the period is complete). In PWM mode, CCPRxH is a read-only register.
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.

When the CCPRxH and 2-bit latch match TMR2 (TMR4), concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 (TMR4) prescaler, the CCP pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by Equation 17-3:

\[
PWM \text{ Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}
\]

**Note:** If the PWM duty cycle value is longer than the PWM period, the CCP pin will not be cleared.

### 17.4.3 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 (PR4) register.
2. Set the PWM duty cycle by writing to the CCPRxL register and CCPxCON<5:4> bits.
3. Make the CCP pin an output by clearing the appropriate TRIS bit.
4. Set the TMR2 (TMR4) prescale value, then enable Timer2 (Timer4) by writing to T2CON (T4CON).
5. Configure the CCP module for PWM operation.

#### TABLE 17-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz

<table>
<thead>
<tr>
<th>PWM Frequency</th>
<th>2.44 kHz</th>
<th>9.77 kHz</th>
<th>39.06 kHz</th>
<th>156.25 kHz</th>
<th>312.50 kHz</th>
<th>416.67 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer Prescaler (1, 4, 16)</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PR2 Value</td>
<td>FFh</td>
<td>FFh</td>
<td>FFh</td>
<td>3Fh</td>
<td>1Fh</td>
<td>17h</td>
</tr>
<tr>
<td>Maximum Resolution (bits)</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>6.58</td>
</tr>
</tbody>
</table>
# PIC18F87J11 FAMILY

## TABLE 17-4: REGISTERS ASSOCIATED WITH PWM, TIMER2 AND TIMER4

<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/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>RCON</td>
<td>IPEN</td>
<td>—</td>
<td>CM</td>
<td>RI</td>
<td>TO</td>
<td>PD</td>
<td>POR</td>
<td>BOR</td>
<td>56</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR3</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMR4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>58</td>
</tr>
<tr>
<td>TRISG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRISG4</td>
<td>TRISG3</td>
<td>TRISG2</td>
<td>TRISG1</td>
<td>TRISG0</td>
<td>58</td>
</tr>
<tr>
<td>TMR2(1)</td>
<td>Timer2 Register</td>
<td>—</td>
<td>—</td>
<td>T2OUTPS3</td>
<td>T2OUTPS2</td>
<td>T2OUTPS1</td>
<td>T2OUTPS0</td>
<td>TMR2ON</td>
<td>T2CKPS1</td>
</tr>
<tr>
<td>PR2(1)</td>
<td>Timer2 Period Register</td>
<td>—</td>
<td>—</td>
<td>T2OUTPS3</td>
<td>T2OUTPS2</td>
<td>T2OUTPS1</td>
<td>T2OUTPS0</td>
<td>TMR2ON</td>
<td>T2CKPS1</td>
</tr>
<tr>
<td>TMR4</td>
<td>Timer4 Register</td>
<td>—</td>
<td>—</td>
<td>T4OUTPS3</td>
<td>T4OUTPS2</td>
<td>T4OUTPS1</td>
<td>T4OUTPS0</td>
<td>TMR4ON</td>
<td>T4CKPS1</td>
</tr>
<tr>
<td>PR4</td>
<td>Timer4 Period Register</td>
<td>—</td>
<td>—</td>
<td>T4OUTPS3</td>
<td>T4OUTPS2</td>
<td>T4OUTPS1</td>
<td>T4OUTPS0</td>
<td>TMR4ON</td>
<td>T4CKPS1</td>
</tr>
<tr>
<td>CCP4L</td>
<td>Capture/Compare/PWM Register 4 Low Byte</td>
<td>—</td>
<td>—</td>
<td>DC4B1</td>
<td>DC4B0</td>
<td>CCP4M3</td>
<td>CCP4M2</td>
<td>CCP4M1</td>
<td>CCP4M0</td>
</tr>
<tr>
<td>CCP4H</td>
<td>Capture/Compare/PWM Register 4 High Byte</td>
<td>—</td>
<td>—</td>
<td>DC5B1</td>
<td>DC5B0</td>
<td>CCP5M3</td>
<td>CCP5M2</td>
<td>CCP5M1</td>
<td>CCP5M0</td>
</tr>
<tr>
<td>CCP5CON</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP5OD</td>
<td>CCP4OD</td>
<td>ECCP3OD</td>
<td>ECCP2OD</td>
<td>ECCP1OD</td>
<td>56</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used by PWM, Timer2 or Timer4.

**Note 1:** Default (legacy) SFR at this address, available when WDTCON<4> = 0.

**Note 2:** Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.
18.0 ENHANCED CAPTURE/COMPARE/PWM (ECCP) MODULE

In the PIC18F87J11 family of devices, three of the CCP modules are implemented as standard CCP modules with Enhanced PWM capabilities. These include the provision for 2 or 4 output channels, user-selectable polarity, dead-band control and automatic shutdown and restart. The Enhanced features are discussed in detail in Section 18.4 “Enhanced PWM Mode”. Capture, Compare and single-output PWM functions of the ECCP module are the same as described for the standard CCP module.

The control register for the Enhanced CCP module is shown in Register 18-1. It differs from the CCP4CON/CCP5CON registers in that the two Most Significant bits are implemented to control PWM functionality.

In addition to the expanded range of modes available through the Enhanced CCPxCON register, the ECCP modules each have two additional registers associated with Enhanced PWM operation and auto-shutdown features. They are:

- ECCPxDEL (ECCPx PWM Delay)
- ECCPxAS (ECCPx Auto-Shutdown Control)

REGISTER 18-1: CCPxCON: ECCPx CONTROL REGISTER (ECCP1/ECCP2/ECCP3)

<table>
<thead>
<tr>
<th>Bit 7-0</th>
<th>PxA1</th>
<th>PxA0</th>
<th>DCxB1</th>
<th>DCxB0</th>
<th>CCPxM3</th>
<th>CCPxM2</th>
<th>CCPxM1</th>
<th>CCPxM0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

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

- **PxM1:PxM0**: Enhanced PWM Output Configuration bits
  - If CCPxM3:CCPxM2 = 00, 01, 10:
    - PxA assigned as Capture/Compare input/output; PxB, PxC, PxD assigned as port pins
  - If CCPxM3:CCPxM2 = 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

- **DCxB1:DCxB0**: PWM Duty Cycle bit 1 and bit 0
  - **Capture mode**: Unused.
  - **Compare mode**: Unused.
  - **PWM mode**: These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPRxL.
18.1 ECCP Outputs and Configuration

Each of the Enhanced CCP modules may have up to four PWM outputs, depending on the selected operating mode. These outputs, designated PxA through PxD, are multiplexed with various I/O pins. Some ECCP pin assignments are constant, while others change based on device configuration. For those pins that do change, the controlling bits are:

- CCP2MX Configuration bit
- ECCPMX Configuration bit (80-pin devices only)
- Program Memory Operating mode, set by the EMB Configuration bits (80-pin devices only)

The pin assignments for the Enhanced CCP modules are summarized in Table 18-1, Table 18-2 and Table 18-3. To configure the I/O pins as PWM outputs, the proper PWM mode must be selected by setting the PxMx and CCPxMx bits (CCPxCON<7:6> and <3:0>, respectively). The appropriate TRIS direction bits for the corresponding port pins must also be set as outputs.

18.1.1 ECCP1/ECCP3 OUTPUTS AND PROGRAM MEMORY MODE

An exception to this configuration is when a 12-bit address width is selected for the external bus (EMB1:EMB0 Configuration bits = 01). In this case, the upper pins of PORTE continue to operate as digital I/O, even when the external bus is active. P1B/P1C and P3B/P3C remain available for use as Enhanced PWM outputs.

If an application requires the use of additional PWM outputs during enhanced microcontroller operation, the P1B/P1C and P3B/P3C outputs can be reassigned to the upper bits of PORTH. This is done by clearing the ECCPMX Configuration bit.

18.1.2 ECCP2 OUTPUTS AND PROGRAM MEMORY MODES

For 80-pin devices, the program memory mode of the device (Section 5.1.3 “PIC18F8xJ11/8XJ16 Program Memory Modes”) also impacts pin multiplexing for the module.

The ECCP2 input/output (ECCP2/P2A) can be multiplexed to one of three pins. The default assignment (CCP2MX Configuration bit is set) for all devices is RC1. Clearing CCP2MX reassigns ECCP2/P2A to RE7.

An additional option exists for 80-pin devices. When these devices are operating in Microcontroller mode, the multiplexing options described above still apply. In Extended Microcontroller mode, clearing CCP2MX reassigns ECCP2/P2A to RB3.

Changing the pin assignment of ECCP2 does not automatically change any requirements for configuring the port pin. Users must always verify that the appropriate TRIS register is configured correctly for ECCP2 operation regardless of where it is located.

Note 1: Implemented only for ECCP1 and ECCP2; same as ‘1010’ for ECCP3.
18.1.3 USE OF CCP4 AND CCP5 WITH ECCP1 AND ECCP3

Only the ECCP2 module has four dedicated output pins that are available for use. Assuming that the I/O ports or other multiplexed functions on those pins are not needed, they may be used whenever needed without interfering with any other CCP module.

ECCP1 and ECCP3, on the other hand, only have three dedicated output pins: ECCPx/PxA, PxB and PxC. Whenever these modules are configured for Quad PWM mode, the pin normally used for CCP4 or CCP5 becomes the PxD output pins for ECCP3 and ECCP1, respectively. The CCP4 and CCP5 modules remain functional but their outputs are overridden.

18.1.4 ECCP MODULES AND TIMER RESOURCES

Like the standard CCP modules, the ECCP modules can utilize Timers 1, 2, 3 or 4, depending on the mode selected. Timer1 and Timer3 are available for modules in Capture or Compare modes, while Timer2 and Timer4 are available for modules in PWM mode. Additional details on timer resources are provided in Section 17.1.1 “CCP Modules and Timer Resources”.

18.1.5 OPEN-DRAIN OUTPUT OPTION

When operating in compare or standard PWM modes, the drivers for the ECCP pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor, and allows the output to communicate with external circuits without the need for additional level shifters. For more information, see Section 10.1.4 “Open-Drain Outputs”.

The open-drain output option is controlled by the bits in the ODCON1 register. Setting the appropriate bit configures the pin for the corresponding module for open-drain operation. The ODCON1 memory shares the same address space as of TMR1H. The ODCON1 register can be accessed by setting the ADSHR bit in the WDTCR register (WDTCR<4>).

### TABLE 18-1: PIN CONFIGURATIONS FOR ECCP1

<table>
<thead>
<tr>
<th>ECCP Mode</th>
<th>CCP1CON Configuration</th>
<th>RC2</th>
<th>RE6</th>
<th>RE5</th>
<th>RG4</th>
<th>RH7</th>
<th>RH6</th>
</tr>
</thead>
<tbody>
<tr>
<td>All PIC18F6XJ1X Devices:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compatible CCP</td>
<td>00xx 11xx</td>
<td>ECCP1</td>
<td>RE6</td>
<td>RE5</td>
<td>RG4/CCP5</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Dual PWM</td>
<td>10xx 11xx</td>
<td>P1A</td>
<td>P1B</td>
<td>RE5</td>
<td>RG4/CCP5</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Quad PWM(1)</td>
<td>x1xx 11xx</td>
<td>P1A</td>
<td>P1B</td>
<td>P1C</td>
<td>P1D</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**PIC18F8XJ1X Devices, ECCPMX = 0, Microcontroller mode:**

<table>
<thead>
<tr>
<th>ECCP Mode</th>
<th>CCP1CON Configuration</th>
<th>RC2</th>
<th>RE6</th>
<th>RE5</th>
<th>RG4</th>
<th>RH7</th>
<th>RH6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatible CCP</td>
<td>00xx 11xx</td>
<td>ECCP1</td>
<td>RE6/AD14</td>
<td>RE5/AD13</td>
<td>RG4/CCP5</td>
<td>RH7/AN15</td>
<td>RH6/AN14</td>
</tr>
<tr>
<td>Dual PWM</td>
<td>10xx 11xx</td>
<td>P1A</td>
<td>RE6/AD14</td>
<td>RE5/AD13</td>
<td>RG4/CCP5</td>
<td>P1B</td>
<td>RH6/AN14</td>
</tr>
<tr>
<td>Quad PWM(1)</td>
<td>x1xx 11xx</td>
<td>P1A</td>
<td>RE6/AD14</td>
<td>RE5/AD13</td>
<td>P1D</td>
<td>P1B</td>
<td>P1C</td>
</tr>
</tbody>
</table>

**PIC18F8XJ1X Devices, ECCPMX = 1, Extended Microcontroller mode, 16-Bit or 20-Bit Address Width:**

<table>
<thead>
<tr>
<th>ECCP Mode</th>
<th>CCP1CON Configuration</th>
<th>RC2</th>
<th>RE6</th>
<th>RE5</th>
<th>RG4</th>
<th>RH7</th>
<th>RH6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatible CCP</td>
<td>00xx 11xx</td>
<td>ECCP1</td>
<td>RE6/AD14</td>
<td>RE5/AD13</td>
<td>RG4/CCP5</td>
<td>RH7/AN15</td>
<td>RH6/AN14</td>
</tr>
</tbody>
</table>

**PIC18F8XJ1X Devices, ECCPMX = 1, Microcontroller mode or Extended Microcontroller mode, 12-Bit Address Width:**

<table>
<thead>
<tr>
<th>ECCP Mode</th>
<th>CCP1CON Configuration</th>
<th>RC2</th>
<th>RE6</th>
<th>RE5</th>
<th>RG4</th>
<th>RH7</th>
<th>RH6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatible CCP</td>
<td>00xx 11xx</td>
<td>ECCP1</td>
<td>RE6/AD14</td>
<td>RE5/AD13</td>
<td>RG4/CCP5</td>
<td>RH7/AN15</td>
<td>RH6/AN14</td>
</tr>
<tr>
<td>Dual PWM</td>
<td>10xx 11xx</td>
<td>P1A</td>
<td>P1B</td>
<td>RE5/AD13</td>
<td>RG4/CCP5</td>
<td>RH7/AN15</td>
<td>RH6/AN14</td>
</tr>
<tr>
<td>Quad PWM(1)</td>
<td>x1xx 11xx</td>
<td>P1A</td>
<td>P1B</td>
<td>P1C</td>
<td>P1D</td>
<td>RH7/AN15</td>
<td>RH6/AN14</td>
</tr>
</tbody>
</table>

**Legend:** x = Don’t care, N/A = Not Available. Shaded cells indicate pin assignments not used by ECCP1 in a given mode.

**Note 1:** With ECCP1 in Quad PWM mode, CCP5’s output is overridden by P1D; otherwise, CCP5 is fully operational.
### TABLE 18-2: PIN CONFIGURATIONS FOR ECCP2

<table>
<thead>
<tr>
<th>ECCP Mode</th>
<th>CCP2CON Configuration</th>
<th>RB3</th>
<th>RC1</th>
<th>RE7</th>
<th>RE2</th>
<th>RE1</th>
<th>RE0</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Devices, CCP2MX = 1, Either Operating mode:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compatible CCP</td>
<td>00xx 11xx</td>
<td>RB3/INT3</td>
<td>ECCP2</td>
<td>RE7</td>
<td>RE2</td>
<td>RE1</td>
<td>RE0</td>
</tr>
<tr>
<td>Dual PWM</td>
<td>10xx 11xx</td>
<td>RB3/INT3</td>
<td>P2A</td>
<td>RE7</td>
<td>P2B</td>
<td>RE1</td>
<td>RE0</td>
</tr>
<tr>
<td>Quad PWM</td>
<td>x1xx 11xx</td>
<td>RB3/INT3</td>
<td>P2A</td>
<td>RE7</td>
<td>P2B</td>
<td>P2C</td>
<td>P2D</td>
</tr>
<tr>
<td>All Devices, CCP2MX = 0, Microcontroller mode:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compatible CCP</td>
<td>00xx 11xx</td>
<td>RB3/INT3</td>
<td>RC1/T1OS1</td>
<td>ECCP2</td>
<td>RE2</td>
<td>RE1</td>
<td>RE0</td>
</tr>
<tr>
<td>Dual PWM</td>
<td>10xx 11xx</td>
<td>RB3/INT3</td>
<td>RC1/T1OS1</td>
<td>P2A</td>
<td>P2B</td>
<td>RE1</td>
<td>RE0</td>
</tr>
<tr>
<td>Quad PWM</td>
<td>x1xx 11xx</td>
<td>RB3/INT3</td>
<td>RC1/T1OS1</td>
<td>P2A</td>
<td>P2B</td>
<td>P2C</td>
<td>P2D</td>
</tr>
<tr>
<td>PIC18F8XJ1X Devices, CCP2MX = 0, Extended Microcontroller mode:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compatible CCP</td>
<td>00xx 11xx</td>
<td>ECCP3</td>
<td>RE4</td>
<td>RE3</td>
<td>RG3/CCP4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Dual PWM</td>
<td>10xx 11xx</td>
<td>P3A</td>
<td>P3B</td>
<td>RE3</td>
<td>RG3/CCP4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Quad PWM(1)</td>
<td>x1xx 11xx</td>
<td>P3A</td>
<td>P3B</td>
<td>P3C</td>
<td>P3D</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Legend:**
- x = Don’t care
- N/A = Not Available
- Shaded cells indicate pin assignments not used by ECCP2 in a given mode.

### TABLE 18-3: PIN CONFIGURATIONS FOR ECCP3

<table>
<thead>
<tr>
<th>ECCP Mode</th>
<th>CCP3CON Configuration</th>
<th>RG0</th>
<th>RE4</th>
<th>RE3</th>
<th>RG3</th>
<th>RH5</th>
<th>RH4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC18F6XJ1X Devices:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compatible CCP</td>
<td>00xx 11xx</td>
<td>ECCP3</td>
<td>RE4</td>
<td>RE3</td>
<td>RG3/CCP4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Dual PWM</td>
<td>10xx 11xx</td>
<td>P3A</td>
<td>P3B</td>
<td>RE3</td>
<td>RG3/CCP4</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Quad PWM(1)</td>
<td>x1xx 11xx</td>
<td>P3A</td>
<td>P3B</td>
<td>P3C</td>
<td>P3D</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>PIC18F8XJ1X Devices, ECCPMX = 0, Microcontroller mode:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quad PWM(1)</td>
<td>x1xx 11xx</td>
<td>P3A</td>
<td>RE6/AD14</td>
<td>RE5/AD13</td>
<td>P3D</td>
<td>P3B</td>
<td>P3C</td>
</tr>
<tr>
<td>PIC18F8XJ1X Devices, ECCPMX = 1, Extended Microcontroller mode, 16-Bit or 20-Bit Address Width:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quad PWM(1)</td>
<td>x1xx 11xx</td>
<td>P3A</td>
<td>P3B</td>
<td>P3C</td>
<td>P3D</td>
<td>RH5/AN13</td>
<td>RH4/AN12</td>
</tr>
<tr>
<td>PIC18F8XJ1X Devices, ECCPMX = 1, Microcontroller mode or Extended Microcontroller mode, 12-Bit Address Width:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dual PWM</td>
<td>10xx 11xx</td>
<td>P3A</td>
<td>P3B</td>
<td>P3C</td>
<td>P3D</td>
<td>RH5/AN13</td>
<td>RH4/AN12</td>
</tr>
<tr>
<td>Quad PWM(1)</td>
<td>x1xx 11xx</td>
<td>P3A</td>
<td>P3B</td>
<td>P3C</td>
<td>P3D</td>
<td>RH5/AN13</td>
<td>RH4/AN12</td>
</tr>
</tbody>
</table>

**Legend:**
- x = Don’t care
- N/A = Not Available
- Shaded cells indicate pin assignments not used by ECCP3 in a given mode.

**Note 1:** With ECCP3 in Quad PWM mode, CCP4’s output is overridden by P1D; otherwise, CCP4 is fully operational.
18.2 Capture and Compare Modes

Except for the operation of the Special Event Trigger discussed below, the Capture and Compare modes of the ECCP module are identical in operation to that of CCP4. These are discussed in detail in Section 17.2 “Capture Mode” and Section 17.3 “Compare Mode”.

18.2.1 SPECIAL EVENT TRIGGER

ECCP1 and ECCP2 incorporate an internal hardware trigger that is generated in Compare mode on a match between the CCPRx register pair and the selected timer. This can be used in turn to initiate an action. This mode is selected by setting CCPxCON<3:0> to ‘1011’.

The Special Event Trigger output of either ECCP1 or ECCP2 resets the TMR1 or TMR3 register pair, depending on which timer resource is currently selected. This allows the CCPRx register pair to effectively be a 16-bit programmable period register for Timer1 or Timer3. In addition, the ECCP2 Special Event Trigger will also start an A/D conversion if the A/D module is enabled.

Special Event Triggers are not implemented for ECCP3, CCP4 or CCP5. Selecting the Special Event Trigger mode for these modules has the same effect as selecting the Compare with Software Interrupt mode (CCPxM3:CCPxM0 = 1010).

Note: The Special Event Trigger from ECCP2 will not set the Timer1 or Timer3 interrupt flag bits.

18.3 Standard PWM Mode

When configured in Single Output mode, the ECCP module functions identically to the standard CCP module in PWM mode, as described in Section 17.4 “PWM Mode”. This is also sometimes referred to as “Compatible CCP” mode as in Tables 18-1 through 18-3.
18.4 Enhanced PWM Mode

The Enhanced PWM mode provides additional PWM output options for a broader range of control applications. The module is a backward compatible version of the standard CCP module and offers up to four outputs, designated PxA through PxD. Users are also able to select the polarity of the signal (either active-high or active-low). The module’s output mode and polarity are configured by setting the PxM1:PxM0 and CCPxM3:CCPxM0 bits of the CCPxCON register (CCPxCON<7:6> and CCPxCON<3:0>, respectively).

For the sake of clarity, Enhanced PWM mode operation is described generically throughout this section with respect to the ECCP1 and TMR2 modules. Control register names are presented in terms of ECCP1. All three Enhanced modules, as well as the two timer resources, can be used interchangeably and function identically. TMR2 or TMR4 can be selected for PWM operation by selecting the proper bits in T3CON.

Figure 18-1 shows a simplified block diagram of PWM operation. All control registers are double-buffered and are loaded at the beginning of a new PWM cycle (the period boundary when Timer2 resets) in order to prevent glitches on any of the outputs. The exception is the ECCPx PWM Delay register, ECCPxDEL, which is loaded at either the duty cycle boundary or the boundary period (whichever comes first). Because of the buffering, the module waits until the assigned timer resets instead of starting immediately. This means that Enhanced PWM waveforms do not exactly match the standard PWM waveforms, but are instead offset by one full instruction cycle (4 Tosc).

As before, the user must manually configure the appropriate TRIS bits for output.

18.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the equation:

\[
\text{PWM Period} = \left(\frac{\text{PR2} + 1}{4} \cdot \text{Tosc} \cdot \frac{\text{TMR2 Prescale Value}}{}\right)
\]

PWM frequency is defined as \(1/[\text{PWM period}]\). When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The ECCP1 pin is set (if PWM duty cycle = 0%, the ECCP1 pin will not be set)
- The PWM duty cycle is copied from CCPR1L into CCPR1H

Note: The Timer2 postscaler (see Section 14.0 “Timer2 Module”) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

---

**FIGURE 18-1: SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODULE**

Note: The 8-bit TMR2 register is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler, to create the 10-bit time base.
18.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The PWM duty cycle is calculated by the following equation:

**EQUATION 18-2:**

\[
PWM\ \text{Duty Cycle} = \frac{(CCPR1L:CCP1CON<5:4>) \times T_{OSC}}{T_{TMR2 \text{ Prescale}}}\]

CCPR1L and CCP1CON<5:4> can be written to at any time but the duty cycle value is not copied into CCPR1H until a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H 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. When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or two bits of the TMR2 prescaler, the ECCP1 pin is cleared. The maximum PWM resolution (bits) for a given PWM frequency is given by the equation:

**EQUATION 18-3:**

\[
PWM\ \text{Resolution (max)} = \log\left(\frac{F_{OSC}}{F_{PWM}}\right) \text{ bits} \]

18.4.3 PWM OUTPUT CONFIGURATIONS

The P1M1:P1M0 bits in the CCP1CON register allow one of four configurations:

- Single Output
- Half-Bridge Output
- Full-Bridge Output, Forward mode
- Full-Bridge Output, Reverse mode

The Single Output mode is the standard PWM mode discussed in Section 18.4 “Enhanced PWM Mode”. The Half-Bridge and Full-Bridge Output modes are covered in detail in the sections that follow.

The general relationship of the outputs in all configurations is summarized in Figure 18-2.

**TABLE 18-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz**

<table>
<thead>
<tr>
<th>PWM Frequency</th>
<th>2.44 kHz</th>
<th>9.77 kHz</th>
<th>39.06 kHz</th>
<th>156.25 kHz</th>
<th>312.50 kHz</th>
<th>416.67 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timer Prescaler (1, 4, 16)</td>
<td>16</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PR2 Value</td>
<td>FFh</td>
<td>FFh</td>
<td>FFh</td>
<td>3Fh</td>
<td>1Fh</td>
<td>17h</td>
</tr>
<tr>
<td>Maximum Resolution (bits)</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>6.58</td>
</tr>
</tbody>
</table>
FIGURE 18-2: PWM OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)

<table>
<thead>
<tr>
<th>CCP1CON&lt;7:6&gt;</th>
<th>SIGNAL</th>
<th>0 Duty Cycle</th>
<th>Delay (1)</th>
<th>PR2 + 1 Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>(Single Output) P1A Modulated</td>
<td>Delay (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1A Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>(Half-Bridge) 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>01</td>
<td>(Full-Bridge, Forward) 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</td>
<td>(Full-Bridge, Reverse) P1B Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1A Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relationships:
• Period = 4 * Tosc * (PR2 + 1) * (TMR2 Prescale Value)
• Duty Cycle = Tosc * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 Prescale Value)
• Delay = 4 * Tosc * (ECCP1DEL<6:0>)

Note 1: Dead-band delay is programmed using the ECCP1DEL register (Section 18.4.6 “Programmable Dead-Band Delay”).

FIGURE 18-3: PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)

<table>
<thead>
<tr>
<th>CCP1CON&lt;7:6&gt;</th>
<th>SIGNAL</th>
<th>0 Duty Cycle</th>
<th>Delay (1)</th>
<th>PR2 + 1 Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>(Single Output) P1A Modulated</td>
<td>Delay (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1A Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>(Half-Bridge) 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>01</td>
<td>(Full-Bridge, Forward) 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</td>
<td>(Full-Bridge, Reverse) P1B Modulated</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P1A Inactive</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Relationships:
• Period = 4 * Tosc * (PR2 + 1) * (TMR2 Prescale Value)
• Duty Cycle = Tosc * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 Prescale Value)
• Delay = 4 * Tosc * (ECCP1DEL<6:0>)

Note 1: Dead-band delay is programmed using the ECCP1DEL register (Section 18.4.6 “Programmable Dead-Band Delay”).
18.4.4 HALF-BRIDGE MODE

In the Half-Bridge Output mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the P1A pin, while the complementary PWM output signal is output on the P1B pin (Figure 18-4). This mode can be used for half-bridge applications, as shown in Figure 18-5, or for full-bridge applications, where four power switches are being modulated with two PWM signals.

In Half-Bridge Output mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of bits P1DC6:P1DC0 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 18.4.6 “Programmable Dead-Band Delay” for more details on dead-band delay operations.

Since the P1A and P1B outputs are multiplexed with the PORTC<2> and PORTE<6> data latches, the TRISC<2> and TRISE<6> bits must be cleared to configure P1A and P1B as outputs.

FIGURE 18-4: HALF-BRIDGE PWM OUTPUT

Figure 18-4 shows the PWM output timing for the Half-Bridge mode. The period, duty cycle, and dead-band delay are indicated.

Note 1: At this time, the TMR2 register is equal to the PR2 register.
2: Output signals are shown as active-high.

FIGURE 18-5: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS

- **Standard Half-Bridge Circuit (“Push-Pull”)**
  - PIC18F87J11
  - P1A and P1B outputs driving FETs
  - Output signals are shown as active-high.

- **Half-Bridge Output Driving a Full-Bridge Circuit**
  - PIC18F87J11
  - P1A and P1B outputs driving FETs
  - Output signals are shown as active-high.
18.4.5 FULL-BRIDGE MODE

In Full-Bridge Output mode, four pins are used as outputs; however, only two outputs are active at a time. In the Forward mode, pin P1A is continuously active and pin P1D is modulated. In the Reverse mode, pin P1C is continuously active and pin P1B is modulated. These are illustrated in Figure 18-6.

P1A, P1B, P1C and P1D outputs are multiplexed with the port pins as described in Table 18-1, Table 18-2 and Table 18-3. The corresponding TRIS bits must be cleared to make the P1A, P1B, P1C and P1D pins outputs.

FIGURE 18-6:  FULL-BRIDGE PWM OUTPUT

Note 1: At this time, the TMR2 register is equal to the PR2 register.
Note 2: Output signal is shown as active-high.
18.4.5.1 Direction Change in Full-Bridge Output Mode

In the Full-Bridge Output mode, the P1M1 bit in the CCP1CON register allows users to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will assume the new direction on the next PWM cycle.

Just before the end of the current PWM period, the modulated outputs (P1B and P1D) are placed in their inactive state, while the unmodulated outputs (P1A and P1C) are switched to drive in the opposite direction. This occurs in a time interval of \((4 \times T_{os} \times \text{Prescale Value})\) before the next PWM period begins. The Timer2 prescaler will be either 1, 4 or 16, depending on the value of the T2CKPS bits (T2CON<1:0>). During the interval from the switch of the unmodulated outputs to the beginning of the next period, the modulated outputs (P1B and P1D) remain inactive. This relationship is shown in Figure 18-8.

Note that in the Full-Bridge Output mode, the ECCP1 module does not provide any dead-band delay. In general, since only one output is modulated at all times, dead-band delay is not required. However, there is a situation where a dead-band delay might be 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 18-9 shows an example where the PWM direction changes from forward to reverse at a near 100% duty cycle. At time t1, the outputs, P1A and P1D, become inactive, while output, P1C, becomes active. In this example, since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current may flow through power devices, QC and QD (see Figure 18-7), 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, one of the following requirements must be met:

1. Reduce PWM for a 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 18-8:  PWM DIRECTION CHANGE

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>Period(1)</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1A (Active-High)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1B (Active-High)</td>
<td></td>
<td>DC</td>
</tr>
<tr>
<td>P1C (Active-High)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1D (Active-High)</td>
<td>DC</td>
<td>(Note 2)</td>
</tr>
</tbody>
</table>

Note 1: The direction bit in the ECCP1 Control register (CCP1CON<7>) is written at 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 at intervals of 4 Tosc, 16 Tosc or 64 Tosc, depending on the Timer2 prescaler value. The modulated P1B and P1D signals are inactive at this time.

FIGURE 18-9:  PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE

<table>
<thead>
<tr>
<th></th>
<th>Forward Period</th>
<th>t1</th>
<th>Reverse Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1A(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1B(1)</td>
<td></td>
<td></td>
<td>DC</td>
</tr>
<tr>
<td>P1C(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1D(1)</td>
<td>DC</td>
<td></td>
<td>tON(2)</td>
</tr>
<tr>
<td>External Switch C(1)</td>
<td></td>
<td></td>
<td>tOFF(3)</td>
</tr>
<tr>
<td>External Switch D(1)</td>
<td></td>
<td></td>
<td>t = tOFF - tON(2,3)</td>
</tr>
<tr>
<td>Potential Shoot-Through Current(1)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1: All signals are shown as active-high.
Note 2: tON is the turn-on delay of power switch QC and its driver.
Note 3: tOFF is the turn-off delay of power switch QD and its driver.
18.4.6 PROGRAMMABLE DEAD-BAND DELAY

In half-bridge applications, where all power switches are modulated at the PWM frequency at all times, 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) may 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 the Half-Bridge Output 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 18-4 for illustration). The lower seven bits of the ECCPxDEL register (Register 18-2) set the delay period in terms of microcontroller instruction cycles (Tcy or 4 Tosc).

18.4.7 ENHANCED PWM AUTO-SHUTDOWN

When the ECP1 is programmed for any of the Enhanced PWM modes, the active output pins may be configured for auto-shutdown. Auto-shutdown immediately places the Enhanced PWM output pins into a defined shutdown state when a shutdown event occurs.

A shutdown event can be caused by either of the two comparator modules or the FLT0 pin (or any combination of these three sources). The comparators may be used to monitor a voltage input proportional to a current being monitored in the bridge circuit. If the voltage exceeds a threshold, the comparator switches state and triggers a shutdown. Alternatively, a low-level digital signal on the FLT0 pin can also trigger a shutdown. The auto-shutdown feature can be disabled by not selecting any auto-shutdown sources. The auto-shutdown sources to be used are selected using the ECCP1AS2:ECCP1AS0 bits (ECCP1AS<6:4>.

When a shutdown occurs, the output pins are asynchronously placed in their shutdown states, specified by the PSS1AC1:PSS1AC0 and PSS1BD1:PSS1BD0 bits (ECCP1AS3:ECCP1AS0). Each pin pair (P1A/P1C and P1B/P1D) may be set to drive high, drive low or be tri-stated (not driving). The ECCP1ASE bit (ECCP1AS<7>) is also set to hold the Enhanced PWM outputs in their shutdown states.

The ECCP1ASE bit is set by hardware when a shutdown event occurs. If automatic restarts are not enabled, the ECCP1ASE bit is cleared by firmware when the cause of the shutdown clears. If automatic restarts are enabled, the ECCP1ASE bit is automatically cleared when the cause of the auto-shutdown has cleared.

If the ECCP1ASE bit is set when a PWM period begins, the PWM outputs remain in their shutdown state for that entire PWM period. When the ECCP1ASE bit is cleared, the PWM outputs will return to normal operation at the beginning of the next PWM period.

Note: Writing to the ECCP1ASE bit is disabled while a shutdown condition is active.
REGISTER 18-2:  ECCPxDEL: ECCPx PWM DELAY REGISTER

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

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

bit 7  **PxRSEN**: PWM Restart Enable bit
1 = Upon auto-shutdown, the ECCPxASE bit clears automatically once the shutdown event goes away; the PWM restarts automatically
0 = Upon auto-shutdown, ECCPxASE must be cleared in software to restart the PWM

bit 6-0  **PxDC6**:PxDC0: PWM Delay Count bits
Delay time, in number of $\frac{F_{OSC}}{4} (4 \times T_{OSC})$ cycles, between the scheduled and actual time for a PWM signal to transition to active.

REGISTER 18-3:  ECCPxAS: ECCPx AUTO-SHUTDOWN CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECCPxASE</td>
<td>ECCPxAS2</td>
<td>ECCPxAS1</td>
<td>ECCPxAS0</td>
<td>PSSxAC1</td>
<td>PSSxAC0</td>
<td>PSSxBD1</td>
<td>PSSxBD0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
</tbody>
</table>

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

bit 7  **ECCPxASE**: ECCPx Auto-Shutdown Event Status bit
0 = ECCPx outputs are operating
1 = A shutdown event has occurred; ECCPx outputs are in shutdown state

bit 6-4  **ECCPxAS2**:ECCPxAS0: ECCPx Auto-Shutdown Source Select bits
000 = Auto-shutdown is disabled
001 = Comparator 1 output
010 = Comparator 2 output
011 = Either Comparator 1 or 2
100 = FLT0
101 = FLT0 or Comparator 1
110 = FLT0 or Comparator 2
111 = FLT0 or Comparator 1 or Comparator 2

bit 3-2  **PSSxAC1**:PSSxAC0: Pins A and C Shutdown State Control bits
00 = Drive Pins A and C to '0'
01 = Drive Pins A and C to '1'
1x = Pins A and C tri-state

bit 1-0  **PSSxBD1**:PSSxBD0: Pins B and D Shutdown State Control bits
00 = Drive Pins B and D to '0'
01 = Drive Pins B and D to '1'
1x = Pins B and D tri-state
18.4.7.1 Auto-Shutdown and Automatic Restart

The auto-shutdown feature can be configured to allow automatic restarts of the module following a shutdown event. This is enabled by setting the P1RSEN bit of the ECCP1DEL register (ECCP1DEL<7>).

In Shutdown mode with P1RSEN = 1 (Figure 18-10), the ECCP1ASE bit will remain set for as long as the cause of the shutdown continues. When the shutdown condition clears, the ECCP1ASE bit is cleared. If P1RSEN = 0 (Figure 18-11), once a shutdown condition occurs, the ECCP1ASE bit will remain set until it is cleared by firmware. Once ECCP1ASE is cleared, the Enhanced PWM will resume at the beginning of the next PWM period.

Note: Writing to the ECCP1ASE bit is disabled while a shutdown condition is active.

Independent of the P1RSEN bit setting, if the auto-shutdown source is one of the comparators, the shutdown condition is a level. The ECCP1ASE bit cannot be cleared as long as the cause of the shutdown persists.

The Auto-Shutdown mode can be forced by writing a ‘1’ to the ECCP1ASE bit.

18.4.8 START-UP CONSIDERATIONS

When the ECCP1 module is used in the PWM mode, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins. 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 CCP1M1:CCP1M0 bits (CCP1CON<1:0>) 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 pins are configured as outputs. Changing the polarity configuration while the PWM pins are configured as outputs 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 pins for output at the same time as the ECCP1 module may cause damage to the application circuit. The ECCP1 module must be enabled in the proper output mode and complete a full PWM cycle before configuring the PWM pins as outputs. The completion of a full PWM cycle is indicated by the TMR2IF bit being set as the second PWM period begins.
18.4.9 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the ECCP module for PWM operation:

1. Configure the PWM pins PxA and PxB (and PxC and PxD, if used) as inputs by setting the corresponding TRIS bits.
2. Set the PWM period by loading the PR2 (PR4) register.
3. Configure the ECCP module for the desired PWM mode and configuration by loading the CCPxCON register with the appropriate values:
   - Select one of the available output configurations and direction with the PxM1:PxM0 bits.
   - Select the polarities of the PWM output signals with the CCPxM3:CCPxM0 bits.
4. Set the PWM duty cycle by loading the CCPRxL register and the CCPxCON<5:4> bits.
5. For auto-shutdown:
   - Disable auto-shutdown; ECCPxASE = 0
   - Configure auto-shutdown source
   - Wait for Run condition
6. For Half-Bridge Output mode, set the dead-band delay by loading ECCPxDEL<6:0> with the appropriate value.
7. If auto-shutdown operation is required, load the ECCPxAS register:
   - Select the auto-shutdown sources using the ECCPxAS2:ECCPxAS0 bits.
   - Select the shutdown states of the PWM output pins using the PSSxAC1:PSSxAC0 and PSSxBD1:PSSxBD0 bits.
   - Set the ECCPxASE bit (ECCPxAS<7>).
8. If auto-restart operation is required, set the PxRSEN bit (ECCPxDEL<7>).
9. Configure and start TMRn (TMR2 or TMR4):
   - Clear the TMRn interrupt flag bit by clearing the TMRnIF bit (PIR1<1> for Timer2 or PIR3<3> for Timer4).
   - Set the TMRn prescale value by loading the TnCKPS bits (TnCON<1:0>).
   - Enable Timer2 (or Timer4) by setting the TMRnON bit (TnCON<2>).
10. Enable PWM outputs after a new PWM cycle has started:
    - Wait until TMRn overflows (TMRnIF bit is set).
    - Enable the ECCPx/PxA, PxB, PxC and/or PxD pin outputs by clearing the respective TRIS bits.
    - Clear the ECCPxASE bit (ECCPxAS<7>).

18.4.10 EFFECTS OF A RESET

Both Power-on Reset and subsequent Resets will force all ports to Input mode and the ECCP registers to their Reset states.

This forces the Enhanced CCP module to reset to a state compatible with the standard CCP module.
## TABLE 18-5: REGISTERS ASSOCIATED WITH ECCP MODULES AND TIMER1 TO TIMER4

<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/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>RCON</td>
<td>IPEN</td>
<td>—</td>
<td>CM</td>
<td>RI</td>
<td>T0</td>
<td>PD</td>
<td>POR</td>
<td>BOR</td>
<td>56</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSCF1F</td>
<td>CM2IF</td>
<td>CM1IF</td>
<td>—</td>
<td>BCL1IF</td>
<td>LVDIF</td>
<td>TMR3IF</td>
<td>CCP2IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSCF1E</td>
<td>CM2IE</td>
<td>CM1IE</td>
<td>—</td>
<td>BCL1IE</td>
<td>LVDIE</td>
<td>TMR3IE</td>
<td>CCP2IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR2</td>
<td>OSCF1P</td>
<td>CM2IP</td>
<td>CM1IP</td>
<td>—</td>
<td>BCL1P</td>
<td>LVDIP</td>
<td>TMR3IP</td>
<td>CCP2IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR3</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMR4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>58</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>58</td>
</tr>
<tr>
<td>TRISC</td>
<td>TRISC7</td>
<td>TRISC6</td>
<td>TRISC5</td>
<td>TRISC4</td>
<td>TRISC3</td>
<td>TRISC2</td>
<td>TRISC1</td>
<td>TRISC0</td>
<td>58</td>
</tr>
<tr>
<td>TRISE</td>
<td>TRISE7</td>
<td>TRISE6</td>
<td>TRISE5</td>
<td>TRISE4</td>
<td>TRISE3</td>
<td>TRISE2</td>
<td>TRISE1</td>
<td>TRISE0</td>
<td>58</td>
</tr>
<tr>
<td>TRISG</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>TRISG4</td>
<td>TRISG3</td>
<td>TRISG2</td>
<td>TRISG1</td>
<td>TRISG0</td>
<td>58</td>
</tr>
<tr>
<td>TRISH(1)</td>
<td>TRISH7</td>
<td>TRISH6</td>
<td>TRISH5</td>
<td>TRISH4</td>
<td>TRISH3</td>
<td>TRISH2</td>
<td>TRISH1</td>
<td>TRISH0</td>
<td>58</td>
</tr>
<tr>
<td>TMR1L(3)</td>
<td>Timer1 Register Low Byte</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR1H(3)</td>
<td>Timer1 Register High Byte</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODCON(4)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>CCP5OD</td>
<td>CCP4OD</td>
<td>CCP3OD</td>
<td>CCP2OD</td>
<td>CCP1OD</td>
</tr>
<tr>
<td>T1CON(3)</td>
<td>RD16</td>
<td>T1RUN</td>
<td>T1CKPS1</td>
<td>T1CKPS0</td>
<td>T1OSCEN</td>
<td>TTSYNC</td>
<td>TMR1CS</td>
<td>TMR1ON</td>
<td>56</td>
</tr>
<tr>
<td>TMR2(3)</td>
<td>Timer2 Register</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2CON(3)</td>
<td>—</td>
<td>T2OUTPS3</td>
<td>T2OUTPS2</td>
<td>T2OUTPS1</td>
<td>T2OUTPS0</td>
<td>TMR2ON</td>
<td>T2CKPS1</td>
<td>T2CKPS0</td>
<td>56</td>
</tr>
<tr>
<td>PR2(3)</td>
<td>Timer2 Period Register</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR3L</td>
<td>Timer3 Register Low Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TMR3H</td>
<td>Timer3 Register High Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3CON</td>
<td>RD16</td>
<td>T3CCP2</td>
<td>T3CKPS1</td>
<td>T3CKPS0</td>
<td>T3CCP1</td>
<td>T3SYNC</td>
<td>TMR3CS</td>
<td>TMR3ON</td>
<td>59</td>
</tr>
<tr>
<td>TMR4</td>
<td>Timer4 Register</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4CON(3)</td>
<td>—</td>
<td>T4OUTPS3</td>
<td>T4OUTPS2</td>
<td>T4OUTPS1</td>
<td>T4OUTPS0</td>
<td>TMR4ON</td>
<td>T4CKPS1</td>
<td>T4CKPS0</td>
<td>59</td>
</tr>
<tr>
<td>PR4(3)</td>
<td>Timer4 Period Register</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCPRxL(2)</td>
<td>Capture/Compare/PWM Register x Low Byte</td>
<td>57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCPRxH(2)</td>
<td>Capture/Compare/PWM Register x High Byte</td>
<td>57,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCpCON(2)</td>
<td>PxM1</td>
<td>PxM0</td>
<td>DCxB1</td>
<td>DCxB0</td>
<td>CCPxM3</td>
<td>CCPxM2</td>
<td>CCPxM1</td>
<td>CCPxB0</td>
<td>57</td>
</tr>
<tr>
<td>ECPxAS(2)</td>
<td>ECCpXAS</td>
<td>ECCpXAS2</td>
<td>ECCpXAS1</td>
<td>ECCpXAS0</td>
<td>PSSxAC1</td>
<td>PSSxAC0</td>
<td>PSSxBD</td>
<td>PSSxBD0</td>
<td>57</td>
</tr>
<tr>
<td>ECPxDEL(2)</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>57</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used during ECCP operation.

**Note 1:** Available on 80-pin devices only.

2: Generic term for all of the identical registers of this name for all Enhanced CCP modules, where ‘x’ identifies the individual module (ECCP1, ECCP2 or ECCP3). Bit assignments and Reset values for all registers of the same generic name are identical.

3: Default (legacy) SFR at this address, available when WDTCON<4> = 0.

4: Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.
19.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

19.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I²C™)
  - Full Master mode
  - Slave mode (with general address call)

The I²C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode with 5-bit and 7-bit address masking (with address masking for both 10-bit and 7-bit addressing)

All members of the PIC18F87J11 Family have two MSSP modules, designated as MSSP1 and MSSP2. Each module operates independently of the other.

Note: 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 and module I/O signals use the generic designator ‘x’ to indicate the use of a numeral to distinguish a particular module when required. Control bit names are not individuated.

19.2 Control Registers

Each MSSP module has three associated control registers. These include a status register (SSPxSTAT) and two control registers (SSPxCON1 and SSPxCON2). The use of these registers and their individual configuration bits differ significantly depending on whether the MSSP module is operated in SPI or I²C mode.

Additional details are provided under the individual sections.

Note: In devices with more than one MSSP module, it is very important to pay close attention to SSPxCON register names. SSP1CON1 and SSP1CON2 control different operational aspects of the same module, while SSP1CON1 and SSP2CON1 control the same features for two different modules.

19.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

- Serial Data Out (SDOx) – RC5/SDO1 or RD4/SDO2
- Serial Data In (SDIx) – RC4/SDI1/SDA1 or RD5/SDI2/SDA2
- Serial Clock (SCKx) – RC3/SCK1/SCL1 or RD6/SCK2/SCL2

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select (SSx) – RF7/SS1 or RD7/SS2

Figure 19-1 shows the block diagram of the MSSP module when operating in SPI mode.

FIGURE 19-1: MSSP BLOCK DIAGRAM (SPI MODE)
19.3.1 REGISTERS

Each MSSP module has four registers for SPI mode operation. These are:

- MSSPx Control Register 1 (SSPxCON1)
- MSSPx Status Register (SSPxSTAT)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- 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.

SSPxSR is the shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from. In receive operations, SSPxSR and SSPxBUF together create a double-buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

REGISTER 19-1: SSPxSTAT: MSSPx STATUS REGISTER (SPI MODE)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMP</td>
<td>CKE(1)</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’

-n = Value at POR  ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

bit 7  SMP: 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.

bit 6  CKE: SPI Clock Select bit(1)

1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state

bit 5  D/A: Data/Address bit

Used in I2C mode only.

bit 4  P: Stop bit

Used in I2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3  S: Start bit

Used in I2C mode only.

bit 2  R/W: Read/Write Information bit

Used in I2C mode only.

bit 1  UA: Update Address bit

Used in I2C mode only.

bit 0  BF: Buffer Full Status bit (Receive mode only)

1 = Receive complete, SSPxBUF is full
0 = Receive not complete, SSPxBUF is empty

Note 1: Polarity of clock state is set by the CKP bit (SSPxCON1<4>).
REGISTER 19-2:  SSPxCON1: MSSPx CONTROL REGISTER 1 (SPI MODE)

<table>
<thead>
<tr>
<th>bit 7</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCOL: Write Collision Detect bit</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>
</tr>
<tr>
<td>0 = No collision</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6</th>
<th>SSPOV: Receive Overflow Indicator bit(1)</th>
</tr>
</thead>
<tbody>
<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. Overflow can only occur in Slave mode. The user must read the SSPxBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).</td>
<td></td>
</tr>
<tr>
<td>0 = No overflow</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5</th>
<th>SSPEN: Master Synchronous Serial Port Enable bit(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = Enables serial port and configures SCKx, SDOx, SDIx and SSx as serial port pins</td>
<td></td>
</tr>
<tr>
<td>0 = Disables serial port and configures these pins as I/O port pins</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 4</th>
<th>CKP: Clock Polarity Select bit</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 3-0</th>
<th>SSPM3:SSPM0: Master Synchronous Serial Port Mode Select bits(3)</th>
</tr>
</thead>
<tbody>
<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>0100 = SPI Slave mode, clock = SCKx pin, SSx pin control enabled</td>
<td></td>
</tr>
<tr>
<td>0011 = SPI Master mode, clock = TMR2 output/2</td>
<td></td>
</tr>
<tr>
<td>0010 = SPI Master mode, clock = Fosc/64</td>
<td></td>
</tr>
<tr>
<td>0001 = SPI Master mode, clock = Fosc/16</td>
<td></td>
</tr>
<tr>
<td>0000 = SPI Master mode, clock = Fosc/4</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: Bit combinations not specifically listed here are either reserved or implemented in I²C mode only.
19.3.2 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPxCON1<5:0> and SSPxSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCKx is the clock output)
- Slave mode (SCKx is the clock input)
- Clock Polarity (Idle state of SCKx)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCKx)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

Each MSSP module 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 (SSPxSTAT<0>) and the interrupt flag bit, SSPxFIF, 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 (SSPxCON1<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPxBUF register completed 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 (SSPxSTAT<0>), indicates when SSPxBUF has been loaded with the received data (transmission is complete). When the SSPxBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 19-1 shows the loading of the SSPxBUF (SSPxSR) for data transmission.

The SSPxSR is not directly readable or writable and can only be accessed by addressing the SSPxBUF register. Additionally, the SSPxSTAT register indicates the various status conditions.

19.3.3 OPEN-DRAIN OUTPUT OPTION

The drivers for the SDOx output and SCKx clock pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor, and allows the output to communicate with external circuits without the need for additional level shifters. For more information, see Section 10.1.4 “Open-Drain Outputs”.

The open-drain output option is controlled by the SPI2OD and SPI1OD bits (ODCON3<1:0>). Setting an SPIxOD bit configures the SDOx and SCKx pins for the corresponding module for open-drain operation.

The ODCON3 register shares the same address as the T1CON register. The ODCON3 register is accessed by setting the ADSHR bit in the WDTCON register (WDTCON<4>).

EXAMPLE 19-1: LOADING THE SSP1BUF (SSP1SR) REGISTER

| LOOP | BTFSS SSP1STAT, BF ;Has data been received (transmit complete)? |
| BRA | LOOP ;No |
| MOVF SSP1BUF, W ;WREG reg = contents of SSP1BUF |
| MOVWF RXDATA ;Save in user RAM, if data is meaningful |
| MOVF TXDATA, W ;W reg = contents of TXDATA |
| MOVWF SSP1BUF ;New data to xmit |
19.3.4 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPxCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize the SSPxCON registers and then set the SSPEN 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 is automatically controlled by the SPI module
- SDOx must have the TRISC<5> or TRISD<4> bit cleared
- SCKx (Master mode) must have the TRISC<3> or TRISD<6> bit cleared
- SCKx (Slave mode) must have the TRISC<3> or TRISD<6> bit set
- SSx must have the TRISF<7> or TRISD<7> 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.

19.3.5 TYPICAL CONNECTION

Figure 19-2 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCKx signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data – Slave sends dummy data
- Master sends data – Slave sends data
- Master sends dummy data – Slave sends data

**FIGURE 19-2: SPI MASTER/SLAVE CONNECTION**
19.3.6 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCKx. The master determines when the slave (Processor 1, Figure 19-2) 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). This could be useful in receiver applications as a “Line Activity Monitor” mode.

The clock polarity is selected by appropriately programming the CKP bit (SSPxCON1<4>). This then, would give waveforms for SPI communication as shown in Figure 19-3, Figure 19-5 and Figure 19-6, 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

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 19-3 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 19-3: SPI MODE WAVEFORM (MASTER MODE)](image-url)
19.3.7 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCKx. When the last bit is latched, the SSPxIF interrupt flag bit is set.

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. When a byte is received, the device can be configured to wake-up from Sleep.

19.3.8 SLAVE SELECT SYNCHRONIZATION

The SSx pin allows a Synchronous Slave mode. The SPI must be in Slave mode with the SSx pin control enabled (SSPxCON1<3:0> = 04h). 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: If the SPI is used in Slave mode with CKE set, then the SSx pin control must be enabled. |

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 SSPEN bit.

To emulate two-wire communication, the SDOx pin can be connected to the SDIx pin. When the SPI needs to operate as a receiver, the SDOx pin can be configured as an input. This disables transmissions from the SDOx. The SDIx can always be left as an input (SDI function) since it cannot create a bus conflict.

**FIGURE 19-4: SLAVE SYNCHRONIZATION WAVEFORM**
FIGURE 19-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

<table>
<thead>
<tr>
<th>SSx</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCKx (CKP = 0, CKE = 0)</td>
<td></td>
</tr>
<tr>
<td>SCKx (CKP = 1, CKE = 0)</td>
<td></td>
</tr>
<tr>
<td>Write to SSPxBUF</td>
<td></td>
</tr>
<tr>
<td>SDOx</td>
<td></td>
</tr>
<tr>
<td>SDIx (SMP = 0)</td>
<td></td>
</tr>
<tr>
<td>Input Sample (SMP = 0)</td>
<td></td>
</tr>
<tr>
<td>SSPxIF Interrupt Flag</td>
<td></td>
</tr>
<tr>
<td>SSPxSR to SSPxBUF</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 19-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)

<table>
<thead>
<tr>
<th>SSx</th>
<th>Not Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCKx (CKP = 0, CKE = 1)</td>
<td></td>
</tr>
<tr>
<td>SCKx (CKP = 1, CKE = 1)</td>
<td></td>
</tr>
<tr>
<td>Write to SSPxBUF</td>
<td></td>
</tr>
<tr>
<td>SDOx</td>
<td></td>
</tr>
<tr>
<td>SDIx (SMP = 0)</td>
<td></td>
</tr>
<tr>
<td>Input Sample (SMP = 0)</td>
<td></td>
</tr>
<tr>
<td>SSPxIF Interrupt Flag</td>
<td></td>
</tr>
<tr>
<td>SSPxSR to SSPxBUF</td>
<td></td>
</tr>
</tbody>
</table>

Next Q4 Cycle after Q2↓
19.3.9 OPERATION IN POWER-MANAGED MODES

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.

In Idle modes, a clock is provided to the peripherals. That clock can be from the primary clock source, the secondary clock (Timer1 oscillator) or the INTOSC source. See Section 2.3 “Clock Sources and Oscillator Switching” for additional information.

In most cases, the speed that the master clocks SPI data is not important; however, this should be evaluated for each system.

If MSSP interrupts are enabled, they can wake the controller from Sleep mode, or one of the Idle modes, when the master completes sending data. If an exit from Sleep or Idle mode is not desired, MSSP interrupts should be disabled.

If 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 any power-managed mode and data to be shifted into the SPI Transmit/Receive Shift register. When all 8 bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

19.3.10 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

19.3.11 BUS MODE COMPATIBILITY

Table 19-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

<table>
<thead>
<tr>
<th>Standard SPI Mode Terminology</th>
<th>Control Bits State</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKP</td>
<td>CKE</td>
</tr>
<tr>
<td>0, 0</td>
<td>0</td>
</tr>
<tr>
<td>0, 1</td>
<td>0</td>
</tr>
<tr>
<td>1, 0</td>
<td>1</td>
</tr>
<tr>
<td>1, 1</td>
<td>1</td>
</tr>
</tbody>
</table>

There is also an SMP bit which controls when the data is sampled.

19.3.12 SPI CLOCK SPEED AND MODULE INTERACTIONS

Because MSSP1 and MSSP2 are independent modules, they can operate simultaneously at different data rates. Setting the SSPM3:SSPM0 bits of the SSPxCON1 register determines the rate for the corresponding module.

An exception is when both modules use Timer2 as a time base in Master mode. In this instance, any changes to the Timer2 module’s operation will affect both MSSP modules equally. If different bit rates are required for each module, the user should select one of the other three time base options for one of the modules.
### TABLE 19-2: 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>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR3</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMR4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>58</td>
</tr>
<tr>
<td>TRISC</td>
<td>TRISC7</td>
<td>TRISC6</td>
<td>TRISC5</td>
<td>TRISC4</td>
<td>TRISC3</td>
<td>TRISC2</td>
<td>TRISC1</td>
<td>TRISC0</td>
<td>58</td>
</tr>
<tr>
<td>TRISD</td>
<td>TRISD7</td>
<td>TRISD6</td>
<td>TRISD5</td>
<td>TRISD4</td>
<td>TRISD3</td>
<td>TRISD2</td>
<td>TRISD1</td>
<td>TRISD0</td>
<td>58</td>
</tr>
<tr>
<td>TRISF</td>
<td>TRISF7</td>
<td>TRISF6</td>
<td>TRISF5</td>
<td>TRISF4</td>
<td>TRISF3</td>
<td>TRISF2</td>
<td>—</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>SSP1BUF</td>
<td>MSSP1 Receive Buffer/Transmit Register</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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>56, 59</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>56, 59</td>
</tr>
<tr>
<td>SSP2BUF</td>
<td>MSSP2 Receive Buffer/Transmit Register</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCON3(1)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>SPI2OD</td>
<td>SPI1OD</td>
</tr>
</tbody>
</table>

**Legend:** Shaded cells are not used by the MSSP module in SPI mode.

**Note 1:** Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.
19.4  \( \text{I}^2\text{C} \) Mode

The MSSP module in \( \text{I}^2\text{C} \) mode fully implements all master and slave functions (including general call support), and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:
- Serial Clock (SCLx) – RC3/SCK1/SCL1 or RD6/SCK2/SCL2
- Serial Data (SDAx) – RC4/SDI1/SDA1 or RD5/SDI2/SDA2

The user must configure these pins as inputs by setting the associated TRIS bits.

FIGURE 19-7: MSSP BLOCK DIAGRAM

(I\textsuperscript{2}C™ MODE)

19.4.1 REGISTERS

The MSSP module has six registers for \( \text{I}^2\text{C} \) operation. These are:
- MSSPx Control Register 1 (SSPxCON1)
- MSSPx Control Register 2 (SSPxCON2)
- MSSPx Status Register (SSPxSTAT)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSPx Shift Register (SSPxSR) – Not directly accessible
- MSSPx Address Register (SSPxADD)
- \( \text{I}^2\text{C} \) Slave Address Mask Register (SSPxMSK)

SSPxCON1, SSPxCON2 and SSPxSTAT are the control and status registers in \( \text{I}^2\text{C} \) mode operation. The SSPxCON1 and SSPxCON2 registers are readable and writeable. The lower 6 bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

SSPxSR is the shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from.

SSPxADD contains the slave device address when the MSSP is configured in \( \text{I}^2\text{C} \) Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPxADD act as the Baud Rate Generator reload value.

SSPxMSK holds the slave address mask value when the module is configured for 7-bit Address Masking mode. While it is a separate register, it shares the same SFR address as SSPxADD; it is only accessible when the SSPM3:SSPM0 bits are specifically set to permit access. Additional details are provided in Section 19.4.3.4 “7-Bit Address Masking Mode”.

In receive operations, SSPxSR and SSPxBUF together, create a double-buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxF interrupt is set. During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.
### REGISTER 19-3: **SSPxSTAT: MSSPx STATUS REGISTER (I²C™ MODE)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>R/W</th>
<th>Readable</th>
<th>Writable</th>
<th>Unimplemented</th>
<th>Value at POR</th>
<th>'1'</th>
<th>'0'</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><strong>SMP:</strong> Slew Rate Control bit</td>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>U</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td><strong>In Master or Slave mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Slew rate control enabled for High-Speed mode (400 kHz)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>CKE:</strong> SMBus Select bit</td>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>U</td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td><strong>In Master or Slave mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Enable SMBus specific inputs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Disable SMBus specific inputs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>D/A:</strong> Data/Address bit</td>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>U</td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td><strong>In Master mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>In Slave mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Indicates that the last byte received or transmitted was data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Indicates that the last byte received or transmitted was address</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>P:</strong> Stop bit(1)</td>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>U</td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td><strong>In Slave mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Indicates that a Stop bit has been detected last</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Stop bit was not detected last</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>S:</strong> Start bit(1)</td>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>U</td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td><strong>In Slave mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Indicates that a Start bit has been detected last</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Start bit was not detected last</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>R/W:</strong> Read/Write Information bit(2,3)</td>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>U</td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td><strong>In Slave mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Read</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>In Master mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Transmit is in progress</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Transmit is not in progress</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td><strong>UA:</strong> Update Address bit (10-Bit Slave mode only)</td>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>U</td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td><strong>In Slave mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = Indicates that the user needs to update the address in the SSPxADD register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Address does not need to be updated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td><strong>BF:</strong> Buffer Full Status bit</td>
<td>R/W</td>
<td>R</td>
<td>R</td>
<td>U</td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td><strong>In Transmit mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = SSPxBUF is full</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = SSPxBUF is empty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>In Receive mode:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 = SSPxBUF is full (does not include the ACK and Stop bits)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = SSPxBUF is empty (does not include the ACK and Stop bits)</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’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

**Note 1:** This bit is cleared on Reset and when SSPEN is cleared.
**Note 2:** 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.
**Note 3:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSPx is in Active mode.
## REGISTER 19-4: SSPxCON1: MSSPx CONTROL REGISTER 1 (I²C™ MODE)

<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>SSPOV: Receive Overflow Indicator bit</td>
<td>SSPEN: Master Synchronous Serial Port Enable bit</td>
<td>CKP: SCKx Release Control bit</td>
<td>SSPM3:SSPM0: Master Synchronous Serial Port Mode Select bits</td>
</tr>
</tbody>
</table>

### WCOL: Write Collision Detect bit
- **In Master Transmit mode:**
  - 1 = A write to the SSPxBUF register was attempted while the I²C conditions were not valid for a transmission to be started (must be cleared in software)
  - 0 = No collision
- **In Slave Transmit mode:**
  - 1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
  - 0 = No collision
- **In Receive mode (Master or Slave modes):**
  - This is a “don’t care” bit.

### SSPOV: Receive Overflow Indicator bit
- **In Receive mode:**
  - 1 = A byte is received while the SSPxBUF register is still holding the previous byte (must be cleared in software)
  - 0 = No overflow
- **In Transmit mode:**
  - This is a "don’t care" bit in Transmit mode.

### SSPEN: Master Synchronous Serial Port Enable bit
- 1 = Enables the serial port and configures the SDAx and SCLx pins as the serial port pins
- 0 = Disables serial port and configures these pins as I/O port pins

### CKP: SCKx Release Control bit
- **In Slave mode:**
  - 1 = Releases clock
  - 0 = Holds clock low (clock stretch), used to ensure data setup time
- **In Master mode:**
  - Unused in this mode.

### SSPM3:SSPM0: Master Synchronous Serial Port Mode Select bits

- **Note 1:** When enabled, the SDAx and SCLx pins must be configured as inputs.
- **Note 2:** Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.
- **Note 3:** When SSPM3:SSPM0 = 1001, any reads or writes to the SSPxADD SFR address actually accesses the SSPxMSK register.
- **Note 4:** This mode is only available when 7-Bit Address Masking mode is selected (MSSPMSK Configuration bit is '1').
## REGISTER 19-5: SSPxCON2: MSSPx CONTROL REGISTER 2 (I²C™ MASTER MODE)

<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>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’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

### Description:
- **bit 7**
  - **GCEN**: General Call Enable bit
  - Unused in Master mode.

- **bit 6**
  - **ACKSTAT**: Acknowledge Status bit (Master Transmit mode only)
  - 1 = Acknowledge was not received from slave
  - 0 = Acknowledge was received from slave

- **bit 5**
  - **ACKDT**: Acknowledge Data bit (Master Receive mode only)
  - 1 = Not Acknowledge
  - 0 = Acknowledge

- **bit 4**
  - **ACKEN**: Acknowledge Sequence Enable bit
  - 1 = Initiates 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 (Master Receive mode only)
  - 1 = Enables Receive mode for I²C
  - 0 = Receive Idle

- **bit 2**
  - **PEN**: Stop Condition Enable bit
  - 1 = Initiates Stop condition on SDAx and SCLx pins. Automatically cleared by hardware.
  - 0 = Stop condition Idle

- **bit 1**
  - **RSEN**: Repeated Start Condition Enable bit
  - 1 = Initiates Repeated Start condition on SDAx and SCLx pins. Automatically cleared by hardware.
  - 0 = Repeated Start condition Idle

- **bit 0**
  - **SEN**: Start Condition Enable bit
  - 1 = Initiates Start condition on SDAx and SCLx pins. Automatically cleared by hardware.
  - 0 = Start condition Idle

### Notes:
1. Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
2. If the I²C module is active, these bits may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).
### REGISTER 19-6: SSPxCON2: MSSPx CONTROL REGISTER 2 (I²C™ SLAVE MODE)

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>GCEN: General Call Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enables interrupt when a general call address (0000h) is received in the SSPSR</td>
</tr>
<tr>
<td>0</td>
<td>General call address disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>ACKSTAT: Acknowledge Status bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Unused in Slave mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 5-2</th>
<th>ADMSK5:ADMSK2: Slave Address Mask Select bits (5-Bit Address Masking mode)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Masking of corresponding bits of SSPxADD enabled</td>
</tr>
<tr>
<td></td>
<td>0 = Masking of corresponding bits of SSPxADD disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>ADMSK1: Slave Address Least Significant bit(s) Mask Select bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In 7-Bit Addressing mode:</td>
</tr>
<tr>
<td></td>
<td>1 = Masking of SSPxADD&lt;1&gt; only enabled</td>
</tr>
<tr>
<td></td>
<td>0 = Masking of SSPxADD&lt;1&gt; only disabled</td>
</tr>
<tr>
<td></td>
<td>In 10-Bit Addressing mode:</td>
</tr>
<tr>
<td></td>
<td>1 = Masking of SSPxADD&lt;1:0&gt; enabled</td>
</tr>
<tr>
<td></td>
<td>0 = Masking of SSPxADD&lt;1:0&gt; disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>SEN: Stretch Enable bit(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)</td>
</tr>
<tr>
<td>0</td>
<td>Clock stretching is disabled</td>
</tr>
</tbody>
</table>

**Note 1:** If the I²C module is active, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

### REGISTER 19-7: SSPxMSK: I²C™ SLAVE ADDRESS MASK REGISTER (7-BIT MASKING MODE)(1)

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>MSK7:MSK0: Slave Address Mask Select bit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 = Masking of corresponding bit of SSPxADD enabled</td>
</tr>
<tr>
<td></td>
<td>0 = Masking of corresponding bit of SSPxADD disabled</td>
</tr>
</tbody>
</table>

**Note 1:** This register shares the same SFR address as SSPxADD, and is only addressable in select MSSPx operating modes. See Section 19.4.3.4 “7-Bit Address Masking Mode” for more details.

**Note 2:** MSK0 is not used as a mask bit in 7-bit addressing.
19.4.2 OPERATION

The MSSP module functions are enabled by setting the MSSP Enable bit, SSPEN (SSPxCON1<5>).

The SSPxCON1 register allows control of the I²C operation. Four mode selection bits (SSPxCON1<3:0>) allow one of the following I²C modes to be selected:

- I²C Master mode, clock
- I²C Slave mode (7-bit address)
- I²C Slave mode (10-bit address)
- I²C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I²C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I²C Firmware Controlled Master mode, slave is Idle

Selection of any I²C mode with the SSPEN bit set forces the SCLx and SDAx pins to be open-drain, provided these pins are programmed as inputs by setting the appropriate TRISC or TRISD bits. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCLx and SDAx pins.

19.4.3 SLAVE MODE

In Slave mode, the SCLx and SDAx pins must be configured as inputs (TRISC<4:3> set). The MSSP module will override the input state with the output data when required (slave-transmitter).

The I²C Slave mode hardware will always generate an interrupt on an address match. Address masking will allow the hardware to generate an interrupt for more than one address (up to 31 in 7-bit addressing and up to 63 in 10-bit addressing). Through the mode select bits, the user can also choose to interrupt on Start and Stop bits.

When an address is matched, or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge (ACK) pulse and load the SSPxBUF register with the received value currently in the SSPxSR register.

Any combination of the following conditions will cause the MSSP module not to give this ACK pulse:

- The Buffer Full bit, BF (SSPxSTAT<0>), was set before the transfer was received.
- The overflow bit, SSPOV (SSPxCON1<6>), was set before the transfer was received.

In this case, the SSPxSR register value is not loaded into the SSPxBUF, but bit SSPxF is set. The BF bit is cleared by reading the SSPxBUF register, while bit SSPOV is cleared through software.

The SCLx clock input must have a minimum high and low for proper operation. The high and low times of the I²C specification, as well as the requirement of the MSSP module, are shown in timing parameter 100 and parameter 101.

19.4.3.1 Addressing

Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the 8 bits are shifted into the SSPxSR register. All incoming bits are sampled with the rising edge of the clock (SCLx) line. The value of register, SSPxSR<7:1>, is compared to the value of the SSPxADD register. The address is compared on the falling edge of the eighth clock (SCLx) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

1. The SSPxSR register value is loaded into the SSPxBUF register.
2. The Buffer Full bit, BF, is set.
3. An ACK pulse is generated.
4. The MSSP Interrupt Flag bit, SSPxIF, is set (and interrupt is generated, if enabled) on the falling edge of the ninth SCLx pulse.

In 10-Bit Addressing mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit R/W (SSPxSTAT<2>) must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '11110 A9 A8 0', where 'A9' and 'A8' are the two MSbs of the address. The sequence of events for 10-bit addressing is as follows, with steps 7 through 9 for the slave-transmitter:

1. Receive first (high) byte of address (bits SSPxF, BF and UA are set on address match).
2. Update the SSPxADD register with second (low) byte of address (clears bit UA and releases the SCLx line).
3. Read the SSPxBUF register (clears bit, BF) and clear flag bit, SSPxF.
4. Receive second (low) byte of address (bits SSPxF, BF and UA are set).
5. Update the SSPxADD register with the first (high) byte of address. If match releases SCLx line, this will clear bit UA.
6. Read the SSPxBUF register (clears bit BF) and clear flag bit SSPxF.
8. Receive first (high) byte of address (bits SSPxF and BF are set).
9. Read the SSPxBUF register (clears bit BF) and clear flag bit, SSPxF.
19.4.3.2 Address Masking Modes

Masking an address bit causes that bit to become a "don't care". When one address bit is masked, two addresses will be Acknowledged and cause an interrupt. It is possible to mask more than one address bit at a time, which greatly expands the number of addresses Acknowledged.

The I2C Slave behaves the same way whether address masking is used or not. However, when address masking is used, the I2C slave can Acknowledge multiple addresses and cause interrupts. When this occurs, it is necessary to determine which address caused the interrupt by checking the SSPxBUF.

The PIC18F87J11 Family of devices is capable of using two different Address Masking modes in I2C Slave operation: 5-Bit Address Masking and 7-Bit Address Masking. The Masking mode is selected at device configuration using the MSSPMSK Configuration bit. The default device configuration is 7-Bit Address Masking.

Both Masking modes, in turn, support address masking of 7-bit and 10-bit addresses. The combination of Masking modes and addresses provide different ranges of Acknowledgable addresses for each combination.

While both Masking modes function in roughly the same manner, the way they use address masks are different.

19.4.3.3 5-Bit Address Masking Mode

As the name implies, 5-Bit Address Masking mode uses an address mask of up to 5 bits to create a range of addresses to be Acknowledged, using bits 5 through 1 of the incoming address. This allows the module to Acknowledge up to 31 addresses when using 7-bit addressing, or 63 addresses with 10-bit addressing (see Example 19-2). This Masking mode is selected when the MSSPMSK Configuration bit is programmed (‘0’).

The address mask in this mode is stored in the SSPxCON2 register, which stops functioning as a control register in I2C Slave mode (Register 19-6). In 7-Bit Address Masking mode, address mask bits, ADMSK<5:1> (SSPxCON2<5:1>), mask the corresponding address bits in the SSPxADD register. For any ADMSK bits that are set (ADMSK<n> = 1), the corresponding address bit is ignored (SSPxADD<n> = x). For the module to issue an address Acknowledge, it is sufficient to match only on addresses that do not have an active address mask.

In 10-Bit Address Masking mode, bits ADMSK<5:2> mask the corresponding address bits in the SSPxADD register. In addition, ADMSK1 simultaneously masks the two LSbs of the address (SSPxADD<1:0>). For any ADMSK bits that are active (ADMSK<n> = 1), the corresponding address bit is ignored (SSPxADD<n> = x). Also note, that although in 10-Bit Address Masking mode, the upper address bits reuse part of the SSPxADD register bits. The address mask bits do not interact with those bits; they only affect the lower address bits.

Note 1: ADMSK1 masks the two Least Significant bits of the address.

2: The two Most Significant bits of the address are not affected by address masking.

**EXAMPLE 19-2: ADDRESS MASKING EXAMPLES IN 5-BIT MASKING MODE**

**7-Bit Addressing:**

SSPAADD<7:1> = A0h (1010000) (SSPAADD<0> is assumed to be '0')

ADMSK<5:1> = 00111

Addresses Acknowledged: A0h, A2h, A4h, A6h, A8h, AAh, ACh, AEh

**10-Bit Addressing:**

SSPAADD<7:0> = A0h (10100000) (The two MSb of the address are ignored in this example, since they are not affected by masking)

ADMSK<5:1> = 00111

Addresses Acknowledged: A0h, A1h, A2h, A3h, A4h, A5h, A6h, A7h, A8h, A9h, AAh, ABh, ACh, ADh, AEh, AFh
19.4.3.4 7-Bit Address Masking Mode

Unlike 5-bit masking, 7-Bit Address Masking mode uses a mask of up to 8 bits (in 10-bit addressing) to define a range of addresses than can be Acknowledged, using the lowest bits of the incoming address. This allows the module to Acknowledge up to 127 different addresses with 7-bit addressing, or 255 with 10-bit addressing (see Example 19-3). This mode is the default configuration of the module, and is selected when MSSPMSK is unprogrammed (‘1’).

The address mask for 7-Bit Address Masking mode is stored in the SSPxMSK register, instead of the SSPxCON2 register. SSPxMSK is a separate hardware register within the module, but it is not directly addressable. Instead, it shares an address in the SFR space with the SSPxADD register. To access the SSPxMSK register, it is necessary to select MSSP mode, ‘1001’ (SSPCON1<3:0> = 1001), and then read or write to the location of SSPxADD.

To use 7-Bit Address Masking mode, it is necessary to initialize SSPxMSK with a value before selecting the \(^2\)C Slave Addressing mode. Thus, the required sequence of events is:

1. Select SSPxMSK Access mode (SSPxCON2<3:0> = 1001).
2. Write the mask value to the appropriate SSPADD register address (FC8h for MSSP1, F6Eh for MSSP2).
3. Set the appropriate \(^2\)C Slave mode (SSPxCON2<3:0> = 0111 for 10-bit addressing, 0110 for 7-bit addressing).

Setting or clearing mask bits in SSPxMSK behaves in the opposite manner of the ADMSK bits in 5-Bit Address Masking mode. That is, clearing a bit in SSPxMSK causes the corresponding address bit to be masked; setting the bit requires a match in that position. SSPxMSK resets to all ‘1’ s upon any Reset condition and, therefore, has no effect on the standard MSSP operation until written with a mask value.

With 7-bit addressing, SSPxMSK<7:1> bits mask the corresponding address bits in the SSPxADD register. For any SSPxMSK bits that are active (SSPxMSK<7:1> = 0), the corresponding SSPxADD address bit is ignored (SSPxADD<7:1> = x). For the module to issue an address Acknowledge, it is sufficient to match only on addresses that do not have an active address mask.

With 10-bit addressing, SSPxMSK<7:0> bits mask the corresponding address bits in the SSPxADD register. For any SSPxMSK bits that are active (= 0), the corresponding SSPxADD address bit is ignored (SSPxADD<7:0> = x).

**Note:** The two Most Significant bits of the address are not affected by address masking.

### EXAMPLE 19-3: ADDRESS MASKING EXAMPLES IN 7-BIT MASKING MODE

**7-Bit Addressing:**

SSPxADD<7:1> = 1010 000
SSPxMSK<7:1> = 1111 001
Addresses Acknowledged = A8h, A6h, A4h, A0h

**10-Bit Addressing:**

SSPxADD<7:0> = 1010 0000
SSPxMSK<5:1> = 1111 0
Addresses Acknowledged = A8h, A6h, A4h, A0h

(The two MSb are ignored in this example since they are not affected)
19.4.3.5 Reception

When the R/W bit of the address byte is clear and an address match occurs, the R/W bit of the SSPxSTAT register is cleared. The received address is loaded into the SSPxBUF register and the SDAx line is held low (ACK).

When the address byte overflow condition exists, then the no Acknowledge (ACK) pulse is given. An overflow condition is defined as either bit, BF (SSPxSTAT<0>), is set or bit, SSPOV (SSPxCON1<6>), is set.

An MSSP interrupt is generated for each data transfer byte. The interrupt flag bit, SSPxIF, must be cleared in software. The SSPxSTAT register is used to determine the status of the byte.

If SEN is enabled (SSPxCON2<0> = 1), SCLx will be held low (clock stretch) following each data transfer. The clock must be released by setting bit, CKP (SSPxCON1<4>). See Section 19.4.4 “Clock Stretching” for more details.

19.4.3.6 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. The ACK pulse will be sent on the ninth bit and pin SCLx is held low regardless of SEN (see Section 19.4.4 “Clock Stretching” for more details). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPxBUF register which also loads the SSPxSR register. Then, pin SCLx should be enabled by setting bit, CKP (SSPxCON1<4>). 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 (Figure 19-10).

The ACK pulse from the master-receiver is latched on the rising edge of the ninth SCLx input pulse. If the SDAx line is high (not ACK), then the data transfer is complete. In this case, when the ACK is latched by the slave, the slave logic is reset and the slave monitors 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, pin SCLx must be enabled by setting bit, CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPxIF bit must be cleared in 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.
FIGURE 19-8: \textsuperscript{1}2C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)

\begin{itemize}
\item \textit{Bus master}\textsuperscript{*} terminates transfer.
\item \textit{Bus master}\textsuperscript{*} is reset because SSP\textsubscript{B}UF is not full.
\item \textit{Bus master}\textsuperscript{*} resets SSP\textsubscript{B}UF.
\item \textit{Bus master}\textsuperscript{*} is reset because SSP\textsubscript{B}UF is not full.
\item \textit{Bus master}\textsuperscript{*} does not need to return to \textit{"0", when SEN = 1.}
\end{itemize}

\textsuperscript{*} SPU = 1, \textit{Bus master}\textsuperscript{*} does not need to return to \textit{"0", when SEN = 1.}
FIGURE 19-9: \(^2\text{C}^\text{TM}\) SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01011
(RECEPTION, 7-BIT ADDRESS)

Note 1: \(x\) = Don't care (i.e., address bit can either be a '1' or a '0').

2: In this example, an address equal to A7.A6.A5.X.A3.X.X will be Acknowledged and cause an interrupt.
FIGURE 19-10: \( \text{i}^2\text{C} \text{™ SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)} \)
FIGURE 19-11: I2C™ SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01001 (RECEPTION, 10-BIT ADDRESS)

1: x = Don’t care (i.e., address bit can either be a ’1’ or a ’0’).
3: Note that the Most Significant bits of the address are not affected by the bit masking.
Figure 19-12: I2C™ Slave Mode Timing with SEN = 0 (Reception, 10-Bit Address)

- **SDAx**: Data input/output.
- **SCLx**: Clock input.
- **SSPxIF (PIR1<3> or PIR3<7>)**: Slave select input.
- **BF (SSPxSTAT<0>)**: Bit flag.
- **UA (SSPxSTAT<1>)**: Update address flag.
- **SSPOV (SSPxCON1<6>)**: SSP output buffer is full flag.
- **CKP (SSPxCON<4>)**: Clock polarity.

Clock is held low until update of SSPxADD has taken place.

- **Receive First Byte of Address**: Clock is held low until update of SSPxADD has taken place.
- **Receive Second Byte of Address**: Clock is held low until update of SSPxADD has taken place.
- **Receive Data Byte**: Clock is held low until update of SSPxADD has taken place.

- **SSPxIF (PIR1<3> or PIR3<7>)**: Cleared in software.
- **BF (SSPxSTAT<0>)**: Cleared in software.
- **UA (SSPxSTAT<1>)**: Cleared in software.
- **SSPOV (SSPxCON1<6>)**: Cleared in software.

Bus master terminates transfer.

- **ACK**: Received data byte.
- **ACK**: Received first byte of address.
- **ACK**: Received second byte of address.

SSPOV is set because SSPBUF is still full. ACK is not sent.

- **SSPOV is set because SSPBUF is still full. ACK is not sent.**
- **SSPOV is set because SSPBUF is still full. ACK is not sent.**
- **SSPOV is set because SSPBUF is still full. ACK is not sent.**
- **SSPOV is set because SSPBUF is still full. ACK is not sent.**

(CKP does not reset to '0' when SEN = 0)
FIGURE 19-13: I²C™ SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)
19.4.4 CLOCK STRETCHING

Both 7-Bit and 10-Bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPxC<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCLx pin to be held low at the end of each data receive sequence.

19.4.4.1 Clock Stretching for 7-Bit Slave Receive Mode (SEN = 1)

In 7-Bit Slave Receive mode, on the falling edge of the ninth clock at the end of the ACK sequence, if the BF bit is set, the CKP bit in the SSPxCON1 register is automatically cleared, forcing the SCLx output to be held low. The CKP bit being cleared to ‘0’ will assert the SCLx line low. The CKP bit must be set in the user’s ISR before reception is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and read the contents of the SSPxBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 19-15).

Note 1: If the user reads the contents of the SSPxBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

19.4.4.2 Clock Stretching for 10-Bit Slave Receive Mode (SEN = 1)

In 10-Bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/W bit cleared to ‘0’. The release of the clock line occurs upon updating SSPxADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

Note: If the user polls the UA bit and clears it by updating the SSPxADD register before the falling edge of the ninth clock occurs, and if the user hasn’t cleared the BF bit by reading the SSPxBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

19.4.4.3 Clock Stretching for 7-Bit Slave Transmit Mode

The 7-Bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user’s ISR must set the CKP bit before transmission is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and load the contents of the SSPxBUF before the master device can initiate another transmit sequence (see Figure 19-10).

Note 1: If the user loads the contents of SSPxBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit.

19.4.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence, which contains the high-order bits of the 10-bit address and the R/W bit set to ‘1’. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-Bit Slave Transmit mode (see Figure 19-13).
19.4.4.5 Clock Synchronization and the CKP bit

When the CKP bit is cleared, the SCLx output is forced to '0'. 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 I²C 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 I²C bus have deasserted SCLx. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCLx (see Figure 19-14).

**FIGURE 19-14:** CLOCK SYNCHRONIZATION TIMING

![CLOCK SYNCHRONIZATION TIMING Diagram](image-url)
FIGURE 19-15: I^2C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)
FIGURE 19-16: I 2C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESS)

Receive First Byte of Address

Receive Second Byte of Address

Receive Data Byte

Receive Data Byte

Receive First Byte of Address

Receive Second Byte of Address

Receive Data Byte

Receive Data Byte

Clock is held low until update of SSPxADD has taken place

Clock is held low until update of SSPxADD has taken place

Clock is held low until CKP is set to '1'

Clock is held low until update of SSPxADD has taken place

Clock is held low because ACR = 1

SSPxF (PIR1<3> or PIR3<7>)

BF (SSPxSTAT<0>)

SSPOV (SSPxCON1<6>)

SSPOV is set because SSPxBUF is still full. ACR is not sent.

SSPOV is set indicating that SSPxADD needs to be updated

CFP written to '1' in software

UC (SSPxSTAT<1>)

UC is set indicating that SSPxADD needs to be updated

UC is set indicating that SSPxADD needs to be updated

UC is set indicating that SSPxADD needs to be updated

SSPxBUF is written with contents of SSPxSR

SSPxBUF is written with contents of SSPxSR

Dummy read of SSPxBUF to clear BF flag

Dummy read of SSPxBUF to clear BF flag

Note: An update of the SSPxADD register before the falling edge of the ninth clock will have no effect on UA and UA will remain set.

Note: An update of the SSPxADD register before the falling edge of the ninth clock will have no effect on UA and UA will remain set.

Clock is held low until update of SSPxADD has taken place

Clock is held low until update of SSPxADD has taken place

Clock is held low until CKP is set to '1'

Clock is held low because ACR = 1

SSPxBUF is written with low byte of address after falling edge of ninth clock

SSPxBUF is written with high byte of address after falling edge of ninth clock

SSPOV is set because SSPxBUF is still full. ACR is not sent.

SSPOV is set indicating that SSPxADD needs to be updated

SSPxBUF is written with contents of SSPxSR

SSPxBUF is written with contents of SSPxSR

Dummy read of SSPxBUF to clear BF flag

Dummy read of SSPxBUF to clear BF flag

Note: An update of the SSPxADD register before the falling edge of the ninth clock will have no effect on UA and UA will remain set.
19.4.5 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I2C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master. 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 one of eight addresses reserved for specific purposes by the I2C protocol. It consists of all '0's with R/W = 0.

The general call address is recognized when the General Call Enable bit, GCEN, is enabled (SSPxCON2<7> set). Following a Start bit detect, 8 bits are shifted into the SSPxSR and the address is compared against the SSPxADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPxSR is transferred to the SSPxBUF, the BF flag bit is set (eighth bit), and on the falling edge of the ninth bit (ACK bit), the SSPxIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPxBUF. The value can be used to determine if the address was device-specific or a general call address.

In 10-Bit Addressing mode, the SSPxADD is required to be updated for the second half of the address to match and the UA bit is set (SSPxSTAT<1>). If the general call address is sampled when the GCEN bit is set, while the slave is configured in 10-Bit Addressing mode, then the second half of the address is not necessary, the UA bit will not be set and the slave will begin receiving data after the Acknowledge (Figure 19-17).

FIGURE 19-17: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESSING MODE)
19.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPxCON1 and by setting the SSPEN bit. In Master mode, the SCLx and SDAx lines are manipulated by the MSSP hardware if the TRIS bits are set.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I²C bus may be taken when the P bit is set, or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all I²C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

1. Assert a Start condition on SDAx and SCLx.
2. Assert a Repeated Start condition on SDAx and SCLx.
3. Write to the SSPxBUF register initiating transmission of data/address.
4. Configure the I²C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDAx and SCLx.

Note: The MSSP module, when configured in I²C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the 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.

The following events will cause the MSSP Interrupt Flag bit, SSPxIF, to be set (and MSSP interrupt, if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmitted
- Repeated Start

FIGURE 19-18: MSSP BLOCK DIAGRAM (I²C™ MASTER MODE)
19.4.6.1 \( \text{I}^2 \text{C} \) 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 \( \text{I}^2 \text{C} \) 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 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 Receiver 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.

The Baud Rate Generator, used for the SPI mode operation, is used to set the SCLx clock frequency for either 100 kHz, 400 kHz or 1 MHz \( \text{I}^2 \text{C} \) operation. See Section 19.4.7 “Baud Rate” for more details.

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the Start Enable bit, SEN (SSPxCON2<0>).
2. SSPxIF is set. The MSSP module will wait the required start time before any other operation takes place.
3. The user loads the SSPxBUF with the slave address to transmit.
4. Address is shifted out the SDAx pin until all 8 bits are transmitted.
5. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPxCON2 register (SSPxCON2<6>).
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
7. The user loads the SSPxBUF with eight bits of data.
8. Data is shifted out the SDAx pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPxCON2 register (SSPxCON2<6>).
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
11. The user generates a Stop condition by setting the Stop Enable bit, PEN (SSPxCON2<2>).
12. Interrupt is generated once the Stop condition is complete.
19.4.7  BAUD RATE

In I²C Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPxADD register (Figure 19-19). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (TCY) on the Q2 and Q4 clocks. In I²C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCLx pin will remain in its last state.

Table 19-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD.

19.4.7.1  Baud Rate and Module Interdependence

Because MSSP1 and MSSP2 are independent, they can operate simultaneously in I²C Master mode at different baud rates. This is done by using different BRG reload values for each module.

Because this mode derives its basic clock source from the system clock, any changes to the clock will affect both modules in the same proportion. It may be possible to change one or both baud rates back to a previous value by changing the BRG reload value.

FIGURE 19-19:  BAUD RATE GENERATOR BLOCK DIAGRAM

TABLE 19-3:  I²C™ CLOCK RATE w/BRG

<table>
<thead>
<tr>
<th>Fosc</th>
<th>FCY</th>
<th>FCY * 2</th>
<th>BRG Value</th>
<th>FSCL (2 Rollovers of BRG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 MHz</td>
<td>10 MHz</td>
<td>20 MHz</td>
<td>18h</td>
<td>400 kHz¹</td>
</tr>
<tr>
<td>40 MHz</td>
<td>10 MHz</td>
<td>20 MHz</td>
<td>1Fh</td>
<td>312.5 kHz</td>
</tr>
<tr>
<td>40 MHz</td>
<td>10 MHz</td>
<td>20 MHz</td>
<td>63h</td>
<td>100 kHz</td>
</tr>
<tr>
<td>16 MHz</td>
<td>4 MHz</td>
<td>8 MHz</td>
<td>09h</td>
<td>400 kHz¹</td>
</tr>
<tr>
<td>16 MHz</td>
<td>4 MHz</td>
<td>8 MHz</td>
<td>0Ch</td>
<td>308 kHz</td>
</tr>
<tr>
<td>16 MHz</td>
<td>4 MHz</td>
<td>8 MHz</td>
<td>27h</td>
<td>100 kHz</td>
</tr>
<tr>
<td>16 MHz</td>
<td>1 MHz</td>
<td>2 MHz</td>
<td>02h</td>
<td>333 kHz¹</td>
</tr>
<tr>
<td>4 MHz</td>
<td>1 MHz</td>
<td>2 MHz</td>
<td>09h</td>
<td>100 kHz</td>
</tr>
<tr>
<td>4 MHz</td>
<td>1 MHz</td>
<td>2 MHz</td>
<td>00h</td>
<td>1 MHz¹</td>
</tr>
</tbody>
</table>

Note 1:  The I²C interface does not conform to the 400 kHz I²C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.
19.4.7.2 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts 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<6:0> 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 19-20).

**FIGURE 19-20: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**

<table>
<thead>
<tr>
<th>SDAx</th>
<th>DX</th>
<th>DX – 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCLx</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCLx deasserted but slave holds SCLx low (clock arbitration)</td>
<td>SCLx allowed to transition high</td>
</tr>
<tr>
<td>BRG Value</td>
<td>03h 02h 01h 00h (hold off) 03h 02h</td>
<td>BRG decrements on Q2 and Q4 cycles</td>
</tr>
<tr>
<td>BRG Reload</td>
<td>SCLx is sampled high, reload takes place and BRG starts its count</td>
<td></td>
</tr>
</tbody>
</table>

---

DS39778C-page 254 Preliminary © 2008 Microchip Technology Inc.
19.4.8 \( \text{I}^2\text{C} \) MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Enable bit, SEN (SSPxCON2<0>). If the SDAx and SCLx pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<6:0> 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 (SSPxSTAT<3>) to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPxADD<6:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit (SSPxCON2<0>) 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:** 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.

19.4.8.1 WCOL Status Flag

If the user writes the SSPxBUF when a Start sequence is in progress, the WCOL bit is set and the contents of the buffer are unchanged (the write doesn’t occur).

**Note:** Because queueing of events is not allowed, writing to the lower 5 bits of SSPxCON2 is disabled until the Start condition is complete.

---

**FIGURE 19-21: FIRST START BIT TIMING**

<table>
<thead>
<tr>
<th>Write to SEN bit occurs here</th>
<th>SDAx = 1, SCLx = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SDAx = 1, SCLx = 1</td>
</tr>
<tr>
<td></td>
<td>TBRG</td>
</tr>
<tr>
<td></td>
<td>TBRG</td>
</tr>
<tr>
<td>Write to SSPxBUF occurs here</td>
<td>1st bit</td>
</tr>
<tr>
<td></td>
<td>2nd bit</td>
</tr>
<tr>
<td></td>
<td>1st bit</td>
</tr>
<tr>
<td></td>
<td>2nd bit</td>
</tr>
<tr>
<td></td>
<td>TBRG</td>
</tr>
</tbody>
</table>

At completion of Start bit, hardware clears SEN bit and sets SSPxIF bit.
19.4.9 \( \text{I}^2\text{C} \) MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPxCON2<1>) is programmed high and the \( \text{I}^2\text{C} \) logic module is in the idle state. 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 with the contents of SSPxADD<5:0> and begins counting. The SDAx pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out and if SDAx is sampled high, the SCLx pin will be deasserted (brought high). When SCLx is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<6:0> 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. Following this, the RSEN bit (SSPxCON2<1>) 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 (SSPxSTAT<3>) 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’.

Immediately following the SSPxIF bit getting set, the user may write the SSPxBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

19.4.9.1 WCOL Status Flag

If the user writes the SSPxBUF when a Repeated Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn’t occur).

**Note:** Because queueing of events is not allowed, writing of the lower 5 bits of SSPxCON2 is disabled until the Repeated Start condition is complete.

---

**FIGURE 19-22: REPEATED START CONDITION WAVEFORM**

Write to SSPxCON2 occurs here: SDAx = 1, SCLx = 1. S bit set by hardware on falling edge of ninth clock, end of XMIT. SCLx is not changed. 1st bit: SDAX = 1, SCLx = 1. At completion of Start bit, hardware clears RSEN bit and sets SSPxIF.
19.4.10 \(i^2C\) 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 (see data hold time specification parameter 106). SCLx is held low for one Baud Rate Generator rollover count (T BRG). Data should be valid before SCLx is released high (see data setup time specification parameter 107). When the SCLx pin is released high, it is held that way for TBRG. The data on the SDAx pin must remain stable for that duration and some hold time after the next falling edge of SCLx. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDAx. This allows the slave device being addressed to respond with an 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 ACKDT bit on the falling edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared; if not, the bit is set. After the ninth clock, the SSPxIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPxBUF, leaving SCLx low and SDAx unchanged (Figure 19-23).

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 deassert 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 (SSPxCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPxIF flag 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.

19.4.10.1 BF Status Flag

In Transmit mode, the BF bit (SSPxSTAT<0>) is set when the CPU writes to SSPxBUF and is cleared when all 8 bits are shifted out.

19.4.10.2 WCOL Status Flag

If the user writes the SSPxBUF when a transmit is already in progress (i.e., SSPxSR is still shifting out a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn’t occur) after 2 Tcy after the SSPxBUF write. If SSPxBUF is rewritten within 2 Tcy, the WCOL bit is set and SSPxBUF is updated. This may result in a corrupted transfer.

The user should verify that the WCOL bit is clear after each write to SSPxBUF to ensure the transfer is correct. In all cases, WCOL must be cleared in software.

19.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPxCON2<6>) 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.

19.4.11 \(i^2C\) MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPxCON2<3>).

The user should verify that the WCOL bit is clear after each write to SSPxBUF to ensure the transfer is correct. In all cases, WCOL must be cleared in software.

19.4.11.1 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.

19.4.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPxSR and the BF flag bit is already set from a previous reception.

19.4.11.3 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 doesn’t occur).
FIGURE 19-23: \( \text{I}^2\text{C}\)™ MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)

- **SDAx**: Data transmit pin
- **SCLx**: Clock pin
- **SSPxIF**: SSPI interrupt flag
- **BF (SSPxBUF<0>)**: Buffer full
- **SEN**: Start condition
- **ACKSTAT**: Acknowledge status
  - **ACKSTAT = 0**: No acknowledge
  - **ACKSTAT = 1**: Acknowledge
- **SSPxBUF**: Receive buffer
  - **SSPxBUF written in software from MSSP interrupt**: The receive buffer is updated by software after an MSSP interrupt.
  - **SSPxBUF written with 7-bit address and R/W, start transmit**: The address and R/W bit are written into the receive buffer to start a transmission.
- **After Start condition, SEN cleared by hardware**: The start condition is cleared internally after the transmission begins.
- **Clear in software**: The ACKSTAT bit is cleared by software.
- **From slave, clear ACKSTAT bit (SSPxC<6>)**: The ACKSTAT bit is cleared by the slave device.
- **Transmit Address to Slave**: The address is transmitted to the slave device.
- **Transmitting Data or Second Half**: Data transmission or transmission of the second half of the address.
- **Cleared in software service routine**: The ACKSTAT bit is cleared by the software service routine.
- **SSPxBUF is written in software from MSSP interrupt**: The receive buffer is updated by software after an MSSP interrupt.
- **SSPxBUF is written with 7-bit address and R/W, start transmit**: The address and R/W bit are written into the receive buffer to start a transmission.
FIGURE 19-24: I²C™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)
19.4.12 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN (SSPxCON2<4>). 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 MSSP module then goes into an inactive state (Figure 19-25).

19.4.12.1 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 doesn’t occur).

FIGURE 19-25: ACKNOWLEDGE SEQUENCE WAVEFORM

Note: TBRG = one Baud Rate Generator period.

19.4.13 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 (SSPxCON2<2>). 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 (SSPxSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPxF flag is set (Figure 19-26).

19.4.13.1 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 doesn’t occur).
19.4.14 SLEEP OPERATION
While in Sleep mode, the \( \text{I}_2\text{C} \) module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

19.4.15 EFFECTS OF A RESET
A Reset disables the MSSP module and terminates the current transfer.

19.4.16 MULTI-MASTER MODE
In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the \( \text{I}_2\text{C} \) bus may be taken when the P bit (SSPxSTAT<4>) is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the MSSP 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 in 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

19.4.17 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 = 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 19-27).

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.
FIGURE 19-27: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE

- **SDAx**: Data changes while SCLx = 0
- **SDAx line pulled low by another source**: SDAx released by master
- **Sample SDAx. While SCLx is high, data doesn’t match what is driven by the master; bus collision has occurred.**: Set bus collision interrupt (BCLxIF)
19.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDAx or SCLx is sampled low at the beginning of the Start condition (Figure 19-28).

b) SCLx is sampled low before SDAx is asserted low (Figure 19-29).

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 MSSP module is reset to its inactive state (Figure 19-28)

The Start condition begins with the SDAx and SCLx pins deasserted. When the SDAx pin is sampled high, the Baud Rate Generator is loaded from SSPxADD<6:0> and counts down to 0. 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 19-30). 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 0. 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 19-29: BUS COLLISION DURING START CONDITION (SCLx = 0)

SDAx = 0, SCLx = 1

SDAx

SCLx

SEN

BCLxIF

S

SSPxIF

FIGURE 19-30: BRG RESET DUE TO SDAx ARBITRATION DURING START CONDITION

SDAx = 0, SCLx = 1

Less than TBRG

Set S

Set SSPxIF

SDAx

SCLx

SEN

BCLxIF

S

SSPxIF
19.4.17.2 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 a low level to a 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 deasserts SDAx and the pin is allowed to float high, the BRG is loaded with SSPxADD<6:0> and counts down to 0. 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 19-31). 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 19-32).

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 19-31: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**

![Bus collision diagram](image-url)

- SDAx
- SCLx
- RSEN
- BCLxIF
- S
- SSPxIF

Sample SDAx when SCLx goes high. If SDAx = 0, set BCLxIF and release SDAx and SCLx.

Cleared in software
FIGURE 19-32: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)

SDAx
SCLx
BCLxIF
RSEN
S
SSPxIF

SCLx goes low before SDAx, set BCLxIF. Release SDAx and SCLx.
Interrupt cleared in software

TBRG

'S0'
19.4.17.3 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<6:0> 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 19-33). 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 19-34).

**FIGURE 19-33: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

![Diagram for Case 1]

**FIGURE 19-34: BUS COLLISION DURING A STOP CONDITION (CASE 2)**

![Diagram for Case 2]
## TABLE 19-4: 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/GIEH</td>
<td>PEIE/PEI</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSCF1IF</td>
<td>CM2IF</td>
<td>CM1IF</td>
<td>—</td>
<td>BCL1IF</td>
<td>LVDIF</td>
<td>TMR3IF</td>
<td>CCP2IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSCF1IE</td>
<td>CM2IE</td>
<td>CM1IE</td>
<td>—</td>
<td>BCL1IE</td>
<td>LVDIE</td>
<td>TMR3IE</td>
<td>CCP2IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR2</td>
<td>OSCF1IP</td>
<td>CM2IP</td>
<td>CM1IP</td>
<td>—</td>
<td>BCL1IP</td>
<td>LVDIP</td>
<td>TMR3IP</td>
<td>CCP2IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR3</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMR4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>58</td>
</tr>
<tr>
<td>TRISC</td>
<td>TRISC7</td>
<td>TRISC6</td>
<td>TRISC5</td>
<td>TRISC4</td>
<td>TRISC3</td>
<td>TRISC2</td>
<td>TRISC1</td>
<td>TRISC0</td>
<td>58</td>
</tr>
<tr>
<td>TRISD</td>
<td>TRISD7</td>
<td>TRISD6</td>
<td>TRISD5</td>
<td>TRISD4</td>
<td>TRISD3</td>
<td>TRISD2</td>
<td>TRISD1</td>
<td>TRISD0</td>
<td>58</td>
</tr>
<tr>
<td>SSP1BUF</td>
<td>MSSP1 Receive Buffer/Transmit Register</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSP1ADD</td>
<td>MSSP1 Address Register (I²C™ Slave mode), MSSP1 Baud Rate Reload Register (I²C Master mode)</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSP1MSK(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>56</td>
</tr>
<tr>
<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>56</td>
</tr>
<tr>
<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>56</td>
</tr>
<tr>
<td>SSP1STAT</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>56</td>
</tr>
<tr>
<td>SSP2BUF</td>
<td>MSSP2 Receive Buffer/Transmit Register</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSP2ADD</td>
<td>MSSP2 Address Register (I²C Slave mode), MSSP2 Baud Rate Reload Register (I²C Master mode)</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<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>59</td>
</tr>
<tr>
<td>SSP2CON1</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>59</td>
</tr>
<tr>
<td>SSP2CON2</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>59</td>
</tr>
<tr>
<td>SSP2STAT</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>59</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used by the MSSP module in I²C™ mode.

**Note 1:** SSPxMSK shares the same address in SFR space as SSPxADD, but is only accessible in certain I²C™ Slave operating modes in 7-bit Masking mode. See Section 19.4.3.4 “7-Bit Address Masking Mode” for more details.

**Note 2:** Alternate bit definitions for use in I²C Slave mode operations only.
20.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is one of two serial I/O modules. (Generically, the EUSART is also known as a Serial Communications Interface or SCI.) The EUSART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The Enhanced USART module implements additional features, including automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These make it ideally suited for use in Local Interconnect Network bus (LIN bus) systems.

All members of the PIC18F87J11 family are equipped with two independent EUSART modules, referred to as EUSART1 and EUSART2. They can be configured in the following modes:

- Asynchronous (full duplex) with:
  - Auto-wake-up on character reception
  - Auto-baud calibration
  - 12-bit Break character transmission
- Synchronous – Master (half duplex) with selectable clock polarity
- Synchronous – Slave (half duplex) with selectable clock polarity

The pins of EUSART1 and EUSART2 are multiplexed with the functions of PORTC (RC6/TX1/CK1 and RC7/RX1/DT1) and PORTG (RG1/TX2/CK2 and RG2/RX2/DT2), respectively. In order to configure these pins as an EUSART:

- For EUSART1:
  - bit SPEN (RCSTA1<7>) must be set (= 1)
  - bit TRISC<7> must be set (= 1)
  - bit TRISC<6> must be cleared (= 0) for Asynchronous and Synchronous Master modes
  - bit TRISC<6> must be set (= 1) for Synchronous Slave mode

- For EUSART2:
  - bit SPEN (RCSTA2<7>) must be set (= 1)
  - bit TRISG<2> must be set (= 1)
  - bit TRISG<1> must be cleared (= 0) for Asynchronous and Synchronous Master modes
  - bit TRISG<6> must be set (= 1) for Synchronous Slave mode

Note: The EUSART control will automatically reconfigure the pin from input to output as needed.

The operation of each Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTAx)
- Receive Status and Control (RCSTAx)
- Baud Rate Control (BAUDCONx)

These are detailed on the following pages in Register 20-1, Register 20-2 and Register 20-3, respectively.

Note: Throughout this section, references to register and bit names that may be associated with a specific EUSART module are referred to generically by the use of ‘x’ in place of the specific module number. Thus, “RCSTAx” might refer to the Receive Status register for either EUSART1 or EUSART2.
REGISTER 20-1: TXSTAx: TRANSMIT STATUS AND CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R-1</th>
<th>R/W-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>

Legend:

R = Readable bit  W = Writable bit  U = Unimplemented bit, read as ‘0’

- Value at POR ‘1’ = Bit is set  ‘0’ = Bit is cleared  x = Bit is unknown

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: 9th bit of Transmit Data
Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode.
## REGISTER 20-2: RCSTAx: RECEIVE STATUS AND CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>SPEN</th>
<th>RX9</th>
<th>SREN</th>
<th>CREN</th>
<th>ADDEN</th>
<th>FERR</th>
<th>OERR</th>
<th>RX9D</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R/W-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-x</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

#### bit 7
**SPEN**: Serial Port Enable bit
- 1 = Serial port enabled (configures RXx/DTx and TXx/CKx 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, enables interrupt and loads 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 9-Bit (RX9 = 0)**:
- Don’t care.

#### bit 2
**FERR**: Framing Error bit
- 1 = Framing error (can be updated by reading RCREGx register and receiving 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**: 9th bit of Received Data
This can be address/data bit or a parity bit and must be calculated by user firmware.
**REGISTER 20-3: BAUDCONx: BAUD RATE CONTROL REGISTER**

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R-1</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>U-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABDOVF</td>
<td>RCIDL</td>
<td>RXDTP</td>
<td>TXCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
</tr>
</tbody>
</table>

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

**bit 7**  
**ABDOVF**: Auto-Baud Acquisition Rollover Status bit
- **1** = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
- **0** = No BRG rollover has occurred

**bit 6**  
**RCIDL**: Receive Operation Idle Status bit
- **1** = Receive operation is Idle
- **0** = Receive operation is active

**bit 5**  
**RXDTP**: Data/Receive Polarity Select bit

**Asynchronous mode:**
- **1** = Receive data (RXx) is inverted (active-low)
- **0** = Receive data (RXx) is not inverted (active-high)

**Synchronous mode:**
- **1** = Data (DTx) is inverted (active-low)
- **0** = Data (DTx) is not inverted (active-high)

**bit 4**  
**TXCKP**: Synchronous Clock Polarity Select bit

**Asynchronous mode:**
- **1** = Idle state for transmit (TXx) is a low level
- **0** = Idle state for transmit (TXx) is a high level

**Synchronous mode:**
- **1** = Idle state for clock (CKx) is a high level
- **0** = Idle state for clock (CKx) is a low level

**bit 3**  
**BRG16**: 16-Bit Baud Rate Register Enable bit
- **1** = 16-bit Baud Rate Generator – SPBRGHx and SPBRGx
- **0** = 8-bit Baud Rate Generator – SPBRGx only (Compatible mode), SPBRGHx value ignored

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

**bit 1**  
**WUE**: Wake-up Enable bit

**Asynchronous mode:**
- **1** = EUSART will continue to sample the RXx pin – interrupt generated on falling edge; bit cleared in hardware upon following rising edge
- **0** = RXx pin not monitored or rising edge detected

**Synchronous mode:**
Unused in this mode.

**bit 0**  
**ABDEN**: Auto-Baud Detect Enable bit

**Asynchronous mode:**
- **1** = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
- **0** = Baud rate measurement disabled or completed

**Synchronous mode:**
Unused in this mode.
20.1 Baud Rate Generator (BRG)

The BRG is a dedicated, 8-bit or 16-bit generator that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode; setting the BRG16 bit (BAUDCONx<3>) selects 16-bit mode.

The SPBRGHx:SPBRGx register pair controls the period of a free-running timer. In Asynchronous mode, bits BRGH (TXSTAx<2>) and BRG16 (BAUDCONx<3>) also control the baud rate. In Synchronous mode, BRGH is ignored. Table 20-1 shows the formula for computation of the baud rate for different EUSART modes which only apply in Master mode (internally generated clock).

Given the desired baud rate and Fosc, the nearest integer value for the SPBRGHx:SPBRGx registers can be calculated using the formulas in Table 20-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 20-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 20-2. It may be advantageous to use the high baud rate (BRGH = 1) or the 16-bit BRG to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGHx:SPBRGx registers causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

20.1.1 OPERATION IN POWER-MANAGED MODES

The device clock is used to generate the desired baud rate. When one of the power-managed modes is entered, the new clock source may be operating at a different frequency. This may require an adjustment to the value in the SPBRGx register pair.

20.1.2 SAMPLING

The data on the RXx pin (either RC7/RX1/DT1 or RG2/RX2/DT2) is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RXx pin.

### TABLE 20-1: 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>SYNC</td>
<td>BRG16</td>
<td>BRGH</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>x</td>
</tr>
</tbody>
</table>

Legend:  \( x = \) Don’t care, \( n = \) value of SPBRGHx:SPBRGx register pair
EXAMPLE 20-1: CALCULATING BAUD RATE ERROR

For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, and 8-bit BRG:

\[
\text{Desired Baud Rate} = \frac{\text{Fosc}}{\left(64 \left(\text{SPBRGH}:\text{SPBRG} + 1\right)\right)}
\]

Solving for SPBRGH:SPBRG:

\[
X = \left(\frac{\text{Fosc}}{\text{Desired Baud Rate}}\right)/64 - 1
= \left(\frac{16000000}{9600}\right)/64 - 1
= \left[25.042\right] = 25
\]

Calculated Baud Rate = 16000000/(64 (25 + 1))
= 9615

Error = (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
= (9615 – 9600)/9600 = 0.16%

TABLE 20-2: REGISTERS ASSOCIATED WITH 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>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXSTAx</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>57</td>
</tr>
<tr>
<td>RCSTAx</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>57</td>
</tr>
<tr>
<td>BAUDCONx</td>
<td>ABDVF</td>
<td>RCIDL</td>
<td>RXDTP</td>
<td>TXCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>59</td>
</tr>
<tr>
<td>SPBRGHx</td>
<td>EUSART</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>SPBRGx</td>
<td>EUSART</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the BRG.
### TABLE 20-3: BAUD RATES FOR ASYNCHRONOUS MODES

<table>
<thead>
<tr>
<th>BAUD RATE (K)</th>
<th>FOSC = 40.000 MHz</th>
<th>FOSC = 20.000 MHz</th>
<th>FOSC = 10.000 MHz</th>
<th>FOSC = 8.000 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate (K)</td>
<td>% Error</td>
<td>SPBRG value (decimal)</td>
<td>Actual Rate (K)</td>
</tr>
<tr>
<td>Sync = 0, BRGH = 0, BRG16 = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sync = 0, BRGH = 1, BRG16 = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sync = 0, BRGH = 1, BRG16 = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 20-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

<table>
<thead>
<tr>
<th>BAUD RATE (K)</th>
<th>FOSC = 40.000 MHz</th>
<th>FOSC = 20.000 MHz</th>
<th>FOSC = 10.000 MHz</th>
<th>FOSC = 8.000 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate (K)</td>
<td>% Error</td>
<td>SPBRG value (decimal)</td>
<td>Actual Rate (K)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.300</td>
<td>0.00</td>
<td>8332</td>
<td>0.300</td>
</tr>
<tr>
<td>1.2</td>
<td>1.200</td>
<td>0.02</td>
<td>2082</td>
<td>1.200</td>
</tr>
<tr>
<td>2.4</td>
<td>2.402</td>
<td>0.06</td>
<td>1040</td>
<td>2.399</td>
</tr>
<tr>
<td>9.6</td>
<td>9.615</td>
<td>0.16</td>
<td>259</td>
<td>9.615</td>
</tr>
<tr>
<td>19.2</td>
<td>19.231</td>
<td>0.16</td>
<td>129</td>
<td>19.231</td>
</tr>
<tr>
<td>57.6</td>
<td>58.140</td>
<td>0.94</td>
<td>42</td>
<td>56.818</td>
</tr>
<tr>
<td>115.2</td>
<td>113.636</td>
<td>-1.36</td>
<td>21</td>
<td>113.636</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BAUD RATE (K)</th>
<th>FOSC = 4.000 MHz</th>
<th>FOSC = 2.000 MHz</th>
<th>FOSC = 1.000 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate (K)</td>
<td>% Error</td>
<td>SPBRG value (decimal)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.300</td>
<td>0.04</td>
<td>832</td>
</tr>
<tr>
<td>1.2</td>
<td>1.202</td>
<td>0.16</td>
<td>207</td>
</tr>
<tr>
<td>2.4</td>
<td>2.404</td>
<td>0.16</td>
<td>103</td>
</tr>
<tr>
<td>9.6</td>
<td>9.615</td>
<td>0.16</td>
<td>259</td>
</tr>
<tr>
<td>19.2</td>
<td>19.231</td>
<td>0.16</td>
<td>129</td>
</tr>
<tr>
<td>57.6</td>
<td>58.140</td>
<td>0.94</td>
<td>42</td>
</tr>
<tr>
<td>115.2</td>
<td>113.636</td>
<td>-1.36</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BAUD RATE (K)</th>
<th>FOSC = 40.000 MHz</th>
<th>FOSC = 20.000 MHz</th>
<th>FOSC = 10.000 MHz</th>
<th>FOSC = 8.000 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate (K)</td>
<td>% Error</td>
<td>SPBRG value (decimal)</td>
<td>Actual Rate (K)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.300</td>
<td>0.00</td>
<td>33332</td>
<td>0.300</td>
</tr>
<tr>
<td>1.2</td>
<td>1.200</td>
<td>0.02</td>
<td>8332</td>
<td>1.200</td>
</tr>
<tr>
<td>2.4</td>
<td>2.400</td>
<td>0.02</td>
<td>4165</td>
<td>2.400</td>
</tr>
<tr>
<td>9.6</td>
<td>9.606</td>
<td>0.06</td>
<td>1040</td>
<td>9.596</td>
</tr>
<tr>
<td>19.2</td>
<td>19.193</td>
<td>-0.03</td>
<td>520</td>
<td>19.231</td>
</tr>
<tr>
<td>57.6</td>
<td>57.803</td>
<td>2.12</td>
<td>122</td>
<td>57.471</td>
</tr>
<tr>
<td>115.2</td>
<td>115.21</td>
<td>-3.55</td>
<td>8</td>
<td>115.23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BAUD RATE (K)</th>
<th>FOSC = 4.000 MHz</th>
<th>FOSC = 2.000 MHz</th>
<th>FOSC = 1.000 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Actual Rate (K)</td>
<td>% Error</td>
<td>SPBRG value (decimal)</td>
</tr>
<tr>
<td>0.3</td>
<td>0.300</td>
<td>0.01</td>
<td>33332</td>
</tr>
<tr>
<td>1.2</td>
<td>1.200</td>
<td>0.04</td>
<td>8332</td>
</tr>
<tr>
<td>2.4</td>
<td>2.404</td>
<td>0.16</td>
<td>415</td>
</tr>
<tr>
<td>9.6</td>
<td>9.615</td>
<td>0.16</td>
<td>103</td>
</tr>
<tr>
<td>19.2</td>
<td>19.231</td>
<td>0.16</td>
<td>51</td>
</tr>
<tr>
<td>57.6</td>
<td>58.824</td>
<td>2.12</td>
<td>16</td>
</tr>
<tr>
<td>115.2</td>
<td>111.11</td>
<td>-3.55</td>
<td>8</td>
</tr>
</tbody>
</table>
20.1.3 AUTO-BAUD RATE DETECT

The Enhanced USART module supports the automatic detection and calibration of baud rate. This feature is active only in Asynchronous mode and while the WUE bit is clear.

The automatic baud rate measurement sequence (Figure 20-1) begins whenever a Start bit is received and the ABDEN bit is set. The calculation is self-averaging.

In the Auto-Baud Rate Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RXx signal, the RXx signal is timing the BRG. In ABD mode, the internal Baud Rate Generator is used as a counter to time the bit period of the incoming serial byte stream.

Once the ABDEN bit is set, the state machine will clear the BRG and look for a Start bit. The Auto-Baud Rate Detect must receive a byte with the value 55h (ASCII “U”, which is also the LIN bus Sync character) in order to calculate the proper bit rate. The measurement is taken over both a low and a high bit time in order to minimize any effects caused by asymmetry of the incoming signal. After a Start bit, the SPBRGx begins counting up, using the preselected clock source on the first rising edge of RXx. After eight bits on the RXx pin or the fifth rising edge, an accumulated value totalling the proper BRG period is left in the SPBRGHx:SPBRGx register pair. Once the 5th edge is seen (this should correspond to the Stop bit), the ABDEN bit is automatically cleared.

If a rollover of the BRG occurs (an overflow from FFFFh to 0000h), the event is trapped by the ABDOVF status bit (BAUDCONx<7>). It is set in hardware by BRG roll-overs and can be set or cleared by the user in software. ABD mode remains active after rollover events and the ABDEN bit remains set (Figure 20-2).

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. Note that the BRG clock will be configured by the BRG16 and BRGH bits. This allows the user to verify that no carry occurred for 8-bit modes by checking for 00h in the SPBRGHx register. Refer to Table 20-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the EUSART state machine is held in Idle. The RCxIF interrupt is set once the fifth rising edge on RXx is detected. The value in the RCREGx needs to be read to clear the RCxIF interrupt. The contents of RCREGx should be discarded.

Note 1: If the WUE bit is set with the ABDEN bit, Auto-Baud Rate Detection will occur on the byte following the Break character.

2: It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible due to bit error rates. Overall system timing and communication baud rates must be taken into consideration when using the Auto-Baud Rate Detection feature.

3: Ensure that BRG16 (BAUDCON<3>) is set, to enable the auto-baud feature.

<table>
<thead>
<tr>
<th>BRG16</th>
<th>BRGH</th>
<th>BRG Counter Clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Fosc/512</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Fosc/128</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Fosc/128</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Fosc/32</td>
</tr>
</tbody>
</table>

Note: During the ABD sequence, SPBRGx and SPBRGHx are both used as a 16-bit counter, independent of BRG16 setting.

20.1.3.1 ABD and EUSART Transmission

Since the BRG clock is reversed during ABD acquisition, the EUSART transmitter cannot be used during ABD. This means that whenever the ABDEN bit is set, TXREGx cannot be written to. Users should also ensure that ABDEN does not become set during a transmit sequence. Failing to do this may result in unpredictable EUSART operation.
FIGURE 20-1: AUTOMATIC BAUD RATE CALCULATION

<table>
<thead>
<tr>
<th>BRG Value</th>
<th>XXXXh</th>
<th>0000h</th>
<th>001Ch</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXx pin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRG Clock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABDEN bit</td>
<td>Set by User</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCxIF bit</td>
<td>(Interrupt)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCREGx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBRGx</td>
<td>XXXXh</td>
<td>1Ch</td>
<td></td>
</tr>
<tr>
<td>SPBRGHx</td>
<td>XXXXh</td>
<td>00h</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The ABD sequence requires the EUSART module to be configured in Asynchronous mode and WUE = 0.

FIGURE 20-2: BRG OVERFLOW SEQUENCE

<table>
<thead>
<tr>
<th>BRG Clock</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ABDEN bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXx pin</td>
<td>Start</td>
<td>Bit 0</td>
<td></td>
</tr>
<tr>
<td>ABDOVF bit</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRG Value</td>
<td>XXXXh</td>
<td>0000h</td>
<td>FFFFh</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0000h</td>
</tr>
</tbody>
</table>
20.2 EUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTAx<4>). In this mode, the EUSART uses standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine data bits and one Stop bit). The most common data format is 8 bits. An on-chip, dedicated 8-bit/16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The EUSART transmits and receives the LSb first. The EUSART’s transmitter and receiver are functionally independent but use the same data format and baud rate. The Baud Rate Generator produces a clock, either x16 or x64 of the bit shift rate, depending on the BRGH and BRG16 bits (TXSTAx<2> and BAUDCONx<3>). Parity is not supported by the hardware but can be implemented in software and stored as the 9th data bit.

When operating in Asynchronous mode, the EUSART module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver
- Auto-Wake-up on Sync Break Character
- 12-Bit Break Character Transmit
- Auto-Baud Rate Detection

20.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 20-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREGx. The TXREGx register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREGx register (if available).

Once the TXREGx register transfers the data to the TSR register (occurs in one Tcy), the TXREGx register is empty and the TXxFIE flag bit is set. This interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxIE. TXxFIE will be set regardless of the state of TXxIE; it cannot be cleared in software. TXxFIF is also not cleared immediately upon loading TXREGx, but becomes valid in the second instruction cycle following the load instruction. Polling TXxFIE immediately following a load of TXREGx will return invalid results.

While TXxFIF indicates the status of the TXREGx register; another bit, TRMT (TXSTAx<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty.

**Note 1:** The TSR register is not mapped in data memory, so it is not available to the user.

**Note 2:** Flag bit, TXxFIE, is set when enable bit, TXxEN, is set.

To set up an Asynchronous Transmission:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit, SPEN.
3. If interrupts are desired, set enable bit, TXxE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit, TXEN, which will also set bit, TXxFIE.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Load data to the TXREGx register (starts transmission).
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

---

**FIGURE 20-3: EUSART TRANSMIT BLOCK DIAGRAM**

[Diagram showing the EUSART transmission block diagram with TXEN, TXxFIE, TXSTAx, TXREGx, TX9D, TRMT, SPEN, and BRG16.]
**TABLE 20-5: 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>Reset Values on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMR4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>58</td>
</tr>
<tr>
<td>RCSTAx</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>57</td>
</tr>
<tr>
<td>TXREGx</td>
<td>EUSARTx Transmit Register</td>
<td>57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXSTAx</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>57</td>
</tr>
<tr>
<td>BAUDCONx</td>
<td>ABDOVF</td>
<td>RCDL</td>
<td>RXDTP</td>
<td>TXCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>59</td>
</tr>
<tr>
<td>SPBRGHx</td>
<td>EUSARTx Baud Rate Generator Register High Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBRGx</td>
<td>EUSARTx Baud Rate Generator Register Low Byte</td>
<td>59</td>
<td></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 for asynchronous transmission.
20.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 20-6. The data is received on the RXx pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

To set up an Asynchronous Reception:
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RCxIE.
4. If 9-bit reception is desired, set bit, RX9.
5. Enable the reception by setting bit, CREN.
6. Flag bit, RCxF, will be set when reception is complete and an interrupt will be generated if enabled bit, RCxIE, was set.
7. Read the RCSTAx register to get the 9th bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREGx register.
9. If any error occurred, clear the error by clearing enable bit, CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

20.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems.
To set up an Asynchronous Reception with Address Detect Enable:
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCxIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCxF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCxIE and GIE bits are set.
8. Read the RCSTAx register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREGx to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 20-6: EUSART RECEIVE BLOCK DIAGRAM
20.2.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the Baud Rate Generator is inactive and a proper byte reception cannot be performed. The auto-wake-up feature allows the controller to wake-up due to activity on the RXx/DTx line while the EUSART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCONx<1>). Once set, the typical receive sequence on RXx/DTx 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 RXx/DTx line. (This coincides with the start of a Sync Break or a Wake-up Signal character for the LIN protocol.)

Following a wake-up event, the module generates an RXIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 20-8) and asynchronously if the device is in Sleep mode (Figure 20-9). The interrupt condition is cleared by reading the RCREGx register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RXx line following the wake-up event. At this point, the EUSART module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.
20.2.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RXx/DTx, information with any state changes before the Stop bit may signal a false End-of-Character (EOC) and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all ’0’ s. This can be 00h (8 bytes) for standard RS-232 devices or 000h (12 bits) for LIN bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., HS or HSPLL 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.

20.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCxIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the EUSART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCxIF bit. The WUE bit is cleared after this when a rising edge is seen on RXx/DTx. The interrupt condition is then cleared by reading the RCREGx register. Ordinarily, the data in RCREGx will be dummy data and should be discarded.

The fact that the WUE bit has been cleared (or is still set) and the RCxIF flag is set should not be used as an indicator of the integrity of the data in RCREGx. Users should consider implementing a parallel method in firmware to verify received data integrity.

To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**FIGURE 20-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION**

**FIGURE 20-9: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**

*Note 1:* The EUSART remains in Idle while the WUE bit is set.

*Note 1:* If the wake-up event requires long oscillator warm-up time, the auto-clear of the WUE bit can occur before the oscillator is ready. This sequence should not depend on the presence of Q clocks.

*Note 2:* The EUSART remains in Idle while the WUE bit is set.
20.2.5 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. The Break character transmit consists of a Start bit, followed by twelve ‘0’ bits and a Stop bit. The Frame Break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift Register is loaded with data. Note that the value of data written to TXREGx 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).

Note that the data value written to the TXREGx for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 20-10 for the timing of the Break character sequence.

20.2.5.1 Break and Sync Transmit Sequence

The following sequence will send 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 set up the Break character.
3. Load the TXREGx with a dummy character to initiate transmission (the value is ignored).
4. Write ‘55h’ to TXREGx to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREGx becomes empty, as indicated by the TXxIF, the next data byte can be written to TXREGx.

20.2.6 RECEIVING A BREAK CHARACTER

The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in Section 20.2.4 “Auto-Wake-up on Sync Break Character”. By enabling this feature, the EUSART will sample the next two transitions on RXx/DTx, cause an RCxIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABDEN bit once the TXxIF interrupt is observed.

FIGURE 20-10: SEND BREAK CHARACTER SEQUENCE

![Send Break Character Sequence Diagram](image-url)
20.3 EUSART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTAx<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit, SYNC (TXSTAx<4>). In addition, enable bit, SPEN (RCSTAx<7>), is set in order to configure the TXx and RXx pins to CKx (clock) and DTx (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CKx line. Clock polarity is selected with the TXCKP bit (BAUDCONx<4>). Setting TXCKP sets the Idle state on CKx as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

20.3.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in Figure 20-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREGx. The TXREGx register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREGx (if available).

Once the TXREGx register transfers the data to the TSR register (occurs in one TCY), the TXREGx is empty and the TXxIF flag bit is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxEI. TXxEI is set regardless of the state of enable bit, TXxEI; it cannot be cleared in software. It will reset only when new data is loaded into the TXREGx register.

While flag bit, TXxEI, indicates the status of the TXREGx register, another bit, TRMT (TXSTAx<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user must poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

To set up a Synchronous Master Transmission:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit, TXxEI.
4. If 9-bit transmission is desired, set bit, TX9.
5. Enable the transmission by setting bit, TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Start transmission by loading data to the TXREGx register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

Note: Sync Master mode, SPBRGx = 0, continuous transmission of two 8-bit words. This example is equally applicable to EUSART2 (RG1/TX2/CK2 and RG2/RX2/DT2).
FIGURE 20-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

TABLE 20-7: 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>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>JPR1</td>
<td>PMP1P</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR3</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMR4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>58</td>
</tr>
<tr>
<td>JPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>58</td>
</tr>
<tr>
<td>RCSTAx</td>
<td>SPEN</td>
<td>RX9</td>
<td>SREN</td>
<td>CREN</td>
<td>ADEN</td>
<td>FERR</td>
<td>OERR</td>
<td>RX9D</td>
<td>57</td>
</tr>
<tr>
<td>TXREGx</td>
<td>EUSARTx Transmit Register</td>
<td>57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXSTAx</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>57</td>
</tr>
<tr>
<td>BAUDCONx</td>
<td>ABDVF</td>
<td>RCIDL</td>
<td>RXDTP</td>
<td>TXCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>59</td>
</tr>
<tr>
<td>SPBRGHx</td>
<td>EUSARTx Baud Rate Generator Register High Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBRGx</td>
<td>EUSARTx Baud Rate Generator Register Low Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous master transmission.
20.3.2 EUSART SYNCHRONOUS MASTER RECEPTION

Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTAx<5>) or the Continuous Receive Enable bit, CREN (RCSTAx<4>). Data is sampled on the RXx pin on the falling edge of the clock.

If enable bit, SREN, is set, only a single word is received. If enable bit, CREN, is set, the reception is continuous until CREN is cleared. If both bits are set, then CREN takes precedence.

To set up a Synchronous Master Reception:
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRG16 bit, 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 enable bit, RCxIE.
5. If 9-bit reception is desired, set bit, RX9.
6. If a single reception is required, set bit, SREN. For continuous reception, set bit, CREN.
7. Interrupt flag bit, RCxIF, will be set when reception is complete and an interrupt will be generated if the enable bit, RCxIE, was set.
8. Read the RCSTAx register to get the 9th bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREGx register.
10. If any error occurred, clear the error by clearing bit CREN.
11. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 20-13: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)

Note: Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0. This example is equally applicable to EUSART2 (RG1/TX2/CK2 and RG2/RX2/DT2).
### 20.4 EUSART Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit, CSRC (TXSTAx<7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the CKx pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

#### 20.4.1 EUSART SYNCHRONOUS SLAVE TRANSMISSION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep mode.

If two words are written to the TXREGx and then the `SLEEP` instruction is executed, the following will occur:

- a) The first word will immediately transfer to the TSR register and transmit.
- b) The second word will remain in the TXREGx register.
- c) Flag bit, TXxF, will not be set.
- d) When the first word has been shifted out of TSR, the TXREGx register will transfer the second word to the TSR and flag bit, TXxF, will now be set.
- e) If enable bit, TXxE, is set, the interrupt will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Transmission:

1. Enable the synchronous slave serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. Clear bits, CREN and SREN.
3. If interrupts are desired, set enable bit, TXxIE.
4. If 9-bit transmission is desired, set bit, TX9.
5. Enable the transmission by setting enable bit, TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

<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/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMRIIF</td>
<td>TMRIIF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMRIE</td>
<td>TMRIE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMRIIP</td>
<td>TMRIIP</td>
<td>58</td>
</tr>
<tr>
<td>PIR3</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMRIIF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMRIE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMRIIP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>58</td>
</tr>
<tr>
<td>RCSTAx</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>57</td>
</tr>
<tr>
<td>RCREgx</td>
<td>EUSARTx Receive Register</td>
<td>57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXSTAx</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>57</td>
</tr>
<tr>
<td>BAUDCONx</td>
<td>ABDOVF</td>
<td>RCIDL</td>
<td>RXDTP</td>
<td>TXCKP</td>
<td>BRG16</td>
<td>WUE</td>
<td>ABDEN</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>SPBRGx</td>
<td>EUSARTx Baud Rate Generator Register High Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPBRGnx</td>
<td>EUSARTx Baud Rate Generator Register Low Byte</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous master reception.
20.4.2 EUSART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep, or any Idle mode and bit, SREN, which is a “don’t care” in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep or any Idle mode, then a word may be received while in this low-power mode. Once the word is received, the RSR register will transfer the data to the RCREGx register. If the RCxIE enable bit is set, the interrupt generated will wake the chip from the low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. If interrupts are desired, set enable bit, RCxIE.
3. If 9-bit reception is desired, set bit, RX9.
4. To enable reception, set enable bit, CREN.
5. Flag bit, RCxIF, will be set when reception is complete. An interrupt will be generated if enable bit, RCxIE, was set.
6. Read the RCSTAx register to get the 9th bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREGx register.
8. If any error occurred, clear the error by clearing bit, CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.
### TABLE 20-10: 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>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INTOIE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INTOIF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1IF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR3</td>
<td>SSP2IF</td>
<td>BCL2IF</td>
<td>RC2IF</td>
<td>TX2IF</td>
<td>TMR4IF</td>
<td>CCP5IF</td>
<td>CCP4IF</td>
<td>CCP3IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE3</td>
<td>SSP2IE</td>
<td>BCL2IE</td>
<td>RC2IE</td>
<td>TX2IE</td>
<td>TMR4IE</td>
<td>CCP5IE</td>
<td>CCP4IE</td>
<td>CCP3IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR3</td>
<td>SSP2IP</td>
<td>BCL2IP</td>
<td>RC2IP</td>
<td>TX2IP</td>
<td>TMR4IP</td>
<td>CCP5IP</td>
<td>CCP4IP</td>
<td>CCP3IP</td>
<td>58</td>
</tr>
<tr>
<td>RCSTAx</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>57</td>
</tr>
<tr>
<td>RCREGx</td>
<td>EUSARTx Receive Register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57</td>
</tr>
<tr>
<td>TXSTAx</td>
<td>CSRC</td>
<td>TX9</td>
<td>TXEN</td>
<td>SYN</td>
<td>SENDB</td>
<td>BRGH</td>
<td>TRMT</td>
<td>TX9D</td>
<td>57</td>
</tr>
<tr>
<td>BAUDCONx</td>
<td>ABDOVF</td>
<td>RCIDL</td>
<td>RXDTP</td>
<td>TXCKP</td>
<td>BRG16</td>
<td>—</td>
<td>WUE</td>
<td>ABDEN</td>
<td>59</td>
</tr>
<tr>
<td>SPBRGHx</td>
<td>EUSARTx Baud Rate Generator Register High Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>SPBRGx</td>
<td>EUSARTx Baud Rate Generator Register Low Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>59</td>
</tr>
</tbody>
</table>

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave reception.
21.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has 11 inputs for the 64-pin devices and 15 for the 80-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

The module has six registers:
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Port Configuration Register 2 (ANCON0)
- A/D Port Configuration Register 1 (ANCON1)
- A/D Result Registers (ADRESH and ADRESL)

The ADCON0 register, shown in Register 21-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 21-2, configures the A/D clock source, programmed acquisition time and justification.

REGISTER 21-1: ADCON0: A/D CONTROL REGISTER 0(1)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCFG1</td>
<td>VCFG0</td>
<td>CHS3</td>
<td>CHS2</td>
<td>CHS1</td>
<td>CHS0</td>
<td>GO/DONE</td>
<td>ADON</td>
</tr>
</tbody>
</table>

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

bit 7-6  VCFG1: Voltage Reference Configuration bit (VREF- source)  
1 = VREF- (AN2)  
0 = AVSS

bit  VCFG0: Voltage Reference Configuration bit (VREF+ source)  
1 = VREF+ (AN3)  
0 = AVDD

bit 5-2  CHS3:CHS0: Analog Channel Select bits  
0000 = Channel 00 (AN0)  
0001 = Channel 01 (AN1)  
0010 = Channel 02 (AN2)  
0011 = Channel 03 (AN3)  
0100 = Channel 04 (AN4)  
0101 = Unused  
0110 = Channel 06 (AN6)  
0111 = Channel 07 (AN7)  
1000 = Channel 08 (AN8)  
1001 = Channel 09 (AN9)  
1010 = Channel 10 (AN10)  
1011 = Channel 11 (AN11)  
1100 = Channel 12 (AN12)(2,3)  
1101 = Channel 13 (AN13)(2,3)  
1110 = Channel 14 (AN14)(2,3)  
1111 = Channel 15 (AN15)(2,3)

bit 1  GO/DONE: A/D Conversion Status bit  
When ADON = 1:  
1 = A/D conversion in progress  
0 = A/D Idle

bit 0  ADON: A/D On bit  
1 = A/D Converter module is enabled  
0 = A/D Converter module is disabled

Note 1: Default (legacy) SFR at this address, available when WDTCON<4> = 0.
2: These channels are not implemented on 64-pin devices.
3: Performing a conversion on unimplemented channels will return random values.
REGISTER 21-2: A/D CONTROL REGISTER 1\(^{(1)}\)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADFM</td>
<td>ADCAL</td>
<td>ACQT2</td>
<td>ACQT1</td>
<td>ACQT0</td>
<td>ADCS2</td>
<td>ADCS1</td>
<td>ADCS0</td>
</tr>
<tr>
<td>bit 7</td>
<td></td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

bit 7  ADFM: A/D Result Format Select bit
   1 = Right justified
   0 = Left justified

bit 6  ADCAL: A/D Calibration bit
   1 = Calibration is performed on next A/D conversion
   0 = Normal A/D Converter operation (no conversion is performed)

bit 5-3  ACQT2:ACQT0: A/D Acquisition Time Select bits
   111 = 20 TAD
   110 = 16 TAD
   101 = 12 TAD
   100 =  8 TAD
   011 =  6 TAD
   010 =  4 TAD
   001 =  2 TAD
   000 =  0 TAD\(^{(1)}\)

bit 2-0  ADCS2:ADCS0: A/D Conversion Clock Select bits
   111 = FRC (clock derived from A/D RC oscillator)\(^{(2)}\)
   110 = Fosc/64
   101 = Fosc/16
   100 = Fosc/4
   011 = FRC (clock derived from A/D RC oscillator)\(^{(2)}\)
   010 = Fosc/32
   001 = Fosc/8
   000 = Fosc/2

Note 1: Default (legacy) SFR at this address, available when WDTCON<4> = 0.

2: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.
The ANCON0 and ANCON1 registers are used to configure the operation of the I/O pin associated with each analog channel. Setting any one of the PCFG bits configures the corresponding pin to operate as a digital only I/O. Clearing a bit configures the pin to operate as an analog input for either the A/D Converter or the comparator module; all digital peripherals are disabled, and digital inputs read as ‘0’. As a rule, I/O pins that are multiplexed with analog inputs default to analog operation on device Resets.

ANCON0 and ANCON1 are shared address SFRs, and use the same addresses as the ADCON1 and ADCON0 registers. The ANCON registers are accessed by setting the ADSHR bit (WDTCON<4>). See Section 5.3.4.1 “Shared Address SFRs” for more information.

**REGISTER 21-3: ANCON0: A/D PORT CONFIGURATION REGISTER 2**

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>PCFG7</td>
<td>PCFG6</td>
</tr>
</tbody>
</table>

**Legend:**

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

**Register Description:**

- **PCFG7:PCFG6**: Analog Port Configuration bits (AN7 and AN6)
  - 1 = Pin configured as a digital port
  - 0 = Pin configured as an analog channel; digital input disabled and reads ‘0’
- **bit 5**: Unimplemented: Read as ‘0’
- **bit 4-0**: PCFG4:PCFG0: Analog Port Configuration bits (AN4 through AN0)
  - 1 = Pin configured as a digital port
  - 0 = Pin configured as an analog channel; digital input disabled and reads ‘0’

**REGISTER 21-4: ANCON1: A/D PORT CONFIGURATION REGISTER 1**

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0</td>
<td>R/W-0</td>
</tr>
<tr>
<td>PCFG15^{(1)}</td>
<td>PCFG14^{(1)}</td>
</tr>
</tbody>
</table>

**Legend:**

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

**Register Description:**

- **PCFG15:PCFG8**: Analog Port Configuration bits (AN15 through AN8)
  - 1 = Pin configured as a digital port
  - 0 = Pin configured as an analog channel; digital input disabled and reads ‘0’

**Note 1:** AN15 through AN12 are implemented only on 80-pin devices. For 64-pin devices, the corresponding PCFGx bits are still implemented for these channels, but have no effect.
The analog reference voltage is software selectable to either the device’s positive and negative supply voltage (AVDD and AVSS), or the voltage level on the RA3/AN3/VREF+ and RA2/AN2/VREF- pins.

The A/D Converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D’s internal RC oscillator.

The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

Each port pin associated with the A/D Converter can be configured as an analog input or as a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH:ADRESL register pair, the GO/DONE bit (ADCON0<1>) is cleared and A/D Interrupt Flag bit, ADIF, is set.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion in progress is aborted. The value in the ADRESH:ADRESL register pair is not modified for a Power-on Reset. These registers will contain unknown data after a Power-on Reset.

The block diagram of the A/D module is shown in Figure 21-1.

**FIGURE 21-1: A/D BLOCK DIAGRAM**

---

**Note 1**: Channels AN15 through AN12 are not available on 64-pin devices.

**Note 2**: I/O pins have diode protection to Vdd and Vss.
After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see Section 21.1 “A/D Acquisition Requirements”. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/DONE bit and the actual start of the conversion.

The following steps should be followed to do an A/D conversion:

1. Configure the A/D module:
   - Configure the required ADC pins as analog pins using ANCON0, ANCON1
   - Set voltage reference using ADCON0
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON1)
   - Select A/D conversion clock (ADCON1)
   - Turn on A/D module (ADCON0)

2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit

3. Wait the required acquisition time (if required).

4. Start conversion:
   - Set GO/DONE bit (ADCON0<1>)

5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/DONE bit to be cleared OR
   - Waiting for the A/D interrupt

6. Read A/D Result registers (ADRESH:ADRESL); clear bit, ADIF, if required.

7. For next conversion, go to step 1 or step 2, as required. The A/D conversion time per bit is defined as TAD. A minimum wait of 2 TAD is required before next acquisition starts.

FIGURE 21-2: ANALOG INPUT MODEL

Legend:
- \( \text{CPIN} \) = input capacitance
- \( \text{VT} \) = threshold voltage
- \( \text{ILEAKAGE} \) = leakage current at the pin due to various junctions
- \( \text{RIC} \) = interconnect resistance
- \( \text{SS} \) = sampling switch
- \( \text{CHOLD} \) = sample/hold capacitance (from DAC)
- \( \text{RSS} \) = sampling switch resistance
21.1 A/D Acquisition Requirements

For the A/D Converter 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 21-2. 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). The source impedance affects the offset voltage at the analog input (due to pin leakage current). The maximum recommended impedance for analog sources is 2.5 kΩ. After the analog input channel is selected (changed), the channel must be sampled for at least the minimum acquisition time before starting a conversion.

Note: When the conversion is started, the holding capacitor is disconnected from the input pin.

To calculate the minimum acquisition time, Equation 21-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Equation 21-3 shows the calculation of the minimum required acquisition time, TACQ. This calculation is based on the following application system assumptions:

- CHOLD = 25 pF
- Rs = 2.5 kΩ
- Conversion Error ≤ 1/2 LSb
- VDD = 3V → Rss = 2kΩ
- Temperature = 85°C (system max.)

EQUATION 21-1: ACQUISITION TIME

\[ T_{ACQ} = T_{AMP} + T_c + T_{COFF} \]

EQUATION 21-2: A/D MINIMUM CHARGING TIME

\[ V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \times (1 - e^{-t/(CHOLD(RIC + RSS + RS))}) \]

or

\[ T_c = -(CHOLD)(RIC + RSS + RS) \ln(1/2048) \]

EQUATION 21-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

\[ T_{ACQ} = T_{AMP} + T_c + T_{COFF} \]

\[ T_{AMP} = 0.2 \mu s \]

\[ T_{COFF} = (T_{COFF}) \ln(1/2048) \mu s \]

\[ = (T_{COFF}) \ln(1/2048) \mu s \]

\[ = (T_{COFF}) \ln(1/2048) \mu s \]

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, TCOFF = 0 ms.

\[ T_c = -(CHOLD)(RIC + RSS + RS) \ln(1/2048) \mu s \]

\[ = -(25 pF)(1 kΩ + 2 kΩ + 2.5 kΩ) \ln(0.0004883) \mu s \]

\[ = 1.05 \mu s \]

\[ T_{ACQ} = 0.2 \mu s + 1.05 \mu s + 1.2 \mu s \]

\[ = 2.45 \mu s \]
21.2 Selecting and Configuring Automatic Acquisition Time

The ADCON1 register allows the user to select an acquisition time that occurs each time the GO/DONE bit is set.

When the GO/DONE bit is set, sampling is stopped and a conversion begins. The user is responsible for ensuring the required acquisition time has passed between selecting the desired input channel and setting the GO/DONE bit. This occurs when the ACQT2:ACQT0 bits (ADCON1<5:3>) remain in their Reset state ('000') and is compatible with devices that do not offer programmable acquisition times.

If desired, the ACQT bits can be set to select a programmable acquisition time for the A/D module. When the GO/DONE bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Since the acquisition time is programmed, there may be no need to wait for an acquisition time between selecting a channel and setting the GO/DONE bit.

In either case, when the conversion is completed, the GO/DONE bit is cleared, the ADIF flag is set and the A/D begins sampling the currently selected channel again. If an acquisition time is programmed, there is nothing to indicate if the acquisition time has ended or if the conversion has begun.

21.3 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 11 TAD per 10-bit conversion. The source of the A/D conversion clock is software selectable.

There are seven possible options for TAD:
- 2 Tosc
- 4 Tosc
- 8 Tosc
- 16 Tosc
- 32 Tosc
- 64 Tosc
- Internal RC Oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be as short as possible but greater than the minimum TAD (see parameter 130 in Table 27-30 for more information).

Table 21-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

<table>
<thead>
<tr>
<th>AD Clock Source (TAD)</th>
<th>Maximum Device Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Tosc</td>
<td>2.86 MHz</td>
</tr>
<tr>
<td>4 Tosc</td>
<td>5.71 MHz</td>
</tr>
<tr>
<td>8 Tosc</td>
<td>11.43 MHz</td>
</tr>
<tr>
<td>16 Tosc</td>
<td>22.86 MHz</td>
</tr>
<tr>
<td>32 Tosc</td>
<td>40.00 MHz</td>
</tr>
<tr>
<td>64 Tosc</td>
<td>40.00 MHz</td>
</tr>
<tr>
<td>RC(2)</td>
<td>1.00 MHz(1)</td>
</tr>
</tbody>
</table>

Note 1: The RC source has a typical TAD time of 4 μs.
2: For device frequencies above 1 MHz, the device must be in Sleep mode for the entire conversion or the A/D accuracy may be out of specification.

21.4 Configuring Analog Port Pins

The ANCON0, ANCON1, TRISA, TRISF and TRISH registers control the operation of the A/D port pins. The port pins needed as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS3:CHS0 bits and the TRIS bits.

Note 1: When reading the PORT register, all pins configured as analog input channels will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will be accurately converted.
2: Analog levels on any pin defined as a digital input may cause the digital input buffer to consume current out of the device's specification limits.
21.5 A/D Conversions

Figure 21-3 shows the operation of the A/D Converter after the GO/DONE bit has been set and the ACQT2:ACQT0 bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 21-4 shows the operation of the A/D Converter after the GO/DONE bit has been set, the ACQT2:ACQT0 bits are set to '010' and selecting a 4 TAD acquisition time before the conversion starts.

Clearing the GO/DONE bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. This means the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers).

After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

Note: The GO/DONE bit should NOT be set in the same instruction that turns on the A/D.

21.6 Use of the ECCP2 Trigger

An A/D conversion can be started by the “Special Event Trigger” of the ECCP2 module. This requires that the CCP2M3:CCP2M0 bits (CCP2CON<3:0>) be programmed as ‘1011’ and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/DONE bit will be set, starting the A/D acquisition and conversion, and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH/ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition period is either timed by the user, or an appropriate TACQ time is selected before the Special Event Trigger sets the GO/DONE bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the Special Event Trigger will be ignored by the A/D module but will still reset the Timer1 (or Timer3) counter.

FIGURE 21-3: A/D CONVERSION TAD CYCLES (ACQT2:ACQT0 = 000, TACQ = 0)

FIGURE 21-4: A/D CONVERSION TAD CYCLES (ACQT2:ACQT0 = 010, TACQ = 4 TAD)
21.7 A/D Converter Calibration

The A/D Converter in the PIC18F87J11 family of devices includes a self-calibration feature which compensates for any offset generated within the module. The calibration process is automated and is initiated by setting the ADCAL bit (ADCON1<6>). The next time the GO/DONE bit is set, the module will perform a "dummy" conversion (that is, with reading none of the input channels) and store the resulting value internally to compensate for the offset. Thus, subsequent offsets will be compensated. An example of a calibration routine is shown in Example 21-1.

The calibration process assumes that the device is in a relatively steady-state operating condition. If A/D calibration is used, it should be performed after each device Reset or if there are other major changes in operating conditions.

21.8 Operation in Power-Managed Modes

The selection of the automatic acquisition time and A/D conversion clock is determined in part by the clock source and frequency while in a power-managed mode.

If the A/D is expected to operate while the device is in a power-managed mode, the ACQT2:ACQT0 and ADCS2:ADCS0 bits in ADCON1 should be updated in accordance with the power-managed mode clock that will be used. After the power-managed mode is entered (either of the power-managed Run modes), an A/D acquisition or conversion may be started. Once an acquisition or conversion is started, the device should continue to be clocked by the same power-managed mode clock source until the conversion has been completed. If desired, the device may be placed into the corresponding power-managed Idle mode during the conversion.

If the power-managed mode clock frequency is less than 1 MHz, the A/D RC clock source should be selected.

Operation in the Sleep mode requires the A/D RC clock source to be selected. If bits, ACQT2:ACQT0, are set to '000' and a conversion is started, the conversion will be delayed one instruction cycle to allow execution of the SLEEP instruction and entry to Sleep mode. The IDLEN and SCS bits in the OSCCON register must have already been cleared prior to starting the conversion.

EXAMPLE 21-1: SAMPLE A/D CALIBRATION ROUTINE

```
BSF WDTCON,ADSHR ;Enable write/read to the shared SFR
BCF ANCON0,PCFG0 ;Make Channel 0 analog
BCF WDTCON,ADSHR ;Disable write/read to the shared SFR
BSF ADCON0,ADON ;Enable A/D module
BSF ADCON1,ADON ;Enable Calibration
BSF ADCON1,ADCAL ;Start a dummy A/D conversion
CALIBRATION ;
BTFSC ADCON0,GO ;Wait for the dummy conversion to finish
BRA CALIBRATION ;
BCF ADCON1,ADCAL ;Calibration done, turn off calibration enable
;Proceed with the actual A/D conversion
```
### TABLE 21-2: SUMMARY OF A/D REGISTERS

<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/GIEH</td>
<td>PIEE/GIEL</td>
<td>TMRO1E</td>
<td>INT0E</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR1</td>
<td>PMPIF</td>
<td>ADIF</td>
<td>RC1IF</td>
<td>TX1IF</td>
<td>SSP1F</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE1</td>
<td>PMPIE</td>
<td>ADIE</td>
<td>RC1IE</td>
<td>TX1IE</td>
<td>SSP1IE</td>
<td>CCP1IE</td>
<td>TMR2IE</td>
<td>TMR1IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR1</td>
<td>PMPIP</td>
<td>ADIP</td>
<td>RC1IP</td>
<td>TX1IP</td>
<td>SSP1IP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
<td>58</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSCFIF</td>
<td>CM2IF</td>
<td>CM1IF</td>
<td>—</td>
<td>BCL1IF</td>
<td>LVDIF</td>
<td>TMR3IF</td>
<td>CCP2IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSCFIE</td>
<td>CM2IE</td>
<td>CM1IE</td>
<td>—</td>
<td>BCL1IE</td>
<td>LVDIE</td>
<td>TMR3IE</td>
<td>CCP2IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR2</td>
<td>OSCFIP</td>
<td>CM2IP</td>
<td>CM1IP</td>
<td>—</td>
<td>BCL1IP</td>
<td>LVDIP</td>
<td>TMR3IP</td>
<td>CCP2IP</td>
<td>58</td>
</tr>
<tr>
<td>ADRESH</td>
<td>A/D Result Register High Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57</td>
</tr>
<tr>
<td>ADRESL</td>
<td>A/D Result Register Low Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57</td>
</tr>
<tr>
<td>ADCON0(2)</td>
<td>VCFG1</td>
<td>VCFG0</td>
<td>CHS3</td>
<td>CHS3</td>
<td>CHS1</td>
<td>CHS0</td>
<td>GO/DONE</td>
<td>ADON</td>
<td>57</td>
</tr>
<tr>
<td>ANCON0(3)</td>
<td>PCFG7</td>
<td>PCFG6</td>
<td>—</td>
<td>PCFG4</td>
<td>PCFG3</td>
<td>PCFG2</td>
<td>PCFG1</td>
<td>PCFG0</td>
<td>57</td>
</tr>
<tr>
<td>ADCON1(2)</td>
<td>ADFM</td>
<td>ADCAL</td>
<td>ACQT2</td>
<td>ACQT1</td>
<td>ACQT0</td>
<td>ADCS2</td>
<td>ADCS1</td>
<td>ADCS0</td>
<td>57</td>
</tr>
<tr>
<td>ANCON1(3)</td>
<td>PCFG15</td>
<td>PCFG14</td>
<td>PCFG13</td>
<td>PCFG12</td>
<td>PCFG11</td>
<td>PCFG10</td>
<td>PCFG9</td>
<td>PCFG8</td>
<td>57</td>
</tr>
<tr>
<td>CCP2CON</td>
<td>P2M1</td>
<td>P2M0</td>
<td>DC2B1</td>
<td>DC2B0</td>
<td>CPP2M3</td>
<td>CPP2M2</td>
<td>CPP2M1</td>
<td>CPP2M0</td>
<td>57</td>
</tr>
<tr>
<td>PORTA</td>
<td>RA7(4)</td>
<td>RA6(4)</td>
<td>RA5</td>
<td>RA4</td>
<td>RA3</td>
<td>RA2</td>
<td>RA1</td>
<td>RA0</td>
<td>59</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7(4)</td>
<td>TRISA6(4)</td>
<td>TRISA5</td>
<td>TRISA4</td>
<td>TRISA3</td>
<td>TRISA2</td>
<td>TRISA1</td>
<td>TRISA0</td>
<td>58</td>
</tr>
<tr>
<td>PORTF</td>
<td>RF7</td>
<td>RF6</td>
<td>RF5</td>
<td>RF4</td>
<td>RF3</td>
<td>RF2</td>
<td>RF1</td>
<td>—</td>
<td>59</td>
</tr>
<tr>
<td>TRISF</td>
<td>TRISF7</td>
<td>TRISF6</td>
<td>TRISF5</td>
<td>TRISF4</td>
<td>TRISF3</td>
<td>TRISF2</td>
<td>TRISF1</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>PORTH</td>
<td>RH7</td>
<td>RH6</td>
<td>RH5</td>
<td>RH4</td>
<td>RH3</td>
<td>RH2</td>
<td>RH1</td>
<td>RH0</td>
<td>59</td>
</tr>
<tr>
<td>TRISH</td>
<td>TRISH7</td>
<td>TRISH6</td>
<td>TRISH5</td>
<td>TRISH4</td>
<td>TRISH3</td>
<td>TRISH2</td>
<td>TRISH1</td>
<td>TRISH0</td>
<td>58</td>
</tr>
</tbody>
</table>

**Legend:** 
— = unimplemented, read as ‘0’. Shaded cells are not used for A/D conversion.

**Note 1:** This register is not implemented on 64-pin devices.

**Note 2:** Default (legacy) SFR at this address, available when WDTCON<4> = 0.

**Note 3:** Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.

**Note 4:** These bits are only available in select oscillator modes (FOSC2 Configuration bit = 0); otherwise, they are unimplemented.
22.0 COMPARATOR MODULE

The analog comparator module contains two comparators that can be independently configured in a variety of ways. The inputs can be selected from the analog inputs and two internal voltage references. The digital outputs are available at the pin level and can also be read through the control register. Multiple output and interrupt event generation are also available. A generic single comparator from the module is shown in Figure 22-1.

Key features of the module include:
- Independent comparator control
- Programmable input configuration
- Output to both pin and register levels
- Programmable output polarity
- Independent interrupt generation for each comparator with configurable interrupt-on-change

22.1 Registers

The CMxCON registers (Register 22-1) select the input and output configuration for each comparator, as well as the settings for interrupt generation.

The CMSTAT register (Register 22-2) provides the output results of the comparators. The bits in this register are read-only.

FIGURE 22-1: COMPARATOR SIMPLIFIED BLOCK DIAGRAM

Note 1: Available in 80-pin devices only.
2: Implemented in Comparator 2 only.
REGISTER 22-1:  CMxCON: COMPARATORx CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CON</td>
<td>COE</td>
<td>CPOL</td>
<td>EVPOL1</td>
<td>EVPOL0</td>
<td>CREF</td>
<td>CCH1</td>
<td>CCH0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>bit 7</th>
<th>CON: Comparator Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Comparator is enabled</td>
</tr>
<tr>
<td>0</td>
<td>Comparator is disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 6</th>
<th>COE: Comparator Output Enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Comparator output is present on the CxOUT pin</td>
</tr>
<tr>
<td>0</td>
<td>Comparator output is internal only</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 5</th>
<th>CPOL: Comparator Output Polarity Select bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Comparator output is inverted</td>
</tr>
<tr>
<td>0</td>
<td>Comparator output is not inverted</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 4-3</th>
<th>EVPOL1:EVPOL0: Interrupt Polarity Select bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Interrupt generation on any change of the output(1)</td>
</tr>
<tr>
<td>10</td>
<td>Interrupt generation only on high-to-low transition of the output</td>
</tr>
<tr>
<td>01</td>
<td>Interrupt generation only on low-to-high transition of the output</td>
</tr>
<tr>
<td>00</td>
<td>Interrupt generation is disabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 2</th>
<th>CREF: Comparator Reference Select bit (non-inverting input)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Non-inverting input connects to internal CVREF voltage</td>
</tr>
<tr>
<td>0</td>
<td>Non-inverting input connects to CxIN() pin</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>bit 1-0</th>
<th>CCH1:CCH0: Comparator Channel Select bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Inverting input of comparator connects to V()R</td>
</tr>
<tr>
<td>10</td>
<td>Inverting input of comparator connects to CxIND pin(2)</td>
</tr>
<tr>
<td>01</td>
<td>Inverting input of comparator connects to CxINC pin(2)</td>
</tr>
<tr>
<td>00</td>
<td>Inverting input of comparator connects to CxINB pin</td>
</tr>
</tbody>
</table>

**Note 1:** The CM\(\)xIF is automatically set any time this mode is selected and must be cleared by the application after the initial configuration.

**2:** Available in 80-pin devices only.
REGISTER 22-2: CMSTAT: COMPARATOR OUTPUT STATUS REGISTER

| bit 7-2 | Unimplemented: Read as '0' |
| bit 1-0 | COUT2:COUT1: Comparator x Status bits |
| bit 1-0 | Unimplemented: Read as '0' |

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

| bit 7-2 | Unimplemented: Read as '0' |
| bit 1-0 | COUT2:COUT1: Comparator x Status bits |
| bit 1-0 | Unimplemented: Read as '0' |

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as '0'
- -n = Value at POR
- '1' = Bit is set
- '0' = Bit is cleared
- x = Bit is unknown
22.2 Comparator Operation

A single comparator is shown in Figure 22-2, along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 22-2 represent the uncertainty due to input offsets and response time.

FIGURE 22-2: SINGLE COMPARATOR

22.3 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level. 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 to a comparator input change. Otherwise, the maximum delay of the comparators should be used (see Section 27.0 “Electrical Characteristics”).

22.4 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 22-3. Since the analog pins are connected to a digital output, they have reverse biased 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 condition may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

FIGURE 22-3: COMPARATOR ANALOG INPUT MODEL

Legend:
- CPIN = Input Capacitance
- VT = Threshold Voltage
- ILEAKAGE = Leakage Current at the pin due to various junctions
- RIC = Interconnect Resistance
- RS = Source Impedance
- VA = Analog Voltage
22.5 Comparator Control and Configuration

Each comparator has up to eight possible combinations of inputs: up to four external analog inputs, and one of two internal voltage references.

Both comparators allow a selection of the signal from pin, CxINA, or the voltage from the comparator reference (CVREF) on the non-inverting channel. This is compared to either CxINB, CxINC, CxIND or the microcontroller’s fixed internal reference voltage (VIRV, 1.2V nominal) on the inverting channel. The comparator inputs and outputs are tied to fixed I/O pins, defined in Table 22-1. The available configurations and their corresponding bit settings are shown in Figure 22-1.

TABLE 22-1: COMPARATOR INPUTS AND OUTPUTS

<table>
<thead>
<tr>
<th>Comparator</th>
<th>Input or Output</th>
<th>I/O Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C1INA (VIN+)</td>
<td>RF6</td>
</tr>
<tr>
<td>1</td>
<td>C1INB (VIN-)</td>
<td>RF5</td>
</tr>
<tr>
<td>1</td>
<td>C1INC (VIN-)</td>
<td>RH6(1)</td>
</tr>
<tr>
<td>1</td>
<td>C1OUT</td>
<td>RF2</td>
</tr>
<tr>
<td>1</td>
<td>C2INA (VIN+)</td>
<td>RF4</td>
</tr>
<tr>
<td>1</td>
<td>C2INB (VIN-)</td>
<td>RF3</td>
</tr>
<tr>
<td>1</td>
<td>C2INC (VIN-)</td>
<td>RH4(1)</td>
</tr>
<tr>
<td>1</td>
<td>C2IND (VIN-)</td>
<td>RH5(1)</td>
</tr>
<tr>
<td>1</td>
<td>C2OUT</td>
<td>RF1</td>
</tr>
</tbody>
</table>

Note 1: Available in 80-pin devices only.

22.5.1 Comparator Configurations in 64-Pin and 80-Pin Devices

In PIC18F87J11 family devices, the C and D input channels for both comparators are linked to pins in PORTH and cannot be reassigned to alternate analog inputs. Because of this, 64-pin devices offer a total of 4 different configurations for each comparator. In contrast, 80-pin devices offer a choice of 6 configurations for Comparator 1, and 8 configurations for Comparator 2. The configurations shown in Figure 22-1 are footnoted to indicate where they are not available.

22.5.2 COMPARATOR ENABLE AND OUTPUT SELECTION

The comparator module also allows the selection of an internally generated voltage reference (CVREF) from the comparator voltage reference module. This module is described in more detail in Section 23.0 "Comparator Voltage Reference Module". The reference from the Comparator Voltage Reference module is only available when CREF = 1. In this mode, the internal voltage reference is applied to the comparator’s VIN+ pin.

Note: The comparator input pin selected by CCH1:CH0 must be configured as an input by setting both the corresponding TRISF or TRISH bit, and the corresponding PCFG bit in the ANCON1 register.

22.5.1.1 Comparator Configurations in 64-Pin and 80-Pin Devices

In PIC18F87J11 family devices, the C and D input channels for both comparators are linked to pins in PORTH and cannot be reassigned to alternate analog inputs. Because of this, 64-pin devices offer a total of 4 different configurations for each comparator. In contrast, 80-pin devices offer a choice of 6 configurations for Comparator 1, and 8 configurations for Comparator 2. The configurations shown in Figure 22-1 are footnoted to indicate where they are not available.

22.5.2 COMPARATOR ENABLE AND OUTPUT SELECTION

The comparator outputs are read through the CMSTAT register. The CMSTAT<0> reads the Comparator 1 output and CMSTAT<1> reads the Comparator 2 output. These bits are read-only.

The comparator outputs may also be directly output to the RF1 and RF2 I/O pins by setting the COE bit (CMxCON<6>). When enabled, multiplexors in the output path of the pins switch to the output of the comparator. The TRISF<1:2> bits still function as the digital output enable for the RF1 and RF2 pins while in this mode.

By default, the comparator’s output is at logic high whenever the voltage on VIN+ is greater than on VIN-. The polarity of the comparator outputs can be inverted using the CPOL bit (CMxCON<5>).

The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications, as discussed in Section 22.2 "Comparator Operation".

Note: The comparator input pin selected by CCH1:CH0 must be configured as an input by setting both the corresponding TRISF or TRISH bit, and the corresponding PCFG bit in the ANCON1 register.
FIGURE 22-4:  COMPARATOR I/O CONFIGURATIONS

<table>
<thead>
<tr>
<th>Comparator Type</th>
<th>Configuration</th>
<th>Pin Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparator Off</td>
<td>CON = 0, CREF = x,</td>
<td>CxOUT pin</td>
</tr>
<tr>
<td></td>
<td>CCH1:CCH0 = xx</td>
<td></td>
</tr>
<tr>
<td>Comparator CxINB &gt; CxINA Compare</td>
<td>CON = 1, CREF = 0,</td>
<td>CxOUT pin</td>
</tr>
<tr>
<td></td>
<td>CCH1:CCH0 = 00</td>
<td></td>
</tr>
<tr>
<td>Comparator CxIND &gt; CxINA Compare</td>
<td>CON = 1, CREF = 0,</td>
<td>CxOUT pin</td>
</tr>
<tr>
<td></td>
<td>CCH1:CCH0 = 01</td>
<td></td>
</tr>
<tr>
<td>Comparator CxINB &gt; CVREF Compare</td>
<td>CON = 1, CREF = 1,</td>
<td>CxOUT pin</td>
</tr>
<tr>
<td></td>
<td>CCH1:CCH0 = 00</td>
<td></td>
</tr>
<tr>
<td>Comparator CxIND &gt; CVREF Compare</td>
<td>CON = 1, CREF = 1,</td>
<td>CxOUT pin</td>
</tr>
<tr>
<td></td>
<td>CCH1:CCH0 = 01</td>
<td></td>
</tr>
<tr>
<td>Comparator VIRV &gt; CVREF Compare</td>
<td>CON = 1, CREF = 1,</td>
<td>CxOUT pin</td>
</tr>
<tr>
<td></td>
<td>CCH1:CCH0 = 01</td>
<td></td>
</tr>
</tbody>
</table>

Legend:  
VIRV = Fixed Interval Reference Voltage (1.2V nominal), CVREF = Comparator Voltage Reference module output.
Configurations are available on both Comparators 1 and 2 in all package sizes unless otherwise noted.

Note 1: Configuration is available in 80-pin devices only.
Note 2: Configuration is available in Comparator 2 only (80-pin devices).
22.6 Comparator Interrupts

The comparator interrupt flag is set whenever any of the following occurs:

- Low-to-high transition of the comparator output
- High-to-low transition of the comparator output
- Any change in the comparator output

The comparator interrupt selection is done by the EVPOL1:EVPOL0 bits in the CMxCON register (CMxCON<4:3>).

In order to provide maximum flexibility, the output of the comparator may be inverted using the CPOL bit in the CMxCON register (CMxCON<5>). This is functionally identical to reversing the inverting and non-inverting inputs of the comparator for a particular mode.

An interrupt is generated on the low-to-high or high-to-low transition of the comparator output. This mode of interrupt generation is dependent on EVPOL<1:0> in the CMxCON register. If EVPOL<1:0> = 01 or 10, the interrupt is generated on a low-to-high or high-to-low transition of the comparator output. Once the interrupt is generated, it is required to clear the interrupt flag by software.

When EVPOL<1:0> = 11, the comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMSTAT<1:0>, to determine the actual change that occurred. The CMxIF bits (PIR2<6:5>) are the Comparator Interrupt Flags. The CMxIF bits must be reset by clearing them. Since it is also possible to write a ‘1’ to this register, a simulated interrupt may be initiated. Table 22-2 shows the interrupt generation with respect to comparator input voltages and EVPOL bit settings.

Both the CMxIE bits (PIE2<6:5>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit (INTCON<7>) must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMxIF bits will still be set if an interrupt condition occurs.

<table>
<thead>
<tr>
<th>CPOL</th>
<th>EVPOL&lt;1:0&gt;</th>
<th>Comparator Input Change</th>
<th>COUTx Transition</th>
<th>Interrupt Generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>( \text{Vin}^+ &gt; \text{Vin}^- )</td>
<td>Low-to-High</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{Vin}^+ &lt; \text{Vin}^- )</td>
<td>High-to-Low</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>( \text{Vin}^+ &gt; \text{Vin}^- )</td>
<td>Low-to-High</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{Vin}^+ &lt; \text{Vin}^- )</td>
<td>High-to-Low</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>( \text{Vin}^+ &gt; \text{Vin}^- )</td>
<td>Low-to-High</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{Vin}^+ &lt; \text{Vin}^- )</td>
<td>High-to-Low</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>( \text{Vin}^+ &gt; \text{Vin}^- )</td>
<td>Low-to-High</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{Vin}^+ &lt; \text{Vin}^- )</td>
<td>High-to-Low</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>00</td>
<td>( \text{Vin}^+ &gt; \text{Vin}^- )</td>
<td>High-to-Low</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{Vin}^+ &lt; \text{Vin}^- )</td>
<td>Low-to-High</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>( \text{Vin}^+ &gt; \text{Vin}^- )</td>
<td>High-to-Low</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{Vin}^+ &lt; \text{Vin}^- )</td>
<td>Low-to-High</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>( \text{Vin}^+ &gt; \text{Vin}^- )</td>
<td>High-to-Low</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{Vin}^+ &lt; \text{Vin}^- )</td>
<td>Low-to-High</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>( \text{Vin}^+ &gt; \text{Vin}^- )</td>
<td>High-to-Low</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{Vin}^+ &lt; \text{Vin}^- )</td>
<td>Low-to-High</td>
<td>Yes</td>
</tr>
</tbody>
</table>
22.7 Comparator Operation During Sleep

When a comparator is active and the device is placed in Sleep mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will wake-up the device from Sleep mode when enabled. Each operational comparator will consume additional current. To minimize power consumption while in Sleep mode, turn off the comparators (CON = 0) before entering Sleep. If the device wakes up from Sleep, the contents of the CMxCON register are not affected.

22.8 Effects of a Reset

A device Reset forces the CMxCON registers to their Reset state. This forces both comparators and the voltage reference to the OFF state.

---

### TABLE 22-3: 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>Reset Values on Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/GIEL</td>
<td>TMR0IE</td>
<td>INT0IE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INT0IF</td>
<td>RBIF</td>
<td>55</td>
</tr>
<tr>
<td>PIR2</td>
<td>OSCFIF</td>
<td>CM2IF</td>
<td>CM1IF</td>
<td>—</td>
<td>BCL1IF</td>
<td>LVDIF</td>
<td>TMR3IF</td>
<td>CCP2IF</td>
<td>58</td>
</tr>
<tr>
<td>PIE2</td>
<td>OSCFIE</td>
<td>CM2IE</td>
<td>CM1IE</td>
<td>—</td>
<td>BCL1E</td>
<td>LVDIE</td>
<td>TMR3IE</td>
<td>CCP2IE</td>
<td>58</td>
</tr>
<tr>
<td>IPR2</td>
<td>OSCFIP</td>
<td>CM2IP</td>
<td>CM1IP</td>
<td>—</td>
<td>BCL1P</td>
<td>LVDIP</td>
<td>TMR3IP</td>
<td>CCP2IP</td>
<td>58</td>
</tr>
<tr>
<td>CM1CON</td>
<td>CON</td>
<td>COE</td>
<td>CPOL</td>
<td>EVPOL1</td>
<td>EVPOL0</td>
<td>CREF</td>
<td>CCH1</td>
<td>CCH0</td>
<td>56</td>
</tr>
<tr>
<td>CM2CON</td>
<td>CON</td>
<td>COE</td>
<td>CPOL</td>
<td>EVPOL1</td>
<td>EVPOL0</td>
<td>CREF</td>
<td>CCH1</td>
<td>CCH0</td>
<td>56</td>
</tr>
<tr>
<td>CMSTAT</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>COUT2</td>
<td>COUT1  56</td>
</tr>
<tr>
<td>CVRCON(2)</td>
<td>CVREN</td>
<td>CVROE</td>
<td>CVRR</td>
<td>CVRSS</td>
<td>CVR3</td>
<td>CVR2</td>
<td>CVR1</td>
<td>CVR0</td>
<td>59</td>
</tr>
<tr>
<td>ANCON(2)</td>
<td>PCFG15</td>
<td>PCFG14</td>
<td>PCFG13</td>
<td>PCFG12</td>
<td>PCFG11</td>
<td>PCFG10</td>
<td>PCFG9</td>
<td>PCFG8</td>
<td>57</td>
</tr>
<tr>
<td>ANCON0(2)</td>
<td>PCFG7</td>
<td>PCFG6</td>
<td>—</td>
<td>PCFG4</td>
<td>PCFG3</td>
<td>PCFG2</td>
<td>PCFG1</td>
<td>PCFG0</td>
<td>57</td>
</tr>
<tr>
<td>PORTF</td>
<td>RF7</td>
<td>RF6</td>
<td>RF5</td>
<td>RF4</td>
<td>RF3</td>
<td>RF2</td>
<td>RF1</td>
<td>—</td>
<td>59</td>
</tr>
<tr>
<td>LATF</td>
<td>LATF7</td>
<td>LATF6</td>
<td>LATF5</td>
<td>LATF4</td>
<td>LATF3</td>
<td>LATF2</td>
<td>LATF1</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>TRISF</td>
<td>TRISF7</td>
<td>TRISF6</td>
<td>TRISF5</td>
<td>TRISF4</td>
<td>TRISF3</td>
<td>TRISF2</td>
<td>TRISF1</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>PORTH(1)</td>
<td>RH7</td>
<td>RH6</td>
<td>RH5</td>
<td>RH4</td>
<td>RH3</td>
<td>RH2</td>
<td>RH1</td>
<td>RH0</td>
<td>59</td>
</tr>
<tr>
<td>TRISH(1)</td>
<td>TRISH7</td>
<td>TRISH6</td>
<td>TRISH5</td>
<td>TRISH4</td>
<td>TRISH3</td>
<td>TRISH2</td>
<td>TRISH1</td>
<td>TRISH0</td>
<td>58</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for A/D conversion.

Note 1: These registers are not implemented on 64-pin devices.

Note 2: Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.
23.0 COMPARATOR VOLTAGE REFERENCE MODULE

The comparator voltage reference is a 16-tap resistor ladder network that provides a selectable reference voltage. Although its primary purpose is to provide a reference for the analog comparators, it may also be used independently of them.

A block diagram of the module is shown in Figure 23-1. The resistor ladder is segmented to provide two ranges of CVREF values and has a power-down function to conserve power when the reference is not being used. The module’s supply reference can be provided from either device VDD/VSS or an external voltage reference.
23.1 Configuring the Comparator Voltage Reference

The comparator voltage reference module is controlled through the CVRCON register (Register 23-1). The comparator voltage reference provides two ranges of output voltage, each with 16 distinct levels. The range to be used is selected by the CVRR bit (CVRCON<5>). The primary difference between the ranges is the size of the steps selected by the CVREF Selection bits (CVR3:CVR0), with one range offering finer resolution.

The equations used to calculate the output of the comparator voltage reference are as follows:

If CVRR = 1:
CVREF = ((CVR3:CVR0)/24) x (CVRSRC)

If CVRR = 0:
CVREF = (CVRSRC/4) + ((CVR3:CVR0)/32) x (CVRSRC)

The comparator reference supply voltage can come from either VDD and VSS, or the external VREF+ and VREF- that are multiplexed with RA2 and RA3. The voltage source is selected by the CVRSS bit (CVRCON<4>).

The settling time of the comparator voltage reference must be considered when changing the CVREF output (see Table 27-3 in Section 27.0 “Electrical Characteristics”).

The CVRCON register is a shared address SFR and uses the same address as the PR4 register. The CVRCON register is accessed by setting the ADSHR bit (WDTCON<4>).

REGISTER 23-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL REGISTER

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVREN</td>
<td>CVROE(1)</td>
<td>CVRR</td>
<td>CVRSS</td>
<td>CVR3</td>
<td>CVR2</td>
<td>CVR1</td>
<td>CVR0</td>
</tr>
</tbody>
</table>

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

bit 7 CVREN: Comparator Voltage Reference Enable bit
1 = CVREF circuit powered on
0 = CVREF circuit powered down

bit 6 CVROE: Comparator VREF Output Enable bit(1)
1 = CVREF voltage level is also output on the RF5/AN10/C1INB/CVREF pin
0 = CVREF voltage is disconnected from the RF5/AN10/C1INB/CVREF pin

bit 5 CVRR: Comparator VREF Range Selection bit
1 = 0 to 0.667 CVRSRC, with CVRSRC/24 step size (low range)
0 = 0.25 CVRSRC to 0.75 CVRSRC, with CVRSRC/32 step size (high range)

bit 4 CVRSS: Comparator VREF Source Selection bit
1 = Comparator reference source, CVRSRC = (VREF+) – (VREF-)
0 = Comparator reference source, CVRSRC = AVDD – AVSS

bit 3-0 CVR3:CVR0: Comparator VREF Value Selection bits (0 ≤ (CVR3:CVR0) ≤ 15)
When CVRR = 1:
CVREF = ((CVR3:CVR0)/24) x (CVRSRC)
When CVRR = 0:
CVREF = (CVRSRC/4) + ((CVR3:CVR0)/32) x (CVRSRC)

Note 1: CVROE overrides the TRISF<5> bit setting.
23.2 Voltage Reference Accuracy/Error

The full range of voltage reference cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 23-1) keep CVREF from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the CVREF output changes with fluctuations in that source. The tested absolute accuracy of the voltage reference can be found in Section 27.0 “Electrical Characteristics”.

23.3 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RF5 pin if the CVROE bit is set. Enabling the voltage reference output onto RA2 when it is configured as a digital input will increase current consumption. Connecting RF5 as a digital output with CVRSS enabled will also increase current consumption.

The RF5 pin can be used as a simple D/A output with limited drive capability. Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to VREF. Figure 23-2 shows an example buffering technique.

23.4 Operation During Sleep

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

23.5 Effects of a Reset

A device Reset disables the voltage reference by clearing CVREN (CVRCON<7>). This Reset also disconnects the reference from the RA2 pin by clearing CVROE, and selects the high-voltage range by clearing CVRR. The CVR value select bits are also cleared.

---

**FIGURE 23-2: COMPARATOR VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE**

**TABLE 23-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Reset Values on Page:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVRCON[2]</td>
<td>CVREN</td>
<td>CVROE</td>
<td>CVRR</td>
<td>CVRSS</td>
<td>CVR3</td>
<td>CVR2</td>
<td>CVR1</td>
<td>CVR0</td>
<td>59</td>
</tr>
<tr>
<td>CM1CON</td>
<td>CON</td>
<td>COE</td>
<td>CPOL</td>
<td>EVPOL1</td>
<td>EVPOL0</td>
<td>CREF</td>
<td>CCH1</td>
<td>CCH0</td>
<td>56</td>
</tr>
<tr>
<td>CM2CON</td>
<td>CON</td>
<td>COE</td>
<td>CPOL</td>
<td>EVPOL1</td>
<td>EVPOL0</td>
<td>CREF</td>
<td>CCH1</td>
<td>CCH0</td>
<td>56</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>58</td>
</tr>
<tr>
<td>TRISF</td>
<td>TRISF7</td>
<td>TRISF6</td>
<td>TRISF5</td>
<td>TRISF4</td>
<td>TRISF3</td>
<td>TRISF2</td>
<td>TRISF1</td>
<td>—</td>
<td>58</td>
</tr>
<tr>
<td>ANCON0[2]</td>
<td>PCFG7</td>
<td>PCFG6</td>
<td>—</td>
<td>PCFG4</td>
<td>PCFG3</td>
<td>PCFG2</td>
<td>PCFG1</td>
<td>PCFG0</td>
<td>57</td>
</tr>
<tr>
<td>ANCON1[2]</td>
<td>PCFG15</td>
<td>PCFG14</td>
<td>PCFG13</td>
<td>PCFG12</td>
<td>PCFG11</td>
<td>PCFG10</td>
<td>PCFG9</td>
<td>PCFG8</td>
<td>57</td>
</tr>
</tbody>
</table>

Legend: — = unimplemented, read as ’0’. Shaded cells are not used with the comparator voltage reference.

**Note 1:** These bits are only available in select oscillator modes (FOSC2 Configuration bit = 0); otherwise, they are unimplemented.

**Note 2:** Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.
24.0 SPECIAL FEATURES OF THE CPU

PIC18F87J11 Family devices include several features intended to maximize reliability and minimize cost through elimination of external components. These are:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Fail-Safe Clock Monitor
- Two-Speed Start-up
- Code Protection
- In-Circuit Serial Programming

The oscillator can be configured for the application depending on frequency, power, accuracy and cost. All of the options are discussed in detail in Section 2.0 “Oscillator Configurations”.

A complete discussion of device Resets and interrupts is available in previous sections of this data sheet. In addition to their Power-up and Oscillator Start-up Timers provided for Resets, the PIC18F87J11 Family of devices have a configurable Watchdog Timer which is controlled in software.

The inclusion of an internal RC oscillator also provides the additional benefits of a Fail-Safe Clock Monitor (FSCM) and Two-Speed Start-up. FSCM provides for background monitoring of the peripheral clock and automatic switchover in the event of its failure. Two-Speed Start-up enables code to be executed almost immediately on start-up, while the primary clock source completes its start-up delays.

All of these features are enabled and configured by setting the appropriate Configuration register bits.

24.1 Configuration Bits

The Configuration bits can be programmed (read as ‘0’) or left unprogrammed (read as ‘1’) to select various device configurations. These bits are mapped starting at program memory location 300000h. A complete list is shown in Table 24-2. A detailed explanation of the various bit functions is provided in Register 24-1 through Register 24-6.

24.1.1 CONSIDERATIONS FOR CONFIGURING THE PIC18F87J11 FAMILY DEVICES

Unlike previous PIC18 microcontrollers, devices of the PIC18F87J11 Family do not use persistent memory registers to store configuration information. The configuration bytes are implemented as volatile memory which means that configuration data must be programmed each time the device is powered up.

Configuration data is stored in the four words at the top of the on-chip program memory space, known as the Flash Configuration Words. It is stored in program memory in the same order shown in Table 24-2, with CONFIG1L at the lowest address and CONFIG3H at the highest. The data is automatically loaded in the proper Configuration registers during device power-up.

When creating applications for these devices, users should always specifically allocate the location of the Flash Configuration Word for configuration data. This is to make certain that program code is not stored in this address when the code is compiled.

The volatile memory cells used for the Configuration bits always reset to ‘1’ on Power-on Resets. For all other type of Reset events, the previously programmed values are maintained and used without reloading from program memory.

The four Most Significant bits of CONFIG1H, CONFIG2H and CONFIG3H in program memory should also be ‘1111’. This makes these Configuration Words appear to be NOP instructions in the remote event that their locations are ever executed by accident. Since Configuration bits are not implemented in the corresponding locations, writing ‘1’s to these locations has no effect on device operation.

To prevent inadvertent configuration changes during code execution, all programmable Configuration bits are write-once. After a bit is initially programmed during a power cycle, it cannot be written to again. Changing a device configuration requires that power to the device be cycled.
### TABLE 24-1: MAPPING OF THE FLASH CONFIGURATION WORDS TO THE CONFIGURATION REGISTERS

<table>
<thead>
<tr>
<th>Configuration Byte</th>
<th>Code Space Address</th>
<th>Configuration Register Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIG1L</td>
<td>XXXF8h</td>
<td>300000h</td>
</tr>
<tr>
<td>CONFIG1H</td>
<td>XXXF9h</td>
<td>300001h</td>
</tr>
<tr>
<td>CONFIG2L</td>
<td>XXXFAh</td>
<td>300002h</td>
</tr>
<tr>
<td>CONFIG2H</td>
<td>XXXFBh</td>
<td>300003h</td>
</tr>
<tr>
<td>CONFIG3L</td>
<td>XXXFCCh</td>
<td>300004h</td>
</tr>
<tr>
<td>CONFIG3H</td>
<td>XXXFDh</td>
<td>300005h</td>
</tr>
<tr>
<td>CONFIG4L(1)</td>
<td>XXXF Eh</td>
<td>300006h</td>
</tr>
<tr>
<td>CONFIG4H(1)</td>
<td>XXXFFh</td>
<td>300007h</td>
</tr>
</tbody>
</table>

**Note 1:** Unimplemented in PIC18F87J11 Family devices.

### TABLE 24-2: CONFIGURATION BITS AND DEVICE IDS

<table>
<thead>
<tr>
<th>File 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>Default/ Unprogrammed Value(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>300000h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONFIG1L</td>
<td>DEBUG</td>
<td>XINST</td>
<td>STVREN</td>
<td></td>
<td></td>
<td></td>
<td>WDTEN</td>
<td></td>
<td>111 - ---1</td>
</tr>
<tr>
<td>300001h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CP0</td>
<td></td>
<td>1111 -111</td>
</tr>
<tr>
<td>300002h</td>
<td>IESO</td>
<td>FCMEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FOSC2</td>
<td></td>
<td>11-- =111</td>
</tr>
<tr>
<td>300003h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WDTPS3</td>
<td>WDTPS2</td>
<td>1111 1111</td>
</tr>
<tr>
<td>300004h</td>
<td>WAIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1111 1---</td>
</tr>
<tr>
<td>300005h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MSSPMSK</td>
<td>PMPMX</td>
<td>1111 1111</td>
</tr>
<tr>
<td>3FFFEh</td>
<td>DEVID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>xxx0 0000</td>
</tr>
<tr>
<td>3FFFFh</td>
<td>DEVID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0100 00xx(4)</td>
</tr>
</tbody>
</table>

**Legend:** x = unknown, u = unchanged, - = unimplemented. Shaded cells are unimplemented, read as ‘0’.

**Note 1:** Values reflect the unprogrammed state as received from the factory and following Power-on Resets. In all other Reset states, the configuration bytes maintain their previously programmed states.

**2:** The value of these bits in program memory should always be ‘1’. This ensures that the location is executed as a NOP if it is accidentally executed.

**3:** Implemented in 80-pin devices only.

**4:** See Register 24-7 and Register 24-8 for DEVID values. These registers are read-only and cannot be programmed by the user.
REGISTER 24-1: CONFIG1L: CONFIGURATION REGISTER 1 LOW (BYTE ADDRESS 300000h)

<table>
<thead>
<tr>
<th>R/WO-1</th>
<th>R/WO-1</th>
<th>R/WO-1</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>R/WO-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEBUG</td>
<td>XINST</td>
<td>STVREN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WDTEN</td>
</tr>
</tbody>
</table>

bit 7    DEBUG: Background Debugger Enable bit
0 = Background debugger disabled; RB6 and RB7 configured as general purpose I/O pins
1 = Background debugger enabled; RB6 and RB7 are dedicated to In-Circuit Debug

bit 6    XINST: Extended Instruction Set Enable bit
0 = Instruction set extension and Indexed Addressing mode disabled (Legacy mode)
1 = Instruction set extension and Indexed Addressing mode enabled

bit 5    STVREN: Stack Overflow/Underflow Reset Enable bit
0 = Reset on stack overflow/underflow disabled
1 = Reset on stack overflow/underflow enabled

bit 4-1  Unimplemented: Read as ‘0’

bit 0    WDTEN: Watchdog Timer Enable bit
0 = WDT disabled (control is placed on SWDTEN bit)
1 = WDT enabled

REGISTER 24-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

<table>
<thead>
<tr>
<th>U-1</th>
<th>U-1</th>
<th>U-1</th>
<th>U-1</th>
<th>U-0</th>
<th>R/WO-1</th>
<th>U-1</th>
<th>U-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CP0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bit 7    Unimplemented: Maintain as ‘01’

bit 2    CP0: Code Protection bit
0 = Program memory is code-protected
1 = Program memory is not code-protected

bit 1-0  Unimplemented: Read as ‘0’

Legend:
R = Readable bit WO = Write-Once bit U = Unimplemented bit, read as ‘0’
-n = Value at POR ‘1’ = Bit is set ‘0’ = Bit is cleared x = Bit is unknown
### REGISTER 24-3: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Legend</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>IESO:</td>
<td>1</td>
<td>Two-Speed Start-up (Internal/External Oscillator Switchover) Control bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>Two-Speed Start-up disabled</td>
</tr>
<tr>
<td>6</td>
<td>FCMEN:</td>
<td>1</td>
<td>Fail-Safe Clock Monitor enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>Fail-Safe Clock Monitor disabled</td>
</tr>
<tr>
<td>5-3</td>
<td>Unimplemented</td>
<td>0</td>
<td>Read as ‘0’</td>
</tr>
<tr>
<td>2-0</td>
<td>FOSC2:FOSC0</td>
<td>111</td>
<td>EC oscillator with PLL enabled; CLKO on RA6 (ECPLL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>110</td>
<td>EC oscillator; CLKO on RA6 (EC)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101</td>
<td>HS oscillator with PLL enabled (HSPLL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>HS oscillator (HS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011</td>
<td>Internal oscillator with PLL enabled; CLKO on RA6, port function on RA7 (INTPLL1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010</td>
<td>Internal oscillator with PLL enabled; port function on RA6 and RA7 (INTPLL2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001</td>
<td>Internal oscillator block; CLKO on RA6, port function on RA7 (INTIO1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000</td>
<td>Internal oscillator block; port function on RA6 and RA7 (INTIO2)</td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **WO** = Write-Once bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown
### REGISTER 24-4: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)

<table>
<thead>
<tr>
<th>U-1</th>
<th>U-1</th>
<th>U-1</th>
<th>U-1</th>
<th>R/WO-1</th>
<th>R/WO-1</th>
<th>R/WO-1</th>
<th>R/WO-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WDTPS3</td>
<td>WDTPS2</td>
<td>WDTPS1</td>
<td>WDTPS0</td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- WO = Write-Once bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

**bit 7-4**  
Unimplemented: Maintain as ‘1’

**bit 3-0**  
WDTPS3:WDTPS0: Watchdog Timer Postscale Select bits

- 1111 = 1:32,768
- 1110 = 1:16,384
- 1101 = 1:8,192
- 1100 = 1:4,096
- 1011 = 1:2,048
- 1010 = 1:1,024
- 1001 = 1:512
- 1000 = 1:256
- 0111 = 1:128
- 0110 = 1:64
- 0101 = 1:32
- 0100 = 1:16
- 0011 = 1:8
- 0010 = 1:4
- 0001 = 1:2
- 0000 = 1:1
REGISTER 24-5: CONFIG3L: CONFIGURATION REGISTER 3 LOW (BYTE ADDRESS 300004h)

<table>
<thead>
<tr>
<th></th>
<th>WAIT&lt;7&gt;</th>
<th>BW&lt;6&gt;</th>
<th>EMB1&lt;5:4&gt;</th>
<th>EMB0&lt;3&gt;</th>
<th>EASHFT&lt;2:0&gt;</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>R/WO-1</td>
<td>R/WO-1</td>
<td>R/WO-1</td>
<td>R/WO-1</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- WO = Write-Once bit
- U = Unimplemented bit, read as ‘0’
- -n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 7 WAIT: External Bus Wait Enable bit<1>
1 = Wait states on the external bus are disabled
0 = Wait states on the external bus are enabled and selected by MEMCON<5:4>

bit 6 BW: Data Bus Width Select bit<1>
1 = 16-Bit Data Width modes
0 = 8-Bit Data Width modes

bit 5-4 EMB1:EMB0: External Memory Bus Configuration bits<1>
11 = Microcontroller mode, external bus disabled
10 = Extended Microcontroller mode, 12-bit address width for external bus
01 = Extended Microcontroller mode, 16-bit address width for external bus
00 = Extended Microcontroller mode, 20-bit address width for external bus

bit 3 EASHFT: External Address Bus Shift Enable bit<1>
1 = Address shifting enabled – external address bus is shifted to start at 000000h
0 = Address shifting disabled – external address bus reflects the PC value

bit 2-0 Unimplemented: Read as ‘0’

Note 1: Implemented on 80-pin devices only.
REGISTER 24-6:  CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

<table>
<thead>
<tr>
<th>bit 7-3</th>
<th>bit 2</th>
<th>bit 1</th>
<th>bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-1</td>
<td>U-1</td>
<td>U-1</td>
<td>U-1</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>MSSPMSK</td>
<td>PMPMX(^{(1)})</td>
</tr>
<tr>
<td>R/WO-1</td>
<td>R/WO-1</td>
<td>R/WO-1</td>
<td>R/WO-1</td>
</tr>
<tr>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

**Legend:**

- **R** = Readable bit
- **WO** = Write-Once bit
- **U** = Unimplemented bit, read as ‘0’
- \(-n\) = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

**bit 7-4**: Unimplemented: Maintain as ‘1’.

**bit 3**: MSSPMSK: MSSP Address Masking Mode Select bit
- 1 = 7-Bit Address Masking mode enabled
- 0 = 5-Bit Address Masking mode enable

**bit 2**: PMPMX: PMP Pin Multiplex bit\(^{(1)}\)
- 1 = PMP data and control multiplexed to same pins as external memory bus (PORTD and PORTE)
- 0 = PMP data and control multiplexed to alternate pin assignments (PORTA, PORTF and PORTH)

**bit 1**: ECCPMX: ECCP\(x\) MUX bit\(^{(1)}\)
- 1 = ECCP1 outputs (P1B/P1C) are multiplexed with RE6 and RE5;
  ECCP3 outputs (P3B/P3C) are multiplexed with RE4 and RE3
- 0 = ECCP1 outputs (P1B/P1C) are multiplexed with RH7 and RH6;
  ECCP3 outputs (P3B/P3C) are multiplexed with RH5 and RH4

**bit 0**: CCP2MX: ECCP2 MUX bit
- 1 = ECCP2/P2A is multiplexed with RC1
- 0 = ECCP2/P2A is multiplexed with RE7 in Microcontroller mode (all devices) or with RB3 in Extended Microcontroller mode (80-pin devices only)

**Note 1**: Implemented on 80-pin devices only.
REGISTER 24-7: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F87J11 FAMILY DEVICES

<table>
<thead>
<tr>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV2</td>
<td>DEV1</td>
<td>DEV0</td>
<td>REV4</td>
<td>REV3</td>
<td>REV2</td>
<td>REV1</td>
<td>REV0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| bit 7 | bit 0 |

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

**bit 7-5**DEV2:DEV0: Device ID bits
See Register 24-8 for a complete listing.

**bit 4-0**REV4:REV0: Revision ID bits
These bits are used to indicate the device revision.

REGISTER 24-8: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F87J11 FAMILY DEVICES

<table>
<thead>
<tr>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEV10</td>
<td>DEV9</td>
<td>DEV8</td>
<td>DEV7</td>
<td>DEV6</td>
<td>DEV5</td>
<td>DEV4</td>
<td>DEV3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| bit 7 | bit 0 |

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

**bit 7-0**DEV10:DEV3: Device ID bits:

<table>
<thead>
<tr>
<th>DEV10:DEV3 (DEVID2&lt;7:0&gt;)</th>
<th>DEV2:DEV0 (DEVID1&lt;7:5&gt;)</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100 0100</td>
<td>010</td>
<td>PIC18F66J11</td>
</tr>
<tr>
<td>0100 0100</td>
<td>011</td>
<td>PIC18F66J16</td>
</tr>
<tr>
<td>0100 0100</td>
<td>100</td>
<td>PIC18F67J11</td>
</tr>
<tr>
<td>0100 0100</td>
<td>111</td>
<td>PIC18F86J11</td>
</tr>
<tr>
<td>0100 0101</td>
<td>000</td>
<td>PIC18F86J16</td>
</tr>
<tr>
<td>0100 0101</td>
<td>001</td>
<td>PIC18F87J11</td>
</tr>
</tbody>
</table>
24.2 Watchdog Timer (WDT)

For PIC18F87J11 Family devices, the WDT is driven by the INTRC oscillator. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexor, controlled by the WDTPS bits in Configuration Register 2H. Available periods range from about 4 ms to 135 seconds (2.25 minutes depending on voltage, temperature and WDT postscaler). The WDT and postscaler are cleared whenever a SLEEP or CLRWDT instruction is executed, or a clock failure (primary or Timer1 oscillator) has occurred.

### FIGURE 24-1: WDT BLOCK DIAGRAM

- **SWDTEN**: Enable WDT
- **INTRC Oscillator**: INTRC Control
- **WDT Counter**: +128
- **Programmable Postscaler**: 1:1 to 1:32,768
- **Reset**: WDT Reset
- **Wake-up from Power-Managed Modes**: All Device Resets
- **WDTPS3:WDTPS0**: Sleep
- **CLRWDT**: INTRC Control

Note 1: The CLRWDT and SLEEP instructions clear the WDT and postscaler counts when executed.

2: When a CLRWDT instruction is executed, the postscaler count will be cleared.

24.2.1 CONTROL REGISTER

The WDTCN register (Register 24-9) is a readable and writable register. The SWDTEN bit enables or disables WDT operation. This allows software to override the WDTCN Configuration bit and enable the WDT only if it has been disabled by the Configuration bit.

The ADSHR bit selects which SFRs are currently selected and accessible. See Section 5.3.4.1 “Shared Address SFRs” for additional details.

The LVDSTAT is a read-only status bit which is continuously updated and provides information about the current level of VDDCORE. This bit is only valid when the on-chip voltage regulator is enabled.
### REGISTER 24-9: WDTCON: WATCHDOG TIMER CONTROL REGISTER

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>REGSLP</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>LVDSTAT</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Unimplemented</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>ADSHR</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Unimplemented</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Unimplemented</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Unimplemented</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>SWDTEN(*)</td>
<td>0</td>
</tr>
</tbody>
</table>

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

#### bit 7: REGSLP
Voltage Regulator Low-Power Operation Enable bit
- **1** = On-chip regulator enters low-power operation when device enters Sleep mode
- **0** = On-chip regulator is active, even in Sleep mode

#### bit 6: LVDSTAT
LVD Status bit
- **1** = VDDCORE > 2.45V
- **0** = VDDCORE < 2.45V

#### bit 0: SWDTEN
Software Controlled Watchdog Timer Enable bit
- **1** = Watchdog Timer is on
- **0** = Watchdog Timer is off

**Note 1:** This bit has no effect if the Configuration bit, WDTEN, is enabled.

### TABLE 24-3: SUMMARY OF WATCHDOG TIMER REGISTERS

<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>RCON</td>
<td>IPEN</td>
<td>CM</td>
<td>CMI</td>
<td>RI</td>
<td>TO</td>
<td>PD</td>
<td>POR</td>
<td>BOR</td>
<td>56</td>
</tr>
<tr>
<td>WDTCON</td>
<td>REGSLP</td>
<td>LVDSTAT</td>
<td></td>
<td>ADSHR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>57</td>
</tr>
</tbody>
</table>

Legend:
- **—** = unimplemented, read as '0'. Shaded cells are not used by the Watchdog Timer.
24.3 On-Chip Voltage Regulator

All of the PIC18F87J11 family devices power their core digital logic at a nominal 2.5V. For designs that are required to operate at a higher typical voltage, such as 3.3V, all devices in the PIC18F87J11 family incorporate an on-chip regulator that allows the device to run its core logic from VDD.

The regulator is controlled by the ENVREG pin. Tying VDD to the pin enables the regulator, which in turn, provides power to the core from the other VDD pins. When the regulator is enabled, a low-ESR filter capacitor must be connected to the VDDCORE/VCAP pin (Figure 24-2). This helps to maintain the stability of the regulator. The recommended value for the filter capacitor is provided in Section 27.3 “DC Characteristics: PIC18F87J11 Family (Industrial)“.

If ENVREG is tied to VSS, the regulator is disabled. In this case, separate power for the core logic at a nominal 2.5V must be supplied to the device on the VDDCORE/VCAP pin to run the I/O pins at higher voltage levels, typically 3.3V. Alternatively, the VDDCORE/VCAP and VDD pins can be tied together to operate at a lower nominal voltage. Refer to Figure 24-2 for possible configurations.

24.3.1 VOLTAGE REGULATOR TRACKING MODE AND LOW-VOLTAGE DETECTION

When it is enabled, the on-chip regulator provides a constant voltage of 2.5V nominal to the digital core logic. The regulator can provide this level from a VDD of about 2.5V, all the way up to the device’s VDDMAX. It does not have the capability to boost VDD levels below 2.5V. In order to prevent “brown-out” conditions, when the voltage drops too low for the regulator, the regulator enters Tracking mode. In Tracking mode, the regulator output follows VDD, with a typical voltage drop of 100 mV.

The on-chip regulator includes a simple, Low-Voltage Detect (LVD) circuit. If VDD drops too low to maintain approximately 2.45V on VDDCORE, the circuit sets the Low-Voltage Detect Interrupt Flag, LVDIF (PIR2<2>). This can be used to generate an interrupt and put the application into a low-power operational mode, or trigger an orderly shutdown. Low-Voltage Detection is only available when the regulator is enabled.

The Low-Voltage Detect interrupt is edge-sensitive. The interrupt flag will only be set once per falling edge of VDDCORE. Firmware can clear the interrupt flag, but a new interrupt will not be generated until VDDCORE rises back above, and then falls below, the 2.45 threshold. Upon device Resets, the interrupt flag will reset to ‘0’, even if VDDCORE is less than 2.45V. When the regulator is enabled, the LVDSTAT bit in the WDTCON register can be polled to determine the current level of VDDCORE.

Note 1: These are typical operating voltages. Refer to Section 27.1 “DC Characteristics: Supply Voltage” for the full operating ranges of VDD and VDDCORE.
24.3.2 ON-CHIP REGULATOR AND BOR
When the on-chip regulator is enabled, PIC18F87J11 family devices also have a simple brown-out capability. If the voltage supplied to the regulator is inadequate to maintain a regulated level, the regulator Reset circuitry will generate a Brown-out Reset. This event is captured by the BOR flag bit (RCON<0>).

The operation of the Brown-out Reset is described in more detail in Section 4.4 "Brown-out Reset (BOR)" and Section 4.4.1 "Detecting BOR". The brown-out voltage levels are specific in Section 27.1 "DC Characteristics: Supply Voltage PIC18F87J11 Family (Industrial)".

24.3.3 POWER-UP REQUIREMENTS
The on-chip regulator is designed to meet the power-up requirements for the device. If the application does not use the regulator, then strict power-up conditions must be adhered to. While powering up, VDDCORE must never exceed VDD by 0.3 volts.

24.3.4 OPERATION IN SLEEP MODE
When enabled, the on-chip regulator always consumes a small incremental amount of current over IDD. This includes when the device is in Sleep mode, even though the core digital logic does not require power. To provide additional savings in applications where power resources are critical, the regulator can be configured to automatically disable itself whenever the device goes into Sleep mode. This feature is controlled by the REGSLP bit (WDTCON<7>, Register 24-9). Setting this bit disables the regulator in Sleep mode and reduces its current consumption to a minimum.

Substantial Sleep mode power savings can be obtained by setting the REGSLP bit, but device wake-up time will increase in order to insure the regulator has enough time to stabilize. The REGSLP bit is automatically cleared by hardware when a Low-Voltage Detect condition occurs.

24.4 Two-Speed Start-up
The Two-Speed Start-up feature helps to minimize the latency period, from oscillator start-up to code execution, by allowing the microcontroller to use the INTRC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO Configuration bit.

Two-Speed Start-up should be enabled only if the primary oscillator mode is HS or HSPLL (Crystal-Based) modes. Since the EC and ECPLL modes do not require an Oscillator Start-up Timer delay, Two-Speed Start-up should be disabled.

When enabled, Resets and wake-ups from Sleep mode cause the device to configure itself to run from the internal oscillator block as the clock source, following the time-out of the Power-up Timer after a Power-on Reset is enabled. This allows almost immediate code execution while the primary oscillator starts and the OST is running. Once the OST times out, the device automatically switches to PRI_RUN mode.

In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

![FIGURE 24-3: TIMING TRANSITION FOR TWO-SPEED START-UP (INTRC TO HSPLL)](image_url)
24.4.1 Special Considerations for Using Two-Speed Start-up

While using the INTRC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power-managed modes, including serial SLEEP instructions (refer to Section 3.1.4 “Multiple Sleep Commands”). In practice, this means that user code can change the SCS1:SCS0 bit settings or issue SLEEP instructions before the OST times out. This would allow an application to briefly wake-up, perform routine “housekeeping” tasks and return to Sleep before the device starts to operate from the primary oscillator.

User code can also check if the primary clock source is currently providing the device clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

24.5 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the microcontroller to continue operation in the event of an external oscillator failure by automatically switching the device clock to the internal oscillator block. The FSCM function is enabled by setting the FCMEN Configuration bit.

When FSCM is enabled, the INTRC oscillator runs at all times to monitor clocks to peripherals and provide a backup clock in the event of a clock failure. Clock monitoring (shown in Figure 24-4) is accomplished by creating a sample clock signal which is the INTRC output divided by 64. This allows ample time between FSCM sample clocks for a peripheral clock edge to occur. The peripheral device clock and the sample clock are presented as inputs to the Clock Monitor (CM) latch. The CM is set on the falling edge of the device clock source but cleared on the rising edge of the sample clock.

Clock failure is tested for on the falling edge of the sample clock. If a sample clock falling edge occurs while CM is still set, a clock failure has been detected (Figure 24-5). This causes the following:

- the FSCM generates an oscillator fail interrupt by setting bit OSCFIF (PIR2<7>);
- the device clock source is switched to the internal oscillator block (OSCCON is not updated to show the current clock source – this is the fail-safe condition); and
- the WDT is reset.

During switchover, the postscaler frequency from the internal oscillator block may not be sufficiently stable for timing sensitive applications. In these cases, it may be desirable to select another clock configuration and enter an alternate power-managed mode. This can be done to attempt a partial recovery or execute a controlled shutdown. See Section 3.1.4 “Multiple Sleep Commands” and Section 24.4.1 “Special Considerations for Using Two-Speed Start-up” for more details.

The FSCM will detect failures of the primary or secondary clock sources only. If the internal oscillator block fails, no failure would be detected, nor would any action be possible.

24.5.1 FSCM and the Watchdog Timer

Both the FSCM and the WDT are clocked by the INTRC oscillator. Since the WDT operates with a separate divider and counter, disabling the WDT has no effect on the operation of the INTRC oscillator when the FSCM is enabled.

As already noted, the clock source is switched to the INTRC clock when a clock failure is detected; this may mean a substantial change in the speed of code execution. If the WDT is enabled with a small prescale value, a decrease in clock speed allows a WDT time-out to occur and a subsequent device Reset. For this reason, fail-safe clock events also reset the WDT and postscaler, allowing it to start timing from when execution speed was changed and decreasing the likelihood of an erroneous time-out.
24.5.2 EXITING FAIL-SAFE OPERATION

The fail-safe condition is terminated by either a device Reset or by entering a power-managed mode. On Reset, the controller starts the primary clock source specified in Configuration Register 2H (with any required start-up delays that are required for the oscillator mode, such as OST or PLL timer). The INTRC oscillator provides the device clock until the primary clock source becomes ready (similar to a Two-Speed Start-up). The clock source is then switched to the primary clock (indicated by the OSTS bit in the OSCCON register becoming set). The Fail-Safe Clock Monitor then resumes monitoring the peripheral clock.

The primary clock source may never become ready during start-up. In this case, operation is clocked by the INTRC oscillator. The OSCCON register will remain in its Reset state until a power-managed mode is entered.

24.5.3 FSCM INTERRUPTS IN POWER-MANAGED MODES

By entering a power-managed mode, the clock multiplexer selects the clock source selected by the OSCCON register. Fail-Safe Clock Monitoring of the power-managed clock source resumes in the power-managed mode.

If an oscillator failure occurs during power-managed operation, the subsequent events depend on whether or not the oscillator failure interrupt is enabled. If enabled (OSCFIF = 1), code execution will be clocked by the INTRC multiplexer. An automatic transition back to the failed clock source will not occur.

If the interrupt is disabled, subsequent interrupts while in Idle mode will cause the CPU to begin executing instructions while being clocked by the INTRC source.

24.5.4 POR OR WAKE-UP FROM SLEEP

The FSCM is designed to detect oscillator failure at any point after the device has exited Power-on Reset (POR) or low-power Sleep mode. When the primary device clock is either the EC or INTRC modes, monitoring can begin immediately following these events.

For HS or HSPLL modes, the situation is somewhat different. Since the oscillator may require a start-up time considerably longer than the FSCM sample clock time, a false clock failure may be detected. To prevent this, the internal oscillator block is automatically configured as the device clock and functions until the primary clock is stable (the OST and PLL timers have timed out). This is identical to Two-Speed Start-up mode. Once the primary clock is stable, the INTRC returns to its role as the FSCM source.

Note: The same logic that prevents false oscillator failure interrupts on POR, or wake from Sleep, will also prevent the detection of the oscillator’s failure to start at all following these events. This can be avoided by monitoring the OSTS bit and using a timing routine to determine if the oscillator is taking too long to start. Even so, no oscillator failure interrupt will be flagged.

As noted in Section 24.4.1 “Special Considerations for Using Two-Speed Start-up”, it is also possible to select another clock configuration and enter an alternate power-managed mode while waiting for the primary clock to become stable. When the new power-managed mode is selected, the primary clock is disabled.
24.6 Program Verification and Code Protection

For all devices in the PIC18F87J11 Family of devices, the on-chip program memory space is treated as a single block. Code protection for this block is controlled by one Configuration bit, CP0. This bit inhibits external reads and writes to the program memory space. It has no direct effect in normal execution mode.

24.6.1 CONFIGURATION REGISTER PROTECTION

The Configuration registers are protected against untoward changes or reads in two ways. The primary protection is the write-once feature of the Configuration bits which prevents reconfiguration once the bit has been programmed during a power cycle. To safeguard against unpredictable events, Configuration bit changes resulting from individual cell level disruptions (such as ESD events) will cause a parity error and trigger a device Reset. This is seen by the user as a Configuration Match Reset.

The data for the Configuration registers is derived from the Flash Configuration Words in program memory. When the CP0 bit set, the source data for device configuration is also protected as a consequence.

24.7 In-Circuit Serial Programming

PIC18F87J11 Family microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

24.8 In-Circuit Debugger

When the DEBUG Configuration bit is programmed to a ‘0’, the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some resources are not available for general use. Table 24-4 shows which resources are required by the background debugger.

<table>
<thead>
<tr>
<th>TABLE 24-4: DEBUGGER RESOURCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O pins:</td>
</tr>
<tr>
<td>Stack:</td>
</tr>
<tr>
<td>Program Memory:</td>
</tr>
<tr>
<td>Data Memory:</td>
</tr>
</tbody>
</table>
25.0 INSTRUCTION SET SUMMARY

The PIC18F87J11 Family of devices incorporate the standard set of 75 PIC18 core instructions, as well as an extended set of 8 new instructions for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

25.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® instruction sets, while maintaining an easy migration from these instruction sets. Most instructions are a single program memory word (16 bits), but there are four instructions that require two program memory locations. Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction. The instruction set is highly orthogonal and is grouped into four basic categories:

• **Byte-oriented** operations
• **Bit-oriented** operations
• **Literal** operations
• **Control** operations

The PIC18 instruction set summary in Table 25-2 lists byte-oriented, bit-oriented, literal and control operations. Table 25-1 shows the opcode field descriptions. Most byte-oriented instructions have three operands:

1. The file register (specified by ‘f’)
2. The destination of the result (specified by ‘d’)
3. The accessed memory (specified by ‘a’)

The file register designator, ‘f’, specifies which file register is to be used by the instruction. The destination designator, ‘d’, specifies where the result of the operation is to be placed. If ‘d’ is ‘0’, the result is placed in the WREG register. If ‘d’ is ‘1’, the result is placed in the file register specified in the instruction.

All bit-oriented instructions have three operands:

1. The file register (specified by ‘f’)
2. The bit in the file register (specified by ‘b’)
3. The accessed memory (specified by ‘a’)

The bit field designator ‘b’ selects the number of the bit affected by the operation, while the file register designator, ‘f’, represents the number of the file in which the bit is located.

The literal instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by ‘k’)
- The desired FSR register to load the literal value into (specified by ‘f’)
- No operand required (specified by ‘—’)

The control instructions may use some of the following operands:

- A program memory address (specified by ‘n’)
- The mode of the CALL or RETURN instructions (specified by ‘s’)
- The mode of the table read and table write instructions (specified by ‘m’)
- No operand required (specified by ‘——’)

All instructions are a single word, except for four double-word instructions. These instructions were made double-word to contain the required information in 32 bits. In the second word, the 4 MSbs are ‘1’s. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μs. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 μs. Two-word branch instructions (if true) would take 3 μs.

Figure 25-1 shows the general formats that the instructions can have. All examples use the convention ‘nnh’ to represent a hexadecimal number.

The instruction set summary, shown in Table 25-2, lists the standard instructions recognized by the Microchip MPASMTM Assembler.

Section 25.1.1 “Standard Instruction Set” provides a description of each instruction.
TABLE 25-1: OPCODE FIELD DESCRIPTIONS

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| a     | RAM access bit:  
|       | a = 0: RAM location in Access RAM (BSR register is ignored)  
|       | a = 1: RAM bank is specified by BSR register  |
| bbb   | Bit address within an 8-bit file register (0 to 7).  |
| BSR   | Bank Select Register. Used to select the current RAM bank.  |
| C, DC, Z, OV, N | ALU Status bits: Carry, Digit Carry, Zero, Overflow, Negative.  |
| d     | Destination select bit:  
|       | d = 0: store result in WREG  
|       | d = 1: store result in file register f  |
| dest  | Destination: either the WREG register or the specified register file location.  |
| f     | 8-bit register file address (00h to FFh), or 2-bit FSR designator (0h to 3h).  |
| f_s   | 12-bit register file address (000h to FFFh). This is the source address.  |
| f_d   | 12-bit register file address (000h to FFFh). This is the destination address.  |
| GIE   | Global Interrupt Enable bit.  |
| k     | Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).  |
| label | Label name.  |
| mm    | The mode of the TBLPTR register for the table read and table write instructions.  
|       | Only used with table read and table write instructions:  
|       | * No Change to register (such as TBLPTR with table reads and writes)  
|       | *+ Post-Increment register (such as TBLPTR with table reads and writes)  
|       | *- Post-Decrement register (such as TBLPTR with table reads and writes)  
|       | + Pre-Increment register (such as TBLPTR with table reads and writes)  |
| n     | The relative address (2's complement number) for relative branch instructions or the direct address for Call/Branch and Return instructions.  |
| PC    | Program Counter.  |
| PCL   | Program Counter Low Byte.  |
| PCH   | Program Counter High Byte.  |
| PCLATH| Program Counter High Byte Latch.  |
| PCLATU| Program Counter Upper Byte Latch.  |
| PD    | Power-Down bit.  |
| PRODH | Product of Multiply High Byte.  |
| PRODL | Product of Multiply Low Byte.  |
| s     | Fast Call/Return mode select bit:  
|       | s = 0: do not update into/from shadow registers  
|       | s = 1: certain registers loaded into/from shadow registers (Fast mode)  |
| TBLPTR| 21-bit Table Pointer (points to a program memory location).  |
| TABLAT| 8-bit Table Latch.  |
| TO    | Time-out bit.  |
| TOS   | Top-of-Stack.  |
| u     | Unused or Unchanged.  |
| WDT   | Watchdog Timer.  |
| WREG  | Working register (accumulator).  |
| x     | Don’t care (‘0’ or ‘1’). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.  |
| z_s   | 7-bit offset value for Indirect Addressing of register files (source).  |
| z_d   | 7-bit offset value for Indirect Addressing of register files (destination).  |
| { }   | Optional argument.  |
| [text] | Indicates Indexed Addressing.  |
| (text) | The contents of text.  |
| [expr]<n> | Specifies bit n of the register indicated by the pointer, expr.  |
| →     | Assigned to.  |
| < >   | Register bit field.  |
| ∈     | In the set of.  |
| italics | User-defined term (font is Courier New).  |
### General Format for Instructions

#### Byte-oriented File Register Operations

<table>
<thead>
<tr>
<th>15</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>d</td>
<td>a</td>
<td>f (FILE #)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **d = 0** for result destination to be WREG register
- **d = 1** for result destination to be file register (f)
- **a = 0** to force Access Bank
- **a = 1** for BSR to select bank
- **f = 8-bit file register address**

#### Byte to Byte Move Operations (2-word)

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>f (Source FILE #)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>f (Destination FILE #)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **f = 12-bit file register address**

#### Bit-oriented File Register Operations

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>b (BIT #)</td>
<td>a</td>
<td>f (FILE #)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **b = 3-bit position of bit in file register (f)**
- **a = 0** to force Access Bank
- **a = 1** for BSR to select bank
- **f = 8-bit file register address**

#### Literal Operations

<table>
<thead>
<tr>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>k (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **k = 8-bit immediate value**

#### Control Operations

**CALL, GOTO and Branch Operations**

<table>
<thead>
<tr>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>n&lt;7:0&gt; (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>n&lt;19:8&gt; (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **n = 20-bit immediate value**

<table>
<thead>
<tr>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>S n&lt;7:0&gt; (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>12</th>
<th>11</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>n&lt;19:8&gt; (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **S = Fast bit**

<table>
<thead>
<tr>
<th>15</th>
<th>11</th>
<th>10</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>n&lt;10:0&gt; (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>n&lt;7:0&gt; (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **BRA MYFUNC**

<table>
<thead>
<tr>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPCODE</td>
<td>n&lt;7:0&gt; (literal)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **BC MYFUNC**

### Example Instructions

- **ADDWF MYREG, W, B**
- **MOVFF MYREG1, MYREG2**
- **BSF MYREG, bit, B**
- **MOVLW 7Fh**
- **GOTO Label**
- **CALL MYFUNC**
- **BRA MYFUNC**
- **BC MYFUNC**
### TABLE 25-2: PIC18F87J11 FAMILY INSTRUCTION SET

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</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>ADDWF f, d, a</td>
<td>Add WREG and f</td>
<td>1</td>
<td>0010 01da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td>1, 2</td>
</tr>
<tr>
<td>ADDWFC f, d, a</td>
<td>Add WREG and Carry bit to f</td>
<td>1</td>
<td>0000 00da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td>1, 2</td>
</tr>
<tr>
<td>ANDWF f, d, a</td>
<td>AND WREG with f</td>
<td>1</td>
<td>0001 01da ffff ffff</td>
<td>Z, N</td>
<td>1, 2</td>
</tr>
<tr>
<td>CLR f, a</td>
<td>Clear f</td>
<td>1</td>
<td>0010 101a ffff ffff</td>
<td>Z</td>
<td>2</td>
</tr>
<tr>
<td>COMF f, d, a</td>
<td>Complement f</td>
<td>1</td>
<td>0001 11da ffff ffff</td>
<td>Z, N</td>
<td>1, 2</td>
</tr>
<tr>
<td>CPFSEQ f, a</td>
<td>Compare f with WREG, Skip = 1</td>
<td>1</td>
<td>0110 001a ffff ffff</td>
<td>None</td>
<td>4</td>
</tr>
<tr>
<td>CPFSGT f, a</td>
<td>Compare f with WREG, Skip &gt; 1</td>
<td>1</td>
<td>0110 010a ffff ffff</td>
<td>None</td>
<td>4</td>
</tr>
<tr>
<td>CPFSLT f, a</td>
<td>Compare f with WREG, Skip &lt; 1</td>
<td>1</td>
<td>0110 000a ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>DEC f, d, a</td>
<td>Decrement f</td>
<td>1</td>
<td>0000 01da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td>1, 2</td>
</tr>
<tr>
<td>DECSZ f, d, a</td>
<td>Decrement f, Skip if 0</td>
<td>1 (2 or 3)</td>
<td>0110 001a ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>DCFSNZ f, d, a</td>
<td>Decrement f, Skip if Not 0</td>
<td>1 (2 or 3)</td>
<td>0100 11da ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>INC f, d, a</td>
<td>Increment f</td>
<td>1</td>
<td>0010 10da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td>1, 2</td>
</tr>
<tr>
<td>INCFSZ f, d, a</td>
<td>Increment f, Skip if 0</td>
<td>1 (2 or 3)</td>
<td>0011 11da ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>INFNSZ f, d, a</td>
<td>Increment f, Skip if Not 0</td>
<td>1 (2 or 3)</td>
<td>0100 10da ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>IORWF f, d, a</td>
<td>Inclusive OR WREG with f</td>
<td>1</td>
<td>0001 00da ffff ffff</td>
<td>Z, N</td>
<td>1, 2</td>
</tr>
<tr>
<td>MOV f, d, a</td>
<td>Move f</td>
<td>1</td>
<td>0101 00da ffff ffff</td>
<td>Z, N</td>
<td>1</td>
</tr>
<tr>
<td>MOVFF fs, fd</td>
<td>Move f fs (source) to 1st word</td>
<td>2</td>
<td>1100 ffff ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ffd (destination) 2nd word</td>
<td></td>
<td>1111 ffff ffff ffff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOVWF f, a</td>
<td>Move WREG to f</td>
<td>1</td>
<td>0110 111a ffff ffff</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MULWF f, a</td>
<td>Multiply WREG with f</td>
<td>1</td>
<td>0000 001a ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>NEGF f, a</td>
<td>Negate f</td>
<td>1</td>
<td>0110 110a ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>RLC f, d, a</td>
<td>Rotate Left f through Carry</td>
<td>1</td>
<td>0011 01da ffff ffff</td>
<td>C, Z, N</td>
<td>1, 2</td>
</tr>
<tr>
<td>RLNCF f, d, a</td>
<td>Rotate Left f (No Carry)</td>
<td>1</td>
<td>0100 01da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>RRC f, d, a</td>
<td>Rotate Right f through Carry</td>
<td>1</td>
<td>0011 00da ffff ffff</td>
<td>C, Z, N</td>
<td></td>
</tr>
<tr>
<td>RRNCF f, d, a</td>
<td>Rotate Right f (No Carry)</td>
<td>1</td>
<td>0100 00da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>SETF f, a</td>
<td>Set f</td>
<td>1</td>
<td>0110 100a ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>SUBF f, d, a</td>
<td>Subtract f from WREG with Borrow</td>
<td>1</td>
<td>0101 01da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>SUBWF f, d, a</td>
<td>Subtract WREG from f</td>
<td>1</td>
<td>0101 11da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td>1, 2</td>
</tr>
<tr>
<td>SUBWFB F, d, a</td>
<td>Subtract WREG from f with Borrow</td>
<td>1</td>
<td>0101 10da ffff ffff</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>SWAPF f, d, a</td>
<td>Swap Nibbles in f</td>
<td>1</td>
<td>0011 10da ffff ffff</td>
<td>None</td>
<td>4</td>
</tr>
<tr>
<td>TSTFSZ f, a</td>
<td>Test f, Skip if 0</td>
<td>1 (2 or 3)</td>
<td>0110 011a ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>XORWF f, d, a</td>
<td>Exclusive OR WREG with f</td>
<td>1</td>
<td>0001 10da ffff ffff</td>
<td>Z, N</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is ‘1’ for a pin configured as input and is driven low by an external device, the data will be written back with a ‘0’.

**2:** If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

**3:** 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.

**4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.
### BIT-ORIENTED OPERATIONS

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCF f, b, a</td>
<td>Bit Clear f</td>
<td>1</td>
<td>1001 bbb a ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>BSF f, b, a</td>
<td>Bit Set f</td>
<td>1</td>
<td>1000 bbb a ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
<tr>
<td>BTFSC f, b, a</td>
<td>Bit Test f, Skip if Clear</td>
<td>1 (2 or 3)</td>
<td>1011 bbb a ffff ffff</td>
<td>None</td>
<td>3, 4</td>
</tr>
<tr>
<td>BTFSS f, b, a</td>
<td>Bit Test f, Skip if Set</td>
<td>1 (2 or 3)</td>
<td>1010 bbb a ffff ffff</td>
<td>None</td>
<td>3, 4</td>
</tr>
<tr>
<td>BTG f, b, a</td>
<td>Bit Toggle f</td>
<td>1</td>
<td>0111 bbb a ffff ffff</td>
<td>None</td>
<td>1, 2</td>
</tr>
</tbody>
</table>

### CONTROL OPERATIONS

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</th>
<th>Status Affected</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BC n</td>
<td>Branch if Carry</td>
<td>1 (2)</td>
<td>1110 0010 nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BN n</td>
<td>Branch if Negative</td>
<td>1 (2)</td>
<td>1110 0110 nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BNC n</td>
<td>Branch if Not Carry</td>
<td>1 (2)</td>
<td>1110 0011 nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BNN n</td>
<td>Branch if Not Negative</td>
<td>1 (2)</td>
<td>1110 0111 nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BNOV n</td>
<td>Branch if Not Overflow</td>
<td>1 (2)</td>
<td>1110 0101 nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BNZ n</td>
<td>Branch if Not Zero</td>
<td>1 (2)</td>
<td>1110 0001 nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BOV n</td>
<td>Branch if Overflow</td>
<td>1 (2)</td>
<td>1110 0100 nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BRA n</td>
<td>Branch Unconditionally</td>
<td>2</td>
<td>1101 0nnn nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>BZ n</td>
<td>Branch if Zero</td>
<td>1 (2)</td>
<td>1110 0000 nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>CALL n, s</td>
<td>Call Subroutine 1st word</td>
<td>1</td>
<td>1110 0100 s kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2nd word</td>
<td></td>
<td>1111 kkkk kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>CLRWD T</td>
<td>Clear Watchdog Timer</td>
<td>1</td>
<td>0000 0000 0000 0100</td>
<td>TO, PD</td>
<td></td>
</tr>
<tr>
<td>DAW —</td>
<td>Decimal Adjust WREG</td>
<td>1</td>
<td>0000 0000 0000 0111</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>GOTO n</td>
<td>Go to Address 1st word</td>
<td>2</td>
<td>1110 0111 kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2nd word</td>
<td></td>
<td>1111 kkkk kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>NOP —</td>
<td>No Operation</td>
<td>1</td>
<td>0000 0000 0000 0000</td>
<td>None</td>
<td>4</td>
</tr>
<tr>
<td>POP —</td>
<td>Pop Top of Return Stack (TOS)</td>
<td>1</td>
<td>1111 xxxx xxxx xxxx</td>
<td>None</td>
<td>0101</td>
</tr>
<tr>
<td>PUSH —</td>
<td>Push Top of Return Stack (TOS)</td>
<td>1</td>
<td>0000 0000 0000 0101</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>RCALL n</td>
<td>Relative Call</td>
<td>2</td>
<td>1101 1nnn nnnn nnnn</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>Software Device Reset</td>
<td>1</td>
<td>0000 0000 1111 1111</td>
<td>All</td>
<td></td>
</tr>
<tr>
<td>RETFIE s</td>
<td>Return from Interrupt Enable</td>
<td>2</td>
<td>0000 0000 0001 0001</td>
<td>GIE/GIEH, PEIE/GIEL</td>
<td></td>
</tr>
<tr>
<td>RETLW k</td>
<td>Return with Literal in WREG</td>
<td>2</td>
<td>0000 1100 kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>RETURN s</td>
<td>Return from Subroutine</td>
<td>2</td>
<td>0000 0000 0001 001s</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>SLEEP</td>
<td>Go into Standby mode</td>
<td>1</td>
<td>0000 0000 0000 0011</td>
<td>TO, PD</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

**Note 2:** If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

**Note 3:** 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 4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a **NOP** unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.
### TABLE 25-2: PIC18F87J11 FAMILY INSTRUCTION SET (CONTINUED)

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</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>LITERAL OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDLW k</td>
<td>Add Literal and WREG</td>
<td>1</td>
<td>0000 1111 kkkk kkkk</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>ANDLW k</td>
<td>AND Literal with WREG</td>
<td>1</td>
<td>0000 1011 kkkk kkkk</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>IORLW k</td>
<td>Inclusive OR Literal with WREG</td>
<td>1</td>
<td>0000 1001 kkkk kkkk</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td>LFSR f, k</td>
<td>Move Literal (12-bit) 2nd word to FSR (f) 1st word</td>
<td>2</td>
<td>1110 1110 00ff kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MOVLB k</td>
<td>Move Literal to BSR&lt;3:0&gt;</td>
<td>1</td>
<td>0000 0001 0000 kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MOVLW k</td>
<td>Move Literal to WREG</td>
<td>1</td>
<td>0000 1110 kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>MULLW k</td>
<td>Multiply Literal with WREG</td>
<td>1</td>
<td>0000 1101 kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>RETLW k</td>
<td>Return with Literal in WREG</td>
<td>2</td>
<td>0000 1100 kkkk kkkk</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>SUBLW k</td>
<td>Subtract WREG from Literal</td>
<td>1</td>
<td>0000 1000 kkkk kkkk</td>
<td>C, DC, Z, OV, N</td>
<td></td>
</tr>
<tr>
<td>XORLW k</td>
<td>Exclusive OR Literal with WREG</td>
<td>1</td>
<td>0000 1010 kkkk kkkk</td>
<td>Z, N</td>
<td></td>
</tr>
<tr>
<td><strong>DATA MEMORY &lt;-&gt; PROGRAM MEMORY OPERATIONS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLRD*</td>
<td>Table Read</td>
<td>2</td>
<td>0000 0000 0000 1000</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TBLRD*+</td>
<td>Table Read with Post-Increment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLRD*-</td>
<td>Table Read with Post-Decrement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLRD++</td>
<td>Table Read with Pre-Increment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLWT*</td>
<td>Table Write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLWT+*</td>
<td>Table Write with Post-Increment</td>
<td>2</td>
<td>0000 0000 0000 1011</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>TBLWT*-</td>
<td>Table Write with Post-Decrement</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TBLWT++</td>
<td>Table Write with Pre-Increment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

**Note 2:** If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

**Note 3:** 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 4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.
### 25.1.1 STANDARD INSTRUCTION SET

<table>
<thead>
<tr>
<th>ADDLW</th>
<th>ADD Literal to W</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>ADDLW k</td>
</tr>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 255</td>
</tr>
<tr>
<td>Operation:</td>
<td>(W) + k → W</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>N, OV, C, DC, Z</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0000 1111 kkkk kkkk</td>
</tr>
<tr>
<td>Description:</td>
<td>The contents of W are added to the 8-bit literal ‘k’ and the result is placed in W.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td>Q1 Q2 Q3 Q4</td>
</tr>
<tr>
<td>Decode</td>
<td>Read literal ‘k’</td>
</tr>
</tbody>
</table>

**Example:** ADDLW 15h

- Before Instruction
  - W = 10h
- After Instruction
  - W = 25h

<table>
<thead>
<tr>
<th>ADDWF</th>
<th>ADD W to f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>ADDWF f (d (a))</td>
</tr>
<tr>
<td>Operands:</td>
<td>0 ≤ f ≤ 255</td>
</tr>
<tr>
<td>d ∈ [0,1]</td>
<td></td>
</tr>
<tr>
<td>a ∈ [0,1]</td>
<td></td>
</tr>
<tr>
<td>Operation:</td>
<td>(W) + (f) → dest</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>N, OV, C, DC, Z</td>
</tr>
<tr>
<td>Encoding:</td>
<td>0010 01da ffff ffff</td>
</tr>
</tbody>
</table>
| Description: | Add W to register ‘f’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default).
| Q Cycle Activity: | Q1 Q2 Q3 Q4   |
| Decode | Read register ‘f’ | Process Data | Write to destination |

**Example:** ADDWF REG, 0, 0

- Before Instruction
  - W = 17h
  - REG = 0C2h
- After Instruction
  - W = 0D9h
  - REG = 0C2h

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).
ADDWFc  ADD W and Carry bit to f
Syntax: ADDWFc f {d {a}}
Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]
Operation: (W) + (f) + (C) → dest
Status Affected: N, OV, C, DC, Z
Encoding: 0010 00da ffff ffff
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’.
If ‘a’ is ‘0’, the Access Bank is selected.
If ‘a’ is ‘1’, the BSR is used to select the GPR bank.
If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.
Words: 1
Cycles: 1
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register ‘f’</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

Example: ADDWFc REG, 0, 1
Before Instruction
Carry bit = 1
REG = 02h
W = 4Dh
After Instruction
Carry bit = 0
REG = 02h
W = 50h

ANDLW AND Literal with W
Syntax: ANDLW k
Operands: 0 ≤ k ≤ 255
Operation: (W) .AND. k → W
Status Affected: N, Z
Encoding: 0000 1011 kkkk kkkk
Description: The contents of W are ANDed with the 8-bit literal ‘k’. The result is placed in W.
Words: 1
Cycles: 1
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal ‘k’</td>
<td>Process Data</td>
<td>Write to W</td>
</tr>
</tbody>
</table>

Example: ANDLW 05Fh
Before Instruction
W = A3h
After Instruction
W = 03h

Example: ADDWFc REG, 0, 1
Before Instruction
Carry bit = 1
REG = 02h
W = 4Dh
After Instruction
Carry bit = 0
REG = 02h
W = 50h
ANDWF

Syntax: ANDWF f, d, a

Operands:
- \(0 \leq f \leq 255\)
- \(d \in [0,1]\)
- \(a \in [0,1]\)

Operation: \((W) . \text{AND.} (f) \rightarrow \text{dest}\)

Status Affected: N, Z

Encoding: 0001 01da ffff ffff

Description: The contents of W are ANDed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \(f \leq 95\) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Example:

Before Instruction
- \(W = 17h\)
- \(\text{REG} = \text{C2h}\)

After Instruction
- \(W = 02h\)
- \(\text{REG} = \text{C2h}\)

BC

Syntax: BC n

Operands: \(-128 \leq n \leq 127\)

Operation: if Carry bit is '1';
\((\text{PC}) + 2 + 2n \rightarrow \text{PC}\)

Status Affected: None

Encoding: 1110 0010 nnnn nnnn

Description: If the Carry bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be \(\text{PC} + 2 + 2n\). This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

- Q1: Decode
- Q2: Read literal 'n'
- Q3: Process Data
- Q4: Write to PC

No operation

If No Jump:

- Q1: Decode
- Q2: Read literal 'n'
- Q3: Process Data
- Q4: No operation

Example:

Before Instruction
- \(\text{PC} = \text{address (HERE)}\)

After Instruction
- If Carry = 1;
- \(\text{PC} = \text{address (HERE + 12)}\)
- If Carry = 0;
- \(\text{PC} = \text{address (HERE + 2)}\)
**BCF**

**Bit Clear f**

**Syntax:**

```
BCF f, b {,a}
```

**Operands:**

- \(0 \leq f \leq 255\)
- \(0 \leq b \leq 7\)
- \(a \in [0,1]\)

**Operation:**

\(0 \rightarrow f < b>\)

**Status Affected:** None

**Encoding:**

```
1001 bbba ffff ffff
```

**Description:**

Bit 'b' in register 'f' is cleared.

- If 'a' is '0', the Access Bank is selected.
- If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \(f \leq 95\) (5Fh). See Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write register 'f'</td>
</tr>
</tbody>
</table>

**Example:**

```
BCF FLAG_REG, 7, 0
```

Before Instruction

- FLAG_REG = C7h

After Instruction

- FLAG_REG = 47h

---

**BN**

**Branch if Negative**

**Syntax:**

```– BN n
```

**Operands:**

\(-128 \leq n \leq 127\)

**Operation:**

- if Negative bit is '1', \((PC) + 2 + 2n \rightarrow PC\)

**Status Affected:** None

**Encoding:**

```
1110 0110 nnnn nnnn
```

**Description:** If the Negative bit is '1', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be \(PC + 2 + 2n\). This instruction is then a two-cycle instruction.

**Words:** 1

**Cycles:** 1(2)

**Q Cycle Activity:**

- If Jump:
  - Q1 Q2 Q3 Q4
  - Decode | Read literal 'n' | Process Data | Write to PC
  - No operation | No operation | No operation | No operation

- If No Jump:
  - Q1 Q2 Q3 Q4
  - Decode | Read literal 'n' | Process Data | No operation

**Example:**

```
HERE BN Jump
```

Before Instruction

PC = address (HERE)

After Instruction

- If Negative = 1:
  - PC = address (Jump)
- If Negative = 0:
  - PC = address (HERE + 2)
### BNC Branch if Not Carry

**Syntax:** BNC \( n \)

**Operands:** \(-128 \leq n \leq 127\)

**Operation:**
- if Carry bit is ‘0’,
- \((PC) + 2 + 2n \rightarrow PC\)

**Status Affected:** None

**Encoding:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110</td>
<td>0011</td>
<td>nnnn</td>
<td>nnnn</td>
</tr>
</tbody>
</table>

**Description:** If the Carry bit is ‘0’, then the program will branch. The 2’s complement number ‘2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be \(PC + 2 + 2n\). This instruction is then a two-cycle instruction.

**Words:** 1

**Cycles:** 1(2)

**Q Cycle Activity:**
- If Jump:
  - Q1 Q2 Q3 Q4
    - Decode: No operation
    - Read literal ‘n’: No operation
    - Process Data: No operation
    - Write to PC: No operation
- If No Jump:
  - Q1 Q2 Q3 Q4
    - Decode: Read literal ‘n’
    - Read literal ‘n’:
    - Process Data: No operation

**Example:**

Before Instruction

- PC = address (HERE)

After Instruction

- If Carry = 0;
  - PC = address (Jump)
- If Carry = 1;
  - PC = address (HERE + 2)

### BNN Branch if Not Negative

**Syntax:** BNN \( n \)

**Operands:** \(-128 \leq n \leq 127\)

**Operation:**
- if Negative bit is ‘0’,
- \((PC) + 2 + 2n \rightarrow PC\)

**Status Affected:** None

**Encoding:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110</td>
<td>0111</td>
<td>nnnn</td>
<td>nnnn</td>
</tr>
</tbody>
</table>

**Description:** If the Negative bit is ‘0’, then the program will branch. The 2’s complement number ‘2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be \(PC + 2 + 2n\). This instruction is then a two-cycle instruction.

**Words:** 1

**Cycles:** 1(2)

**Q Cycle Activity:**
- If Jump:
  - Q1 Q2 Q3 Q4
    - Decode: No operation
    - Read literal ‘n’: No operation
    - Process Data: No operation
    - Write to PC: No operation
- If No Jump:
  - Q1 Q2 Q3 Q4
    - Decode: Read literal ‘n’
    - Read literal ‘n’:
    - Process Data: No operation

**Example:**

Before Instruction

- PC = address (HERE)

After Instruction

- If Negative = 0;
  - PC = address (Jump)
- If Negative = 1;
  - PC = address (HERE + 2)
BNOV Branch if Not Overflow

Syntax: BNOV n

Operands: -128 ≤ n ≤ 127

Operation: if Overflow bit is ‘0’,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0101 | nnnn | nnnn |

Description: If the Overflow bit is ‘0’, then the program will branch.
The 2’s complement number ‘2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'n'</td>
<td>Process Data</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If No Jump:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'n'</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:
Before Instruction PC = address (HERE)
After Instruction If Overflow = 0;
PC = address (Jump)
If Overflow = 1;
PC = address (HERE + 2)

BNZ Branch if Not Zero

Syntax: BNZ n

Operands: -128 ≤ n ≤ 127

Operation: if Zero bit is ‘0’,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0001 | nnnn | nnnn |

Description: If the Zero bit is ‘0’, then the program will branch.
The 2’s complement number ‘2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'n'</td>
<td>Process Data</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If No Jump:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'n'</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:
Before Instruction PC = address (HERE)
After Instruction If Zero = 0;
PC = address (Jump)
If Zero = 1;
PC = address (HERE + 2)
BRA  Unconditional Branch
Syntax:  BRA  n
Operands:  -1024 ≤ n ≤ 1023
Operation:  (PC) + 2 + 2n → PC
Status Affected:  None
Encoding:  1101 0nnn nnnn nnnn
Description:  Add the 2’s complement number ‘2n’ to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction.

Words:  1
Cycles:  2

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal ‘n’</td>
<td>Process Data</td>
<td>Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:

Before Instruction
PC  = address (HERE)

After Instruction
PC  = address (Jump)

BSF  Bit Set f
Syntax:  BSF  f, b {,a}
Operands:  0 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]
Operation:  l → f<b>
Status Affected:  None
Encoding:  1000 bbba ffff ffff
Description:  Bit ‘b’ in register ‘f’ is set.
If ‘a’ is ‘0’, the Access Bank is selected.
If ‘a’ is ‘1’, the BSR is used to select the GPR bank.

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words:  1
Cycles:  1

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register ‘f’</td>
<td>Process Data</td>
<td>Write register ‘f’</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:

Before Instruction
FLAG_REG  = 0Ah
After Instruction
FLAG_REG  = 8Ah
### Bit Test File, Skip if Clear

**Syntax:**

\[ \text{BTFSC } f, b \{,a\} \]

**Operands:**

\[ 0 \leq f \leq 255 \]
\[ 0 \leq b \leq 7 \]
\[ a \in [0,1] \]

**Operation:**

skip if \((f<b>) = 0\)

**Status Affected:**

None

**Encoding:**

\[
\begin{array}{c}
1011 \\
bba \\
ffff \\
ffff
\end{array}
\]

**Description:**

If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.

- If 'a' is '0', the Access Bank is selected.
- If 'a' is '1', the BSR is used to select the GPR bank.
- If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \(f \leq 95\) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:**

1

**Cycles:**

1(2)

**Note:**

3 cycles if skip and followed by a 2-word instruction.

### Bit Test Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

Before Instruction

<table>
<thead>
<tr>
<th>PC</th>
<th>\text{Before Instruction}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address (\text{HERE})</td>
<td>PC = address (HERE)</td>
</tr>
</tbody>
</table>

After Instruction

<table>
<thead>
<tr>
<th>If</th>
<th>After Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAG&lt;1&gt;</td>
<td>If FLAG&lt;1&gt; = 0;</td>
</tr>
<tr>
<td>PC</td>
<td>PC = address (TRUE)</td>
</tr>
<tr>
<td>If</td>
<td>If FLAG&lt;1&gt; = 1;</td>
</tr>
<tr>
<td>PC</td>
<td>PC = address (FALSE)</td>
</tr>
</tbody>
</table>

### Bit Test File, Skip if Set

**Syntax:**

\[ \text{BTFSS } f, b \{,a\} \]

**Operands:**

\[ 0 \leq f \leq 255 \]
\[ 0 < b < 7 \]
\[ a \in [0,1] \]

**Operation:**

skip if \((f<b>) = 1\)

**Status Affected:**

None

**Encoding:**

\[
\begin{array}{c}
1010 \\
bba \\
ffff \\
ffff
\end{array}
\]

**Description:**

If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.

- If 'a' is '0', the Access Bank is selected.
- If 'a' is '1', the BSR is used to select the GPR bank.
- If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \(f < 95\) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:**

1

**Cycles:**

1(2)

**Note:**

3 cycles if skip and followed by a 2-word instruction.

### Bit Test Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

Before Instruction

<table>
<thead>
<tr>
<th>PC</th>
<th>\text{Before Instruction}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address (\text{HERE})</td>
<td>PC = address (HERE)</td>
</tr>
</tbody>
</table>

After Instruction

<table>
<thead>
<tr>
<th>If</th>
<th>After Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAG&lt;1&gt;</td>
<td>If FLAG&lt;1&gt; = 0;</td>
</tr>
<tr>
<td>PC</td>
<td>PC = address (FALSE)</td>
</tr>
<tr>
<td>If</td>
<td>If FLAG&lt;1&gt; = 1;</td>
</tr>
<tr>
<td>PC</td>
<td>PC = address (TRUE)</td>
</tr>
</tbody>
</table>
BTG Bit Toggle f

Syntax: BTG f, b {,a}
Operands: 0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]
Operation: (f<b>) → f<b>
Status Affected: None
Encoding: 0111 bbba ffff ffff
Description: Bit ‘b’ in data memory location ‘f’ is inverted.
   If ‘a’ is ‘0’, the Access Bank is selected.
   If ‘a’ is ‘1’, the BSR is used to select the GPR bank.
   If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates
   in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 “Byte-Oriented and
   Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1
Q Cycle Activity:

Example:
Before Instruction:
PORTC = 0111 0101 [75h]
After Instruction:
PORTC = 0110 0101 [65h]

BOV Branch if Overflow

Syntax: BOV n
Operands: -128 ≤ n ≤ 127
Operation: if Overflow bit is ‘1’;
(PC) + 2 + 2n → PC
Status Affected: None
Encoding: 1110 0100 nnnn nnnn
Description: If the Overflow bit is ‘1’, then the program will branch.
   The 2’s complement number ‘2n’ is added to the PC. Since the PC will have
   incremented to fetch the next instruction, the new address will be
   PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1
Cycles: 1(2)
Q Cycle Activity:
If Jump:

Example:
Before Instruction
PC = address (HERE)
After Instruction
PC = address (Jump)

If No Jump:

Example: HERE BOV Jump
Before Instruction
PC = address (HERE)
After Instruction
If Overflow = 1;
PCC = address (Jump)
If Overflow = 0;
PCC = address (HERE + 2)
**BZ**  Branch if Zero

Syntax:  BZ n

Operands:  \(-128 \leq n \leq 127\)

Operation:  if Zero bit is ‘1’,

\(\text{(PC)} + 2 + 2n \rightarrow \text{PC}\)

Status Affected:  None

Encoding:  

\[
\begin{array}{c|c|c|c}
 & 1110 & 0000 & nnnn \\
\hline
n & nnnn
\end{array}
\]

Description:  If the Zero bit is ‘1’, the program will branch.

The 2’s complement number ‘2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words:  1

Cycles:  1(2)

Q Cycle Activity:

If Jump:  

\[
\begin{array}{cccc}
Q1 & Q2 & Q3 & Q4 \\
\hline
\text{Decode} & \text{Read literal } & \text{Process} & \text{Write to} \\
\text{No operation} & \text{‘n’} & \text{Data} & \text{PC} \\
\text{No operation} & \text{No operation} & \text{No operation} & \text{No operation}
\end{array}
\]

If No Jump:  

\[
\begin{array}{cccc}
Q1 & Q2 & Q3 & Q4 \\
\hline
\text{Decode} & \text{Read literal } & \text{Process} & \text{No operation} \\
\text{No operation} & \text{‘n’} & \text{Data} & \text{No operation}
\end{array}
\]

Example:  

Before Instruction  

\[
\begin{align*}
\text{PC} & = \text{address (HERE)} \\
\text{After Instruction} & \begin{cases} 
\text{If Zero} = 1; \\
\text{PC} = \text{address (Jump)} \\
\text{If Zero} = 0; \\
\text{PC} = \text{address (HERE + 2)}
\end{cases}
\end{align*}
\]

**CALL**  Subroutine Call

Syntax:  CALL k {,s}

Operands:  \(0 \leq k \leq 1048575\)

\(s \in \{0,1\}\)

Operation:  

\[
\begin{align*}
\text{(PC)} + 4 & \rightarrow \text{TOS}, \\
k & \rightarrow \text{PC<20:1>}; \\
\text{if } s = 1, \\
\text{(W)} & \rightarrow \text{WS}, \\
\text{(STATUS)} & \rightarrow \text{STATUSS}, \\
\text{(BSR)} & \rightarrow \text{BSRS}
\end{align*}
\]

Status Affected:  None

Encoding:  

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c|c}
 & 1110 & 110a & k_{19}kkk & kkkk_{0} & 1111 \\
\hline
1st word \langle 7:0 \rangle & k_{7}kkk & kkkk & kkkk_{8}
\end{array}
\]

Description:  Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If ‘s’ = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If ‘s’ = 0, no update occurs (default). Then, the 20-bit value ‘k’ is loaded into PC<20:1>. CALL is a two-cycle instruction.

Words:  2

Cycles:  2

Q Cycle Activity:

\[
\begin{array}{cccc}
Q1 & Q2 & Q3 & Q4 \\
\hline
\text{Decode} & \text{Read literal } & \text{Push PC to} & \text{Read literal} \\
\text{No operation} & \text{‘k’<7:0>}, & \text{stack} & \text{‘k’<19:8>}, \\
\text{No operation} & \text{No operation} & \text{Write to PC} & \text{Write to PC} \\
\text{No operation} & \text{No operation} & \text{No operation} & \text{No operation}
\end{array}
\]

Example:  

Before Instruction  

\[
\begin{align*}
\text{PC} & = \text{address (HERE)} \\
\text{After Instruction} & \begin{cases} 
\text{PC} = \text{address (HERE)} \\
\text{TOS} = \text{address (HERE + 4)} \\
\text{WS} = \text{W} \\
\text{BSRS} = \text{BSR} \\
\text{STATUSS} = \text{STATUS}
\end{cases}
\end{align*}
\]
### CLRF
**Clear f**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>CLRF ( f {a} )</th>
</tr>
</thead>
</table>
| Operands        | \( 0 \leq f \leq 255 \)  \
|                 | \( a \in \{0,1\} \) |
| Operation       | \( 000h \rightarrow f, \)  \
|                 | \( 1 \rightarrow Z \) |
| Status Affected | \( Z \) |
| Encoding        | \[ \begin{array}{cccc}
0110 & 101a & ffff & ffff
\end{array} \] |
| Description     | Clears the contents of the specified register. \nIf 'a' is '0', the Access Bank is selected.  
If 'a' is '1', the BSR is used to select the GPR bank.  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. |

| Words | 1 |
| Cycles | 1 |

<table>
<thead>
<tr>
<th>Q Cycle Activity:</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>Process Data</td>
<td>No operation</td>
<td></td>
</tr>
<tr>
<td>Read register 'f'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write register 'f'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**  
CLRF  
FLAG_REG,1  

<table>
<thead>
<tr>
<th>Before Instruction</th>
<th>After Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAG_REG = 5Ah</td>
<td>FLAG_REG = 00h</td>
</tr>
</tbody>
</table>

### CLRWDT
**Clear Watchdog Timer**

<table>
<thead>
<tr>
<th>Syntax</th>
<th>CLRWDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>None</td>
</tr>
</tbody>
</table>
| Operation   | \( 000h \rightarrow \text{WDT}, \)  
|             | \( 000h \rightarrow \text{WDT postscaler}, \)  
|             | \( 1 \rightarrow \text{TO}, \)  
|             | \( 1 \rightarrow \text{PD} \) |
| Status Affected | \( \text{TO}, \text{PD} \) |
| Encoding     | \[ \begin{array}{cccc}
0000 & 0000 & 0000 & 0100
\end{array} \] |
| Description  | CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, \( \text{TO} \) and \( \text{PD} \), are set. |

| Words | 1 |
| Cycles | 1 |

<table>
<thead>
<tr>
<th>Q Cycle Activity:</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>Process Data</td>
<td>No operation</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**  
CLRWDT  

<table>
<thead>
<tr>
<th>Before Instruction</th>
<th>After Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDT Counter = ?</td>
<td>WDT Counter = 00h</td>
</tr>
<tr>
<td>WDT Postscaler = 0</td>
<td>TO = 1</td>
</tr>
<tr>
<td>PD = 1</td>
<td></td>
</tr>
</tbody>
</table>
**COMF** Complement f

**Syntax:** `COMF f {,d {,a}}`

**Operands:**
- `0 ≤ f ≤ 255`
- `d ∈ [0,1]`
- `a ∈ [0,1]`

**Operation:** `f → dest`

**Status Affected:** `N, Z`

**Encoding:** 0001 l1da ffff ffff

**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' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever `f ≤ 95 (5Fh)`. See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**Example:**
- **Before Instruction**
  - `REG = 13h`
  - `W = ECh`
  - `PC Address = HERE`

- **After Instruction**
  - `REG = 13h`
  - `W = ECh`

**CPFSEQ** Compare f with W, Skip if f = W

**Syntax:** `CPFSEQ f {,a}`

**Operands:**
- `0 ≤ f ≤ 255`
- `a ∈ [0,1]`

**Operation:** `(f) – (W), skip if (f) = (W)` (unsigned comparison)

**Status Affected:** None

**Encoding:** 0110 001a ffff ffff

**Description:** Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.

If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever `f ≤ 95 (5Fh)`. See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1

**Cycles:** 1(2)

**Note:** 3 cycles if skip and followed by a 2-word instruction.

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**If skip:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**If skip and followed by 2-word instruction:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**
- **HERE** CPFSEQ REG, 0
  - NEQUAL :
  - EQUAL :

  **Before Instruction**
  - `PC Address = HERE`
  - `W = ?`
  - `REG = ?`

  **After Instruction**
  - If `REG = W;`
  - If `REG ≠ W;`
  - `PC = Address (EQUAL)`
  - `PC = Address (NEQUAL)`
CPFSGT  Compare f with W, Skip if f > W
Syntax: CPFSGT  f {,a}
Operands: 0 ≤ f ≤ 255
a ∈ [0,1]
Operation: (f) – (W), skip if (f) > (W)
(unsigned comparison)
Status Affected: None
Encoding: 0110 010a ffff ffff
Description: Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction.
If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.
Words: 1
Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.
Q Cycle Activity:
Q1 Q2 Q3 Q4
Decode Read Process No
register 'f' Data operation
If skip:
Q1 Q2 Q3 Q4
No operation No operation No operation No operation
If skip and followed by 2-word instruction:
Q1 Q2 Q3 Q4
No operation No operation No operation No operation
Example: HERE CPFSGT REG, 0
NGREATER :
GREATER :
Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG > W:
PC = Address (GREATER)
If REG ≤ W:
PC = Address (NGREATER)

CPFSLT  Compare f with W, Skip if f < W
Syntax: CPFSLT  f {,a}
Operands: 0 ≤ f ≤ 255
a ∈ [0,1]
Operation: (f) – (W), skip if (f) < (W)
(unsigned comparison)
Status Affected: None
Encoding: 0110 000a ffff ffff
Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
Words: 1
Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.
Q Cycle Activity:
Q1 Q2 Q3 Q4
Decode Read Process No
register 'f' Data operation
If skip:
Q1 Q2 Q3 Q4
No operation No operation No operation No operation
If skip and followed by 2-word instruction:
Q1 Q2 Q3 Q4
No operation No operation No operation No operation
Example: HERE CPFSLT REG, 1
NLESS :
LESS :
Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG < W:
PC = Address (LESS)
If REG ≥ W:
PC = Address (NLESS)
**DAW**

**Decimal Adjust W Register**

**Syntax:** DAW

**Operands:** None

**Operation:** If \([W<3:0> > 9]\) or \([DC = 1]\) then, 
\((W<3:0>) + 6 \rightarrow W<3:0>\); 
else, 
\((W<3:0>) \rightarrow W<3:0>\)

If \([W<7:4> > 9]\) or \([C = 1]\) then, 
\((W<7:4>) + 6 \rightarrow W<7:4>\), 
\(C = 1\); 
else, 
\((W<7:4>) \rightarrow W<7:4>\)

**Status Affected:** C

**Encoding:**

```
0000 0000 0000 0111
```

**Description:** DAW adjusts the eight-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register W</td>
<td>Process Data</td>
<td>Write to W</td>
</tr>
</tbody>
</table>

**Example 1:** DAW

**Before Instruction**

\(W = A5h\)
\(C = 0\)
\(DC = 0\)

**After Instruction**

\(W = 05h\)
\(C = 1\)
\(DC = 0\)

**Example 2:**

**Before Instruction**

\(W = CEnh\)
\(C = 0\)
\(DC = 0\)

**After Instruction**

\(W = 34h\)
\(C = 1\)
\(DC = 0\)

---

**DECF**

**Decrement f**

**Syntax:** DECF \(f\{d,a\}\)

**Operands:**

- \(0 \leq f \leq 255\)
- \(d \in [0,1]\)
- \(a \in [0,1]\)

**Operation:** \((f) - 1 \rightarrow dest\)

**Status Affected:** C, DC, N, OV, Z

**Encoding:**

```
0000 01da ffff ffff
```

**Description:** Decrement register \(f\). If \(d\) is '0', the result is stored in W. If \(d\) is '1', the result is stored back in register \(f\) (default).

If \(a\) is '0', the Access Bank is selected. If \(a\) is '1', the BSR is used to select the GPR bank.

If \(a\) is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \(f \leq 95\) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register (f)</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**Example:** DECF CNT, 1, 0

**Before Instruction**

\(CNT = 01h\)
\(Z = 0\)

**After Instruction**

\(CNT = 00h\)
\(Z = 1\)
DECFSZ Decrement f, Skip if 0

Syntax: 
DECFSZ f, {d, {a}}

Operands:

\[ 0 \leq f \leq 255 \]
\[ d \in [0, 1] \]
\[ a \in [0, 1] \]

Operation:

\[ (f) \rightarrow \text{dest}, \skip \text{if result = 0} \]

Status Affected: None

Encoding:

\[ 0010 \quad \text{lda} \quad \text{ffff} \quad \text{ffff} \]

Description:

The contents of register ‘f’ are decremented. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default).

If the result is ‘0’, the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1(2)

Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:

HERE     DECFSZ    CNT, 1, 1
CONTINUE
GOTO     LOOP

DCFSNZ Decrement f, Skip if not 0

Syntax: 
DCFSNZ f, {d, {a}}

Operands:

\[ 0 \leq f \leq 255 \]
\[ d \in [0, 1] \]
\[ a \in [0, 1] \]

Operation:

\[ (f) \rightarrow \text{dest}, \skip \text{if result \neq 0} \]

Status Affected: None

Encoding:

\[ 0100 \quad \text{lda} \quad \text{ffff} \quad \text{ffff} \]

Description:

The contents of register ‘f’ are decremented. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default).

If the result is not ‘0’, the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1(2)

Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

If skip:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:

HERE     DCFSNZ    TEMP, 1, 0
ZERO : 
NZERO :

Before Instruction
TEMP = ?
After Instruction
TEMP = TEMP – 1,
If TEMP = 0;
PC = Address (ZERO)
If TEMP \neq 0;
PC = Address (NZERO)
GOTO          Unconditional Branch
Syntax:  GOTO   k
Operands:  0 ≤ k ≤ 1048575
Operation:  k → PC<20:1>
Status Affected:  None
Encoding:
1st word (k<7:0>)
1110 1111  k,kkk  kkkk0
2nd word(k<19:8>)
1111  k13kkk  kkkk  kkkk0
Description:
GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle instruction.
Words: 2
Cycles: 2
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'k'&lt;7:0&gt;,</td>
<td>No operation</td>
<td>Read literal 'k'&lt;19:8&gt;, Write to PC</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:  GOTO THERE
After Instruction
PC = Address (THERE)

INCF          Increment f
Syntax:  INCF   f {,d {,a}}
Operands:  0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]
Operation:  (f) + 1 → dest
Status Affected:  C, DC, N, OV, Z
Encoding:
0010 10da  ffff  ffff
Description:
The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.
Words: 1
Cycles: 1
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example:  INCF    CNT, 1, 0
Before Instruction
CNT = FFh
Z = 0
C = ?
DC = ?
After Instruction
CNT = 00h
Z = 1
C = 1
DC = 1
## INCFSZ: Increment f, Skip if 0

**Syntax:**

\[
\text{INCFSZ } f \{,d \{,a\}\}
\]

**Operands:**

\[
0 \leq f \leq 255 \\
d \in [0,1] \\
a \in [0,1]
\]

**Operation:**

\[
(f) + 1 \rightarrow \text{dest}, \quad \text{skip if result } = 0
\]

**Status Affected:** None

**Encoding:**

\[
0011 \ 11da \ ffff \ ffff
\]

**Description:**

The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. (default)

If the result is '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1

**Cycles:** 1(2)

**Example:**

```
HERE    INCFSZ CNT, 1, 0 
NZERO   :
ZERO   :
```

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

```
HERE    INCFSZ CNT, 1, 0 
NZERO   :
ZERO   :
```

## INFSNZ: Increment f, Skip if not 0

**Syntax:**

\[
\text{INFSNZ } f \{,d \{,a\}\}
\]

**Operands:**

\[
0 \leq f \leq 255 \\
d \in [0,1] \\
a \in [0,1]
\]

**Operation:**

\[
(f) + 1 \rightarrow \text{dest}, \quad \text{skip if result } \neq 0
\]

**Status Affected:** None

**Encoding:**

\[
0100 \ 10da \ ffff \ ffff
\]

**Description:**

The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If the result is not '0', the next instruction which is already fetched is discarded and a NOP is executed instead, making it a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1

**Cycles:** 1(2)

**Example:**

```
HERE    INFSNZ REG, 1, 0 
NZERO   :
ZERO   :
```

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

```
HERE    INFSNZ CNT, 1, 0 
NZERO   :
ZERO   :
```
IORLW  Inclusive OR Literal with W

Syntax:   IORLW  k
Operands:  0 ≤ k ≤ 255
Operation:   (W) .OR. k → W
Status Affected:   N, Z
Encoding:   0000 1001 kkkk kkkk
Description: The contents of W are ORed with the eight-bit literal 'k'. The result is placed in W.
Words:  1
Cycles:  1

Q Cycle Activity:

Decide | Read literal 'k' | Process Data | Write to W

Example:   IORLW  35h
Before Instruction
W = 9Ah
After Instruction
W = BFh

IORWF  Inclusive OR W with f

Syntax:   IORWF  f(d, (a))
Operands:  0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]
Operation:   (W) .OR. (f) → dest
Status Affected:   N, Z
Encoding:   0001 00da ffff ffff
Description: Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.
Words:  1
Cycles:  1

Q Cycle Activity:

Decide | Read register 'f' | Process Data | Write to destination

Example:   IORWF  RESULT, 0, 1
Before Instruction
RESULT = 13h
W = 91h
After Instruction
RESULT = 13h
W = 93h
LFSR Load FSR

Syntax: LFSR f, k

Operands: 0 ≤ f ≤ 2
0 ≤ k ≤ 4095

Operation: k → FSRf

Status Affected: None

Encoding:

<table>
<thead>
<tr>
<th>Value</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110</td>
<td>1110 00ff kkk kkk</td>
</tr>
<tr>
<td>1111</td>
<td>0000 kkk kkk</td>
</tr>
</tbody>
</table>

Description: The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'.

Words: 2
Cycles: 2

Example: LFSR 2, 3ABh

After Instruction

FSR2H = 03h
FSR2L = ABh

MOVF Move f

Syntax: MOVF f, d, a

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: f → dest

Status Affected: N, Z

Encoding:

<table>
<thead>
<tr>
<th>Value</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>0101</td>
<td>00da fffe fffe</td>
</tr>
</tbody>
</table>

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256-byte bank.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1

Example: MOVF REG, 0, 0

Before Instruction

REG = 22h
W = FFh

After Instruction

REG = 22h
W = 22h
### MOVFF: Move f to f

**Syntax:**

`MOVFF f_s, f_d`

**Operands:**

- `0 ≤ f_s ≤ 4095`
- `0 ≤ f_d ≤ 4095`

**Operation:**

`(f_s) → f_d`

**Status Affected:** None

**Encoding:**

<table>
<thead>
<tr>
<th>1st word (source)</th>
<th>2nd word (destin.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1100</td>
<td>fff  fff  fff  fff</td>
</tr>
<tr>
<td>1111</td>
<td>fff  fff  fff  fff</td>
</tr>
</tbody>
</table>

**Description:**

The contents of source register `f_s` are moved to destination register `f_d`. Location of source `f_s` can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination `f_d` can also be anywhere from 000h to FFFh. Either source or destination can be W (a useful special situation).

MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).

The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

**Words:** 2

**Cycles:** 2

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register ‘f’ (src)</td>
<td>Process Data</td>
<td>No operation</td>
</tr>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>No operation</td>
<td>Write register ‘f’ (dest)</td>
</tr>
</tbody>
</table>

**Example:**

MOVFF REG1, REG2

Before Instruction

<table>
<thead>
<tr>
<th>REG1</th>
<th>REG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>33h</td>
<td>11h</td>
</tr>
</tbody>
</table>

After Instruction

<table>
<thead>
<tr>
<th>REG1</th>
<th>REG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>33h</td>
<td>33h</td>
</tr>
</tbody>
</table>

### MOVLB: Move Literal to Low Nibble in BSR

**Syntax:**

`MOVLB k`

**Operands:**

- `0 ≤ k ≤ 255`

**Operation:**

`k → BSR`

**Status Affected:** None

**Encoding:**

<table>
<thead>
<tr>
<th>0000</th>
<th>0001</th>
<th>kkkk</th>
<th>kkkk</th>
</tr>
</thead>
</table>

**Description:**

The eight-bit literal ‘k’ is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains ‘0’ regardless of the value of k7:k4.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal ‘k’</td>
<td>Process Data</td>
<td>Write literal ‘k’ to BSR</td>
</tr>
</tbody>
</table>

**Example:**

MOVLB 5

Before Instruction

<table>
<thead>
<tr>
<th>BSR Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>02h</td>
</tr>
</tbody>
</table>

After Instruction

<table>
<thead>
<tr>
<th>BSR Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>05h</td>
</tr>
</tbody>
</table>
MOVLW Move Literal to W

Syntax: MOVLW k
Operands: 0 ≤ k ≤ 255
Operation: k → W
Status Affected: None
Encoding: 0000 1110 kkkk kkkk
Description: The eight-bit literal 'k' is loaded into W.
Words: 1
Cycles: 1
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'k'</td>
<td>Process Data</td>
<td>Write to W</td>
</tr>
</tbody>
</table>

Example: MOVLW 5Ah
After Instruction
W = 5Ah

MOVWF Move W to f

Syntax: MOVWF f {,a}
Operands: 0 ≤ f ≤ 255
a ∈ [0,1]
Operation: (W) → f
Status Affected: None
Encoding: 0110 111a ffff ffff
Description: Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.
Words: 1
Cycles: 1
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write register 'f'</td>
</tr>
</tbody>
</table>

Example: MOVWF REG, 0
Before Instruction
W = 4Fh
REG = FFh
After Instruction
W = 4Fh
REG = 4Fh
### MULLW Multiply Literal with W

**Syntax:** MULLW k

**Operands:** 0 ≤ k ≤ 255

**Operation:** (W) x k → PRODH:PRODL

**Status Affected:** None

**Encoding:**

| 0000 | 1101 | kkkk | kkkk |

**Description:** An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.

None of the Status flags are affected.

Note that neither Overflow nor Carry is possible in this operation. A Zero result is possible but not detected.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'k'</td>
<td>Process Data</td>
<td>Write registers PRODH: PRODL</td>
</tr>
</tbody>
</table>

**Example:** MULLW 0C4h

**Before Instruction**

<table>
<thead>
<tr>
<th>W</th>
<th>PRODH</th>
<th>PRODL</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2h</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

**After Instruction**

<table>
<thead>
<tr>
<th>W</th>
<th>PRODH</th>
<th>PRODL</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2h</td>
<td>ADh</td>
<td>08h</td>
</tr>
</tbody>
</table>

### MULWF Multiply W with f

**Syntax:** MULWF f {,a}

**Operands:** 0 ≤ f ≤ 255

a ∈ [0,1]

**Operation:** (W) x (f) → PRODH:PRODL

**Status Affected:** None

**Encoding:**

| 0000 | 001a | ffff | ffff |

**Description:** An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged.

None of the Status flags are affected.

Note that neither Overflow nor Carry is possible in this operation. A Zero result is possible but not detected.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write registers PRODH: PRODL</td>
</tr>
</tbody>
</table>

**Example:** MULWF REG, 1

**Before Instruction**

<table>
<thead>
<tr>
<th>W</th>
<th>REG</th>
<th>PRODH</th>
<th>PRODL</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4h</td>
<td>B5h</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

**After Instruction**

<table>
<thead>
<tr>
<th>W</th>
<th>REG</th>
<th>PRODH</th>
<th>PRODL</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4h</td>
<td>B5h</td>
<td>8Ah</td>
<td>94h</td>
</tr>
</tbody>
</table>
NEGF Negate f

| Syntax:     | NEGF f{a} |
| Operands:   | 0 ≤ f ≤ 255   
| a ∈ [0,1]  |
| Operation:  | (f) + 1 → f  |
| Status Affected: | N, OV, C, DC, Z |
| Encoding:   | 0110 110a ffff ffff |

Description: Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'.

If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words: 1
Cycles: 1

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write register 'f'</td>
</tr>
</tbody>
</table>

Example:

NEGF REG, 1

Before Instruction
REG = 0011 1010 [3Ah]

After Instruction
REG = 1100 0110 [C6h]

NOP No Operation

| Syntax:     | NOP |
| Operands:   | None |
| Operation:  | No operation |
| Status Affected: | None |
| Encoding:   | 0000 0000 0000 0000 1111 xxxx xxxx xxxx |

Description: No operation.

Words: 1
Cycles: 1

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example: None.
### POP

**Pop Top of Return Stack**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>POP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>None</td>
</tr>
<tr>
<td>Operation:</td>
<td>(TOS) → bit bucket</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td><img src="image" alt="Encoding" /></td>
</tr>
<tr>
<td>Description:</td>
<td>The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td><img src="image" alt="Activity" /></td>
</tr>
</tbody>
</table>

**Example:**

Before Instruction
- TOS = 0031A2h
- Stack (1 level down) = 014332h

After Instruction
- TOS = 014332h
- PC = NEW

### PUSH

**Push Top of Return Stack**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>PUSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>None</td>
</tr>
<tr>
<td>Operation:</td>
<td>(PC + 2) → TOS</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td><img src="image" alt="Encoding" /></td>
</tr>
<tr>
<td>Description:</td>
<td>The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td><img src="image" alt="Activity" /></td>
</tr>
</tbody>
</table>

**Example:**

Before Instruction
- TOS = 345Ah
- PC = 0124h

After Instruction
- TOS = 0126h
- PC = 345Ah
### RCALL
**Relative Call**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>RCALL ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>-1024 ( \leq n \leq 1023 )</td>
</tr>
<tr>
<td>Operation:</td>
<td>(PC) + 2 ( \rightarrow ) TOS, ((PC) + 2 + 2n \rightarrow PC)</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>\begin{tabular}{cccc} 1101 \ 1nnn \ nnnn \ nnnn \end{tabular}</td>
</tr>
<tr>
<td>Description:</td>
<td>Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>2</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td></td>
</tr>
<tr>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>Decode</td>
<td>Read literal 'n' ( \rightarrow ) PUSH PC to stack</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**

Before Instruction:
- PC = Address \( \text{(HERE)} \)

After Instruction:
- PC = Address \( \text{(Jump)} \)
- TOS = Address \( \text{(HERE + 2)} \)

### RESET
**Reset**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>RESET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>None</td>
</tr>
<tr>
<td>Operation:</td>
<td>Reset all registers and flags that are affected by a MCLR Reset.</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>All</td>
</tr>
<tr>
<td>Encoding:</td>
<td>\begin{tabular}{cccc} 0000 \ 0000 \ 1111 \ 1111 \end{tabular}</td>
</tr>
<tr>
<td>Description:</td>
<td>This instruction provides a way to execute a MCLR Reset in software.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td></td>
</tr>
<tr>
<td>Q1</td>
<td>Q2</td>
</tr>
<tr>
<td>Decode</td>
<td>Start reset</td>
</tr>
</tbody>
</table>

**Example:**

Before Instruction:
- Registers = Reset Value
- Flags* = Reset Value

After Instruction:
- Registers = Reset Value
- Flags* = Reset Value

**Example:**

RESET
RETIFIE  Return from Interrupt  

**Syntax:**  RETFIE  {s}  

**Operands:**  s ∈ [0,1]  

**Operation:**  (TOS) → PC,  

1 → GIE/GIEH or PEIE/GIEL;  

if s = 1,  

(WS) → W,  

(STATUSS) → STATUS,  

(BSRS) → BSR,  

PCLATU, PCLATH are unchanged  

**Status Affected:**  GIE/GIEH, PEIE/GIEL.  

**Encoding:**  

| 0000 | 0000 | 0001 | 000s |  

**Description:**  Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low-priority global interrupt enable bit. If ‘s’ = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers W, STATUS and BSR. If ‘s’ = 0, no update of these registers occurs (default).  

**Words:**  1  

**Cycles:**  2  

**Q Cycle Activity:**  

| Q1 | Q2 | Q3 | Q4 |  

| Decode | No operation | No operation | No operation | No operation |  

**Example:**  RETFIE  1  

After Interrupt  

| PC | W | BSR | STATUS | GIE/GIEH, PEIE/GIEL |  

| = | TOS | = | WS | = |  

| = | BSRS | = | STATUS | = |  

| = | 1 |  

RETIFIE  Return from Interrupt  

**Syntax:**  RETLW  k  

**Operands:**  0 ≤ k ≤ 255  

**Operation:**  k → W,  

(TOS) → PC,  

PCLATU, PCLATH are unchanged  

**Status Affected:**  None  

**Encoding:**  

| 0000 | 1100 | kkkk | kkkk |  

**Description:**  W is loaded with the eight-bit literal ‘k’. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.  

**Words:**  1  

**Cycles:**  2  

**Q Cycle Activity:**  

| Q1 | Q2 | Q3 | Q4 |  

| Decode | Read literal ‘k’ | Process Data | POP PC from stack, write to W |  

| No operation | No operation | No operation | No operation |  

**Example:**  

CALL TABLE ; W contains table  

; offset value  

; W now has  

; table value  

;  

TABLE  

ADDWF PCL ; W = offset  

RETLW k0 ; Begin table  

RETLW k1 ;  

;  

RETLW kn ; End of table  

Before Instruction  

| W | 07h |  

After Instruction  

| W | value of kn |
**RETURN**  
Return from Subroutine

**Syntax:**
\[ \text{RETURN} \ {s} \]

**Operands:**
\( s \in [0,1] \)

**Operation:**
- \((\text{TOS}) \rightarrow \text{PC} ;\)
- If \( s = 1 \), \((\text{WS}) \rightarrow \text{W},\)
- \((\text{STATUS}) \rightarrow \text{STATUS},\)
- \((\text{BSRS}) \rightarrow \text{BSR},\)
- \(\text{PCLATU}, \text{PCLATH} \) are unchanged

**Status Affected:**
None

**Encoding:**
0000 0000 0001 001s

**Description:**
Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If \( s = 1 \), the contents of the shadow registers WS, STATUS and BSRS are loaded into their corresponding registers W, STATUS and BSR. If \( s = 0 \), no update of these registers occurs (default).

**Words:**
1

**Cycles:**
2

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>Process Data</td>
<td>POP PC from stack</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**
\[ \text{RETURN} \]

After Instruction:
\( \text{PC} = \text{TOS} \)

---

**RLCF**  
Rotate Left \( f \) through Carry

**Syntax:**
\[ \text{RLCF} \ f, d, a \]

**Operands:**
- \( 0 \leq f \leq 255 \)
- \( d \in [0,1] \)
- \( a \in [0,1] \)

**Operation:**
- \((f_{<n>}) \rightarrow \text{dest}_{<n+1>} ,\)
- \((f_{<7>}) \rightarrow \text{C},\)
- \((\text{C}) \rightarrow \text{dest}_{<0>} \)

**Status Affected:**
C, N, Z

**Encoding:**
0011 01da ffff ffff

**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 W. If 'd' is '1', the result is stored back in register 'f' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:**
1

**Cycles:**
1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

**Example:**
\[ \text{RLCF REG, 0, 0} \]

Before Instruction:
\[ \begin{align*}
\text{REG} &= 1110\ 0110 \\
\text{C} &= 0
\end{align*} \]

After Instruction:
\[ \begin{align*}
\text{REG} &= 1110\ 0110 \\
\text{W} &= 1100\ 1100 \\
\text{C} &= 1
\end{align*} \]
## RLNCF Rotate Left f (No Carry)

**Syntax:**

```
RLNCF f,{d, {a}}
```

**Operands:**

- \( f \) \( \leq 255 \)
- \( d \in [0, 1] \)
- \( a \in [0, 1] \)

**Operation:**

\[
\begin{align*}
(f_{<n>}) & \rightarrow \text{dest}_{<n+1>}, \\
(f_{<7>}) & \rightarrow \text{dest}_{<0>}
\end{align*}
\]

**Status Affected:**

- \( N \)
- \( Z \)

**Encoding:**

```
0100 01da ffff ffff
```

**Description:**

The contents of register \('f'\) are rotated one bit to the left. If \('d'\) is \('0'\), the result is placed in \(W\). If \('d'\) is \('1'\), the result is stored back in register \('f'\) (default).

If \('a'\) is \('0'\), the Access Bank is selected. If \('a'\) is \('1'\), the BSR is used to select the GPR bank.

If \('a'\) is \('0'\) and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.


### Example:

Before Instruction

- \( \text{REG} = 1010 \ 1011 \)
- \( \text{C} = 0 \)

After Instruction

- \( \text{REG} = 0101 \ 0111 \)
- \( \text{C} = 0 \)

## RRCF Rotate Right f through Carry

**Syntax:**

```
RRCF f,{d, {a}}
```

**Operands:**

- \( f \) \( \leq 255 \)
- \( d \in [0, 1] \)
- \( a \in [0, 1] \)

**Operation:**

\[
\begin{align*}
(f_{<n>}) & \rightarrow \text{dest}_{<n-1>}, \\
(f_{<0>}) & \rightarrow C, \\
(C) & \rightarrow \text{dest}_{<7>}
\end{align*}
\]

**Status Affected:**

- \( C \)
- \( N \)
- \( Z \)

**Encoding:**

```
0011 00da ffff ffff
```

**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 \(W\). If \('d'\) is \('1'\), the result is placed back in register \('f'\) (default).

If \('a'\) is \('0'\), the Access Bank is selected. If \('a'\) is \('1'\), the BSR is used to select the GPR bank.

If \('a'\) is \('0'\) and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.
RRNCF  Rotate Right f (No Carry)

Syntax: RRNCF f{,d {,a}}
Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]
Operation: \( (f^{<n>} \rightarrow \text{dest}^{<n-1>}, \)  
\( (f^{<0>} \rightarrow \text{dest}^{<7>}) \)
Status Affected: N, Z
Encoding: 0100 00da ffff ffff
Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
  
If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).
  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1

Example 1:

RRNCF  REG, 1, 0

Before Instruction  
REG = 1101 0111  
After Instruction  
REG = 1110 1011

Example 2:

RRNCF  REG, 0, 0

Before Instruction  
W = ?  
REG = 1101 0111  
After Instruction  
W = 1110 1011  
REG = 1101 0111

SETF  Set f

Syntax: SETF f{,a}
Operands: 0 ≤ f ≤ 255
a ∈ [0,1]
Operation: FFh → f
Status Affected: None
Encoding: 0110 100a ffff ffff
Description: The contents of the specified register are set to FFh.
  
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
  
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1

Example:

SETF  REG, 1

Before Instruction  
REG = 5Ah  
After Instruction  
REG = FFh
SLEEP

Enter Sleep Mode

Syntax: SLEEP
Operands: None
Operation: 00h → WDT,
0 → WDT postscaler,
1 → TO,
0 → PD
Status Affected: TO, PD
Encoding: 0000 0000 0000 0011
Description: The Power-Down status bit (PD) is cleared. The Time-out status bit (TO) is set. The Watchdog Timer and its postscaler are cleared.
The processor is put into Sleep mode with the oscillator stopped.

Words: 1
Cycles: 1
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>Process Data</td>
<td>Go to Sleep</td>
</tr>
</tbody>
</table>

Example:

Before Instruction
TO = ?
PD = ?

After Instruction
TO = 1†
PD = 0

† If WDT causes wake-up, this bit is cleared.

SUBFWB

Subtract f from W with Borrow

Syntax: SUBFWB f {,d {,a}}
Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]
Operation: (W) – (f) – (C) → dest
Status Affected: N, OV, C, DC, Z
Encoding: 0101 01da ffff ffff
Description: Subtract register 'f' and Carry flag (borrow) from W (2’s complement method). If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored in register ‘f’ (default).

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

Words: 1
Cycles: 1
Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

Example 1:

Before Instruction
REG = 3
W = 2
C = 1
After Instruction
REG = FF
W = 2
C = 0
Z = 0
N = 1 ; result is negative

Example 2:

Before Instruction
REG = 2
W = 5
C = 1
After Instruction
REG = 2
W = 3
C = 1
Z = 0
N = 0 ; result is positive

Example 3:

Before Instruction
REG = 1
W = 2
C = 0
After Instruction
REG = 0
W = 2
C = 1
Z = 1 ; result is zero
N = 0
### SUBLW Subtract W from Literal

**Syntax:**  
SUBLW  \( k \)

**Operands:**  
0 \( \leq k \leq 255 \)

**Operation:**  
\( k - (W) \rightarrow W \)

**Status Affected:**  
N, OV, C, DC, Z

**Encoding:**  
0000 1000 kkkk kkkk

**Description:**  
W is subtracted from the eight-bit literal ‘k’. The result is placed in W.

**Words:** 1

**Cycles:** 1

#### Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal ‘k’</td>
<td>Process Data</td>
<td>Write to W</td>
</tr>
</tbody>
</table>

**Example 1:**  
SUBLW 02h

Before Instruction  
W = 01h  
C = ?

After Instruction  
W = 01h  
C = 1 ; result is positive  
Z = 0  
N = 0

**Example 2:**  
SUBLW 02h

Before Instruction  
W = 02h  
C = ?

After Instruction  
W = 00h  
C = 1 ; result is zero  
Z = 1  
N = 0

**Example 3:**  
SUBLW 02h

Before Instruction  
W = 03h  
C = ?

After Instruction  
W = FFh ; (2’s complement)  
C = 0 ; result is negative  
Z = 0  
N = 1

### SUBWF Subtract W from f

**Syntax:**  
SUBWF  \( f (d \,(a)) \)

**Operands:**  
0 \( \leq f \leq 255 \)  
d \( \in \{0,1\} \)  
a \( \in \{0,1\} \)

**Operation:**  
(\( f \)) - (W) \( \rightarrow \) dest

**Status Affected:**  
N, OV, C, DC, Z

**Encoding:**  
0101 11da ffff ffff

**Description:**  
Subtract W 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’ (default).

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1

**Cycles:** 1

#### Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register ‘f’</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**Example 1:**  
SUBWF  REG, 1, 0

Before Instruction  
REG = 3  
W = 2  
C = ?

After Instruction  
REG = 1  
W = 2  
C = 1 ; result is positive  
Z = 0  
N = 0

**Example 2:**  
SUBWF  REG, 0, 0

Before Instruction  
REG = 2  
W = 2  
C = ?

After Instruction  
REG = 2  
W = 0  
C = 1 ; result is zero  
Z = 1  
N = 0

**Example 3:**  
SUBWF  REG, 1, 0

Before Instruction  
REG = 1  
W = 2  
C = ?

After Instruction  
REG = FFh ; (2’s complement)  
W = 2  
C = 0 ; result is negative  
Z = 0  
N = 1
### SUBWFB
**Subtract W from f with Borrow**

**Syntax:**  
SUBWFB  \( f \{,d\ \{,a\}\} \)

**Operands:**  
\( 0 \leq f \leq 255 \)  
\( d \in [0,1] \)  
\( a \in [0,1] \)

**Operation:**  
\((f) - (W) - (C) \rightarrow \text{dest}\)

**Status Affected:**  
N, OV, C, DC, Z

**Encoding:**  
```
0101 10da ffff ffff
```

**Description:**  
Subtract W and the Carry flag (borrow) 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' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1  
**Cycles:** 1

### SWAPF
**Swap f**

**Syntax:**  
SWAPF  \( f \{,d\ \{,a\}\} \)

**Operands:**  
\( 0 \leq f \leq 255 \)  
\( d \in [0,1] \)  
\( a \in [0,1] \)

**Operation:**  
\((f<3:0>) \rightarrow \text{dest}<7:4>\),  
\((f<7:4>) \rightarrow \text{dest}<3:0>\)

**Status Affected:** None

**Encoding:**  
```
0011 10da ffff ffff
```

**Description:**  
The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever \( f \leq 95 \) (5Fh). See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:** 1  
**Cycles:** 1

---

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read register 'f'</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

#### Example 1:
**Before Instruction**

- REG = 19h  
- W = 0Dh  
- C = 1

**After Instruction**

- REG = 0Ch  
- W = 0Dh  
- C = 1

- Z = 0  
- N = 0  

: result is positive

#### Example 2:

**Before Instruction**

- REG = 1Bh  
- W = 1Ah  
- C = 0

**After Instruction**

- REG = 1Bh  
- W = 00h  
- C = 1

- Z = 1  
- N = 0  

: result is zero

#### Example 3:

**Before Instruction**

- REG = 03h  
- W = 0Eh  
- C = 1

**After Instruction**

- REG = F5h  
- W = 0Eh  
- C = 0

- Z = 0  
- N = 1  

: result is negative
TBLRD Table Read

Syntax: TBLRD ( *; *+; *-; +*)

Operands: None

Operation:
- if TBLRD *,
  (Prog Mem (TBLPTR)) → TABLAT,
  TBLPTR – No Change;
- if TBLRD *+,
  (Prog Mem (TBLPTR)) → TABLAT,
  (TBLPTR) + 1 → TBLPTR;
- if TBLRD *-,
  (Prog Mem (TBLPTR)) → TABLAT,
  (TBLPTR) – 1 → TBLPTR;
- if TBLRD +*,
  (TBLPTR) + 1 → TBLPTR,
  (Prog Mem (TBLPTR)) → TABLAT

Status Affected: None

Encoding:

<table>
<thead>
<tr>
<th>0000</th>
<th>0000</th>
<th>0000</th>
<th>10nn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1 *+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-2 *-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-3 ++</td>
</tr>
</tbody>
</table>

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

- TBLPTR<0> = 0: Least Significant Byte of Program Memory Word
- TBLPTR<0> = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:
- no change
- post-increment
- post-decrement
- pre-increment

Words: 1
Cycles: 2

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation (Read Program Memory)</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation (Write TABLAT)</td>
</tr>
</tbody>
</table>
**TBLWT**  
**Table Write**  

<table>
<thead>
<tr>
<th>Syntax</th>
<th>TBLWT (&quot;*; *+; <em>-; +</em>&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands</td>
<td>None</td>
</tr>
</tbody>
</table>
| Operation       | if TBLWT*;
|                 | (TABLAT) → Holding Register,
|                 | TBLPTR – No Change;
|                 | if TBLWT*+;
|                 | (TABLAT) → Holding Register,
|                 | (TBLPTR) + 1 → TBLPTR;
|                 | if TBLWT*-;
|                 | (TABLAT) → Holding Register,
|                 | (TBLPTR) – 1 → TBLPTR;
|                 | if TBLWT+;
|                 | (TBLPTR) + 1 → TBLPTR,
|                 | (TABLAT) → Holding Register |

Status Affected: None

Encoding:  

<table>
<thead>
<tr>
<th></th>
<th>0000</th>
<th>0000</th>
<th>0000</th>
<th>11nn</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nn=0</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>*+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>*-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3</td>
<td>+*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description: This instruction uses the 3 LSBs of TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to Section 5.0 “Memory Organization” for additional details on programming Flash memory.)

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR<0> = 0: Least Significant Byte of Program Memory Word

TBLPTR<0> = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1
Cycles: 2

Q Cycle Activity:

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>operation</td>
<td>operation</td>
<td>operation</td>
<td>operation</td>
</tr>
</tbody>
</table>

Words: 1
Cycles: 2

Example 1:  
TBLWT **;

Before Instruction

| TABLAT | = 55h    |
| TBLPTR | = 00A356h|
| HOLDING REGISTER (00A356h) | = FFh |

After Instructions (table write completion)

| TABLAT | = 55h    |
| TBLPTR | = 00A357h|
| HOLDING REGISTER (00A356h) | = FFh |

Example 2:  
TBLWT +*;

Before Instruction

| TABLAT | = 34h    |
| TBLPTR | = 01389Ah|
| HOLDING REGISTER (01389Ah) | = FFh |
| HOLDING REGISTER (01389Bh) | = FFh |

After Instruction (table write completion)

| TABLAT | = 34h    |
| TBLPTR | = 01389Bh|
| HOLDING REGISTER (01389Ah) | = FFh |
| HOLDING REGISTER (01389Bh) | = 34h |
TSTFSZ  Test f, Skip if 0
Syntax:  TSTFSZ f, (a)
Operands:  0 ≤ f ≤ 255
           a ∈ [0,1]
Operation:  skip if f = 0
Status Affected:  None
Encoding:  0110 011a ffff ffff
Description:  If 'f' = 0, the next instruction fetched during the current instruction execution is discarded and a NOP is executed, making this a two-cycle instruction.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.
Words: 1
Cycles: 1(2)
Note:  3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>No</td>
</tr>
<tr>
<td>register 'f'</td>
<td>Data</td>
<td>operation</td>
<td></td>
</tr>
</tbody>
</table>

If skip:
<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>operation</td>
<td>No</td>
<td>operation</td>
</tr>
</tbody>
</table>

If skip and followed by 2-word instruction:
<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>operation</td>
<td>No</td>
<td>operation</td>
</tr>
<tr>
<td>No</td>
<td>operation</td>
<td>No</td>
<td>operation</td>
</tr>
</tbody>
</table>

Example:
HERE   TSTFSZ  CNT, 1
NZERO   :
ZERO   :

Before Instruction
PC  =  Address (HERE)

After Instruction
If CNT  = 00h,
PC  =  Address (ZERO)
If CNT  = 00h,
PC  =  Address (NZERO)

XORLW  Exclusive OR Literal with W
Syntax:  XORLW  k
Operands:  0 ≤ k ≤ 255
Operation:  (W) .XOR. k → W
Status Affected:  N, Z
Encoding:  0000 1010 kkkk kkkk
Description:  The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.
Words: 1
Cycles: 1

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>Write to</td>
</tr>
<tr>
<td>literal 'k'</td>
<td>Data</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

Example:
XORLW  0AFh
Before Instruction
W  =  B5h
After Instruction
W  =  1Ah
### XORWF

**Exclusive OR W with f**

**Syntax:**

```
XORWF  f, (d, {a})
```

**Operands:**

- `0 ≤ f ≤ 255`
- `d ∈ [0, 1]`
- `a ∈ [0, 1]`

**Operation:**

```
(W).XOR. (f) → dest
```

**Status Affected:**

- `N, Z`

**Encoding:**

```
0001 10da ffff ffff
```

**Description:**

Exclusive OR the contents of W with register `f`. If `d` is ‘0’, the result is stored in W. If `d` is ‘1’, the result is stored back in the register `f` (default).

- If `a` is ‘0’, the Access Bank is selected.
- If `a` is ‘1’, the BSR is used to select the GPR bank.

If `a` is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever `f ≤ 95 (5Fh)`. See Section 25.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.

**Words:**

1

**Cycles:**

1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read</td>
<td>Process</td>
<td>Write to</td>
</tr>
<tr>
<td>register 'f'</td>
<td>Data</td>
<td>destination</td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

```
XORWF  REG, 1, 0
```

**Before Instruction**

- `REG = AFh`
- `W = B5h`

**After Instruction**

- `REG = 1Ah`
- `W = B5h`
25.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, the PIC18F87J11 Family family of devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment Indirect and Indexed Addressing operations and the implementation of Indexed Literal Offset Addressing for many of the standard PIC18 instructions.

The additional features of the extended instruction set are enabled by default on unprogrammed devices. Users must properly set or clear the XINST Configuration bit during programming to enable or disable these features.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for Indexed Addressing. Two of the instructions, ADDFSR and SUBFSR, each have an additional special instantiation for using FSR2. These versions (ADDFULN and SUBULN) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 25-3. Detailed descriptions are provided in Section 25.2.2 “Extended Instruction Set”. The opcode field descriptions in Table 25-1 (page 330) apply to both the standard and extended PIC18 instruction sets.

### Table 25-3: Extensions to the PIC18 Instruction Set

<table>
<thead>
<tr>
<th>Mnemonic, Operands</th>
<th>Description</th>
<th>Cycles</th>
<th>16-Bit Instruction Word</th>
<th>Status Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDFSR f, k</td>
<td>Add Literal to FSR</td>
<td>1</td>
<td>1110 1000 ffkk kkkk</td>
<td>None</td>
</tr>
<tr>
<td>ADDULNk f, k</td>
<td>Add Literal to FSR2 and Return</td>
<td>2</td>
<td>1110 1000 11kk kkkk</td>
<td>None</td>
</tr>
<tr>
<td>CALLW k</td>
<td>Call Subroutine using WREG</td>
<td>2</td>
<td>0000 0000 0001 0100</td>
<td>None</td>
</tr>
<tr>
<td>MOVSF z, f</td>
<td>Move zs (source) to 1st word f (destination) 2nd word</td>
<td>2</td>
<td>1110 1011 0zzz zzzz</td>
<td>None</td>
</tr>
<tr>
<td>MOVSS z, z</td>
<td>Move zₙ (source) to 1st word zₙ (destination) 2nd word</td>
<td>2</td>
<td>1110 1011 1zzz xxxx</td>
<td>None</td>
</tr>
<tr>
<td>PUSHL k</td>
<td>Store Literal at FSR2, Decrement FSR2</td>
<td>1</td>
<td>1110 1010 kkkk kkkk</td>
<td>None</td>
</tr>
<tr>
<td>SUBFSR f, k</td>
<td>Subtract Literal from FSR</td>
<td>2</td>
<td>1110 1001 ffkk kkkk</td>
<td>None</td>
</tr>
<tr>
<td>SUBULNk f, k</td>
<td>Subtract Literal from FSR2 and Return</td>
<td>2</td>
<td>1110 1001 11kk kkkk</td>
<td>None</td>
</tr>
</tbody>
</table>

Note: The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

25.2.1 Extended Instruction Syntax

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of Indexed Addressing, it is enclosed in square brackets ("[]"). This is done to indicate that the argument is used as an index or offset. The MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see Section 25.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”.

Note: In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces ("{}").
### 25.2.2 EXTENDED INSTRUCTION SET

<table>
<thead>
<tr>
<th>ADDFSR</th>
<th>Add Literal to FSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>ADDFSR ( f, k )</td>
</tr>
<tr>
<td>Operands:</td>
<td>( 0 \leq k \leq 63 )</td>
</tr>
<tr>
<td>( f \in [0, 1, 2] )</td>
<td></td>
</tr>
<tr>
<td>Operation:</td>
<td>( \text{FSR}(f) + k \rightarrow \text{FSR}(f) )</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>\begin{array}{cccc} 1110 &amp; 1000 &amp; \text{ffkk} &amp; \text{kkkk} \end{array}</td>
</tr>
<tr>
<td>Description:</td>
<td>The 6-bit literal ( 'k' ) is added to the contents of the FSR specified by ( 'f' ).</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'k'</td>
<td>Process Data</td>
<td>Write to FSR</td>
</tr>
</tbody>
</table>

**Example:** ADDFSR 2, 23h

**Before Instruction**

| FSR2 | 03FFh |

**After Instruction**

| FSR2 | 0422h |

<table>
<thead>
<tr>
<th>ADDULNK</th>
<th>Add Literal to FSR2 and Return</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>ADDULNK ( k )</td>
</tr>
<tr>
<td>Operands:</td>
<td>( 0 \leq k \leq 63 )</td>
</tr>
<tr>
<td>Operation:</td>
<td>( \text{FSR}_2 + k \rightarrow \text{FSR}_2, ) ( \text{TOS} \rightarrow \text{PC} )</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>\begin{array}{cccc} 1110 &amp; 1000 &amp; \text{11kk} &amp; \text{kkkk} \end{array}</td>
</tr>
<tr>
<td>Description:</td>
<td>The 6-bit literal ( 'k' ) is added to the contents of FSR2. A \texttt{RETURN} is then executed by loading the PC with the TOS.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>2</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read literal 'k'</td>
<td>Process Data</td>
<td>Write to FSR</td>
</tr>
<tr>
<td>No Operation</td>
<td>No Operation</td>
<td>No Operation</td>
<td>No Operation</td>
</tr>
</tbody>
</table>

**Example:** ADDULNK 23h

**Before Instruction**

| FSR2 | 03FFh |
| PC | 0100h |

**After Instruction**

| FSR2 | 0422h |
| PC | (TOS) |

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: \{label\} instruction argument(s).
CALLW Subroutine Call using WREG

Syntax: CALLW

Operands: None

Operation:
- \((PC + 2) \rightarrow TOS,\)
- \((W) \rightarrow PCL,\)
- \((PCLATH) \rightarrow PCH,\)
- \((PCLATU) \rightarrow PCU\)

Status Affected: None

Encoding:

<table>
<thead>
<tr>
<th>1st word (source)</th>
<th>2nd word (destin.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000 0001 0100</td>
<td>1110 1011 0zzz zzzz_s</td>
</tr>
<tr>
<td>fffy fffy fffy</td>
<td>fffy</td>
</tr>
</tbody>
</table>

Description: First, the return address \((PC + 2)\) is pushed onto the return stack. Next, the contents of \(W\) are written to \(PCL\); the existing value is discarded. Then, the contents of \(PCLATH\) and \(PCLATU\) are latched into \(PCH\) and \(PCU\), respectively. The second cycle is executed as a NOP instruction while the new next instruction is fetched.

Unlike CALL, there is no option to update \(W\), STATUS or BSR.

Words: 1

Cycles: 2

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read WREG</td>
<td>Push PC to stack</td>
<td>No operation</td>
</tr>
<tr>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
<td>No operation</td>
</tr>
</tbody>
</table>

Example: HERE CALLW

Before Instruction
- \(PC = \text{address (HERE)}\)
- \(PCLATH = 10h\)
- \(PCLATU = 00h\)
- \(W = 06h\)

After Instruction
- \(PC = 001006h\)
- \(TOS = \text{address (HERE + 2)}\)
- \(PCLATH = 10h\)
- \(PCLATU = 00h\)
- \(W = 06h\)

MOVSF Move Indexed to \(f\)

Syntax: MOVSF \([z_s], f_d\)

Operands:
- \(0 \leq z_s \leq 127\)
- \(0 \leq f_d \leq 4095\)

Operation:
- \(((FSR2) + z_s) \rightarrow f_d\)

Status Affected: None

Encoding:

<table>
<thead>
<tr>
<th>1st word (source)</th>
<th>2nd word (destin.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1110 1011 0zzz zzzz_s</td>
<td></td>
</tr>
<tr>
<td>fffy fffy fffy</td>
<td>fffy</td>
</tr>
</tbody>
</table>

Description: The contents of the source register are moved to destination register \(f_d\). The actual address of the source register is determined by adding the 7-bit literal offset \(z_s\), in the first word, to the value of FSR2. The address of the destination register is specified by the 12-bit literal \(f_d\) in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh).

The MOVSF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an Indirect Addressing register, the value returned will be 00h.

Words: 2

Cycles: 2

Q Cycle Activity:

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Determine source addr</td>
<td>Determine source addr</td>
<td>Read source reg</td>
</tr>
<tr>
<td>No operation</td>
<td>No dummy read</td>
<td>No operation</td>
<td>Write register ‘f’ (dest)</td>
</tr>
</tbody>
</table>

Example: MOVSF \((05h), REG2\)

Before Instruction
- \(FSR2 = 80h\)
- \(Contents of 85h = 33h\)
- \(REG2 = 11h\)

After Instruction
- \(FSR2 = 80h\)
- \(Contents of 85h = 33h\)
- \(REG2 = 33h\)
### MOVSS

**Syntax:** MOVSS [zs], [zd]

**Operands:**
- \(0 \leq zs \leq 127\)
- \(0 \leq zd \leq 127\)

**Operation:**
\[((\text{FSR2}) + zs) \rightarrow ((\text{FSR2}) + zd)\]

**Status Affected:** None

**Encoding:**
- 1st word (source): \(1110\ 1111\ 1zzz\ xzzz\)
- 2nd word (dest.): \(1111\ xxxx\ xzzz\ zzzz\)

**Description:**
The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets ‘zs’ or ‘zd’, respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).

The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an Indirect Addressing register, the value returned will be 00h. If the resultant destination address points to an Indirect Addressing register, the instruction will execute as a NOP.

**Words:** 2

**Cycles:** 2

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Determine source addr</td>
<td>Determine source addr</td>
<td>Read source reg</td>
</tr>
</tbody>
</table>

### Example:

MOVSS [05h], [06h]

**Before Instruction**
- FSR2
  - Contents of 05h = 80h
  - Contents of 06h = 11h

**After Instruction**
- FSR2
  - Contents of 05h = 80h
  - Contents of 06h = 33h

---

### PUSHL

**Syntax:** PUSHL k

**Operands:** \(0 \leq k \leq 255\)

**Operation:**
- \(k \rightarrow \text{FSR2}, \text{FSR2} - 1 \rightarrow \text{FSR2}\)

**Status Affected:** None

**Encoding:**
- \(1111\ 1010\ kkkk\ kkkk\)

**Description:**
The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation.

This instruction allows users to push values onto a software stack.

**Words:** 1

**Cycles:** 1

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read ‘k’</td>
<td>Process data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**Example:**

PUSHL 03h

**Before Instruction**
- FSR2H:FSR2L = 01ECh
- Memory (01ECh) = 00h

**After Instruction**
- FSR2H:FSR2L = 01EBh
- Memory (01ECh) = 08h
## SUBFSR

**Subtract Literal from FSR**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>SUBFSR f, k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 63</td>
</tr>
<tr>
<td>f ∈ [0, 1, 2]</td>
<td></td>
</tr>
<tr>
<td>Operation:</td>
<td>FSRf – k → FSRf</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>1110 1001 ffkk kkkk</td>
</tr>
<tr>
<td>Description:</td>
<td>The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>1</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td></td>
</tr>
</tbody>
</table>

### Example:

**Before Instruction**

FSR2 = 03FFh

**After Instruction**

FSR2 = 03DCh

## SUBULNK

**Subtract Literal from FSR2 and Return**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>SUBULNK k</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>0 ≤ k ≤ 63</td>
</tr>
<tr>
<td>Operation:</td>
<td>FSR2 – k → FSR2, (TOS) → PC</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
<tr>
<td>Encoding:</td>
<td>1110 1001 11kk kkkk</td>
</tr>
<tr>
<td>Description:</td>
<td>The 6-bit literal 'k' is subtracted from the contents of the FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a NOP is performed during the second cycle. This may be thought of as a special case of the SUBFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.</td>
</tr>
<tr>
<td>Words:</td>
<td>1</td>
</tr>
<tr>
<td>Cycles:</td>
<td>2</td>
</tr>
<tr>
<td>Q Cycle Activity:</td>
<td></td>
</tr>
</tbody>
</table>

### Example:

**Before Instruction**

FSR2 = 03FFh

PC = 0100h

**After Instruction**

FSR2 = 03DCh

PC = (TOS)
25.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

Note: Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing (Section 5.6.1 “Indexed Addressing with Literal Offset”). This has a significant impact on the way that many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: either as a location in the Access Bank (a = 0) or in a GPR bank designated by the BSR (a = 1). When the extended instruction set is enabled and a = 0, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions that use the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward-compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between C and assembly routines in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see Section 25.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”).

Although the Indexed Literal Offset mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

25.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument ‘f’ in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value ‘k’. As already noted, this occurs only when ‘f’ is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets (“[ ]”). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within the brackets, will generate an error in the MPASM Assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be ‘0’. This is in contrast to standard operation (extended instruction set disabled), when ‘a’ is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM Assembler.

The destination argument ‘d’ functions as before.

In the latest versions of the MPASM Assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, /y, or the PE directive in the source listing.

25.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18F87J11 Family family, it is very important to consider the type of code. A large, re-entrant application that is written in C and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.
### ADDWF

**ADD W to Indexed (Indexed Literal Offset mode)**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>ADDWF [k] {,d}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>(0 \leq k \leq 95) (d \in [0, 1])</td>
</tr>
<tr>
<td>Operation:</td>
<td>((W) + ((FSR2) + k) \rightarrow \text{dest})</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>N, OV, C, DC, Z</td>
</tr>
</tbody>
</table>
| Encoding: | \[
|\begin{array}{c}
0010 \quad 01d0 \quad \text{kkkk} \quad \text{kkkk}
\end{array}\] |
| Description: | The contents of W are added to the contents of the register indicated by FSR2, offset by the value ‘k’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). |
| Words: | 1 |
| Cycles: | 1 |

**Example:**

Before Instruction

<table>
<thead>
<tr>
<th>W</th>
<th>17h</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFST</td>
<td>2Ch</td>
</tr>
<tr>
<td>FSR2 Contents of 0A2Ch</td>
<td>FFh</td>
</tr>
</tbody>
</table>

After Instruction

<table>
<thead>
<tr>
<th>W</th>
<th>37h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents of 0A2Ch</td>
<td>20h</td>
</tr>
</tbody>
</table>

### BSF

**Bit Set Indexed (Indexed Literal Offset mode)**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>BSF [k], b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>(0 \leq f \leq 95) (0 \leq b \leq 7)</td>
</tr>
<tr>
<td>Operation:</td>
<td>(1 \rightarrow ((FSR2) + k))(&lt;b&gt;)</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
</tbody>
</table>
| Encoding: | \[
|\begin{array}{c}
1000 \quad \text{bbb0} \quad \text{kkkk} \quad \text{kkkk}
\end{array}\] |
| Description: | Bit ‘b’ of the register indicated by FSR2, offset by the value ‘k’, is set. |
| Words: | 1 |
| Cycles: | 1 |

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read ‘k’</td>
<td>Process Data</td>
<td>Write to destination</td>
</tr>
</tbody>
</table>

**Example:**

| BSF [FLAG_OFST], 7 |

Before Instruction

| FLAG_OFST | 0Ah |
| FSR2 Contents of 0A0Ah | 55h |

After Instruction

| Contents of 0A0Ah | D5h |

### SETF

**Set Indexed (Indexed Literal Offset mode)**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>SETF [k]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operands:</td>
<td>(0 \leq k \leq 95)</td>
</tr>
<tr>
<td>Operation:</td>
<td>FFh (\rightarrow ((FSR2) + k))</td>
</tr>
<tr>
<td>Status Affected:</td>
<td>None</td>
</tr>
</tbody>
</table>
| Encoding: | \[
|\begin{array}{c}
0110 \quad 1000 \quad \text{kkkk} \quad \text{kkkk}
\end{array}\] |
| Description: | The contents of the register indicated by FSR2, offset by ‘k’, are set to FFh. |
| Words: | 1 |
| Cycles: | 1 |

**Q Cycle Activity:**

<table>
<thead>
<tr>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decode</td>
<td>Read ‘k’</td>
<td>Process Data</td>
<td>Write to register</td>
</tr>
</tbody>
</table>

**Example:**

| SETF [OFST] |

Before Instruction

| OFST | 2Ch |
| FSR2 Contents of 0A2Ch | FFh |

After Instruction

| Contents of 0A2Ch | 0FFh |
25.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip’s software tools have been designed to fully support the extended instruction set for the PIC18F87J11 Family family. This includes the MPLAB C18 C Compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is ‘0’, disabling the extended instruction set and Indexed Literal Offset Addressing. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

• A menu option or dialog box within the environment that allows the user to configure the language tool and its settings for the project
• A command line option
• A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.
26.0 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

26.1 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.
26.2 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

26.3 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.

26.4 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

26.5 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

26.6 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.
26.7 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.

26.8 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.

26.9 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.

26.10 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.
26.11 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.

26.12 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.

26.13 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, Keeloq® 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.
27.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings (†)

Ambient temperature under bias ............................................................................................................. -40°C to +100°C
Storage temperature ...................................................................................................................................... -65°C to +150°C
Voltage on any digital only input pin or MCLR with respect to Vss (except Vdd) .............................................. -0.3V to 6.0V
Voltage on any combined digital and analog pin with respect to Vss .............................................................. -0.3V to (Vdd + 0.3V)
Voltage on VDDCORE with respect to Vss ...................................................................................................... -0.3V to 2.75V
Voltage on VDD with respect to Vss ................................................................................................................ -0.3V to 4.0V
Total power dissipation (Note 1) .................................................................................................................. 1.0W
Maximum current out of Vss pin ....................................................................................................................... 300 mA
Maximum current into Vdd pin .......................................................................................................................... 250 mA
Input clamp current, I(ik) (VI < 0 or VI > VDD) (Note 2) ........................................................................... ±0 mA
Output clamp current, IOk (VO < 0 or VO > Vdd) (Note 2) ............................................................................... ±0 mA
Maximum output current sunk by any PORTB and PORTC I/O pins .............................................................. 25 mA
Maximum output current sunk by any PORTD, PORTE and PORTJ I/O pins ................................................. 8 mA
Maximum output current sunk by any PORTA, PORTF, PORTG and PORTH I/O pins ......................... 2 mA
Maximum output current sourced by any PORTB and PORTC I/O pins ...................................................... 25 mA
Maximum output current sourced by any PORTD, PORTE and PORTJ I/O pins .................................... 8 mA
Maximum output current sourced by any PORTA, PORTF, PORTG and PORTH I/O pins ................... 2 mA
Maximum current sunk by all ports combined ............................................................................................. 200 mA
Maximum current sourced by all ports combined ......................................................................................... 200 mA

Note 1: Power dissipation is calculated as follows:

\[ P_{\text{dis}} = VDD \times (IDD - \sum IOH) + \sum \{VDD - VOH \times IOH\} + \sum (VOL \times IOL) + \sum (VTPOUT \times ITPOUT) \]

2: No clamping diodes are present.

† 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 to maximum rating conditions for extended periods may affect device reliability.
FIGURE 27-1: PIC18F87J11 FAMILY VOLTAGE-FREQUENCY GRAPH, REGULATOR ENABLED (INDUSTRIAL)

FIGURE 27-2: PIC18F87J11 FAMILY VOLTAGE-FREQUENCY GRAPH, REGULATOR DISABLED (INDUSTRIAL)(1)

Note 1: When the on-chip voltage regulator is disabled, VDD and VDDCORE must be maintained so that VDDCORE ≤ VDD ≤ 3.6V.
### 27.1 DC Characteristics: Supply Voltage
**PIC18F87J11 Family (Industrial)**

<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>D001</td>
<td>VDD</td>
<td>Supply Voltage</td>
<td></td>
<td></td>
<td>VDD</td>
<td>V</td>
<td>/envreg tied to VSS</td>
</tr>
<tr>
<td>D001B</td>
<td>VDDCORE</td>
<td>External Supply for Microcontroller Core</td>
<td>2.0</td>
<td></td>
<td>3.6</td>
<td>V</td>
<td>/envreg tied to VDD</td>
</tr>
<tr>
<td>D001C</td>
<td>AVDD</td>
<td>Analog Supply Voltage</td>
<td>VDD</td>
<td>0.3</td>
<td>VDD+0.3</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D001D</td>
<td>AVSS</td>
<td>Analog Ground Potential</td>
<td>VSS</td>
<td>0.3</td>
<td>VSS+0.3</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D002</td>
<td>VDR</td>
<td>RAM Data Retention Voltage(1)</td>
<td>1.5</td>
<td></td>
<td></td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D003</td>
<td>VPOR</td>
<td>VDD Power-on Reset Voltage</td>
<td></td>
<td></td>
<td>0.7</td>
<td>V</td>
<td>See Section 4.3 “Power-on Reset (POR)” for details</td>
</tr>
<tr>
<td>D004</td>
<td>SVDD</td>
<td>VDD Rise Rate to Ensure Internal Power-on Reset Signal</td>
<td>0.05</td>
<td></td>
<td></td>
<td>V/ms</td>
<td>See Section 4.3 “Power-on Reset (POR)” for details</td>
</tr>
<tr>
<td>D005</td>
<td>VBOR</td>
<td>Brown-out Reset Voltage</td>
<td></td>
<td>1.8</td>
<td></td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.
### 27.2 DC Characteristics: Power-Down and Supply Current

**PIC18F87J11 Family (Industrial)**

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device</th>
<th>Typ Max Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power-Down Current (IPD)(^{(1)})</td>
<td>All devices</td>
<td>0.5</td>
<td>1.4 μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td>1.4 μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5</td>
<td>10.2 μA</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.6</td>
<td>1.5 μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.6</td>
<td>1.5 μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.8</td>
<td>12.6 μA</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>2.9</td>
<td>7 μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.6</td>
<td>7 μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.6</td>
<td>19 μA</td>
</tr>
</tbody>
</table>

**Note 1:** 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 or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**Note 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

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 enabled/disabled as specified.

**Note 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**Note 4:** Voltage regulator disabled (ENVREG = 0), tied to Vss).

**Note 5:** Voltage regulator enabled (ENVREG = 1, tied to Vdd, REGSLP = 1).
## 27.2 DC Characteristics: Power-Down and Supply Current

### PIC18F87J11 Family (Industrial) (Continued)

<table>
<thead>
<tr>
<th>Supply Current (IoD)(^{(2,3)})</th>
<th>Device</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>All devices</td>
<td>5</td>
<td>14.2</td>
<td>μA</td>
<td>-40°C</td>
<td>VDD = 2.0V, VDDCORE = 2.0V(^{(4)})</td>
</tr>
<tr>
<td></td>
<td>5.5</td>
<td>14.2</td>
<td>μA</td>
<td>+25°C</td>
<td>Fosc = 31 kHz (RC_RUN mode, internal oscillator source)</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>19.0</td>
<td>μA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>6.8</td>
<td>16.5</td>
<td>μA</td>
<td>-40°C</td>
<td>VDD = 2.5V, VDDCORE = 2.5V(^{(4)})</td>
</tr>
<tr>
<td></td>
<td>7.6</td>
<td>16.5</td>
<td>μA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>22.4</td>
<td>μA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>37</td>
<td>84</td>
<td>μA</td>
<td>-40°C</td>
<td>Fosc = 1 MHz (RC_RUN mode, internal oscillator source)</td>
</tr>
<tr>
<td></td>
<td>51</td>
<td>84</td>
<td>μA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>108</td>
<td>μA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.43</td>
<td>0.82</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 2.0V, VDDCORE = 2.0V(^{(4)})</td>
</tr>
<tr>
<td></td>
<td>0.47</td>
<td>0.82</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.52</td>
<td>0.95</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.52</td>
<td>0.98</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 2.5V, VDDCORE = 2.5V(^{(4)})</td>
</tr>
<tr>
<td></td>
<td>0.57</td>
<td>0.98</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.63</td>
<td>1.10</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.59</td>
<td>0.96</td>
<td>mA</td>
<td>-40°C</td>
<td>Fosc = 4 MHz (RC_RUN mode, internal oscillator source)</td>
</tr>
<tr>
<td></td>
<td>0.65</td>
<td>0.96</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.72</td>
<td>1.18</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.88</td>
<td>1.45</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 2.0V, VDDCORE = 2.0V(^{(4)})</td>
</tr>
<tr>
<td></td>
<td>1.1</td>
<td>1.58</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>1.2</td>
<td>1.72</td>
<td>mA</td>
<td>-40°C</td>
<td>Fosc = 1 MHz (RC_RUN mode, internal oscillator source)</td>
</tr>
<tr>
<td></td>
<td>1.3</td>
<td>1.72</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>1.85</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>1.3</td>
<td>2.87</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 3.3V(^{(5)})</td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>2.87</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>2.96</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
</tbody>
</table>

### Note

1. 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 or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).
2. The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
   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 enabled/disabled as specified.
3. Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.
4. Voltage regulator disabled (ENVREG = 0, tied to VSS).
5. Voltage regulator enabled (ENVREG = 1, tied to VDD, REGSLP = 1).
## 27.2 DC Characteristics:  Power-Down and Supply Current

### PIC18F87J11 Family (Industrial) (Continued)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All devices</td>
<td>4</td>
<td>10.5</td>
<td>μA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>4.3</td>
<td>10.5</td>
<td>μA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>10.3</td>
<td>19.5</td>
<td>μA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>34</td>
<td>82</td>
<td>μA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>48</td>
<td>82</td>
<td>μA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>69</td>
<td>105</td>
<td>μA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.33</td>
<td>0.75</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.37</td>
<td>0.75</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.41</td>
<td>0.84</td>
<td>mA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.39</td>
<td>0.78</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.42</td>
<td>0.78</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.47</td>
<td>0.91</td>
<td>mA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.43</td>
<td>0.82</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.48</td>
<td>0.82</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.54</td>
<td>0.95</td>
<td>mA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.53</td>
<td>0.98</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.57</td>
<td>0.98</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.61</td>
<td>1.12</td>
<td>mA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.63</td>
<td>1.14</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.67</td>
<td>1.14</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.72</td>
<td>1.25</td>
<td>mA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.7</td>
<td>1.27</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.76</td>
<td>1.27</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.82</td>
<td>1.45</td>
<td>mA</td>
<td>+85°C</td>
</tr>
</tbody>
</table>

**Note 1:** 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 or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**Note 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

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 enabled/disabled as specified.

**Note 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**Note 4:** Voltage regulator disabled (ENVREG = 0, tied to Vss).

**Note 5:** Voltage regulator enabled (ENVREG = 1, tied to Vdd, REGSLP = 1).
27.2 DC Characteristics: Power-Down and Supply Current
PIC18F87J11 Family (Industrial) (Continued)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device</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>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supply Current (Id) Cont.2,3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.17</td>
<td>0.35</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 2.0V, VDDCORE = 2.0V(4)</td>
</tr>
<tr>
<td>All devices</td>
<td>0.18</td>
<td>0.35</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.20</td>
<td>0.42</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.29</td>
<td>0.52</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 2.5V, VDDCORE = 2.5V(4)</td>
</tr>
<tr>
<td>All devices</td>
<td>0.31</td>
<td>0.52</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.34</td>
<td>0.61</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.59</td>
<td>1.1</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 3.3V(5)</td>
</tr>
<tr>
<td>All devices</td>
<td>0.44</td>
<td>0.85</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.42</td>
<td>0.85</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.70</td>
<td>1.25</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 2.0V, VDDCORE = 2.0V(4)</td>
</tr>
<tr>
<td>All devices</td>
<td>0.75</td>
<td>1.25</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.79</td>
<td>1.36</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>1.10</td>
<td>1.7</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 2.5V, VDDCORE = 2.5V(4)</td>
</tr>
<tr>
<td>All devices</td>
<td>1.10</td>
<td>1.7</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>1.12</td>
<td>1.82</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>1.55</td>
<td>1.95</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 3.3V(5)</td>
</tr>
<tr>
<td>All devices</td>
<td>1.47</td>
<td>1.89</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>1.54</td>
<td>1.92</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>9.9</td>
<td>14.8</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 2.5V, VDDCORE = 2.5V(4)</td>
</tr>
<tr>
<td>All devices</td>
<td>9.5</td>
<td>14.8</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>10.1</td>
<td>15.2</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>13.3</td>
<td>23.2</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 3.3V(5)</td>
</tr>
<tr>
<td>All devices</td>
<td>12.2</td>
<td>22.7</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>12.1</td>
<td>22.7</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: 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 or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption. 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 enabled/disabled as specified.

3: Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

4: Voltage regulator disabled (ENVREG = 0, tied to VSS).

5: Voltage regulator enabled (ENVREG = 1, tied to VDD, REGSLP = 1).
27.2 DC Characteristics: Power-Down and Supply Current
PIC18F87J11 Family (Industrial) (Continued)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device</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>
</tr>
<tr>
<td>Supply Current (IDD) Cont(^{(2,3)})</td>
<td>All devices</td>
<td>4.5</td>
<td>5.2</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.4</td>
<td>5.2</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>4.5</td>
<td>5.2</td>
<td>mA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>5.7</td>
<td>6.7</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5</td>
<td>6.3</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.3</td>
<td>6.3</td>
<td>mA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>10.8</td>
<td>13.5</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10.8</td>
<td>13.5</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.9</td>
<td>13.0</td>
<td>mA</td>
<td>+85°C</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>13.4</td>
<td>24.1</td>
<td>mA</td>
<td>-40°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12.3</td>
<td>20.2</td>
<td>mA</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11.2</td>
<td>19.5</td>
<td>mA</td>
<td>+85°C</td>
</tr>
</tbody>
</table>

Note 1: 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 or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption. 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 enabled/disabled as specified.

3: Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

4: Voltage regulator disabled (ENVREG = 0, tied to VSS).

5: Voltage regulator enabled (ENVREG = 1, tied to VDD, REGSLP = 1).
### 27.2 DC Characteristics: Power-Down and Supply Current

#### PIC18F87J11 Family (Industrial) (Continued)

<table>
<thead>
<tr>
<th>PIC18F87J11 Family (Industrial)</th>
<th>Standard Operating Conditions (unless otherwise stated)</th>
<th>Operating temperature</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>-40°C ≤ TA ≤ +85°C for industrial</td>
<td></td>
</tr>
<tr>
<td><strong>Param No.</strong></td>
<td><strong>Device</strong></td>
<td><strong>Typ</strong></td>
<td><strong>Max</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>mA</td>
<td>mA</td>
</tr>
<tr>
<td>Supply Current (IDD) Cont.</td>
<td>All devices</td>
<td>0.10</td>
<td>0.26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.07</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.09</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.25</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.13</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.10</td>
<td>0.26</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.45</td>
<td>0.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.26</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.30</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.36</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.33</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.35</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.52</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.45</td>
<td>0.70</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.46</td>
<td>0.70</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>0.80</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.66</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.65</td>
<td>0.98</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>5.2</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.9</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.4</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>All devices</td>
<td>6.2</td>
<td>12.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.9</td>
<td>11.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.8</td>
<td>11.5</td>
</tr>
</tbody>
</table>

#### Note

1. 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 or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2. The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

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 enabled/disabled as specified.

3. Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

4. Voltage regulator disabled (ENVREG = 0, tied to VSS).

5. Voltage regulator enabled (ENVREG = 1, tied to VDD, REGSLP = 1).
## 27.2 DC Characteristics: Power-Down and Supply Current

PIC18F87J11 Family (Industrial) (Continued)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Supply Current (IDD) Cont.&lt;sup&gt;(2,3)&lt;/sup&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>18</td>
<td>35</td>
<td>µA</td>
<td>-40°C</td>
<td>VDD = 2.0V, VDDCORE = 2.0V&lt;sup&gt;(4)&lt;/sup&gt;</td>
</tr>
<tr>
<td>All devices</td>
<td>19</td>
<td>35</td>
<td>µA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>28</td>
<td>49</td>
<td>µA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>20</td>
<td>45</td>
<td>µA</td>
<td>-40°C</td>
<td>VDD = 2.5V, VDDCORE = 2.5V&lt;sup&gt;(4)&lt;/sup&gt;</td>
</tr>
<tr>
<td>All devices</td>
<td>21</td>
<td>45</td>
<td>µA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>32</td>
<td>61</td>
<td>µA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.06</td>
<td>0.11</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 3.3V&lt;sup&gt;(5)&lt;/sup&gt;</td>
</tr>
<tr>
<td>All devices</td>
<td>0.07</td>
<td>0.11</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.09</td>
<td>0.15</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>14</td>
<td>28</td>
<td>µA</td>
<td>-40°C</td>
<td>VDD = 2.0V, VDDCORE = 2.0V&lt;sup&gt;(4)&lt;/sup&gt;</td>
</tr>
<tr>
<td>All devices</td>
<td>15</td>
<td>28</td>
<td>µA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>24</td>
<td>43</td>
<td>µA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>15</td>
<td>31</td>
<td>µA</td>
<td>-40°C</td>
<td>VDD = 2.5V, VDDCORE = 2.5V&lt;sup&gt;(4)&lt;/sup&gt;</td>
</tr>
<tr>
<td>All devices</td>
<td>16</td>
<td>31</td>
<td>µA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>27</td>
<td>50</td>
<td>µA</td>
<td>+85°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.05</td>
<td>0.10</td>
<td>mA</td>
<td>-40°C</td>
<td>VDD = 3.3V&lt;sup&gt;(5)&lt;/sup&gt;</td>
</tr>
<tr>
<td>All devices</td>
<td>0.06</td>
<td>0.10</td>
<td>mA</td>
<td>+25°C</td>
<td></td>
</tr>
<tr>
<td>All devices</td>
<td>0.08</td>
<td>0.14</td>
<td>mA</td>
<td>+85°C</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** 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 or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**Note 2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

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 enabled/disabled as specified.

**Note 3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**Note 4:** Voltage regulator disabled (ENVREG = 0, tied to Vss).

**Note 5:** Voltage regulator enabled (ENVREG = 1, tied to Vdd, REGSLP = 1).
## 27.2 DC Characteristics: Power-Down and Supply Current

### PIC18F87J11 Family (Industrial) (Continued)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>D022</td>
<td>Module Differential Currents (ΔIWDT, ΔIOSCB, ΔIAD)</td>
<td>2.1</td>
<td>7.0</td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.2</td>
<td>7.0</td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.3</td>
<td>9.5</td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.0</td>
<td>8.0</td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.1</td>
<td>8.0</td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.5</td>
<td>10.4</td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5.9</td>
<td>12.1</td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.2</td>
<td>12.1</td>
<td>μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.9</td>
<td>13.6</td>
<td>μA</td>
</tr>
</tbody>
</table>

| D025 (ΔIOSCB) | Timer1 Oscillator | 14  | 24  | μA  | -40°C |
|              |                   | 15  | 24  | μA  | +25°C |
|              |                   | 23  | 36  | μA  | +85°C |
|              |                   | 17  | 26  | μA  | -40°C |
|              |                   | 18  | 26  | μA  | +25°C |
|              |                   | 25  | 38  | μA  | +85°C |
|              |                   | 19  | 35  | μA  | -40°C |
|              |                   | 21  | 35  | μA  | +25°C |
|              |                   | 28  | 44  | μA  | +85°C |

| D026 (ΔIAD)  | A/D Converter     | 3.0  | 10.0| μA  | -40°C to +85°C |
|             |                   | 3.0  | 10.0| μA  | -40°C to +85°C |
|             |                   | 3.2  | 11.0| μA  | -40°C to +85°C |

### Standard Operating Conditions (unless otherwise stated)

Operating temperature: -40°C ≤ TA ≤ +85°C for industrial

**Note**

1. 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 or VSS and all features that add delta current disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2. The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

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 enabled/disabled as specified.

3. Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

4. Voltage regulator disabled (ENVREG = 0, tied to VSS).

5. Voltage regulator enabled (ENVREG = 1, tied to VDD, REGSLP = 1).
### 27.3 DC Characteristics: PIC18F87J11 Family (Industrial)

#### DC CHARACTERISTICS

<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></td>
<td>VIL</td>
<td>Input Low Voltage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D030</td>
<td></td>
<td>All I/O Ports:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>with TTL Buffer</td>
<td>Vss</td>
<td>0.15</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>with Schmitt Trigger Buffer</td>
<td>Vss</td>
<td>0.2</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>D031</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D032</td>
<td></td>
<td>MCLR</td>
<td>Vss</td>
<td>0.2</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>D033</td>
<td></td>
<td>OSC1</td>
<td>Vss</td>
<td>0.3</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>D033A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D034</td>
<td></td>
<td>T13CKI</td>
<td>Vss</td>
<td>0.3</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VIH</td>
<td>Input High Voltage</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D040</td>
<td></td>
<td>I/O Ports with Analog Functions:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>with TTL Buffer</td>
<td>0.25</td>
<td>VDD</td>
<td>0.8</td>
<td>VDD &lt; 3.3V</td>
</tr>
<tr>
<td>D041</td>
<td></td>
<td>with Schmitt Trigger Buffer</td>
<td>0.8</td>
<td>VDD</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>D042</td>
<td></td>
<td>Digital-only I/O Ports:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D043</td>
<td></td>
<td>with TTL Buffer</td>
<td>0.25</td>
<td>5.5</td>
<td>VDD</td>
<td>3.3V ≤ VDD ≤ 3.6V</td>
</tr>
<tr>
<td>D043A</td>
<td></td>
<td>with Schmitt Trigger Buffer</td>
<td>0.8</td>
<td>5.5</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>D044</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D060</td>
<td></td>
<td>MCLR</td>
<td>0.8</td>
<td>VDD</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>D061</td>
<td></td>
<td>OSC1</td>
<td>0.7</td>
<td>VDD</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>D063</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D070</td>
<td></td>
<td>T13CKI</td>
<td>1.6</td>
<td>VDD</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>D071</td>
<td></td>
<td>PORTB Weak Pull-up Current</td>
<td>80</td>
<td>400</td>
<td>μA</td>
<td>VDD = 3.3V, VPIN = VSS</td>
</tr>
</tbody>
</table>

**Note 1:** Negative current is defined as current sourced by the pin.
## DC Characteristics: PIC18F87J11 Family (Industrial) (Continued)

### DC CHARACTERISTICS

<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>D080</td>
<td>VOL</td>
<td>Output Low Voltage</td>
<td></td>
<td></td>
<td>V</td>
<td>IOL = 2 mA, VDD = 3.3V, -40°C to +85°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I/O Ports:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTA, PORTF, PORTG, PORTH</td>
<td>—</td>
<td>0.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTD, PORTE, PORTJ</td>
<td>—</td>
<td>0.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTB, PORTC</td>
<td>—</td>
<td>0.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D083</td>
<td>VOL</td>
<td>Output Low Voltage</td>
<td></td>
<td></td>
<td>V</td>
<td>IOL = 1.6 mA, VDD = 3.3V, -40°C to +85°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I/O Ports:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTA, PORTF, PORTG, PORTH</td>
<td>—</td>
<td>0.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTD, PORTE, PORTJ</td>
<td>—</td>
<td>0.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTB, PORTC</td>
<td>—</td>
<td>0.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D090</td>
<td>VOH</td>
<td>Output High Voltage(^{(1)})</td>
<td></td>
<td></td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>I/O Ports:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTA, PORTF, PORTG, PORTH</td>
<td>2.4</td>
<td>—</td>
<td>V</td>
<td>IOH = -2 mA, VDD = 3.3V, -40°C to +85°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTD, PORTE, PORTJ</td>
<td>2.4</td>
<td>—</td>
<td>V</td>
<td>IOH = -2 mA, VDD = 3.3V, -40°C to +85°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTB, PORTC</td>
<td>2.4</td>
<td>—</td>
<td>V</td>
<td>IOH = -2 mA, VDD = 3.3V, -40°C to +85°C</td>
</tr>
<tr>
<td>D092</td>
<td>VOH</td>
<td>Output High Voltage(^{(1)})</td>
<td></td>
<td></td>
<td>V</td>
<td>IOh = -1 mA, VDD = 3.3V, -40°C to +85°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I/O Ports:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTA, PORTF, PORTG, PORTH</td>
<td>2.4</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTD, PORTE, PORTJ</td>
<td>2.4</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PORTB, PORTC</td>
<td>2.4</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>D100(^{(4)})</td>
<td>COSC2</td>
<td>Capacitive Loading Specs on Output Pins</td>
<td></td>
<td>15</td>
<td>pF</td>
<td>In HS mode when external clock is used to drive OSC1</td>
</tr>
<tr>
<td>D101</td>
<td>ClO</td>
<td>All I/O pins and OSC2</td>
<td></td>
<td>50</td>
<td>pF</td>
<td>To meet the AC Timing Specifications</td>
</tr>
<tr>
<td>D102</td>
<td>Cb</td>
<td>SCLx, SDAx</td>
<td>—</td>
<td>400</td>
<td>pF</td>
<td>(\mu)C™ Specification</td>
</tr>
</tbody>
</table>

**Note 1:** Negative current is defined as current sourced by the pin.
### TABLE 27-1: MEMORY PROGRAMMING REQUIREMENTS

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Sym</th>
<th>Characteristic</th>
<th>Min</th>
<th>Typ†</th>
<th>Max</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>D130</td>
<td>EP</td>
<td>Program Flash Memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cell Endurance</td>
<td>10K</td>
<td>—</td>
<td>—</td>
<td>E/W</td>
<td>-40°C to +85°C</td>
</tr>
<tr>
<td>D131</td>
<td>VPR</td>
<td>VDD for Read</td>
<td>VMIN</td>
<td>—</td>
<td>3.6</td>
<td>V</td>
<td>VMIN = Minimum operating voltage</td>
</tr>
<tr>
<td>D132B</td>
<td>VPEW</td>
<td>VDD for Self-Timed Write</td>
<td>VMIN</td>
<td>—</td>
<td>3.6</td>
<td>V</td>
<td>VMIN = Minimum operating voltage</td>
</tr>
<tr>
<td>D133A</td>
<td>TiW</td>
<td>Self-Timed Write Cycle Time</td>
<td>—</td>
<td>2.8</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>D134</td>
<td>TRET</td>
<td>Characteristic Retention</td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>Year</td>
<td>Provided no other specifications are violated</td>
</tr>
<tr>
<td>D135</td>
<td>IDDP</td>
<td>Supply Current during Programming</td>
<td>—</td>
<td>3</td>
<td>14</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>D1xXX</td>
<td>TWE</td>
<td>Writes per Erase Cycle</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>1</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.
### TABLE 27-2: COMPARATOR 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>D300</td>
<td>VIOFF</td>
<td>Input Offset Voltage</td>
<td>—</td>
<td>±5.0</td>
<td>±10</td>
<td>mV</td>
<td></td>
</tr>
<tr>
<td>D301</td>
<td>VICM</td>
<td>Input Common Mode Voltage*</td>
<td>0</td>
<td>—</td>
<td>AVDD – 1.5</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VRV</td>
<td>Internal Reference Voltage</td>
<td>—</td>
<td>±1.2</td>
<td>—</td>
<td>V</td>
<td>±1.2%</td>
</tr>
<tr>
<td>D302</td>
<td>CMRR</td>
<td>Common Mode Rejection Ratio*</td>
<td>55</td>
<td>—</td>
<td>—</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>TRESP</td>
<td>Response Time(1)</td>
<td>—</td>
<td>150</td>
<td>400</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>301</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>
</tbody>
</table>

* These parameters are characterized but not tested.

**Note 1:** Response time measured with one comparator input at (VDD – 1.5)/2, while the other input transitions from VSS to VDD.

**Note 2:** Tolerance is ±1.2%.

### Operating Conditions:
3.0V < VDD < 3.6V, -40°C < TA < +85°C (unless otherwise stated)

### TABLE 27-3: VOLTAGE REFERENCE 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>D310</td>
<td>VRES</td>
<td>Resolution</td>
<td>VDD/24</td>
<td>—</td>
<td>VDD/32</td>
<td>LSb</td>
<td></td>
</tr>
<tr>
<td>D311</td>
<td>VRAA</td>
<td>Absolute Accuracy</td>
<td>—</td>
<td>—</td>
<td>1/2</td>
<td>LSb</td>
<td></td>
</tr>
<tr>
<td>D312</td>
<td>VRUR</td>
<td>Unit Resistor Value (R)</td>
<td>—</td>
<td>2k</td>
<td>—</td>
<td>Ω</td>
<td></td>
</tr>
<tr>
<td>310</td>
<td>TSET</td>
<td>Settling Time(1)</td>
<td>—</td>
<td>—</td>
<td>10</td>
<td>μs</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** Settling time measured while CVRR = 1 and the CVR3:CVR0 bits transition from '0000' to '1111'.

### TABLE 27-4: INTERNAL VOLTAGE REGULATOR 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>VRGOUT</td>
<td></td>
<td>Regulator Output Voltage*</td>
<td>—</td>
<td>2.5</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>CF</td>
<td></td>
<td>External Filter Capacitor Value*</td>
<td>4.7</td>
<td>10</td>
<td>—</td>
<td>μF</td>
<td>Capacitor must be low-ESR</td>
</tr>
</tbody>
</table>

* These parameters are characterized but not tested. Parameter numbers not yet assigned for these specifications.
27.4 AC (Timing) Characteristics

27.4.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. TppS2ppS</td>
<td>3. Tcc:st</td>
<td>(i^2C specifications only)</td>
</tr>
<tr>
<td>2. TppS</td>
<td>4. Ts</td>
<td>(i^2C specifications only)</td>
</tr>
</tbody>
</table>

Lowercase letters (pp) and their meanings:

<table>
<thead>
<tr>
<th>pp</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>cc</td>
<td>CCP1</td>
<td>osc</td>
</tr>
<tr>
<td>ck</td>
<td>CLKO</td>
<td>rd</td>
</tr>
<tr>
<td>cs</td>
<td>GS</td>
<td>rw</td>
</tr>
<tr>
<td>di</td>
<td>SDI</td>
<td>sc</td>
</tr>
<tr>
<td>do</td>
<td>SDO</td>
<td>ss</td>
</tr>
<tr>
<td>dt</td>
<td>Data in</td>
<td>t0</td>
</tr>
<tr>
<td>io</td>
<td>I/O port</td>
<td>t1</td>
</tr>
<tr>
<td>mc</td>
<td>MCLR</td>
<td>wr</td>
</tr>
</tbody>
</table>

Uppercase letters and their meanings:

<table>
<thead>
<tr>
<th>S</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Fall</td>
<td>P</td>
</tr>
<tr>
<td>H</td>
<td>High</td>
<td>R</td>
</tr>
<tr>
<td>I</td>
<td>Invalid (High-impedance)</td>
<td>V</td>
</tr>
<tr>
<td>L</td>
<td>Low</td>
<td>Z</td>
</tr>
</tbody>
</table>

(i^2C only)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>output access</td>
<td>High</td>
</tr>
<tr>
<td>BUF</td>
<td>Bus free</td>
<td>Low</td>
</tr>
</tbody>
</table>

Tcc:st (i^2C specifications only)

<table>
<thead>
<tr>
<th>CC</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>HD</td>
<td>Hold</td>
<td>SU</td>
</tr>
<tr>
<td>ST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAT</td>
<td>DATA input hold</td>
<td>STO</td>
</tr>
<tr>
<td>STA</td>
<td>Start condition</td>
<td></td>
</tr>
</tbody>
</table>
27.4.2 TIMING CONDITIONS

The temperature and voltages specified in Table 27-5 apply to all timing specifications unless otherwise noted. Figure 27-3 specifies the load conditions for the timing specifications.

**TABLE 27-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

<table>
<thead>
<tr>
<th>AC CHARACTERISTICS</th>
<th>Standard Operating Conditions (unless otherwise stated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating temperature</td>
<td>(-40°C \leq T_A \leq +85°C) for industrial</td>
</tr>
<tr>
<td>Operating voltage VDD range</td>
<td>as described in Section 27.1 and Section 27.3.</td>
</tr>
</tbody>
</table>

**FIGURE 27-3: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**

- **Load Condition 1**
  - RL = 464Ω
  - CL = 50 pF for all pins except OSC2/CLKO/RA6
  - CL = 15 pF for OSC2/CLKO/RA6

- **Load Condition 2**
  - Pin
  - RL
  - CL
  - Vss
27.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 27-4: EXTERNAL CLOCK TIMING

TABLE 27-6: EXTERNAL CLOCK TIMING REQUIREMENTS

<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>1A</td>
<td>Fosc</td>
<td>External CLKI Frequency(1)</td>
<td>DC</td>
<td>48</td>
<td>MHz</td>
<td>EC Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oscillator Frequency(1)</td>
<td>DC</td>
<td>10</td>
<td>MHz</td>
<td>ECPLL Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>25</td>
<td>MHz</td>
<td>HS Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>25</td>
<td></td>
<td>HSPLL Oscillator mode</td>
</tr>
<tr>
<td>1</td>
<td>Tosc</td>
<td>External CLKI Period(1)</td>
<td>20.8</td>
<td>—</td>
<td>ns</td>
<td>EC Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oscillator Period(1)</td>
<td>100</td>
<td>—</td>
<td>ns</td>
<td>ECPLL Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>40.0</td>
<td>250</td>
<td>ns</td>
<td>HS Oscillator mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>250</td>
<td></td>
<td>HSPLL Oscillator mode</td>
</tr>
<tr>
<td>2</td>
<td>Tcy</td>
<td>Instruction Cycle Time(1)</td>
<td>83.3</td>
<td>—</td>
<td>ns</td>
<td>Tcy = 4/Fosc, Industrial</td>
</tr>
<tr>
<td>3</td>
<td>TosL,</td>
<td>High or Low Time</td>
<td>10</td>
<td>—</td>
<td>ns</td>
<td>HS Oscillator mode</td>
</tr>
<tr>
<td></td>
<td>TosH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TosR,</td>
<td>External Clock in (OSC1)</td>
<td>—</td>
<td>7.5</td>
<td>ns</td>
<td>HS Oscillator mode</td>
</tr>
<tr>
<td></td>
<td>TosF</td>
<td>Rise or Fall Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Instruction cycle period (Tcy) equals four times the input oscillator time base period for all configurations except PLL. 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/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.
### TABLE 27-7: PLL CLOCK TIMING SPECIFICATIONS (VDD = 2.15V TO 3.6V)

<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>10</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>40</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>CLKO Stability (jitter)</td>
<td>-2</td>
<td>—</td>
<td>+2</td>
<td>%</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.

### TABLE 27-8: INTERNAL RC ACCURACY (INTOSC AND INTRC SOURCES)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Device</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>INTOSC Accuracy @ Freq = 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHr, 250 kHz, 125 kHz, 31 kHz†</td>
<td>-2</td>
<td>+/-1</td>
<td>2</td>
<td>%</td>
<td>+25°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-5</td>
<td>—</td>
<td>5</td>
<td>%</td>
<td>-10°C to +85°C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-10</td>
<td>+/-1</td>
<td>10</td>
<td>%</td>
<td>-40°C to +85°C</td>
</tr>
<tr>
<td></td>
<td>INTRC Accuracy @ Freq = 31 kHz†</td>
<td>21.7</td>
<td>—</td>
<td>40.3</td>
<td>kHz</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: The accuracy specification of the 31 kHz clock is determined by which source is providing it at a given time. When INTSRC (OSCTUNE<7>) is ‘1’, use the INTOSC accuracy specification. When INTSRC is ‘0’, use the INTRC accuracy specification.
FIGURE 27-5: CLKO AND I/O TIMING

TABLE 27-9: CLKO AND I/O TIMING 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>10</td>
<td>ToSH2ckL</td>
<td>OSC1 ↑ to CLKO ↓</td>
<td>—</td>
<td>75</td>
<td>200</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>11</td>
<td>ToSH2ckH</td>
<td>OSC1 ↑ to CLKO ↑</td>
<td>—</td>
<td>75</td>
<td>200</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>12</td>
<td>TcxR</td>
<td>CLKO Rise Time</td>
<td>—</td>
<td>15</td>
<td>30</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>13</td>
<td>TcxF</td>
<td>CLKO Fall Time</td>
<td>—</td>
<td>15</td>
<td>30</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>14</td>
<td>TcxL2ioV</td>
<td>CLKO ↓ to Port Out Valid</td>
<td>—</td>
<td>—</td>
<td>0.5 TCY + 20</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>15</td>
<td>TcvO2ckH</td>
<td>Port In Valid before CLKO ↑</td>
<td>0.25 TCY + 25</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>TcxH2iO1</td>
<td>Port In Hold after CLKO ↑</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>ToSH2iOv</td>
<td>OSC1 ↑ (Q1 cycle) to Port Out Valid</td>
<td>—</td>
<td>50</td>
<td>150</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>18</td>
<td>ToSH2iO1</td>
<td>OSC1 ↑ (Q2 cycle) to Port Input Invalid</td>
<td>100</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>19</td>
<td>TcvO2osiH</td>
<td>Port Input Valid to OSC1 ↑</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>20</td>
<td>TciO</td>
<td>Port Output Rise Time</td>
<td>—</td>
<td>—</td>
<td>6</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>TciF</td>
<td>Port Output Fall Time</td>
<td>—</td>
<td>—</td>
<td>5</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>22†</td>
<td>Tinp</td>
<td>INTx pin High or Low Time</td>
<td>TCY</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>23†</td>
<td>TRBP</td>
<td>RB7:RB4 Change INTx High or Low Time</td>
<td>TCY</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in EC mode, where CLKO output is 4 x Tosc.
TABLE 27-10: CLKO AND I/O TIMING REQUIREMENTS

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Characteristics</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>TadV2aiL</td>
<td>Address Out Valid to ALE ↓ (address setup time)</td>
<td>0.25 TCY – 10</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>151</td>
<td>TalL2adL</td>
<td>ALE ↓ to Address Out Invalid (address hold time)</td>
<td>5</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>155</td>
<td>TalL2oeL</td>
<td>ALE ↑ to OE ↓ (bus release to OE)</td>
<td>10</td>
<td>0.125 TCY</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>160</td>
<td>TadZ2oeL</td>
<td>AD high-Z to OE ↓ (bus release to OE)</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>161</td>
<td>ToeH2adD</td>
<td>OE ↑ to AD Driven</td>
<td>0.125 TCY – 5</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>162</td>
<td>TadV2oeH</td>
<td>Least Significant Data Valid before OE ↑ (data setup time)</td>
<td>20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>163</td>
<td>ToeH2adL</td>
<td>OE ↑ to Data In Invalid (data hold time)</td>
<td>0</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>164</td>
<td>TalH2aiL</td>
<td>ALE Pulse Width</td>
<td>—</td>
<td>0.25 TCY</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>165</td>
<td>Toel2oeH</td>
<td>OE Pulse Width</td>
<td>0.5 TCY – 5</td>
<td>0.5 TCY</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>166</td>
<td>TalH2alH</td>
<td>ALE ↑ to ALE ↑ (cycle time)</td>
<td>—</td>
<td>TCY</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>167</td>
<td>Tacc</td>
<td>Address Valid to Data Valid</td>
<td>0.75 TCY – 25</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>168</td>
<td>Toe</td>
<td>OE ↓ to Data Valid</td>
<td>—</td>
<td>0.5 TCY – 25</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>169</td>
<td>TalL2oeH</td>
<td>ALE ↓ to OE ↑</td>
<td>0.625 TCY – 10</td>
<td>—</td>
<td>0.625 TCY + 10</td>
<td>ns</td>
</tr>
<tr>
<td>171</td>
<td>TalH2csL</td>
<td>Chip Enable Active to ALE ↓</td>
<td>0.25 TCY – 20</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>171A</td>
<td>TubL2oeH</td>
<td>AD Valid to Chip Enable Active</td>
<td>—</td>
<td>—</td>
<td>10</td>
<td>ns</td>
</tr>
</tbody>
</table>
TABLE 27-11: PROGRAM MEMORY WRITE TIMING REQUIREMENTS

<table>
<thead>
<tr>
<th>Param. No</th>
<th>Symbol</th>
<th>Characteristics</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>TadV2aIL</td>
<td>Address Out Valid to ALE ↓ (address setup time)</td>
<td>0.25 TCY</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>151</td>
<td>TalL2adl</td>
<td>ALE ↓ to Address Out Invalid (address hold time)</td>
<td>5</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>153</td>
<td>TwrH2adl</td>
<td>WRn ↑ to Data Out Invalid (data hold time)</td>
<td>5</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>154</td>
<td>TwrL</td>
<td>WRn Pulse Width</td>
<td>0.5 TCY</td>
<td>0.5 TCY</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>156</td>
<td>TadV2wrH</td>
<td>Data Valid before WRn ↑ (data setup time)</td>
<td>0.5 TCY</td>
<td>0.5 TCY</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>157</td>
<td>TbsV2wrL</td>
<td>Byte Select Valid before WRn ↓ (byte select setup time)</td>
<td>0.25 TCY</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>157A</td>
<td>TwrH2bsL</td>
<td>WRn ↑ to Byte Select Invalid (byte select hold time)</td>
<td>0.125 TCY</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>166</td>
<td>TalH2aIL</td>
<td>ALE ↑ to ALE ↑ (cycle time)</td>
<td>—</td>
<td>TCY</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>171</td>
<td>TalH2csL</td>
<td>Chip Enable Active to ALE ↓</td>
<td>0.25 TCY</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>171A</td>
<td>TubL2oeH</td>
<td>AD Valid to Chip Enable Active</td>
<td>—</td>
<td>—</td>
<td>10</td>
<td>ns</td>
</tr>
</tbody>
</table>

Operating Conditions: 2.0V < Vcc < 3.6V, -40°C < Ta < +125°C unless otherwise stated.
TABLE 27-12: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET 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>30</td>
<td>TMCCL</td>
<td>MCLR Pulse Width (low)</td>
<td>2</td>
<td>—</td>
<td>—</td>
<td>TCY</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>31</td>
<td>TWDT</td>
<td>Watchdog Timer Time-out Period (no postscaler)</td>
<td>3.4</td>
<td>4.0</td>
<td>4.6</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>TOST</td>
<td>Oscillator Start-up Timer Period</td>
<td>1024 Tosc</td>
<td>—</td>
<td>1024 Tosc</td>
<td>—</td>
<td>Tosc = OSC1 period</td>
</tr>
<tr>
<td>33</td>
<td>TPWRT</td>
<td>Power-up Timer Period</td>
<td>45.8</td>
<td>65.5</td>
<td>85.2</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>TIOZ</td>
<td>I/O High-Impedance from MCLR Low or Watchdog Timer Reset</td>
<td>—</td>
<td>2</td>
<td>—</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>TCSD</td>
<td>CPU Start-up Time</td>
<td>—</td>
<td>200</td>
<td>—</td>
<td>µs</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** To ensure device reset, MCLR must be low for at least 2 TCY or 400 µs, whichever is lower.
### TABLE 27-13: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

![Diagram of Timer0 and Timer1 External Clock Timings]

<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>40</td>
<td>T0H</td>
<td>T0CKI High Pulse Width</td>
<td>No prescaler: 0.5 TCY + 20</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With prescaler: 10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>T0L</td>
<td>T0CKI Low Pulse Width</td>
<td>No prescaler: 0.5 TCY + 20</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With prescaler: 10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>T0P</td>
<td>T0CKI Period</td>
<td>No prescaler: TCY + 10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>With prescaler: Greater of: 20 ns or (TCY + 40)/N</td>
<td>—</td>
<td>ns</td>
<td>N = prescale value (1, 2, 4, ..., 256)</td>
</tr>
</tbody>
</table>

### TABLE 27-14: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

<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>45</td>
<td>T1H</td>
<td>T13CKI High Time</td>
<td>Synchronous, no prescaler: 0.5 TCY + 20</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Synchronous, with prescaler: 10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Asynchronous: 30</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>T1L</td>
<td>T13CKI Low Time</td>
<td>Synchronous, no prescaler: 0.5 TCY + 5</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Synchronous, with prescaler: 10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Asynchronous: 30</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>T1P</td>
<td>T13CKI Input Period</td>
<td>Synchronous: Greater of: 20 ns or (TCY + 40)/N</td>
<td>—</td>
<td>ns</td>
<td>N = prescale value (1, 2, 4, 8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Asynchronous: 60</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>FT1</td>
<td>T13CKI Oscillator Input Frequency Range</td>
<td>DC</td>
<td>50 kHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>TCKE2TMR</td>
<td>Delay from External T13CKI Clock Edge to Timer Increment</td>
<td>2 Tosc</td>
<td>7 Tosc</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

*Note: Refer to Figure 27-3 for load conditions.*
FIGURE 27-9: PARALLEL SLAVE PORT TIMING

TABLE 27-15: PARALLEL SLAVE PORT REQUIREMENTS

<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>PS1</td>
<td>TdvV2wrH</td>
<td>Data In Valid before PMWR or PMCSx Inactive (setup time)</td>
<td>20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PS2</td>
<td>TdvH2dtI</td>
<td>PMWR or PMCSx Inactive to Data–In Invalid (hold time)</td>
<td>20</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PS3</td>
<td>TdlL2dtV</td>
<td>PMRD and PMCSx Active to Data–Out Valid</td>
<td></td>
<td>80</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PS4</td>
<td>TdlH2dtI</td>
<td>PMRD Active or PMCSx Inactive to Data–Out Invalid</td>
<td>10</td>
<td>30</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

Note: Refer to Figure 27-3 for load conditions.
FIGURE 27-10: PARALLEL MASTER PORT READ TIMING DIAGRAM

TABLE 27-16: PARALLEL MASTER PORT READ TIMING REQUIREMENTS

<table>
<thead>
<tr>
<th>Param. No</th>
<th>Symbol</th>
<th>Characteristics</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM1</td>
<td>PMALL/PMALH Pulse Width</td>
<td>—</td>
<td>0.5 T CY</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PM2</td>
<td>Address out valid to PMALL/PMALH Invalid (address setup time)</td>
<td>—</td>
<td>0.75 T CY</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PM3</td>
<td>PMALL/PMALH Invalid to Address Out Invalid (address hold time)</td>
<td>—</td>
<td>0.25 T CY</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PM5</td>
<td>PMRD Pulse Width</td>
<td>—</td>
<td>0.5 T CY</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PM6</td>
<td>PMRD or PMENB Active to Data In Valid (data setup time)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PM7</td>
<td>PMRD or PMENB Inactive to Data In Invalid (data hold time)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

Operating Conditions: 2.0V < Vcc < 3.6V, -40°C < Ta < +85°C unless otherwise stated.
FIGURE 27-11: PARALLEL MASTER PORT WRITE TIMING DIAGRAM

Operating Conditions: 2.0V < Vcc < 3.6V, -40°C < Tα < +85°C unless otherwise stated.

TABLE 27-17: PARALLEL MASTER PORT WRITE TIMING REQUIREMENTS

<table>
<thead>
<tr>
<th>Param. No</th>
<th>Symbol</th>
<th>Characteristics</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM11</td>
<td>PMWR Pulse Width</td>
<td>—</td>
<td></td>
<td>0.5 Tcy</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>PM12</td>
<td>Data Out Valid before PMWR or PMENB Goes Inactive (data setup time)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PM13</td>
<td>PMWR or PMENB Invalid to Data Out Invalid (data hold time)</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>PM16</td>
<td>PMCSx Pulse Width</td>
<td>Tcy – 5</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>ns</td>
</tr>
</tbody>
</table>
FIGURE 27-12: CAPTURE/COMPARE/PWM TIMINGS (INCLUDING ECCP MODULES)

![Diagram showing CAPTURE/COMPARE/PWM timings]

**Note:** Refer to Figure 27-3 for load conditions.

TABLE 27-18: CAPTURE/COMPARE/PWM REQUIREMENTS (INCLUDING ECCP MODULES)

<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>50</td>
<td>TccL</td>
<td>CCPx Input Low Time</td>
<td>0.5 TCY + 20</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No prescaler</td>
<td>0.5 TCY + 20</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With prescaler</td>
<td>10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>TccH</td>
<td>CCPx Input High Time</td>
<td>0.5 TCY + 20</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>No prescaler</td>
<td>0.5 TCY + 20</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>With prescaler</td>
<td>10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>TccP</td>
<td>CCPx Input Period</td>
<td>3 TCY + 40 N</td>
<td>—</td>
<td>ns</td>
<td>N = prescale value (1, 4 or 16)</td>
</tr>
<tr>
<td>53</td>
<td>TccR</td>
<td>CCPx Output Fall Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>TccF</td>
<td>CCPx Output Fall Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 27-13: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)

Note: Refer to Figure 27-3 for load conditions.

TABLE 27-19: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)

<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>73</td>
<td>T_{DIV2SCH}, T_{DIV2SCL}</td>
<td>Setup Time of SDIx Data Input to SCKx Edge</td>
<td>100</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>73A</td>
<td>T_{ba}</td>
<td>Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2</td>
<td>1.5 TCY + 40</td>
<td></td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>T_{DO}R</td>
<td>SDOx Data Output Rise Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>T_{DO}F</td>
<td>SDOx Data Output Fall Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>T_{SC}R</td>
<td>SCKx Output Rise Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>T_{SC}F</td>
<td>SCKx Output Fall Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>T_{CH2DO}, T_{SCCL2DOV}</td>
<td>SDOx Data Output Valid after SCKx Edge</td>
<td>—</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 27-14: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)

```
FIGURE 27-14: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)

Note: Refer to Figure 27-3 for load conditions.
```

TABLE 27-20: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)

<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>73</td>
<td>T0iV2scH, T0iV2scL</td>
<td>Setup Time of SDIx Data Input to SCKx Edge</td>
<td>100</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>TscH2diL, TscL2diL</td>
<td>Hold Time of SDIx Data Input to SCKx Edge</td>
<td>100</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>T0oR</td>
<td>SDOx Data Output Rise Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>T0oF</td>
<td>SDOx Data Output Fall Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>TscR</td>
<td>SCKx Output Rise Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>TscF</td>
<td>SCKx Output Fall Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>TscH2doV, TscL2doV</td>
<td>SDOx Data Output Valid after SCKx Edge</td>
<td>—</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>T0oV2scH, T0oV2scL</td>
<td>SDOx Data Output Setup to SCKx Edge</td>
<td>Tcy</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>
FIGURE 27-15: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)

TABLE 27-21: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)

<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>70</td>
<td>TssL2sch, TssL2scl</td>
<td>SSx ↓ to SCKx ↓ or SCKx ↑ Input</td>
<td>3 Tcy</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>70A</td>
<td>TssL2WB</td>
<td>SSx ↓ to write to SSPxBUF</td>
<td>3 Tcy</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>TsCH</td>
<td>SCKx Input High Time</td>
<td>Continuous</td>
<td>1.25 Tcy + 30</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>71A</td>
<td></td>
<td>Single byte</td>
<td>40</td>
<td>—</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>72</td>
<td>TsCL</td>
<td>SCKx Input Low Time</td>
<td>Continuous</td>
<td>1.25 Tcy + 30</td>
<td>—</td>
<td>ns</td>
</tr>
<tr>
<td>72A</td>
<td></td>
<td>Single byte</td>
<td>40</td>
<td>—</td>
<td>ns</td>
<td>(Note 1)</td>
</tr>
<tr>
<td>73</td>
<td>Td/V2sch, Td/V2scl</td>
<td>Setup Time of SDIx Data Input to SCKx Edge</td>
<td>100</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>73A</td>
<td>Ta2b</td>
<td>Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2</td>
<td>1.5 Tcy + 40</td>
<td>—</td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td>74</td>
<td>TsCH2dIL, TsCL2dIL</td>
<td>Hold Time of SDIx Data Input to SCKx Edge</td>
<td>100</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>TdoR</td>
<td>SDOx Data Output Rise Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>TdoF</td>
<td>SDOx Data Output Fall Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>TssH2doZ</td>
<td>SSx ↑ to SDOx Output High-Impedance</td>
<td>10</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>TsCH2doV, TsCL2doV</td>
<td>SDOx Data Output Valid after SCKx Edge</td>
<td>—</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>TsCH2ssH, TsCL2ssH</td>
<td>SSx ↑ after SCKx Edge</td>
<td>1.5 Tcy + 40</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Requires the use of Parameter #73A.
Note 2: Only if Parameter #71A and #72A are used.
FIGURE 27-16: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)

TABLE 27-22: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)

<table>
<thead>
<tr>
<th>Param No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>Min (ns)</th>
<th>Max (ns)</th>
<th>Units</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>TssL2sch, TssL2scl</td>
<td>SSx ↓ to SCKx ↓ or SCKx ↑ Input</td>
<td>3 TCy</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>70A</td>
<td>TssL2WB</td>
<td>SSx ↓ to write to SSPxBUF</td>
<td>3 TCy</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>Tsch</td>
<td>SCKx Input High Time</td>
<td>Continuous</td>
<td>1.25 TCy + 30</td>
<td>—</td>
<td>ns (Note 1)</td>
</tr>
<tr>
<td>71A</td>
<td></td>
<td>Single byte</td>
<td>40</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>TscL</td>
<td>SCKx Input Low Time</td>
<td>Continuous</td>
<td>1.25 TCy + 30</td>
<td>—</td>
<td>ns (Note 1)</td>
</tr>
<tr>
<td>72A</td>
<td></td>
<td>Single byte</td>
<td>40</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>73A</td>
<td>T82b</td>
<td>Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2</td>
<td>1.5 TCy + 40</td>
<td>—</td>
<td>ns (Note 2)</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>Tsch2dIL, TscL2dIL</td>
<td>Hold Time of SDIx Data Input to SCKx Edge</td>
<td>100</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>TdoR</td>
<td>SDOx Data Output Rise Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>TdoF</td>
<td>SDOx Data Output Fall Time</td>
<td>—</td>
<td>25</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>TssH2doZ</td>
<td>SSx ↑ to SDOx Output High-Impedance</td>
<td>10</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Tsch2doV, TscL2doV</td>
<td>SDOx Data Output Valid after SCKx Edge</td>
<td>—</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>TssL2doV</td>
<td>SDOx Data Output Valid after SSx ↓ Edge</td>
<td>—</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>Tsch2ssH, TscL2ssH</td>
<td>SSx ↑ after SCKx Edge</td>
<td>1.5 TCy + 40</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Requires the use of Parameter #73A.
Note 2: Only if Parameter #71A and #72A are used.
FIGURE 27-17: $I^2C$™ BUS START/STOP BITS TIMING

Note: Refer to Figure 27-3 for load conditions.

TABLE 27-23: $I^2C$™ BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

<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>90</td>
<td>TSU:STA</td>
<td>Start Condition</td>
<td>100 kHz mode</td>
<td>4700</td>
<td>— ns</td>
<td>Only relevant for Repeated Start condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setup Time</td>
<td>400 kHz mode</td>
<td>600</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>THD:STA</td>
<td>Start Condition</td>
<td>100 kHz mode</td>
<td>4000</td>
<td>— ns</td>
<td>After this period, the first clock pulse is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hold Time</td>
<td>400 kHz mode</td>
<td>600</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>TSU:STO</td>
<td>Stop Condition</td>
<td>100 kHz mode</td>
<td>4700</td>
<td>— ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setup Time</td>
<td>400 kHz mode</td>
<td>600</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>THD:STO</td>
<td>Stop Condition</td>
<td>100 kHz mode</td>
<td>4000</td>
<td>— ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hold Time</td>
<td>400 kHz mode</td>
<td>600</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

Note: Refer to Figure 27-3 for load conditions.

FIGURE 27-18: $I^2C$™ BUS DATA TIMING

Note: Refer to Figure 27-3 for load conditions.
### TABLE 27-24: I²C™ BUS DATA REQUIREMENTS (SLAVE MODE)

<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>100</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>MSSP modules</td>
<td>1.5 TCY</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>101</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>MSSP modules</td>
<td>1.5 TCY</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>102</td>
<td>Tr</td>
<td>SDAx and SCLx Rise Time</td>
<td>100 kHz mode</td>
<td>—</td>
<td>1000 ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>20 + 0.1 Ca</td>
<td>300 ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MSSP modules</td>
<td>1.5 TCY</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>103</td>
<td>Tf</td>
<td>SDAx and SCLx Fall Time</td>
<td>100 kHz mode</td>
<td>—</td>
<td>300 ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>20 + 0.1 Ca</td>
<td>300 ns</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>TSU:STA</td>
<td>Start Condition Setup 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>0.6</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>91</td>
<td>THD:STA</td>
<td>Start Condition Hold 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>106</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 μs</td>
<td></td>
</tr>
<tr>
<td>107</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>92</td>
<td>TSU:STO</td>
<td>Stop Condition Setup 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>0.6</td>
<td>—</td>
<td>μs</td>
</tr>
<tr>
<td>109</td>
<td>TAA</td>
<td>Output Valid from Clock</td>
<td>100 kHz mode</td>
<td>—</td>
<td>3500 ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>110</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>D102</td>
<td>Ca</td>
<td>Bus Capacitive Loading</td>
<td>—</td>
<td>400 pF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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 I²C™ bus device can be used in a Standard mode I²C bus system, but the requirement, TSU:DAT ≥ 250 ns, must then be met. This will automatically be the case if the device does not stretch the LOW period of the 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.
FIGURE 27-19: MSSP I²C™ BUS START/STOP BITS TIMING WAVEFORMS

TABLE 27-25: MSSP I²C™ BUS START/STOP BITS REQUIREMENTS

<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>90</td>
<td>TSU:STA</td>
<td>Start Condition</td>
<td>100 kHz mode</td>
<td>2(Tosc)(BRG + 1)</td>
<td>— ns</td>
<td>Only relevant for Repeated Start condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setup Time</td>
<td>400 kHz mode</td>
<td>2(Tosc)(BRG + 1)</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode(1)</td>
<td>2(Tosc)(BRG + 1)</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>THD:STA</td>
<td>Start Condition</td>
<td>100 kHz mode</td>
<td>2(Tosc)(BRG + 1)</td>
<td>— ns</td>
<td>After this period, the first clock pulse is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hold Time</td>
<td>400 kHz mode</td>
<td>2(Tosc)(BRG + 1)</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode(1)</td>
<td>2(Tosc)(BRG + 1)</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>TSU:STO</td>
<td>Stop Condition</td>
<td>100 kHz mode</td>
<td>2(Tosc)(BRG + 1)</td>
<td>— ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Setup Time</td>
<td>400 kHz mode</td>
<td>2(Tosc)(BRG + 1)</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode(1)</td>
<td>2(Tosc)(BRG + 1)</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>THD:STO</td>
<td>Stop Condition</td>
<td>100 kHz mode</td>
<td>2(Tosc)(BRG + 1)</td>
<td>— ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hold Time</td>
<td>400 kHz mode</td>
<td>2(Tosc)(BRG + 1)</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode(1)</td>
<td>2(Tosc)(BRG + 1)</td>
<td>—</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: Maximum pin capacitance = 10 pF for all I²C™ pins.

FIGURE 27-20: MSSP I²C™ BUS DATA TIMING

Note: Refer to Figure 27-3 for load conditions.
### TABLE 27-26: MSSP I²C™ BUS DATA REQUIREMENTS

<table>
<thead>
<tr>
<th>Param. No.</th>
<th>Symbol</th>
<th>Characteristic</th>
<th>100 kHz mode</th>
<th>400 kHz mode</th>
<th>1 MHz mode</th>
<th>Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Clock High Time</td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>THIGH</td>
<td></td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>TLOW</td>
<td></td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>Tr</td>
<td>SDAx and SCLx Rise Time</td>
<td>—</td>
<td>1000 ns</td>
<td>C B is specified to be from 10 to 400 pF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20 + 0.1 C B</td>
<td>300 ns</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>300 ns</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>Tf</td>
<td>SDAx and SCLx Fall Time</td>
<td>—</td>
<td>300 ns</td>
<td>C B is specified to be from 10 to 400 pF</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>20 + 0.1 C B</td>
<td>300 ns</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>—</td>
<td>100 ns</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>TSU:STA</td>
<td>Start Condition Setup Time</td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td>Only relevant for Repeated Start condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>THD:STA</td>
<td>Start Condition Hold Time</td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td>After this period, the first clock pulse is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>THD:DAT</td>
<td>Data Input Hold Time</td>
<td>0</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0.9 ms</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode</td>
<td>TBD</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>TSU:DAT</td>
<td>Data Input Setup Time</td>
<td>250</td>
<td>—</td>
<td>ns</td>
<td>(Note 2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode</td>
<td>TBD</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>TSU:STO</td>
<td>Stop Condition Setup Time</td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2(T Osc)(BRG + 1)</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>TAA</td>
<td>Output Valid from Clock</td>
<td>—</td>
<td>3500</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>1000</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>TBUF</td>
<td>Bus Free Time</td>
<td>4.7</td>
<td>—</td>
<td>ms</td>
<td>Time the bus must be free before a new transmission can start</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>400 kHz mode</td>
<td>1.3</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 MHz mode</td>
<td>—</td>
<td>ms</td>
<td></td>
</tr>
<tr>
<td>D102</td>
<td>Cb</td>
<td>Bus Capacitive Loading</td>
<td>—</td>
<td>400 pF</td>
<td>pF</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:** TBD = To Be Determined

**Note 1:** Maximum pin capacitance = 10 pF for all I²C™ pins.

**Note 2:** A Fast mode I²C bus device can be used in a Standard mode I²C bus system, but parameter #107 ≥ 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, parameter #102 + parameter #107 = 1000 + 250 = 1250 ns (for 100 kHz mode), before the SCLx line is released.
FIGURE 27-21: EUSART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

Note: Refer to Figure 27-3 for load conditions.

TABLE 27-27: EUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS

<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>120</td>
<td>TCKH2DTV</td>
<td>SYNC XMIT (MASTER and SLAVE) Clock High to Data Out Valid</td>
<td>—</td>
<td>40</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>121</td>
<td>TCKRF</td>
<td>Clock Out Rise Time and Fall Time (Master mode)</td>
<td>—</td>
<td>20</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>TDTREF</td>
<td>Data Out Rise Time and Fall Time</td>
<td>—</td>
<td>20</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 27-22: EUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

Note: Refer to Figure 27-3 for load conditions.

TABLE 27-28: EUSART SYNCHRONOUS RECEIVE REQUIREMENTS

<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>125</td>
<td>TDTV2CKL</td>
<td>SYNC RCV (MASTER and SLAVE) Data Hold before CKx ↓ (DTx hold time)</td>
<td>10</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
<tr>
<td>126</td>
<td>TCKL2DTL</td>
<td>Data Hold after CKx ↓ (DTx hold time)</td>
<td>15</td>
<td>—</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>
TABLE 27-29: A/D CONVERTER CHARACTERISTICS: PIC18F87J11 FAMILY (INDUSTRIAL)

<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>A01</td>
<td>NR</td>
<td>Resolution</td>
<td>—</td>
<td>—</td>
<td>10</td>
<td>bit</td>
<td>ΔVREF ≥ 3.0V</td>
</tr>
<tr>
<td>A03</td>
<td>EIL</td>
<td>Integral Linearity Error</td>
<td>—</td>
<td>—</td>
<td>&lt;±1</td>
<td>LSb</td>
<td>ΔVREF ≥ 3.0V</td>
</tr>
<tr>
<td>A04</td>
<td>EDL</td>
<td>Differential Linearity Error</td>
<td>—</td>
<td>—</td>
<td>&lt;±1</td>
<td>LSb</td>
<td>ΔVREF ≥ 3.0V</td>
</tr>
<tr>
<td>A06</td>
<td>Eoff</td>
<td>Offset Error</td>
<td>—</td>
<td>—</td>
<td>&lt;±3</td>
<td>LSb</td>
<td>ΔVREF ≥ 3.0V</td>
</tr>
<tr>
<td>A07</td>
<td>EGn</td>
<td>Gain Error</td>
<td>—</td>
<td>—</td>
<td>&lt;±3</td>
<td>LSb</td>
<td>ΔVREF ≥ 3.0V</td>
</tr>
<tr>
<td>A10</td>
<td>—</td>
<td>Monotonicity</td>
<td>—</td>
<td>—</td>
<td></td>
<td></td>
<td>Guaranteed(1)</td>
</tr>
<tr>
<td>A20</td>
<td>ΔVREF</td>
<td>Reference Voltage Range (VREFH – VREFL)</td>
<td>2.0</td>
<td>3</td>
<td>—</td>
<td>V</td>
<td>VDD &lt; 3.0V</td>
</tr>
<tr>
<td>A21</td>
<td>VREFH</td>
<td>Reference Voltage High</td>
<td>VSS</td>
<td>—</td>
<td>VREFH</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>A22</td>
<td>VREFL</td>
<td>Reference Voltage Low</td>
<td>VSS – 0.3V</td>
<td>—</td>
<td>VDD – 3.0V</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>A25</td>
<td>VAIN</td>
<td>Analog Input Voltage</td>
<td>VREFL</td>
<td>—</td>
<td>VREFH</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>A30</td>
<td>ZAIN</td>
<td>Recommended Impedance of Analog Voltage Source</td>
<td>—</td>
<td>—</td>
<td>2.5</td>
<td>kΩ</td>
<td></td>
</tr>
<tr>
<td>A50</td>
<td>IREF</td>
<td>VREF Input Current(2)</td>
<td>—</td>
<td>—</td>
<td>5</td>
<td>μA</td>
<td>During VAIN acquisition. During A/D conversion cycle.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>150</td>
<td>μA</td>
<td></td>
</tr>
</tbody>
</table>

Note 1: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

2: VREFH current is from RA3/AN3/VREF+ pin or VDD, whichever is selected as the VREFH source.

VREFL current is from RA2/AN2/VREF- pin or VSS, whichever is selected as the VREFL source.

FIGURE 27-23: A/D CONVERSION TIMING

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.

2: This is a minimal RC delay (typically 100 ns), which also disconnects the holding capacitor from the analog input.
### TABLE 27-30: A/D CONVERSION REQUIREMENTS

<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>130</td>
<td>TAD</td>
<td>A/D Clock Period</td>
<td>0.7</td>
<td>25.0(i)</td>
<td>μs</td>
<td>TOSC based, VREF ≥ 3.0V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TBD</td>
<td>1</td>
<td>μs</td>
<td>A/D RC mode</td>
</tr>
<tr>
<td>131</td>
<td>TCNV</td>
<td>Conversion Time (not including acquisition time) (Note 2)</td>
<td>11</td>
<td>12</td>
<td>TAD</td>
<td></td>
</tr>
<tr>
<td>132</td>
<td>TACQ</td>
<td>Acquisition Time (Note 3)</td>
<td>1.4</td>
<td>—</td>
<td>μs</td>
<td>-40°C to +85°C</td>
</tr>
<tr>
<td>135</td>
<td>TSWC</td>
<td>Switching Time from Convert → Sample</td>
<td>—</td>
<td>—</td>
<td>μs</td>
<td></td>
</tr>
<tr>
<td>TBD</td>
<td>TDIS</td>
<td>Discharge Time</td>
<td>0.2</td>
<td>—</td>
<td>μs</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:** TBD = To Be Determined  
**Note**  
1: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.  
2: ADRES registers may be read on the following TCY cycle.  
3: The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD). The source impedance (Rs) on the input channels is 50Ω.  
4: On the following cycle of the device clock.
28.0 PACKAGING INFORMATION

28.1 Package Marking Information

64-Lead TQFP

Example

80-Lead TQFP

Example

Legend:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX...X</td>
<td>Customer-specific information</td>
</tr>
<tr>
<td>Y</td>
<td>Year code (last digit of calendar year)</td>
</tr>
<tr>
<td>YY</td>
<td>Year code (last 2 digits of calendar year)</td>
</tr>
<tr>
<td>WW</td>
<td>Week code (week of January 1 is week '01')</td>
</tr>
<tr>
<td>NNN</td>
<td>Alphanumeric traceability code</td>
</tr>
<tr>
<td>®</td>
<td>Pb-free JEDEC designator for Matte Tin (Sn)</td>
</tr>
</tbody>
</table>

Note: This package is Pb-free. The Pb-free JEDEC designator (®) can be found on the outer packaging for this package.

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.
28.2 Package Details

The following sections give the technical details of the packages.

64-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension Limits</td>
<td>MIN</td>
</tr>
<tr>
<td>Number of Leads</td>
<td>N</td>
</tr>
<tr>
<td>Lead 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>Foot Length</td>
<td>L</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1</td>
</tr>
<tr>
<td>Foot Angle</td>
<td>φ</td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
</tr>
<tr>
<td>Molded Package Length</td>
<td>D1</td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>Lead Width</td>
<td>b</td>
</tr>
<tr>
<td>Mold Draft Angle Top</td>
<td>α</td>
</tr>
<tr>
<td>Mold Draft Angle Bottom</td>
<td>β</td>
</tr>
</tbody>
</table>

Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-085B
64-Lead Plastic Thin Quad Flatpack (PT) – 10x10x1 mm Body, 2.00 mm [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

![Recommended Land Pattern Diagram]

<table>
<thead>
<tr>
<th>Units</th>
<th>Dimension Limits</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIN</td>
<td>NOM</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
<td>0.50 BSC</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1</td>
<td>11.40</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2</td>
<td>11.40</td>
</tr>
<tr>
<td>Contact Pad Width (X64)</td>
<td>X1</td>
<td>0.30</td>
</tr>
<tr>
<td>Contact Pad Length (X64)</td>
<td>Y1</td>
<td>1.50</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>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. C04-2085A
80-Lead Plastic Thin Quad Flatpack (PT) – 12x12x1 mm Body, 2.00 mm [TQFP]

<table>
<thead>
<tr>
<th>Units</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Leads</td>
<td>N</td>
</tr>
<tr>
<td>MIN</td>
<td>NOM</td>
</tr>
<tr>
<td>Lead Pitch</td>
<td>e</td>
</tr>
<tr>
<td>0.50 BSC</td>
<td></td>
</tr>
<tr>
<td>Overall Height</td>
<td>A</td>
</tr>
<tr>
<td>–</td>
<td></td>
</tr>
<tr>
<td>–</td>
<td></td>
</tr>
<tr>
<td>1.20 BSC</td>
<td></td>
</tr>
<tr>
<td>Molded Package Thickness</td>
<td>A2</td>
</tr>
<tr>
<td>0.95</td>
<td>1.00</td>
</tr>
<tr>
<td>Standoff</td>
<td>A1</td>
</tr>
<tr>
<td>0.05</td>
<td>–</td>
</tr>
<tr>
<td>Foot Length</td>
<td>L</td>
</tr>
<tr>
<td>0.45</td>
<td>0.60</td>
</tr>
<tr>
<td>Footprint</td>
<td>L1</td>
</tr>
<tr>
<td>1.00 REF</td>
<td></td>
</tr>
<tr>
<td>Foot Angle</td>
<td>φ</td>
</tr>
<tr>
<td>0°</td>
<td></td>
</tr>
<tr>
<td>3.5°</td>
<td></td>
</tr>
<tr>
<td>7°</td>
<td></td>
</tr>
<tr>
<td>Overall Width</td>
<td>E</td>
</tr>
<tr>
<td>14.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Overall Length</td>
<td>D</td>
</tr>
<tr>
<td>14.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Molded Package Width</td>
<td>E1</td>
</tr>
<tr>
<td>12.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Molded Package Length</td>
<td>D1</td>
</tr>
<tr>
<td>12.00 BSC</td>
<td></td>
</tr>
<tr>
<td>Lead Thickness</td>
<td>c</td>
</tr>
<tr>
<td>0.09</td>
<td>–</td>
</tr>
<tr>
<td>Lead Width</td>
<td>b</td>
</tr>
<tr>
<td>0.17</td>
<td>0.22</td>
</tr>
<tr>
<td>Mold Draft Angle Top</td>
<td>α</td>
</tr>
<tr>
<td>11°</td>
<td>12°</td>
</tr>
<tr>
<td>Mold Draft Angle Bottom</td>
<td>β</td>
</tr>
<tr>
<td>11°</td>
<td>12°</td>
</tr>
</tbody>
</table>

Notes:
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.
80-Lead Plastic Thin Quad Flatpack (PT) – 12x12x1 mm Body, 2.00 mm [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

![](image)

**RECOMMENDED LAND PATTERN**

<table>
<thead>
<tr>
<th>Units</th>
<th>Dimension Limits</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIN</td>
<td>NOM</td>
</tr>
<tr>
<td>Contact Pitch</td>
<td>E</td>
<td>0.50 BSC</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C1</td>
<td>13.40</td>
</tr>
<tr>
<td>Contact Pad Spacing</td>
<td>C2</td>
<td>13.40</td>
</tr>
<tr>
<td>Contact Pad Width (X80)</td>
<td>X1</td>
<td>0.30</td>
</tr>
<tr>
<td>Contact Pad Length (X80)</td>
<td>Y1</td>
<td>1.50</td>
</tr>
<tr>
<td>Distance Between Pads</td>
<td>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. C04-2092A
APPENDIX A: REVISION HISTORY

Revision A (January 2007)
Original data sheet for the PIC18F87J11 Family of devices.

Revision B (February 2007)
Updated values in Power-Down and Supply Current table in “DC Characteristics” section.

Revision C (January 2008)
Updated text and values in several chapters and added land pattern diagrams for both packages.

APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table B-1.

TABLE B-1: DEVICE DIFFERENCES BETWEEN PIC18F87J11 FAMILY MEMBERS

<table>
<thead>
<tr>
<th>Features</th>
<th>PIC18F66J11</th>
<th>PIC18F66J16</th>
<th>PIC18F67J11</th>
<th>PIC18F86J11</th>
<th>PIC18F86J16</th>
<th>PIC18F87J11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program memory</td>
<td>64K</td>
<td>96K</td>
<td>128K</td>
<td>64K</td>
<td>96K</td>
<td>128K</td>
</tr>
<tr>
<td>Program Memory (Instructions)</td>
<td>32764</td>
<td>49148</td>
<td>65532</td>
<td>32764</td>
<td>49148</td>
<td>65532</td>
</tr>
<tr>
<td>I/O Ports</td>
<td>Ports A, B, C, D, E, F, G</td>
<td>Ports A, B, C, D, E, F, G, H, J</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMB</td>
<td>No</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-Bit ADC module</td>
<td></td>
<td></td>
<td></td>
<td>11 Input Channels</td>
<td>15 Input Channels</td>
<td></td>
</tr>
<tr>
<td>Packages</td>
<td>64-Pin TQFP</td>
<td></td>
<td></td>
<td></td>
<td>80-Pin TQFP</td>
<td></td>
</tr>
</tbody>
</table>
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: PIC18F87J11 Family
Literature Number: DS39778C

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?

__________________________________________________________________
Operation During Code-Protect ........................................ 96
Reading ................................................................. 91
Table Pointer ........................................................................ 90
Boundaries Based on Operation ........................................ 90
Table Pointer Boundaries ................................................ 90
Table Reads and Table Writes ........................................ 87
Write Sequence .......................................................... 93
Write Sequence (Word Programming) ............................ 95
Writing ....................................................................... 93
Unexpected Termination ............................................ 96
Write Verify ............................................................. 96

FSCM. See Fail-Safe Clock Monitor.

G
GOTO ........................................................................... 350

H
Hardware Multiplier ..................................................... 109
8 x 8 Multiplication Algorithms .................................... 109
Operation ..................................................................... 109
Performance Comparison (table) ............................... 109

I
I/O Ports ........................................................................ 127
Input Pull-up Configuration ........................................ 128
Open-Drain Outputs .................................................. 128
Pin Capabilities ........................................................... 127
I2C Mode (MSSP).......................................................... 253
Acknowledge Sequence Timing .................................. 260
Associated Registers ................................................... 268
Baud Rate Generator ................................................... 253
Bus Collision
  During a Repeated Start Condition .......................... 265
  During a Stop Condition ........................................... 267
Clock Arbitration ........................................................ 254
Clock Stretching .......................................................... 246
10-Bit Slave Transmit Mode ........................................... 246
10-Bit Slave Transmit Mode ........................................... 246
7-Bit Slave Receive Mode (SEN = 1) ......................... 246
7-Bit Slave Transmit Mode ........................................... 246
Clock Synchronization and the CKP bit ....................... 247
Effects of a Reset ......................................................... 261
General Call Address Support .................................... 250
I2C Clock Rate w/BRG .................................................. 253
Master Mode ................................................................ 251
  Operation .................................................................. 252
  Reception .................................................................. 257
  Repeated Start Condition Timing ............................... 256
  Start Condition Timing ............................................. 255
  Transmission .......................................................... 257
Multi-Master Communication, Bus Collision
  and Arbitration ......................................................... 261
Multi-Master Mode ....................................................... 261
  Operation ............................................................... 236
Read/Write Bit Information (R/W Bit) ....................... 236, 239
Registers ..................................................................... 231
Serial Clock (RC3/SCKx/SCLx) ..................................... 239
Slave Mode .................................................................... 236
  Address Masking Modes ........................................... 237
    5-Bit .................................................................... 237
    7-Bit .................................................................... 238
  Addressing ............................................................. 236
  Reception ............................................................... 239
  Transmission .......................................................... 239
Sleep Operation .......................................................... 261
Stop Condition Timing .................................................. 260

INCF ........................................................................... 350
INCFSZ ................................................................. 351
In-Circuit Debugger ....................................................... 327
In-Circuit Serial Programming (ICSP) ...................... 313, 327
Indexed Literal Offset Addressing
  and Standard PIC18 Instructions ............................ 376
Indexed Literal Offset Mode ....................................... 376
Indirect Addressing .................................................... 382
INFSNZ ................................................................. 351
Initialization Conditions for all Registers ................... 55–60
Instruction Cycle .......................................................... 68
  Clocking Scheme .................................................... 68
  Flow/Pipelining ..................................................... 68
Instruction Set ................................................................ 329
  ADDLW .................................................................. 335
  ADDWF .................................................................. 335
  ADDWFC (Indexed Literal Offset Mode) .................... 377
  ANDLW .................................................................. 336
  ANDWF .................................................................. 337
  BC ......................................................................... 337
  BCF ......................................................................... 338
  BN .......................................................................... 338
  BNC ........................................................................ 339
  BNN ........................................................................ 339
  BNOV ...................................................................... 340
  BNZ ........................................................................ 340
  BOV ........................................................................ 343
  BRA ......................................................................... 341
  BSF ......................................................................... 341
  BSF (Indexed Literal Offset Mode) ......................... 377
  BTFS ........................................................................ 342
  BTFS ........................................................................ 342
  BTG ......................................................................... 343
  BZ .......................................................................... 344
  CALL ....................................................................... 344
  CLRF ....................................................................... 345
  CLRWD ................................................................. 345
  COMF ....................................................................... 346
  CPFSEQ ................................................................. 346
  CPFSGT .................................................................. 347
  CPFSLT .................................................................... 347
  DAW ......................................................................... 348
  DCFSNZ ................................................................. 349
  DECF ....................................................................... 348
  DECSZ ................................................................. 349
  Extended Instructions ............................................. 371
    Considerations when Enabling ............................ 376
    Syntax ............................................................... 371
    Use with MPLAB IDE Tools ................................. 378
General Format ........................................................... 331
  GOTO ...................................................................... 350
  INC ....................................................................... 350
  INCF ...................................................................... 351
  INCFSZ .................................................................. 351
  INFSNZ ................................................................. 351
  IOR .......................................................................... 352
  IOR ......................................................................... 352
  IOR .......................................................................... 352
  IOR .......................................................................... 352
  IOR .......................................................................... 352
  IOR .......................................................................... 352
  IOR .......................................................................... 352
  IOR .......................................................................... 352
  LFSR ........................................................................ 353
  MOVF ....................................................................... 353
  MOVFF ..................................................................... 354
  MOVLB ..................................................................... 354
  MOVLF ..................................................................... 355
  MOVWF ..................................................................... 355
  MULL ........................................................................ 356
  MULL ........................................................................ 356
  MULL ........................................................................ 356

DS39778C-page 436
<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>©</td>
</tr>
<tr>
<td>34</td>
<td>INTCON Registers</td>
</tr>
<tr>
<td>35</td>
<td>Use with WDT</td>
</tr>
<tr>
<td>36</td>
<td>Internal Voltage Reference Specifications</td>
</tr>
<tr>
<td>37</td>
<td>Internet Address</td>
</tr>
<tr>
<td>38</td>
<td>Interrupt Sources</td>
</tr>
<tr>
<td>39</td>
<td>A/D Conversion Complete</td>
</tr>
<tr>
<td>40</td>
<td>Capture Complete (CCP)</td>
</tr>
<tr>
<td>41</td>
<td>Compare Complete (CCP)</td>
</tr>
<tr>
<td>42</td>
<td>Interrupt-on-Change (RB7:RB4)</td>
</tr>
<tr>
<td>43</td>
<td>TMR0 Overflow</td>
</tr>
<tr>
<td>44</td>
<td>TMR2 to PR2 Match (PWM)</td>
</tr>
<tr>
<td>45</td>
<td>TMR3 Overflow</td>
</tr>
<tr>
<td>46</td>
<td>TMR4 to PR4 Match</td>
</tr>
<tr>
<td>47</td>
<td>Intertumps</td>
</tr>
<tr>
<td>48</td>
<td>Internal Oscillator Block</td>
</tr>
<tr>
<td>49</td>
<td>INTIO Modes</td>
</tr>
<tr>
<td>50</td>
<td>INTOSC Frequency Drift</td>
</tr>
<tr>
<td>51</td>
<td>INTOSC Output Frequency</td>
</tr>
<tr>
<td>52</td>
<td>INTPLL Modes</td>
</tr>
<tr>
<td>53</td>
<td>Internal RC Block</td>
</tr>
<tr>
<td>54</td>
<td>Use with WDT</td>
</tr>
<tr>
<td>55</td>
<td>Use with WDT</td>
</tr>
<tr>
<td>56</td>
<td>Internal Voltage Reference Specifications</td>
</tr>
<tr>
<td>57</td>
<td>Internet Address</td>
</tr>
<tr>
<td>58</td>
<td>Interrupt Sources</td>
</tr>
<tr>
<td>59</td>
<td>A/D Conversion Complete</td>
</tr>
<tr>
<td>60</td>
<td>Capture Complete (CCP)</td>
</tr>
<tr>
<td>61</td>
<td>Compare Complete (CCP)</td>
</tr>
<tr>
<td>62</td>
<td>Interrupt-on-Change (RB7:RB4)</td>
</tr>
<tr>
<td>63</td>
<td>TMR0 Overflow</td>
</tr>
<tr>
<td>64</td>
<td>TMR2 to PR2 Match (PWM)</td>
</tr>
<tr>
<td>65</td>
<td>TMR3 Overflow</td>
</tr>
<tr>
<td>66</td>
<td>TMR4 to PR4 Match</td>
</tr>
<tr>
<td>67</td>
<td>Intertumps</td>
</tr>
<tr>
<td>68</td>
<td>Internal Oscillator Block</td>
</tr>
<tr>
<td>69</td>
<td>INTIO Modes</td>
</tr>
<tr>
<td>70</td>
<td>INTOSC Frequency Drift</td>
</tr>
<tr>
<td>71</td>
<td>INTOSC Output Frequency</td>
</tr>
<tr>
<td>72</td>
<td>INTPLL Modes</td>
</tr>
<tr>
<td>73</td>
<td>Internal RC Block</td>
</tr>
<tr>
<td>74</td>
<td>Use with WDT</td>
</tr>
<tr>
<td>75</td>
<td>Use with WDT</td>
</tr>
<tr>
<td>76</td>
<td>Internal Voltage Reference Specifications</td>
</tr>
<tr>
<td>77</td>
<td>Internet Address</td>
</tr>
<tr>
<td>78</td>
<td>Interrupt Sources</td>
</tr>
<tr>
<td>79</td>
<td>A/D Conversion Complete</td>
</tr>
<tr>
<td>80</td>
<td>Capture Complete (CCP)</td>
</tr>
<tr>
<td>81</td>
<td>Compare Complete (CCP)</td>
</tr>
<tr>
<td>82</td>
<td>Interrupt-on-Change (RB7:RB4)</td>
</tr>
<tr>
<td>83</td>
<td>TMR0 Overflow</td>
</tr>
<tr>
<td>84</td>
<td>TMR2 to PR2 Match (PWM)</td>
</tr>
<tr>
<td>85</td>
<td>TMR3 Overflow</td>
</tr>
<tr>
<td>86</td>
<td>TMR4 to PR4 Match</td>
</tr>
<tr>
<td>87</td>
<td>Intertumps</td>
</tr>
<tr>
<td>88</td>
<td>Internal Oscillator Block</td>
</tr>
<tr>
<td>89</td>
<td>INTIO Modes</td>
</tr>
<tr>
<td>90</td>
<td>INTOSC Frequency Drift</td>
</tr>
<tr>
<td>91</td>
<td>INTOSC Output Frequency</td>
</tr>
<tr>
<td>92</td>
<td>INTPLL Modes</td>
</tr>
<tr>
<td>93</td>
<td>Internal RC Block</td>
</tr>
</tbody>
</table>

**L**
- LFSR .................................................................................................................. 353

**M**
- Master Clear (MCLR) ......................................................................................... 51
- Master Synchronous Serial Port (MSSP). See MSSP. ........................................ 431
- Memory Organization ....................................................................................... 61
- Data Memory ....................................................................................................... 70
- Program Memory ................................................................................................. 61
- Memory Programming Requirements ................................................................. 363
- Microchip Internet Web Site ............................................................................. 380
- PIC18F87J11 FAMILY
- MPLAB ASM30 Assembler, Linker, Librarian ...................................................... 380
- MPLAB ICD 2 In-Circuit Debugger ..................................................................... 381
- MPLAB ICE 2000 High-Performance Universal In-Circuit Emulator .................... 381
- MPLAB Integrated Development
  - Environment Software ..................................................................................... 379
- MPLAB PM3 Device Programmer ........................................................................... 381
- MPLAB REAL ICE In-Circuit Emulator System ................................................... 381
- MPLINK Object Linker/MPLIB Object Librarian ................................................ 380
- MSSP
  - ACK Pulse ....................................................................................................... 236, 239
  - PIC Mode. See PIC Mode. Module Overview .................................................... 221
  - SPI Master/Slave Connection .......................................................................... 225
  - MULLW ........................................................................................................... 356
  - MULWF .......................................................................................................... 356

**N**
- NEGF ................................................................................................................... 357
- NOP ...................................................................................................................... 357

**O**
- Open-Drain Outputs ......................................................................................... 128
- Oscillator Configuration .................................................................................... 31
- EC ......................................................................................................................... 31
- ECPLL ................................................................................................................... 31
- HS ......................................................................................................................... 31
- HSPLL ................................................................................................................... 31
- Internal Oscillator Block ................................................................................... 37
- INTIO1 .................................................................................................................. 31
- INTIO2 .................................................................................................................. 31
- INTPLL1 ............................................................................................................... 31
- INTPLL2 ............................................................................................................... 31
- Oscillator Selection ............................................................................................ 313
- Oscillator Start-up Timer (OST) ........................................................................ 40
- Oscillator Switching ............................................................................................ 33
- Oscillator Transitions ......................................................................................... 34
- Oscillator, Timer1 ............................................................................................... 181, 191
- Oscillator, Timer3 ............................................................................................... 189

**P**
- Packaging ........................................................................................................... 423
- Details .................................................................................................................. 424
- Marking ............................................................................................................... 423
**Parallel Master Port (PMP)**

PIN Functions:

- AVdd ................................................................. 29
- AVdd ................................................................. 19
- AVss ................................................................. 19
- AVss ................................................................. 19
- ENVRREG ........................................................... 19
- MCLR ................................................................. 12
- OSC1/CLKI/RA7 ................................................... 12
- OSC2/CLKO/RA6 ................................................... 12
- RA0/AN0 ............................................................. 13
- RA1/AN1 ............................................................. 13
- RA2/AN2/VREF- .................................................... 13
- RA3/AN3/VREF+ ................................................... 13
- RA4/PMD5/TOCKI ................................................ 21
- RA4/T0CKI ......................................................... 13
- RA5/AN4 ............................................................. 13
- RA5/PMD4/AN4 .................................................... 21
- RA6 ................................................................. 13
- RA7 ................................................................. 13
- RB0/FLT0/INT0 ................................................... 14
- RB1/INT1/PA4 ...................................................... 14
- RB2/INT2/PA3 ...................................................... 14
- RB3/INT3/PA2/ECCP2/P2A .................................... 22
- RB3/INT3/PA2 ..................................................... 14
- RB4/KB0/PMA1 .................................................... 14
- RB5/KB1/PMA0 .................................................... 14
- RB6/KB2/PGC ..................................................... 14
- RB7/KB3/PGD ..................................................... 14
- RC0/T1SO/T13CKI ............................................... 15
- RC1/T1SO1/ECCP2/P2A ........................................ 15
- RC2/ECCP1/P1A .................................................. 15
- RC3/SCK1/SCL1 .................................................. 15
- RC4/SDI1/SDA1 .................................................. 15
- RC5/SDO1 .......................................................... 15
- RC6/TX1/CK1 ...................................................... 15
- RC7/RX1/DT1 ...................................................... 15
- RD0/AD0/PMD0 .................................................. 24
- RD0/PMD0 ........................................................ 16
- RD1/AD1/PMD1 .................................................. 24
- RD1/PMD1 .......................................................... 16
- RD2/AD2/PMD2 .................................................. 24
- RD2/PMD2 .......................................................... 16
- RD3/AD3/PMD3 .................................................. 24
- RD3/PMD3 .......................................................... 16
- RD4/AD4/PMD4/SDO2 ......................................... 24
- RD4/PMD4/SDO2 .................................................. 16
- RD5/AD5/PMD5/SDI2/SDA2 ................................... 24
- RD5/PMD5/SDI2/SDA2 ......................................... 16
- RD6/AD6/PMD6/SCK2/SCL2 ................................... 24
- RD6/PMD6/SCK2/SCL2 ........................................ 16
- RD7/AD7/PMD7/SS2 ............................................ 24
- RD7/PMD7/SS2 ................................................... 16
- RE0/AD8/PMD0/P2D ............................................ 24
- RE0/PMD0/P2D .................................................... 17
- RE1/AD9/PWR/P2C ............................................. 25
- RE1/PWR/P2C ...................................................... 17
- RE2/AD10/PMD/P2B ............................................. 25
- RE2/PMD/P2B ...................................................... 25
- RE3/AD11/PMA1/P3C/REFO ................................. 17
- RE3/PMA1/P3C/REFO .......................................... 17
- RE4/AD12/PMD/P3B ............................................. 25
- RE4/PMD/P3B ...................................................... 17
- RE5/AD13/PMA11/P1C ......................................... 25
- RE5/PMA11/P1C .................................................. 17
- RE6/AD14/PMA10/P1B ......................................... 25
- RE6/PMA10/P1B .................................................. 17
- RE7/AD15/PMA9/ECCP2/P2A ............................... 25
- RE7/PMA9/ECCP2/P2A ........................................ 17
- RF1/AN6/C2OUT .................................................. 18
- RF2/AN5/AN7/C1OUT .......................................... 18
- RF3/AN8/C2INB .................................................. 18
- RF4/AN9/C2INA .................................................. 18
- RF5/AN10/C1INB/CVREF ...................................... 18
- RF5/PMD2/AN10/C1INB/CFREF ................................ 26
- RF6/AN11/C1INA .................................................. 18
- RF6/PMD1/AN11/C1INA ....................................... 18
- RF7/PMD0/SST .................................................... 26
- RF7/SST ............................................................ 18
- RG0/PMA8/ECCP3/P3A ........................................ 19
- RG1/PMA7/TX2/CK2 ............................................ 19
- RG2/PMA6/RX2/DT2 ............................................ 19
- RG3/PMS1/CCP4/P3D ........................................... 19
- RG4/PMS2/CCP5/P1D ........................................... 19
- RH0/AN6 ........................................................... 28
- RH1/AN7 ........................................................... 28
- RH2/AN8/PMD7 .................................................. 28
- RH3/AN9/PMD6 .................................................. 28
- RH4/PMD3/AN12/P3C/C2INC ................................. 28
- RH5/PMD5/AN13/P3B/C2IN ................................. 28
- RH6/PMD6/AN14/P1C/C1INC ................................. 28
- RH7/PWR/AN15/P1B ............................................ 28
- RJ0/ALE ............................................................ 29
- RJ1/CE .............................................................. 29
- RJ2/WRL ............................................................ 29
- RJ3/WRH ............................................................ 29
- RJ4/BA0 ............................................................ 29
- RJ5/CE .............................................................. 29
- RJ6/LSB ............................................................. 29
- RJ7/UB .............................................................. 29
- Vdd ................................................................. 29
- Vdd ................................................................. 19
- Vccore/VCAP ...................................................... 19
- Vss ................................................................. 29
- Vss ................................................................. 19

Pinout I/O Descriptions

- PIC18F68J1X (64-TQFP) ........................................ 12
- PIC18F68J1X (84-TQFP) ........................................ 20

**PIR Registers**

- PIR Registers .................................................. 116

**PLL**

- HSPLL and ECPLL Oscillator Modes .......................... 36
- Use with INTOSC ................................................. 36

**POP**

- See Power-on Reset.

**PORTA**

- Associated Registers ......................................... 132
- LATA Register .................................................. 130
- PORTA Register .................................................. 130
- TRISA Register .................................................. 130
PORTB
Associated Registers ........................................... 134
LATB Register .................................................. 132
PORTB Register .............................................. 132
RB7:RB4 Interrupt-on-Change Flag
(RBIF Bit) .................................................... 132
TRISB Register ................................................ 132

PORTC
Associated Registers ........................................... 136
LATC Register .................................................. 134
PORTC Register .............................................. 134
RC3/SCKx/SCLx Pin ........................................... 239
TRISC Register ................................................ 134

PORTD
Associated Registers ........................................... 138
LATD Register .................................................. 136
PORTD Register .............................................. 136
TRISD Register ................................................ 136

PORTE
Associated Registers ........................................... 141
LATE Register .................................................. 139
PORTE Register ............................................. 139
TRISE Register ................................................ 139

PORTF
Associated Registers ........................................... 144
LATF Register .................................................. 142
PORTF Register .............................................. 142
TRISF Register ................................................ 142

PORTG
Associated Registers ........................................... 146
LATG Register .................................................. 144
PORTG Register .............................................. 144
TRISG Register ................................................ 144

PORTH
Associated Registers ........................................... 148
LATH Register .................................................. 146
PORTH Register .............................................. 146
TRISH Register ................................................ 146

PORTJ
Associated Registers ........................................... 150
LATJ Register .................................................. 149
PORTJ Register .............................................. 149
TRISJ Register ................................................ 149

Power-Managed Modes ........................................ 41
and EUSART Operation ...................................... 273
and SPI Operation .......................................... 229

Entering ................................................................ 41
Exiting Idle and Sleep Modes .............................. 47
By Interrupt .................................................... 47
By Reset ......................................................... 47
By WDT Time-out ............................................. 47
Without an Oscillator Start-up Delay ..................... 47

Idle Modes ....................................................... 45
PRI_IDLE ....................................................... 46
RC_IDLE ......................................................... 47
SEC_IDLE ....................................................... 46

Multiple Sleep Commands ................................... 42
Run Modes ...................................................... 42
PRI_RUN ......................................................... 44
RC_RUN ......................................................... 44
SEC_RUN ....................................................... 42
Selecting ....................................................... 41

Sleep Mode ..................................................... 45
OSC1 and OSC2 Pin States .................................. 40
Summary (table) ............................................... 41

Power-on Reset (POR) ......................................... 51
Power-up Delays .............................................. 40

Power-up Timer (PWRT) ..................................... 40, 52
Time-out Sequence .......................................... 52

Prescaler
Timer2 .......................................................... 209
Prescaler, Timer0 ............................................. 179

PRI_IDLE Mode ............................................... 46
PRI_RUN Mode ............................................... 42

Program Counter ............................................. 65

PCL, PCH and PCU Registers ................................. 65

Program Memory
ALU
Status ............................................................ 80

Extended Instruction Set .................................... 84
Flash Configuration Words .................................. 62
Hard Memory Vectors ........................................ 62

Instructions .................................................... 69

Two-Word ....................................................... 69

Interrupt Vector ............................................. 62

Look-up Tables ............................................... 67

Memory Maps .................................................. 61

Hard Vectors and Configuration Words .................... 62
Modes ........................................................... 64

Modes ........................................................... 63

Extended Microcontroller .................................. 63

Extended Microcontroller (Address Shifting) .......... 64

Memory Access (table) ...................................... 64

Microcontroller .............................................. 63

Reset Vector ................................................... 62

Program Verification and Code Protection .............. 327
Programming, Device Instructions ....................... 329

Pull-up Configuration ....................................... 128

Pulse-Width Modulation. See PWM (CCP Module)
and PWM (ECCP Module).

PUSH .................................................................. 358

PUSH and POP Instructions .................................. 66

PUSHL ................................................................ 374

PWM (CCP Module)
Associated Registers ........................................... 202

Duty Cycle ....................................................... 200
Example Frequencies/Resolutions ......................... 201
Operation Setup ............................................... 201
Period ............................................................ 200

PR2/PR4 Registers ............................................. 200

TMR2 (TMR4) to PR2 (PR4) Match ....................... 200

TMR2 to PR2 Match .......................................... 208

TMR4 to PR4 Match .......................................... 193

PWM (ECCP Module) .......................................... 208

CCPR1H:CCPR1L Registers ................................. 208

Direction Change in Full-Bridge Output Mode ........... 213

Duty Cycle ....................................................... 209

Effects of a Reset ............................................. 218

Enhanced PWM Auto-Shutdown ......................... 215

Example Frequencies/Resolutions ....................... 209

Full-Bridge Mode ............................................. 212

Full-Bridge Output Application Example ............... 213

Half-Bridge Mode ............................................ 211

Half-Bridge Output Mode Applications Example ....... 211

Output Configurations ....................................... 209
## PIC18F87J11 FAMILY

Output Relationships (Active-High) ........................................ 210  
Output Relationships (Active-Low) ..................................... 210  
Period ........................................................................ 208  
Programmable Dead-Band Delay ..................................... 215  
Setup for PWM Operation ............................................ 218  
Start-up Considerations .............................................. 217  

### Q

<table>
<thead>
<tr>
<th>Q Clock</th>
<th>.</th>
<th>.</th>
<th>201, 209</th>
</tr>
</thead>
</table>

### R

RAM. See Data Memory. 

| RC_IDLE Mode | . | 47 |
| RC_RUN Mode | . | 44 |
| RCON | . | 359 |

RCON Register 

| Bit Status During Initialization | . | 54 |
| Reader Response | . | 432 |
| Reference Clock Output | . | 38 |
| Register File | . | 72 |

Register File Summary ........................................ 75–79

### Registers

<table>
<thead>
<tr>
<th>Registers</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCON0 (A/D Control 0)</td>
<td>.</td>
<td>291</td>
</tr>
<tr>
<td>ADCON0 (A/D Control 1)</td>
<td>.</td>
<td>292</td>
</tr>
<tr>
<td>ANCON0 (A/D Port Configuration 2)</td>
<td>.</td>
<td>293</td>
</tr>
<tr>
<td>ANCON1 (A/D Port Configuration 1)</td>
<td>.</td>
<td>293</td>
</tr>
<tr>
<td>BAUDCON (Baud Rate Control)</td>
<td>.</td>
<td>272</td>
</tr>
<tr>
<td>CCPCON (Capture/Compare/PWM Control)</td>
<td>.</td>
<td>195</td>
</tr>
<tr>
<td>CCPxCON (ECCPx Control)</td>
<td>.</td>
<td>203</td>
</tr>
<tr>
<td>CMSTAT (Comparator Output Status)</td>
<td>.</td>
<td>303</td>
</tr>
<tr>
<td>CMxCON (Comparator Control)</td>
<td>.</td>
<td>302</td>
</tr>
<tr>
<td>CONFIG1H (Configuration 1 High)</td>
<td>.</td>
<td>315</td>
</tr>
<tr>
<td>CONFIG1L (Configuration 1 Low)</td>
<td>.</td>
<td>315</td>
</tr>
<tr>
<td>CONFIG2H (Configuration 2 High)</td>
<td>.</td>
<td>317</td>
</tr>
<tr>
<td>CONFIG3H (Configuration 3 High)</td>
<td>.</td>
<td>319</td>
</tr>
<tr>
<td>CONFIG3L (Configuration 3 Low)</td>
<td>.</td>
<td>63, 318</td>
</tr>
<tr>
<td>CVRCON (Comparator Voltage Reference Control)</td>
<td>.</td>
<td>310</td>
</tr>
<tr>
<td>DEVID1 (Device ID 1)</td>
<td>.</td>
<td>320</td>
</tr>
<tr>
<td>DEVID2 (Device ID 2)</td>
<td>.</td>
<td>320</td>
</tr>
<tr>
<td>ECCPxAS (ECCPx Auto-Shutdown Control)</td>
<td>.</td>
<td>216</td>
</tr>
<tr>
<td>ECCPxDEL (ECCPx PWM Delay)</td>
<td>.</td>
<td>216</td>
</tr>
<tr>
<td>EECON1 (EEPROM Control 1)</td>
<td>.</td>
<td>89</td>
</tr>
<tr>
<td>INTCON (Interrupt Control)</td>
<td>.</td>
<td>113</td>
</tr>
<tr>
<td>INTCON2 (Interrupt Control 2)</td>
<td>.</td>
<td>114</td>
</tr>
<tr>
<td>INTCON3 (Interrupt Control 3)</td>
<td>.</td>
<td>115</td>
</tr>
<tr>
<td>IPR1 (Peripheral Interrupt Priority 1)</td>
<td>.</td>
<td>122</td>
</tr>
<tr>
<td>IPR2 (Peripheral Interrupt Priority 2)</td>
<td>.</td>
<td>123</td>
</tr>
<tr>
<td>IPR3 (Peripheral Interrupt Priority 3)</td>
<td>.</td>
<td>124</td>
</tr>
<tr>
<td>MEMCON (External Memory Bus Control)</td>
<td>.</td>
<td>98</td>
</tr>
<tr>
<td>ODCON1 (External Open-Drain Control 1)</td>
<td>.</td>
<td>129</td>
</tr>
<tr>
<td>ODCON2 (External Open-Drain Control 2)</td>
<td>.</td>
<td>129</td>
</tr>
<tr>
<td>ODCON3 (External Open-Drain Control 3)</td>
<td>.</td>
<td>129</td>
</tr>
<tr>
<td>OSCCON (Oscillator Control)</td>
<td>.</td>
<td>32</td>
</tr>
<tr>
<td>OSCPSC (Oscillator Power Control)</td>
<td>.</td>
<td>32</td>
</tr>
<tr>
<td>PADC (Pad Configuration Control)</td>
<td>.</td>
<td>130</td>
</tr>
<tr>
<td>PIE1 (Peripheral Interrupt Enable 1)</td>
<td>.</td>
<td>119</td>
</tr>
<tr>
<td>PIE2 (Peripheral Interrupt Enable 2)</td>
<td>.</td>
<td>120</td>
</tr>
<tr>
<td>PIE3 (Peripheral Interrupt Enable 3)</td>
<td>.</td>
<td>121</td>
</tr>
<tr>
<td>PIR1 (Peripheral Interrupt Request (Flag 1))</td>
<td>.</td>
<td>116</td>
</tr>
<tr>
<td>PIR2 (Peripheral Interrupt Request (Flag 2))</td>
<td>.</td>
<td>117</td>
</tr>
<tr>
<td>PIR3 (Peripheral Interrupt Request (Flag 3))</td>
<td>.</td>
<td>118</td>
</tr>
<tr>
<td>PMADD (Parallel Port Address)</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>PMADDH (Parallel Port Control High Byte)</td>
<td>.</td>
<td>152</td>
</tr>
<tr>
<td>PMADDL (Parallel Port Control Low Byte)</td>
<td>.</td>
<td>153</td>
</tr>
<tr>
<td>PMCONH (Parallel Port Control High Byte)</td>
<td>.</td>
<td>152</td>
</tr>
<tr>
<td>PMCONL (Parallel Port Control Low Byte)</td>
<td>.</td>
<td>153</td>
</tr>
<tr>
<td>PMECON (Parallel Port Enable Control)</td>
<td>.</td>
<td>155</td>
</tr>
<tr>
<td>PMEL (Parallel Port Enable Low Byte)</td>
<td>.</td>
<td>156</td>
</tr>
<tr>
<td>PMMODEH (Parallel Port Mode High Byte)</td>
<td>.</td>
<td>154</td>
</tr>
<tr>
<td>PMMODEL (Parallel Port Mode Low Byte)</td>
<td>.</td>
<td>155</td>
</tr>
<tr>
<td>PMSTAT (Parallel Port Status High Byte)</td>
<td>.</td>
<td>156</td>
</tr>
<tr>
<td>PMSTAT (Parallel Port Status Low Byte)</td>
<td>.</td>
<td>157</td>
</tr>
<tr>
<td>RCON (Reset Control)</td>
<td>.</td>
<td>50, 125</td>
</tr>
<tr>
<td>RCON (Receive Status)</td>
<td>.</td>
<td>271</td>
</tr>
<tr>
<td>REFCON (Reference Control)</td>
<td>.</td>
<td>39</td>
</tr>
<tr>
<td>SCON (Serial Port Control)</td>
<td>.</td>
<td>89</td>
</tr>
<tr>
<td>SCCON (Serial Communications Control)</td>
<td>.</td>
<td>89</td>
</tr>
<tr>
<td>SFRCON (System File Register Control)</td>
<td>.</td>
<td>89</td>
</tr>
<tr>
<td>SPPCON (SPI Control)</td>
<td>.</td>
<td>223</td>
</tr>
<tr>
<td>SPS (Serial Peripheral Interface)</td>
<td>.</td>
<td>221</td>
</tr>
<tr>
<td>SPPST (SPI Status)</td>
<td>.</td>
<td>223</td>
</tr>
<tr>
<td>SPPST (SPI Status)</td>
<td>.</td>
<td>223</td>
</tr>
<tr>
<td>SPPST (SPI Status)</td>
<td>.</td>
<td>223</td>
</tr>
<tr>
<td>SPPST (SPI Status)</td>
<td>.</td>
<td>223</td>
</tr>
<tr>
<td>STATUS</td>
<td>.</td>
<td>80</td>
</tr>
<tr>
<td>STKPTR (Stack Pointer)</td>
<td>.</td>
<td>66</td>
</tr>
<tr>
<td>T0CON (Timer0 Control)</td>
<td>.</td>
<td>177</td>
</tr>
<tr>
<td>T1CON (Timer1 Control)</td>
<td>.</td>
<td>181</td>
</tr>
<tr>
<td>T2CON (Timer2 Control)</td>
<td>.</td>
<td>187</td>
</tr>
<tr>
<td>T3CON (Timer3 Control)</td>
<td>.</td>
<td>189</td>
</tr>
<tr>
<td>T4CON (Timer4 Control)</td>
<td>.</td>
<td>193</td>
</tr>
<tr>
<td>TXSTAx (Transmit Status and Control)</td>
<td>.</td>
<td>270</td>
</tr>
<tr>
<td>WDT (Watchdog Timer Control)</td>
<td>.</td>
<td>74, 322</td>
</tr>
</tbody>
</table>

#### RESET

| Reset | . | 49 |
| Brown-out Reset (BOR) | . | 49 |
| Configuration Mismatch (CM) | . | 49 |
| MCLR Reset, During Power-Managed Modes | . | 49 |
| MCLR Reset, Normal Operation | . | 49 |
| Power-on Reset (POR) | . | 49 |
| RESET Instruction | . | 49 |
| Stack Full Reset | . | 49 |
| Stack Underflow Reset | . | 49 |
| Watchdog Timer (WDT) Reset | . | 49 |

#### Resets

| Brown-out Reset (BOR) | . | 313 |
| Oscillator Start-up Timer (OST) | . | 313 |
| Power-on Reset (POR) | . | 313 |
| Power-up Timer (PWRT) | . | 313 |
| RETFIE | . | 360 |
| RETLW | . | 360 |
| RETURN | . | 361 |
| Revision History | . | 429 |
| RLCF | . | 361 |
| RLNC | . | 362 |
| RRCF | . | 362 |
| RRCF | . | 363 |

#### S

| SCKx | . | 221 |
| SDx | . | 221 |
| SDOx | . | 221 |
| SEC_IDLE Mode | . | 46 |
| SEC_RUN Mode | . | 42 |
| Serial Clock, SCKx | . | 221 |
| Serial Data In (SDx) | . | 221 |
| Serial Data Out (SDOx) | . | 221 |
| Serial Peripheral Interface. See SPI Mode. | . | 363 |
| SETF | . | 363 |
| Slave Select (SSx) | . | 221 |
| SLEEP | . | 364 |
| Software Simulator (MPLAB SIM) | . | 380 |

Special Event Trigger. See Compare (ECCP Module).
Special Features of the CPU ........................................................................................................... 313
Special Function Registers ............................................................................................................. 74
SPI Mode (MSSP) ........................................................................................................................... 221
Associated Registers ....................................................................................................................... 230
Bus Mode Compatibility .................................................................................................................. 229
Clock Speed, Interactions ................................................................................................................ 229
Effects of a Reset ............................................................................................................................. 229
Enabling SPI I/O ............................................................................................................................... 225
Master Mode .................................................................................................................................... 226
Master/Slave Connection .................................................................................................................. 225
Operation .......................................................................................................................................... 224
Operation in Power-Managed Modes .............................................................................................. 229
Serial Clock ..................................................................................................................................... 221
Serial Data In ................................................................................................................................... 221
Serial Data Out ................................................................................................................................ 221
Slave Mode ........................................................................................................................................ 227
Slave Select ....................................................................................................................................... 221
Slave Select Synchronization ........................................................................................................... 227
SPI Clock .......................................................................................................................................... 226
SSPxBUF Register ............................................................................................................................ 226
SSPxSR Register ............................................................................................................................... 226
Typical Connection ........................................................................................................................... 225
SSPOV ............................................................................................................................................. 257
SSPOV Status Flag ............................................................................................................................ 257
SSPxSTAT Register ........................................................................................................................... 226
R/W Bit ............................................................................................................................................. 236, 239
SSx .................................................................................................................................................. 221
Stack Full/Underflow Resets ........................................................................................................... 67
SUBFSR .......................................................................................................................................... 375
SUBFWB .......................................................................................................................................... 364
SUBLW ............................................................................................................................................ 365
SUBULNK ........................................................................................................................................ 375
SUBWF ............................................................................................................................................ 365
SUBWFB .......................................................................................................................................... 366
SWAPF .............................................................................................................................................. 366
T
Table Pointer Operations (table) ...................................................................................................... 90
Table Reads/Table Writes .................................................................................................................. 67
TBLRD .............................................................................................................................................. 367
TBLWT .............................................................................................................................................. 368
Timer0 ............................................................................................................................................... 177
Associated Registers ....................................................................................................................... 179
Operation .......................................................................................................................................... 178
Overflow Interrupt ........................................................................................................................... 179
Prescaler .......................................................................................................................................... 179
Switching Assignment .................................................................................................................... 179
Prescaler Assignment (PSC Bit) .......................................................................................................... 179
Prescaler Select (T0PS2:T0PS0 Bits) ................................................................................................. 179
Prescaler. See Prescaler, Timer0. ...................................................................................................... 179
Reads and Writes in 16-Bit Mode ..................................................................................................... 178
Source Edge Select (T0SE Bit) .......................................................................................................... 178
Source Select (T0CS Bit) ................................................................................................................... 178
Timer1 ............................................................................................................................................... 181
16-Bit Read/Write Mode .................................................................................................................... 183
Associated Registers ....................................................................................................................... 186
Considerations in Asynchronous Counter Mode ............................................................................... 185
Interrupt .......................................................................................................................................... 184
Operation .......................................................................................................................................... 182
Oscillator .......................................................................................................................................... 181, 183
Layout Considerations .................................................................................................................... 183
Oscillator, as Secondary Clock ....................................................................................................... 33
Resetting, Using the ECCPx
Special Event Trigger ...................................................................................................................... 184
Special Event Trigger (ECCP) .......................................................................................................... 207
TMR1H Register ............................................................................................................................... 181
TMR1L Register ............................................................................................................................... 181
Use as a Clock Source ...................................................................................................................... 183
Use as a Real-Time Clock ................................................................................................................ 184
Timer2 ............................................................................................................................................... 187
Associated Registers ....................................................................................................................... 188
Interrupt .......................................................................................................................................... 188
Operation .......................................................................................................................................... 187
Output ............................................................................................................................................. 188
PR2 Register ................................................................................................................................... 208
TMR2 to PR2 Match Interrupt .......................................................................................................... 208
Timer3 ............................................................................................................................................... 189
16-Bit Read/Write Mode .................................................................................................................... 191
Associated Registers ....................................................................................................................... 191
Operation .......................................................................................................................................... 190
Oscillator .......................................................................................................................................... 189, 191
Overflow Interrupt .......................................................................................................................... 189, 191
Special Event Trigger (ECCPx) ........................................................................................................ 191
TMR4H Register ............................................................................................................................... 189
TMR4L Register ................................................................................................................................ 189
Timer4 ............................................................................................................................................... 193
Associated Registers ....................................................................................................................... 194
Operation .......................................................................................................................................... 193
Output ............................................................................................................................................. 194
Postscaler. See Postscaler, Timer4.
PR4 Register .................................................................................................................................... 193
Prescaler. See Prescaler, Timer4.
TMR4 Register ................................................................................................................................ 193
TMR4 to PR4 Match Interrupt .......................................................................................................... 193, 194
Timing Diagrams
A/D Conversion ............................................................................................................................... 420
Asynchronous Reception .................................................................................................................. 282
Asynchronous Transmission ............................................................................................................ 280
Asynchronous Transmission (Back to Back) .................................................................................. 280
Automatic Baud Rate Calculation ................................................................................................. 278
Auto-Wake-up Bit (WUE) During Normal Operation ..................................................................... 283
Auto-Wake-up Bit (WUE) During Sleep ......................................................................................... 283
Baud Rate Generator with Clock Arbitration .................................................................................. 254
BRG Overflow Sequence ................................................................................................................ 278
BRG Reset Due to SD Ax Arbitration During Start Condition ....................................................... 264
Bus Collision During a Repeated Start Condition (Case 1) ............................................................ 265
Bus Collision During a Repeated Start Condition (Case 2) ............................................................ 266
Bus Collision During a Start Condition (SCx = 0) ........................................................................ 264
Bus Collision During a Stop Condition (Case 1) .......................................................................... 267
Bus Collision During a Stop Condition (Case 2) .......................................................................... 267
Bus Collision During Start Condition (SDAx Only) ...................................................................... 263
Bus Collision for Transmit and Acknowledge .............................................................................. 262
Capture/Compare/PWM (Including ECCP Modules) .................................................................... 410
CLKO and I/O ................................................................................................................................. 402
Clock Synchronization ................................................................................................................... 247
Clock/Instruction Cycle .................................................................................................................... 68

© 2008 Microchip Technology Inc.

Preliminary

DS39778C-page 441

PIC18F87J11 FAMILY
PIC18F87J11 FAMILY

- EUSART Synchronous Receive
  - (Master/Slave) ............................................. 419
- EUSART Synchronous Transmission
  - (Master/Slave) ............................................. 419
- Example SPI Master Mode (CKE = 0) ............... 411
- Example SPI Master Mode (CKE = 1) ............... 412
- Example SPI Slave Mode (CKE = 0) ............... 413
- Example SPI Slave Mode (CKE = 1) ............... 414
- External Clock (All Modes Except PLL) .......... 400
- External Memory Bus for Sleep (Extended
  - Microcontroller Mode) .................. 104, 106
- External Memory Bus for TBLRD (Extended
  - Microcontroller Mode) .................. 104, 106
- Fail-Safe Clock Monitor .................. 326
- First Start Bit Timing .................. 255
- Full-Bridge PWM Output .................. 212
- Half-Bridge PWM Output .................. 211
- I2C Acknowledge Sequence ............... 260
- I2C Bus Data .............................................. 415
- I2C Bus Start/Stop Bits ............... 415
- I2C Master Mode (7 or 10-Bit Transmission) .... 258
- I2C Slave Mode (7-Bit Reception) .......... 244
- I2C Slave Mode (7-Bit Reception, SEN = 0) .... 244
- I2C Slave Mode (7-Bit Transmission) .......... 244
- I2C Slave Mode General Call Address Sequence . 250
- I2C Stop Condition Receive or Transmit Mode .. 261
- MSSP I2C Bus Data ....................... 417
- MSSP I2C Bus Start/Stop Bits ........... 417
- Parallel Master Port Read .................. 408
- Parallel Master Port Write .................. 409
- Parallel Slave Port ....................... 407
- Parallel Slave Port Read ............... 160, 163
- Parallel Slave Port Write ............... 160, 163
- Program Memory Read .................... 403
- Program Memory Write .................... 404
- PWM Auto-Shutdown (P1RSEN = 0,
  - Auto-Restart Disabled) .................. 217
- PWM Auto-Shutdown (P1RSEN = 1,
  - Auto-Restart Enabled) .................. 217
- PWM Direction Change .................... 214
- PWM Direction Change at Near
  - 100% Duty Cycle ....................... 214
- PWM Output .............................................. 200
- Read and Write, 8-Bit Data,
  - Demultiplexed Address .................. 167
- Read, 16-Bit Data, Demultiplexed Address .... 170
- Read, 16-Bit Multiplexed Data, Fully
  - Multiplexed 16-Bit Address ............ 172
- Read, 16-Bit Multiplexed Data, Partially
  - Multiplexed Address .................. 171
- Read, 8-Bit Data, Fully Multiplexed
  - 16-Bit Address ....................... 169
- Read, 8-Bit Data, Partially Multiplexed Address .... 167
- Read, 8-Bit Data, Partially Multiplexed
  - Address, Enable Strobe ............... 169
- Read, 8-Bit Data, Wait States Enabled,
  - Partially Multiplexed Address .......... 168
- Repeated Start Condition .................. 256
- Reset, Watchdog Timer (WDT), Oscillator Start-up
  - Timer (OST) and Power-up Timer (PWRT) .... 405
- Send Break Character Sequence ............. 284
- Slave Synchronization .................... 227
- Slow Rise Time (MCLR Tied to VDD, Vdd Rise > TPWRT) .... 53
- SPI Mode (Master Mode) .................... 226
- SPI Mode (Slave Mode, CKE = 0) ........... 228
- SPI Mode (Slave Mode, CKE = 1) ........... 228
- Synchronous Reception (Master Mode, SREN = 0) .... 287
- Synchronous Transmission .................. 285
- Synchronous Transmission (Through TXEN) ........ 286
- Time-out Sequence on Power-up
  - (MCLR Not Tied to VDD), Case 1 .......... 52
  - (MCLR Not Tied to VDD), Case 2 .......... 52
- Transition From SEC_RUN Mode to
  - PRI_RUN Mode .................................. 44
- Transition From RC_RUN Mode to
  - PRI_RUN Mode (HSPLL) .......... 324
- Transition From SEC_RUN Mode to
  - PRI_RUN Mode (HSPLL) .......... 43
- Transition to RC_RUN Mode ............... 44
- Write, 16-Bit Multiplexed Data, Fully
  - Multiplexed 16-Bit Address ............ 172
- Write, 16-Bit Multiplexed Data, Partially
  - Multiplexed Address .................. 171
- Write, 8-Bit Data, Demultiplexed Address .... 170
- Write, 8-Bit Data, Fully Multiplexed
  - 16-Bit Address ....................... 170
- Write, 8-Bit Data, Partially Multiplexed
  - Address ........................................ 168
- Write, 8-Bit Data, Partially Multiplexed
  - Address, Enable Strobe .................. 169
- Write, 8-Bit Data, Wait States Enabled,
  - Partially Multiplexed Address .......... 168
- Timing Diagrams and Specifications
  - Capture/Compare/PWM Requirements
    - (Including ECCP Modules) .............. 410
  - CLKO and I/O Requirements .............. 402, 403
  - EUSART Synchronous Receive
    - Requirements .................................. 419
- EUSART Synchronous Transmission
  - Requirements .................................. 419
- Example SPI Mode Requirements
  - (Master Mode, CKE = 0) ............... 411
- Example SPI Mode Requirements
  - (Master Mode, CKE = 1) ............... 412
- Example SPI Mode Requirements
  - (Slave Mode, CKE = 0) ............... 413
- Example SPI Slave Mode Requirements
  - (CE = 1) ........................................... 414
- External Clock Requirements ............. 400
- I2C Bus Data Requirements (Slave Mode) .... 416
- I2C Bus Data Requirements (Slave Mode, CKE = 0) .... 413
- I2C Bus Data Requirements (Slave Mode, CKE = 1) .... 414
- I2C Stop Condition Receive or Transmit Mode .. 261
- Program Memory Read .................... 403
- Program Memory Write .................... 404
- PWM Auto-Shutdown (P1RSEN = 0,
  - Auto-Restart Disabled) .................. 217
- PWM Auto-Shutdown (P1RSEN = 1,
  - Auto-Restart Enabled) .................. 217
- PWM Direction Change .................... 214
- PWM Direction Change at Near
  - 100% Duty Cycle ....................... 214
- PWM Output .............................................. 200
- Read and Write, 8-Bit Data,
  - Demultiplexed Address .................. 167
- Read, 16-Bit Data, Demultiplexed Address .... 170
- Read, 16-Bit Multiplexed Data, Fully
  - Multiplexed 16-Bit Address ............ 172
- Read, 16-Bit Multiplexed Data, Partially
  - Multiplexed Address .................. 171
- Read, 8-Bit Data, Fully Multiplexed
  - 16-Bit Address ....................... 169
- Read, 8-Bit Data, Partially Multiplexed
  - Address, Enable Strobe ............... 169
- Slow Rise Time (MCLR Tied to VDD, Vdd Rise > TPWRT) .... 53
- SPI Mode (Master Mode) .................... 226
- SPI Mode (Slave Mode, CKE = 0) ........... 228
- SPI Mode (Slave Mode, CKE = 1) ........... 228
- Synchronous Reception (Master Mode, SREN = 0) .... 287
- Synchronous Transmission .................. 285
- Synchronous Transmission (Through TXEN) ........ 286
- Time-out Sequence on Power-up
  - (MCLR Not Tied to VDD), Case 1 .......... 52
  - (MCLR Not Tied to VDD), Case 2 .......... 52
- Transition From SEC_RUN Mode to
  - PRI_RUN Mode .................................. 44
- Transition From RC_RUN Mode to
  - PRI_RUN Mode (HSPLL) .......... 324
- Transition to RC_RUN Mode ............... 44
- Write, 16-Bit Multiplexed Data, Fully
  - Multiplexed 16-Bit Address ............ 172
- Write, 16-Bit Multiplexed Data, Partially
  - Multiplexed Address .................. 171
- Write, 8-Bit Data, Demultiplexed Address .... 170
- Write, 8-Bit Data, Fully Multiplexed
  - 16-Bit Address ....................... 170
- Write, 8-Bit Data, Partially Multiplexed
  - Address ........................................ 168
- Write, 8-Bit Data, Partially Multiplexed
  - Address, Enable Strobe .................. 169
- Write, 8-Bit Data, Wait States Enabled,
  - Partially Multiplexed Address .......... 168
- Timing Diagrams and Specifications
  - Capture/Compare/PWM Requirements
    - (Including ECCP Modules) .............. 410
  - CLKO and I/O Requirements .............. 402, 403
  - EUSART Synchronous Receive
    - Requirements .................................. 419
- EUSART Synchronous Transmission
  - Requirements .................................. 419
- Example SPI Mode Requirements
  - (Master Mode, CKE = 0) ............... 411
- Example SPI Mode Requirements
  - (Master Mode, CKE = 1) ............... 412
- Example SPI Mode Requirements
  - (Slave Mode, CKE = 0) ............... 413
- Example SPI Slave Mode Requirements
  - (CE = 1) ........................................... 414
- External Clock Requirements ............. 400
- I2C Bus Data Requirements (Slave Mode) .... 416
PIC18F87J11 FAMILY

I2C Bus Start/Stop Bits Requirements ................................................... 415
Internal RC Accuracy (INTOSC, INTRC Sources) .......................... 401
MSSP I2C Bus Data Requirements .................................................... 418
MSSP I2C Bus Start/Stop Bits Requirements .................................. 417
Parallel Master Port Read Requirements ........................................ 408
Parallel Master Port Write ............................................................. 409
Parallel Slave Port Requirements .................................................. 407
PLL Clock .................................................................................. 401
Program Memory Write Requirements ........................................ 404
Reset, Watchdog Timer (WDT), Oscillator
  Start-up Timer (OST), Power-up
  Timer (PWRT) and Brown-out Reset ............................................ 405
Timer0 and Timer1 External Clock
  Requirements ................................................................................. 406
TSTFSZ ...................................................................................... 369
Two-Speed Start-up ................................................................. 313, 324
Texas Instruments Animation Instructions .............................................. 69
TXSTAx Register .............................................................................. 273
V
VDDCORE/VCAP Pin ................................................................. 323
Voltage Reference Specifications .................................................... 397
Voltage Regulator (On-Chip) .......................................................... 323
  Operation in Sleep Mode ......................................................... 324
  Power-up Requirements .......................................................... 324
W
Watchdog Timer (WDT) ................................................................. 313, 321
  Associated Registers ............................................................... 322
  Control Register ....................................................................... 321
  During Oscillator Failure .......................................................... 325
  Programming Considerations ..................................................... 321
WCOL ......................................................................................... 255, 256, 257, 260
WCOL Status Flag ........................................................................... 255, 256, 257, 260
WWW Address ................................................................................. 431
WWW, On-Line Support ................................................................. 5
X
XORLW ......................................................................................... 369
XORWF ......................................................................................... 370
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></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Temperature Range</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pattern</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Device**
PIC18F66J11/66J16/67J11(1),
PIC18F86J11/86J16/87J11(1),
PIC18F86J11/86J16/67J11T(2),
PIC18F86J11/86J16/67J11T(3)

**Temperature Range**
I = -40°C to +85°C (Industrial)

**Package**
PT = TQFP (Thin Quad Flatpack)

**Pattern**
QTP, SQTP, Code or Special Requirements (blank otherwise)

**Examples:**

a) PIC18F87J11-I/PT 301 = Industrial temp., TQFP package, QTP pattern #301.
b) PIC18F66J16T-I/PT = Tape and reel, Industrial temp., TQFP package.

**Note 1:**
F = Standard Voltage Range

**Note 2:**
T = in tape and reel
### AMERICAS

**Corporate Office**
2355 West Chandler Blvd.,
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277

Technical Support: [http://support.microchip.com](http://support.microchip.com)

Web Address:
[www.microchip.com](http://www.microchip.com)

**Atlanta**
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

**Kokomo**
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

**Santa Clara**
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

**Toronto**
Mississauga, Ontario, Canada
Tel: 905-673-0699
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**
Suits 3707-14, 37th Floor
Tower 6, The Gateway Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

**Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

**China - Beijing**
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

**China - Chengdu**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**China - Hong Kong SAR**
Tel: 852-2401-1200
Fax: 852-2401-3431

**China - Nanjing**
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

**China - Qingdao**
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

**China - Shanghai**
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

**China - Shenyang**
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

**China - Shenzhen**
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

**China - Wuhan**
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

**China - Xi'an**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

**China - Zhuhai**
Tel: 86-756-3210040
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**
Tel: 91-80-4182-8400
Fax: 91-80-4182-8422

**India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**India - Pune**
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

**Japan - Yokohama**
Tel: 81-45-471-6166
Fax: 81-45-471-6122

**Korea - Daegu**
Tel: 82-53-744-4301
Fax: 82-53-744-4302

**Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932

**Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**Taiwan - Hsin Chu**
Tel: 886-3-572-9526
Fax: 886-3-572-6459

**Taiwan - Kaohsiung**
Tel: 886-7-536-4818
Fax: 886-7-536-4803

**Taiwan - Taipei**
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

**Thailand - Bangkok**
Tel: 66-2-694-1351
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**Denmark - Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**UK - Wokingham**
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/02/08