# PIC18F87J11 FAMILY

## 64/80-Pin, High-Performance, 1-Mbit Flash Microcontrollers

### 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 (FSCM):
  - 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 $I^2C$™ Master and Slave modes
- Two Enhanced USART modules:
  - Supports RS-485, RS-232 and LIN/J2602
  - Auto-wake-up on Start bit
  - Auto-Baud Detect

### Peripheral Highlights (continued):

- 8-Bit Parallel Master Port/Enhanced Parallel Slave Port (PMP/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

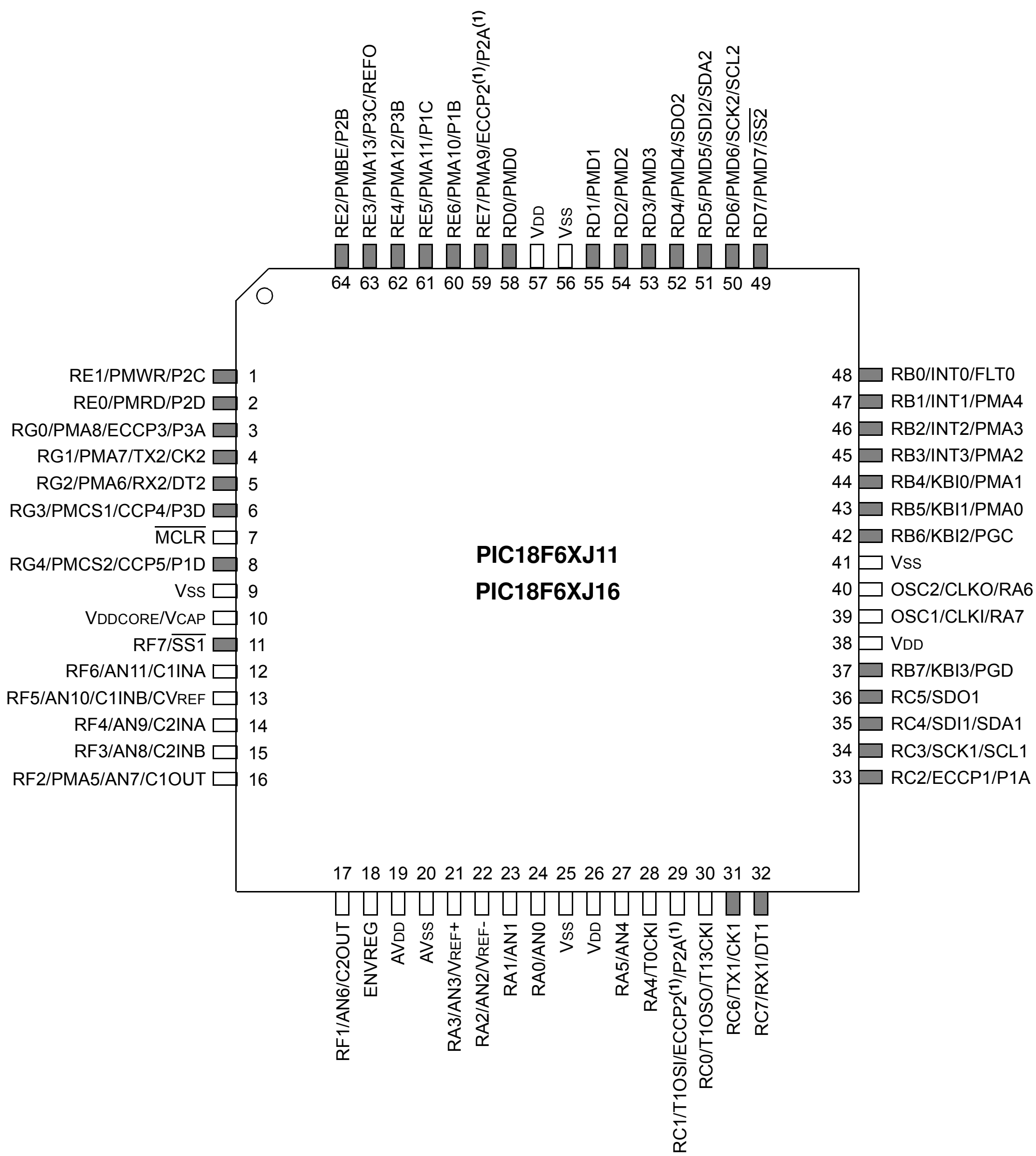
### Special Microcontroller Features:

- 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

| Device | Flash Program Memory (bytes) | SRAM Data Memory (bytes) | I/O | 10-Bit A/D (ch) | CCP/ECCP (PWM) | MSSP | | | EUSART | Comparators | Timers 8/16-Bit | External Bus | PMP/EPSP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | SPI | Master $I^2C$™ | | | | | |
| PIC18F66J11 | 64 kB | 3904 | 52 | 11 | 2/3 | 2 | Y | Y | 2 | 2 | 2/3 | N | Y |
| PIC18F66J16 | 96 kB | 3904 | 52 | 11 | 2/3 | 2 | Y | Y | 2 | 2 | 2/3 | N | Y |
| PIC18F67J11 | 128 kB | 3904 | 52 | 11 | 2/3 | 2 | Y | Y | 2 | 2 | 2/3 | N | Y |
| PIC18F86J11 | 64 kB | 3904 | 68 | 15 | 2/3 | 2 | Y | Y | 2 | 2 | 2/3 | Y | Y |
| PIC18F86J16 | 96 kB | 3904 | 68 | 15 | 2/3 | 2 | Y | Y | 2 | 2 | 2/3 | Y | Y |
| PIC18F87J11 | 128 kB | 3904 | 68 | 15 | 2/3 | 2 | Y | Y | 2 | 2 | 2/3 | Y | Y |

## Pin Diagrams

64-Pin TQFP

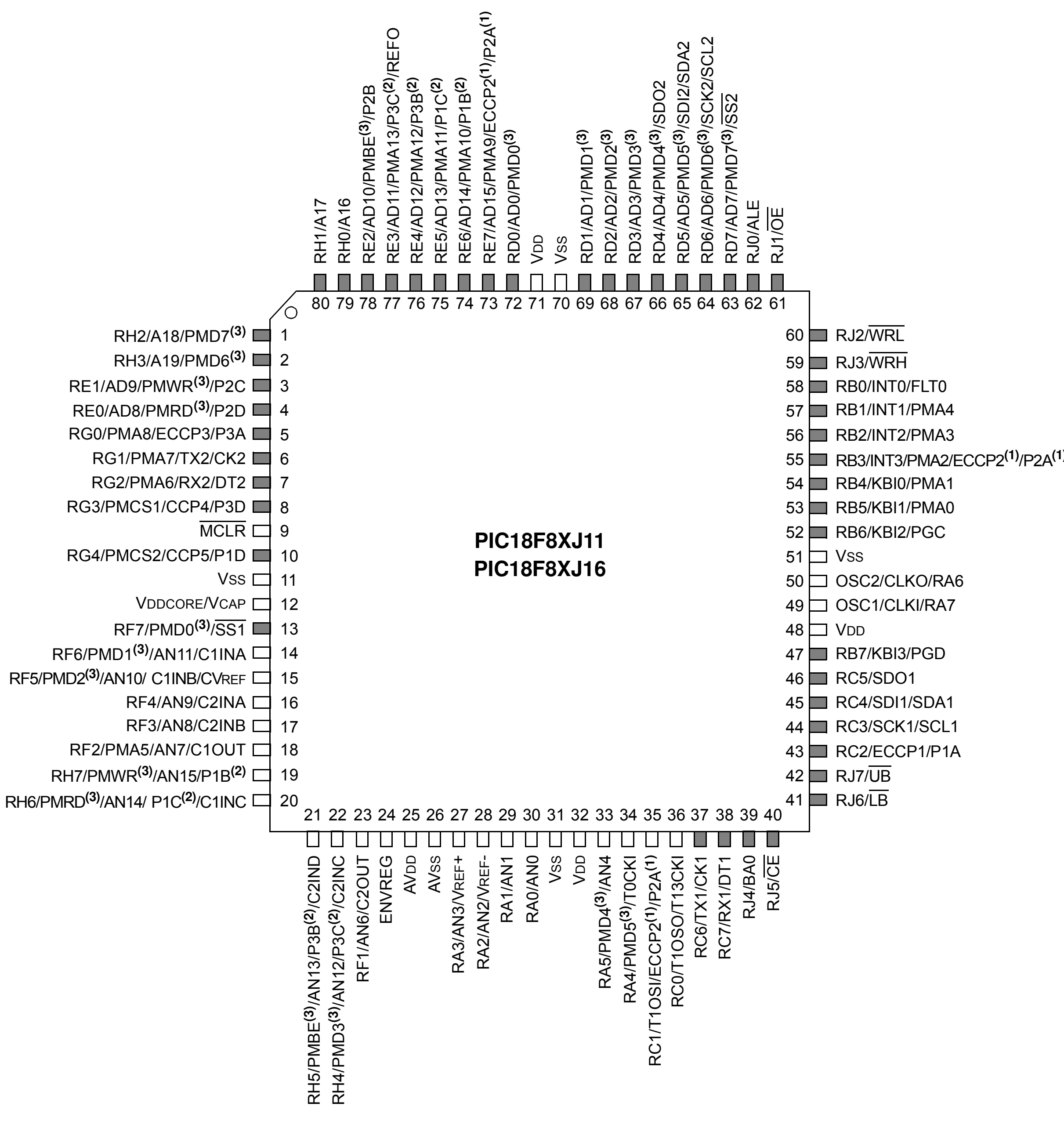



**Legend:** Shaded pins indicate pins that are tolerant up to +5.5V.

**Note 1:** The ECCP2/P2A pin placement depends on the CCP2MX Configuration bit setting.

## Pin Diagrams (Continued)

**80-Pin TQFP**




**Legend:** Shaded pins indicate pins that are tolerant up to +5.5V.

**Note 1:** The ECCP2/P2A pin placement depends on the CCP2MX Configuration bit and Processor mode settings.

**2:** P1B, P1C, P3B, and P3C pin placement depends on the ECCPMX Configuration bit setting.

**3:** PMP pin placement depends on the PMPMX Configuration bit setting.

## Table of Contents

## 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.

**NOTES:**

# 1.0 DEVICE OVERVIEW

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

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

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 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 $V_{DD}$). The oscillator block also provides 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 3904 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 (PC) 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 $I^2C$™ (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 (PMP) 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 28.0 "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.

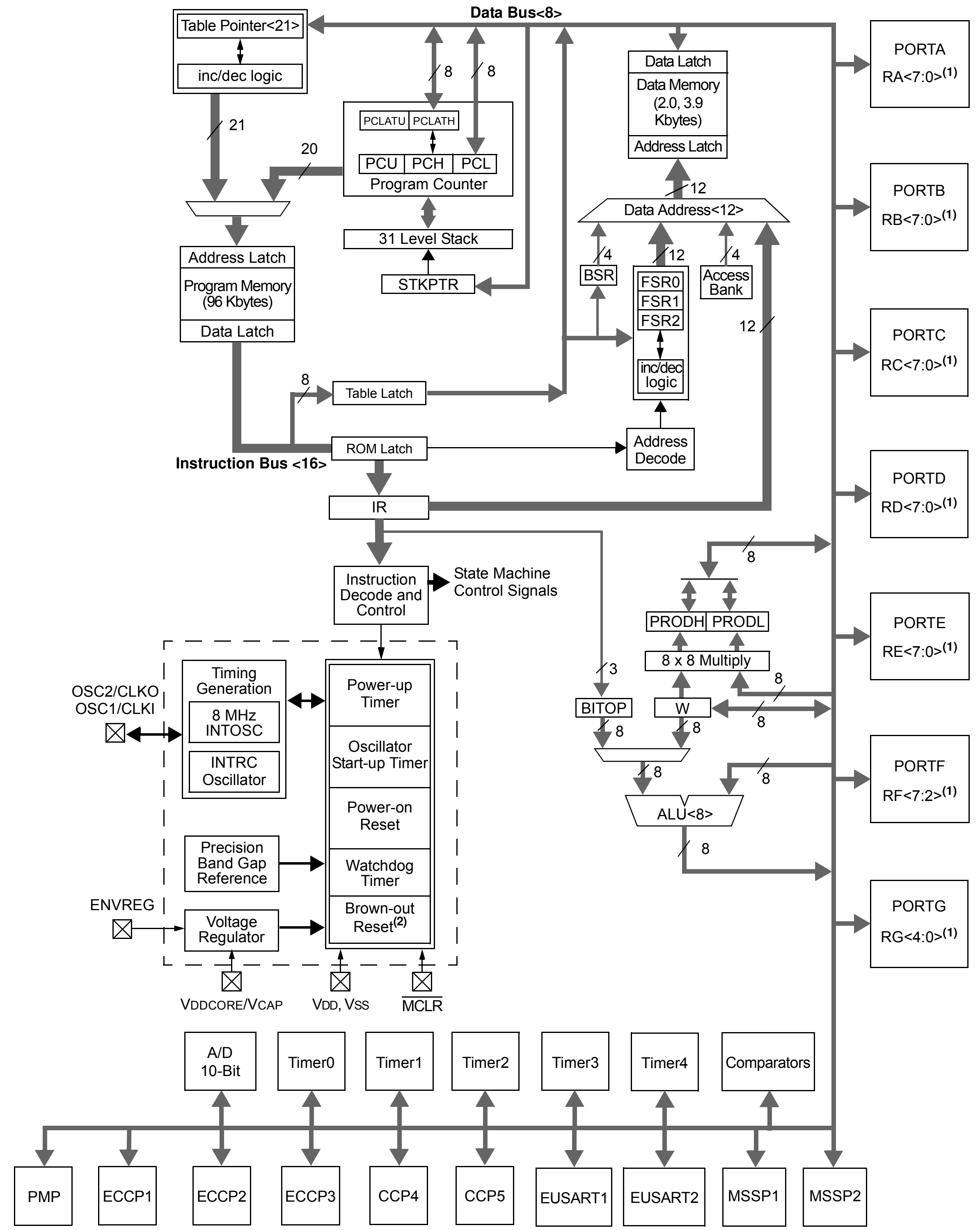
**TABLE 1-1: DEVICE FEATURES FOR THE PIC18F6XJ1X (64-PIN DEVICES)**

| Features | PIC18F66J11 | PIC18F66J16 | PIC18F67J11 |
|---|---|---|---|
| Operating Frequency | DC – 48 MHz | DC – 48 MHz | DC – 48 MHz |
| Program Memory (Bytes) | 64K | 96K | 128K |
| Program Memory (Instructions) | 32768 | 49152 | 65536 |
| Data Memory (Bytes) | 3904 | 3904 | 3904 |
| Interrupt Sources | 29 | | |
| I/O Ports | Ports A, B, C, D, E, F, G | | |
| Timers | 5 | | |
| Capture/Compare/PWM Modules | 2 | | |
| Enhanced Capture/Compare/PWM Modules | 3 | | |
| Serial Communications | MSSP (2), Enhanced USART (2) | | |
| Parallel Communications (PMP) | Yes | | |
| 10-Bit Analog-to-Digital Module | 11 Input Channels | | |
| Resets (and Delays) | POR, BOR, RESET Instruction, Stack Full, Stack Underflow, $\overline{\text{MCLR}}$, WDT (PWRT, OST) | | |
| Instruction Set | 75 Instructions, 83 with Extended Instruction Set Enabled | | |
| Packages | 64-Pin TQFP | | |

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

| Features | PIC18F86J11 | PIC18F86J16 | PIC18F87J11 |
|---|---|---|---|
| Operating Frequency | DC – 48 MHz | DC – 48 MHz | DC – 48 MHz |
| Program Memory (Bytes) | 64K | 96K | 128K |
| Program Memory (Instructions) | 32768 | 49152 | 65536 |
| Data Memory (Bytes) | 3904 | 3904 | 3904 |
| Interrupt Sources | 29 | | |
| I/O Ports | Ports A, B, C, D, E, F, G, H, J | | |
| Timers | 5 | | |
| Capture/Compare/PWM Modules | 2 | | |
| Enhanced Capture/Compare/PWM Modules | 3 | | |
| Serial Communications | MSSP (2), Enhanced USART (2) | | |
| Parallel Communications (PMP) | Yes | | |
| 10-Bit Analog-to-Digital Module | 15 Input Channels | | |
| Resets (and Delays) | POR, BOR, RESET Instruction, Stack Full, Stack Underflow, $\overline{\text{MCLR}}$, WDT (PWRT, OST) | | |
| Instruction Set | 75 Instructions, 83 with Extended Instruction Set Enabled | | |
| Packages | 80-Pin TQFP | | |

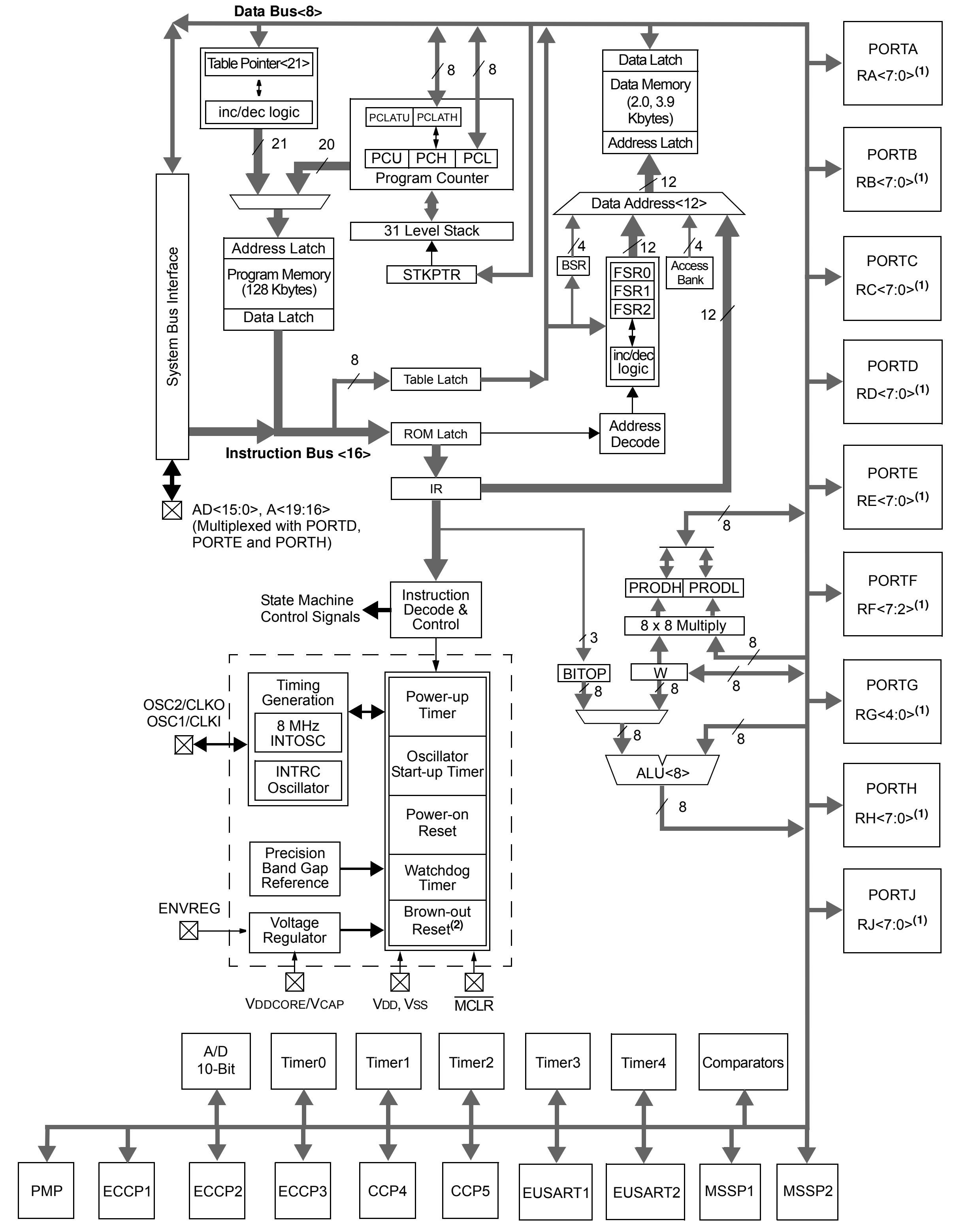
**FIGURE 1-1: PIC18F6XJ1X (64-PIN) BLOCK DIAGRAM**




**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 1-3: PIC18F6XJ1X PINOUT I/O DESCRIPTIONS**

| Pin Name | Pin Number 64-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| $\overline{MCLR}$ | 7 | I | ST | Master Clear (Reset) input. This pin is an active-low Reset to the device. |
| OSC1/CLKI/RA7 | 39 | | | Oscillator crystal or external clock input. Available only in External Oscillator modes (EC/ECPLL and HS/HSPLL). |
| OSC1 | | I | ST | Main oscillator input connection. Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; CMOS otherwise. |
| CLKI | | I | CMOS | Main clock input connection. External clock source input. Always associated with pin function, OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.) |
| RA7 | | I/O | TTL | General purpose I/O pin. Available only in INTIO2 and INTPLL2 Oscillator modes. |
| OSC2/CLKO/RA6 | 40 | | | Oscillator crystal or clock output. Available only in External Oscillator modes (EC/ECPLL and HS/HSPLL). |
| OSC2 | | O | — | Main oscillator feedback output connection. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| CLKO | | O | — | 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. |
| RA6 | | I/O | TTL | General purpose I/O pin. Available only in INTIO1 and INTPLL1 Oscillator modes. |

**Legend:**
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
$I^2C$ = ST with $I^2C$™ or SMB levels
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
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)**

| Pin Name | Pin Number 64-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTA is a bidirectional I/O port. |
| RA0/AN0 | 24 | | | |
| RA0 | | I/O | TTL | Digital I/O. |
| AN0 | | I | Analog | Analog Input 0. |
| RA1/AN1 | 23 | | | |
| RA1 | | I/O | TTL | Digital I/O. |
| AN1 | | I | Analog | Analog Input 1. |
| RA2/AN2/VREF- | 22 | | | |
| RA2 | | I/O | TTL | Digital I/O. |
| AN2 | | I | Analog | Analog Input 2. |
| VREF- | | I | Analog | A/D reference voltage (low) input. |
| RA3/AN3/VREF+ | 21 | | | |
| RA3 | | I/O | TTL | Digital I/O. |
| AN3 | | I | Analog | Analog Input 3. |
| VREF+ | | I | Analog | A/D reference voltage (high) input. |
| RA4/T0CKI | 28 | | | |
| RA4 | | I/O | ST | Digital I/O. |
| T0CKI | | I | ST | Timer0 external clock input. |
| RA5/AN4 | 27 | | | |
| RA5 | | I/O | TTL | Digital I/O. |
| AN4 | | I | Analog | Analog Input 4. |
| RA6 | — | — | — | See the OSC2/CLKO/RA6 pin. |
| RA7 | — | — | — | See the OSC1/CLKI/RA7 pin. |

**Legend:**
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
$I^2C$ = ST with $I^2C$™ or SMB levels
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
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)**

| Pin Name | Pin Number 64-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/FLT0/INT0 | 48 | | | |
| RB0 | | I/O | TTL | Digital I/O. |
| FLT0 | | I | ST | ECCP1/2/3 Fault input. |
| INT0 | | I | ST | External Interrupt 0. |
| RB1/INT1/PMA4 | 47 | | | |
| RB1 | | I/O | TTL | Digital I/O. |
| INT1 | | I | ST | External Interrupt 1. |
| PMA4 | | O | — | Parallel Master Port address. |
| RB2/INT2/PMA3 | 46 | | | |
| RB2 | | I/O | TTL | Digital I/O. |
| INT2 | | I | ST | External Interrupt 2. |
| PMA3 | | O | — | Parallel Master Port address. |
| RB3/INT3/PMA2 | 45 | | | |
| RB3 | | I/O | TTL | Digital I/O. |
| INT3 | | I | ST | External Interrupt 3. |
| PMA2 | | O | — | Parallel Master Port address. |
| RB4/KBI0/PMA1 | 44 | | | |
| RB4 | | I/O | TTL | Digital I/O. |
| KBI0 | | I | TTL | Interrupt-on-change pin. |
| PMA1 | | I/O | — | Parallel Master Port address. |
| RB5/KBI1/PMA0 | 43 | | | |
| RB5 | | I/O | TTL | Digital I/O. |
| KBI1 | | I | TTL | Interrupt-on-change pin. |
| PMA0 | | I/O | — | Parallel Master Port address. |
| RB6/KBI2/PGC | 42 | | | |
| RB6 | | I/O | TTL | Digital I/O. |
| KBI2 | | I | TTL | Interrupt-on-change pin. |
| PGC | | I/O | ST | In-Circuit Debugger and ICSP™ programming clock pin. |
| RB7/KBI3/PGD | 37 | | | |
| RB7 | | I/O | TTL | Digital I/O. |
| KBI3 | | I | TTL | Interrupt-on-change pin. |
| PGD | | I/O | ST | In-Circuit Debugger and ICSP programming data pin. |

**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); $I^2C$ = ST with $I^2C$™ or SMB levels

**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)**

| Pin Name | Pin Number 64-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTC is a bidirectional I/O port. |
| RC0/T1OSO/T13CKI | 30 | | | |
| RC0 | | I/O | ST | Digital I/O. |
| T1OSO | | O | — | Timer1 oscillator output. |
| T13CKI | | I | ST | Timer1/Timer3 external clock input. |
| RC1/T1OSI/ECCP2/P2A | 29 | | | |
| RC1 | | I/O | ST | Digital I/O. |
| T1OSI | | I | CMOS | Timer1 oscillator input. |
| ECCP2(1) | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A(1) | | O | — | ECCP2 PWM Output A. |
| RC2/ECCP1/P1A | 33 | | | |
| RC2 | | I/O | ST | Digital I/O. |
| ECCP1 | | I/O | ST | Capture 1 input/Compare 1 output/PWM1 output. |
| P1A | | O | — | ECCP1 PWM Output A. |
| RC3/SCK1/SCL1 | 34 | | | |
| RC3 | | I/O | ST | Digital I/O. |
| SCK1 | | I/O | ST | Synchronous serial clock input/output for SPI mode. |
| SCL1 | | I/O | $I^2C$ | Synchronous serial clock input/output for $I^2C$ mode. |
| RC4/SDI1/SDA1 | 35 | | | |
| RC4 | | I/O | ST | Digital I/O. |
| SDI1 | | I | ST | SPI data in. |
| SDA1 | | I/O | $I^2C$ | $I^2C$ data I/O. |
| RC5/SDO1 | 36 | | | |
| RC5 | | I/O | ST | Digital I/O. |
| SDO1 | | O | — | SPI data out. |
| RC6/TX1/CK1 | 31 | | | |
| RC6 | | I/O | ST | Digital I/O. |
| TX1 | | O | — | EUSART1 asynchronous transmit. |
| CK1 | | I/O | ST | EUSART1 synchronous clock (see related RX1/DT1). |
| RC7/RX1/DT1 | 32 | | | |
| RC7 | | I/O | ST | Digital I/O. |
| RX1 | | I | ST | EUSART1 asynchronous receive. |
| DT1 | | I/O | ST | 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); $I^2C$ = ST with $I^2C$™ or SMB levels

**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)**

| Pin Name | Pin Number 64-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTD is a bidirectional I/O port. |
| RD0/PMD0 | 58 | | | |
| RD0 | | I/O | ST | Digital I/O. |
| PMD0 | | I/O | TTL | Parallel Master Port data. |
| RD1/PMD1 | 55 | | | |
| RD1 | | I/O | ST | Digital I/O. |
| PMD1 | | I/O | TTL | Parallel Master Port data. |
| RD2/PMD2 | 54 | | | |
| RD2 | | I/O | ST | Digital I/O. |
| PMD2 | | I/O | TTL | Parallel Master Port data. |
| RD3/PMD3 | 53 | | | |
| RD3 | | I/O | ST | Digital I/O. |
| PMD3 | | I/O | TTL | Parallel Master Port data. |
| RD4/PMD4/SDO2 | 52 | | | |
| RD4 | | I/O | ST | Digital I/O. |
| PMD4 | | I/O | TTL | Parallel Master Port data. |
| SDO2 | | O | — | SPI data out. |
| RD5/PMD5/SDI2/SDA2 | 51 | | | |
| RD5 | | I/O | ST | Digital I/O. |
| PMD5 | | I/O | TTL | Parallel Master Port data. |
| SDI2 | | I | ST | SPI data in. |
| SDA2 | | I/O | ST | I²C data I/O. |
| RD6/PMD6/SCK2/SCL2 | 50 | | | |
| RD6 | | I/O | ST | Digital I/O. |
| PMD6 | | I/O | TTL | Parallel Master Port data. |
| 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/$\overline{SS2}$ | 49 | | | |
| RD7 | | I/O | ST | Digital I/O. |
| PMD7 | | I/O | TTL | Parallel Master Port data. |
| $\overline{SS2}$ | | I | TTL | SPI slave select input. |

**Legend:** TTL = TTL compatible input; ST = Schmitt Trigger input with CMOS levels; I = Input; P = Power; I²C = ST with I²C™ or SMB levels; CMOS = CMOS compatible input or output; Analog = Analog input; O = Output; 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)**

| Pin Name | Pin Number 64-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTE is a bidirectional I/O port. |
| RE0/PMRD/P2D | 2 | | | |
| RE0 | | I/O | ST | Digital I/O. |
| PMRD | | I/O | — | Parallel Master Port read strobe. |
| P2D | | O | — | ECCP2 PWM Output D. |
| RE1/PMWR/P2C | 1 | | | |
| RE1 | | I/O | ST | Digital I/O. |
| PMWR | | I/O | — | Parallel Master Port write strobe. |
| P2C | | O | — | ECCP2 PWM Output C. |
| RE2/PMBE/P2B | 64 | | | |
| RE2 | | I/O | ST | Digital I/O. |
| PMBE | | O | — | Parallel Master Port byte enable |
| P2B | | O | — | ECCP2 PWM Output B. |
| RE3/PMA13/P3C/REFO | 63 | | | |
| RE3 | | I/O | ST | Digital I/O. |
| PMA13 | | O | — | Parallel Master Port address. |
| P3C | | O | — | ECCP3 PWM Output C. |
| REFO | | O | — | Reference clock out. |
| RE4/PMA12/P3B | 62 | | | |
| RE4 | | I/O | ST | Digital I/O. |
| PMA12 | | O | — | Parallel Master Port address. |
| P3B | | O | — | ECCP3 PWM Output B. |
| RE5/PMA11/P1C | 61 | | | |
| RE5 | | I/O | ST | Digital I/O. |
| PMA11 | | O | — | Parallel Master Port address. |
| P1C | | O | — | ECCP1 PWM Output C. |
| RE6/PMA10/P1B | 60 | | | |
| RE6 | | I/O | ST | Digital I/O. |
| PMA10 | | O | — | Parallel Master Port address. |
| P1B | | O | — | ECCP1 PWM Output B. |
| RE7/PMA9/ECCP2/P2A | 59 | | | |
| RE7 | | I/O | ST | Digital I/O. |
| PMA9 | | O | — | Parallel Master Port address. |
| ECCP2(2) | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A(2) | | O | — | ECCP2 PWM Output A. |

**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); $I^2C$ = ST with $I^2C$™ or SMB levels

**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)**

| Pin Name | Pin Number 64-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTF is a bidirectional I/O port. |
| RF1/AN6/C2OUT | 17 | | | |
| RF1 | | I/O | ST | Digital I/O. |
| AN6 | | I | Analog | Analog Input 6. |
| C2OUT | | O | — | Comparator 2 output. |
| RF2/PMA5/AN7/C1OUT | 16 | | | |
| RF2 | | I/O | ST | Digital I/O. |
| PMA5 | | O | — | Parallel Master Port address. |
| AN7 | | I | Analog | Analog Input 7. |
| C1OUT | | O | — | Comparator 1 output. |
| RF3/AN8/C2INB | 15 | | | |
| RF3 | | I/O | ST | Digital input. |
| AN8 | | I | Analog | Analog Input 8. |
| C2INB | | I | Analog | Comparator 2 Input B. |
| RF4/AN9/C2INA | 14 | | | |
| RF4 | | I/O | ST | Digital input. |
| AN9 | | I | Analog | Analog Input 8. |
| C2INA | | I | Analog | Comparator 2 Input A. |
| RF5/AN10/C1INB/CVREF | 13 | | | |
| RF5 | | I/O | ST | Digital input. |
| AN10 | | I | Analog | Analog Input 10. |
| C1INB | | I | Analog | Comparator 1 Input B. |
| CVREF | | O | Analog | Comparator reference voltage output. |
| RF6/AN11/C1INA | 12 | | | |
| RF6 | | I/O | ST | Digital I/O. |
| AN11 | | I | Analog | Analog Input 11. |
| C1INA | | I | Analog | Comparator 1 Input A. |
| RF7/$\overline{\text{SS1}}$ | 11 | | | |
| RF7 | | I/O | ST | Digital I/O. |
| $\overline{\text{SS1}}$ | | I | TTL | SPI slave select input. |

**Legend:**
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
I²C = ST with I²C™ or SMB levels
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
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)**

| Pin Name | Pin Number 64-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTG is a bidirectional I/O port. |
| RG0/PMA8/ECCP3/P3A | 3 | | | |
| RG0 | | I/O | ST | Digital I/O. |
| PMA8 | | O | — | Parallel Master Port address. |
| ECCP3 | | I/O | ST | Capture 3 input/Compare 3 output/PWM3 output. |
| P3A | | O | — | ECCP3 PWM Output A. |
| RG1/PMA7/TX2/CK2 | 4 | | | |
| RG1 | | I/O | ST | Digital I/O. |
| PMA7 | | O | — | Parallel Master Port address. |
| TX2 | | O | — | EUSART2 asynchronous transmit. |
| CK2 | | I/O | ST | EUSART2 synchronous clock (see related RX2/DT2). |
| RG2/PMA6/RX2/DT2 | 5 | | | |
| RG2 | | I/O | ST | Digital I/O. |
| PMA6 | | O | — | Parallel Master Port address. |
| RX2 | | I | ST | EUSART2 asynchronous receive. |
| DT2 | | I/O | ST | EUSART2 synchronous data (see related TX2/CK2). |
| RG3/PMCS1/CCP4/P3D | 6 | | | |
| RG3 | | I/O | ST | Digital I/O. |
| PMCS1 | | O | — | Parallel Master Port Chip Select 1. |
| CCP4 | | I/O | ST | Capture 4 input/Compare 4 output/PWM4 output. |
| P3D | | O | — | ECCP3 PWM Output D. |
| RG4/PMCS2/CCP5/P1D | 8 | | | |
| RG4 | | I/O | ST | Digital I/O. |
| PMCS2 | | O | — | Parallel Master Port Chip Select 2. |
| CCP5 | | I/O | ST | Capture 5 input/Compare 5 output/PWM5 output. |
| P1D | | O | — | ECCP1 PWM Output D. |
| VSS | 9, 25, 41, 56 | P | — | Ground reference for logic and I/O pins. |
| VDD | 26, 38, 57 | P | — | Positive supply for peripheral digital logic and I/O pins. |
| AVSS | 20 | P | — | Ground reference for analog modules. |
| AVDD | 19 | P | — | Positive supply for analog modules. |
| ENVREG | 18 | I | ST | Enable for on-chip voltage regulator. |
| VDDCORE/VCAP | 10 | | | Core logic power or external filter capacitor connection. |
| VDDCORE | | P | — | Positive supply for microcontroller core logic (regulator disabled). |
| VCAP | | P | — | 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); I²C = ST with I²C™ or SMB levels

**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

| Pin Name | Pin Number 80-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| $\overline{MCLR}$ | 9 | I | ST | Master Clear (Reset) input. This pin is an active-low Reset to the device. |
| OSC1/CLKI/RA7 | 49 | | | Oscillator crystal or external clock input. Available only in External Oscillator modes (EC/ECPLL and HS/HSPLL). |
| OSC1 | | I | ST | Main oscillator input connection. Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; CMOS otherwise. |
| CLKI | | I | CMOS | Main clock input connection. External clock source input. Always associated with pin function, OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.) |
| RA7 | | I/O | TTL | General purpose I/O pin. Available only in INTIO2 and INTPLL2 Oscillator modes. |
| OSC2/CLKO/RA6 | 50 | | | Oscillator crystal or clock output. Available only in External Oscillator modes (EC/ECPLL and HS/HSPLL). |
| OSC2 | | O | — | Main oscillator feedback output connection. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| CLKO | | O | — | System cycle clock output ($F_{OSC}$/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. |
| RA6 | | I/O | TTL | General purpose I/O pin. Available only in INTIO and INTPLL Oscillator modes. |

**Legend:**
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
$I^2C$ = ST with $I^2C$™ or SMB levels
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to $V_{DD}$)

**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)**

| Pin Name | Pin Number 80-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTA is a bidirectional I/O port. |
| RA0/AN0 | 30 | | | |
| RA0 | | I/O | TTL | Digital I/O. |
| AN0 | | I | Analog | Analog Input 0. |
| RA1/AN1 | 29 | | | |
| RA1 | | I/O | TTL | Digital I/O. |
| AN1 | | I | Analog | Analog Input 1. |
| RA2/AN2/VREF- | 28 | | | |
| RA2 | | I/O | TTL | Digital I/O. |
| AN2 | | I | Analog | Analog Input 2. |
| VREF- | | I | Analog | A/D reference voltage (low) input. |
| RA3/AN3/VREF+ | 27 | | | |
| RA3 | | I/O | TTL | Digital I/O. |
| AN3 | | I | Analog | Analog Input 3. |
| VREF+ | | I | Analog | A/D reference voltage (high) input. |
| RA4/PMD5/T0CKI | 34 | | | |
| RA4 | | I/O | ST | Digital I/O. |
| PMD5(7) | | I/O | TTL | Parallel Master Port data. |
| T0CKI | | I | ST | Timer0 external clock input. |
| RA5/PMD4/AN4 | 33 | | | |
| RA5 | | I/O | TTL | Digital I/O. |
| PMD4(7) | | I/O | TTL | Parallel Master Port data. |
| AN4 | | I | Analog | Analog Input 4. |
| RA6 | — | — | — | See the OSC2/CLKO/RA6 pin. |
| RA7 | — | — | — | See the OSC1/CLKI/RA7 pin. |

**Legend:**
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
$I^2C$ = ST with $I^2C$™ or SMB levels
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
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)**

| Pin Name | Pin Number 80-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/FLT0/INT0 | 58 | | | |
| RB0 | | I/O | TTL | Digital I/O. |
| FLT0 | | I | ST | ECCP1/2/3 Fault input. |
| INT0 | | I | ST | External Interrupt 0. |
| RB1/INT1/PMA4 | 57 | | | |
| RB1 | | I/O | TTL | Digital I/O. |
| INT1 | | I | ST | External Interrupt 1. |
| PMA4 | | O | — | Parallel Master Port address. |
| RB2/INT2/PMA3 | 56 | | | |
| RB2 | | I/O | TTL | Digital I/O. |
| INT2 | | I | ST | External Interrupt 2. |
| PMA3 | | O | — | Parallel Master Port address. |
| RB3/INT3/PMA2/ ECCP2/P2A | 55 | | | |
| RB3 | | I/O | TTL | Digital I/O. |
| INT3 | | I | ST | External Interrupt 3. |
| PMA2 | | O | — | Parallel Master Port address. |
| ECCP2(1) | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A(1) | | O | — | ECCP2 PWM Output A. |
| RB4/KBI0/PMA1 | 54 | | | |
| RB4 | | I/O | TTL | Digital I/O. |
| KBI0 | | I | TTL | Interrupt-on-change pin. |
| PMA1 | | I/O | — | Parallel Master Port address. |
| RB5/KBI1/PMA0 | 53 | | | |
| RB5 | | I/O | TTL | Digital I/O. |
| KBI1 | | I | TTL | Interrupt-on-change pin. |
| PMA0 | | I/O | — | Parallel Master Port address. |
| RB6/KBI2/PGC | 52 | | | |
| RB6 | | I/O | TTL | Digital I/O. |
| KBI2 | | I | TTL | Interrupt-on-change pin. |
| PGC | | I/O | ST | In-Circuit Debugger and ICSP™ programming clock pin. |
| RB7/KBI3/PGD | 47 | | | |
| RB7 | | I/O | TTL | Digital I/O. |
| KBI3 | | I | TTL | Interrupt-on-change pin. |
| PGD | | I/O | ST | In-Circuit Debugger and ICSP programming data pin. |

**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 $V_{DD}$)
$I^2C$ = ST with $I^2C$™ or SMB levels

**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)**

| Pin Name | Pin Number 80-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTC is a bidirectional I/O port. |
| RC0/T1OSO/T13CKI | 36 | | | |
| RC0 | | I/O | ST | Digital I/O. |
| T1OSO | | O | — | Timer1 oscillator output. |
| T13CKI | | I | ST | Timer1/Timer3 external clock input. |
| RC1/T1OSI/ECCP2/P2A | 35 | | | |
| RC1 | | I/O | ST | Digital I/O. |
| T1OSI | | I | CMOS | Timer1 oscillator input. |
| ECCP2[(2)] | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A[(2)] | | O | — | ECCP2 PWM Output A. |
| RC2/ECCP1/P1A | 43 | | | |
| RC2 | | I/O | ST | Digital I/O. |
| ECCP1 | | I/O | ST | Capture 1 input/Compare 1 output/PWM1 output. |
| P1A | | O | — | ECCP1 PWM Output A. |
| RC3/SCK1/SCL1 | 44 | | | |
| RC3 | | I/O | ST | Digital I/O. |
| SCK1 | | I/O | ST | Synchronous serial clock input/output for SPI mode. |
| SCL1 | | I/O | $I^2C$ | Synchronous serial clock input/output for $I^2C$ mode. |
| RC4/SDI1/SDA1 | 45 | | | |
| RC4 | | I/O | ST | Digital I/O. |
| SDI1 | | I | ST | SPI data in. |
| SDA1 | | I/O | $I^2C$ | $I^2C$ data I/O. |
| RC5/SDO1 | 46 | | | |
| RC5 | | I/O | ST | Digital I/O. |
| SDO1 | | O | — | SPI data out. |
| RC6/TX1/CK1 | 37 | | | |
| RC6 | | I/O | ST | Digital I/O. |
| TX1 | | O | — | EUSART1 asynchronous transmit. |
| CK1 | | I/O | ST | EUSART1 synchronous clock (see related RX1/DT1). |
| RC7/RX1/DT1 | 38 | | | |
| RC7 | | I/O | ST | Digital I/O. |
| RX1 | | I | ST | EUSART1 asynchronous receive. |
| DT1 | | I/O | ST | 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 $V_{DD}$); $I^2C$ = ST with $I^2C$™ or SMB levels

**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)**

| Pin Name | Pin Number 80-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTD is a bidirectional I/O port. |
| RD0/AD0/PMD0 | 72 | | | |
| RD0 | | I/O | ST | Digital I/O. |
| AD0 | | I/O | TTL | External Memory Address/Data 0. |
| PMD0[(6)] | | I/O | TTL | Parallel Master Port data. |
| RD1/AD1/PMD1 | 69 | | | |
| RD1 | | I/O | ST | Digital I/O. |
| AD1 | | I/O | TTL | External Memory Address/Data 1. |
| PMD1[(6)] | | I/O | TTL | Parallel Master Port data. |
| RD2/AD2/PMD2 | 68 | | | |
| RD2 | | I/O | ST | Digital I/O. |
| AD2 | | I/O | TTL | External Memory Address/Data 2. |
| PMD2[(6)] | | I/O | TTL | Parallel Master Port data. |
| RD3/AD3/PMD3 | 67 | | | |
| RD3 | | I/O | ST | Digital I/O. |
| AD3 | | I/O | TTL | External Memory Address/Data 3. |
| PMD3[(6)] | | I/O | TTL | Parallel Master Port data. |
| RD4/AD4/PMD4/SDO2 | 66 | | | |
| RD4 | | I/O | ST | Digital I/O. |
| AD4 | | I/O | TTL | External Memory Address/Data 4. |
| PMD4[(6)] | | I/O | TTL | Parallel Master Port data. |
| SDO2 | | O | — | SPI data out. |
| RD5/AD5/PMD5/SDI2/SDA2 | 65 | | | |
| RD5 | | I/O | ST | Digital I/O. |
| AD5 | | I/O | TTL | External Memory Address/Data 5. |
| PMD5[(6)] | | I/O | TTL | Parallel Master Port data. |
| SDI2 | | I | ST | SPI data in. |
| SDA2 | | I/O | ST | $I^2C$ data I/O. |
| RD6/AD6/PMD6/SCK2/SCL2 | 64 | | | |
| RD6 | | I/O | ST | Digital I/O. |
| AD6 | | I/O | TTL | External Memory Address/Data 6. |
| PMD6[(6)] | | I/O | TTL | Parallel Master Port data. |
| SCK2 | | I/O | ST | Synchronous serial clock input/output for SPI mode. |
| SCL2 | | I/O | ST | Synchronous serial clock input/output for $I^2C$ mode. |
| RD7/AD7/PMD7/$\overline{\text{SS2}}$ | 63 | | | |
| RD7 | | I/O | ST | Digital I/O. |
| AD7 | | I/O | TTL | External Memory Address/Data 7. |
| PMD7[(6)] | | I/O | TTL | Parallel Master Port data. |
| $\overline{\text{SS2}}$ | | 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 $V_{DD}$); $I^2C$ = ST with $I^2C$™ or SMB levels

**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)**

| Pin Name | Pin Number 80-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTE is a bidirectional I/O port. |
| RE0/AD8/PMRD/P2D | 4 | | | |
| RE0 | | I/O | ST | Digital I/O. |
| AD8 | | I/O | TTL | External Memory Address/Data 8. |
| PMRD(6) | | I/O | — | Parallel Master Port read strobe. |
| P2D | | O | — | ECCP2 PWM Output D. |
| RE1/AD9/PMWR/P2C | 3 | | | |
| RE1 | | I/O | ST | Digital I/O. |
| AD9 | | I/O | TTL | External Memory Address/Data 9. |
| PMWR(6) | | I/O | — | Parallel Master Port write strobe. |
| P2C | | O | — | ECCP2 PWM Output C. |
| RE2/AD10/PMBE/P2B | 78 | | | |
| RE2 | | I/O | ST | Digital I/O. |
| AD10 | | I/O | TTL | External Memory Address/Data 10. |
| PMBE(6) | | O | — | Parallel Master Port byte enable. |
| P2B | | O | — | ECCP2 PWM Output B. |
| RE3/AD11/PMA13/P3C/REFO | 77 | | | |
| RE3 | | I/O | ST | Digital I/O. |
| AD11 | | I/O | TTL | External Memory Address/Data 11. |
| PMA13 | | O | — | Parallel Master Port address. |
| P3C(3) | | O | — | ECCP3 PWM Output C. |
| REFO | | O | — | Reference clock out. |
| RE4/AD12/PMA12/P3B | 76 | | | |
| RE4 | | I/O | ST | Digital I/O. |
| AD12 | | I/O | TTL | External Memory Address/Data 12. |
| PMA12 | | O | — | Parallel Master Port address. |
| P3B(3) | | O | — | ECCP3 PWM Output B. |
| RE5/AD13/PMA11/P1C | 75 | | | |
| RE5 | | I/O | ST | Digital I/O. |
| AD13 | | I/O | TTL | External Memory Address/Data 13. |
| PMA11 | | O | — | Parallel Master Port address. |
| P1C(3) | | O | — | ECCP1 PWM Output C. |
| RE6/AD14/PMA10/P1B | 74 | | | |
| RE6 | | I/O | ST | Digital I/O. |
| AD14 | | I/O | TTL | External Memory Address/Data 14. |
| PMA10 | | O | — | Parallel Master Port address. |
| P1B(3) | | O | — | ECCP1 PWM Output B. |
| RE7/AD15/PMA9/ECCP2/P2A | 73 | | | |
| RE7 | | I/O | ST | Digital I/O. |
| AD15 | | I/O | TTL | External Memory Address/Data 15. |
| PMA9 | | O | — | Parallel Master Port address. |
| ECCP2(4) | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A(4) | | O | — | ECCP2 PWM Output A. |

**Legend:**
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
$I^2C$ = ST with $I^2C$™ or SMB levels
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to $V_{DD}$)

**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)**

| Pin Name | Pin Number 80-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTF is a bidirectional I/O port. |
| RF1/AN6/C2OUT | 23 | | | |
| RF1 | | I/O | ST | Digital I/O. |
| AN6 | | I | Analog | Analog Input 6. |
| C2OUT | | O | — | Comparator 2 output. |
| RF2/PMA5/AN7/C1OUT | 18 | | | |
| RF2 | | I/O | ST | Digital I/O. |
| PMA5 | | O | — | Parallel Master Port address. |
| AN7 | | I | Analog | Analog Input 7. |
| C1OUT | | O | — | Comparator 1 output. |
| RF3/AN8/C2INB | 17 | | | |
| RF3 | | I/O | ST | Digital input. |
| AN8 | | I | Analog | Analog Input 8. |
| C2INB | | I | Analog | Comparator 2 Input B. |
| RF4/AN9/C2INA | 16 | | | |
| RF4 | | I/O | ST | Digital input. |
| AN9 | | I | Analog | Analog Input 8. |
| C2INA | | I | Analog | Comparator 2 Input A. |
| RF5/PMD2/AN10/ C1INB/CVREF | 15 | | | |
| RF5 | | I/O | ST | Digital I/O. |
| PMD2(7) | | I/O | TTL | Parallel Master Port address. |
| AN10 | | I | Analog | Analog Input 10. |
| C1INB | | I | Analog | Comparator 1 Input B. |
| CVREF | | O | Analog | Comparator reference voltage output. |
| RF6/PMD1/AN11/C1INA | 14 | | | |
| RF6 | | I/O | ST | Digital I/O. |
| PMD1(7) | | I/O | TTL | Parallel Master Port address. |
| AN11 | | I | Analog | Analog Input 11. |
| C1INA | | I | Analog | Comparator 1 Input A. |
| RF7/PMD0/$\overline{\text{SS1}}$ | 13 | | | |
| RF7 | | I/O | ST | Digital I/O. |
| PMD0(7) | | I/O | TTL | Parallel Master Port address. |
| $\overline{\text{SS1}}$ | | 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); $I^2C$ = ST with $I^2C$™ or SMB levels

**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)**

| Pin Name | Pin Number 80-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTG is a bidirectional I/O port. |
| RG0/PMA8/ECCP3/P3A | 5 | | | |
| RG0 | | I/O | ST | Digital I/O. |
| PMA8 | | O | — | Parallel Master Port address. |
| ECCP3 | | I/O | ST | Capture 3 input/Compare 3 output/PWM3 output. |
| P3A | | O | — | ECCP3 PWM Output A. |
| RG1/PMA7/TX2/CK2 | 6 | | | |
| RG1 | | I/O | ST | Digital I/O. |
| PMA7 | | O | — | Parallel Master Port address. |
| TX2 | | O | — | EUSART2 asynchronous transmit. |
| CK2 | | I/O | ST | EUSART2 synchronous clock (see related RX2/DT2). |
| RG2/PMA6/RX2/DT2 | 7 | | | |
| RG2 | | I/O | ST | Digital I/O. |
| PMA6 | | I/O | — | Parallel Master Port address. |
| RX2 | | I | ST | EUSART2 asynchronous receive. |
| DT2 | | I/O | ST | EUSART2 synchronous data (see related TX2/CK2). |
| RG3/PMCS1/CCP4/P3D | 8 | | | |
| RG3 | | I/O | ST | Digital I/O. |
| PMCS1 | | I/O | — | Parallel Master Port Chip Select 1. |
| CCP4 | | I/O | ST | Capture 4 input/Compare 4 output/PWM4 output. |
| P3D | | O | — | ECCP3 PWM Output D. |
| RG4/PMCS2/CCP5/P1D | 10 | | | |
| RG4 | | I/O | ST | Digital I/O. |
| PMCS2 | | O | — | Parallel Master Port Chip Select 2. |
| CCP5 | | I/O | ST | Capture 5 input/Compare 5 output/PWM5 output. |
| P1D | | O | — | ECCP1 PWM Output D. |

**Legend:**
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
$I^2C$ = ST with $I^2C$™ or SMB levels
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to $V_{DD}$)

**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)

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

**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); $I^2C$ = ST with $I^2C$™ or SMB levels

**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)**

| Pin Name | Pin Number 80-TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTJ is a bidirectional I/O port. |
| RJ0/ALE | 62 | | | |
| RJ0 | | I/O | ST | Digital I/O. |
| ALE | | O | — | External memory address latch enable. |
| RJ1/$\overline{OE}$ | 61 | | | |
| RJ1 | | I/O | ST | Digital I/O. |
| $\overline{OE}$ | | O | — | External memory output enable. |
| RJ2/$\overline{WRL}$ | 60 | | | |
| RJ2 | | I/O | ST | Digital I/O. |
| $\overline{WRL}$ | | O | — | External memory write low control. |
| RJ3/$\overline{WRH}$ | 59 | | | |
| RJ3 | | I/O | ST | Digital I/O. |
| $\overline{WRH}$ | | O | — | External memory write high control. |
| RJ4/BA0 | 39 | | | |
| RJ4 | | I/O | ST | Digital I/O. |
| BA0 | | O | — | External Memory Byte Address 0 control. |
| RJ5/$\overline{CE}$ | 40 | | | |
| RJ5 | | I/O | ST | Digital I/O |
| $\overline{CE}$ | | O | — | External memory chip enable control. |
| RJ6/$\overline{LB}$ | 41 | | | |
| RJ6 | | I/O | ST | Digital I/O. |
| $\overline{LB}$ | | O | — | External memory low byte control. |
| RJ7/$\overline{UB}$ | 42 | | | |
| RJ7 | | I/O | ST | Digital I/O. |
| $\overline{UB}$ | | O | — | External memory high byte control. |
| VSS | 11, 31, 51, 70 | P | — | Ground reference for logic and I/O pins. |
| VDD | 32, 48, 71 | P | — | Positive supply for peripheral digital logic and I/O pins. |
| AVSS | 26 | P | — | Ground reference for analog modules. |
| AVDD | 25 | P | — | Positive supply for analog modules. |
| ENVREG | 24 | I | ST | Enable for on-chip voltage regulator. |
| VDDCORE/VCAP | 12 | | | Core logic power or external filter capacitor connection. |
| VDDCORE | | P | — | Positive supply for microcontroller core logic (regulator disabled). |
| VCAP | | P | — | 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)
$I^2C$ = ST with $I^2C$™ or SMB levels

**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).

**NOTES:**

# 2.0 GUIDELINES FOR GETTING STARTED WITH PIC18FJ MICROCONTROLLERS

## 2.1 Basic Connection Requirements

Getting started with the PIC18F87J11 family family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All VDD and VSS pins (see **Section 2.2 "Power Supply Pins"**)
- All AVDD and AVSS pins, regardless of whether or not the analog device features are used (see **Section 2.2 "Power Supply Pins"**)
- MCLR pin (see **Section 2.3 "Master Clear (MCLR) Pin"**)
- ENVREG (if implemented) and VCAP/VDDCORE pins (see **Section 2.4 "Voltage Regulator Pins (ENVREG and VCAP/VDDCORE)"**)

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

- PGC/PGD pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see **Section 2.5 "ICSP Pins"**)
- OSCI and OSCO pins when an external oscillator source is used (see **Section 2.6 "External Oscillator Pins"**)

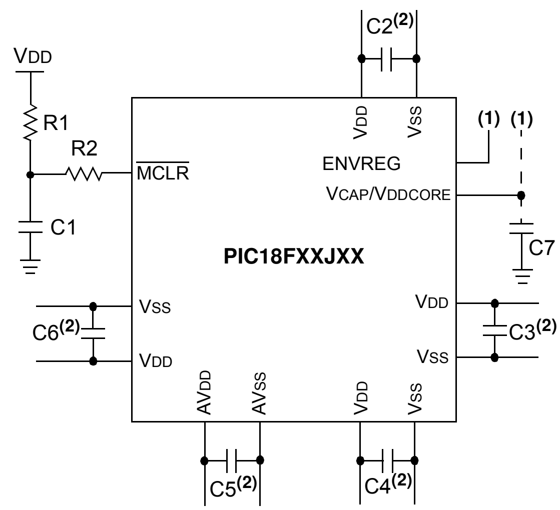
Additionally, the following pins may be required:

- VREF+/VREF- pins are used when external voltage reference for analog modules is implemented

| | |
|---|---|
| **Note:** | The AVDD and AVSS pins must always be connected, regardless of whether any of the analog modules are being used. |

The minimum mandatory connections are shown in Figure 2-1.

**FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS**




**Key (all values are recommendations):**

C1 through C6: 0.1 µF, 20V ceramic

C7: 10 µF, 6.3V or greater, tantalum or ceramic

R1: 10 kΩ

R2: 100Ω to 470Ω

**Note 1:** See **Section 2.4 "Voltage Regulator Pins (ENVREG and VCAP/VDDCORE)"** for explanation of ENVREG pin connections.

**2:** The example shown is for a PIC18F device with five VDD/VSS and AVDD/AVSS pairs. Other devices may have more or less pairs; adjust the number of decoupling capacitors appropriately.

## 2.2 Power Supply Pins

### 2.2.1 DECOUPLING CAPACITORS

The use of decoupling capacitors on every pair of power supply pins, such as $V_{DD}$, $V_{SS}$, $AV_{DD}$ and $AV_{SS}$, is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** A 0.1 μF (100 nF), 10-20V capacitor is recommended. The capacitor should be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- **Placement on the printed circuit board:** The decoupling capacitors should be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).
- **Handling high-frequency noise:** If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of 0.01 μF to 0.001 μF. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1 μF in parallel with 0.001 μF).
- **Maximizing performance:** On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

### 2.2.2 TANK CAPACITORS

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

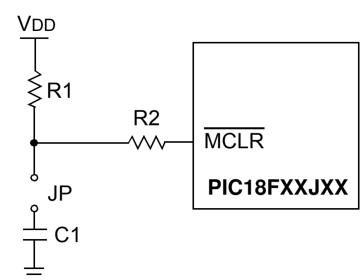
## 2.3 Master Clear ($\overline{MCLR}$) Pin

The $\overline{MCLR}$ pin provides two specific device functions: Device Reset, and Device Programming and Debugging. If programming and debugging are not required in the end application, a direct connection to $V_{DD}$ may be all that is required. The addition of other components, to help increase the application's resistance to spurious Resets from voltage sags, may be beneficial. A typical configuration is shown in Figure 2-1. Other circuit designs may be implemented, depending on the application's requirements.

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

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

**FIGURE 2-2: EXAMPLE OF $\overline{MCLR}$ PIN CONNECTIONS**




**Note 1:** R1 ≤ 10 kΩ is recommended. A suggested starting value is 10 kΩ. Ensure that the $\overline{MCLR}$ pin $V_{IH}$ and $V_{IL}$ specifications are met.

**2:** R2 ≤ 470Ω will limit any current flowing into $\overline{MCLR}$ from the external capacitor, C, in the event of $\overline{MCLR}$ pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS). Ensure that the $\overline{MCLR}$ pin $V_{IH}$ and $V_{IL}$ specifications are met.

## 2.4 Voltage Regulator Pins (ENVREG and VCAP/VDDCORE)

The on-chip voltage regulator enable pin, ENVREG, must always be connected directly to either a supply voltage or to ground. Tying ENVREG to VDD enables the regulator, while tying it to ground disables the regulator. Refer to **Section 25.3 "On-Chip Voltage Regulator"** for details on connecting and using the on-chip regulator.

When the regulator is enabled, a low-ESR (< 5Ω) capacitor is required on the VCAP/VDDCORE pin to stabilize the voltage regulator output voltage. The VCAP/VDDCORE pin must not be connected to VDD and must use a capacitor of 10 µF connected to ground. The type can be ceramic or tantalum. Suitable examples of capacitors are shown in Table 2-1. Capacitors with equivalent specifications can be used.

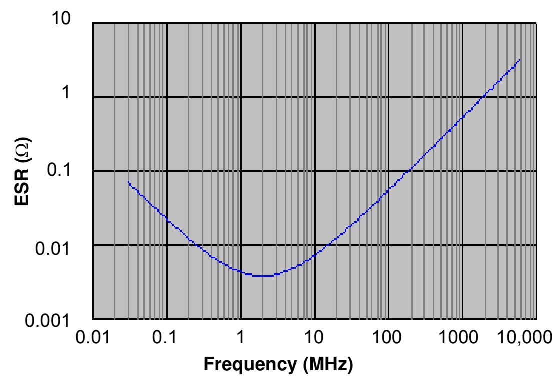
Designers may use Figure 2-3 to evaluate ESR equivalence of candidate devices.

It is recommended that the trace length not exceed 0.25 inch (6 mm). Refer to **Section 28.0 "Electrical Characteristics"** for additional information.

When the regulator is disabled, the VCAP/VDDCORE pin must be tied to a voltage supply at the VDDCORE level. Refer to **Section 28.0 "Electrical Characteristics"** for information on VDD and VDDCORE.

Note that the "LF" versions of some low pin count PIC18FJ parts (e.g., the PIC18LF45J10) do not have the ENVREG pin. These devices are provided with the voltage regulator permanently disabled; they must always be provided with a supply voltage on the VDDCORE pin.

**FIGURE 2-3: FREQUENCY vs. ESR PERFORMANCE FOR SUGGESTED VCAP**




**Note:** Typical data measurement at 25°C, 0V DC bias.

**TABLE 2-1: SUITABLE CAPACITOR EQUIVALENTS**

| Make | Part # | Nominal Capacitance | Base Tolerance | Rated Voltage | Temp. Range |
|---|---|---|---|---|---|
| TDK | C3216X7R1C106K | 10 µF | ±10% | 16V | -55 to 125ºC |
| TDK | C3216X5R1C106K | 10 µF | ±10% | 16V | -55 to 85ºC |
| Panasonic | ECJ-3YX1C106K | 10 µF | ±10% | 16V | -55 to 125ºC |
| Panasonic | ECJ-4YB1C106K | 10 µF | ±10% | 16V | -55 to 85ºC |
| Murata | GRM32DR71C106KA01L | 10 µF | ±10% | 16V | -55 to 125ºC |
| Murata | GRM31CR61C106KC31L | 10 µF | ±10% | 16V | -55 to 85ºC |

### 2.4.1 CONSIDERATIONS FOR CERAMIC CAPACITORS

In recent years, large value, low-voltage, surface-mount ceramic capacitors have become very cost effective in sizes up to a few tens of microfarad. The low-ESR, small physical size and other properties make ceramic capacitors very attractive in many types of applications.

Ceramic capacitors are suitable for use with the VDDCORE voltage regulator of this microcontroller. However, some care is needed in selecting the capacitor to ensure that it maintains sufficient capacitance over the intended operating range of the application.

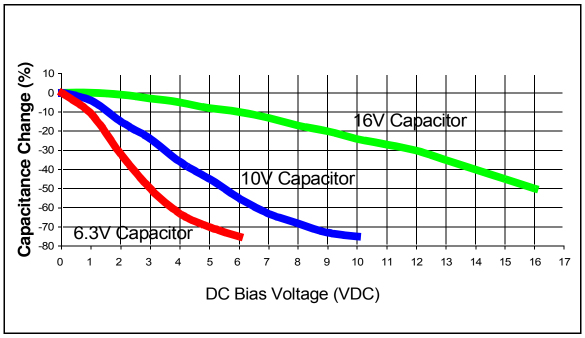
Typical low-cost, 10 µF ceramic capacitors are available in X5R, X7R and Y5V dielectric ratings (other types are also available, but are less common). The initial tolerance specifications for these types of capacitors are often specified as ±10% to ±20% (X5R and X7R), or -20%/+80% (Y5V). However, the effective capacitance that these capacitors provide in an application circuit will also vary based on additional factors, such as the applied DC bias voltage and the temperature. The total in-circuit tolerance is, therefore, much wider than the initial tolerance specification.

The X5R and X7R capacitors typically exhibit satisfactory temperature stability (ex: ±15% over a wide temperature range, but consult the manufacturer's data sheets for exact specifications). However, Y5V capacitors typically have extreme temperature tolerance specifications of +22%/-82%. Due to the extreme temperature tolerance, a 10 µF nominal rated Y5V type capacitor may not deliver enough total capacitance to meet minimum VDDCORE voltage regulator stability and transient response requirements. Therefore, Y5V capacitors are not recommended for use with the VDDCORE regulator if the application must operate over a wide temperature range.

In addition to temperature tolerance, the effective capacitance of large value ceramic capacitors can vary substantially, based on the amount of DC voltage applied to the capacitor. This effect can be very significant, but is often overlooked or is not always documented.

A typical DC bias voltage vs. capacitance graph for X7R type and Y5V type capacitors is shown in Figure 2-4.

**FIGURE 2-4: DC BIAS VOLTAGE vs. CAPACITANCE CHARACTERISTICS**




When selecting a ceramic capacitor to be used with the VDDCORE voltage regulator, it is suggested to select a high-voltage rating, so that the operating voltage is a small percentage of the maximum rated capacitor voltage. For example, choose a ceramic capacitor rated at 16V for the 2.5V VDDCORE voltage. Suggested capacitors are shown in Table 2-1.

## 2.5 ICSP Pins

The PGC and PGD pins are used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor is recommended, with the value in the range of a few tens of ohms, not to exceed 100Ω.

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

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

For more information on available Microchip development tools connection requirements, refer to **Section 27.0 “Development Support”**.

## 2.6 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to **Section 3.0 "Oscillator Configurations"** for details).

The oscillator circuit should be placed on the same side of the board as the device. Place the oscillator circuit close to the respective oscillator pins with no more than 0.5 inch (12 mm) between the circuit components and the pins. The load capacitors should be placed next to the oscillator itself, on the same side of the board.

Use a grounded copper pour around the oscillator circuit to isolate it from surrounding circuits. The grounded copper pour should be routed directly to the MCU ground. Do not run any signal traces or power traces inside the ground pour. Also, if using a two-sided board, avoid any traces on the other side of the board where the crystal is placed.

Layout suggestions are shown in Figure 2-5. In-line packages may be handled with a single-sided layout that completely encompasses the oscillator pins. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

In planning the application's routing and I/O assignments, ensure that adjacent port pins, and other signals in close proximity to the oscillator, are benign (i.e., free of high frequencies, short rise and fall times, and other similar noise).

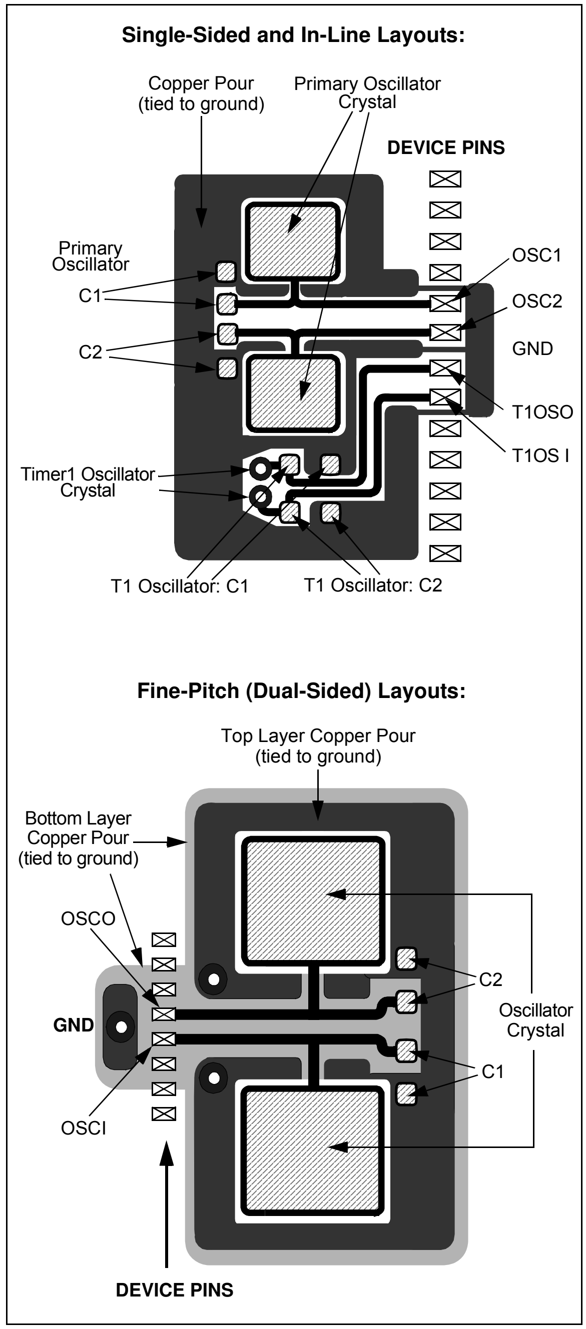
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate web site (www.microchip.com):

- *AN826, "Crystal Oscillator Basics and Crystal Selection for rfPIC™ and PICmicro® Devices"*
- *AN849, "Basic PICmicro® Oscillator Design"*
- *AN943, "Practical PICmicro® Oscillator Analysis and Design"*
- *AN949, "Making Your Oscillator Work"*

## 2.7 Unused I/Os

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

**FIGURE 2-5: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT**



**NOTES:**

# 3.0 OSCILLATOR CONFIGURATIONS

## 3.1 Oscillator Types

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

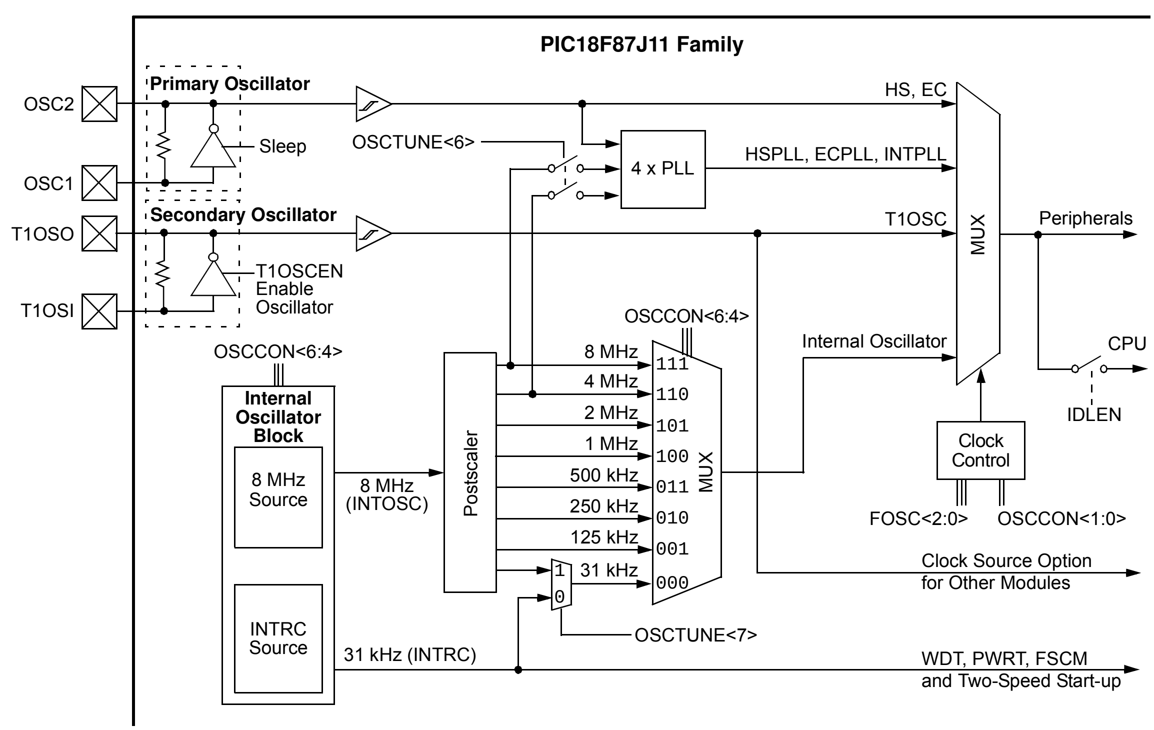
1. HS — High-Speed Crystal/Resonator
2. HSPLL — High-Speed Crystal/Resonator with Software PLL Control
3. EC — External Clock with FOSC/4 Output
4. ECPLL — External Clock with Software PLL Control
5. INTIO1 — Internal Oscillator Block with FOSC/4 Output on RA6 and I/O on RA7
6. INTIO2 — Internal Oscillator 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 FOSC<2:0> 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 3-1.

**FIGURE 3-1: PIC18F87J11 FAMILY CLOCK DIAGRAM**



## 3.2 Control Registers

The OSCCON register (Register 3-1) 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 3-2) 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 3.4.3 "PLL Frequency Multiplier"**).

### REGISTER 3-1: OSCCON: OSCILLATOR CONTROL REGISTER[1]

| R/W-0 | R/W-1 | R/W-1 | R/W-0 | R[2] | U-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IDLEN | IRCF2[3] | IRCF1[3] | IRCF0[3] | OSTS | — | SCS1[5] | SCS0[5] |
| 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 **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 **IRCF<2:0>:** 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:** 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 **SCS<1:0>:** System Clock Select bits[5]
11 = Internal oscillator block
10 = Primary oscillator
01 = Timer1 oscillator
00 = Default primary oscillator (as defined by the FOSC<2:0> Configuration bits)

**Note 1:** Default (legacy) SFR at this address, available when WDTCON<4> = 0.
**2:** Reset state depends on the 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:** The source is selected by the INTSRC bit (OSCTUNE<7>), see text.
**5:** Modifying these bits will cause an immediate clock source switch.

**REGISTER 3-2: OSCTUNE: OSCILLATOR TUNING REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| INTSRC | PLLEN | TUN5 | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 |
| 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 **INTSRC:** Internal Oscillator Low-Frequency Source Select bit
1 = 31.25 kHz device clock derived from 8 MHz INTOSC source (divide-by-256 enabled)
0 = 31 kHz device clock derived from INTRC 31 kHz oscillator

bit 6 **PLLEN:** Frequency Multiplier PLL Enable bit
1 = PLL is enabled
0 = PLL is disabled

bit 5-0 **TUN<5:0>:** Fast RC Oscillator (INTOSC) Frequency Tuning bits
011111 = Maximum frequency
• •
• •
000001
000000 = Center frequency. Fast RC Oscillator is running at the calibrated frequency.
111111
• •
• •
100000 = Minimum frequency

## 3.3 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 FOSC<2:0> 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 FOSCx Configuration bits. The details of these modes are covered in **Section 3.4 “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 14.0 “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 3.5 “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.

### 3.3.1 CLOCK SOURCE SELECTION

The System Clock Select bits, SCS<1:0> (OSCCON<1:0>), select the clock source. The available clock sources are the primary clock defined by the FOSC<2:0> 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 4.0 "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. |

#### 3.3.1.1 System Clock Selection and Device Resets

Since the SCSx bits are cleared on all forms of Reset, this means the primary oscillator defined by the FOSC<2:0> 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 sources (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 IRCF<2:0> 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 FOSCx 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 SCS<1:0> bits at any given time.

### 3.3.2 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 4.1.2 "Entering Power-Managed Modes"**.

## 3.4 External Oscillator Modes

### 3.4.1 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 3-2 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 3-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS**

| Typical Capacitor Values Used: | | | |
|---|---|---|---|
| Mode | Freq. | OSC1 | OSC2 |
| HS | 8.0 MHz<br>16.0 MHz | 27 pF<br>22 pF | 27 pF<br>22 pF |

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 3-2 for additional information.

**TABLE 3-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR**

| Osc Type | Crystal Freq. | Typical Capacitor Values Tested: | |
|---|---|---|---|
| | | C1 | C2 |
| HS | 4 MHz | 27 pF | 27 pF |
| | 8 MHz | 22 pF | 22 pF |
| | 20 MHz | 15 pF | 15 pF |

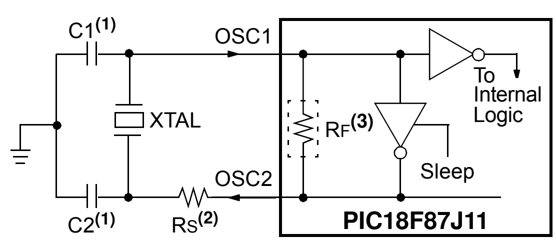
Capacitor values are for design guidance only.

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 3-1 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 3-2: CRYSTAL/CERAMIC RESONATOR OPERATION (HS OR HSPLL CONFIGURATION)**




**Note 1:** See Table 3-1 and Table 3-2 for initial values of C1 and C2.

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

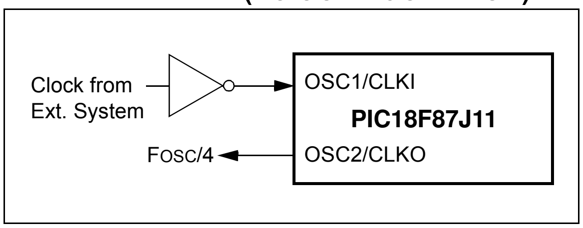
**3:** RF varies with the oscillator mode chosen.

### 3.4.2 EXTERNAL CLOCK INPUT (EC MODES)

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.

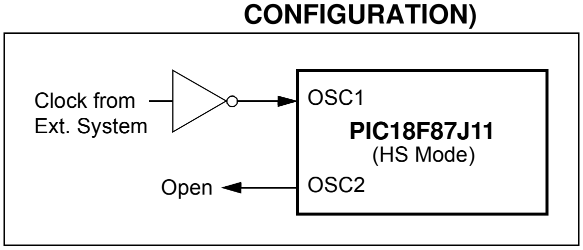
In the EC Oscillator mode, the oscillator frequency, divided by 4, is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 3-3 shows the pin connections for the EC Oscillator mode.

**FIGURE 3-3: 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 3-4. 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 3-4: EXTERNAL CLOCK INPUT OPERATION (HS OSC CONFIGURATION)**




### 3.4.3 PLL FREQUENCY MULTIPLIER

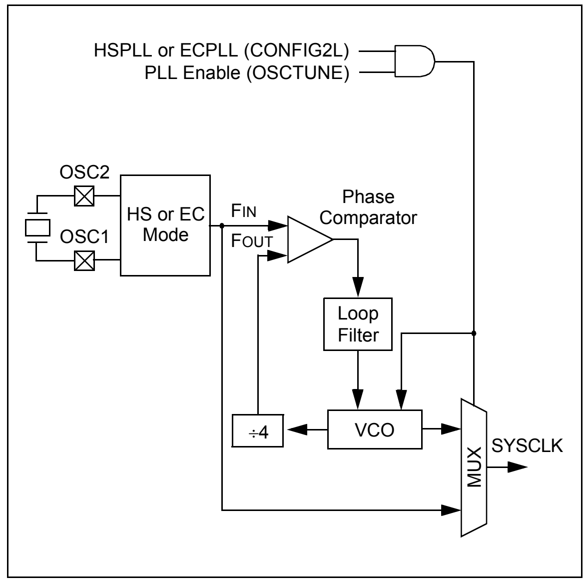
A Phase Locked Loop (PLL) circuit is provided as an option for users who want to use a lower frequency oscillator circuit, or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with the External Memory Interface (EMI) due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

#### 3.4.3.1 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 FOSC<2:0> 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 3-5: PLL BLOCK DIAGRAM**




#### 3.4.3.2 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 3.5.2 "INTPLL Modes"**.

## 3.5 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 25.0 "Special Features of the CPU"**.

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

### 3.5.1 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 FOSCx Configuration bits, are available:

- In INTIO1 mode, the OSC2 pin outputs FOSC/4, while OSC1 functions as RA7 (see Figure 3-6) for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6 (see Figure 3-7), both for digital input and output.

**FIGURE 3-6: INTIO1 OSCILLATOR MODE**

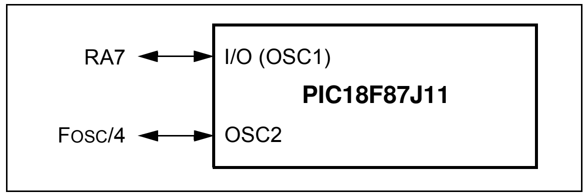
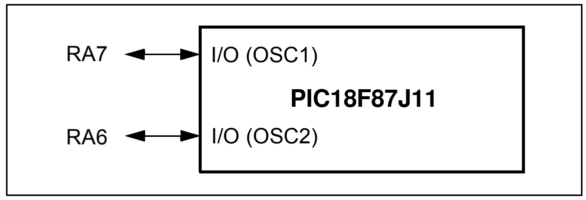



**FIGURE 3-7: INTIO2 OSCILLATOR MODE**




### 3.5.2 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 (FOSC<2:0> = `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 3-6).
- 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 3-7).

### 3.5.3 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 TUN<5:0> (OSCTUNE<5:0>) in the OSCTUNE register (Register 3-2).

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.

### 3.5.4 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.

#### 3.5.4.1 Compensating with the EUSARTx

An adjustment may be required when the EUSARTx 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.

#### 3.5.4.2 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.

#### 3.5.4.3 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.

## 3.6 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 3-3). Setting the ROON bit (REFOCON<7>) makes the clock signal available on the REFO (RE3) pin. The RODIV<3:0> 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 3-3: REFOCON: REFERENCE OSCILLATOR CONTROL REGISTER

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ROON | — | ROSSLP | ROSEL[1] | RODIV3 | RODIV2 | RODIV1 | RODIV0 |
| 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 **ROON:** Reference Oscillator Output Enable bit
1 = Reference oscillator output is available on REFO pin
0 = Reference oscillator output is disabled

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **ROSSLP:** 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) is used as the base clock
0 = System clock is used as the base clock; base clock reflects any clock switching of the device

bit 3-0 **RODIV<3:0>:** 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 FOSCx Configuration bits to maintain clock output during Sleep mode.

## 3.7 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 25.2 "Watchdog Timer (WDT)"** through **Section 25.5 "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 28.2, DC Characteristics: Power-Down and Supply Current PIC18F87J11 Family (Industrial)**.

## 3.8 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 5.6 "Power-up Timer (PWRT)"**.

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up (Parameter 33, Table 28-13); 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, $T_{CSD}$ (Parameter 38, Table 28-13), following POR, while the controller becomes ready to execute instructions.

**TABLE 3-3: OSC1 AND OSC2 PIN STATES IN SLEEP MODE**

| Oscillator Mode | OSC1 Pin | OSC2 Pin |
|---|---|---|
| EC, ECPLL | Floating, pulled by external clock | At logic low (clock/4 output) |
| HS, HSPLL | Feedback inverter disabled at quiescent voltage level | Feedback inverter disabled at quiescent voltage level |
| INTOSC, INTPLL1/2 | I/O pin, RA6, direction controlled by TRISA<6> | I/O pin RA6, direction controlled by TRISA<7> |

**Note:** See **Section 5.0 "Reset"** for time-outs due to Sleep and $\overline{MCLR}$ Reset.

# 4.0 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® MCU devices, where all device clocks are stopped.

## 4.1 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 SCS<1:0> bits (OSCCON<1:0>) select the clock source. The individual modes, bit settings, clock sources and affected modules are summarized in Table 4-1.

### 4.1.1 CLOCK SOURCES

The SCS<1:0> bits allow the selection of one of three clock sources for power-managed modes. They are:

- The primary clock, as defined by the FOSC<2:0> Configuration bits
- The secondary clock (Timer1 oscillator)
- The internal oscillator

### 4.1.2 ENTERING POWER-MANAGED MODES

Switching from one power-managed mode to another begins by loading the OSCCON register. The SCS<1:0> 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 4.1.3 "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 4-1: POWER-MANAGED MODES**

| Mode | OSCCON<7,1:0> | | Module Clocking | | Available Clock and Oscillator Source |
|---|---|---|---|---|---|
| | IDLEN(1) | SCS<1:0> | CPU | Peripherals | |
| Sleep | 0 | N/A | Off | Off | None – All clocks are disabled |
| PRI_RUN | N/A | 10 | Clocked | Clocked | Primary – HS, EC, HSPLL, ECPLL, INTOSC oscillator; this is the normal, full-power execution mode |
| SEC_RUN | N/A | 01 | Clocked | Clocked | Secondary – Timer1 oscillator |
| RC_RUN | N/A | 11 | Clocked | Clocked | Internal oscillator block(2) |
| PRI_IDLE | 1 | 10 | Off | Clocked | Primary – HS, EC, HSPLL, ECPLL, INTOSC |
| SEC_IDLE | 1 | 01 | Off | Clocked | Secondary – Timer1 oscillator |
| RC_IDLE | 1 | 11 | Off | Clocked | Internal oscillator block(2) |

**Note 1:** IDLEN reflects its value when the SLEEP instruction is executed.
**2:** Includes the INTRC and INTOSC postcaler (internal oscillator block).

### 4.1.3 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.

| | |
|---|---|
| **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. |

### 4.1.4 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.

## 4.2 Run Modes

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

### 4.2.1 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 25.4 "Two-Speed Start-up"** for details). In this mode, the OSTS bit is set (see **Section 3.2 "Control Registers"**).

### 4.2.2 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 SCS<1:0> bits to '`01`'. The device clock source is switched to the Timer1 oscillator (see Figure 4-1). The primary oscillator is shut down, the T1RUN bit (T1CON<6>) is set and the OSTS bit is cleared.

| | |
|---|---|
| **Note:** | The Timer1 oscillator should already be running prior to entering SEC_RUN mode. If the T1OSCEN bit is not set when the SCS<1:0> 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. |

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 4-2). 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 SCSx bits are not affected by the wake-up; the Timer1 oscillator continues to run.

**FIGURE 4-1: TRANSITION TIMING FOR ENTRY TO SEC_RUN MODE**

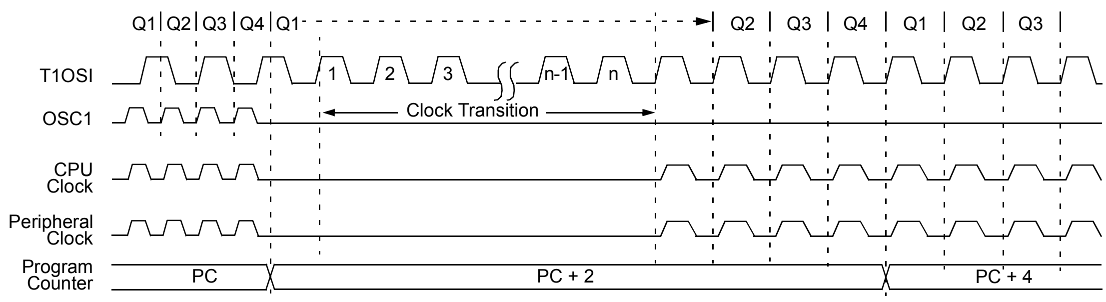
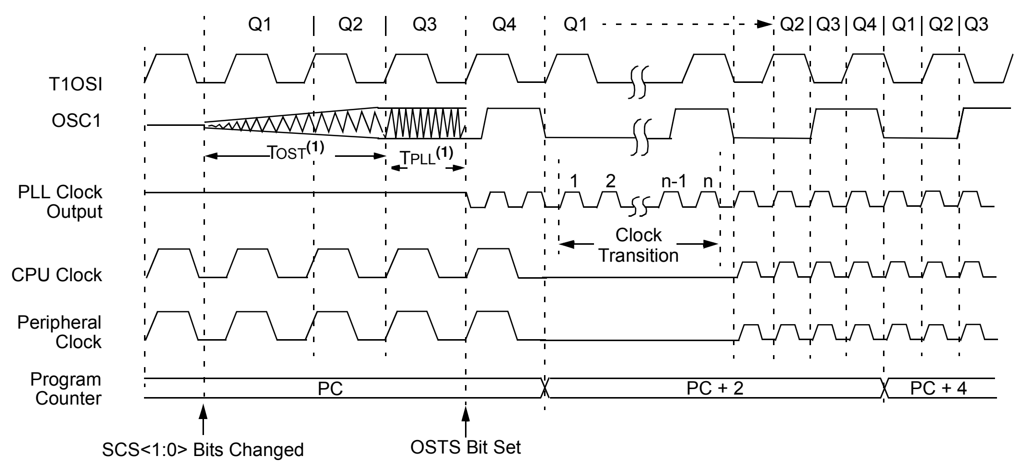



**FIGURE 4-2: TRANSITION TIMING FROM SEC_RUN MODE TO PRI_RUN MODE (HSPLL)**




**Note 1:** $T_{OST}$ = 1024 $T_{OSC}$; $T_{PLL}$ = 2 ms (approx). These intervals are not shown to scale.

### 4.2.3 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 4-3), 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 4-4). When the clock switch is complete, the OSTS bit is set and the primary clock is providing the device clock. The IDLEN and SCSx 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 4-3: TRANSITION TIMING TO RC_RUN MODE**

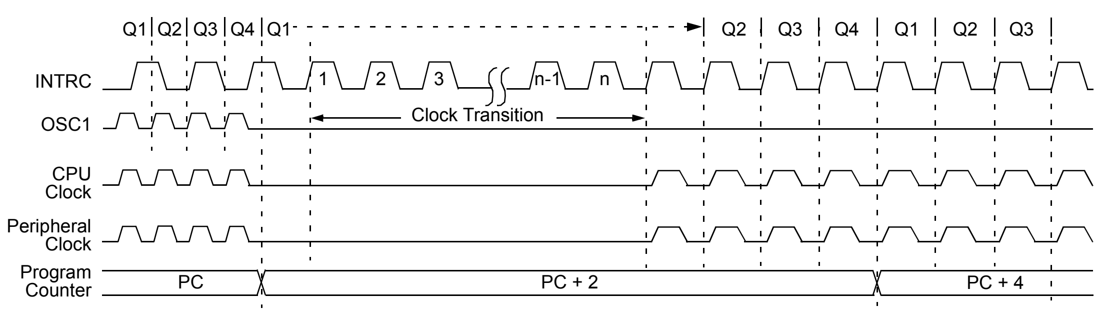
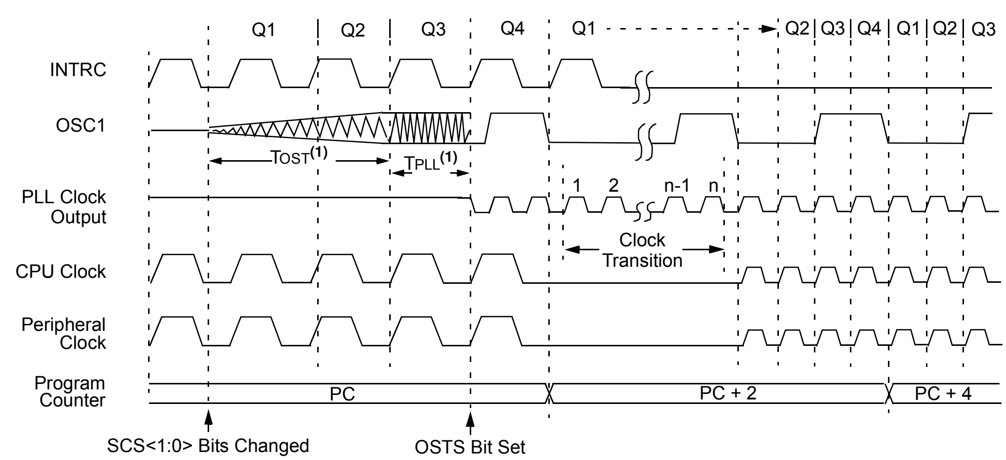



**FIGURE 4-4: TRANSITION TIMING FROM RC_RUN MODE TO PRI_RUN MODE**




**Note 1:** $T_{OST}$ = 1024 $T_{OSC}$; $T_{PLL}$ = 2 ms (approx). These intervals are not shown to scale.

## 4.3 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 4-5). All clock source status bits are cleared.

Entering 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 until the clock source selected by the SCS<1:0> bits becomes ready (see Figure 4-6), 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 25.0 "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 SCSx bits are not affected by the wake-up.

## 4.4 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 SCS<1:0> 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 $T_{CSD}$ (Parameter 38, Table 28-13) 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 SCSx 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 SCS<1:0> bits.

**FIGURE 4-5: TRANSITION TIMING FOR ENTRY TO SLEEP MODE**

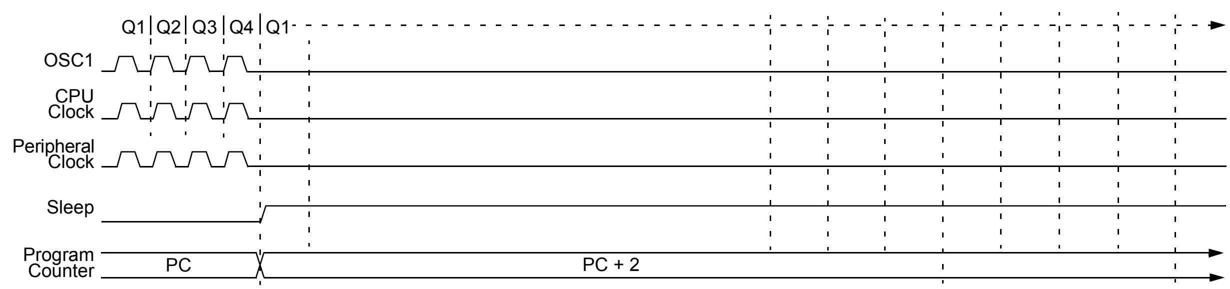
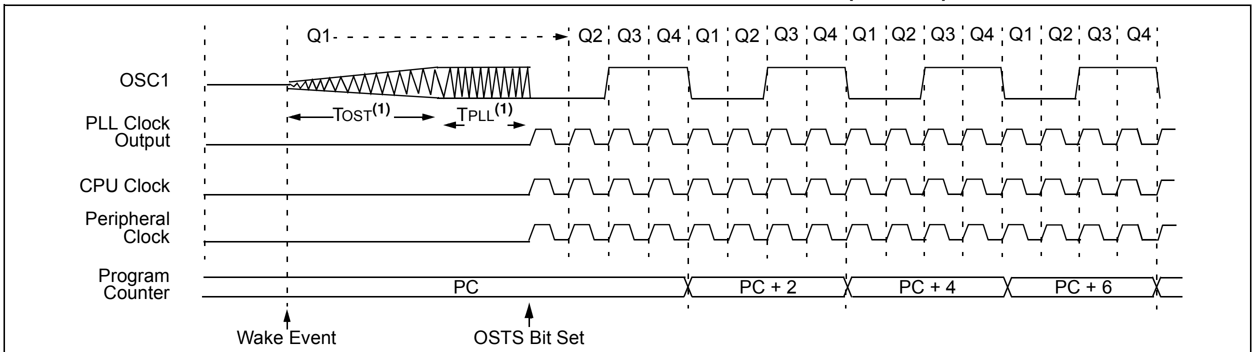



**FIGURE 4-6: TRANSITION TIMING FOR WAKE FROM SLEEP (HSPLL)**




**Note 1:** $T_{OST}$ = 1024 $T_{OSC}$; $T_{PLL}$ = 2 ms (approx). These intervals are not shown to scale.

### 4.4.1 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 SCSx 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 FOSC<1:0> Configuration bits. The OSTS bit remains set (see Figure 4-7).

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 SCSx bits are not affected by the wake-up (see Figure 4-8).

### 4.4.2 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 SCS<1:0> 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 SCSx bits are not affected by the wake-up; the Timer1 oscillator continues to run (see Figure 4-8).

| | |
|---|---|
| **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 4-7: TRANSITION TIMING FOR ENTRY TO IDLE MODE**

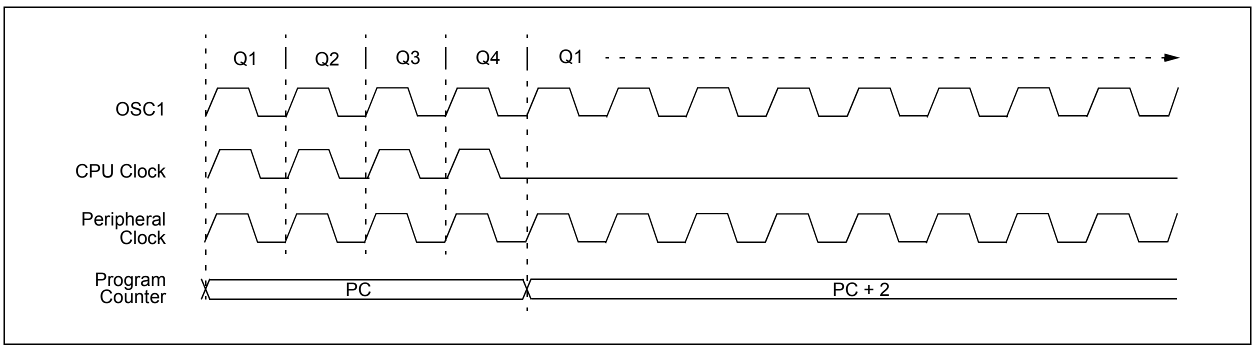
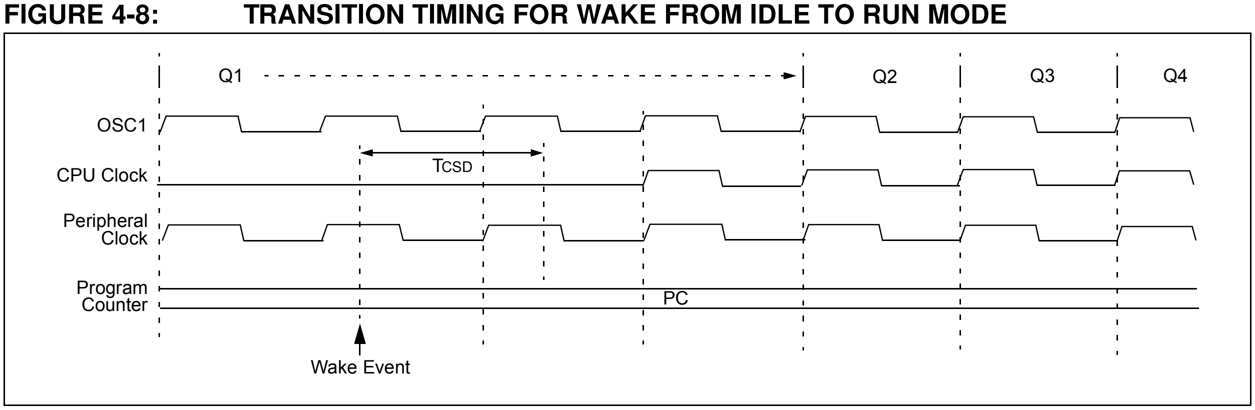



**FIGURE 4-8: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE**



#### 4.4.3 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 SCSx bits and execute SLEEP. When the clock source is switched to the INTOSC 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 $T_{CSD}$ following the wake event, the CPU begins executing code being clocked by the INTRC. The IDLEN and SCSx 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.

## 4.5 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 4.2 "Run Modes"**, **Section 4.3 "Sleep Mode"** and **Section 4.4 "Idle Modes"**).

#### 4.5.1 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 10.0 "Interrupts"**).

A fixed delay of interval, $T_{CSD}$, 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.

#### 4.5.2 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 4.2 "Run Modes"** and **Section 4.3 "Sleep Mode"**). If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see **Section 25.2 "Watchdog Timer (WDT)"**).

The Watchdog Timer and postscaler are cleared by one of the following events:

- Executing a SLEEP or CLRWDT instruction
- The loss of a currently selected clock source (if the Fail-Safe Clock Monitor is enabled)

#### 4.5.3 EXIT BY RESET

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

#### 4.5.4 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
- 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, $T_{CSD}$, 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.

**NOTES:**

## 5.0 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 6.1.6.4 "Stack Full and Underflow Resets"**. WDT Resets are covered in **Section 25.2 "Watchdog Timer (WDT)"**.

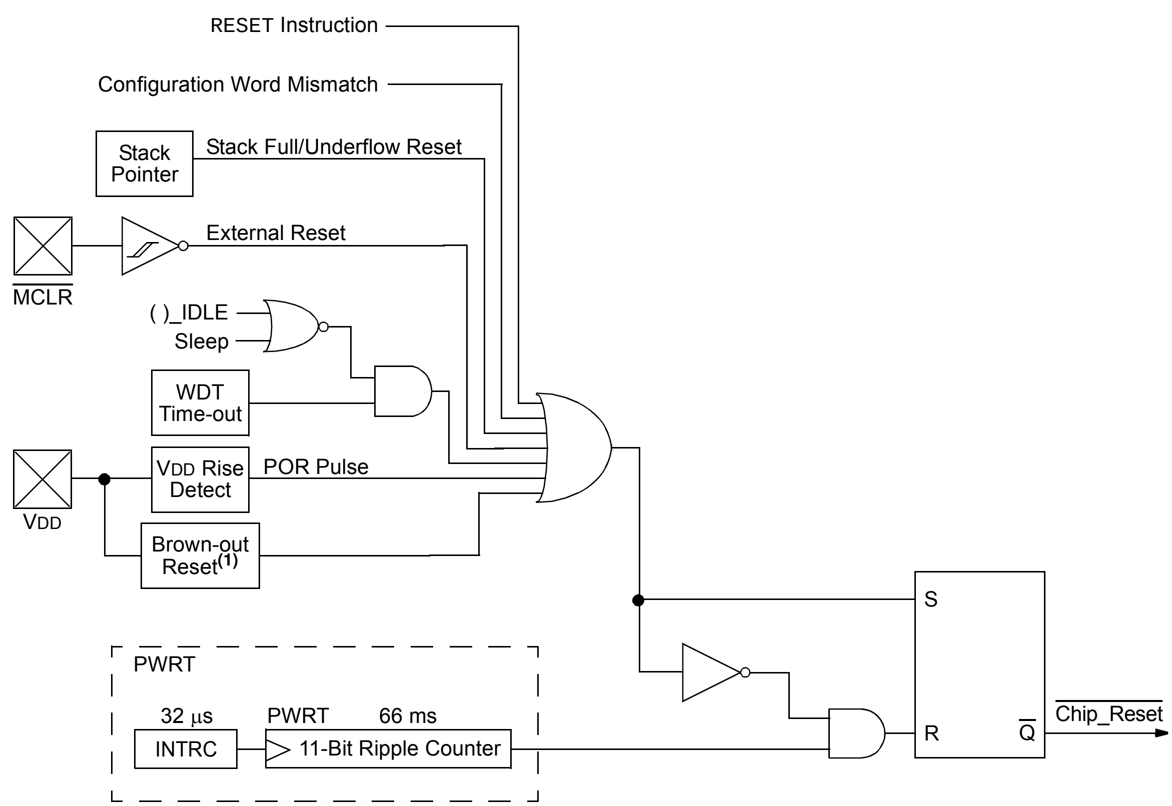
A simplified block diagram of the on-chip Reset circuit is shown in Figure 5-1.

## 5.1 RCON Register

Device Reset events are tracked through the RCON register (Register 5-1). 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 5.7 "Reset State of Registers"**.

The RCON register also has a control bit for setting interrupt priority (IPEN). Interrupt priority is discussed in **Section 10.0 "Interrupts"**.

**FIGURE 5-1: 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 5-1: RCON: RESET CONTROL REGISTER

| R/W-0 | U-0 | R/W-1 | R/W-1 | R-1 | R-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | — | $\overline{\text{CM}}$ | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |
| 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 **IPEN:** Interrupt Priority Enable bit
- `1` = Enables priority levels on interrupts
- `0` = Disables priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6 **Unimplemented:** Read as '`0`'

bit 5 **$\overline{\text{CM}}$:** 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)

bit 4 **$\overline{\text{RI}}$:** `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)

bit 3 **$\overline{\text{TO}}$:** Watchdog Time-out Flag bit
- `1` = Set by power-up, `CLRWDT` instruction or `SLEEP` instruction
- `0` = A WDT time-out occurred

bit 2 **$\overline{\text{PD}}$:** Power-Down Detection Flag bit
- `1` = Set by power-up or by the `CLRWDT` instruction
- `0` = Set by execution of the `SLEEP` instruction

bit 1 **$\overline{\text{POR}}$:** 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)

bit 0 **$\overline{\text{BOR}}$:** 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 $\overline{\text{POR}}$ 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, $\overline{\text{BOR}}$ remains at '`0`' at all times. See **Section 5.4.1 "Detecting BOR"** for more information.

**3:** Brown-out Reset is said to have occurred when $\overline{\text{BOR}}$ is '`0`' and $\overline{\text{POR}}$ is '`1`' (assuming that $\overline{\text{POR}}$ was set to '`1`' by software immediately after a Power-on Reset).

## 5.2 Master Clear ($\overline{\text{MCLR}}$)

The $\overline{\text{MCLR}}$ 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 $\overline{\text{MCLR}}$ Reset path which detects and ignores small pulses.

The $\overline{\text{MCLR}}$ pin is not driven low by any internal Resets, including the WDT.

## 5.3 Power-on Reset (POR)

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

To take advantage of the POR circuitry, tie the $\overline{\text{MCLR}}$ pin through a resistor (1 kΩ to 10 kΩ) to $V_{DD}$. This will eliminate external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for $V_{DD}$ is specified (Parameter D004). For a slow rise time, see Figure 5-2.

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.

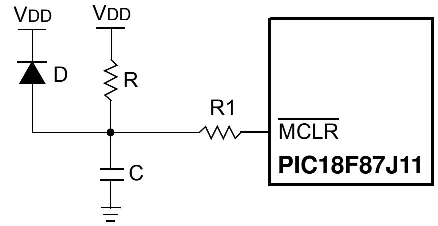
Power-on Reset events are captured by the $\overline{\text{POR}}$ bit (RCON<1>). The state of the bit is set to '0' whenever a Power-on Reset occurs; it does not change for any other Reset event. $\overline{\text{POR}}$ is not reset to '1' by any hardware event. To capture multiple events, the user manually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

## 5.4 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 $V_{DD}$). Any drop of $V_{DD}$ below $V_{BOR}$ (Parameter D005) for greater than time, $T_{BOR}$, will reset the device. A Reset may or may not occur if $V_{DD}$ falls below $V_{BOR}$ for less than $T_{BOR}$. The chip will remain in Brown-out Reset until $V_{DD}$ rises above $V_{BOR}$.

Once a Brown-out Reset has occurred, the Power-up Timer will keep the chip in Reset for $T_{PWRT}$ (Parameter 33). If $V_{DD}$ drops below $V_{BOR}$ 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 $V_{DD}$ rises above $V_{BOR}$, the Power-up Timer will execute the additional time delay.

**FIGURE 5-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW $V_{DD}$ POWER-UP)**




**Note 1:** External Power-on Reset circuit is required only if the $V_{DD}$ power-up slope is too slow. The diode, D, helps discharge the capacitor quickly when $V_{DD}$ 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 $\overline{\text{MCLR}}$ from external capacitor, C, in the event of $\overline{\text{MCLR}}$/$V_{PP}$ pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

### 5.4.1 DETECTING BOR

The $\overline{\text{BOR}}$ 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 $\overline{\text{BOR}}$ alone. A more reliable method is to simultaneously check the state of both $\overline{\text{POR}}$ and $\overline{\text{BOR}}$. This assumes that the $\overline{\text{POR}}$ bit is reset to '1' in software immediately after any Power-on Reset event. If $\overline{\text{BOR}}$ is '0' while $\overline{\text{POR}}$ 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 $\overline{\text{BOR}}$ bit cannot be used to determine a Brown-out Reset event. The $\overline{\text{BOR}}$ bit is still cleared by a Power-on Reset event.

## 5.5 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 $\overline{\text{CM}}$ 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, 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.

## 5.6 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 x 32 μs = 66 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.

### 5.6.1 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 5-3, Figure 5-4, Figure 5-5 and Figure 5-6 all depict time-out sequences on power-up with the Power-up Timer enabled.

Since the time-outs occur from the POR pulse and if MCLR is kept low long enough, the PWRT will expire. Bringing MCLR high will begin execution immediately (Figure 5-5). This is useful for testing purposes, or to synchronize more than one PIC18FXXXX device operating in parallel.

**TABLE 5-1: TIME-OUT IN VARIOUS SITUATIONS**

| Oscillator Configuration | Power-up(2) and Brown-out Reset | | Exit from Power-Managed Mode |
|---|---|---|---|
| | PWRTEN = 0 | PWRTEN = 1 | |
| HSPLL | 66 ms(1) + 1024 TOSC + 2 ms(2) | 1024 TOSC + 2 ms(2) | 1024 TOSC + 2 ms(2) |
| HS, XT, LP | 66 ms(1) + 1024 TOSC | 1024 TOSC | 1024 TOSC |
| EC, ECIO | 66 ms(1) | — | — |
| RC, RCIO | 66 ms(1) | — | — |
| INTIO1, INTIO2 | 66 ms(1) | — | — |

**Note 1:** 66 ms (65.5 ms) is the nominal Power-up Timer (PWRT) delay.
**2:** 2 ms is the nominal time required for the PLL to lock.

**FIGURE 5-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD, VDD RISE < TPWRT)**

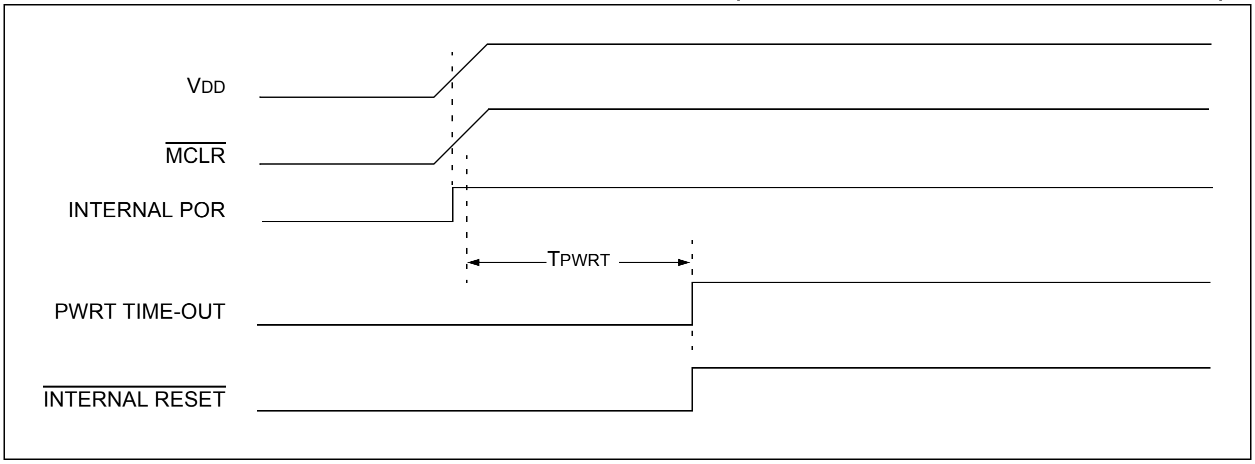


**FIGURE 5-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1**

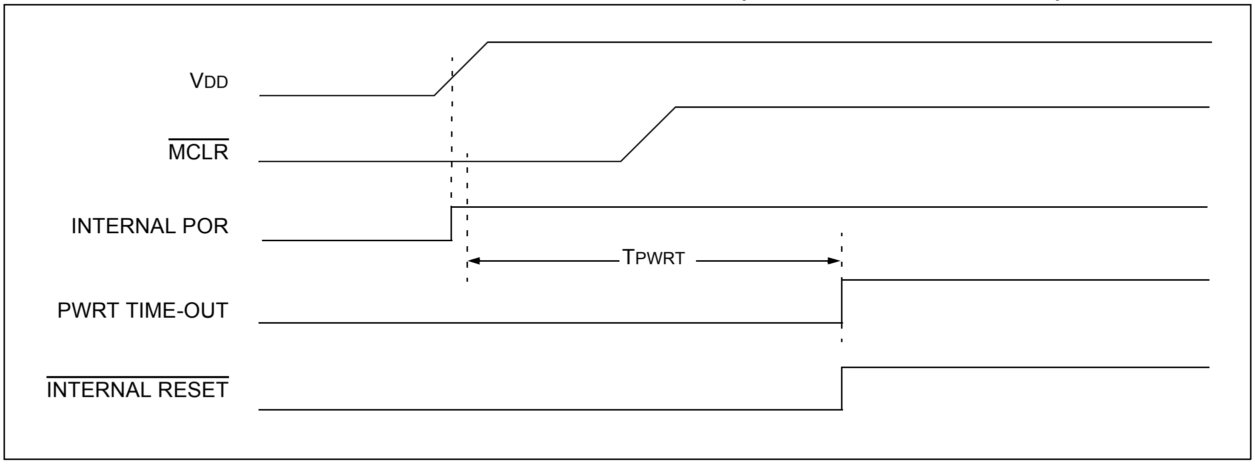



**FIGURE 5-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2**

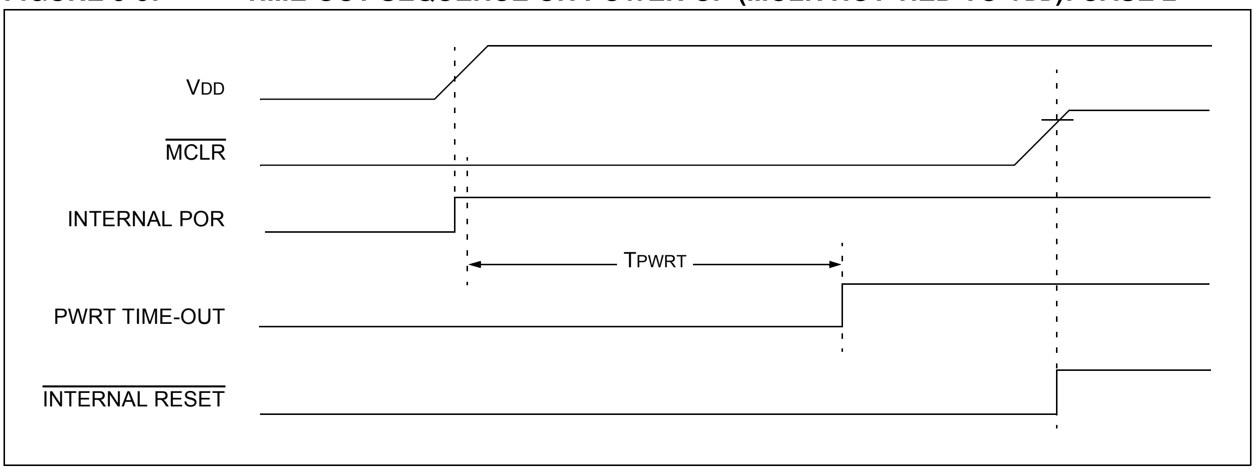
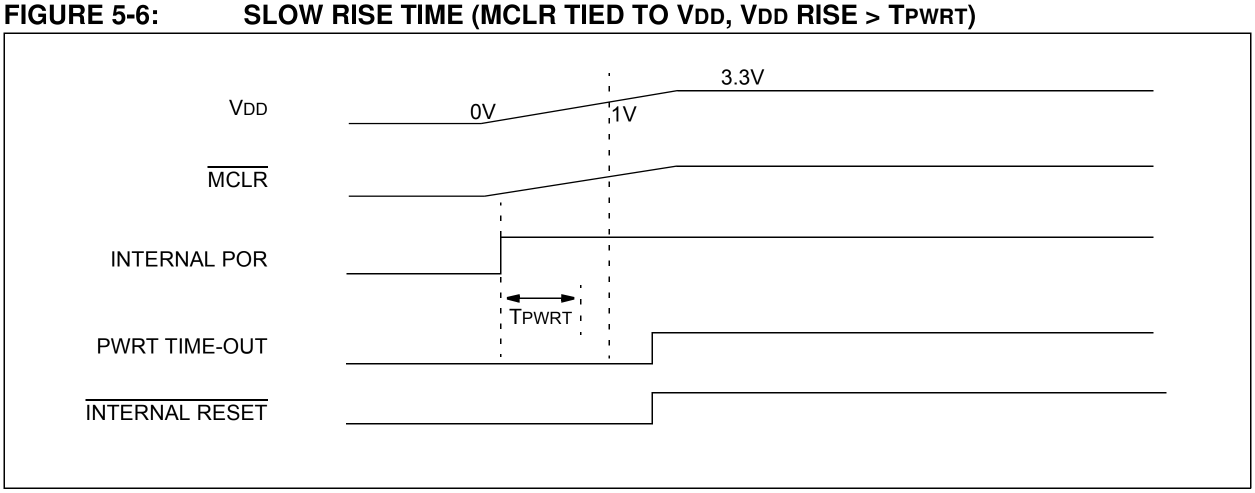



**FIGURE 5-6: SLOW RISE TIME (MCLR TIED TO VDD, VDD RISE > TPWRT)**



## 5.7 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 ($\overline{CM}$, $\overline{RI}$, $\overline{TO}$, $\overline{PD}$, $\overline{POR}$ and $\overline{BOR}$) are set or cleared differently in different Reset situations, as indicated in Table 5-2. These bits are used in software to determine the nature of the Reset.

Table 5-3 describes the Reset states for all of the Special Function Registers (SFRs). These are categorized by Power-on and Brown-out Resets, Master Clear and WDT Resets, and WDT wake-ups.

**TABLE 5-2: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER**

| Condition | Program Counter[(1)] | RCON Register | | | | | | STKPTR Register | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | STKFUL | STKUNF |
| Power-on Reset | 0000h | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| RESET instruction | 0000h | u | 0 | u | u | u | u | u | u |
| Brown-out Reset | 0000h | 1 | 1 | 1 | 1 | u | 0 | u | u |
| Configuration Mismatch Reset | 0000h | 0 | u | u | u | u | u | u | u |
| $\overline{MCLR}$ Reset during power-managed Run modes | 0000h | u | u | 1 | u | u | u | u | u |
| $\overline{MCLR}$ Reset during power-managed Idle modes and Sleep mode | 0000h | u | u | 1 | 0 | u | u | u | u |
| $\overline{MCLR}$ Reset during full-power execution | 0000h | u | u | u | u | u | u | u | u |
| Stack Full Reset (STVREN = 1) | 0000h | u | u | u | u | u | u | 1 | u |
| Stack Underflow Reset (STVREN = 1) | 0000h | u | u | u | u | u | u | u | 1 |
| Stack Underflow Error (not an actual Reset, STVREN = 0) | 0000h | u | u | u | u | u | u | u | 1 |
| WDT time-out during full-power or power-managed Run modes | 0000h | u | u | 0 | u | u | u | u | u |
| WDT time-out during power-managed Idle or Sleep modes | PC + 2 | u | u | 0 | 0 | u | u | u | u |
| Interrupt exit from power-managed modes | PC + 2 | u | u | u | 0 | u | u | u | u |

**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 5-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS[4]**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| TOSU | PIC18F6XJ1X | PIC18F8XJ1X | ---0 0000 | ---0 0000 | ---0 uuuu[1] |
| TOSH | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu[1] |
| TOSL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu[1] |
| STKPTR | PIC18F6XJ1X | PIC18F8XJ1X | 00-0 0000 | uu-0 0000 | uu-u uuuu[1] |
| PCLATU | PIC18F6XJ1X | PIC18F8XJ1X | ---0 0000 | ---0 0000 | ---u uuuu |
| PCLATH | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | PC + 2[2] |
| TBLPTRU | PIC18F6XJ1X | PIC18F8XJ1X | --00 0000 | --00 0000 | --uu uuuu |
| TBLPTRH | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| TBLPTRL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| TABLAT | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PRODH | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PRODL | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INTCON | PIC18F6XJ1X | PIC18F8XJ1X | 0000 000x | 0000 000u | uuuu uuuu[3] |
| INTCON2 | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu[3] |
| INTCON3 | PIC18F6XJ1X | PIC18F8XJ1X | 1100 0000 | 1100 0000 | uuuu uuuu[3] |
| INDF0 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| POSTINC0 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| POSTDEC0 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| PREINC0 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| PLUSW0 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| FSR0H | PIC18F6XJ1X | PIC18F8XJ1X | ---- xxxx | ---- 0000 | ---- uuuu |
| FSR0L | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| WREG | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INDF1 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| POSTINC1 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| POSTDEC1 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| PREINC1 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| PLUSW1 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| FSR1H | PIC18F6XJ1X | PIC18F8XJ1X | ---- xxxx | ---- 0000 | ---- uuuu |
| FSR1L | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| BSR | PIC18F6XJ1X | PIC18F8XJ1X | ---- 0000 | ---- 0000 | ---- uuuu |

**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 effected (to cause wake-up).

**4:** See Table 5-2 for Reset value for specific conditions.

**TABLE 5-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS(4) (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| INDF2 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| POSTINC2 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| POSTDEC2 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| PREINC2 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| PLUSW2 | PIC18F6XJ1X | PIC18F8XJ1X | N/A | N/A | N/A |
| FSR2H | PIC18F6XJ1X | PIC18F8XJ1X | `---- xxxx` | `---- 0000` | `---- uuuu` |
| FSR2L | PIC18F6XJ1X | PIC18F8XJ1X | `xxxx xxxx` | `uuuu uuuu` | `uuuu uuuu` |
| STATUS | PIC18F6XJ1X | PIC18F8XJ1X | `---x xxxx` | `---u uuuu` | `---u uuuu` |
| TMR0H | PIC18F6XJ1X | PIC18F8XJ1X | `0000 0000` | `0000 0000` | `uuuu uuuu` |
| TMR0L | PIC18F6XJ1X | PIC18F8XJ1X | `xxxx xxxx` | `uuuu uuuu` | `uuuu uuuu` |
| T0CON | PIC18F6XJ1X | PIC18F8XJ1X | `1111 1111` | `1111 1111` | `uuuu uuuu` |
| OSCCON | PIC18F6XJ1X | PIC18F8XJ1X | `0110 q100` | `0110 q100` | `0110 q10u` |
| REFOCON | PIC18F6XJ1X | PIC18F8XJ1X | `0-00 0000` | `u-uu uuuu` | `u-uu uuuu` |
| CM1CON | PIC18F6XJ1X | PIC18F8XJ1X | `0001 1111` | `0001 1111` | `uuuu uuuu` |
| CM2CON | PIC18F6XJ1X | PIC18F8XJ1X | `0001 1111` | `0001 1111` | `uuuu uuuu` |
| RCON(4) | PIC18F6XJ1X | PIC18F8XJ1X | `0-11 1100` | `0-qq qquu` | `u-qq qquu` |
| TMR1H | PIC18F6XJ1X | PIC18F8XJ1X | `xxxx xxxx` | `uuuu uuuu` | `uuuu uuuu` |
| ODCON1 | PIC18F6XJ1X | PIC18F8XJ1X | `---0 0000` | `---u uuuu` | `---u uuuu` |
| TMR1L | PIC18F6XJ1X | PIC18F8XJ1X | `xxxx xxxx` | `uuuu uuuu` | `uuuu uuuu` |
| ODCON2 | PIC18F6XJ1X | PIC18F8XJ1X | `---- --00` | `---- --uu` | `---- --uu` |
| T1CON | PIC18F6XJ1X | PIC18F8XJ1X | `0000 0000` | `u0uu uuuu` | `uuuu uuuu` |
| ODCON3 | PIC18F6XJ1X | PIC18F8XJ1X | `---- --00` | `---- --uu` | `---- --uu` |
| TMR2 | PIC18F6XJ1X | PIC18F8XJ1X | `0000 0000` | `0000 0000` | `uuuu uuuu` |
| PADCFG1 | PIC18F6XJ1X | PIC18F8XJ1X | `---- ---0` | `---- ---u` | `---- ---u` |
| PR2 | PIC18F6XJ1X | PIC18F8XJ1X | `1111 1111` | `1111 1111` | `1111 1111` |
| MEMCON | PIC18F6XJ1X | PIC18F8XJ1X | `0-00 --00` | `0-00 --00` | `u-uu --uu` |
| T2CON | PIC18F6XJ1X | PIC18F8XJ1X | `-000 0000` | `-000 0000` | `-uuu uuuu` |
| SSP1BUF | PIC18F6XJ1X | PIC18F8XJ1X | `xxxx xxxx` | `uuuu uuuu` | `uuuu uuuu` |
| SSP1ADD | PIC18F6XJ1X | PIC18F8XJ1X | `0000 0000` | `0000 0000` | `uuuu uuuu` |
| SSP1MSK | PIC18F6XJ1X | PIC18F8XJ1X | `1111 1111` | `uuuu uuuu` | `uuuu uuuu` |
| SSP1STAT | PIC18F6XJ1X | PIC18F8XJ1X | `0000 0000` | `0000 0000` | `uuuu uuuu` |
| SSP1CON1 | PIC18F6XJ1X | PIC18F8XJ1X | `0000 0000` | `0000 0000` | `uuuu uuuu` |
| SSP1CON2 | PIC18F6XJ1X | PIC18F8XJ1X | `0000 0000` | `0000 0000` | `uuuu uuuu` |

**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 effected (to cause wake-up).

**4:** See Table 5-2 for Reset value for specific conditions.

**TABLE 5-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS[4] (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| ADRESH | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADRESL | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADCON0 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| ADCON1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| ANCON0 | PIC18F6XJ1X | PIC18F8XJ1X | 00-0 0000 | uu-u uuuu | uu-u uuuu |
| ANCON1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | uuuu uuuu | uuuu uuuu |
| WDTCON | PIC18F6XJ1X | PIC18F8XJ1X | 0x-0 ---0 | 0x-u ---0 | ux-u ---u |
| ECCP1AS | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP1DEL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| CCPR1H | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR1L | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP1CON | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP2AS | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP2DEL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| CCPR2H | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR2L | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP2CON | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP3AS | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP3DEL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| CCPR3H | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR3L | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP3CON | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| SPBRG1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| RCREG1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXREG1 | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TXSTA1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0010 | 0000 0010 | uuuu uuuu |
| RCSTA1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 000x | 0000 000x | uuuu uuuu |
| SPBRG2 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| RCREG2 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXREG2 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXSTA2 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0010 | 0000 0010 | uuuu uuuu |
| EECON2 | PIC18F6XJ1X | PIC18F8XJ1X | ---- ---- | ---- ---- | ---- ---- |
| EECON1 | PIC18F6XJ1X | PIC18F8XJ1X | --00 x00- | --00 u00- | --00 u00- |

**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 effected (to cause wake-up).

**4:** See Table 5-2 for Reset value for specific conditions.

**TABLE 5-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS[(4)](CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| IPR3 | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIR3 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu[(3)] |
| PIE3 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| IPR2 | PIC18F6XJ1X | PIC18F8XJ1X | 111- 1111 | 111- 1111 | uuu- uuuu |
| PIR2 | PIC18F6XJ1X | PIC18F8XJ1X | 000- 0000 | 000- 0000 | uuu- uuuu[(3)] |
| PIE2 | PIC18F6XJ1X | PIC18F8XJ1X | 000- 0000 | 000- 0000 | uuu- uuuu |
| IPR1 | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIR1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu[(3)] |
| PIE1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| RCSTA2 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 000x | 0000 000x | uuuu uuuu |
| OSCTUNE | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| TRISJ | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISH | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISG | PIC18F6XJ1X | PIC18F8XJ1X | ---1 1111 | ---1 1111 | ---u uuuu |
| TRISF | PIC18F6XJ1X | PIC18F8XJ1X | 1111 111- | 1111 111- | uuuu uuu- |
| TRISE | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISD | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISC | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISB | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISA | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | uuuu uuuu |
| LATJ | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATH | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATG | PIC18F6XJ1X | PIC18F8XJ1X | ---x xxxx | ---u uuuu | ---u uuuu |
| LATF | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxx- | uuuu uuu- | uuuu uuu- |
| LATE | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATD | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATC | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATB | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATA | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |

**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 effected (to cause wake-up).

**4:** See Table 5-2 for Reset value for specific conditions.

**TABLE 5-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS[4] (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | MCLR Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| PORTJ | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTH | PIC18F6XJ1X | PIC18F8XJ1X | 0000 xxxx | uuuu uuuu | uuuu uuuu |
| PORTG | PIC18F6XJ1X | PIC18F8XJ1X | 000x xxxx | 000u uuuu | uuuu uuuu |
| PORTF | PIC18F6XJ1X | PIC18F8XJ1X | x001 100- | xuuu uuu- | xuuu uuu- |
| PORTE | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTD | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTC | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTB | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTA | PIC18F6XJ1X | PIC18F8XJ1X | 000x 0000 | 000u 0000 | uuuu uuuu |
| SPBRGH1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| BAUDCON1 | PIC18F6XJ1X | PIC18F8XJ1X | 0100 0-00 | 0100 0-00 | uuuu u-uu |
| SPBRGH2 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| BAUDCON2 | PIC18F6XJ1X | PIC18F8XJ1X | 0100 0-00 | 0100 0-00 | uuuu u-uu |
| TMR3H | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR3L | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T3CON | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | uuuu uuuu | uuuu uuuu |
| TMR4 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PR4 | PIC18F6XJ1X | PIC18F8XJ1X | 1111 1111 | 1111 1111 | 1111 1111 |
| CVRCON | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| T4CON | PIC18F6XJ1X | PIC18F8XJ1X | -000 0000 | -000 0000 | -uuu uuuu |
| CCPR4H | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR4L | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP4CON | PIC18F6XJ1X | PIC18F8XJ1X | --00 0000 | --00 0000 | --uu uuuu |
| CCPR5H | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR5L | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP5CON | PIC18F6XJ1X | PIC18F8XJ1X | --00 0000 | --00 0000 | --uu uuuu |
| SSP2BUF | PIC18F6XJ1X | PIC18F8XJ1X | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| SSP2ADD | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP2MSK | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP2STAT | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP2CON1 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP2CON2 | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| CMSTAT | PIC18F6XJ1X | PIC18F8XJ1X | ---- --11 | ---- --11 | ---- --uu |

**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 effected (to cause wake-up).

**4:** See Table 5-2 for Reset value for specific conditions.

**TABLE 5-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS[4] (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{MCLR}$ Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| PMADDRH | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMDOUT1H | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMADDRL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMDOUT1L | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMDIN1H | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMDIN1L | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMCONH | PIC18F6XJ1X | PIC18F8XJ1X | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| PMCONL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMMODEH | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMMODEL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMDOUT2H | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMDOUT2L | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMDIN2H | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMDIN2L | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMEH | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMEL | PIC18F6XJ1X | PIC18F8XJ1X | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMSTATH | PIC18F6XJ1X | PIC18F8XJ1X | 00-- 0000 | 00-- 0000 | uu-- uuuu |
| PMSTATL | PIC18F6XJ1X | PIC18F8XJ1X | 10-- 1111 | 10-- 1111 | uu-- uuuu |

**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 effected (to cause wake-up).

**4:** See Table 5-2 for Reset value for specific conditions.

## 6.0 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.

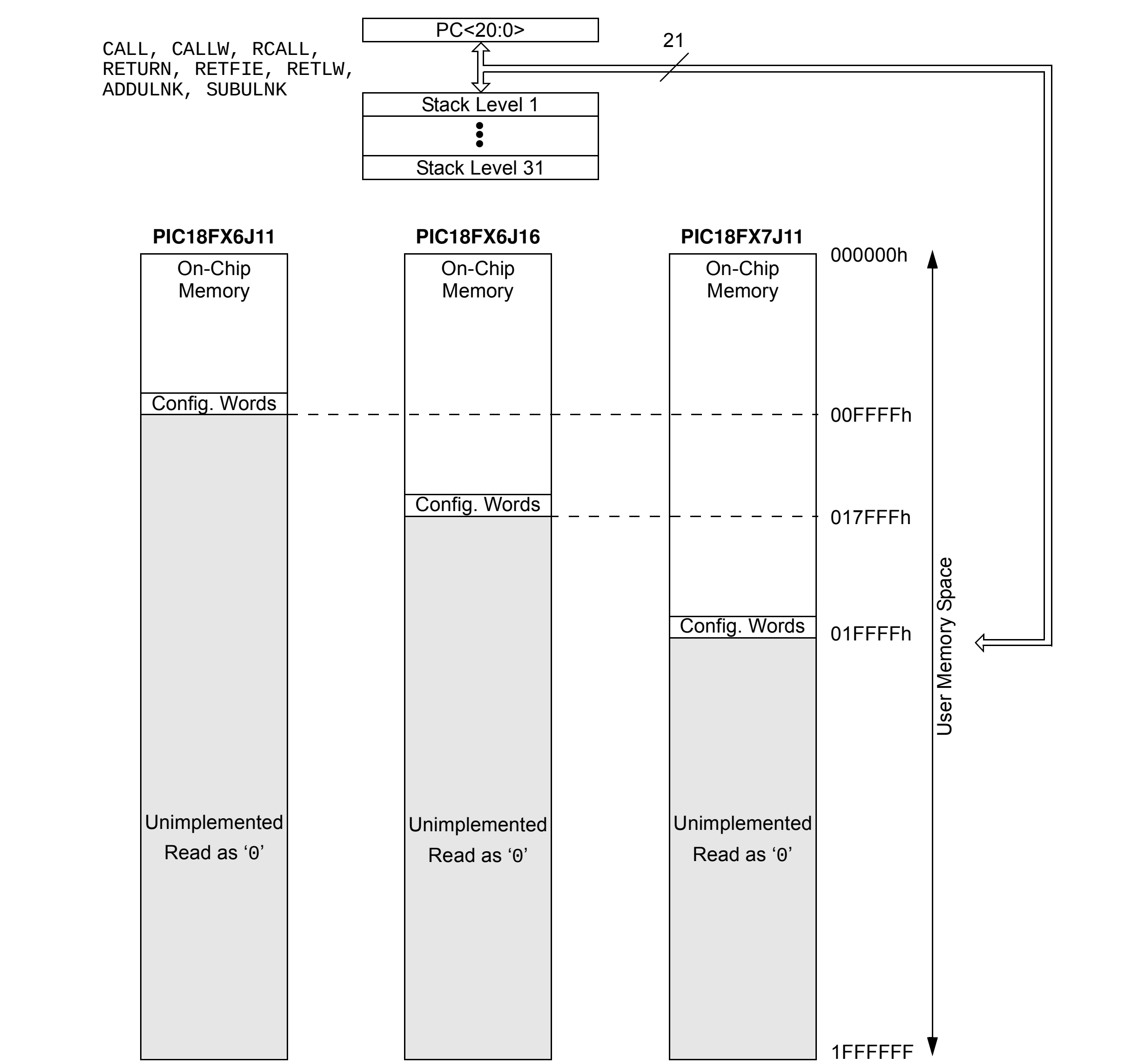
Additional detailed information on the operation of the Flash program memory is provided in **Section 7.0 "Flash Program Memory"**.

## 6.1 Program Memory Organization

PIC18 microcontrollers implement a 21-bit Program Counter (PC) 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 6-3.

**FIGURE 6-1: MEMORY MAPS FOR PIC18F87J11 FAMILY DEVICES**



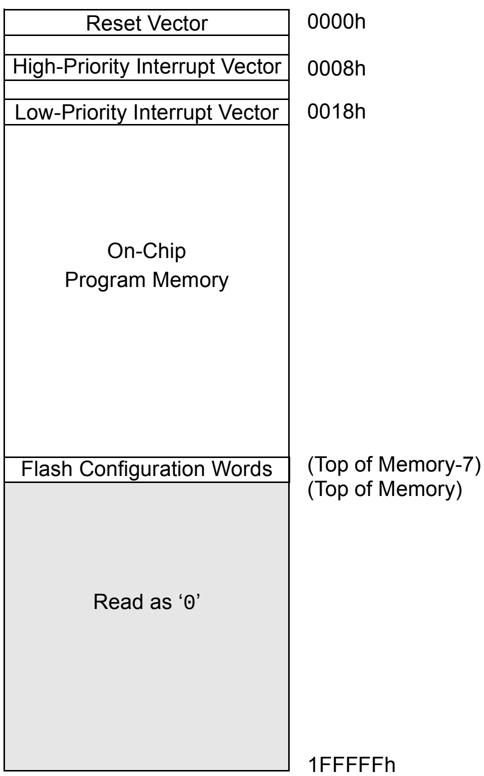

**Note:** Sizes of memory areas are not to scale. Sizes of program memory areas are enhanced to show detail.

### 6.1.1 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 6-2.

**FIGURE 6-2: HARD VECTOR AND CONFIGURATION WORD LOCATIONS FOR PIC18F87J11 FAMILY DEVICES**




**Legend:** (Top of Memory) represents upper boundary of on-chip program memory space (see Figure 6-1 for device-specific values). Shaded area represents unimplemented memory. Areas are not shown to scale.

### 6.1.2 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 6-1. Their location in the memory map is shown with the other memory vectors in Figure 6-2.

Additional details on the device Configuration Words are provided in **Section 25.1 "Configuration Bits"**.

**TABLE 6-1: FLASH CONFIGURATION WORD FOR PIC18F87J11 FAMILY DEVICES**

| Device | Program Memory (Kbytes) | Configuration Word Addresses |
|---|---|---|
| PIC18F66J11 | 64 | FFF8h to FFFFh |
| PIC18F86J11 | | |
| PIC18F66J16 | 96 | 17FF8h to 17FFFh |
| PIC18F86J16 | | |
| PIC18F67J11 | 128 | 1FFF8h to 1FFFFh |
| PIC18F87J11 | | |

### 6.1.3 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 (EMB). There are two distinct operating modes available to the controllers:

- Microcontroller (MC)
- Extended Microcontroller (EMC)

The program memory mode is determined by setting the EMBx Configuration bits (CONFIG3L<5:4>), as shown in Register 6-1. (See also **Section 25.1 "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 EMBx Configuration bits also controls the address bus width of the External Memory Bus. This is covered in more detail in **Section 8.0 "External Memory Bus"**.

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

Figure 6-3 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 6-2.

**REGISTER 6-1: CONFIG3L: CONFIGURATION REGISTER 3 LOW**

| R/WO-1 | R/WO-1 | R/WO-1 | R/WO-1 | R/WO-1 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| WAIT(1) | BW(1) | EMB1(1) | EMB0(1) | EASHFT(1) | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | WO = Write-Once bit | | |
|---|---|---|---|
| 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 **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 is enabled – external address bus is shifted to start at 000000h
0 = Address shifting is disabled – external address bus reflects the PC value

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

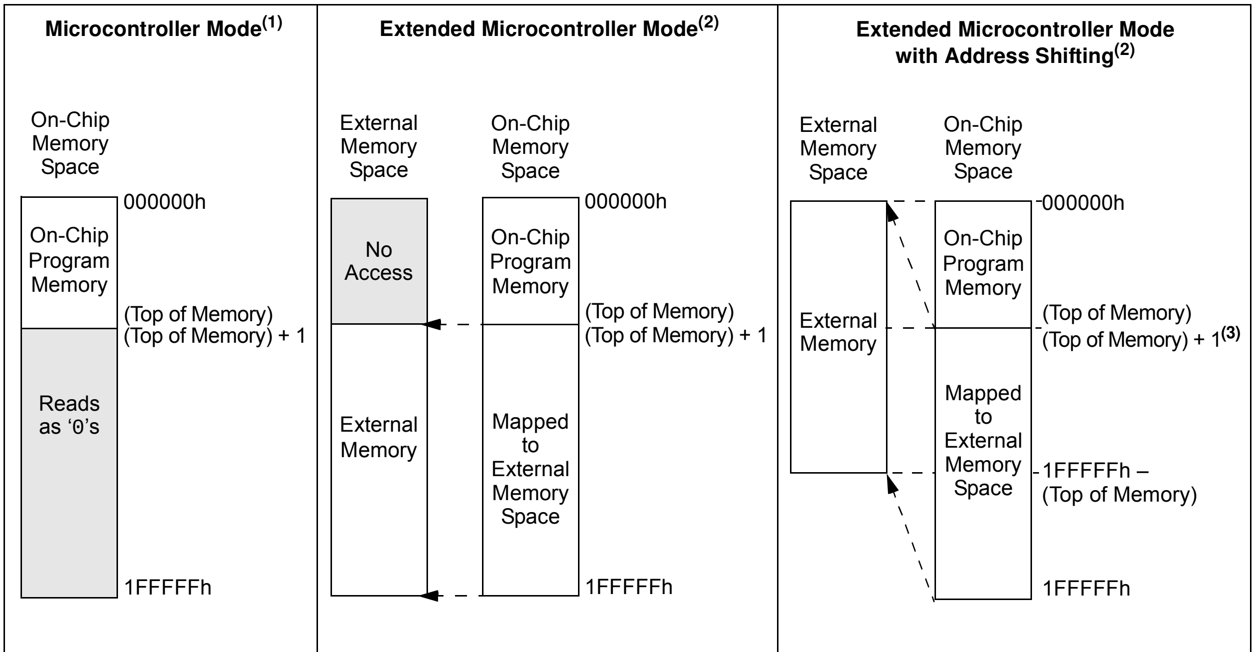
**Note 1:** These bits are implemented only on 80-pin devices.

### 6.1.4 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 6-3: MEMORY MAPS FOR PIC18F87J11 FAMILY PROGRAM MEMORY MODES**




**Legend:** (Top of Memory) represents upper boundary of on-chip program memory space (see Figure 6-1 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 mode 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 EASHFT Configuration bit is set.

**TABLE 6-2: MEMORY ACCESS FOR PIC18F8X11/8616 PROGRAM MEMORY MODES**

| Operating Mode | Internal Program Memory | | | External Program Memory | | |
|---|---|---|---|---|---|---|
| | Execution From | Table Read From | Table Write To | Execution From | Table Read From | Table Write To |
| Microcontroller | Yes | Yes | Yes | No Access | No Access | No Access |
| Extended Microcontroller | Yes | Yes | Yes | Yes | Yes | Yes |

### 6.1.5 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 6.1.8.1 "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.

### 6.1.6 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 affected by any of the RETURN or CALL instructions.

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.

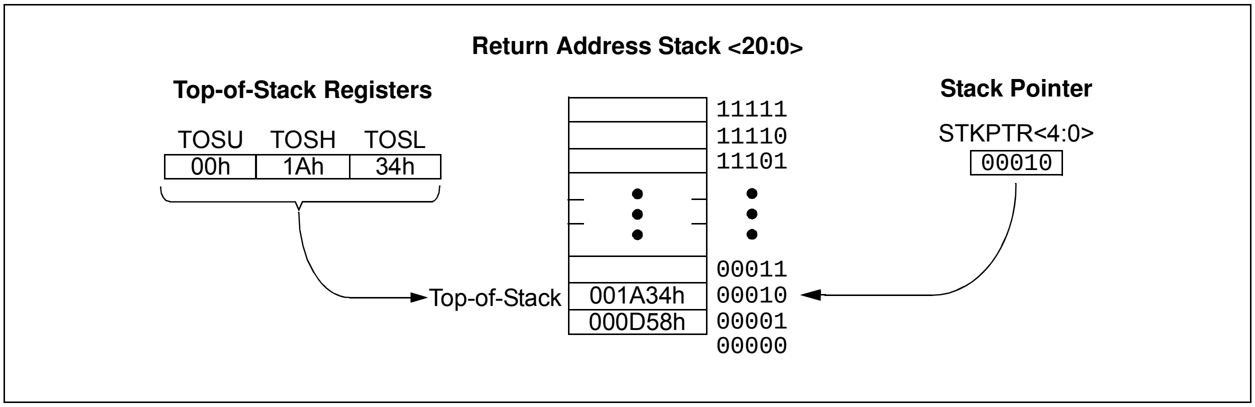
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits indicate if the stack is full, has overflowed or has underflowed.

#### 6.1.6.1 Top-of-Stack (TOS) Access

Only the top of the return address stack 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 6-4). 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 6-4: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**



#### 6.1.6.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 6-2) 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 STKFUL bit is cleared by software or by a POR.

The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 25.1 "Configuration Bits"** for a description of the device Configuration bits.) If STVREN is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit and reset the device. The STKFUL bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the Stack Pointer will increment to 31. Any additional pushes will not overwrite the 31st push and the STKPTR will remain at 31.

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. |
|---|---|

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

### REGISTER 6-2: STKPTR: STACK POINTER REGISTER

| R/C-0 | R/C-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| STKFUL[(1)] | STKUNF[(1)] | — | SP4 | SP3 | SP2 | SP1 | SP0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | C = Clearable Only bit | | |
|---|---|---|---|
| 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 | **STKFUL:** Stack Full Flag bit[(1)]<br>1 = Stack became full or overflowed<br>0 = Stack has not become full or overflowed |
| bit 6 | **STKUNF:** Stack Underflow Flag bit[(1)]<br>1 = Stack underflow occurred<br>0 = Stack underflow did not occur |
| bit 5 | **Unimplemented:** Read as '0' |
| bit 4-0 | **SP<4:0>:** Stack Pointer Location bits |

**Note 1:** Bit 7 and bit 6 are cleared by user software or by a POR.

#### 6.1.6.4 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.

### 6.1.7 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 `RETURN, FAST` instruction is then executed to restore these registers from the Fast Register Stack.

Example 6-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

**EXAMPLE 6-1: 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
```

### 6.1.8 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

#### 6.1.8.1 Computed `GOTO`

A computed `GOTO` is accomplished by adding an offset to the Program Counter. An example is shown in Example 6-2.

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 6-2: COMPUTED `GOTO` USING AN OFFSET VALUE**

```
        MOVF    OFFSET, W
        CALL    TABLE
ORG     nn00h
TABLE   ADDWF   PCL
        RETLW   nnh
        RETLW   nnh
        RETLW   nnh
        .
        .
        .
```

#### 6.1.8.2 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 7.1 "Table Reads and Table Writes"**.

## 6.2 PIC18 Instruction Cycle

### 6.2.1 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 6-5.

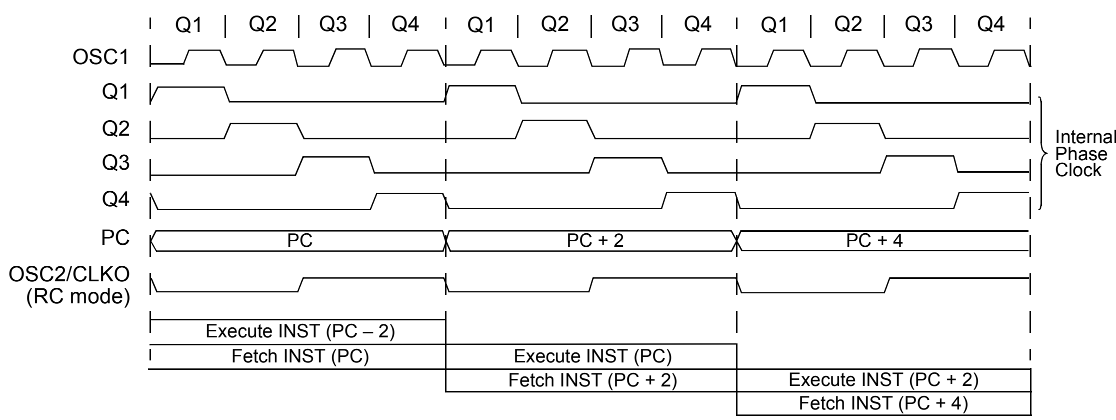
### 6.2.2 INSTRUCTION FLOW/PIPELINING

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 6-3).

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 6-5: CLOCK/INSTRUCTION CYCLE**




**EXAMPLE 6-3: INSTRUCTION PIPELINE FLOW**

| | TCY0 | TCY1 | TCY2 | TCY3 | TCY4 | TCY5 |
|---|---|---|---|---|---|---|
| 1. `MOVLW 55h` | Fetch 1 | Execute 1 | | | | |
| 2. `MOVWF PORTB` | | Fetch 2 | Execute 2 | | | |
| 3. `BRA SUB_1` | | | Fetch 3 | Execute 3 | | |
| 4. `BSF PORTA, BIT3 (Forced NOP)` | | | | Fetch 4 | Flush (`NOP`) | |
| 5. `Instruction @ address SUB_1` | | | | | Fetch SUB_1 | Execute SUB_1 |

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.

### 6.2.3 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 6.1.5 "Program Counter"**).

Figure 6-6 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 6-6 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 26.0 "Instruction Set Summary"** provides further details of the instruction set.

**FIGURE 6-6: INSTRUCTIONS IN PROGRAM MEMORY**

Program Memory Byte Locations →

| | | | LSB = 1 | LSB = 0 | Word Address ↓ |
|---|---|---|---|---|---|
| | | | | | 000000h |
| | | | | | 000002h |
| | | | | | 000004h |
| | | | | | 000006h |
| Instruction 1: | `MOVLW` | `055h` | 0Fh | 55h | 000008h |
| Instruction 2: | `GOTO` | `0006h` | EFh | 03h | 00000Ah |
| | | | F0h | 00h | 00000Ch |
| Instruction 3: | `MOVFF` | `123h, 456h` | C1h | 23h | 00000Eh |
| | | | F4h | 56h | 000010h |
| | | | | | 000012h |
| | | | | | 000014h |

### 6.2.4 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 6-4 shows how this works.

> **Note:** See **Section 6.5 "Program Memory and the Extended Instruction Set"** for information on two-word instructions in the extended instruction set.

**EXAMPLE 6-4: TWO-WORD INSTRUCTIONS**

**CASE 1:**

| Object Code | Source Code |
|---|---|
| `0110 0110 0000 0000` | `TSTFSZ REG1 ; is RAM location 0?` |
| `1100 0001 0010 0011` | `MOVFF REG1, REG2 ; No, skip this word` |
| `1111 0100 0101 0110` | `; Execute this word as a NOP` |
| `0010 0100 0000 0000` | `ADDWF REG3 ; continue code` |

**CASE 2:**

| Object Code | Source Code |
|---|---|
| `0110 0110 0000 0000` | `TSTFSZ REG1 ; is RAM location 0?` |
| `1100 0001 0010 0011` | `MOVFF REG1, REG2 ; Yes, execute this word` |
| `1111 0100 0101 0110` | `; 2nd word of instruction` |
| `0010 0100 0000 0000` | `ADDWF REG3 ; continue code` |

## 6.3 Data Memory Organization

| Note: | The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See **Section 6.6 "Data Memory and the Extended Instruction Set"** for more information. |
|---|---|

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 6-7 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 6.3.2 "Access Bank"** provides a detailed description of the Access RAM.

### 6.3.1 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 (BSR<3:0>). 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.

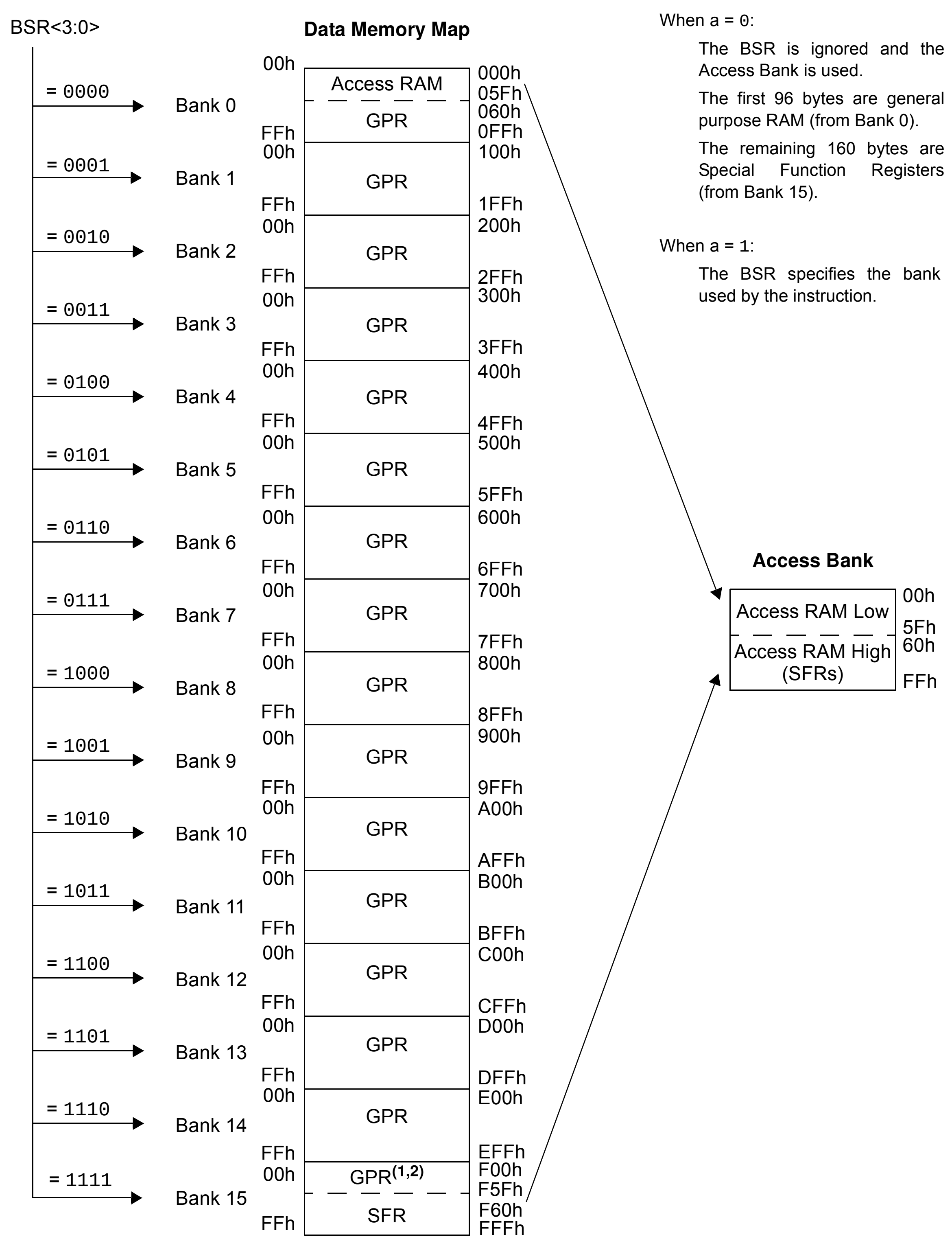
The value of the BSR indicates the bank in data memory. The 8 bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 6-8.

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 6-7 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.

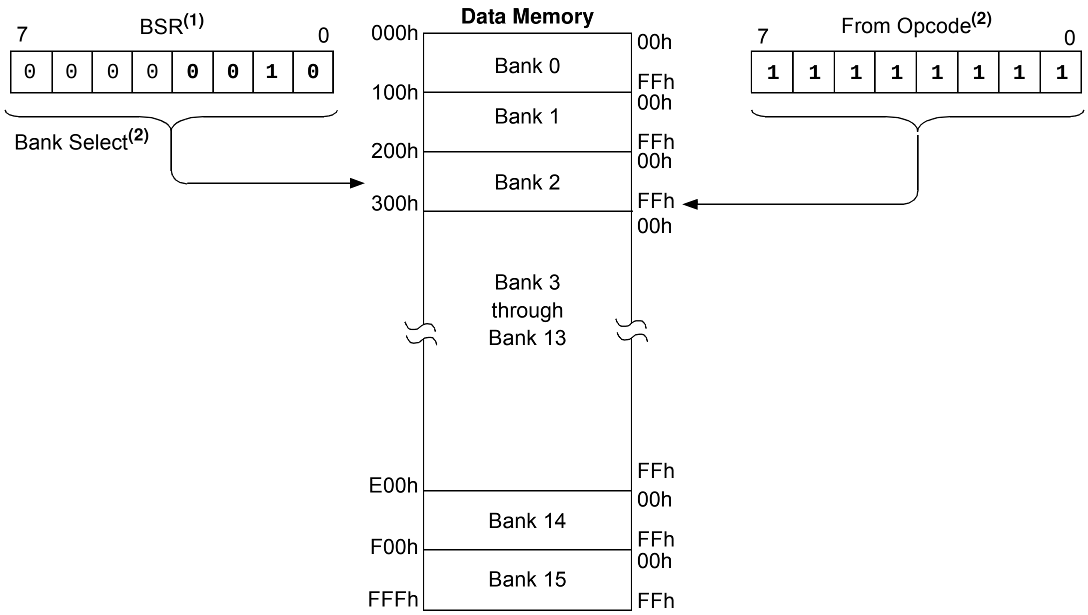
**FIGURE 6-7: DATA MEMORY MAP FOR PIC18F87J11 FAMILY DEVICES**




**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.

**2:** Addresses, F40h to F59h, are not implemented and are not accessible to the user.

**FIGURE 6-8: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)**




**Note 1:** The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.

**2:** The `MOVFF` instruction embeds the entire 12-bit address in the instruction.

### 6.3.2 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 6-7).

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 6.6.3 "Mapping the Access Bank in Indexed Literal Offset Mode"**.

### 6.3.3 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.

### 6.3.4 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 inTable 6-3, Table 6-4 and Table 6-5.

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. Addresses, F40h to F59h, are not implemented and are not accessible to the user. |
|---|---|

**TABLE 6-3: SPECIAL FUNCTION REGISTER MAP FOR PIC18F87J11 FAMILY DEVICES**

| Address | Name | Address | Name | Address | Name | Address | Name | Address | Name | Address | Name |
|---|---|---|---|---|---|---|---|---|---|---|---|
| FFFh | TOSU | FDFh | INDF2(1) | FBFh | ECCP1AS | F9Fh | IPR1 | F7Fh | SPBRGH1 | F5Fh | PMDIN2H |
| FFEh | TOSH | FDEh | POSTINC2(1) | FBEh | ECCP1DEL | F9Eh | PIR1 | F7Eh | BAUDCON1 | F5Eh | PMDIN2L |
| FFDh | TOSL | FDDh | POSTDEC2(1) | FBDh | CCPR1H | F9Dh | PIE1 | F7Dh | SPBRGH2 | F5Dh | PMEH |
| FFCh | STKPTR | FDCh | PREINC2(1) | FBCh | CCPR1L | F9Ch | RCSTA2 | F7Ch | BAUDCON2 | F5Ch | PMEL |
| FFBh | PCLATU | FDBh | PLUSW2(1) | FBBh | CCP1CON | F9Bh | OSCTUNE | F7Bh | TMR3H | F5Bh | PMSTATH |
| FFAh | PCLATH | FDAh | FSR2H | FBAh | ECCP2AS | F9Ah | TRISJ(2) | F7Ah | TMR3L | F5Ah | PMSTATL |
| FF9h | PCL | FD9h | FSR2L | FB9h | ECCP2DEL | F99h | TRISH(2) | F79h | T3CON | F59h | — |
| FF8h | TBLPTRU | FD8h | STATUS | FB8h | CCPR2H | F98h | TRISG | F78h | TMR4 | F58h | — |
| FF7h | TBLPTRH | FD7h | TMR0H | FB7h | CCPR2L | F97h | TRISF | F77h | PR4(3) | F57h | — |
| FF6h | TBLPTRL | FD6h | TMR0L | FB6h | CCP2CON | F96h | TRISE | F76h | T4CON | F56h | — |
| FF5h | TABLAT | FD5h | T0CON | FB5h | ECCP3AS | F95h | TRISD | F75h | CCPR4H | F55h | — |
| FF4h | PRODH | FD4h | — | FB4h | ECCP3DEL | F94h | TRISC | F74h | CCPR4L | F54h | — |
| FF3h | PRODL | FD3h | OSCCON(3) | FB3h | CCPR3H | F93h | TRISB | F73h | CCP4CON | F53h | — |
| FF2h | INTCON | FD2h | CM1CON | FB2h | CCPR3L | F92h | TRISA | F72h | CCPR5H | F52h | — |
| FF1h | INTCON2 | FD1h | CM2CON | FB1h | CCP3CON | F91h | LATJ(2) | F71h | CCPR5L | F51h | — |
| FF0h | INTCON3 | FD0h | RCON | FB0h | SPBRG1 | F90h | LATH(2) | F70h | CCP5CON | F50h | — |
| FEFh | INDF0(1) | FCFh | TMR1H(3) | FAFh | RCREG1 | F8Fh | LATG | F6Fh | SSP2BUF | F4Fh | — |
| FEEh | POSTINC0(1) | FCEh | TMR1L(3) | FAEh | TXREG1 | F8Eh | LATF | F6Eh | SSP2ADD | F4Eh | — |
| FEDh | POSTDEC0(1) | FCDh | T1CON(3) | FADh | TXSTA1 | F8Dh | LATE | F6Dh | SSP2STAT | F4Dh | — |
| FECh | PREINC0(1) | FCCh | TMR2(3) | FACh | RCSTA1 | F8Ch | LATD | F6Ch | SSP2CON1 | F4Ch | — |
| FEBh | PLUSW0(1) | FCBh | PR2(3) | FABh | SPBRG2 | F8Bh | LATC | F6Bh | SSP2CON2 | F4Bh | — |
| FEAh | FSR0H | FCAh | T2CON | FAAh | RCREG2 | F8Ah | LATB | F6Ah | CMSTAT | F4Ah | — |
| FE9h | FSR0L | FC9h | SSP1BUF | FA9h | TXREG2 | F89h | LATA | F69h | PMADDRH(4) | F49h | — |
| FE8h | WREG | FC8h | SSP1ADD | FA8h | TXSTA2 | F88h | PORTJ(2) | F68h | PMADDRL(4) | F48h | — |
| FE7h | INDF1(1) | FC7h | SSP1STAT | FA7h | EECON2 | F87h | PORTH(2) | F67h | PMDIN1H | F47h | — |
| FE6h | POSTINC1(1) | FC6h | SSP1CON1 | FA6h | EECON1 | F86h | PORTG | F66h | PMDIN1L | F46h | — |
| FE5h | POSTDEC1(1) | FC5h | SSP1CON2 | FA5h | IPR3 | F85h | PORTF | F65h | PMCONH | F45h | — |
| FE4h | PREINC1(1) | FC4h | ADRESH | FA4h | PIR3 | F84h | PORTE | F64h | PMCONL | F44h | — |
| FE3h | PLUSW1(1) | FC3h | ADRESL | FA3h | PIE3 | F83h | PORTD | F63h | PMMODEH | F43h | — |
| FE2h | FSR1H | FC2h | ADCON0(3) | FA2h | IPR2 | F82h | PORTC | F62h | PMMODEL | F42h | — |
| FE1h | FSR1L | FC1h | ADCON1(3) | FA1h | PIR2 | F81h | PORTB | F61h | PMDOUT2H | F41h | — |
| FE0h | BSR | FC0h | WDTCON | FA0h | PIE2 | F80h | PORTA | F60h | PMDOUT2L | F40h | — |

**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 6-4 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:** Addresses, F40 to F59, are not implemented and are not accessible to the user.

#### 6.3.4.1 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 6-4.

Access to the alternate registers is enabled in software by setting the ADSHR bit in the WDTCON register (Register 6-3). 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.

#### 6.3.4.2 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 MSSPx module determines which register is being accessed. See **Section 20.4.3.4 “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 12.1.2 “Data Registers”** for additional details.

**TABLE 6-4: SHARED SFR ADDRESSES FOR PIC18F87J11 FAMILY DEVICES**

| Address | | Name | Address | | Name | Address | | Name |
|---|---|---|---|---|---|---|---|---|
| FD3h | (D) | OSCCON | FCDh | (D) | T1CON | FC2h | (D) | ADCON0 |
| | (A) | REFOCON | | (A) | ODCON3 | | (A) | ANCON1 |
| FCFh | (D) | TMR1H | FCCh | (D) | TMR2 | FC1h | (D) | ADCON1 |
| | (A) | ODCON1 | | (A) | PADCFG1 | | (A) | ANCON0 |
| FCEh | (D) | TMR1L | FCBh | (D) | PR2 | F77h | (D) | PR4 |
| | (A) | ODCON2 | | (A) | MEMCON(1) | | (A) | CVRCON |

**Legend:** (D) = Default SFR, accessible only when ADSHR = 0; (A) = Alternate SFR, accessible only when ADSHR = 1.

**Note 1:** This bit is implemented in 80-pin devices only.

## REGISTER 6-3: WDTCON: WATCHDOG TIMER CONTROL REGISTER

| R/W-0 | R-x | U-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| REGSLP | LVDSTAT | — | ADSHR | — | — | — | SWDTEN |
| 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 **REGSLP:** Voltage Regulator Low-Power Operation Enable bit
For details of bit operation, see Register 25-9.

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

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 25-9.

### TABLE 6-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY)

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| TOSU | — | — | — | Top-of-Stack Upper Byte (TOS<20:16>) | | | | | ---0 0000 | 61, 71 |
| TOSH | Top-of-Stack High Byte (TOS<15:8>) | | | | | | | | 0000 0000 | 61, 71 |
| TOSL | Top-of-Stack Low Byte (TOS<7:0>) | | | | | | | | 0000 0000 | 61, 71 |
| STKPTR | STKFUL | STKUNF | — | SP4 | SP3 | SP2 | SP1 | SP0 | 00-0 0000 | 61, 72 |
| PCLATU | — | — | bit 21(1) | Holding Register for PC<20:16> | | | | | ---0 0000 | 61, 71 |
| PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000 0000 | 61, 71 |
| PCL | PC Low Byte (PC<7:0>) | | | | | | | | 0000 0000 | 61, 71 |
| TBLPTRU | — | — | bit 21 | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | --00 0000 | 61, 104 |
| TBLPTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 | 61, 104 |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 | 61, 104 |
| TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 | 61, 104 |
| PRODH | Product Register High Byte | | | | | | | | xxxx xxxx | 61, 117 |
| PRODL | Product Register Low Byte | | | | | | | | xxxx xxxx | 61, 117 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x | 61, 121 |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | INTEDG3 | TMR0IP | INT3IP | RBIP | 1111 1111 | 61, 121 |
| INTCON3 | INT2IP | INT1IP | INT3IE | INT2IE | INT1IE | INT3IF | INT2IF | INT1IF | 1100 0000 | 61, 121 |
| INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) | | | | | | | | N/A | 61, 89 |
| POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | N/A | 61, 90 |
| POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) | | | | | | | | N/A | 61, 90 |
| PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) | | | | | | | | N/A | 61, 90 |
| PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W | | | | | | | | N/A | 61, 90 |
| FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0 High Byte | | | | ---- 0000 | 61, 89 |
| FSR0L | Indirect Data Memory Address Pointer 0 Low Byte | | | | | | | | xxxx xxxx | 61, 89 |
| WREG | Working Register | | | | | | | | xxxx xxxx | 61, 73 |
| INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) | | | | | | | | N/A | 61, 89 |
| POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | N/A | 61, 90 |
| POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) | | | | | | | | N/A | 61, 90 |
| PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) | | | | | | | | N/A | 61, 90 |
| PLUSW1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W | | | | | | | | N/A | 61, 90 |
| FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1 High Byte | | | | ---- 0000 | 61, 89 |
| FSR1L | Indirect Data Memory Address Pointer 1 Low Byte | | | | | | | | xxxx xxxx | 61, 89 |
| BSR | — | — | — | — | Bank Select Register | | | | ---- 0000 | 61, 76 |
| INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) | | | | | | | | N/A | 62, 89 |
| POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | N/A | 62, 90 |
| POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) | | | | | | | | N/A | 62, 90 |
| PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) | | | | | | | | N/A | 62, 90 |
| PLUSW2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W | | | | | | | | N/A | 62, 90 |

**Legend:** x = unknown; u = unchanged; – = unimplemented; q = value depends on condition; **Bold** = 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 20.4.3.2 "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 12.1.2 "Data Registers"** for more information.

## TABLE 6-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY) (CONTINUED)

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2 High Byte | | | | ---- 0000 | 62, 89 |
| FSR2L | Indirect Data Memory Address Pointer 2 Low Byte | | | | | | | | xxxx xxxx | 62, 89 |
| STATUS | — | — | — | N | OV | Z | DC | C | ---x xxxx | 62, 87 |
| TMR0H | Timer0 Register High Byte | | | | | | | | 0000 0000 | 62, 195 |
| TMR0L | Timer0 Register Low Byte | | | | | | | | xxxx xxxx | 62, 195 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 1111 1111 | 62, 194 |
| **OSCCON**[2]/ | IDLEN | IRCF2 | IRCF1 | IRCF0 | OSTS[4] | — | SCS1 | SCS0 | 0110 q100 | 62, 38 |
| **REFOCON**[3] | ROON | — | ROSSLP | ROSEL | RODIV3 | RODIV2 | RODIV1 | RODIV0 | 0-00 0000 | 62, 45 |
| CM1CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 | 0001 1111 | 62, 320 |
| CM2CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 | 0001 1111 | 62, 320 |
| RCON | IPEN | — | $\overline{\text{CM}}$ | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 0-11 1100 | 60, 62, 133 |
| **TMR1H**[2]/ | Timer1 Register High Byte | | | | | | | | xxxx xxxx | 62, 198 |
| **ODCON1**[3] | — | — | — | CCP5OD | CCP4OD | ECCP3OD | ECCP2OD | ECCP1OD | ---0 0000 | 62, 138 |
| **TMR1L**[2]/ | Timer1 Register Low Byte | | | | | | | | xxxx xxxx | 62, 198 |
| **ODCON2**[3] | — | — | — | — | — | — | U2OD | U1OD | ---- --00 | 62, 138 |
| **T1CON**[2]/ | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 0000 0000 | 62, 198 |
| **ODCON3**[3] | — | — | — | — | — | — | SPI2OD | SPI1OD | ---- --00 | 62, 138 |
| **TMR2**[2]/ | Timer2 Register | | | | | | | | 0000 0000 | 62, 203 |
| **PADCFG1**[3] | — | — | — | — | — | — | — | PMPTTL | ---- ---0 | 62, 139 |
| **PR2**[2]/ | Timer2 Period Register | | | | | | | | 1111 1111 | 62, 203 |
| **MEMCON**[3,7] | EDBIS | — | WAIT1 | WAIT0 | — | — | WM1 | WM0 | 0-00 --00 | 62, 106 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 | 62, 203 |
| SSP1BUF | MSSP1 Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | 62, 238, 248 |
| SSP1ADD/ | MSSP1 Address Register ($I^2C$™ Slave mode), MSSP1 Baud Rate Reload Register ($I^2C$ Master mode) | | | | | | | | 0000 0000 | 62, 248 |
| SSP1MSK[5] | MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0 | 0000 0000 | 62, 255 |
| SSP1STAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 0000 0000 | 62, 239, 249 |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 62, 240, 250 |
| SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN/ | SEN | 0000 0000 | 62, 251, 283 |
| | GCEN | ACKSTAT | ADMSK5[6] | ADMSK4[6] | ADMSK3[6] | ADMSK2[6] | ADMSK1[6] | SEN | | |
| ADRESH | A/D Result Register High Byte | | | | | | | | xxxx xxxx | 63, 309 |
| ADRESL | A/D Result Register Low Byte | | | | | | | | xxxx xxxx | 63, 309 |
| **ADCON0**[2]/ | VCFG1 | VCFG0 | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON | 0000 0000 | 63, 309 |
| **ANCON1**[3] | PCFG15 | PCFG14 | PCFG13 | PCFG12 | PCFG11 | PCFG10 | PCFG9 | PCFG8 | 0000 0000 | 63, 311 |
| **ADCON1**[2]/ | ADFM | ADCAL | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 0000 0000 | 63, 310 |
| **ANCON0**[3] | PCFG7 | PCFG6 | — | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 00-0 0000 | 63, 311 |
| WDTCON | REGSLP | LVDSTAT | — | ADSHR | — | — | — | SWDTEN | 0x-0 ---0 | 63, 339 |

**Legend:** x = unknown; u = unchanged; – = unimplemented; q = value depends on condition; **Bold** = 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 $I^2C$™ Slave mode. See **Section 20.4.3.2 “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 12.1.2 “Data Registers”** for more information.

**TABLE 6-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY) (CONTINUED)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| ECCP1AS | ECCP1ASE | ECCP1AS2 | ECCP1AS1 | ECCP1AS0 | PSS1AC1 | PSS1AC0 | PSS1BD1 | PSS1BD0 | 0000 0000 | 63, 235 |
| ECCP1DEL | P1RSEN | P1DC6 | P1DC5 | P1DC4 | P1DC3 | P1DC2 | P1DC1 | P1DC0 | 0000 0000 | 63, 235 |
| CCPR1H | Capture/Compare/PWM Register 1 HIgh Byte | | | | | | | | xxxx xxxx | 63, 235 |
| CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | 63, 235 |
| CCP1CON | P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 0000 0000 | 63, 235 |
| ECCP2AS | ECCP2ASE | ECCP2AS2 | ECCP2AS1 | ECCP2AS0 | PSS2AC1 | PSS2AC0 | PSS2BD1 | PSS2BD0 | 0000 0000 | 63, 235 |
| ECCP2DEL | P2RSEN | P2DC6 | P2DC5 | P2DC4 | P2DC3 | P2DC2 | P2DC1 | P2DC0 | 0000 0000 | 63, 235 |
| CCPR2H | Capture/Compare/PWM Register 2 High Byte | | | | | | | | xxxx xxxx | 63, 235 |
| CCPR2L | Capture/Compare/PWM Register 2 Low Byte | | | | | | | | xxxx xxxx | 63, 235 |
| CCP2CON | P2M1 | P2M0 | DC2B1 | DC2B0 | CCP2M3 | CCP2M2 | CCP2M1 | CCP2M0 | 0000 0000 | 63, 235 |
| ECCP3AS | ECCP3ASE | ECCP3AS2 | ECCP3AS1 | ECCP3AS0 | PSS3AC1 | PSS3AC0 | PSS3BD1 | PSS3BD0 | 0000 0000 | 63, 235 |
| ECCP3DEL | P3RSEN | P3DC6 | P3DC5 | P3DC4 | P3DC3 | P3DC2 | P3DC1 | P3DC0 | 0000 0000 | 63, 235 |
| CCPR3H | Capture/Compare/PWM Register 1 High Byte | | | | | | | | xxxx xxxx | 63, 235 |
| CCPR3L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | | xxxx xxxx | 63, 235 |
| CCP3CON | P3M1 | P3M0 | DC3B1 | DC3B0 | CCP3M3 | CCP3M2 | CCP3M1 | CCP3M0 | 0000 0000 | 63, 235 |
| SPBRG1 | EUSART1 Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 63, 289 |
| RCREG1 | EUSART1 Receive Register | | | | | | | | 0000 0000 | 63, 297, 299 |
| TXREG1 | EUSART1 Transmit Register | | | | | | | | xxxx xxxx | 63, 295, 296 |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 63, 295 |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 63, 297 |
| SPBRG2 | EUSART2 Baud Rate Generator Register Low Byte | | | | | | | | 0000 0000 | 63, 289 |
| RCREG2 | EUSART2 Receive Register | | | | | | | | 0000 0000 | 63, 297, 299 |
| TXREG2 | EUSART2 Transmit Register | | | | | | | | 0000 0000 | 63, 295, 296 |
| TXSTA2 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 | 63, 295 |
| EECON2 | Program Memory Control Register 2 (not a physical register) | | | | | | | | ---- ---- | 63, 96 |
| EECON1 | — | — | WPROG | FREE | WRERR | WREN | WR | — | --00 x00- | 63, 96 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 1111 1111 | 64, 130 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 0000 0000 | 64, 124 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 0000 0000 | 64, 127 |
| IPR2 | OSCFIP | CM2IP | CM1IP | — | BCL1IP | LVDIP | TMR3IP | CCP2IP | 111- 1111 | 64, 130 |
| PIR2 | OSCFIF | CM2IF | CM1IF | — | BCL1IF | LVDIF | TMR3IF | CCP2IF | 000- 0000 | 64, 124 |
| PIE2 | OSCFIE | CM2IE | CM1IE | — | BCL1IE | LVDIE | TMR3IE | CCP2IE | 000- 0000 | 64, 127 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 1111 1111 | 64, 130 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 0000 0000 | 64, 124 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 0000 0000 | 64, 127 |
| RCSTA2 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 64, 297 |
| OSCTUNE | INTSRC | PLLEN | TUN5 | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 | 0000 0000 | 64, 39 |

**Legend:** x = unknown; u = unchanged; – = unimplemented; q = value depends on condition; **Bold** = 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 I$^2$C™ Slave mode. See **Section 20.4.3.2 "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 12.1.2 "Data Registers"** for more information.

## TABLE 6-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY) (CONTINUED)

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| TRISJ(7) | TRISJ7 | TRISJ6 | TRISJ5 | TRISJ4 | TRISJ3 | TRISJ2 | TRISJ1 | TRISJ0 | 1111 1111 | 64, 165 |
| TRISH(7) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 | 1111 1111 | 64, 163 |
| TRISG | — | — | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 | ---1 1111 | 64, 160 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — | 1111 111- | 64, 157 |
| TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 | 1111 1111 | 64, 154 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 1111 1111 | 64, 151 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 1111 1111 | 64, 148 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 1111 1111 | 64, 145 |
| TRISA | TRISA7(8) | TRISA6(8) | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 1111 1111 | 64, 142 |
| LATJ(7) | LATJ7 | LATJ6 | LATJ5 | LATJ4 | LATJ3 | LATJ2 | LATJ1 | LATJ0 | xxxx xxxx | 64, 165 |
| LATH(7) | LATH7 | LATH6 | LATH5 | LATH4 | LATH3 | LATH2 | LATH1 | LATH0 | xxxx xxxx | 64, 163 |
| LATG | — | — | — | LATG4 | LATG3 | LATG2 | LATG1 | LATG0 | ---x xxxx | 64, 160 |
| LATF | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | — | xxxx xxx- | 64, 157 |
| LATE | LATE7 | LATE6 | LATE5 | LATE4 | LATE3 | LATE2 | LATE1 | LATE0 | xxxx xxxx | 64, 154 |
| LATD | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | xxxx xxxx | 64, 151 |
| LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | xxxx xxxx | 64, 148 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | xxxx xxxx | 64, 145 |
| LATA | LATA7(8) | LATA6(8) | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | xxxx xxxx | 64, 142 |
| PORTJ(7) | RJ7 | RJ6 | RJ5 | RJ4 | RJ3 | RJ2 | RJ1 | RJ0 | xxxx xxxx | 65, 165 |
| PORTH(7) | RH7 | RH6 | RH5 | RH4 | RH3 | RH2 | RH1 | RH0 | 0000 xxxx | 65, 163 |
| PORTG | RDPU | REPU | RJPU(7) | RG4 | RG3 | RG2 | RG1 | RG0 | 000x xxxx | 65, 160 |
| PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | — | x000 000- | 65, 157 |
| PORTE | RE7 | RE6 | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 | xxxx xxxx | 65, 154 |
| PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxx xxxx | 65, 151 |
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | xxxx xxxx | 65, 148 |
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | xxxx xxxx | 65, 145 |
| PORTA | RA7(8) | RA6(8) | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 000x 0000 | 65, 142 |
| SPBRGH1 | EUSART1 Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 65, 289 |
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 0100 0-00 | 65, 289 |
| SPBRGH2 | EUSART2 Baud Rate Generator Register High Byte | | | | | | | | 0000 0000 | 65, 289 |
| BAUDCON2 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 0100 0-00 | 65, 289 |
| TMR3H | Timer3 Register High Byte | | | | | | | | xxxx xxxx | 65, 210 |
| TMR3L | Timer3 Register Low Byte | | | | | | | | xxxx xxxx | 65, 210 |
| T3CON | RD16 | T3CCP2 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 0000 0000 | 65, 210 |
| TMR4 | Timer4 Register | | | | | | | | 0000 0000 | 65, 209 |
| **PR4**(2)/ | Timer4 Period Register | | | | | | | | 1111 1111 | 65, 210 |
| **CVRCON**(3) | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 0000 0000 | 65, 328 |
| T4CON | — | T4OUTPS3 | T4OUTPS2 | T4OUTPS1 | T4OUTPS0 | TMR4ON | T4CKPS1 | T4CKPS0 | -000 0000 | 65, 209 |

**Legend:** x = unknown; u = unchanged; – = unimplemented; q = value depends on condition; **Bold** = 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 $I^2C$™ Slave mode. See **Section 20.4.3.2 "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 12.1.2 "Data Registers"** for more information.

### TABLE 6-5: REGISTER FILE SUMMARY (PIC18F87J11 FAMILY) (CONTINUED)

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|---|---|---|---|---|---|---|---|---|---|---|
| CCPR4H | Capture/Compare/PWM Register 4 High Byte | | | | | | | | xxxx xxxx | 65, 212 |
| CCPR4L | Capture/Compare/PWM Register 4 Low Byte | | | | | | | | xxxx xxxx | 65, 212 |
| CCP4CON | — | — | DC4B1 | DC4B0 | CCP4M3 | CCP4M2 | CCP4M1 | CCP4M0 | --00 0000 | 65, 212 |
| CCPR5H | Capture/Compare/PWM Register 5 High Byte | | | | | | | | xxxx xxxx | 65, 212 |
| CCPR5L | Capture/Compare/PWM Register 5 Low Byte | | | | | | | | xxxx xxxx | 65, 212 |
| CCP5CON | — | — | DC5B1 | DC5B0 | CCP5M3 | CCP5M2 | CCP5M1 | CCP5M0 | --00 0000 | 65, 212 |
| SSP2BUF | MSSP2 Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx | 65, 238, 248 |
| SSP2ADD/ | MSSP2 Address Register ($I^2C$™ Slave mode), MSSP2 Baud Rate Reload Register ($I^2C$ Master mode) | | | | | | | | 0000 0000 | 65, 248 |
| SSP2MSK[5] | MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0 | 0000 0000 | 65, 255 |
| SSP2STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 0000 0000 | 65, 239, 249 |
| SSP2CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 | 65, 240, 250 |
| SSP2CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN/ | SEN | 0000 0000 | 65, 251, 283 |
| | GCEN | ACKSTAT | ADMSK5[6] | ADMSK4[6] | ADMSK3[6] | ADMSK2[6] | ADMSK1[6] | SEN | | |
| CMSTAT | — | — | — | — | — | — | COUT2 | COUT1 | ---- --11 | 65, 321 |
| **PMADDRH**/ | CS2 | CS1 | Parallel Master Port Address High Byte | | | | | | 0000 0000 | 66, 174 |
| **PMDOUT1H**[9] | Parallel Port Out Data High Byte (Buffer 1) | | | | | | | | 0000 0000 | 66, 177 |
| **PMADDRL**/ | Parallel Master Port Address Low Byte | | | | | | | | 0000 0000 | 66, 174 |
| **PMDOUT1L**[9] | Parallel Port Out Data Low Byte (Buffer 0) | | | | | | | | 0000 0000 | 66, 174 |
| PMDIN1H | Parallel Port In Data High Byte (Buffer 1) | | | | | | | | 0000 0000 | 66, 174 |
| PMDIN1L | Parallel Port In Data Low Byte (Buffer 0) | | | | | | | | 0000 0000 | 66, 174 |
| PMCONH | PMPEN | — | PSIDL | ADRMUX1 | ADRMUX0 | PTBEEN | PTWREN | PTRDEN | 0-00 0000 | 66, 168 |
| PMCONL | CSF1 | CSF0 | ALP | CS2P | CS1P | BEP | WRSP | RDSP | 0000 0000 | 66, 169 |
| PMMODEH | BUSY | IRQM1 | IRQM0 | INCM1 | INCM0 | MODE16 | MODE1 | MODE0 | 0000 0000 | 66, 170 |
| PMMODEL | WAITB1 | WAITB0 | WAITM3 | WAITM2 | WAITM1 | WAITM0 | WAITE1 | WAITE0 | 0000 0000 | 66, 171 |
| PMDOUT2H | Parallel Port Out Data High Byte (Buffer 3) | | | | | | | | 0000 0000 | 66, 174 |
| PMDOUT2L | Parallel Port Out Data Low Byte (Buffer 2) | | | | | | | | 0000 0000 | 66, 174 |
| PMDIN2H | Parallel Port In Data High Byte (Buffer 3) | | | | | | | | 0000 0000 | 66, 174 |
| PMDIN2L | Parallel Port In Data Low Byte (Buffer 2) | | | | | | | | 0000 0000 | 66, 174 |
| PMEH | PTEN15 | PTEN14 | PTEN13 | PTEN12 | PTEN11 | PTEN10 | PTEN9 | PTEN8 | 0000 0000 | 66, 171 |
| PMEL | PTEN7 | PTEN6 | PTEN5 | PTEN4 | PTEN3 | PTEN2 | PTEN1 | PTEN0 | 0000 0000 | 66, 172 |
| PMSTATH | IBF | IBOV | — | — | IB3F | IB2F | IB1F | IB0F | 00-- 0000 | 66, 172 |
| PMSTATL | OBE | OBUF | — | — | OB3E | OB2E | OB1E | OB0E | 10-- 1111 | 66, 173 |

**Legend:** x = unknown; u = unchanged; – = unimplemented; q = value depends on condition; **Bold** = 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 $I^2C$™ Slave mode. See **Section 20.4.3.2 "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 12.1.2 "Data Registers"** for more information.

### 6.3.5 STATUS REGISTER

The STATUS register, shown in Register 6-4, 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 '`000u u1uu`'. 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 26-2 and Table 26-3.

> **Note:** The C and DC bits operate as a $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ bit respectively, in subtraction.

**REGISTER 6-4: STATUS REGISTER**

| U-0 | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | — | N | OV | Z | DC(1) | C(2) |
| 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 **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/$\overline{\text{borrow}}$ 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/$\overline{\text{borrow}}$ 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 $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand.

**2:** For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand.

## 6.4 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 6.6 "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 6.6.1 "Indexed Addressing with Literal Offset"**.

### 6.4.1 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.

### 6.4.2 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 6.3.3 "General Purpose Register File"**), or a location in the Access Bank (**Section 6.3.2 "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 6.3.1 "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.

### 6.4.3 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 6-5. It also enables users to perform Indexed Addressing and other Stack Pointer operations for program memory in data memory.

**EXAMPLE 6-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING**

```
        LFSR    FSR0, 100h ;
NEXT    CLRF    POSTINC0   ; Clear INDF
                           ; register then
                           ; inc pointer
        BTFSS   FSR0H, 1   ; All done with
                           ; Bank1?
        BRA     NEXT       ; NO, clear next
CONTINUE                   ; YES, continue
```

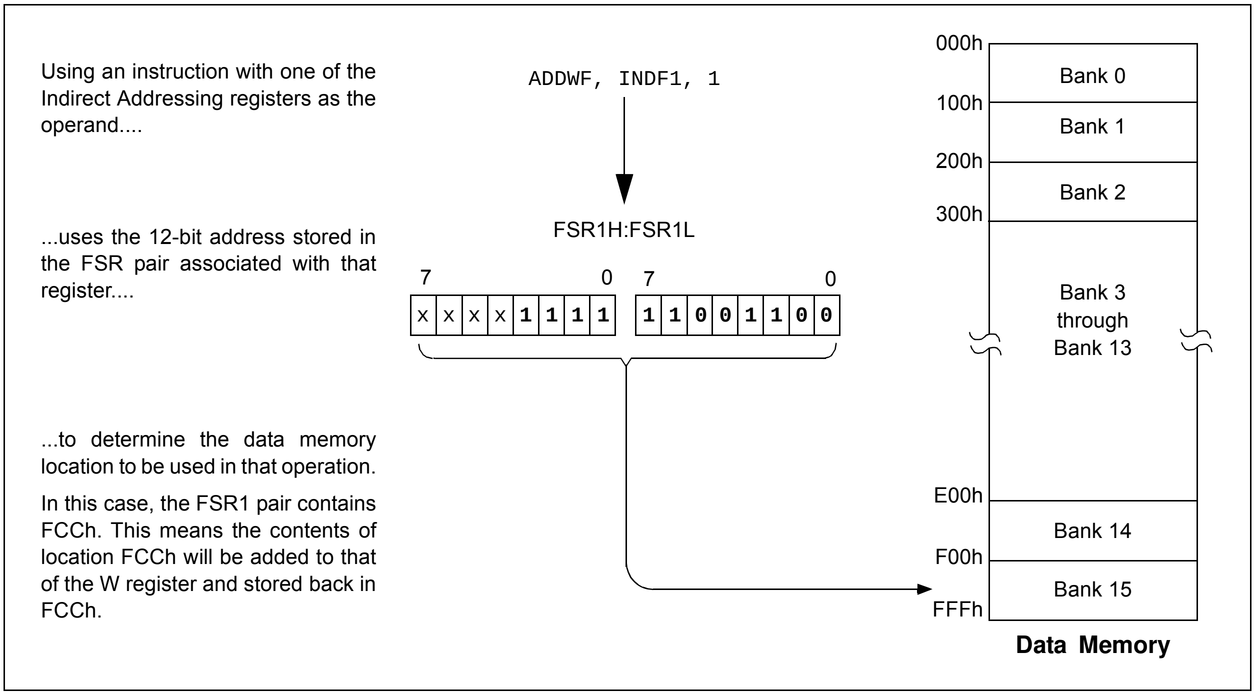
#### 6.4.3.1 FSR Registers and the INDF Operand

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 6-9: INDIRECT ADDRESSING**



#### 6.4.3.2 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 -128 to 127) 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, roll-overs 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.

#### 6.4.3.3 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.

## 6.5 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 6.2.4 "Two-Word Instructions"**.

## 6.6 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.

### 6.6.1 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, which are most of the 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.

### 6.6.2 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 do not 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 6-10.

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 26.2.1 "Extended Instruction Syntax"**.

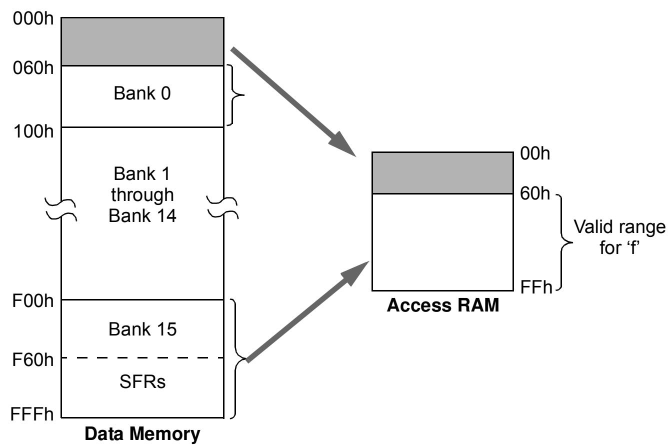
**FIGURE 6-10: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

**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.

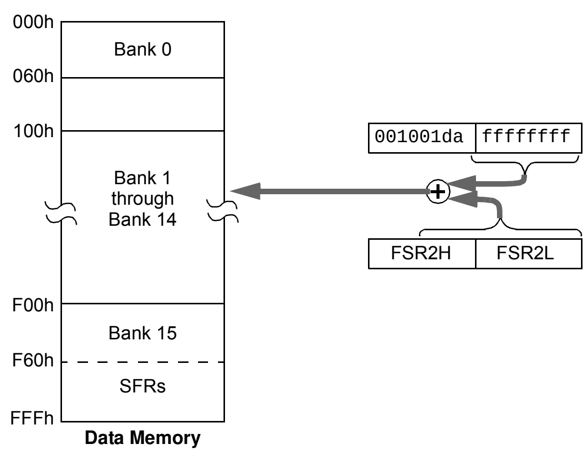
Locations below 060h are not available in this addressing mode.




**When a = `0` and f ≤ 5Fh:**

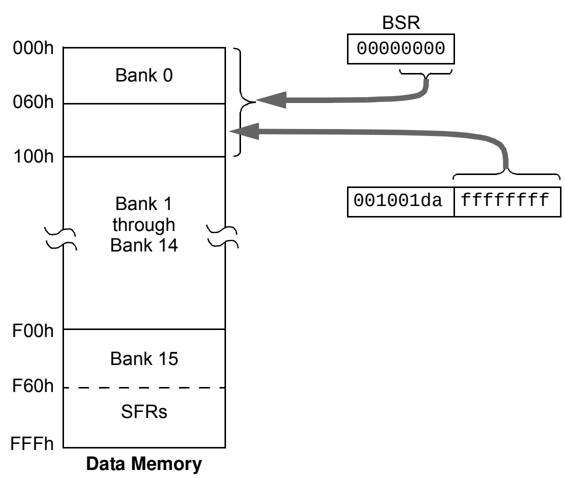
The instruction executes in Indexed Literal Offset mode. ‘f’ is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

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.



### 6.6.3 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 6.3.2 "Access Bank"**). An example of Access Bank remapping in this addressing mode is shown in Figure 6-11.

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.

### 6.6.4 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 6-11: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING**

**Example Situation:**

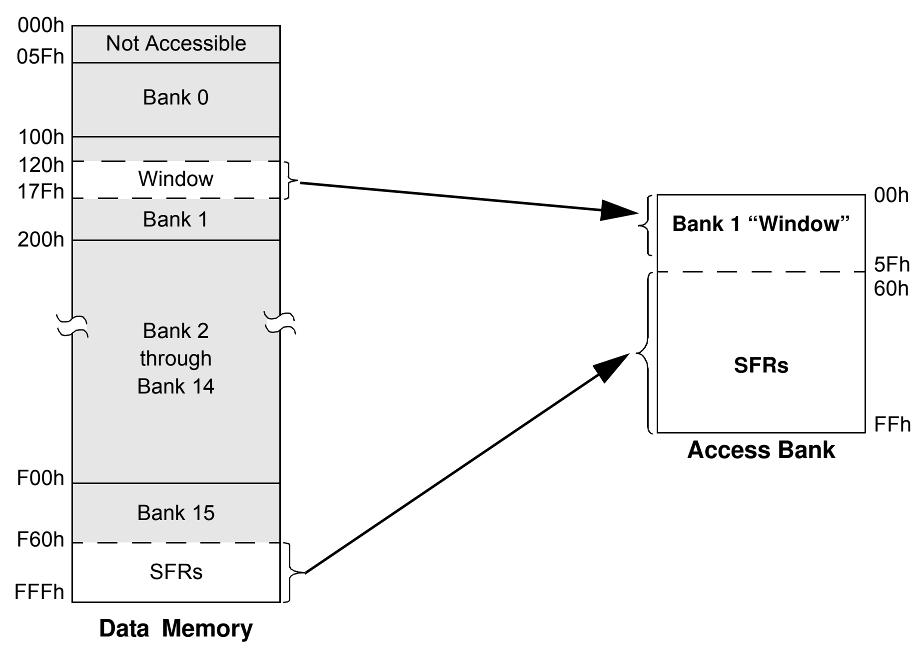
`ADDWF f, d, a`

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.



**NOTES:**

## 7.0 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.

### 7.1 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)

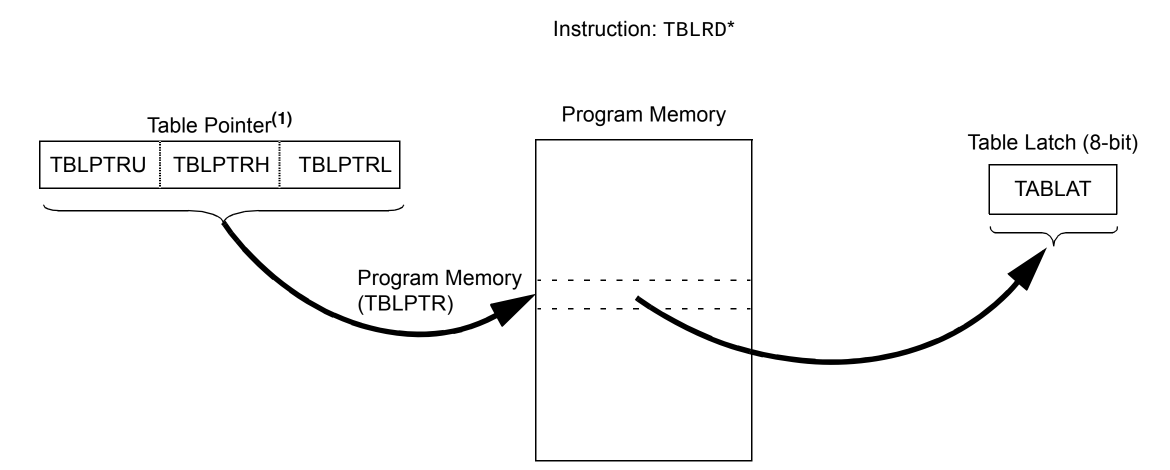
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 7-1 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 7.5 "Writing to Flash Program Memory"**. Figure 7-2 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.

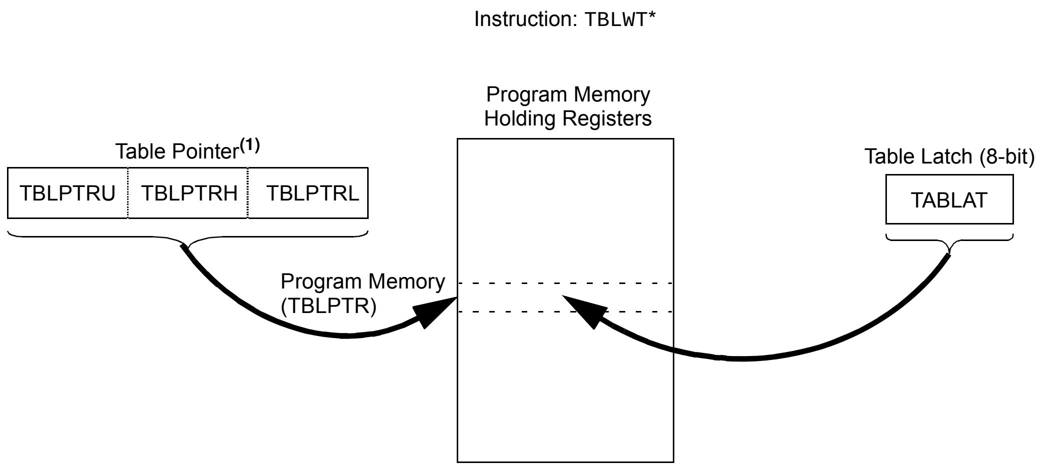
**FIGURE 7-1: TABLE READ OPERATION**




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

**FIGURE 7-2: TABLE WRITE OPERATION**




**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 7.5 "Writing to Flash Program Memory"**.

## 7.2 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

### 7.2.1 EECON1 AND EECON2 REGISTERS

The EECON1 register (Register 7-1) 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.

**REGISTER 7-1: EECON1: EEPROM CONTROL REGISTER 1**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-x | R/W-0 | R/S-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | WPROG | FREE | WRERR | WREN | WR | — |
| bit 7 | | | | | | | bit 0 |

| | | | |
|---|---|---|---|
| **Legend:** | S = Settable bit (cannot be cleared in software) | | |
| 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 **Unimplemented:** Read as ‘0’

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

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erases the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation)
0 = Performs 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’

### 7.2.2 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.

### 7.2.3 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 7-1. These operations on the TBLPTR only affect the low-order 21 bits.

### 7.2.4 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 7.5 "Writing to Flash Program Memory"**.

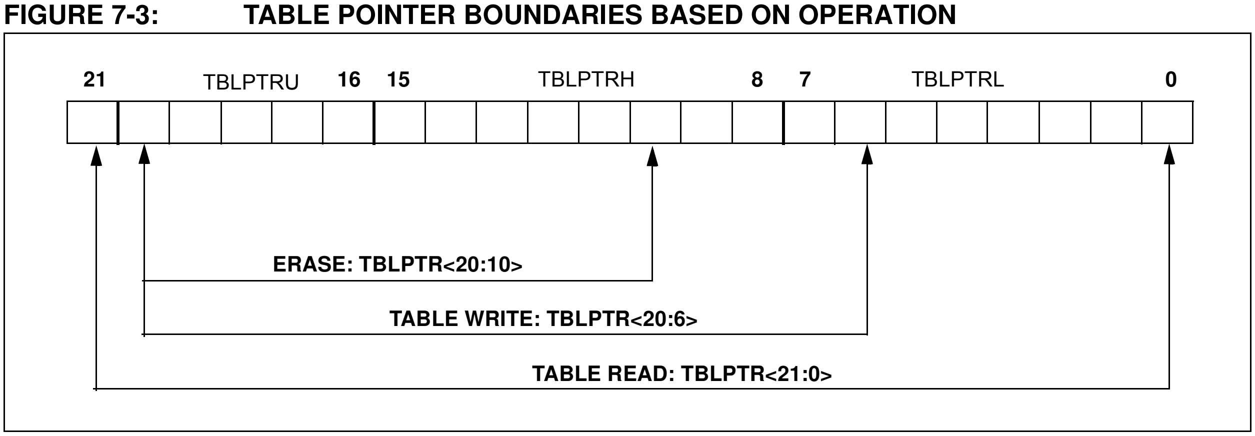
When an erase of program memory is executed, the 12 MSbs of the Table Pointer register point to the 1024-byte block that will be erased. The Least Significant bits are ignored.

Figure 7-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

**TABLE 7-1: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS**

| Example | Operation on Table Pointer |
|---|---|
| TBLRD*<br>TBLWT* | TBLPTR is not modified |
| TBLRD*+<br>TBLWT*+ | TBLPTR is incremented after the read/write |
| TBLRD*-<br>TBLWT*- | TBLPTR is decremented after the read/write |
| TBLRD+*<br>TBLWT+* | TBLPTR is incremented before the read/write |

**FIGURE 7-3: TABLE POINTER BOUNDARIES BASED ON OPERATION**



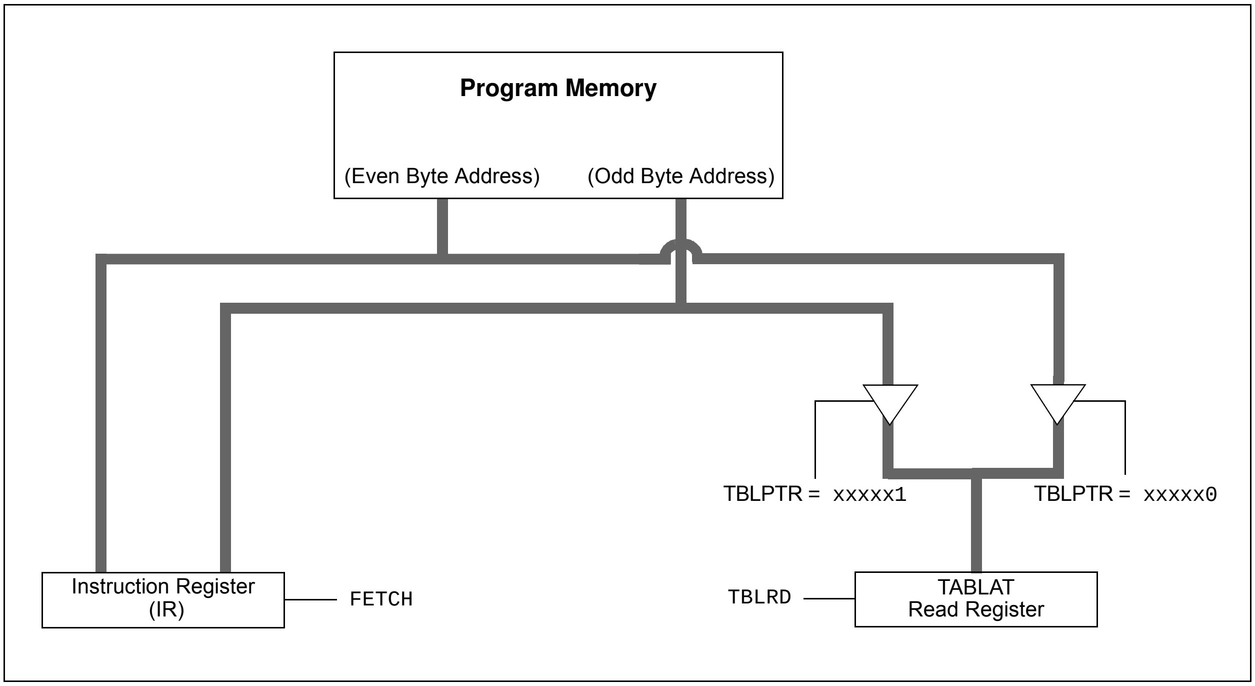
## 7.3 Reading the Flash Program Memory

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 7-4 shows the interface between the internal program memory and the TABLAT.

**FIGURE 7-4: READS FROM FLASH PROGRAM MEMORY**




**EXAMPLE 7-1: READING A FLASH PROGRAM MEMORY WORD**

```
           MOVLW    CODE_ADDR_UPPER         ; Load TBLPTR with the base
           MOVWF    TBLPTRU                 ; address of the word
           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
```

## 7.4 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.

### 7.4.1 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 H'55' to EECON2.
5. Write H'AA' 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 7-2: ERASING A FLASH PROGRAM MEMORY ROW**

```
                    MOVLW   CODE_ADDR_UPPER      ; load TBLPTR with the base
                    MOVWF   TBLPTRU              ; address of the memory block
                    MOVLW   CODE_ADDR_HIGH
                    MOVWF   TBLPTRH
                    MOVLW   CODE_ADDR_LOW
                    MOVWF   TBLPTRL
        ERASE_ROW
                    BSF     EECON1, WREN         ; enable write to memory
                    BSF     EECON1, FREE         ; enable Row Erase operation
                    BCF     INTCON, GIE          ; disable interrupts
    Required        MOVLW   H'55'
    Sequence        MOVWF   EECON2               ; write H'55'
                    MOVLW   H'AA'
                    MOVWF   EECON2               ; write H'AA'
                    BSF     EECON1, WR           ; start erase (CPU stall)
                    BSF     INTCON, GIE          ; re-enable interrupts
```

## 7.5 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.

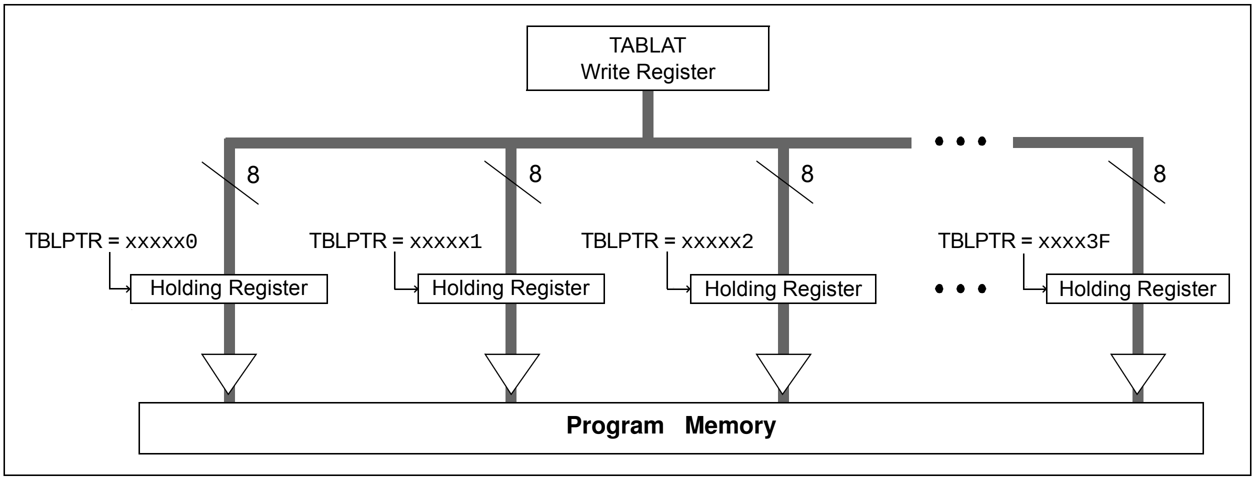
Since the Table Latch (TABLAT) is only a single byte, the `TBLWT` instruction may need to be executed 64 times for each programming operation (if WPROG = `0`). All of the table write operations will essentially be short writes because only the holding registers are written. At the end of updating the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

The on-chip timer controls the write time. The write/erase voltages are generated by an on-chip charge pump, rated to operate over the voltage range of the device.

> **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.
>
> **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.

**FIGURE 7-5: TABLE WRITES TO FLASH PROGRAM MEMORY**




### 7.5.1 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 H'55' to EECON2.
10. Write H'AA' to EECON2.
11. Set the WR bit. This will begin the write cycle.
12. The CPU will stall for duration of the write for $T_{IW}$ (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 7-3 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.

**EXAMPLE 7-3: WRITING TO FLASH PROGRAM MEMORY**

```
                  MOVLW   CODE_ADDR_UPPER       ; Load TBLPTR with the base address
                  MOVWF   TBLPTRU               ; of the memory block, minus 1
                  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   H'55'
                  MOVWF   EECON2                ; write H'55'
                  MOVLW   H'AA'
                  MOVWF   EECON2                ; write H'AA'
                  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   H'55'
    Required      MOVWF   EECON2                ; write H'55'
    Sequence      MOVLW   H'AA'
                  MOVWF   EECON2                ; write H'AA'
                  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
```

### 7.5.2 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 H'55' to EECON2.
6. Write H'AA' to EECON2.
7. Set the WR bit. This will begin the write cycle.
8. The CPU will stall for duration of the write for $T_{IW}$ (see Parameter D133A).
9. Re-enable interrupts.

**EXAMPLE 7-4: 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
                MOVLW   H'55'
    Required    MOVWF   EECON2                ; write H'55'
    Sequence    MOVLW   H'AA'
                MOVWF   EECON2                ; write H'AA'
                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
```

#### 7.5.3 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.

#### 7.5.4 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 $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset, during normal operation, the user can check the WRERR bit and rewrite the location(s) as needed.

## 7.6 Flash Program Operation During Code Protection

See **Section 25.6 "Program Verification and Code Protection"** for details on code protection of Flash program memory.

**TABLE 7-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| TBLPTRU | — | — | bit 21 | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | 61 |
| TBPLTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 61 |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 61 |
| TABLAT | Program Memory Table Latch | | | | | | | | 61 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| EECON2 | Program Memory Control Register 2 (not a physical register) | | | | | | | | 63 |
| EECON1 | — | — | WPROG | FREE | WRERR | WREN | WR | — | 63 |

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

## 8.0 EXTERNAL MEMORY BUS

| Note: | The External Memory Bus (EMB) is not implemented on 64-pin devices. |
|---|---|

The External Memory Bus 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 8-1.

**TABLE 8-1: PIC18F87J11 FAMILY EXTERNAL BUS – I/O PORT FUNCTIONS**

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

**Note:** 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.

## 8.1 External Memory Bus Control

The operation of the interface is controlled by the MEMCON register (Register 8-1). 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 8.5 "Program Memory Modes and the External Memory Bus"**.

The WAITx bits allow for the addition of Wait states to external memory operations. The use of these bits is discussed in **Section 8.3 "Wait States"**.

The WMx 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 8.6 "16-Bit Data Width Modes"**. These bits have no effect when an 8-bit Data Width mode is selected.

The MEMCON register (see Register 8-1) 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 25-9).

### REGISTER 8-1: MEMCON: EXTERNAL MEMORY BUS CONTROL REGISTER

| R/W-0 | U-0 | R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| EBDIS | — | WAIT1 | WAIT0 | — | — | WM1 | WM0 |
| 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 **EBDIS**: External Bus Disable bit
1 = External bus is enabled when the microcontroller accesses external memory; otherwise, all external bus drivers are mapped as I/O ports
0 = External bus is always enabled, I/O ports are disabled

bit 6 **Unimplemented**: Read as '0'

bit 5-4 **WAIT<1:0>:** 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 **WM<1:0>:** TBLWT Operation with 16-Bit Data Bus Width Select bits
1x = Word Write mode: TABLAT word output, $\overline{\text{WRH}}$ is active when TABLAT is written
01 = Byte Select mode: TABLAT data is copied on both MSB and LSB, $\overline{\text{WRH}}$ and ($\overline{\text{UB}}$ or $\overline{\text{LB}}$) will activate
00 = Byte Write mode: TABLAT data is copied on both MSB and LSB, $\overline{\text{WRH}}$ or $\overline{\text{WRL}}$ will activate

## 8.2 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 EMB<1:0> 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 (EMB<1:0> = `01`) disables A<19:16> 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 EMBx 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 8-2.

### 8.2.1 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 the 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.

### 8.2.2 21-BIT ADDRESSING

As an extension of the 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 $\overline{UB}$ and $\overline{LB}$ 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 8.6.3 "16-Bit Byte Select Mode"** and **Section 8.7 "8-Bit Data Width Mode"**.

**TABLE 8-2: ADDRESS AND DATA LINES FOR DIFFERENT ADDRESS AND DATA WIDTHS**

| Data Width | Address Width | Multiplexed Data and Address Lines (and Corresponding Ports) | Address Only Lines (and Corresponding Ports) | Ports Available for I/O |
|---|---|---|---|---|
| 8-Bit | 12-Bit | AD<7:0> (PORTD<7:0>) | AD<11:8> (PORTE<3:0>) | PORTE<7:4>, All of PORTH |
| | 16-Bit | | AD<15:8> (PORTE<7:0>) | All of PORTH |
| | 20-Bit | | A<19:16>, AD<15:8> (PORTH<3:0>, PORTE<7:0>) | — |
| 16-Bit | 16-Bit | AD<15:0> (PORTD<7:0>, PORTE<7:0>) | — | All of PORTH |
| | 20-Bit | | A<19:16> (PORTH<3:0>) | — |

## 8.3 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 WAIT<1:0> 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).

## 8.4 Port Pin Weak Pull-ups

With the exception of the upper address lines, A<19:16> 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, REPU 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').

## 8.5 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 $\overline{\text{CE}}$, $\overline{\text{OE}}$, $\overline{\text{WRH}}$, $\overline{\text{WRL}}$, $\overline{\text{UB}}$ and $\overline{\text{LB}}$ 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; A<19:16> (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.

## 8.6 16-Bit Data Width Modes

In 16-Bit Data Width mode, the External Memory Interface (EMI) 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 WM<1:0> 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 Address Latch 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 ($\overline{\text{OE}}$) will enable both bytes of program memory at once to form a 16-bit instruction word. The Chip Enable signal ($\overline{\text{CE}}$) 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.

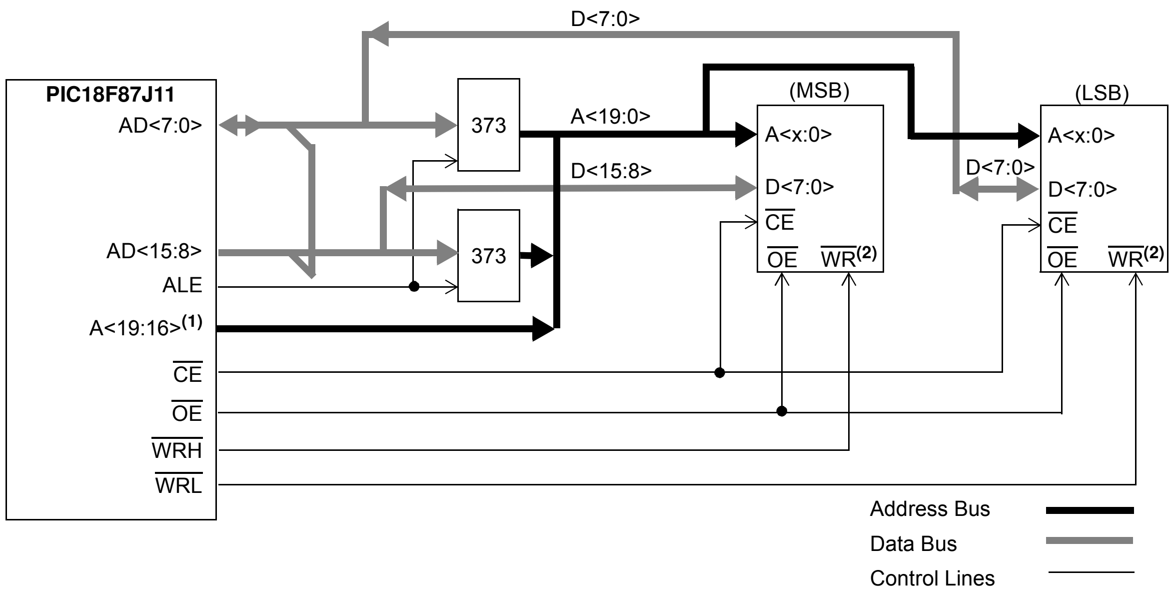
In Byte Select mode, JEDEC standard Flash memories will require BA0 for the byte address line and one I/O line to select between Byte and Word mode. The other 16-bit modes do not need BA0. JEDEC standard static RAM memories will use the $\overline{\text{UB}}$ or $\overline{\text{LB}}$ signals for byte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

### 8.6.1 16-BIT BYTE WRITE MODE

Figure 8-1 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 AD<15:0> bus. The appropriate $\overline{\text{WRH}}$ or $\overline{\text{WRL}}$ control line is strobed on the LSb of the TBLPTR.

**FIGURE 8-1: 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 7.1 "Table Reads and Table Writes"**.

### 8.6.2 16-BIT WORD WRITE MODE

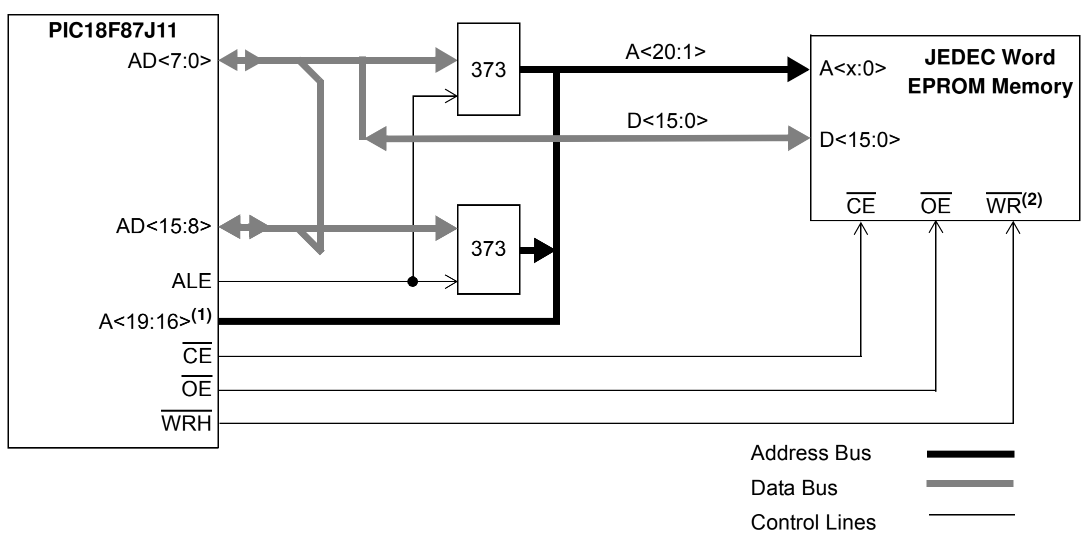
Figure 8-2 shows an example of 16-Bit Word Write mode for PIC18F87J11 family devices. This mode is used for word-wide memories which include some of the EPROM and Flash type memories. This mode allows opcode fetches and table reads from all forms of 16-bit memory and table writes to any type of word-wide 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 AD<15:0> bus. The contents of the holding latch are presented on the lower byte of the AD<15:0> 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 8-2: 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 7.1 "Table Reads and Table Writes"**.

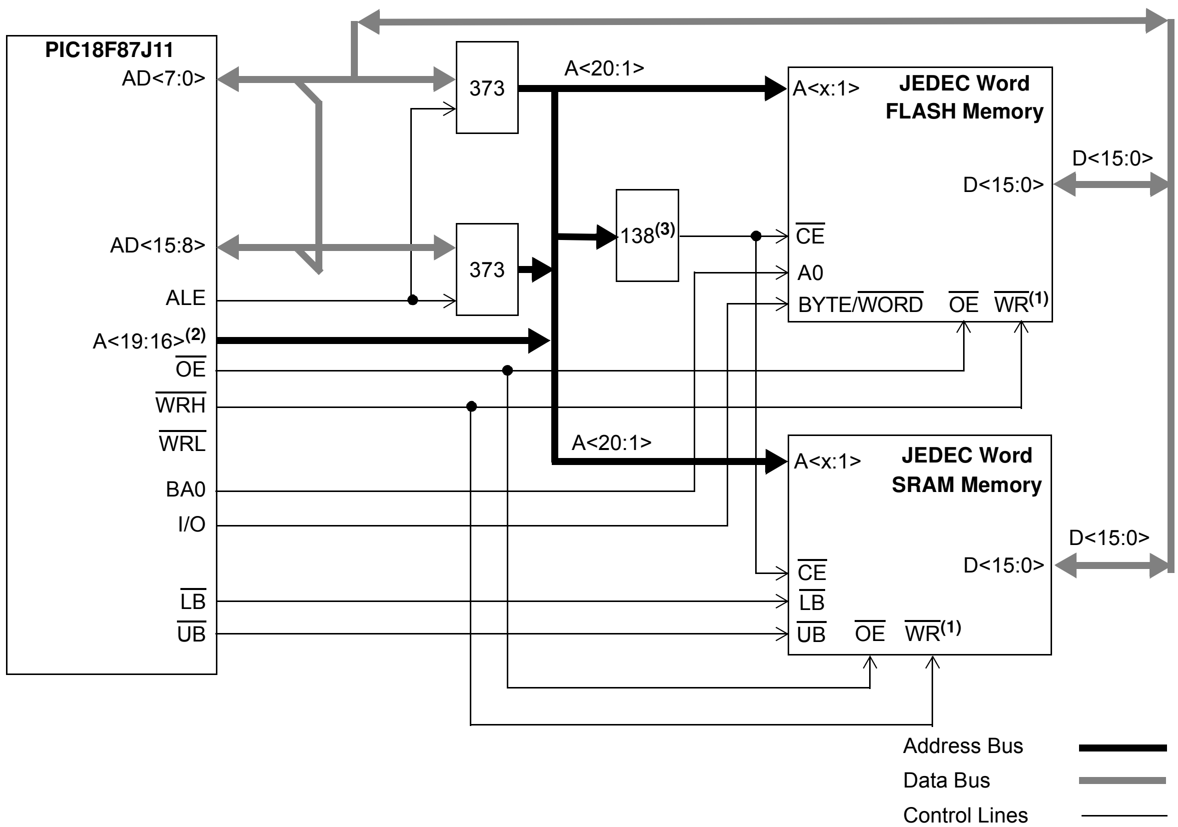
### 8.6.3 16-BIT BYTE SELECT MODE

Figure 8-3 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 AD<15:0> 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 8-3: 16-BIT BYTE SELECT MODE EXAMPLE**




**Note 1:** This signal only applies to table writes. See **Section 7.1 "Table Reads and Table Writes"**.

**2:** Upper order address lines are used only for 20-bit address width.

**3:** Demultiplexing is only required when multiple memory devices are accessed.

### 8.6.4 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 8-4 and Figure 8-5.

**FIGURE 8-4: EXTERNAL MEMORY BUS TIMING FOR TBLRD (EXTENDED MICROCONTROLLER MODE)**

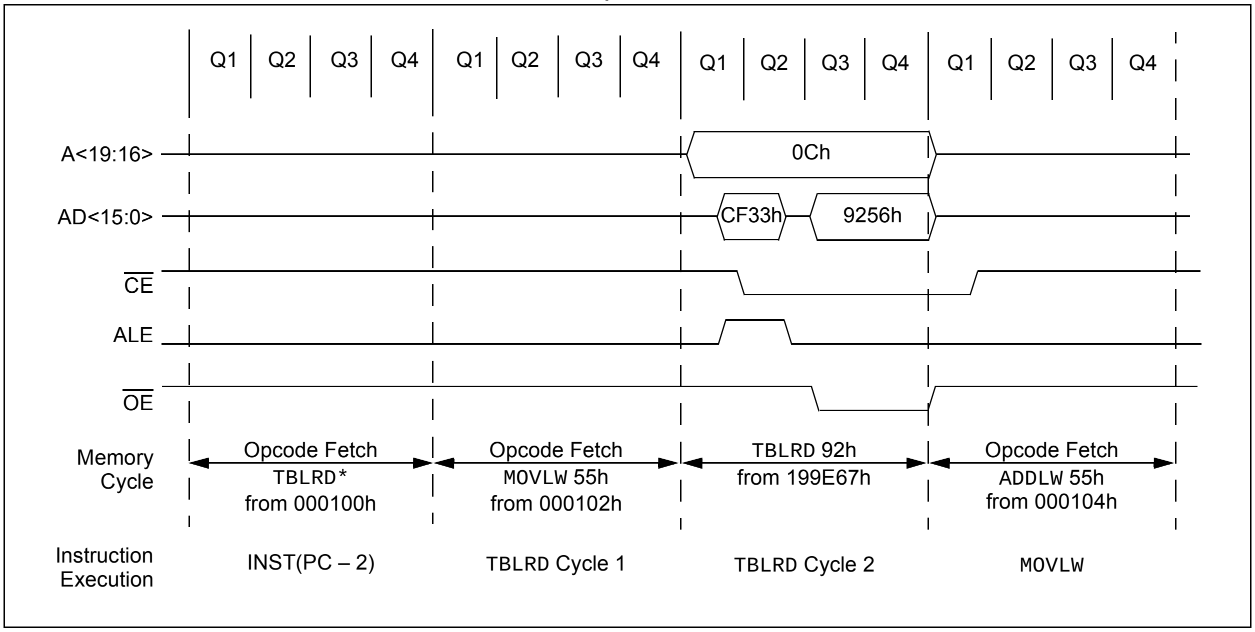
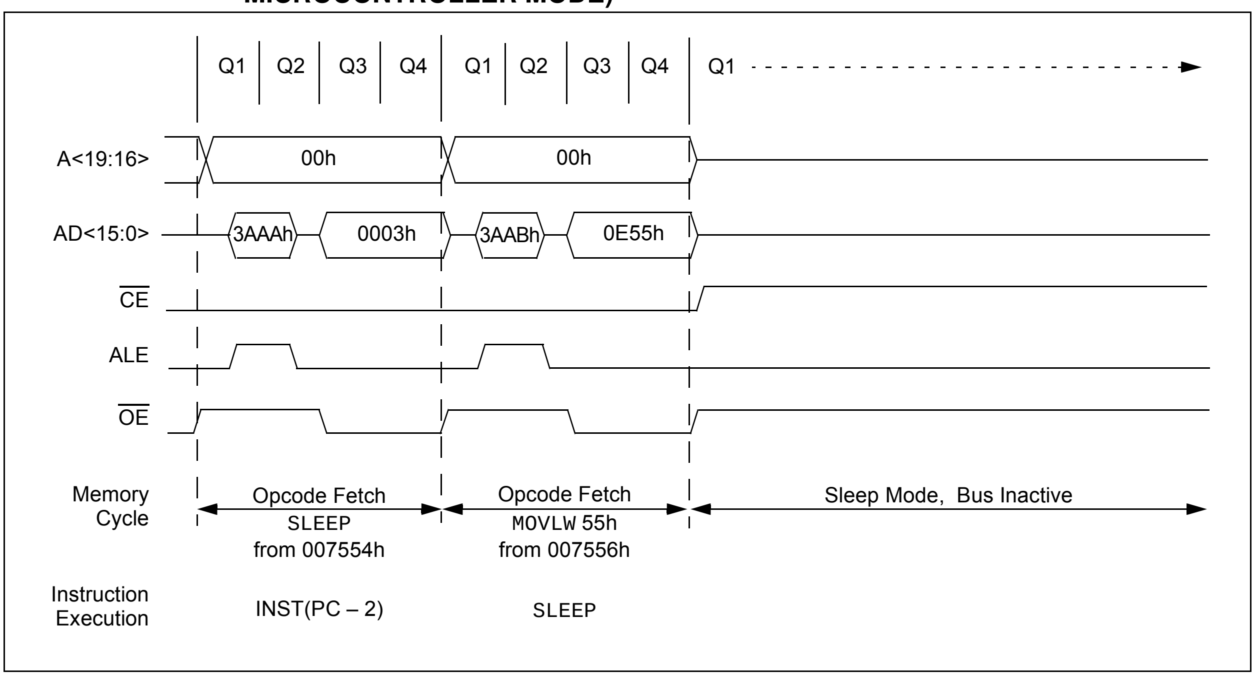



**FIGURE 8-5: EXTERNAL MEMORY BUS TIMING FOR SLEEP (EXTENDED MICROCONTROLLER MODE)**



## 8.7 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.

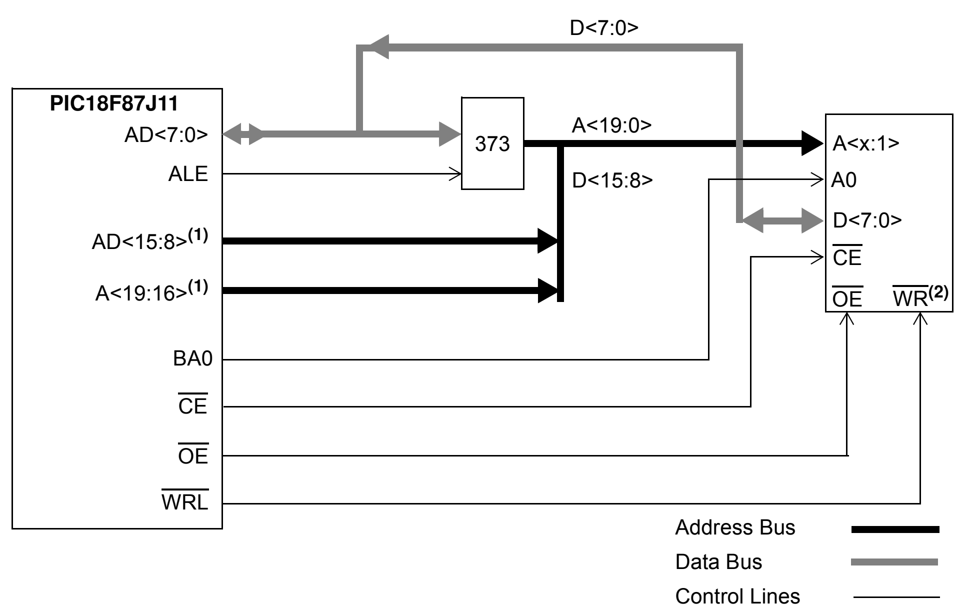
Figure 8-6 shows an example of 8-Bit Multiplexed mode for 80-pin devices. This mode is used for a single 8-bit memory connected for 16-bit operation. The instructions will be fetched as two 8-bit bytes on a shared data/address bus. The two bytes are sequentially fetched within one instruction cycle (TCY). Therefore, the designer must choose external memory devices according to timing calculations based on 1/2 TCY (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 AD<15:0> bus. The appropriate level of the BA0 control line is strobed on the LSb of the TBLPTR.

**FIGURE 8-6: 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 7.1 "Table Reads and Table Writes"**.

### 8.7.1 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 8-7 and Figure 8-8.

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

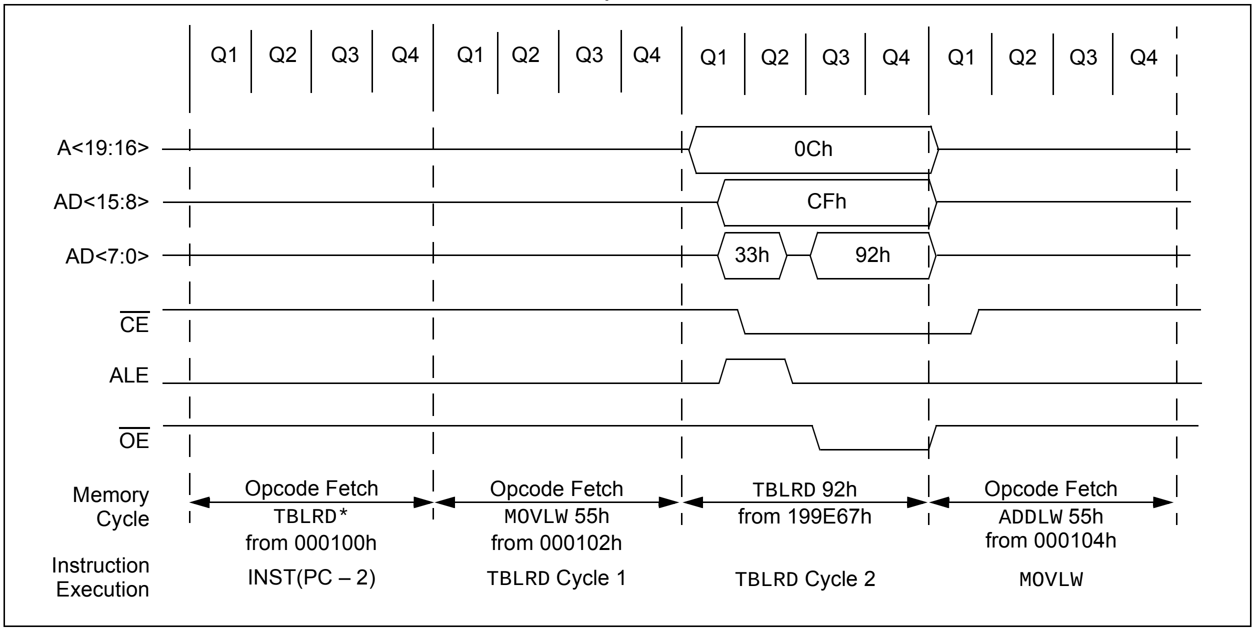
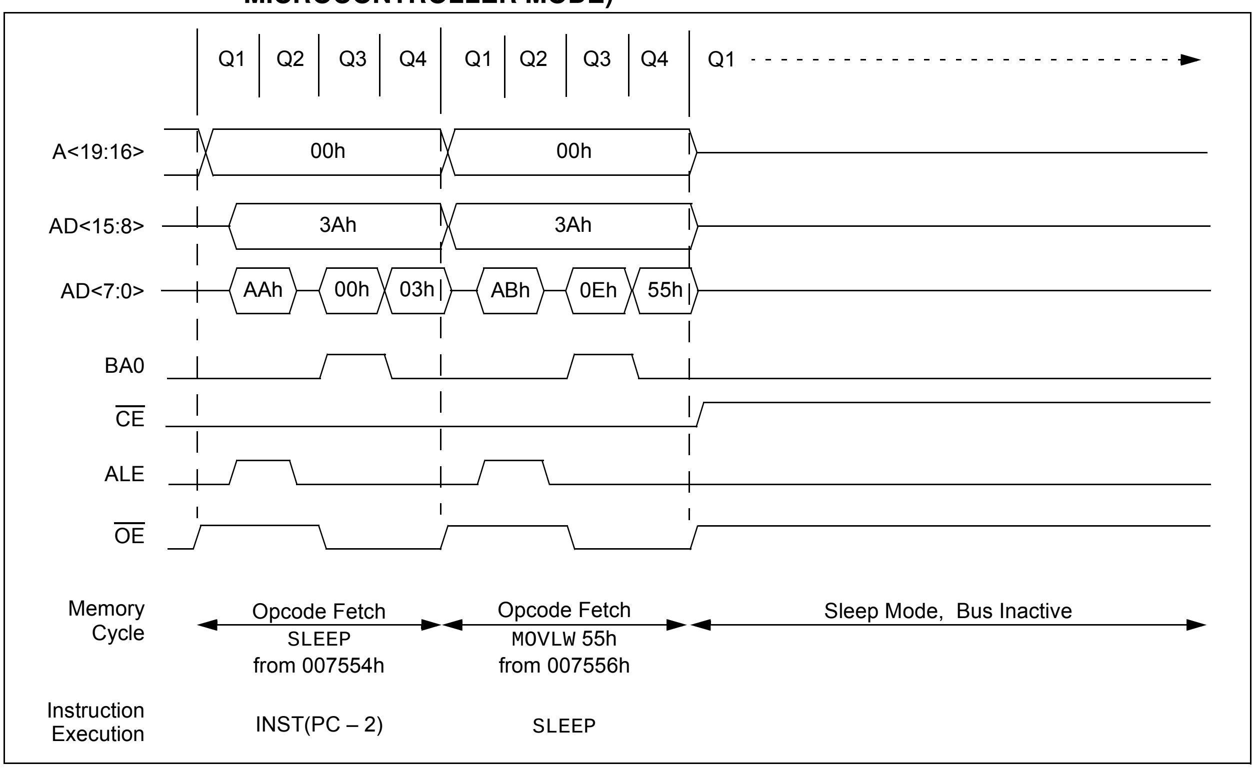



**FIGURE 8-8: EXTERNAL MEMORY BUS TIMING FOR SLEEP (EXTENDED MICROCONTROLLER MODE)**



## 8.8 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 $\overline{CE}$, $\overline{LB}$ and $\overline{UB}$ pins, which are held at logic high.

**NOTES:**

# 9.0 8 x 8 HARDWARE MULTIPLIER

## 9.1 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 9-1.

## 9.2 Operation

Example 9-1 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 9-2 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.

**EXAMPLE 9-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE**

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

**EXAMPLE 9-2: 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
```

**TABLE 9-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS**

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

Example 9-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 9-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES3:RES0).

### EQUATION 9-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3{:}RES0 &= ARG1H{:}ARG1L \bullet ARG2H{:}ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

### EXAMPLE 9-3: 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
    MOVF    PRODL, 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
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
```

Example 9-4 shows the sequence to do a 16 x 16 signed multiply. Equation 9-2 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 9-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3{:}RES0 &= ARG1H{:}ARG1L \bullet ARG2H{:}ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H\langle 7\rangle \bullet ARG1H{:}ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H\langle 7\rangle \bullet ARG2H{:}ARG2L \bullet 2^{16})
\end{aligned}
$$

### EXAMPLE 9-4: 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
    MOVF    PRODL, 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
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    BTFSS   ARG2H, 7        ; ARG2H:ARG2L neg?
    BRA     SIGN_ARG1       ; no, check ARG1
    MOVF    ARG1L, W        ;
    SUBWF   RES2            ;
    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
   :
```

## 10.0 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>), along with the GIEH bit, 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 (ISR), 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.

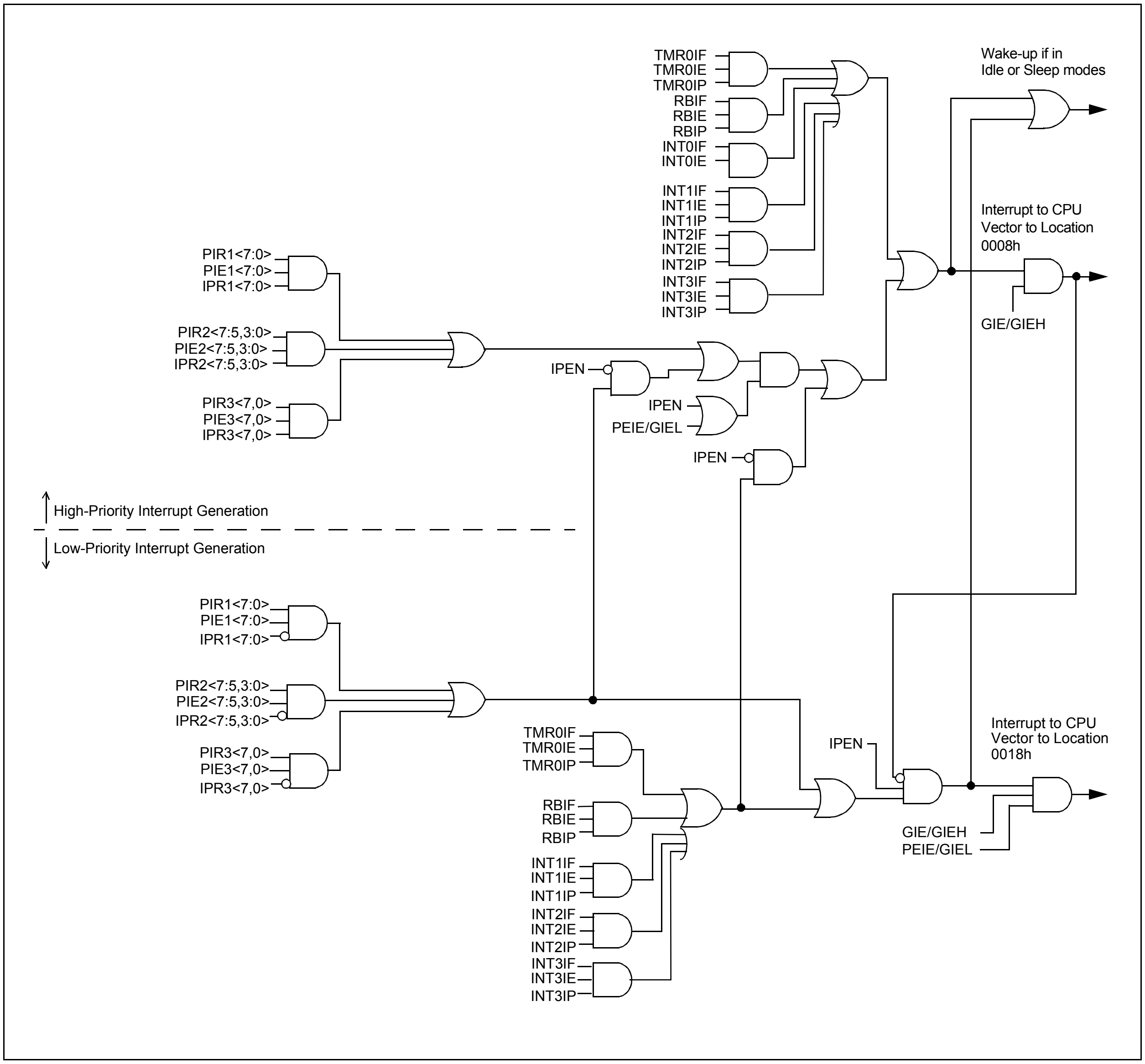
The "Return from Interrupt" instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used) which re-enables interrupts.

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 10-1: PIC18F87J11 FAMILY INTERRUPT LOGIC**



## 10.1 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 10-1: INTCON: INTERRUPT CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-x |
|---|---|---|---|---|---|---|---|
| GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF(1) |
| 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 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high-priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low-priority peripheral interrupts (if GIEH = 1)
0 = Disables all low-priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit(1)
1 = At least one of the RB<7:4> pins changed state (must be cleared in software)
0 = None of the RB<7:4> pins have changed state

**Note 1:** A mismatch condition will continue to set this bit. Reading PORTB, and then waiting one additional instruction cycle, will end the mismatch condition and allow the bit to be cleared.

**REGISTER 10-2: INTCON2: INTERRUPT CONTROL REGISTER 2**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | INTEDG3 | TMR0IP | INT3IP | RBIP |
| 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 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
- 1 = All PORTB pull-ups are disabled
- 0 = PORTB pull-ups are enabled by individual port TRIS 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 10-3: INTCON3: INTERRUPT CONTROL REGISTER 3

| R/W-1 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| INT2IP | INT1IP | INT3IE | INT2IE | INT1IE | INT3IF | INT2IF | INT1IF |
| 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 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **INT3IE:** INT3 External Interrupt Enable bit
1 = Enables the INT3 external interrupt
0 = Disables the INT3 external interrupt

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

bit 2 **INT3IF:** INT3 External Interrupt Flag bit
1 = The INT3 external interrupt occurred (must be cleared in software)
0 = The INT3 external interrupt did not occur

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

**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.

## 10.2 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 10-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

| R/W-0 | R/W-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF |
| 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 **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 10-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2**

| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| OSCFIF | CM2IF | CM1IF | — | BCL1IF | LVDIF | TMR3IF | CCP2IF |
| 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 **OSCFIF:** 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 = $V_{DDCORE}$ 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 10-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3**

| R/W-0 | R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF |
| 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 **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.

## 10.3 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 10-7: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE |
| 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 **PMPIE:** Parallel Master Port Read/Write Interrupt Enable bit
1 = Enables the PM read/write interrupt
0 = Disables the PM read/write interrupt

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

bit 5 **RC1IE:** EUSART1 Receive Interrupt Enable bit
1 = Enables the EUSART1 receive interrupt
0 = Disables the EUSART1 receive interrupt

bit 4 **TX1IE:** EUSART1 Transmit Interrupt Enable bit
1 = Enables the EUSART1 transmit interrupt
0 = Disables the EUSART1 transmit interrupt

bit 3 **SSP1IE:** MSSP1 Interrupt Enable bit
1 = Enables the MSSP1 interrupt
0 = Disables the MSSP1 interrupt

bit 2 **CCP1IE:** ECCP1 Interrupt Enable bit
1 = Enables the ECCP1 interrupt
0 = Disables the ECCP1 interrupt

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

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

**REGISTER 10-8: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2**

| R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| OSCFIE | CM2IE | CM1IE | — | BCL1IE | LVDIE | TMR3IE | CCP2IE |
| 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 **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 10-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE |
| 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 **SSP2IE:** MSSP2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **BCL2IE:** Bus Collision Interrupt Enable bit (MSSP2 module)
1 = Enabled
0 = Disabled

bit 5 **RC2IE:** EUSART2 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **TX2IE:** EUSART2 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **TMR4IE:** TMR4 to PR4 Match Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **CCP5IE:** CCP5 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **CCP4IE**: CCP4 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **CCP3IE:** ECCP3 Interrupt Enable bit
1 = Enabled
0 = Disabled

## 10.4 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 10-10: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP |
| 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 **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

### REGISTER 10-11: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

| R/W-1 | R/W-1 | R/W-1 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| OSCFIP | CM2IP | CM1IP | — | BCL1IP | LVDIP | TMR3IP | CCP2IP |
| 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 **OSCFIP:** Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **CM2IP:** Comparator 2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **C12IP:** Comparator 1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **BCL1IP:** Bus Collision Interrupt Priority bit (MSSP1 module)
1 = High priority
0 = Low priority

bit 2 **LVDIP:** Low-Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP:** TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP2IP:** ECCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

### REGISTER 10-12: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP |
| 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 **SSP2IP:** MSSP2 Interrupt Priority bit
`1` = High priority
`0` = Low priority

bit 6 **BCL2IP:** Bus Collision Interrupt Priority bit (MSSP2 module)
`1` = High priority
`0` = Low priority

bit 5 **RC2IP:** EUSART2 Receive Interrupt Priority bit
`1` = High priority
`0` = Low priority

bit 4 **TX2IP:** EUSART2 Transmit Interrupt Priority bit
`1` = High priority
`0` = Low priority

bit 3 **TMR4IE:** TMR4 to PR4 Interrupt Priority bit
`1` = High priority
`0` = Low priority

bit 2 **CCP5IP:** CCP5 Interrupt Priority bit
`1` = High priority
`0` = Low priority

bit 1 **CCP4IP:** CCP4 Interrupt Priority bit
`1` = High priority
`0` = Low priority

bit 0 **CCP3IP:** ECCP3 Interrupt Priority bit
`1` = High priority
`0` = Low priority

## 10.5 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 10-13: RCON: RESET CONTROL REGISTER

| R/W-0 | U-0 | R/W-1 | R/W-1 | R-1 | R-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | — | $\overline{\text{CM}}$ | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |
| 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 **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 **$\overline{\text{CM}}$:** Configuration Mismatch Flag bit
For details of bit operation, see Register 5-1.

bit 4 **$\overline{\text{RI}}$:** RESET Instruction Flag bit
For details of bit operation, see Register 5-1.

bit 3 **$\overline{\text{TO}}$:** Watchdog Timer Time-out Flag bit
For details of bit operation, see Register 5-1.

bit 2 **$\overline{\text{PD}}$:** Power-Down Detection Flag bit
For details of bit operation, see Register 5-1.

bit 1 **$\overline{\text{POR}}$:** Power-on Reset Status bit
For details of bit operation, see Register 5-1.

bit 0 **$\overline{\text{BOR}}$:** Brown-out Reset Status bit
For details of bit operation, see Register 5-1.

## 10.6 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, with the exception of Deep Sleep, which can only be woken from INT0. 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.

## 10.7 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 TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP (INTCON2<2>). See **Section 13.0 "Timer0 Module"** for further details on the Timer0 module.

## 10.8 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>).

## 10.9 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 6.3 "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 10-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

**EXAMPLE 10-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM**

```
MOVWF    W_TEMP                       ; W_TEMP is in virtual bank
MOVFF    STATUS, STATUS_TEMP          ; STATUS_TEMP located anywhere
MOVFF    BSR, BSR_TEMP                ; BSR_TMEP located anywhere
;
; USER ISR CODE
;
MOVFF    BSR_TEMP, BSR                ; Restore BSR
MOVF     W_TEMP, W                    ; Restore WREG
MOVFF    STATUS_TEMP, STATUS          ; Restore STATUS
```

## 11.0 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)

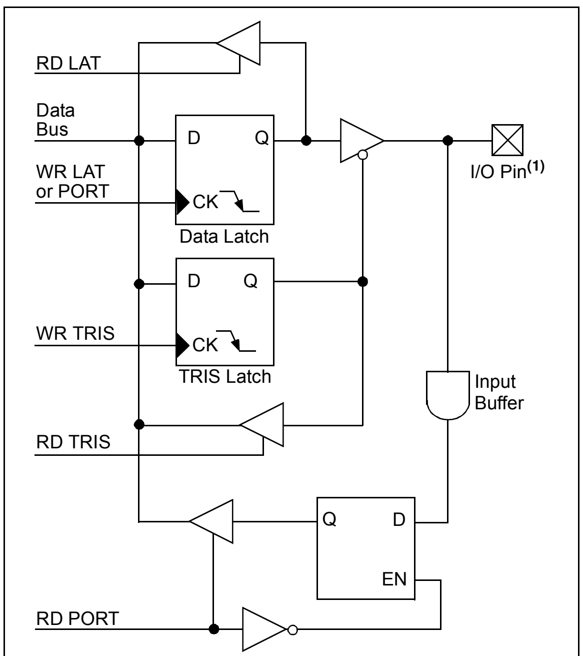
Reading the PORT register reads the current status of the pins, whereas writing to the PORT register writes to the Output Latch (LAT) register.

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 11-1.

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




**Note 1:** I/O pins have diode protection to VDD and VSS.

## 11.1 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 VDD input levels.

### 11.1.1 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 VDD. Voltage excursions beyond VDD on these pins should be avoided.

Table 11-1 summarizes the input capabilities. Refer to **Section 28.0 "Electrical Characteristics"** for more details.

**TABLE 11-1: INPUT VOLTAGE LEVELS**

| Port or Pin | Tolerated Input | Description |
|---|---|---|
| PORTA<7:0> | VDD | Only VDD input levels are tolerated. |
| PORTC<1:0> | | |
| PORTF<6:1> | | |
| PORTH<7:4>(1) | | |
| PORTB<7:0> | 5.5V | Tolerates input levels above VDD, useful for most standard logic. |
| PORTC<7:2> | | |
| PORTD<7:0> | | |
| PORTE<7:0> | | |
| PORTF<7> | | |
| PORTG<4:0> | | |
| PORTH<3:0>(1) | | |
| PORTJ<7:0>(1) | | |

**Note 1:** These ports are not available on 64-pin devices.

### 11.1.2 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 11-2 summarizes the output capabilities of the ports. Refer to the **"Absolute Maximum Ratings"** in **Section 28.0 "Electrical Characteristics"** for more details.

**TABLE 11-2: OUTPUT DRIVE LEVELS**

| Port | Drive | Description |
|---|---|---|
| PORTA | Minimum | Intended for indication. |
| PORTF | | |
| PORTG | | |
| PORTH(1) | | |
| PORTD | Medium | Sufficient drive levels for external memory interfacing as well as indication. |
| PORTE | | |
| PORTJ(1) | | |
| PORTB | High | Suitable for direct LED drive levels. |
| PORTC | | |

**Note 1:** These ports are not available on 64-pin devices.

### 11.1.3 PULL-UP CONFIGURATION

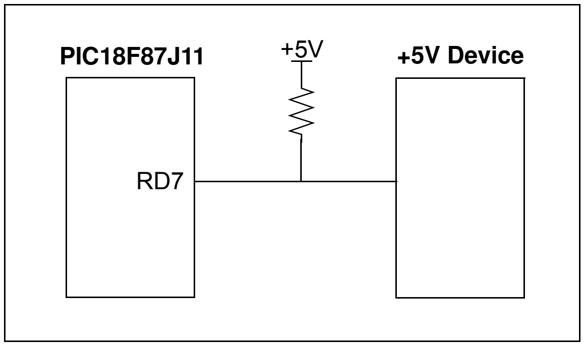
Four of the I/O ports (PORTB, PORTD, PORTE and PORTJ) implement configurable weak pull-ups on all pins. These are internal pull-ups that allow floating digital input signals to be pulled to a consistent level, without the use of external resistors.

The pull-ups are enabled with a single bit for each of the ports: $\overline{\text{RBPU}}$ (INTCON2<7>) for PORTB, and RDPU, REPU and RJPU (PORTG<7:5>) for the other ports.

### 11.1.4 INTERFACING TO A 5V SYSTEM

Though the VDDMAX of the PIC18F87J11 family is 3.6V, these devices are still capable of interfacing with 5V systems, even if the VIH of the target system is above 3.6V. This is accomplished by adding a pull-up resistor to the port pin (Figure 11-2), clearing the LAT bit for that pin and manipulating the corresponding TRIS bit (Figure 11-1) to either allow the line to be pulled high, or to drive the pin low. Only port pins that are tolerant of voltages up to 5.5V can be used for this type of interface (refer to **Section 11.1.1 "Input Pins and Voltage Considerations"**).

**FIGURE 11-2: +5V SYSTEM HARDWARE INTERFACE**




**EXAMPLE 11-1: COMMUNICATING WITH THE +5V SYSTEM**

```
BCF   LATD, 7   ; set up LAT register so
                ; changing TRIS bit will
                ; drive line low
BCF   TRISD, 7  ; send a 0 to the 5V system
BSF   TRISD, 7  ; send a 1 to the 5V system
```

### 11.1.5 OPEN-DRAIN OUTPUTS

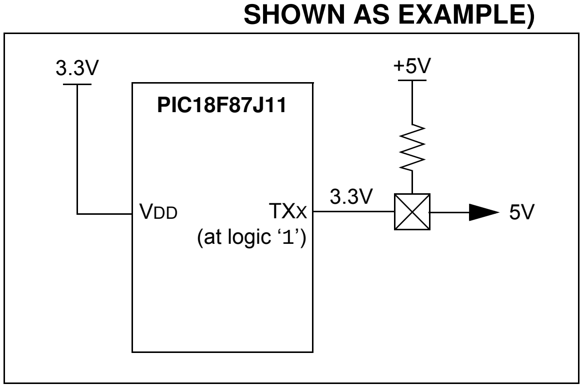
The output pins for several peripherals are also equipped with a configurable, open-drain output option. This allows the peripherals to communicate with external digital logic operating at a higher voltage level, without the use of level translators.

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 11-1, Register 11-2 and Register 11-3). 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 6.3.4.1 "Shared Address SFRs"** for more details). The ODCON registers are accessed by setting the ADSHR bit (WDTCON<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 11-3). When a digital logic high signal is output, it is pulled up to the higher voltage level.

**FIGURE 11-3: USING THE OPEN-DRAIN OUTPUT (EUSARTx SHOWN AS EXAMPLE)**




### 11.1.6 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 11-4). 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 (WDTCON<4>).

### REGISTER 11-1: ODCON1: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 1

| U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | CCP5OD | CCP4OD | ECCP3OD | ECCP2OD | ECCP1OD |
| 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 **Unimplemented:** Read as ‘0’

bit 4-3 **CCP5OD:CCP4OD:** CCPx Open-Drain Output Enable bits
1 = Open-drain output is on the CCPx pin (Capture/PWM modes) is enabled
0 = Open-drain output is disabled

bit 2-0 **ECCP3OD:ECCP1OD:** ECCPx Open-Drain Output Enable bits
1 = Open-drain output is on the ECCPx pin (Capture mode) is enabled
0 = Open-drain output is disabled

### REGISTER 11-2: ODCON2: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 2

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | U2OD | U1OD |
| 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 7-2 **Unimplemented:** Read as ‘0’

bit 1-0 **U2OD:U1OD:** EUSARTx Open-Drain Output Enable bits
1 = Open-drain output is on the TXx pin is enabled
0 = Open-drain output is disabled

### REGISTER 11-3: ODCON3: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 3

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | SPI2OD | SPI1OD |
| 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 7-2 **Unimplemented:** Read as ‘0’

bit 1-0 **SPI2OD:SPI1OD:** SPI Open-Drain Output Enable bits
1 = Open-drain output is on the SDOx pin is enabled
0 = Open-drain output is disabled

### REGISTER 11-4: PADCFG1: I/O PAD CONFIGURATION CONTROL REGISTER

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | PMPTTL |
| 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 7-1 **Unimplemented:** Read as ‘0’

bit 0 **PMPTTL:** PMP Module TTL Input Buffer Select bit
1 = PMP module uses TTL input buffers
0 = PMP module uses Schmitt Trigger input buffers

## 11.2 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 PCFGx control bits in the ANCON0 register.

> **Note 1:** RA5 (RA5/PMD4/AN4) is multiplexed as an analog input in all devices and Parallel Master Port data in 80-pin devices.
>
> **2:** RA5 and RA<3:0> are configured as analog inputs on any Reset and are read as '0'. RA4 is configured as a digital input.

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.

OSC2/CLKO/RA6 and OSC1/CLKI/RA7 normally serve as the external circuit connections for the external (primary) oscillator circuit (HS and HSPLL Oscillator modes), or the external clock input (EC and ECPLL Oscillator modes). In these cases, RA6 and RA7 are not available as digital I/O, and their corresponding TRIS and LAT bits are read as '0'.

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 11-3. For these pins, the corresponding PORTA, TRISA and LATA bits are only defined when the pins are configured as I/O.

**TABLE 11-3: FUNCTION OF RA<7:6> IN INTIO AND INTPLL MODES**

| Oscillator Mode (FOSC<2:0> 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 11-2: INITIALIZING PORTA**

```
CLRF    PORTA          ; Initialize PORTA by
                       ; clearing output
                       ; data latches
CLRF    LATA           ; Alternate method to
                       ; clear  data latches
BSF     WDTCON,ADSHR   ; Enable write/read to
                       ; the shared SFR
MOVLW   1Fh            ; Configure A/D
MOVWF   ANCON0         ; for digital inputs
BCF     WDTCON,ADSHR   ; Disable write/read
                       ; to the shared SFR
MOVLW   H'CF'          ; Value used to
                       ; initialize
                       ; data direction
MOVWF   TRISA          ; Set RA<3:0> as inputs,
                       ; RA<5:4> as outputs
```

**TABLE 11-4: PORTA FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RA0/AN0 | RA0 | 0 | O | DIG | LATA<0> data output; not affected by analog input. |
| | | 1 | I | TTL | PORTA<0> data input; disabled when analog input is enabled. |
| | AN0 | 1 | I | ANA | A/D Input Channel 0. Default input configuration on POR; does not affect digital output. |
| RA1/AN1 | RA1 | 0 | O | DIG | LATA<1> data output; not affected by analog input. |
| | | 1 | I | TTL | PORTA<1> data input; disabled when analog input is enabled. |
| | AN1 | 1 | I | ANA | A/D Input Channel 1. Default input configuration on POR; does not affect digital output. |
| RA2/AN2/VREF- | RA2 | 0 | O | DIG | LATA<2> data output; not affected by analog input. Disabled when CVREF output is enabled. |
| | | 1 | I | TTL | PORTA<2> data input. Disabled when analog functions enabled; disabled when CVREF output is enabled. |
| | AN2 | 1 | I | ANA | A/D Input Channel 2. Default input configuration on POR; not affected by analog output. |
| | VREF- | 1 | I | ANA | A/D low reference voltage input. |
| RA3/AN3/VREF+ | RA3 | 0 | O | DIG | LATA<3> data output; not affected by analog input. |
| | | 1 | I | TTL | PORTA<3> data input; disabled when analog input is enabled. |
| | AN3 | 1 | I | ANA | A/D Input Channel 3. Default input configuration on POR. |
| | VREF+ | 1 | I | ANA | A/D high reference voltage input. |
| RA4/PMD5/T0CKI/ | RA4 | 0 | O | DIG | LATA<4> data output. |
| | | 1 | I | ST | PORTA<4> data input; default configuration on POR. |
| | PMD5(1) | x | O | DIG | Parallel Master Port data output. |
| | | x | I | TTL | Parallel Master Port data output. |
| | T0CKI | x | I | ST | Timer0 clock input. |
| RA5/PMD4/AN4 | RA5 | 0 | O | DIG | LATA<5> data output; not affected by analog input. |
| | | 1 | I | TTL | PORTA<5> data input; disabled when analog input is enabled. |
| | PMD4(1) | x | O | DIG | Parallel Master Port data output. |
| | | x | I | TTL | Parallel Master Port data output. |
| | AN4 | 1 | I | ANA | A/D Input Channel 4. Default configuration on POR. |
| OSC2/CLKO/RA6 | OSC2 | x | O | ANA | Main oscillator feedback output connection (HS and HSPLL modes). |
| | CLKO | x | O | DIG | System cycle clock output, FOSC/4 (EC, ECPLL, INTIO1 and INTPLL1 modes). |
| | RA6 | 0 | O | DIG | LATA<6> data output; disabled when FOSC2 Configuration bit is set. |
| | | 1 | I | TTL | PORTA<6> data input; disabled when FOSC2 Configuration bit is set. |
| OSC1/CLKI/RA7 | OSC1 | x | I | ANA | Main oscillator input connection (HS and HSPLL modes). |
| | CLKI | x | I | ANA | Main external clock source input (EC and ECPLL modes). |
| | RA7 | 0 | O | DIG | LATA<7> data output; disabled when FOSC2 Configuration bit is set. |
| | | 1 | I | TTL | PORTA<7> data input; disabled when FOSC2 Configuration bit is set. |

**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.

**TABLE 11-5: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTA | RA7[(1)] | RA6[(1)] | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 65 |
| LATA | LATA7[(1)] | LATA6[(1)] | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | 64 |
| TRISA | TRISA7[(1)] | TRISA6[(1)] | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 64 |
| ANCON0[(2)] | PCFG7 | PCFG6 | — | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 63 |

**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 the default SFR at this address; available only when WDTCON<4> = 1.

## 11.3 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, $\overline{\text{RBPU}}$ (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 (RB<7:4>) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB<7:4> pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB<7:4>) are compared with the old value latched on the last read of PORTB. The “mismatch” outputs of RB<7:4> 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:

1. Any read or write of PORTB (except with the `MOVFF (ANY), PORTB` instruction).
2. Wait one instruction cycle (such as executing a `NOP` instruction).
3. 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 after a one $T_{CY}$ delay.

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 11-3: INITIALIZING PORTB**

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method to clear
                ; output data latches
MOVLW   H'CF'   ; Value used to initialize
                ; data direction
MOVWF   TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

**TABLE 11-6: PORTB FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RB0/INT0/FLT0 | RB0 | 0 | O | DIG | LATB<0> data output. |
| | | 1 | I | TTL | PORTB<0> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. |
| | INT0 | 1 | I | ST | External Interrupt 0 input. |
| | FLT0 | 1 | I | ST | Enhanced PWM Fault input (ECCP1 module); enabled in software. |
| RB1/INT1/ PMA4 | RB1 | 0 | O | DIG | LATB<1> data output. |
| | | 1 | I | TTL | PORTB<1> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. |
| | INT1 | 1 | I | ST | External Interrupt 1 input. |
| | PMA4 | x | O | — | Parallel Master Port address out. |
| RB2/INT2/ PMA3 | RB2 | 0 | O | DIG | LATB<2> data output. |
| | | 1 | I | TTL | PORTB<2> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. |
| | INT2 | 1 | I | ST | External Interrupt 2 input. |
| | PMA3 | x | O | — | Parallel Master Port address out. |
| RB3/INT3/ PMA2/ECCP2/ P2A | RB3 | 0 | O | DIG | LATB<3> data output. |
| | | 1 | I | TTL | PORTB<3> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. |
| | INT3 | 1 | I | ST | External Interrupt 3 input. |
| | PMA2 | x | O | — | Parallel Master Port address out. |
| | ECCP2(1) | 0 | O | DIG | ECCP2 compare output and CCP2 PWM output; takes priority over port data. |
| | | 1 | I | ST | ECCP2 capture input. |
| | P2A(1) | 0 | O | DIG | ECCP2 Enhanced PWM output, Channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data. |
| RB4/KBI0/ PMA1 | RB4 | 0 | O | DIG | LATB<4> data output. |
| | | 1 | I | TTL | PORTB<4> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. |
| | KBI0 | | I | TTL | Interrupt-on-pin change. |
| | PMA1 | x | O | — | Parallel Master Port address out. |
| RB5/KBI1/ PMA0 | RB5 | 0 | O | DIG | LATB<5> data output. |
| | | 1 | I | TTL | PORTB<5> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. |
| | KBI1 | | I | TTL | Interrupt-on-pin change. |
| | PMA0 | x | O | — | Parallel Master Port address out. |
| RB6/KBI2/PGC | RB6 | 0 | O | DIG | LATB<6> data output. |
| | | 1 | I | TTL | PORTB<6> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. |
| | KBI2 | 1 | I | TTL | Interrupt-on-pin change. |
| | PGC | x | I | ST | Serial execution (ICSP™) clock input for ICSP and ICD operation.(2) |
| RB7/KBI3/PGD | RB7 | 0 | O | DIG | LATB<7> data output. |
| | | 1 | I | TTL | PORTB<7> data input; weak pull-up when $\overline{\text{RBPU}}$ bit is cleared. |
| | KBI3 | 1 | I | TTL | Interrupt-on-pin change. |
| | PGD | x | O | DIG | Serial execution data output for ICSP and ICD operation.(2) |
| | | x | I | ST | Serial execution data input for ICSP and ICD operation.(2) |

**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); the default assignment is RC1.

**2:** All other pin functions are disabled when ICSP™ or ICD is enabled.

**TABLE 11-7: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | 65 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | 64 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 64 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| INTCON2 | $\overline{RBPU}$ | INTEDG0 | INTEDG1 | INTEDG2 | INTEDG3 | TMR0IP | INT3IP | RBIP | 61 |
| INTCON3 | INT2IP | INT1IP | INT3IE | INT2IE | INT1IE | INT3IF | INT2IF | INT1IF | 61 |

**Legend:** Shaded cells are not used by PORTB.

## 11.4 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, MSSPx and EUSARTx peripheral functions (Table 11-8). The pins have Schmitt Trigger input buffers. The pins for ECCP, SPI and EUSARTx 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 11.1.3 “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 11-4: INITIALIZING PORTC**

```
CLRF    PORTC   ; Initialize PORTC by
                ; clearing output
                ; data latches
CLRF    LATC    ; Alternate method to clear
                ; output data latches
MOVLW   H'CF'   ; 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 11-8: PORTC FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RC0/T1OSO/T13CKI | RC0 | 0 | O | DIG | LATC<0> data output. |
| | | 1 | I | ST | PORTC<0> data input. |
| | T1OSO | x | O | ANA | Timer1 oscillator output; enabled when Timer1 oscillator is enabled. Disables digital I/O. |
| | T13CKI | 1 | I | ST | Timer1/Timer3 counter input. |
| RC1/T1OSI/ECCP2/P2A | RC1 | 0 | O | DIG | LATC<1> data output. |
| | | 1 | I | ST | PORTC<1> data input. |
| | T1OSI | x | I | ANA | Timer1 oscillator input; enabled when Timer1 oscillator is enabled. Disables digital I/O. |
| | ECCP2(1) | 0 | O | DIG | ECCP2 compare output and ECCP2 PWM output; takes priority over port data. |
| | | 1 | I | ST | ECCP2 capture input. |
| | P2A(1) | 0 | O | DIG | ECCP2 Enhanced PWM output, Channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data. |
| RC2/ECCP1/P1A | RC2 | 0 | O | DIG | LATC<2> data output. |
| | | 1 | I | ST | PORTC<2> data input. |
| | ECCP1 | 0 | O | DIG | ECCP1 compare output and ECCP1 PWM output; takes priority over port data. |
| | | 1 | I | ST | ECCP1 capture input. |
| | P1A | 0 | O | DIG | ECCP1 Enhanced PWM output, Channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data. |
| RC3/SCK1/SCL1 | RC3 | 0 | O | DIG | LATC<3> data output. |
| | | 1 | I | ST | PORTC<3> data input. |
| | SCK1 | 0 | O | DIG | SPI clock output (MSSP1 module); takes priority over port data. |
| | | 1 | I | ST | SPI clock input (MSSP1 module). |
| | SCL1 | 0 | O | DIG | I$^2$C™ clock output (MSSP1 module); takes priority over port data. |
| | | 1 | I | ST | I$^2$C clock input (MSSP1 module); input type depends on module setting. |
| RC4/SDI1/SDA1 | RC4 | 0 | O | DIG | LATC<4> data output. |
| | | 1 | I | ST | PORTC<4> data input. |
| | SDI1 | 1 | I | ST | SPI data input (MSSP1 module). |
| | SDA1 | 1 | O | DIG | I$^2$C data output (MSSP1 module); takes priority over port data. |
| | | 1 | I | ST | I$^2$C data input (MSSP1 module); input type depends on module setting. |
| RC5/SDO1 | RC5 | 0 | O | DIG | LATC<5> data output. |
| | | 1 | I | ST | PORTC<5> data input. |
| | SDO1 | 0 | O | DIG | SPI data output (MSSP1 module); takes priority over port data. |
| RC6/TX1/CK1 | RC6 | 0 | O | DIG | LATC<6> data output. |
| | | 1 | I | ST | PORTC<6> data input. |
| | TX1 | 1 | O | DIG | Synchronous serial data output (EUSART1 module); takes priority over port data. |
| | CK1 | 1 | O | DIG | Synchronous serial data input (EUSART1 module). User must configure as an input. |
| | | 1 | I | ST | Synchronous serial clock input (EUSART1 module). |
| RC7/RX1/DT1 | RC7 | 0 | O | DIG | LATC<7> data output. |
| | | 1 | I | ST | PORTC<7> data input. |
| | RX1 | 1 | I | ST | Asynchronous serial receive data input (EUSART1 module). |
| | DT1 | 1 | O | DIG | Synchronous serial data output (EUSART1 module); takes priority over port data. |
| | | 1 | I | ST | Synchronous serial data input (EUSART1 module). User must configure as an input. |

**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 the CCP2MX Configuration bit is set.

**TABLE 11-9: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | 65 |
| LATC | LATC7 | LATBC6 | LATC5 | LATCB4 | LATC3 | LATC2 | LATC1 | LATC0 | 64 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 64 |

## 11.5 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 (EMI). 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 (AD<7:0>). 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 12.0 "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 11-5: INITIALIZING PORTD**

```
CLRF    PORTD   ; Initialize PORTD by
                ; clearing output
                ; data latches
CLRF    LATD    ; Alternate method to clear
                ; output data latches
MOVLW   H'CF'   ; 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 11-10: PORTD FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RD0/AD0/PMD0 | RD0 | 0 | O | DIG | LATD<0> data output. |
| | | 1 | I | ST | PORTD<0> data input. |
| | AD0[2] | x | O | DIG | External Memory Interface, Address/Data Bit 0 output.[1] |
| | | x | I | TTL | External Memory Interface, Data Bit 0 input.[1] |
| | PMD0[3] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| RD1/AD1/PMD1 | RD1 | 0 | O | DIG | LATD<1> data output. |
| | | 1 | I | ST | PORTD<1> data input. |
| | AD1[2] | x | O | DIG | External Memory Interface, Address/Data bit 1 output.[1] |
| | | x | I | TTL | External Memory Interface, Data Bit 1 input.[1] |
| | PMD1[3] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| RD2/AD2/PMD2 | RD2 | 0 | O | DIG | LATD<2> data output. |
| | | 1 | I | ST | PORTD<2> data input. |
| | AD2[2] | x | O | DIG | External Memory Interface, Address/Data Bit 2 output.[1] |
| | | x | I | TTL | External Memory Interface, Data Bit 2 input.[1] |
| | PMD2[3] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| RD3/AD3/PMD3 | RD3 | 0 | O | DIG | LATD<3> data output. |
| | | 1 | I | ST | PORTD<3> data input. |
| | AD3[2] | x | O | DIG | External Memory Interface, Address/Data Bit 3 output.[1] |
| | | x | I | TTL | External Memory Interface, Data Bit 3 input.[1] |
| | PMD3[3] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| RD4/AD4/PMD4/SDO2 | RD4 | 0 | O | DIG | LATD<4> data output. |
| | | 1 | I | ST | PORTD<4> data input. |
| | AD4[2] | x | O | DIG | External Memory Interface, Address/Data Bit 4 output.[1] |
| | | x | I | TTL | External Memory Interface, Data Bit 4 input.[1] |
| | PMD4[3] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| | SDO2 | 0 | O | DIG | SPI data output (MSSP2 module); takes priority over port data. |
| RD5/AD5/PMD5/SDI2/SDA2 | RD5 | 0 | O | DIG | LATD<5> data output. |
| | | 1 | I | ST | PORTD<5> data input. |
| | AD5[2] | x | O | DIG | External Memory Interface, Address/Data Bit 5 output.[1] |
| | | x | I | TTL | External Memory Interface, Data Bit 5 input.[1] |
| | PMD5[3] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| | SDI2 | 1 | I | ST | SPI data input (MSSP2 module). |
| | SDA2 | 1 | O | DIG | $I^2C$™ data output (MSSP2 module); takes priority over port data. |
| | | 1 | I | ST | $I^2C$ data input (MSSP2 module); input type depends on module setting. |

**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.

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

**3:** Default configuration for PMP (PMPMX Configuration bit = 1).

**TABLE 11-10: PORTD FUNCTIONS (CONTINUED)**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RD6/AD6/PMD6/SCK2/SCL2 | RD6 | 0 | O | DIG | LATD<6> data output. |
| | | 1 | I | ST | PORTD<6> data input. |
| | AD6[2] | x | O | DIG-3 | External Memory Interface, Address/Data Bit 6 output.[1] |
| | | x | I | TTL | External Memory Interface, Data Bit 6 input.[1] |
| | PMD6[3] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| | SCK2 | 0 | O | DIG | SPI clock output (MSSP2 module); takes priority over port data. |
| | | 1 | I | ST | SPI clock input (MSSP2 module). |
| | SCL2 | 0 | O | DIG | $I^2C$™ clock output (MSSP2 module); takes priority over port data. |
| | | 1 | I | ST | $I^2C$ clock input (MSSP2 module); input type depends on module setting. |
| RD7/AD7/PMD7/$\overline{SS2}$ | RD7 | 0 | O | DIG | LATD<7> data output. |
| | | 1 | I | ST | PORTD<7> data input. |
| | AD7[2] | x | O | DIG | External Memory Interface, Address/Data Bit 7 output.[1] |
| | | x | I | TTL | External Memory Interface, Data Bit 7 input.[1] |
| | PMD7[3] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| | $\overline{SS2}$ | x | I | TTL | Slave select input for MSSP2 module. |

**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.

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

**3:** Default configuration for PMP (PMPMX Configuration bit = 1).

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

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | 65 |
| LATD | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | 64 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 64 |
| PORTG | RDPU | REPU | RJPU[1] | RG4 | RG3 | RG2 | RG1 | RG0 | 65 |

**Legend:** Shaded cells are not used by PORTD.

**Note 1:** Unimplemented on 64-pin devices, read as '0'.

## 11.6 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 (AD<15:8>). 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, the RE7 pin 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 3.6 "Reference Clock Output"**.

**EXAMPLE 11-6: INITIALIZING PORTE**

```
CLRF    PORTE    ; Initialize PORTE by
                 ; clearing output
                 ; data latches
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 11-12: PORTE FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RE0/AD8/PMRD/P2D | RE0 | 0 | O | DIG | LATE<0> data output. |
| | | 1 | I | ST | PORTE<0> data input. |
| | AD8(3) | x | O | DIG | External Memory Interface, Address/Data Bit 8 output.(2) |
| | | x | I | TTL | External Memory Interface, Data Bit 8 input.(2) |
| | PMRD(5) | x | O | DIG | Parallel Master Port read strobe pin. |
| | | x | I | TTL | Parallel Master Port read pin. |
| | P2D | 0 | O | DIG | ECCP2 Enhanced PWM output, Channel D; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| RE1/AD9/PMWR/P2C | RE1 | 0 | O | DIG | LATE<1> data output. |
| | | 1 | I | ST | PORTE<1> data input. |
| | AD9(3) | x | O | DIG | External Memory Interface, Address/Data Bit 9 output.(2) |
| | | x | I | TTL | External Memory Interface, Data Bit 9 input.(2) |
| | PMWR(5) | x | O | DIG | Parallel Master Port write strobe pin. |
| | | x | I | TTL | Parallel Master Port write pin. |
| | P2C | 0 | O | DIG | ECCP2 Enhanced PWM output, Channel C; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| RE2/AD10/PMBE/P2B | RE2 | 0 | O | DIG | LATE<2> data output. |
| | | 1 | I | ST | PORTE<2> data input. |
| | AD10(3) | x | O | DIG | External Memory Interface, Address/Data Bit 10 output.(2) |
| | | x | I | TTL | External Memory Interface, Data Bit 10 input.(2) |
| | PMBE(5) | x | O | DIG | Parallel Master Port byte enable. |
| | P2B | 0 | O | DIG | ECCP2 Enhanced PWM output, Channel B; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| RE3/AD11/PMA13/P3C/REFO | RE3 | 0 | O | DIG | LATE<3> data output. |
| | | 1 | I | ST | PORTE<3> data input. |
| | AD11(3) | x | O | DIG | External Memory Interface, Address/Data Bit 11 output.(2) |
| | | x | I | TTL | External Memory Interface, Data Bit 11 input.(2) |
| | PMA13 | x | O | DIG | Parallel Master Port address. |
| | P3C(1) | 0 | O | DIG | ECCP3 Enhanced PWM output, Channel C; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| | REFO | x | O | DIG | Reference output clock. |
| RE4/AD12/PMA12/P3B | RE4 | 0 | O | DIG | LATE<4> data output. |
| | | 1 | I | ST | PORTE<4> data input. |
| | AD12(3) | x | O | DIG | External Memory Interface, Address/Data Bit 12 output.(2) |
| | | x | I | TTL | External Memory Interface, Data Bit 12 input.(2) |
| | PMA12 | x | O | DIG | Parallel Master Port address. |
| | P3B(1) | 0 | O | DIG | ECCP3 Enhanced PWM output, Channel B; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |

**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 11-12: PORTE FUNCTIONS (CONTINUED)

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RE5/AD13/ PMA11/P1C | RE5 | 0 | O | DIG | LATE<5> data output. |
| | | 1 | I | ST | PORTE<5> data input. |
| | AD13(3) | x | O | DIG | External Memory Interface, Address/Data Bit 13 output.(2) |
| | | x | I | TTL | External Memory Interface, Data Bit 13 input.(2) |
| | PMA11 | x | O | DIG | Parallel Master Port address. |
| | P1C(1) | 0 | O | DIG | ECCP1 Enhanced PWM output, Channel C; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| RE6/AD14/ PMA10/P1B | RE6 | 0 | O | DIG | LATE<6> data output. |
| | | 1 | I | ST | PORTE<6> data input. |
| | AD14(3) | x | O | DIG | External Memory Interface, Address/Data Bit 14 output.(2) |
| | | x | I | TTL | External Memory Interface, Data Bit 14 input.(2) |
| | PMA10 | x | O | DIG | Parallel Master Port address. |
| | P1B(1) | 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/ PMA9/ECCP2/ P2A | RE7 | 0 | O | DIG | LATE<7> data output. |
| | | 1 | I | ST | PORTE<7> data input. |
| | AD15(3) | x | O | DIG | External Memory Interface, Address/Data Bit 15 output.(2) |
| | | x | I | TTL | External Memory Interface, Data Bit 15 input.(2) |
| | PMA9 | x | O | DIG | Parallel Master Port address. |
| | ECCP2(4) | 0 | O | DIG | ECCP2 compare output and ECCP2 PWM output; takes priority over port data. |
| | | 1 | I | ST | ECCP2 capture input. |
| | P2A(4) | 0 | O | DIG | ECCP2 Enhanced PWM output, Channel A; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |

**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 11-13: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTE | RE7 | RE6 | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 | 65 |
| LATE | LATE7 | LATE6 | LATE5 | LATE4 | LATE3 | LATE2 | LATE1 | LATE0 | 64 |
| TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 | 64 |
| PORTG | RDPU | REPU | RJPU(1) | RG4 | RG3 | RG2 | RG1 | RG0 | 65 |

**Legend:** Shaded cells are not used by PORTE.

**Note 1:** Unimplemented on 64-pin devices, read as '0'.

## 11.7 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, the RF<6:1> pins are configured as analog inputs and are read as '0'. |
| **2:** | To configure PORTF as digital I/O, set the corresponding bits in the ANCON0 and ANCON1 registers. |

When Configuration bit, PMPMX = 0, PORTF is multiplexed with the Parallel Master Port data. This multiplexing is available only in 80-pin devices.

### EXAMPLE 11-7: 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
MOVLW  C0h           ; make RF1:RF2 digital
MOVWF  ANCON0        ;
MOVLW  0Fh           ; make RF<6:3> digital
MOVWF  ANCON1        ;
BCF    WDTCON,ADSHR  ; Disable write/read to
                     ; the shared SFR
MOVLW  CEh           ;
MOVWF  TRISF         ; Set RF5:RF4 as outputs,
                     ; RF<7:6>,<3:1> as inputs
```

### TABLE 11-14: PORTF FUNCTIONS

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RF1/AN6/C2OUT | RF1 | 0 | O | DIG | LATF<1> data output; not affected by analog input. |
| | | 1 | I | ST | PORTF<1> data input; disabled when analog input is enabled. |
| | AN6 | 1 | I | ANA | A/D Input Channel 6. Default configuration on POR. |
| | C2OUT | x | O | DIG | Comparator 2 output. |
| RF2/PMA5/AN7//C1OUT | RF2 | 0 | O | DIG | LATF<2> data output; not affected by analog input. |
| | | 1 | I | ST | PORTF<2> data input; disabled when analog input is enabled. |
| | PMA5 | x | O | DIG | Parallel Master Port address. |
| | AN7 | 1 | I | ANA | A/D Input Channel 7. Default configuration on POR. |
| | C1OUT | x | O | DIG | Comparator 1 output. |
| RF3/AN8/C2INB | RF3 | 0 | O | DIG | LATF<3> data output; not affected by analog input. |
| | | 1 | I | ST | PORTF<3> data input; disabled when analog input is enabled. |
| | AN8 | 1 | I | ANA | A/D Input Channel 8. Default configuration on POR. |
| | C2INB | x | I | ANA | Comparator 2 Input B. |
| RF4/AN9/C2INA | RF4 | 0 | O | DIG | LATF<4> data output; not affected by analog input. |
| | | 1 | I | ST | PORTF<4> data input; disabled when analog input is enabled. |
| | AN9 | 1 | I | ANA | A/D Input Channel 9. Default configuration on POR. |
| | C2INA | x | I | ANA | Comparator 2 Input A. |
| RF5/PMD2/AN10/C1INB/CVREF | RF5 | 0 | O | DIG | LATF<5> data output; not affected by analog input. Disabled when CVREF output is enabled. |
| | | 1 | I | ST | PORTF<5> data input; disabled when analog input is enabled. Disabled when CVREF output is enabled. |
| | PMD2[1] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| | AN10 | 1 | I | ANA | A/D Input Channel 10 and Comparator C1+ input. Default input configuration on POR. |
| | C1INB | x | I | ANA | Comparator 1 Input B. |
| | CVREF | x | O | ANA | Comparator voltage reference output. Enabling this feature disables digital I/O. |
| RF6/PMD1/AN11/C1INA | RF6 | 0 | O | DIG | LATF<6> data output; not affected by analog input. |
| | | 1 | I | ST | PORTF<6> data input; disabled when analog input is enabled. |
| | PMD1[1] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| | AN11 | 1 | I | ANA | A/D Input Channel 11 and Comparator C1- input. Default input configuration on POR; does not affect digital output. |
| | C1INA | x | I | ANA | Comparator 1 Input A. |
| RF7/PMD0/$\overline{\text{SS1}}$ | RF7 | 0 | O | DIG | LATF<7> data output. |
| | | 1 | I | ST | PORTF<7> data input. |
| | PMD0[1] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| | $\overline{\text{SS1}}$ | 1 | I | TTL | Slave select input for MSSP1 module. |

**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 11-15: SUMMARY OF REGISTERS ASSOCIATED WITH PORTF**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | — | 65 |
| LATF | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | — | 64 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — | 64 |
| ANCON0(1) | PCFG7 | PCFG6 | — | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 63 |
| ANCON1(1) | PCFG15 | PCFG14 | PCFG13 | PCFG12 | PCFG11 | PCFG10 | PCFG9 | PCFG8 | 63 |

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

**Note 1:** Configuration SFR overlaps with the default SFR at this address; available only when WDTCON<4> = 1.

## 11.8 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 11-16). 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 11-8: INITIALIZING PORTG**

```
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 11-16: PORTG FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RG0/PMA8/ ECCP3/P3A | RG0 | 0 | O | DIG | LATG<0> data output. |
| | | 1 | I | ST | PORTG<0> data input. |
| | PMA8 | x | O | DIG | Parallel Master Port address. |
| | ECCP3 | | O | DIG | ECCP3 compare and PWM output; takes priority over port data. |
| | | | I | ST | ECCP3 capture input. |
| | P3A | 0 | O | DIG | ECCP3 Enhanced PWM output, Channel A; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| RG1/PMA7/ TX2/CK2 | RG1 | 0 | O | DIG | LATG<1> data output. |
| | | 1 | I | ST | PORTG<1> data input. |
| | PMA7 | x | O | DIG | Parallel Master Port address. |
| | TX2 | 1 | O | DIG | Synchronous serial data output (EUSART2 module); takes priority over port data. |
| | CK2 | 1 | O | DIG | Synchronous serial data input (EUSART2 module). User must configure as an input. |
| | | 1 | I | ST | Synchronous serial clock input (EUSART2 module). |
| RG2/PMA6/ RX2/DT2 | RG2 | 0 | O | DIG | LATG<2> data output. |
| | | 1 | I | ST | PORTG<2> data input. |
| | PMA6 | x | O | DIG | Parallel Master Port address. |
| | RX2 | 1 | I | ST | Asynchronous serial receive data input (EUSART2 module). |
| | DT2 | 1 | O | DIG | Synchronous serial data output (EUSART2 module); takes priority over port data. |
| | | 1 | I | ST | Synchronous serial data input (EUSART2 module). User must configure as an input. |
| RG3/PMCS1/ CCP4/P3D | RG3 | 0 | O | DIG | LATG<3> data output. |
| | | 1 | I | ST | PORTG<3> data input. |
| | PMCS1 | x | O | DIG | Parallel Master Port Address Chip Select 1 |
| | | x | I | TTL | Parallel Master Port Address Chip Select 1. |
| | CCP4 | 0 | O | DIG | CCP4 compare output and CCP4 PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP4 capture input. |
| | P3D | 0 | O | DIG | ECCP3 Enhanced PWM output, Channel D; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| RG4/PMCS2/ CCP5/P1D | RG4 | 0 | O | DIG | LATG<4> data output. |
| | | 1 | I | ST | PORTG<4> data input. |
| | PMCS2 | x | O | DIG | Parallel Master Port Address Chip Select 2 |
| | CCP5 | 0 | O | DIG | CCP5 compare output and CCP5 PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP5 capture input. |
| | P1D | 0 | O | DIG | ECCP1 Enhanced PWM output, Channel D; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |

**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).

**TABLE 11-17: SUMMARY OF REGISTERS ASSOCIATED WITH PORTG**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTG | RDPU | REPU | RJPU(1) | RG4 | RG3 | RG2 | RG1 | RG0 | 65 |
| LATG | — | — | — | LATG4 | LATG3 | LATG2 | LATG1 | LATG0 | 64 |
| TRISG | — | — | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 | 64 |

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

**Note 1:** Unimplemented on 64-pin devices, read as '0'.

## 11.9 PORTH, LATH and TRISH Registers

| **Note:** | PORTH is available only on 80-pin devices. |
|---|---|

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 11-9: INITIALIZING PORTH**

```
CLRF   PORTH        ; Initialize PORTH by
                    ; clearing output
                    ; data latches
CLRF   LATH         ; Alternate method to
                    ; clear output latches
BSF    WDTCON,ADSHR ; Enable write/read to
                    ; the shared SFR
MOVLW  F0h          ; Configure PORTH as
MOVWF  ANCON1       ; digital I/O
BCF    WDTCON,ADSHR ; Disable write/read to
                    ; the shared SFR
MOVLW  H'CF'        ; Value used to initialize
                    ; data direction
MOVWF  TRISH        ; Set RH<3:0> as inputs
                    ; RH<5:4> as outputs
                    ; RH<7:6> as inputs
```

### TABLE 11-18: PORTH FUNCTIONS

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RH0/A16 | RH0 | 0 | O | DIG | LATH<0> data output. |
| | | 1 | I | ST | PORTH<0> data input. |
| | A16 | x | O | DIG | External Memory Interface, Address Line 16. Takes priority over port data. |
| RH1/A17 | RH1 | 0 | O | DIG | LATH<1> data output. |
| | | 1 | I | ST | PORTH<1> data input. |
| | A17 | x | O | DIG | External Memory Interface, Address Line 17. Takes priority over port data. |
| RH2/A18/PMD7 | RH2 | 0 | O | DIG | LATH<2> data output. |
| | | 1 | I | ST | PORTH<2> data input. |
| | A18 | x | O | DIG | External Memory Interface, Address Line 18. Takes priority over port data. |
| | PMD7[2] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| RH3/A19/PMD6 | RH3 | 0 | O | DIG | LATH<3> data output. |
| | | 1 | I | ST | PORTH<3> data input. |
| | A19 | x | O | DIG | External Memory Interface, Address Line 19. Takes priority over port data. |
| | PMD6[2] | x | O | DIG | Parallel Master Port data out. |
| | | x | I | TTL | Parallel Master Port data input. |
| RH4/PMD3/AN12/P3C/C2INC | RH4 | 0 | O | DIG | LATH<4> data output. |
| | | 1 | I | ST | PORTH<4> data input. |
| | PMD3[2] | x | I | TTL | Parallel Master Port data out. |
| | | x | O | DIG | Parallel Master Port data input. |
| | AN12 | | I | ANA | A/D Input Channel 12. Default input configuration on POR; does not affect digital output. |
| | P3C[1] | 0 | O | DIG | ECCP3 Enhanced PWM output, Channel C; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| | C2INC | x | I | ANA | Comparator 2 Input C. |
| RH5/PMBE/AN13/P3B/C2IND | RH5 | 0 | O | DIG | LATH<5> data output. |
| | | 1 | I | ST | PORTH<5> data input. |
| | PMBE[2] | x | O | DIG | Parallel Master Port data byte enable. |
| | AN13 | | I | ANA | A/D Input Channel 13. Default input configuration on POR; does not affect digital output. |
| | P3B[1] | 0 | O | DIG | ECCP3 Enhanced PWM output, Channel B; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| | C2IND | x | I | ANA | Comparator 2 Input D. |
| RH6/PMRD/AN14/P1C/C1INC | RH6 | 0 | O | DIG | LATH<6> data output. |
| | | 1 | I | ST | PORTH<6> data input. |
| | PMRD[2] | x | O | DIG | Parallel Master Port read strobe. |
| | | x | I | TTL | Parallel Master Port read in. |
| | AN14 | | I | ANA | A/D Input Channel 14. Default input configuration on POR; does not affect digital output. |
| | P1C[1] | 0 | O | DIG | ECCP1 Enhanced PWM output, Channel C; takes priority over port and PMP data. May be configured for tri-state during Enhanced PWM shutdown events. |
| | C1INC | x | I | ANA | Comparator 1 Input C. |

**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>.

**2:** Alternate PMP configuration when the PMPMX Configuration bit = 0; available on 80-pin devices only.

### TABLE 11-18: PORTH FUNCTIONS (CONTINUED)

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RH7/PMWR/AN15/P1B | RH7 | 0 | O | DIG | LATH<7> data output. |
| | | 1 | I | ST | PORTH<7> data input. |
| | PMWR(2) | x | O | DIG | Parallel Master Port write strobe. |
| | | x | I | TTL | Parallel Master Port write in. |
| | AN15 | | I | ANA | A/D input channel 15. Default input configuration on POR; does not affect digital output. |
| | P1B(1) | 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. |

**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>.

**2:** Alternate PMP configuration when the PMPMX Configuration bit = 0; available on 80-pin devices only.

### TABLE 11-19: SUMMARY OF REGISTERS ASSOCIATED WITH PORTH

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTH(1) | RH7 | RH6 | RH5 | RH4 | RH3 | RH2 | RH1 | RH0 | 64 |
| LATH(1) | LATH7 | LATH6 | LATH5 | LATH4 | LATH3 | LATH2 | LATH1 | LATH0 | 65 |
| TRISH(1) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 | 64 |
| ANCON1(2) | PCFG15 | PCFG14 | PCFG13 | PCFG12 | PCFG11 | PCFG10 | PCFG9 | PCFG8 | 63 |

**Legend:** Shaded cells are not used by PORTH.

**Note 1:** Unimplemented on 64-pin devices, read as '0'.

**2:** Configuration SFR, overlaps with the default SFR at this address; available only when WDTCON<4> = 1.

## 11.10 PORTJ, TRISJ and LATJ Registers

| **Note:** | PORTJ is available only on 80-pin devices. |
|---|---|

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.

| **Note:** | These pins are configured as digital inputs on any device Reset. |
|---|---|

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 11-10: INITIALIZING PORTJ**

```
CLRF    PORTJ   ; Initialize PORTG by
                ; clearing output
                ; data latches
CLRF    LATJ    ; Alternate method to clear
                ; output data latches
MOVLW   H'CF'   ; Value used to initialize
                ; data direction
MOVWF   TRISJ   ; Set RJ3:RJ0 as inputs
                ; RJ5:RJ4 as output
                ; RJ7:RJ6 as inputs
```

**TABLE 11-20: PORTJ FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RJ0/ALE | RJ0 | 0 | O | DIG | LATJ<0> data output. |
| | | 1 | I | ST | PORTJ<0> data input. |
| | ALE | x | O | DIG | External Memory Interface address latch enable control output; takes priority over digital I/O. |
| RJ1/$\overline{OE}$ | RJ1 | 0 | O | DIG | LATJ<1> data output. |
| | | 1 | I | ST | PORTJ<1> data input. |
| | $\overline{OE}$ | x | O | DIG | External Memory Interface output enable control output; takes priority over digital I/O. |
| RJ2/$\overline{WRL}$ | RJ2 | 0 | O | DIG | LATJ<2> data output. |
| | | 1 | I | ST | PORTJ<2> data input. |
| | $\overline{WRL}$ | x | O | DIG | External Memory Bus write low byte control; takes priority over digital I/O. |
| RJ3/$\overline{WRH}$ | RJ3 | 0 | O | DIG | LATJ<3> data output. |
| | | 1 | I | ST | PORTJ<3> data input. |
| | $\overline{WRH}$ | x | O | DIG | External Memory Interface write high byte control output; takes priority over digital I/O. |
| RJ4/BA0 | RJ4 | 0 | O | DIG | LATJ<4> data output. |
| | | 1 | I | ST | PORTJ<4> data input. |
| | BA0 | x | O | DIG | External Memory Interface Byte Address 0 control output; takes priority over digital I/O. |
| RJ5/$\overline{CE}$ | RJ5 | 0 | O | DIG | LATJ<5> data output. |
| | | 1 | I | ST | PORTJ<5> data input. |
| | $\overline{CE}$ | x | O | DIG | External Memory Interface chip enable control output; takes priority over digital I/O. |
| RJ6/$\overline{LB}$ | RJ6 | 0 | O | DIG | LATJ<6> data output. |
| | | 1 | I | ST | PORTJ<6> data input. |
| | $\overline{LB}$ | x | O | DIG | External Memory Interface lower byte enable control output; takes priority over digital I/O. |
| RJ7/$\overline{UB}$ | RJ7 | 0 | O | DIG | LATJ<7> data output. |
| | | 1 | I | ST | PORTJ<7> data input. |
| | $\overline{UB}$ | x | O | DIG | External Memory Interface upper byte enable control output; takes priority over digital I/O. |

**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 11-21: SUMMARY OF REGISTERS ASSOCIATED WITH PORTJ**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| PORTJ(1) | RJ7 | RJ6 | RJ5 | RJ4 | RJ3 | RJ2 | RJ1 | RJ0 | 65 |
| LATJ(1) | LATJ7 | LATJ6 | LATJ5 | LATJ4 | LATJ3 | LATJ2 | LATJ1 | LATJ0 | 64 |
| TRISJ(1) | TRISJ7 | TRISJ6 | TRISJ5 | TRISJ4 | TRISJ3 | TRISJ2 | TRISJ1 | TRISJ0 | 64 |
| PORTG | RDPU | REPU | RJPU(1) | RG4 | RG3 | RG2 | RG1 | RG0 | 65 |

**Legend:** Shaded cells are not used by PORTJ.

**Note 1:** Unimplemented on 64-pin devices, read as '0'.

NOTES:

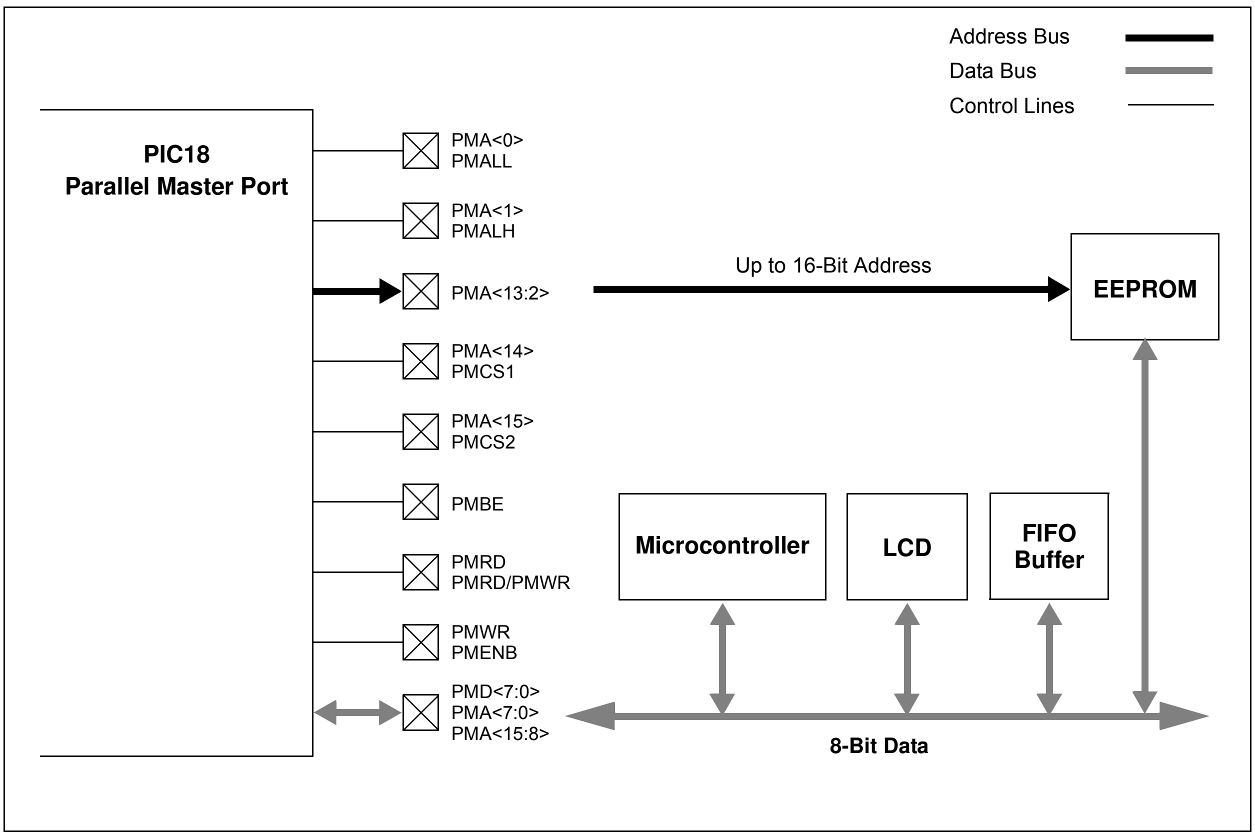
## 12.0 PARALLEL MASTER PORT

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/$\overline{\text{Write}}$ 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 12-1: PMP MODULE OVERVIEW**



## 12.1 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.

### 12.1.1 CONTROL REGISTERS

The eight PMP Control registers are:

- PMCONH and PMCONL
- PMMODEH and PMMODEL
- PMSTATL and PMSTATH
- PMEH and PMEL

The PMCON registers (Register 12-1 and Register 12-2) control basic module operations, including turning the module on or off. They also configure address multiplexing and control strobe configuration.

The PMMODE registers (Register 12-3 and Register 12-4) configure the various Master and Slave Operating modes, the data width and interrupt generation.

The PMEH and PMEL registers (Register 12-5 and Register 12-6) configure the module's operation at the hardware (I/O pin) level.

The PMSTAT registers (Register 12-7 and Register 12-8) provide status flags for the module's input and output buffers, depending on the operating mode.

**REGISTER 12-1: PMCONH: PARALLEL PORT CONTROL HIGH BYTE REGISTER**

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PMPEN | — | PSIDL | ADRMUX1 | ADRMUX0 | PTBEEN | PTWREN | PTRDEN |
| 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 | **PMPEN:** Parallel Master Port Enable bit<br>1 = PMP is enabled<br>0 = PMP is disabled, no off-chip access is performed |
| bit 6 | **Unimplemented:** Read as '0' |
| bit 5 | **PSIDL:** Stop in Idle Mode bit<br>1 = Discontinues module operation when device enters Idle mode<br>0 = Continues module operation in Idle mode |
| bit 4-3 | **ADRMUX<1:0>:** Address/Data Multiplexing Selection bits<br>11 = Reserved<br>10 = All 16 bits of address are multiplexed on PMD<7:0> pins<br>01 = Lower 8 bits of address are multiplexed on PMD<7:0> pins, upper 8 bits are on PMA<15:8><br>00 = Address and data appear on separate pins |
| bit 2 | **PTBEEN:** Byte Enable Port Enable bit (16-bit Master mode)<br>1 = PMBE port is enabled<br>0 = PMBE port is disabled |
| bit 1 | **PTWREN:** Write Enable Strobe Port Enable bit<br>1 = PMWR/PMENB port is enabled<br>0 = PMWR/PMENB port is disabled |
| bit 0 | **PTRDEN:** Read/Write Strobe Port Enable bit<br>1 = PMRD/PMWR port is enabled<br>0 = PMRD/PMWR port is disabled |

## REGISTER 12-2: PMCONL: PARALLEL PORT CONTROL LOW BYTE REGISTER

| R/W-0 | R/W-0 | R/W-0(1) | R/W-0(1) | R/W-0(1) | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CSF1 | CSF0 | ALP | CS2P | CS1P | BEP | WRSP | RDSP |
| 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 7-6 **CSF<1:0>:** Chip Select Function bits
11 = Reserved
10 = PMCS1 and PMCS2 function as chip select
01 = PMCS2 functions as chip select, PMCS1 is used as Address Bit 14 (PMADDRH Address Bit 6)
00 = PMCS2 and PMCS1 are 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 (PMMODEH<1:0> = 00, 01, 10):
1 = Write strobe active-high (PMWR)
0 = Write strobe active-low (PMWR)
For Master mode 1 (PMMODEH<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 (PMMODEH<1:0> = 00, 01, 10):
1 = Read strobe active-high (PMRD)
0 = Read strobe active-low (PMRD)
For Master mode 1 (PMMODEH<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 12-3: PMMODEH: PARALLEL PORT MODE HIGH BYTE REGISTER

| R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| BUSY | IRQM1 | IRQM0 | INCM1 | INCM0 | MODE16 | MODE1 | MODE0 |
| 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 **BUSY:** Busy bit (Master mode only)
1 = Port is busy
0 = Port is not busy

bit 6-5 **IRQM<1:0>:** Interrupt Request Mode bits
11 = Interrupt is 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 is activated
01 = Interrupt is generated at the end of the read/write cycle
00 = No interrupt is generated

bit 4-3 **INCM<1:0>:** Increment Mode bits
11 = PSP read and write buffers auto-increment (Legacy PSP mode only)
10 = Decrements ADDR<15,13:0> by 1 every read/write cycle
01 = Increments 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 **MODE<1:0>:** 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 mode, control signals (PMRD, PMWR, PMCS and PMD<7:0>)

**REGISTER 12-4: PMMODEL: PARALLEL PORT MODE LOW BYTE REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| WAITB1[(1)] | WAITB0[(1)] | WAITM3 | WAITM2 | WAITM1 | WAITM0 | WAITE1[(1)] | WAITE0[(1)] |
| 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 7-6 **WAITB<1:0>:** Data Setup to Read/Write Wait State Configuration bits[(1)]
11 = Data wait of 4 TCY; multiplexed address phase of 4 TCY
10 = Data wait of 3 TCY; multiplexed address phase of 3 TCY
01 = Data wait of 2 TCY; multiplexed address phase of 2 TCY
00 = Data wait of 1 TCY; multiplexed address phase of 1 TCY

bit 5-2 **WAITM<3:0>:** Read to Byte Enable Strobe Wait State Configuration bits
1111 = Wait of additional 15 TCY
...
0001 = Wait of additional 1 TCY
0000 = No additional Wait cycles (operation forced into one TCY)

bit 1-0 **WAITE1:WAITE0:** Data Hold After Strobe Wait State Configuration bits[(1)]
11 = Wait of 4 TCY
10 = Wait of 3 TCY
01 = Wait of 2 TCY
00 = Wait of 1 TCY

**Note 1:** WAITB and WAITE bits are ignored whenever WAITM<3:0> = 0000.

**REGISTER 12-5: PMEH: PARALLEL PORT ENABLE HIGH BYTE REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PTEN15 | PTEN14 | PTEN13 | PTEN12 | PTEN11 | PTEN10 | PTEN9 | PTEN8 |
| 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 7-6 **PTEN<15:14>:** PMCSx Strobe Enable bits
1 = PMA15 and PMA14 function as either PMA<15:14> or PMCS2 and PMCS1
0 = PMA15 and PMA14 function as port I/O

bit 5-0 **PTEN<13:8>:** PMP Address Port Enable bits
1 = PMA<13:8> function as PMP address lines
0 = PMA<13:8> function as port I/O

### REGISTER 12-6: PMEL: PARALLEL PORT ENABLE LOW BYTE REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PTEN7 | PTEN6 | PTEN5 | PTEN4 | PTEN3 | PTEN2 | PTEN1 | PTEN0 |
| 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 7-2 **PTEN<7:2>:** 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 **PTEN<1:0>:** PMALH/PMALL Strobe Enable bits
1 = PMA1 and PMA0 function as either PMA<1:0> or PMALH and PMALL
0 = PMA1 and PMA0 pads function as port I/O

### REGISTER 12-7: PMSTATH: PARALLEL PORT STATUS HIGH BYTE REGISTER

| R-0 | R/W-0 | U-0 | U-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| IBF | IBOV | — | — | IB3F | IB2F | IB1F | IB0F |
| 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 **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 12-8: PMSTATL: PARALLEL PORT STATUS LOW BYTE REGISTER

| R-1 | R/W-0 | U-0 | U-0 | R-1 | R-1 | R-1 | R-1 |
|---|---|---|---|---|---|---|---|
| OBE | OBUF | — | — | OB3E | OB2E | OB1E | OB0E |
| 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 **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 **OB3E:OB0E:** Output Buffer n Status Empty bits
1 = Output buffer is empty (writing data to the buffer will clear this bit)
0 = Output buffer contains data that has not been transmitted

#### 12.1.2 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/PMDOUT1H and PMADDRL/PMDOUT1L
- 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/PMDOUT1x 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 PMDOUT1H and PMDOUT1L 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 12-9.

The PMDOUT2H and PMDOUT2L registers are only used in buffered Slave modes and serve as a buffer for outgoing data.

#### 12.1.3 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 PADCFG1 register.

The PADCFG1 register is one of the shared address SFRs, and has the same address as the TMR2 register. PADCFG1 is accessed by setting the ADSHR bit (WDTCON<4>). Refer to **Section 6.3.4.1 "Shared Address SFRs"** for more information.

**REGISTER 12-9: PMADDRH: PARALLEL PORT ADDRESS REGISTER, HIGH BYTE (MASTER MODES ONLY)[1]**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CS2 | CS1 | ADDR13 | ADDR12 | ADDR11 | ADDR10 | ADDR9 | ADDR8 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = 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
If PMCON<7:6> = `11` or `00`:
Bit functions as ADDR<15>.

bit 6 **CS1:** Chip Select 1 bit
If PMCON<7:6> = `10`:
`1` = Chip Select 1 is active
`0` = Chip Select 1 is inactive
If PMCON<7:6> = `11` or `0x`:
Bit functions as ADDR<14>.

bit 5-0 **ADDR<13:8>:** Destination Address bits

**Note 1:** In Enhanced Slave mode, PMADDRH functions as PMDOUT1H, one of the Output Data Buffer registers.

### 12.1.4 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 12-1.

**TABLE 12-1: PMP PIN MULTIPLEXING FOR 80-PIN DEVICES**

| PMP Function | Pin Assignment | |
|---|---|---|
| | PMPMX = 1 | PMPMX = 0 |
| PMD0 | PORTD<0> | PORTF<7> |
| PMD1 | PORTD<1> | PORTF<6> |
| PMD2 | PORTD<2> | PORTF<5> |
| PMD3 | PORTD<3> | PORTH<4> |
| PMD4 | PORTD<4> | PORTA<5> |
| PMD5 | PORTD<5> | PORTA<4> |
| PMD6 | PORTD<6> | PORTH<3> |
| PMD7 | PORTD<7> | PORTH<2> |
| PMBE | PORTE<2> | PORTH<5> |
| PMWR | PORTE<1> | PORTH<7> |
| PMRD | PORTE<0> | PORTH<6> |

## 12.2 Slave Port Modes

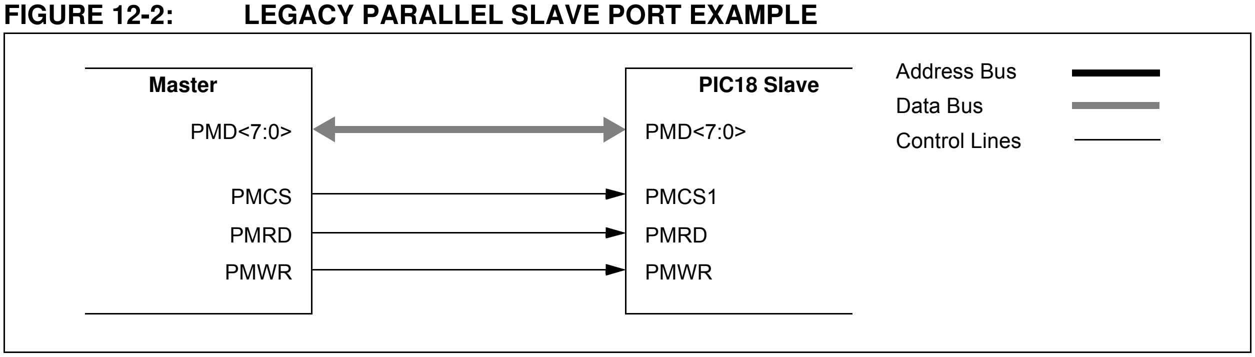
The primary mode of operation for the module is configured using the MODE<1:0> bits in the PMMODEH register. The setting affects whether the module acts as a slave or a master and it determines the usage of the control pins.

### 12.2.1 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 12-2 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.

**FIGURE 12-2: LEGACY PARALLEL SLAVE PORT EXAMPLE**



#### 12.2.1.1 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 12-3. The polarity of the control signals are configurable.

#### 12.2.1.2 READ FROM SLAVE PORT

When chip select is active and a read strobe occurs (PMCS = 1 and PMRD = 1), the data from the PMDOUTL1 register (PMDOUTL1<7:0>) is presented onto PMD<7:0>.The timing for the control signals in Read mode is shown in Figure 12-4.

**FIGURE 12-3: PARALLEL SLAVE PORT WRITE WAVEFORMS**

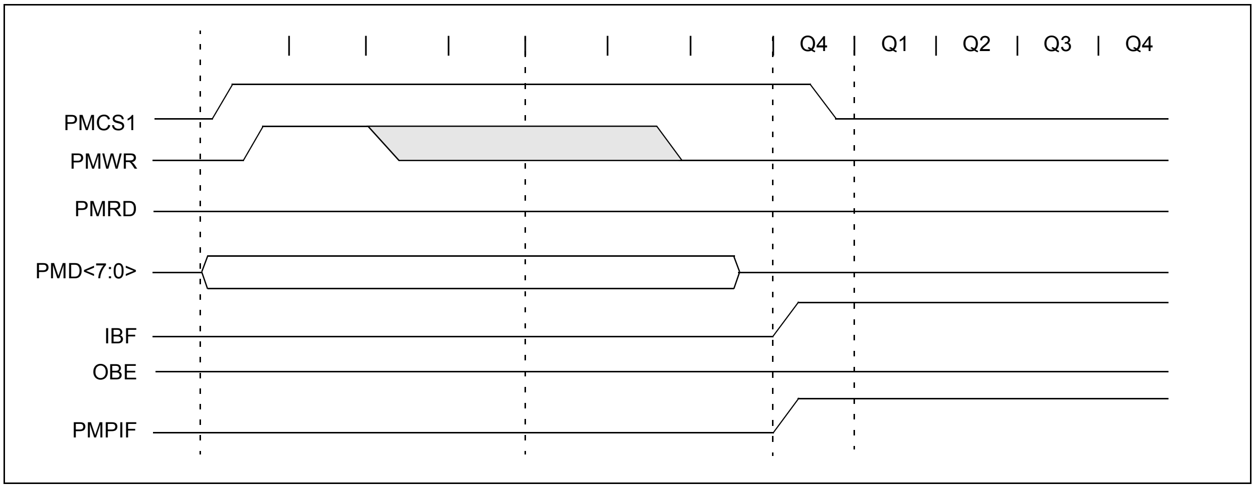
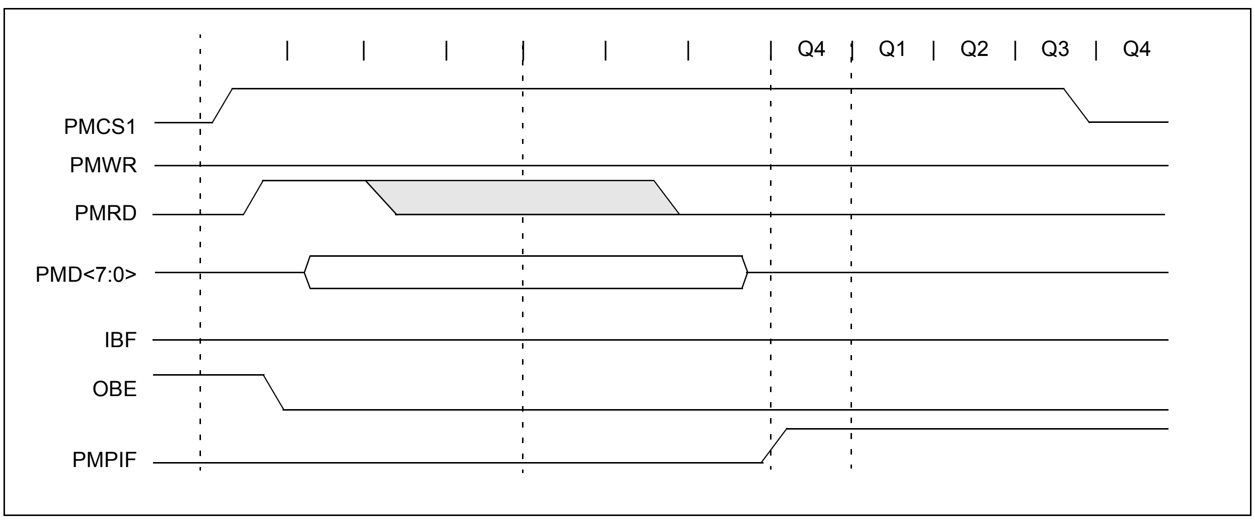



**FIGURE 12-4: PARALLEL SLAVE PORT READ WAVEFORMS**



### 12.2.2 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 INCMx 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.

#### 12.2.2.1 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.

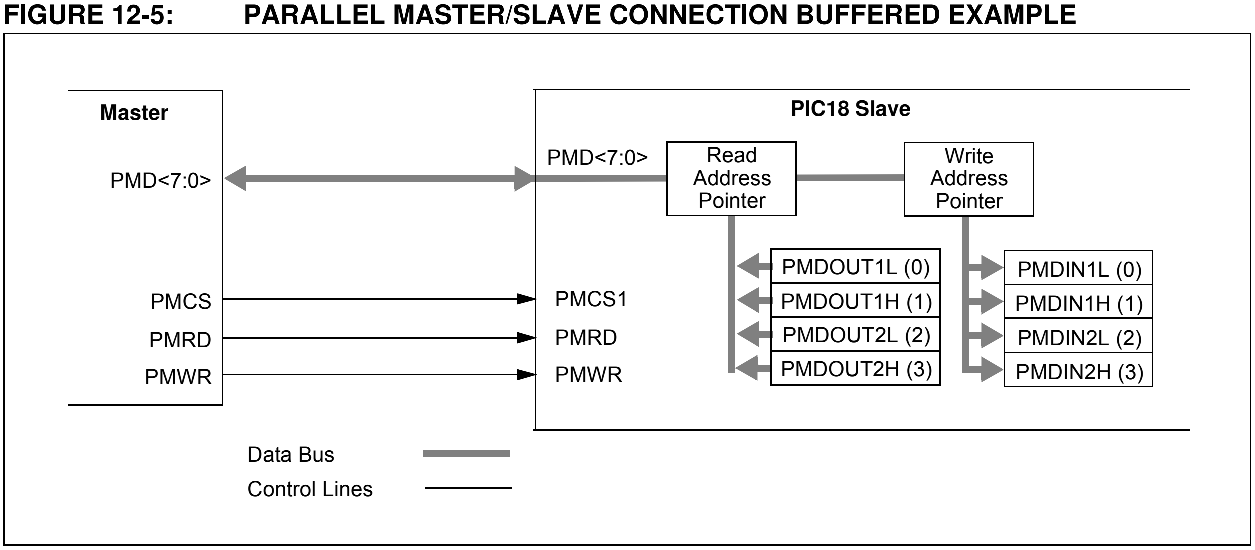
#### 12.2.2.2 WRITE TO SLAVE PORT

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 (IRQM<1:0> = 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 (IRQM<1:0> = 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 their is a risk of hitting an overflow condition.

**FIGURE 12-5: PARALLEL MASTER/SLAVE CONNECTION BUFFERED EXAMPLE**



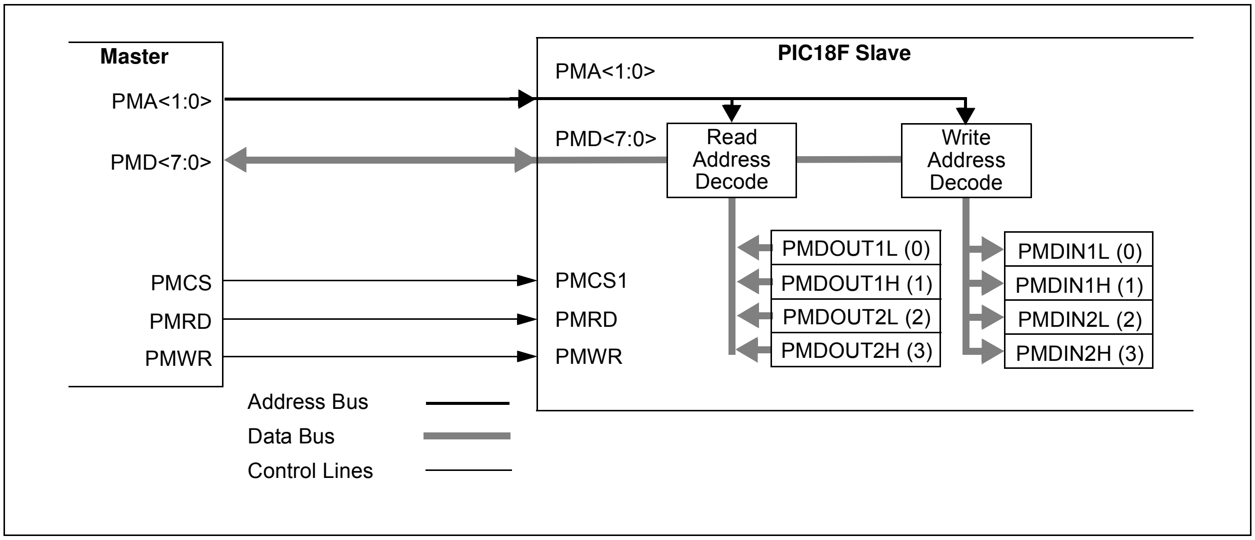
### 12.2.3 ADDRESSABLE PARALLEL SLAVE PORT MODE

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 12-2 shows the buffer addressing for the incoming address to the input and output registers.

**TABLE 12-2: SLAVE MODE BUFFER ADDRESSING**

| PMADDR <1:0> | Output Register (Buffer) | Input Register (Buffer) |
|---|---|---|
| 00 | PMDOUT1L (0) | PMDIN1L (0) |
| 01 | PMDOUT1H (1) | PMDIN1H (1) |
| 10 | PMDOUT2L (2) | PMDIN2L (2) |
| 11 | PMDOUT2H (3) | PMDIN2H (3) |

**FIGURE 12-6: PARALLEL MASTER/SLAVE CONNECTION ADDRESSED BUFFER EXAMPLE**



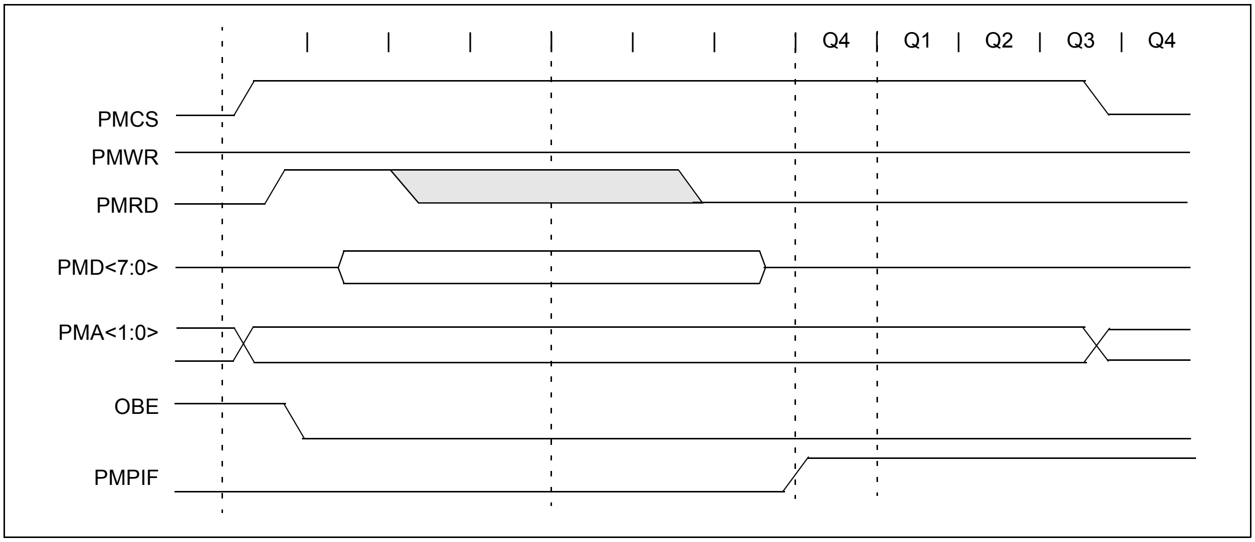

 

#### 12.2.3.1 READ FROM SLAVE PORT

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 12-2 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 12-7: PARALLEL SLAVE PORT READ WAVEFORMS**

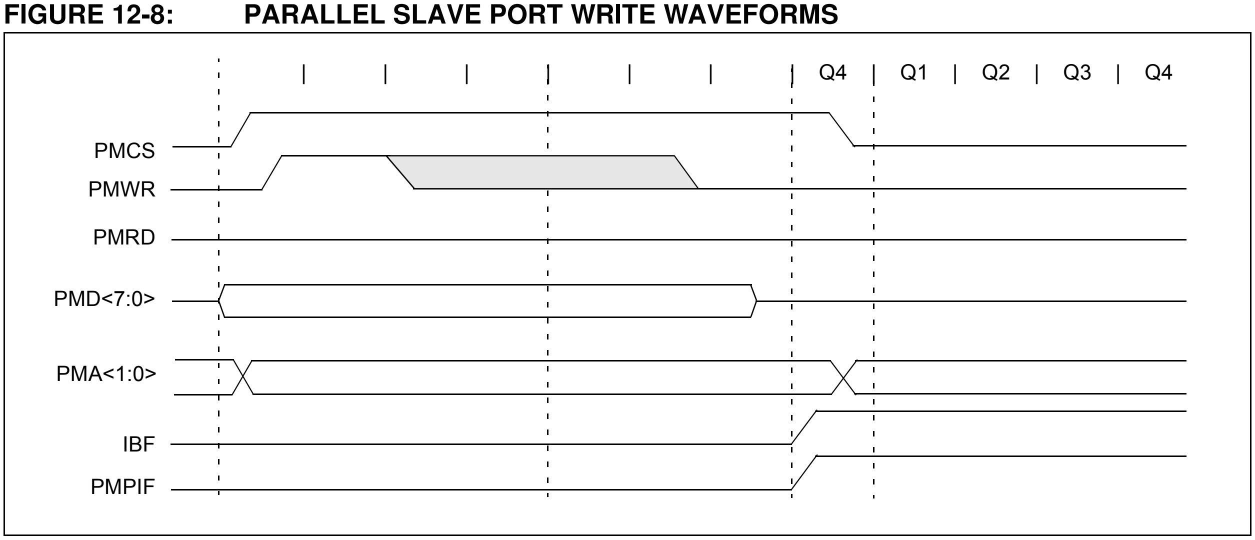



#### 12.2.3.2 WRITE TO SLAVE PORT

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 12-2 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 12-8: PARALLEL SLAVE PORT WRITE WAVEFORMS**



## 12.3 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

### 12.3.1 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, PTRDEN 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 PTENx 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.

### 12.3.2 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.

### 12.3.3 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 Port Enable (PMBE) bit control strobe is used to signal when the Most Significant Byte of data is being presented on the data lines.

### 12.3.4 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 ADRMUX<1:0> bits (PMCONH<4:3>). There are three address multiplexing modes available; typical pinout configurations for these modes are shown in Figure 12-9, Figure 12-10 and Figure 12-11.

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 PMADDRL<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 12-9: DEMULTIPLEXED ADDRESSING MODE (SEPARATE READ AND WRITE STROBES, TWO CHIP SELECTS)**

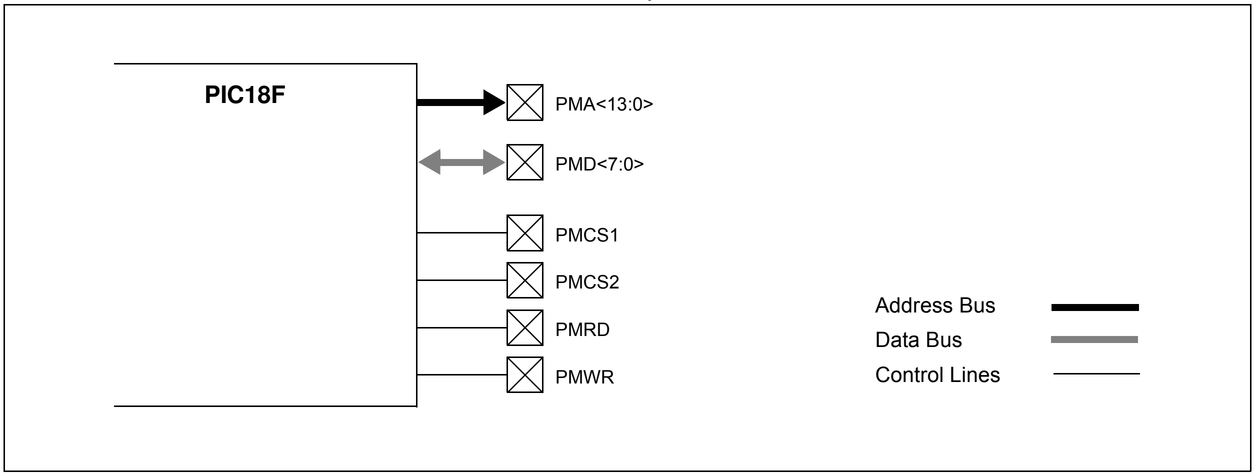



**FIGURE 12-10: PARTIALLY MULTIPLEXED ADDRESSING MODE (SEPARATE READ AND WRITE STROBES, TWO CHIP SELECTS)**

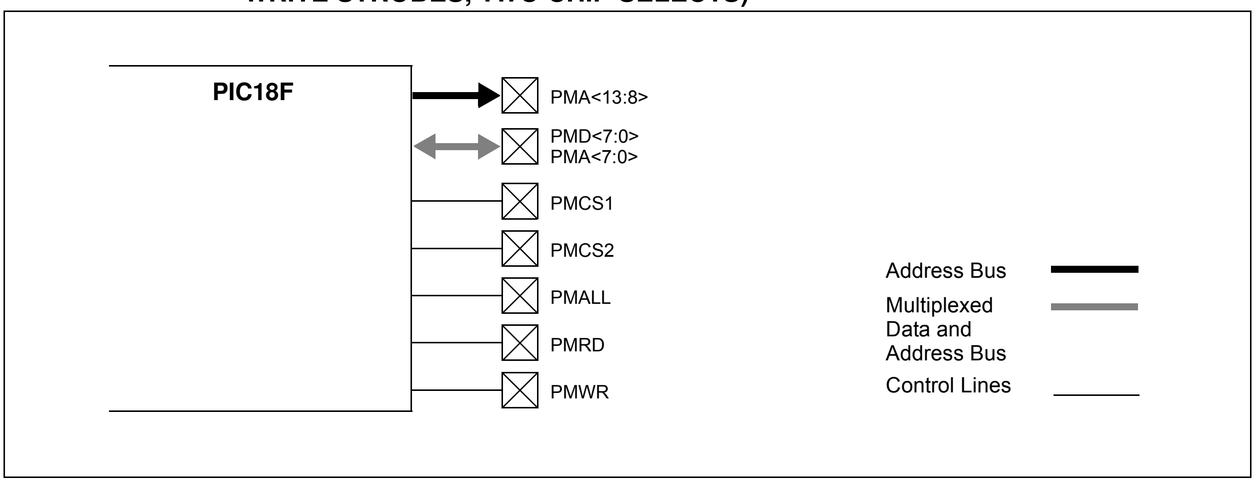
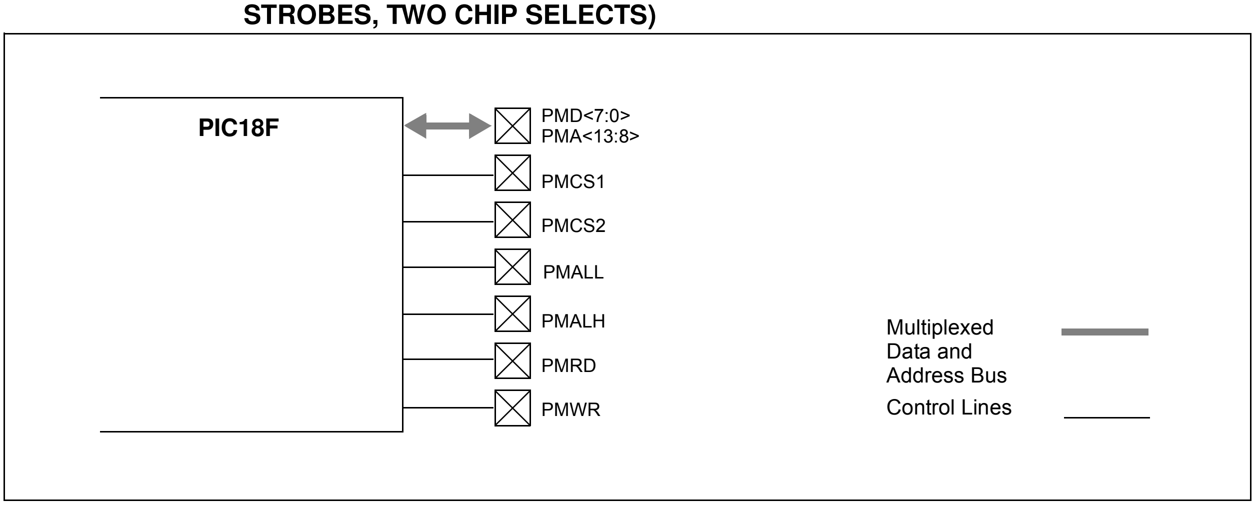



**FIGURE 12-11: FULLY MULTIPLEXED ADDRESSING MODE (SEPARATE READ AND WRITE STROBES, TWO CHIP SELECTS)**



### 12.3.5 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>).

### 12.3.6 AUTO-INCREMENT/DECREMENT

While the module is operating in one of the Master modes, the INCMx 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.

### 12.3.7 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 WAITB<1:0> 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 WAITM<3:0> 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 WAITBx and WAITEx have no effect. The WAITE<1:0> 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.

### 12.3.8 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.

### 12.3.9 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.

### 12.3.10 PARALLEL MASTER PORT STATUS

#### 12.3.10.1 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).

#### 12.3.10.2 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.

### 12.3.11 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 12-12: READ AND WRITE TIMING, 8-BIT DATA, DEMULTIPLEXED ADDRESS**

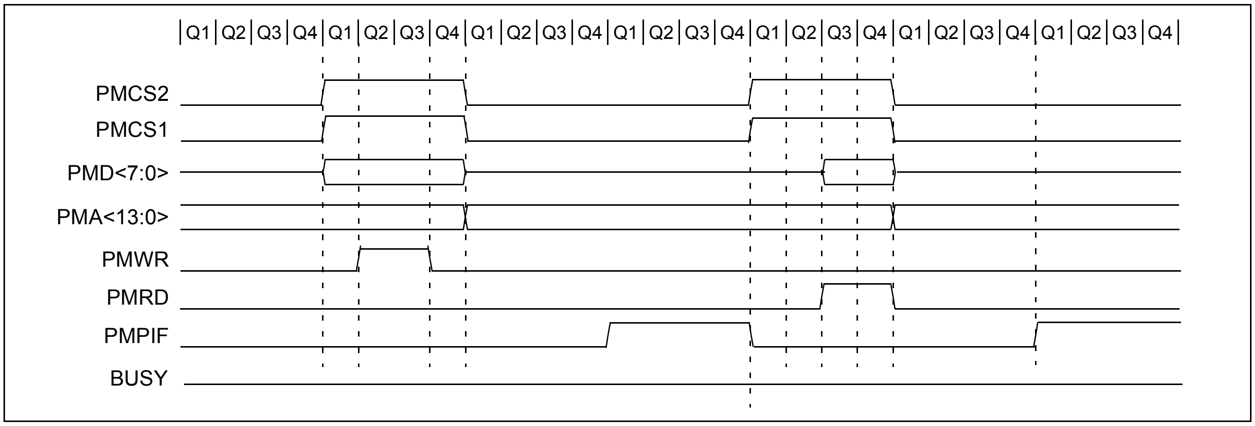



**FIGURE 12-13: READ TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS**

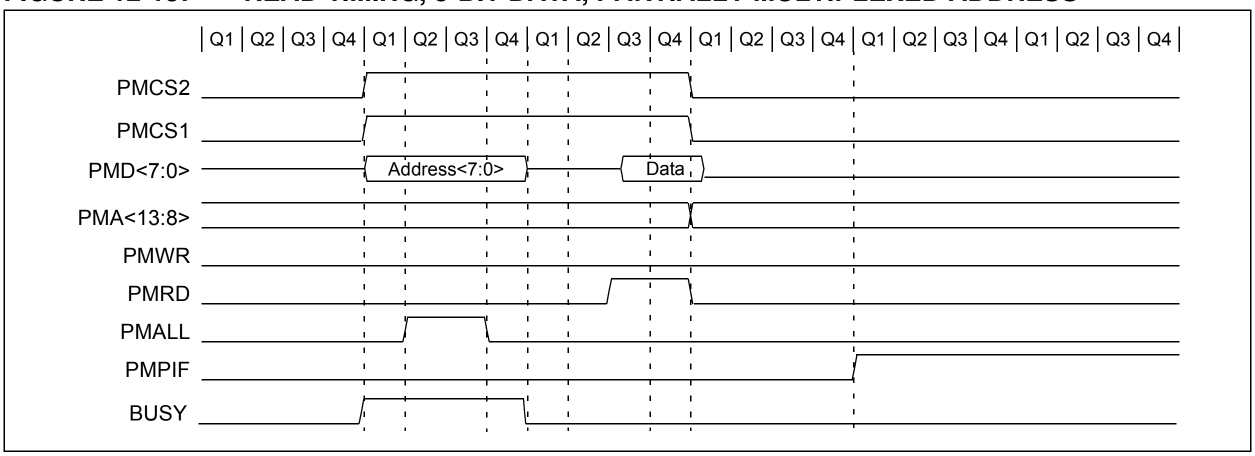


**FIGURE 12-14: READ TIMING, 8-BIT DATA, WAIT STATES ENABLED, PARTIALLY MULTIPLEXED ADDRESS**

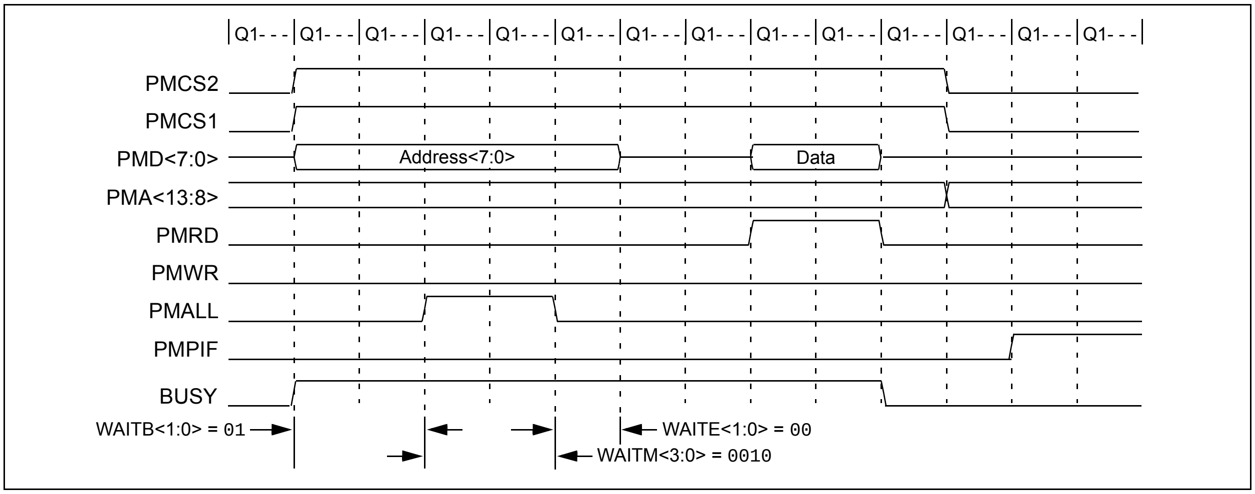



**FIGURE 12-15: WRITE TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS**

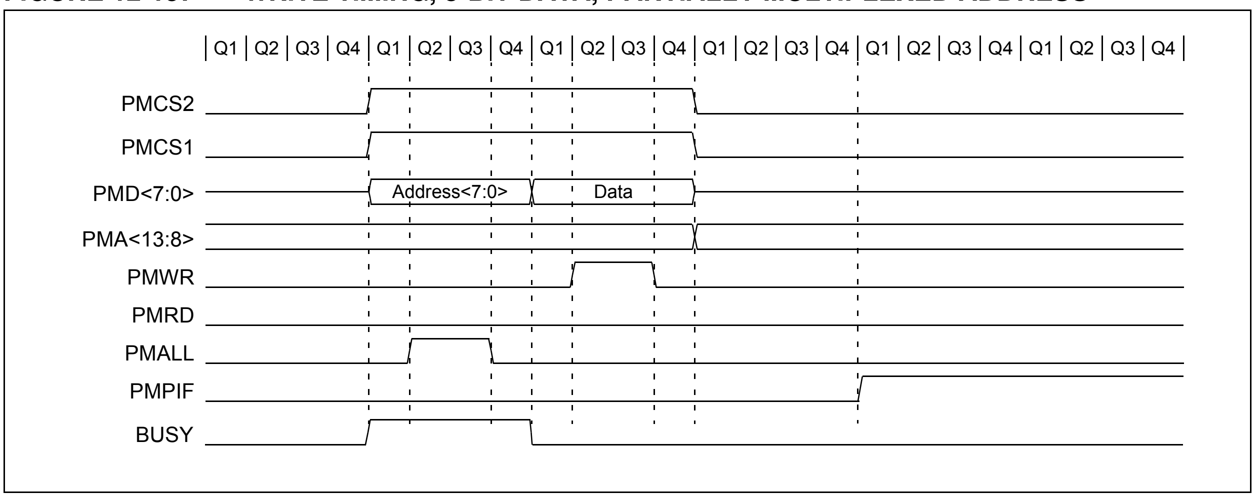



**FIGURE 12-16: WRITE TIMING, 8-BIT DATA, WAIT STATES ENABLED, PARTIALLY MULTIPLEXED ADDRESS**

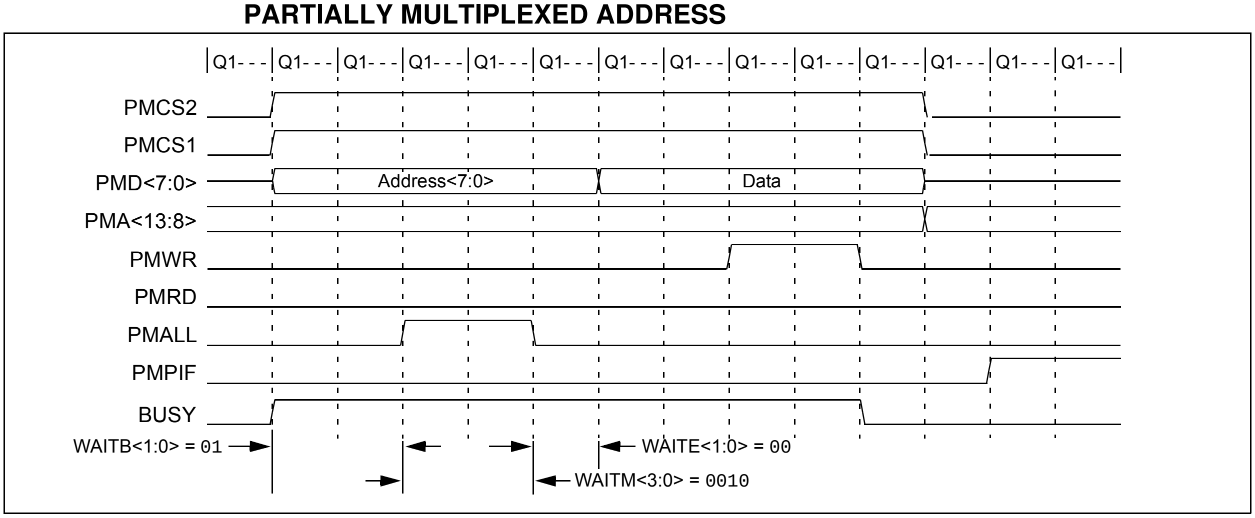


**FIGURE 12-17: READ TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS, ENABLE STROBE**

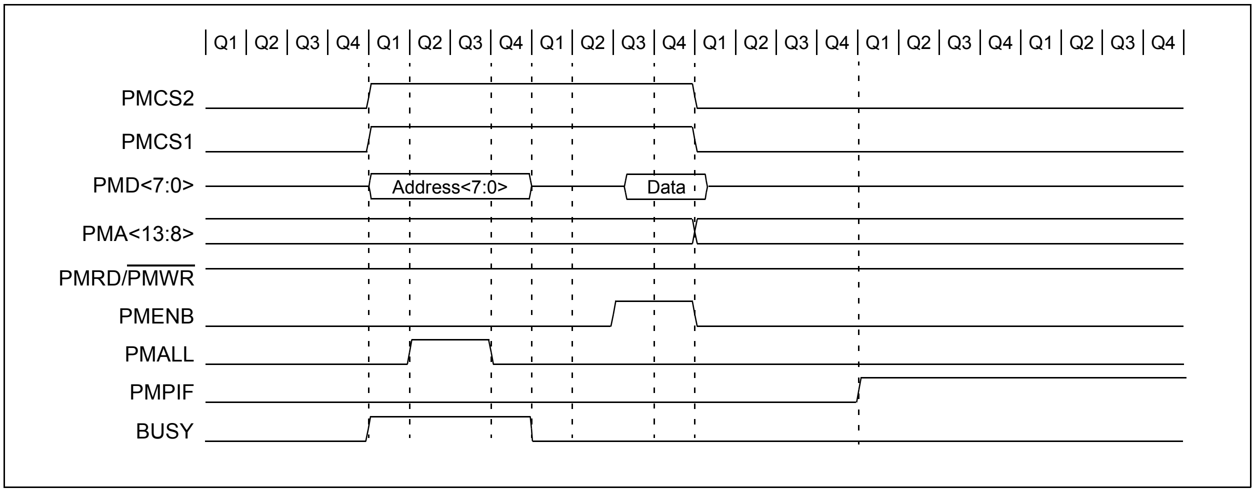



**FIGURE 12-18: WRITE TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS, ENABLE STROBE**

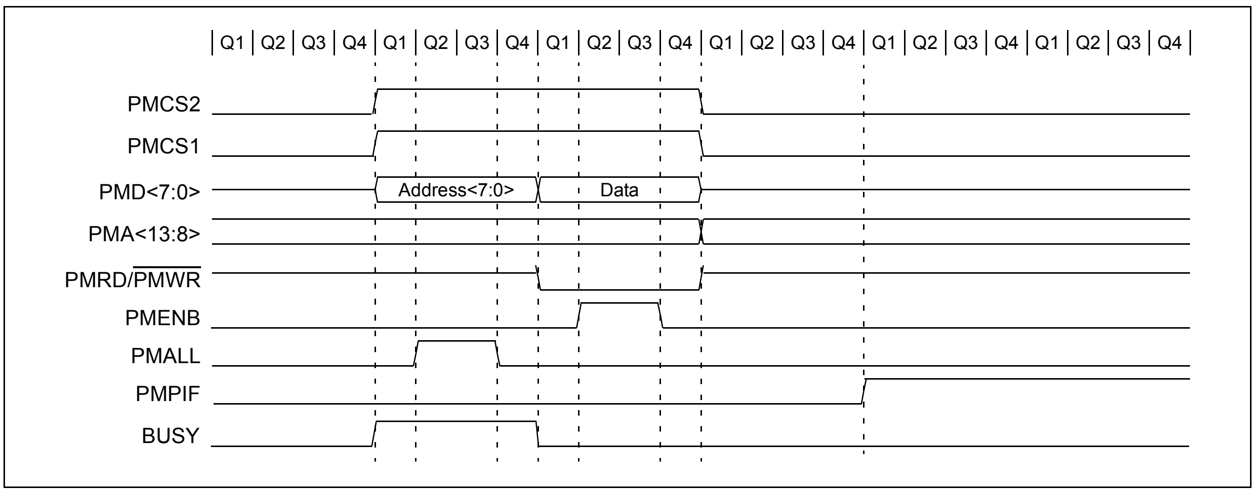



**FIGURE 12-19: READ TIMING, 8-BIT DATA, FULLY MULTIPLEXED 16-BIT ADDRESS**

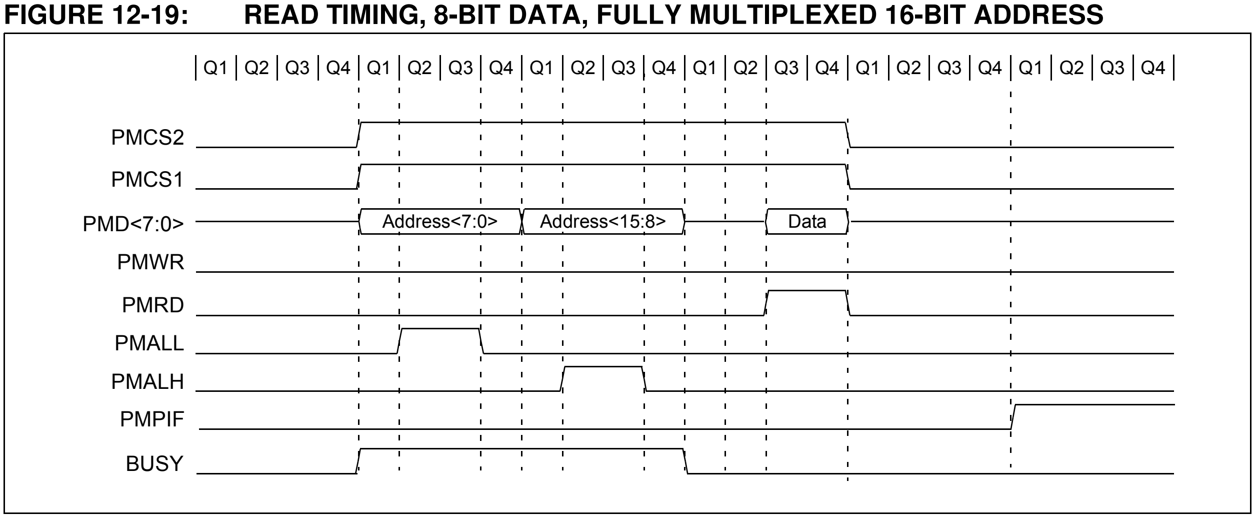


**FIGURE 12-20: WRITE TIMING, 8-BIT DATA, FULLY MULTIPLEXED 16-BIT ADDRESS**

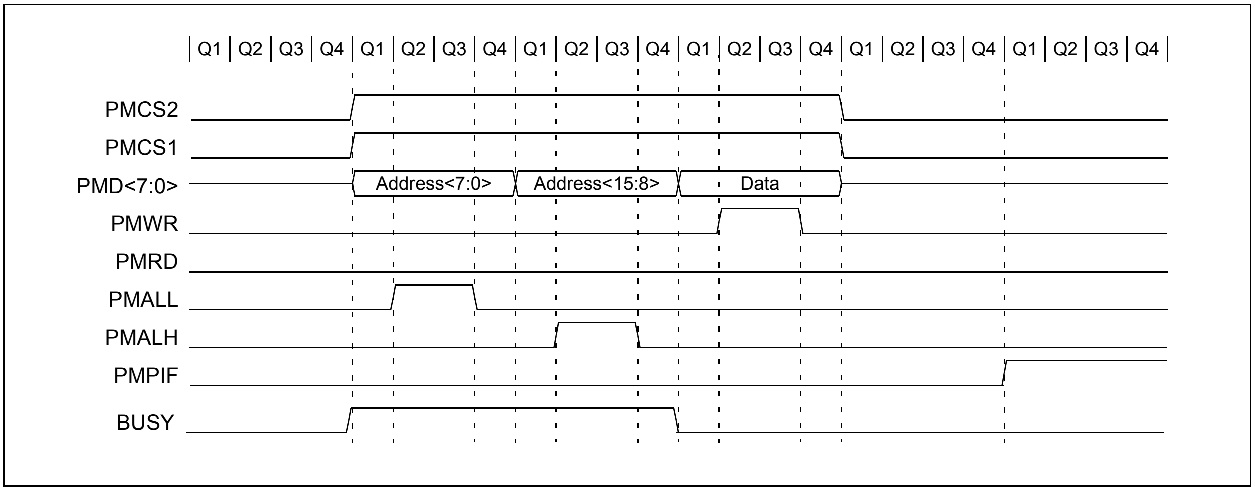



**FIGURE 12-21: READ TIMING, 16-BIT DATA, DEMULTIPLEXED ADDRESS**

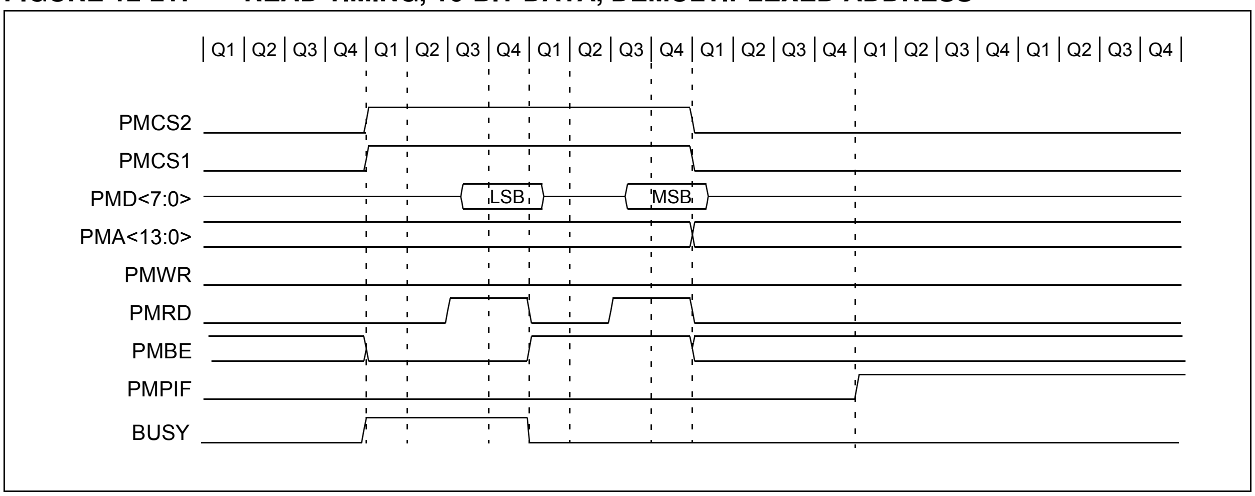



**FIGURE 12-22: WRITE TIMING, 16-BIT DATA, DEMULTIPLEXED ADDRESS**

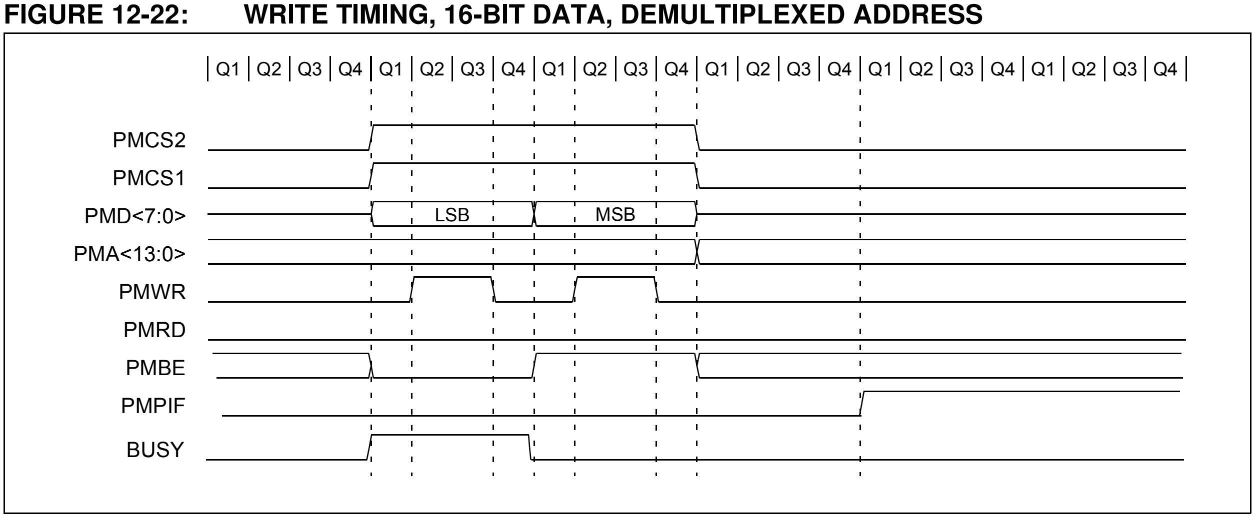


**FIGURE 12-23: READ TIMING, 16-BIT MULTIPLEXED DATA, PARTIALLY MULTIPLEXED ADDRESS**

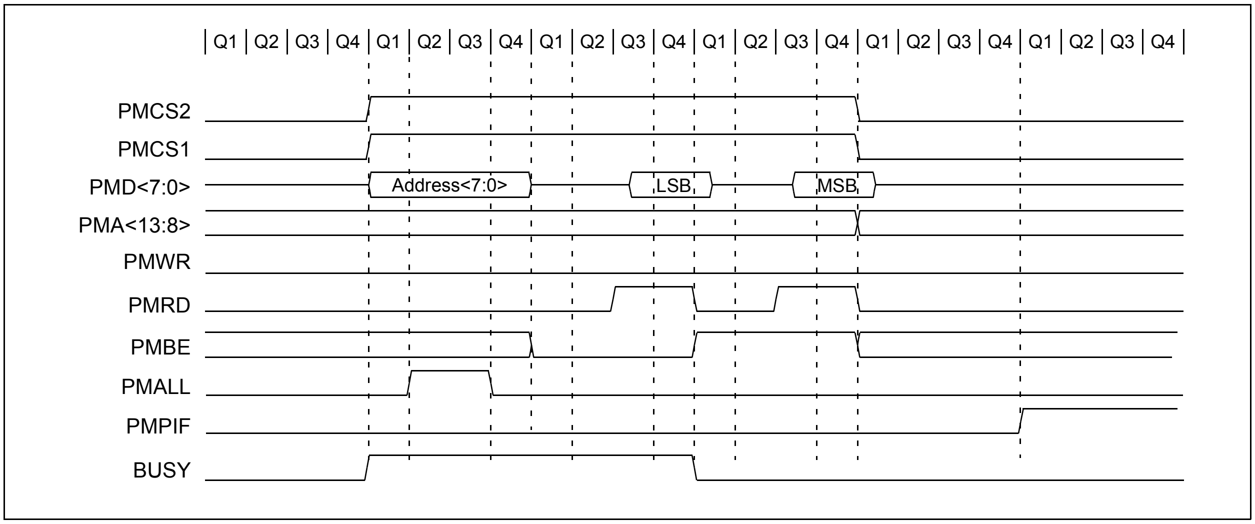



**FIGURE 12-24: WRITE TIMING, 16-BIT MULTIPLEXED DATA, PARTIALLY MULTIPLEXED ADDRESS**

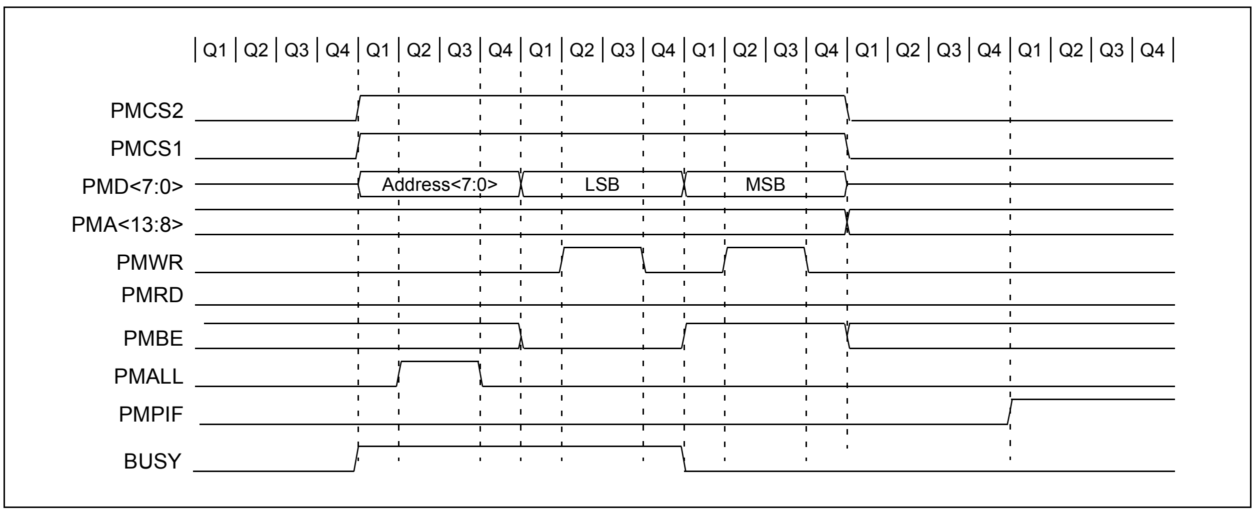


**FIGURE 12-25: READ TIMING, 16-BIT MULTIPLEXED DATA, FULLY MULTIPLEXED 16-BIT ADDRESS**

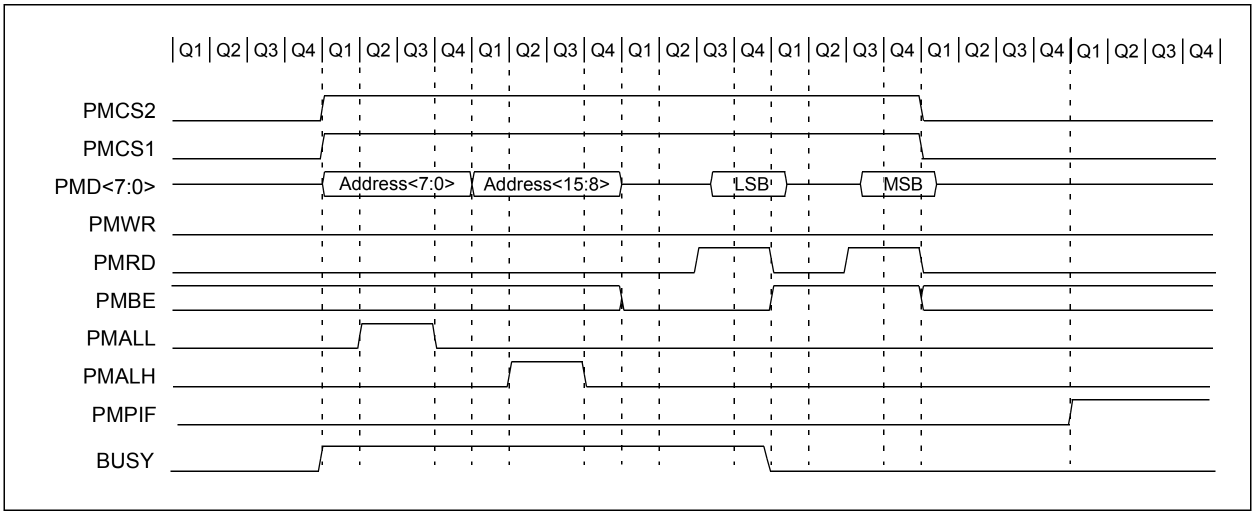
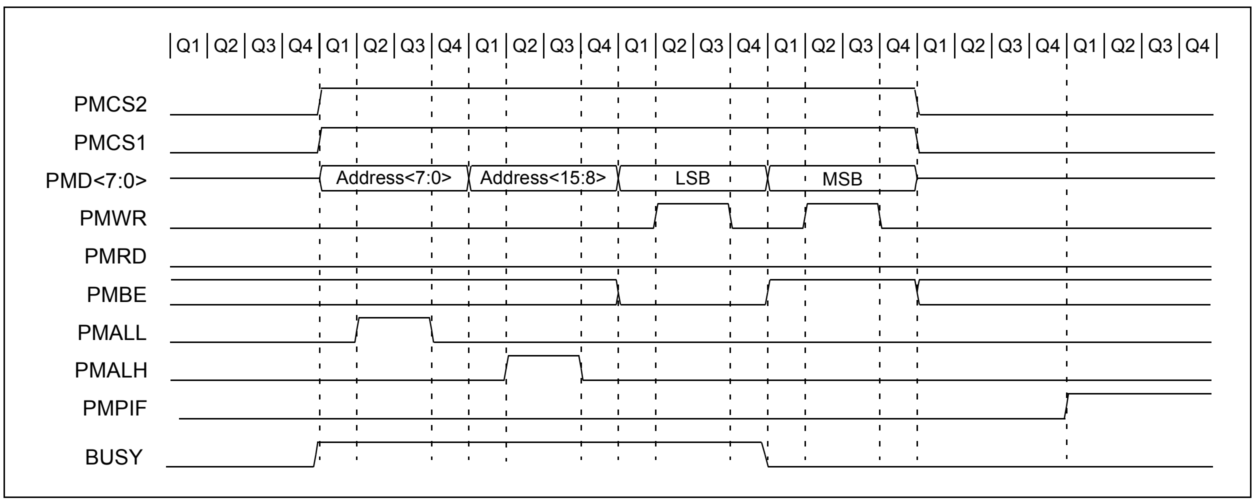



**FIGURE 12-26: WRITE TIMING, 16-BIT MULTIPLEXED DATA, FULLY MULTIPLEXED 16-BIT ADDRESS**



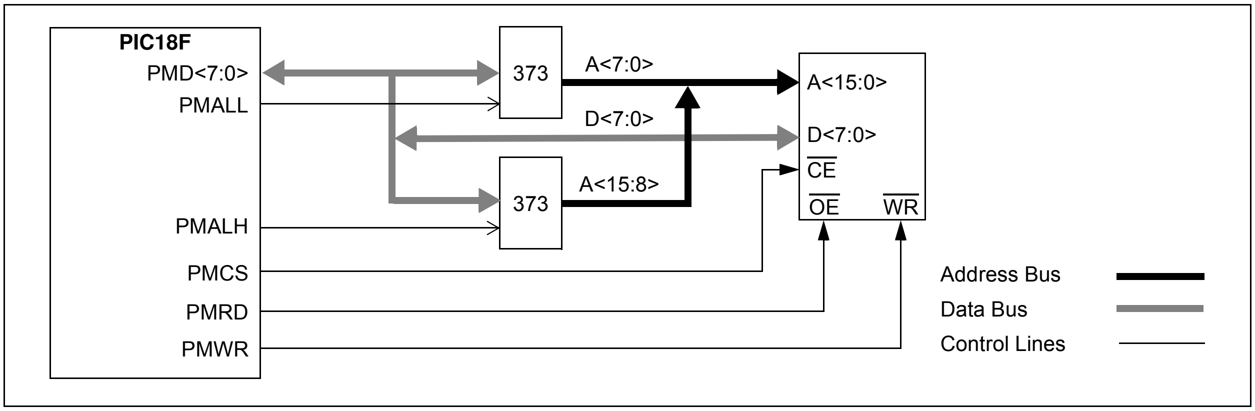
## 12.4 Application Examples

This section introduces some potential applications for the PMP module.

### 12.4.1 MULTIPLEXED MEMORY OR PERIPHERAL

Figure 12-27 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 12-27: EXAMPLE OF A MULTIPLEXED ADDRESSING APPLICATION**



### 12.4.2 PARTIALLY MULTIPLEXED MEMORY OR PERIPHERAL

Partial multiplexing implies using more pins; however, for a few extra pins, some extra performance can be achieved. Figure 12-28 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 12-29, then no extra circuitry is required except for the peripheral itself.

**FIGURE 12-28: EXAMPLE OF A PARTIALLY MULTIPLEXED ADDRESSING APPLICATION**

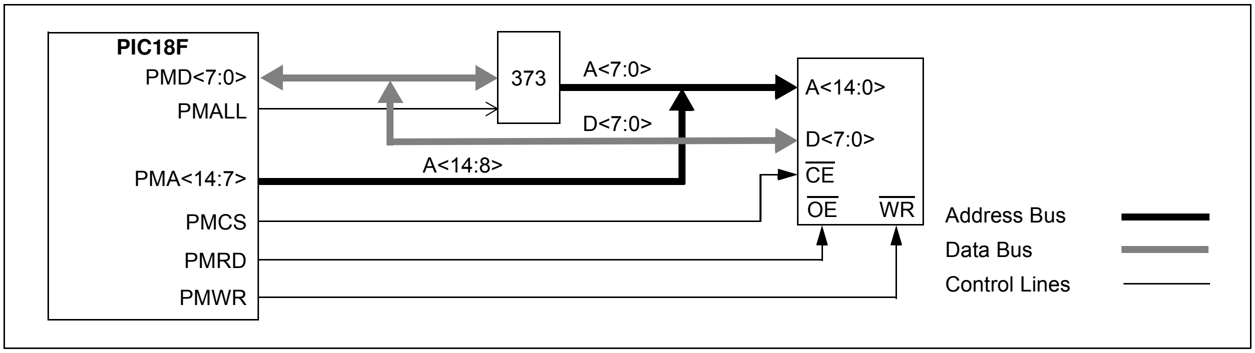
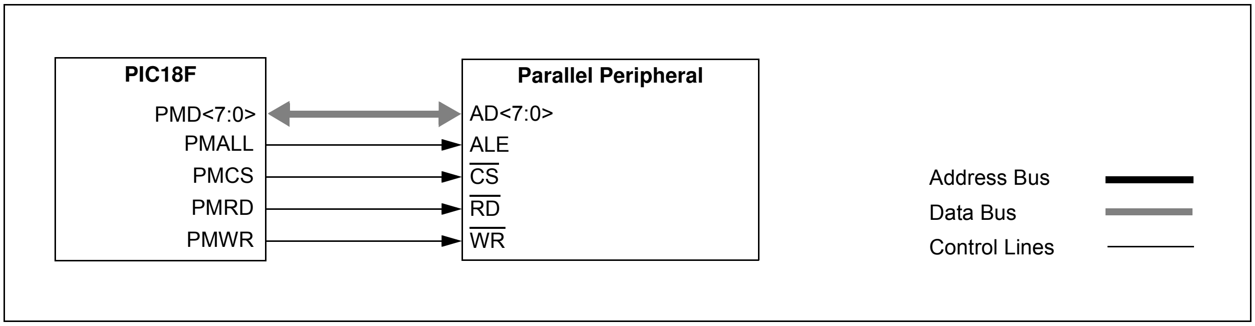



**FIGURE 12-29: EXAMPLE OF AN 8-BIT MULTIPLEXED ADDRESS AND DATA APPLICATION**



### 12.4.3 PARALLEL EEPROM EXAMPLE

Figure 12-30 shows an example connecting parallel EEPROM to the PMP. Figure 12-31 shows a slight variation to this, configuring the connection for 16-bit data from a single EEPROM.

**FIGURE 12-30: PARALLEL EEPROM EXAMPLE (UP TO 15-BIT ADDRESS, 8-BIT DATA)**

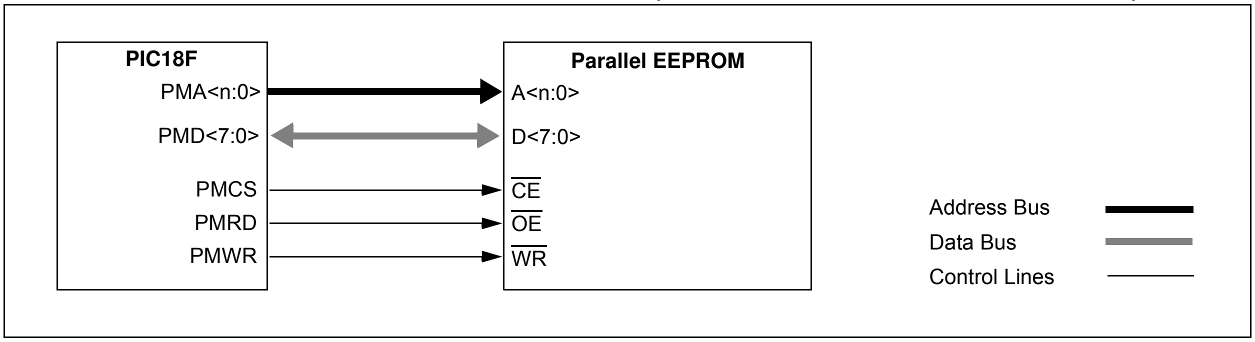
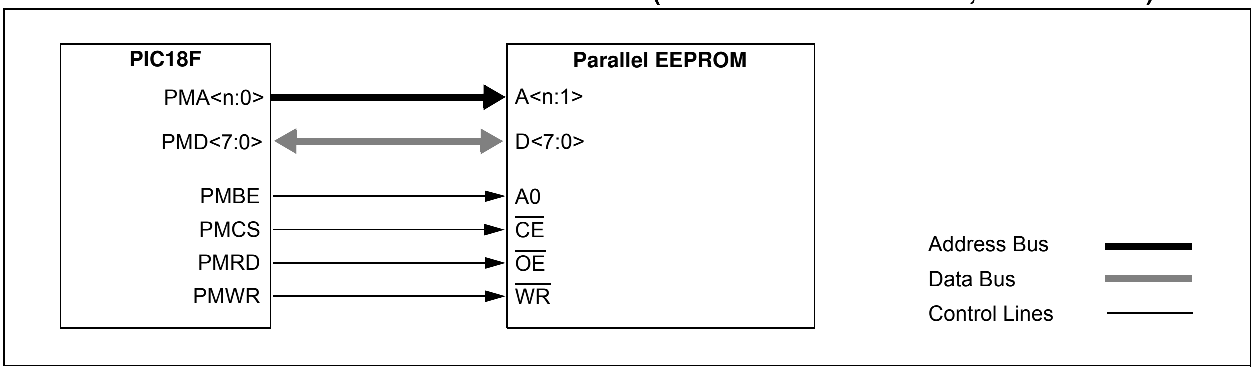



**FIGURE 12-31: PARALLEL EEPROM EXAMPLE (UP TO 15-BIT ADDRESS, 16-BIT DATA)**



### 12.4.4 LCD CONTROLLER EXAMPLE

The PMP module can be configured to connect to a typical LCD controller interface, as shown in Figure 12-32. In this case, the PMP module is configured for active-high control signals since common LCD displays require active-high control.

**FIGURE 12-32: LCD CONTROL EXAMPLE (BYTE MODE OPERATION)**

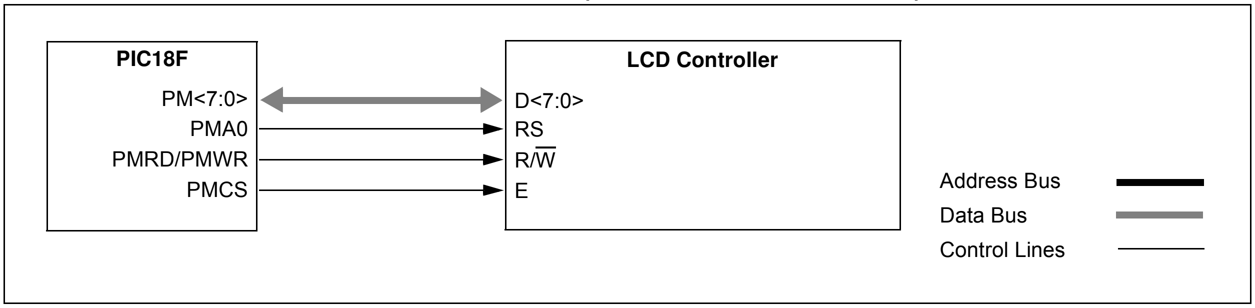



**TABLE 12-3: REGISTERS ASSOCIATED WITH PMP MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PMCONH | PMPEN | — | PSIDL | ADRMUX1 | ADRMUX0 | PTBEEN | PTWREN | PTRDEN | 66 |
| PMCONL | CSF1 | CSF0 | ALP | CS2P | CS1P | BEP | WRSP | RDSP | 66 |
| PMADDRH/ | CS2 | CS1 | Parallel Master Port Address High Byte | | | | | | 66 |
| PMDOUT1H[1] | Parallel Port Out Data High Byte (Buffer 1) | | | | | | | | 66 |
| PMADDRL/ | Parallel Master Port Address Low Byte | | | | | | | | 66 |
| PMDOUT1L[1] | Parallel Port Out Data Low Byte (Buffer 0) | | | | | | | | 66 |
| PMDOUT2H | Parallel Port Out Data High Byte (Buffer 3) | | | | | | | | 66 |
| PMDOUT2L | Parallel Port Out Data Low Byte (Buffer 2) | | | | | | | | 66 |
| PMDIN1H | Parallel Port In Data High Byte (Buffer 1) | | | | | | | | 66 |
| PMDIN1L | Parallel Port In Data Low Byte (Buffer 0) | | | | | | | | 66 |
| PMDIN2H | Parallel Port In Data High Byte (Buffer 3) | | | | | | | | 66 |
| PMDIN2L | Parallel Port In Data Low Byte (Buffer 2) | | | | | | | | 66 |
| PMMODEH | BUSY | IRQM1 | IRQM0 | INCM1 | INCM0 | MODE16 | MODE1 | MODE0 | 66 |
| PMMODEL | WAITB1 | WAITB0 | WAITM3 | WAITM2 | WAITM1 | WAITM0 | WAITE1 | WAITE0 | 66 |
| PMEH | PTEN15 | PTEN14 | PTEN13 | PTEN12 | PTEN11 | PTEN10 | PTEN9 | PTEN8 | 66 |
| PMEL | PTEN7 | PTEN6 | PTEN5 | PTEN4 | PTEN3 | PTEN2 | PTEN1 | PTEN0 | 66 |
| PMSTATH | IBF | IBOV | — | — | IB3F | IB2F | IB1F | IB0F | 66 |
| PMSTATL | OBE | OBUF | — | — | OB3E | OB2E | OB1E | OB0E | 66 |
| PADCFG1[2] | — | — | — | — | — | — | — | PMPTTL | 62 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during PMP operation.

**Note 1:** 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.

**2:** Configuration SFR overlaps with the default SFR at this address; available only when WDTCON<4> = 1.

## 13.0 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

The T0CON register (Register 13-1) controls all aspects of the module's operation, including the prescale selection; it is both readable and writable.

A simplified block diagram of the Timer0 module in 8-bit mode is shown in Figure 13-1. Figure 13-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

### REGISTER 13-1: T0CON: TIMER0 CONTROL REGISTER

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 |
| 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 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-Bit/16-Bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin input edge
0 = Internal clock (FOSC/4)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increments on high-to-low transition on T0CKI pin
0 = Increments on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is not assigned; Timer0 clock input bypasses prescaler
0 = Timer0 prescaler is assigned; Timer0 clock input comes from prescaler output

bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

## 13.1 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 13.3 "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.

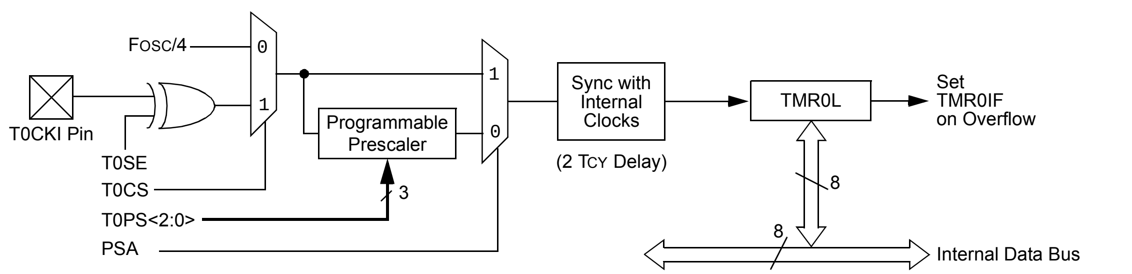
An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (TOSC). There is a delay between synchronization and the onset of incrementing the timer/counter.

## 13.2 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 13-2). 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.

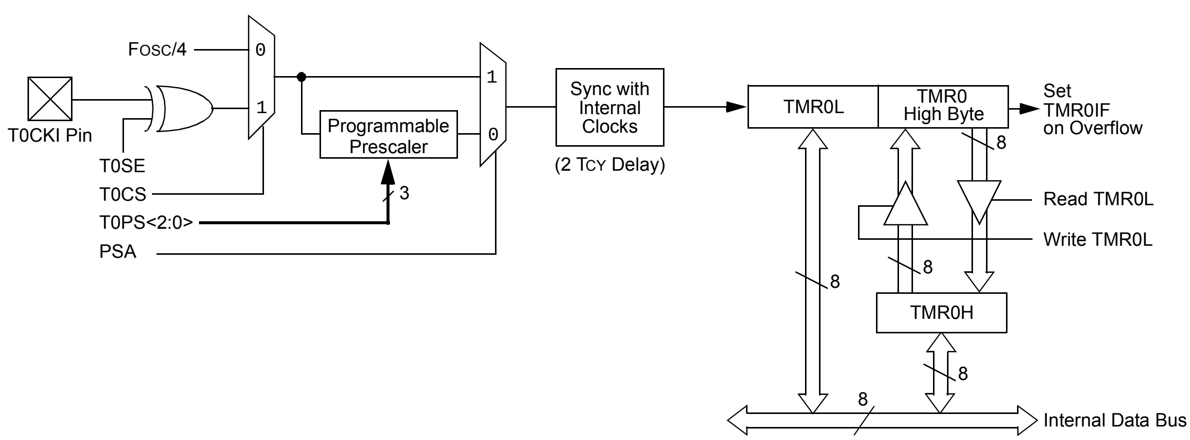
Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

**FIGURE 13-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)**




**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

**FIGURE 13-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)**




**Note:** Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

## 13.3 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 T0PS<2:0> 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.

### 13.3.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control and can be changed "on-the-fly" during program execution.

## 13.4 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 TMR0IE bit (INTCON<5>). Before re-enabling the interrupt, the TMR0IF 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 13-1: REGISTERS ASSOCIATED WITH TIMER0**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| TMR0L | Timer0 Register Low Byte | | | | | | | | 62 |
| TMR0H | Timer0 Register High Byte | | | | | | | | 62 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 | 62 |
| TRISA | TRISA7[(1)] | TRISA6[(1)] | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 64 |

**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.

**NOTES:**

## 14.0 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 14-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 14-2.

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 14-1). 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 14-1: T1CON: TIMER1 CONTROL REGISTER[(1)]

| R/W-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON |
| 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 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of TImer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **T1RUN:** Timer1 System Clock Status bit
1 = Device clock is derived from Timer1 oscillator
0 = Device clock is derived from another source

bit 5-4 **T1CKPS<1:0>:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Does not synchronize external clock input
0 = Synchronizes external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from the RC0/T1OSO/T13CKI pin (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

**Note 1:** Default (legacy) SFR at this address, available when WDTCON<4> = 0.

## 14.1 Timer1 Operation

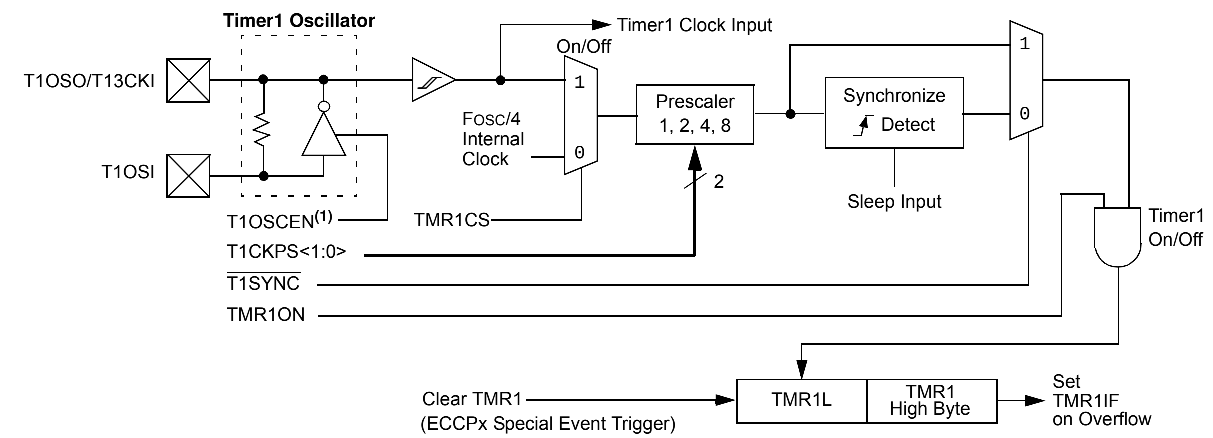
Timer1 can operate in one of these modes:

- 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.

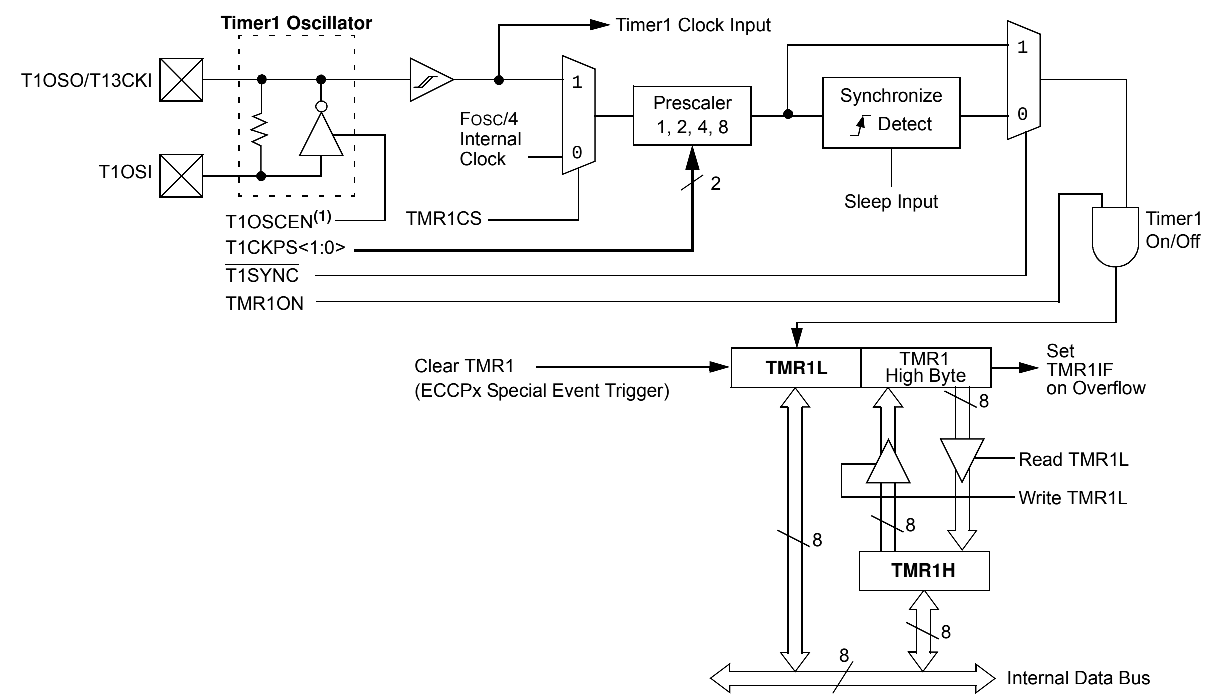
When Timer1 is enabled, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as ‘`0`’.

**FIGURE 14-1: TIMER1 BLOCK DIAGRAM**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

**FIGURE 14-2: TIMER1 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.

## 14.2 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 14-2). 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.

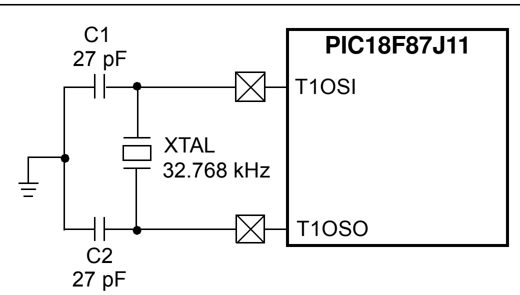
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

## 14.3 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 14-3. Table 14-1 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 14-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR**




**Note:** See the notes with Table 14-1 for additional information about capacitor selection.

**TABLE 14-1: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR(2,3,4)**

| Oscillator Type | Freq. | C1 | C2 |
|---|---|---|---|
| LP | 32 kHz | 27 pF(1) | 27 pF(1) |

**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.

### 14.3.1 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, SCS<1:0> (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 4.0 "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 Fail-Safe 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.

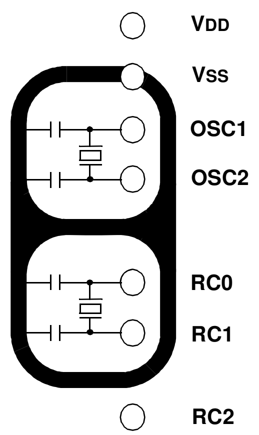
### 14.3.2 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

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 14-3, 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 14-4, may be helpful when used on a single-sided PCB or in addition to a ground plane.

**FIGURE 14-4: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING**




**Note:** Not drawn to scale.

## 14.4 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>).

## 14.5 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 (CCPxM<3:0> = `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 19.2.1 "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>). |
|---|---|

## 14.6 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 14.3 "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 14-1, 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.

## 14.7 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 14-1 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 14-1: 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
            MOVWF     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 14-2: REGISTERS ASSOCIATED WITH TIMER1 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: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| TMR1L[(1)] | Timer1 Register Low Byte | | | | | | | | 62 |
| TMR1H[(1)] | Timer1 Register High Byte | | | | | | | | 62 |
| T1CON[(1)] | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 62 |

**Legend:** Shaded cells are not used by the Timer1 module.

**Note 1:** Default (legacy) SFR at this address, available when WDTCON<4> = 0.

## 15.0 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 15-1) 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 15-1.

### 15.1 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, T2CKPS<1:0> (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 15.2 "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, $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

**REGISTER 15-1: T2CON: TIMER2 CONTROL REGISTER**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| 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 **Unimplemented:** Read as '0'

bit 6-3 **T2OUTPS<3:0>:** Timer2 Output Postscale Select bits
`0000` = 1:1 Postscale
`0001` = 1:2 Postscale
•
•
•
`1111` = 1:16 Postscale

bit 2 **TMR2ON:** Timer2 On bit
`1` = Timer2 is on
`0` = Timer2 is off

bit 1-0 **T2CKPS<1:0>:** Timer2 Clock Prescale Select bits
`00` = Prescaler is 1
`01` = Prescaler is 4
`1x` = Prescaler is 16

### 15.2 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, T2OUTPS<3:0> (T2CON<6:3>).

### 15.3 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 20.0 "Master Synchronous Serial Port (MSSP) Module"**.

**FIGURE 15-1: TIMER2 BLOCK DIAGRAM**

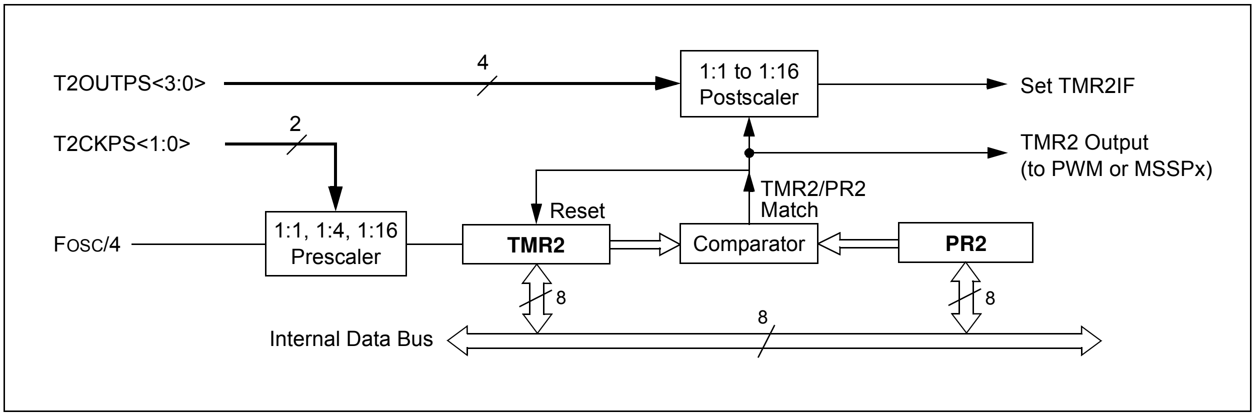



**TABLE 15-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| TMR2[(1)] | Timer2 Register | | | | | | | | 62 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | 62 |
| PR2[(1)] | Timer2 Period Register | | | | | | | | 62 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

**Note 1:** Default (legacy) SFR at this address, available when WDTCON<4> = 0.

## 16.0 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 16-1. A block diagram of the module's operation in Read/Write mode is shown in Figure 16-2.

The Timer3 module is controlled through the T3CON register (Register 16-1). It also selects the clock source options for the CCP and ECCP modules; see **Section 18.1.1 "CCP Modules and Timer Resources"** for more information.

### REGISTER 16-1: T3CON: TIMER3 CONTROL REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| RD16 | T3CCP2 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON |
| 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 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6,3 **T3CCP<2:1>:** Timer3 and Timer1 to ECCPx/CCPx Enable bits
11 = Timer3 and Timer4 are the clock sources for all ECCPx/CCPx modules
10 = Timer3 and Timer4 are the clock sources for ECCP3, CCP4 and CCP5; Timer1 and Timer2 are the clock sources for ECCP1 and ECCP2
01 = Timer3 and Timer4 are the clock sources for ECCP2, ECCP3, CCP4 and CCP5; Timer1 and Timer2 are the clock sources for ECCP1
00 = Timer1 and Timer2 are the clock sources for all ECCPx/CCPx modules

bit 5-4 **T3CKPS<1:0>**: Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)
When TMR3CS = 1:
1 = Does not synchronize external clock input
0 = Synchronizes external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T13CKI (on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

## 16.1 Timer3 Operation

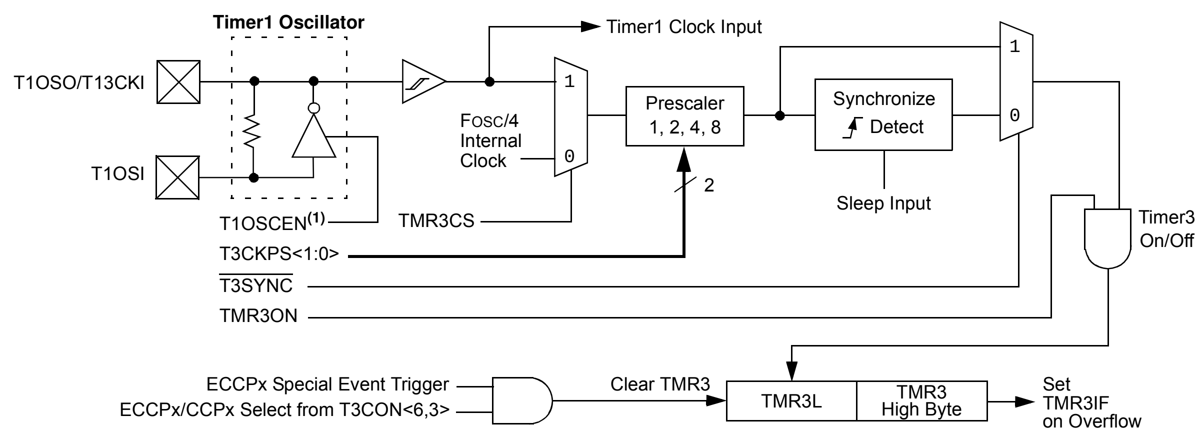
Timer3 can operate in one of three modes:

- 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.

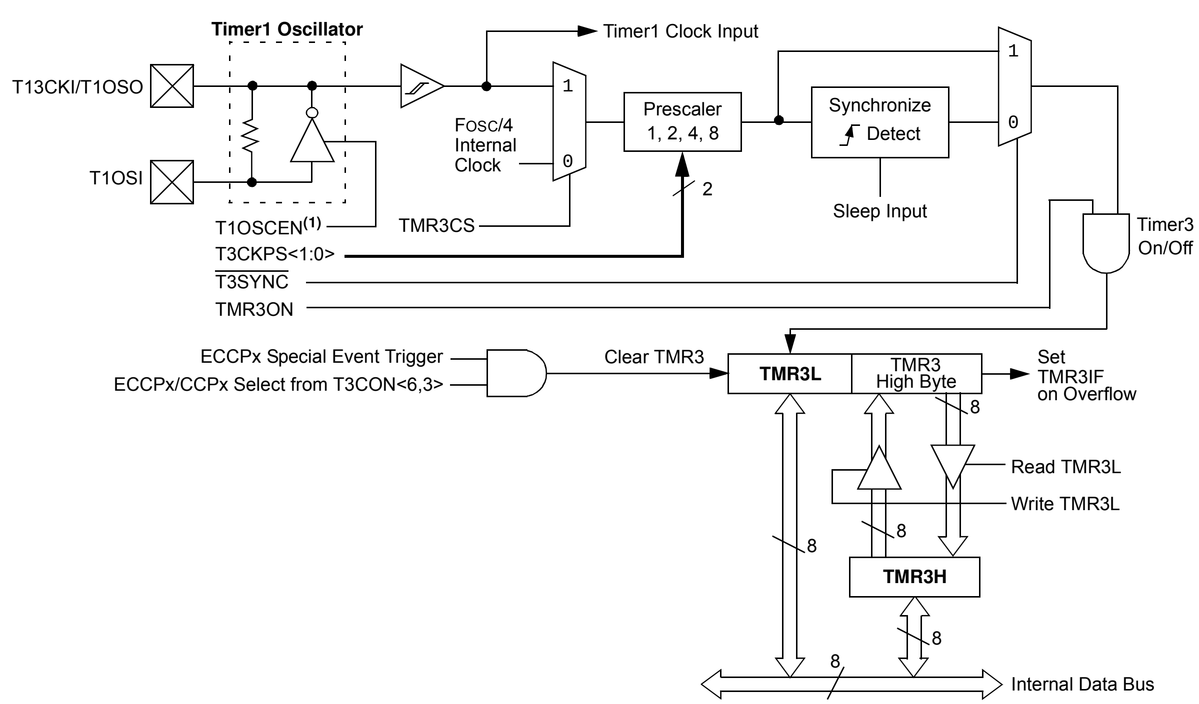
As with Timer1, the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs when the Timer1 oscillator is enabled. This means the values of TRISC<1:0> are ignored and the pins are read as ‘`0`’.

**FIGURE 16-1: TIMER3 BLOCK DIAGRAM**




**Note 1:** When enable bit, T1OSCEN, is cleared, the inverter and feedback resistor are turned off to eliminate power drain.

**FIGURE 16-2: 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.

## 16.2 Timer3 16-Bit Read/Write Mode

Timer3 can be configured for 16-bit reads and writes (see Figure 16-2). 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 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 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.

## 16.3 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 TMR3CS 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 14.0 "Timer1 Module"**.

## 16.4 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>).

## 16.5 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 (CCPxM<3:0> = `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 19.2.1 "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 CCPRxH:CCPRxL 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 16-1: REGISTERS ASSOCIATED WITH TIMER3 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: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR2 | OSCFIF | CM2IF | CM1IF | — | BCL1IF | LVDIF | TMR3IF | CCP2IF | 64 |
| PIE2 | OSCFIE | CM2IE | CM1IE | — | BCL1IE | LVDIE | TMR3IE | CCP2IE | 64 |
| IPR2 | OSCFIP | CM2IP | CM1IP | — | BCL1IP | LVDIP | TMR3IP | CCP2IP | 64 |
| TMR3L | Timer3 Register Low Byte | | | | | | | | 65 |
| TMR3H | Timer3 Register High Byte | | | | | | | | 65 |
| T1CON(1) | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 62 |
| T3CON | RD16 | T3CCP2 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 65 |

**Legend:** — = unimplemented, read as '`0`'. Shaded cells are not used by the Timer3 module.

**Note 1:** Default (legacy) SFR at this address, available when WDTCON<4> = `0`.

**NOTES:**

## 17.0 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 17-1. 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 17-1 is a simplified block diagram of the Timer4 module.

### 17.1 Timer4 Operation

Timer4 can be used as the PWM time base for the PWM mode of the ECCPx/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 T4CKPS<1:0> (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, $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset or Brown-out Reset)

TMR4 is not cleared when T4CON is written.

**REGISTER 17-1: T4CON: TIMER4 CONTROL REGISTER**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | T4OUTPS3 | T4OUTPS2 | T4OUTPS1 | T4OUTPS0 | TMR4ON | T4CKPS1 | T4CKPS0 |
| 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 **Unimplemented:** Read as ‘0’

bit 6-3 **T4OUTPS<3:0>:** Timer4 Output Postscale Select bits
`0000` = 1:1 Postscale
`0001` = 1:2 Postscale
•
•
•
`1111` = 1:16 Postscale

bit 2 **TMR4ON:** Timer4 On bit
`1` = Timer4 is on
`0` = Timer4 is off

bit 1-0 **T4CKPS<1:0>:** Timer4 Clock Prescale Select bits
`00` = Prescaler is 1
`01` = Prescaler is 4
`1x` = Prescaler is 16

## 17.2 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.

## 17.3 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 17-1: TIMER4 BLOCK DIAGRAM**

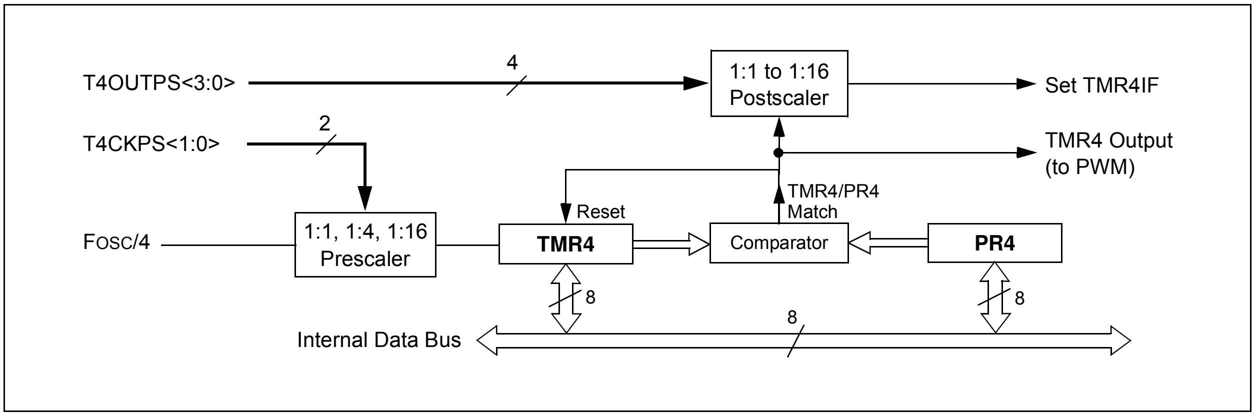



**TABLE 17-1: REGISTERS ASSOCIATED WITH TIMER4 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| TMR4 | Timer4 Register | | | | | | | | 65 |
| T4CON | — | T4OUTPS3 | T4OUTPS2 | T4OUTPS1 | T4OUTPS0 | TMR4ON | T4CKPS1 | T4CKPS0 | 65 |
| PR4 | Timer4 Period Register | | | | | | | | 65 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer4 module.

## 18.0 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 19.0 "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.

Capture and Compare operations described in this chapter apply to all standard and Enhanced CCP modules. The operations of PWM mode, described in **Section 18.4 "PWM Mode"**, apply to CCP4 and CCP5 only.

> **Note:** Throughout this section and **Section 19.0 "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.

**REGISTER 18-1: CCPxCON: CCPx CONTROL REGISTER (CCP4 MODULE, CCP5 MODULE)**

| U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | CCPxX | CCPxY | CCPxM3 | CCPxM2 | CCPxM1 | CCPxM0 |
| 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 7-6 **Unimplemented:** Read as '0'

bit 5-4 **CCPx<X:Y>**: PWM Duty Cycle Bit 1 and Bit 0 for CCPx Module bits

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 (DCx<9:2>) of the duty cycle are found in CCPRxL.

bit 3-0 **CCPxM<3:0>**: 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)
`11xx` = PWM mode

## 18.1 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.

### 18.1.1 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 18-1: CCP MODE – TIMER RESOURCE**

| CCP Mode | Timer Resource |
|---|---|
| Capture | Timer1 or Timer3 |
| Compare | Timer1 or Timer3 |
| PWM | Timer2 or Timer4 |

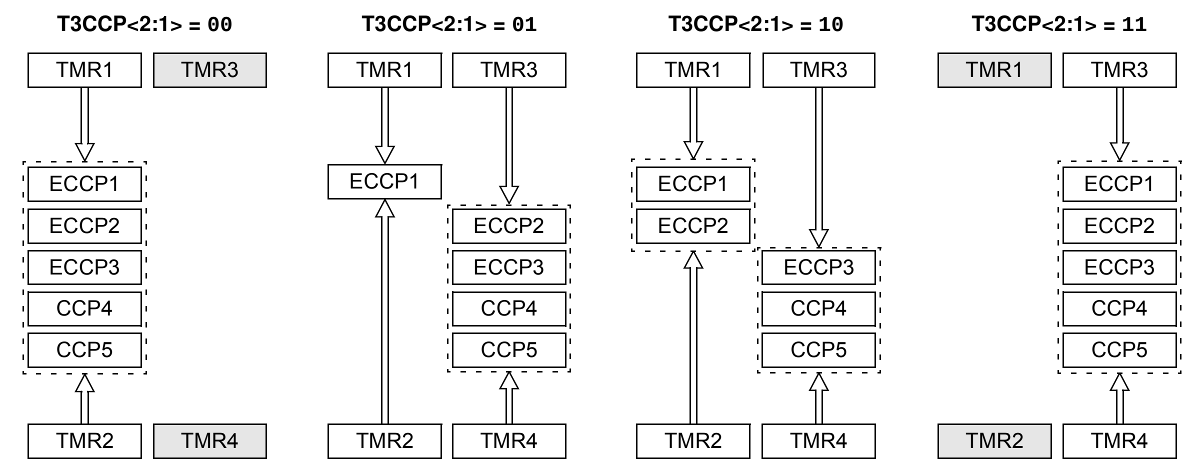
The assignment of a particular timer to a module is determined by the timer to CCP enable bits in the T3CON register (Register 16-1, page 205). Depending on the configuration selected, up to four timers may be active at once, with modules in the same configuration (Capture/Compare or PWM) sharing timer resources. The possible configurations are shown in Figure 18-1.

### 18.1.2 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 11.1.5 "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 18-1: 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.

Timer1 and Timer2 are used for Capture and Compare or PWM operations for ECCP1 and ECCP2 only (depending on the mode selected for each module). Both modules may use a timer as a common time base if they are both 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.

Timer3 is used for all Capture and Compare operations for all CCP modules. Timer4 is used for PWM operations for all CCP modules. Modules may share either timer resource as a common time base.

Timer1 and Timer2 are not available.

## 18.2 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, CCPxM<3:0> (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.

### 18.2.1 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. |
|---|---|

### 18.2.2 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 18.1.1 "CCP Modules and Timer Resources"**).

### 18.2.3 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.

### 18.2.4 CCP PRESCALER

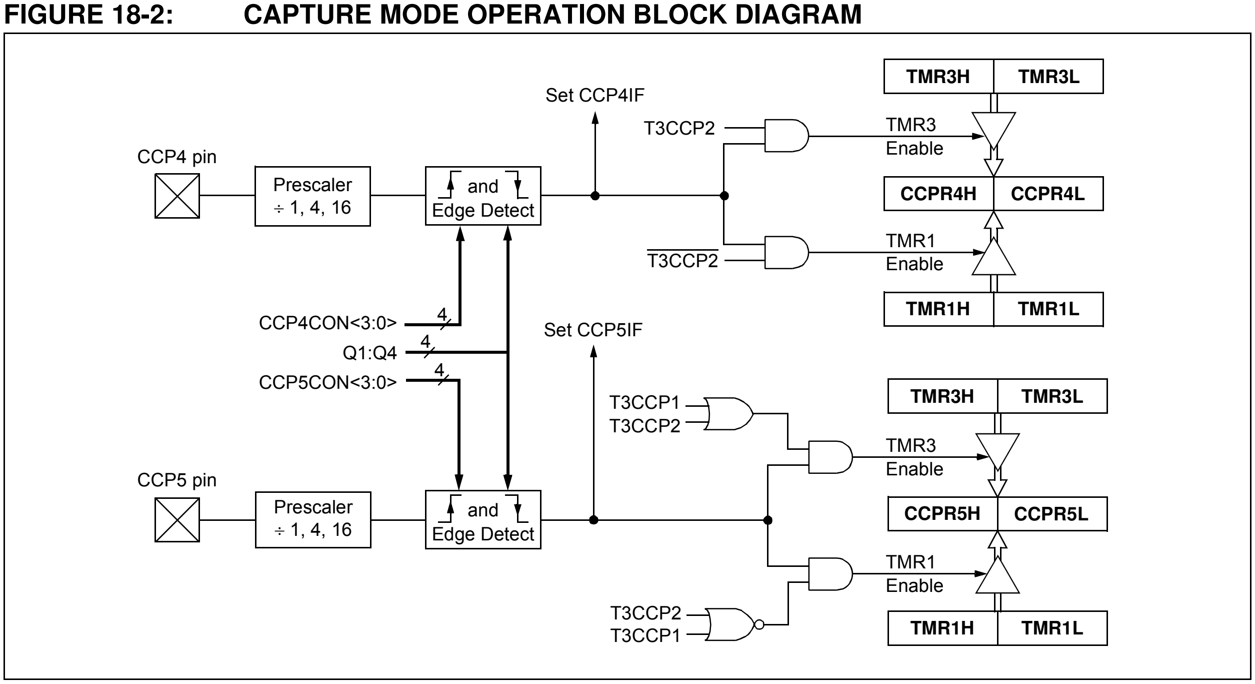
There are four prescaler settings in Capture mode. They are specified as part of the operating mode selected by the mode select bits (CCPxM<3:0>). Whenever the CCP module is turned off or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

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 18-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the "false" interrupt.

**EXAMPLE 18-1: CHANGING BETWEEN CAPTURE PRESCALERS (CCP5 SHOWN)**

```
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 18-2: CAPTURE MODE OPERATION BLOCK DIAGRAM**



## 18.3 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 (CCPxM<3:0>). At the same time, the interrupt flag bit, CCPxIF, is set.

### 18.3.1 CCP PIN CONFIGURATION

The user must configure the CCP pin as an output by clearing the appropriate TRIS bit.

> **Note:** Clearing the CCP5CON register will force the RG4 compare output latch (depending on device configuration) to the default low level. This is not the PORTB or PORTC I/O data latch.

### 18.3.2 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.

### 18.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCPxM<3:0> = `1010`), the corresponding CCP pin is not affected. Only a CCP interrupt is generated, if enabled, and the CCPxIE bit is set.

**FIGURE 18-3: COMPARE MODE OPERATION BLOCK DIAGRAM**

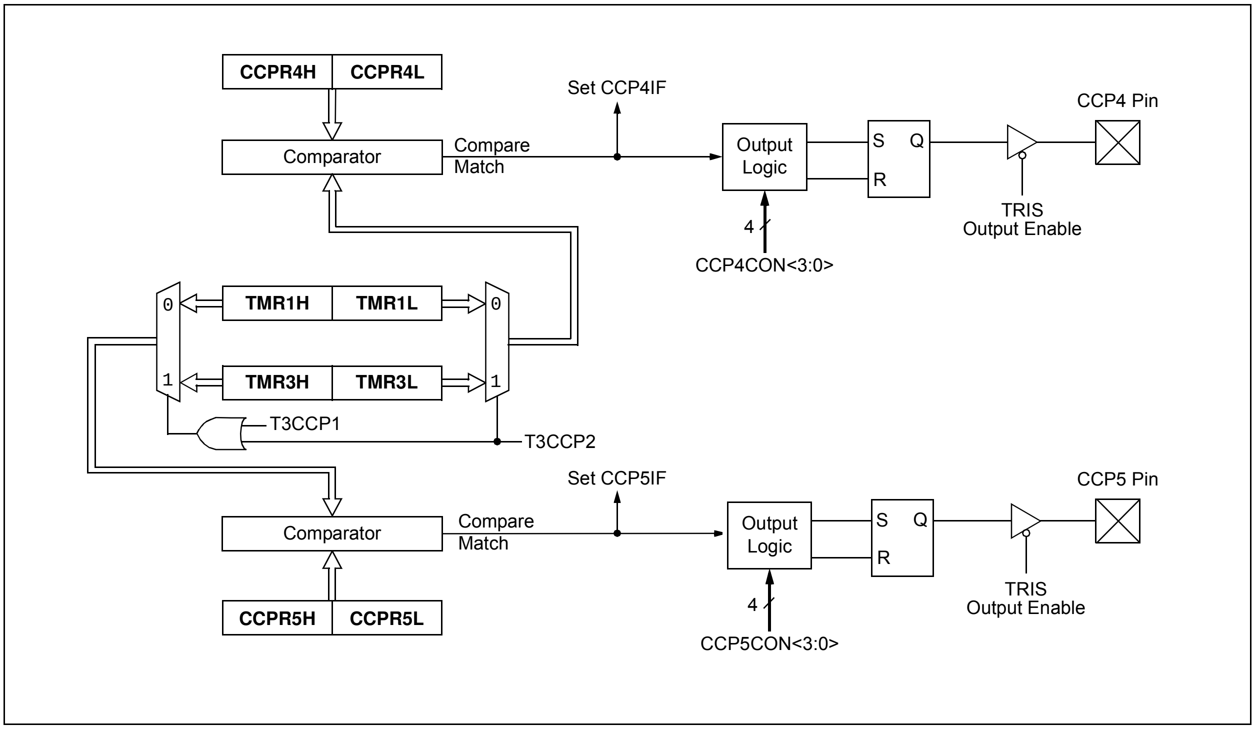


**TABLE 18-2: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| RCON | IPEN | — | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 62 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR2 | OSCFIF | CM2IF | CM1IF | — | BCL1IF | LVDIF | TMR3IF | CCP2IF | 64 |
| PIE2 | OSCFIE | CM2IE | CM1IE | — | BCL1IE | LVDIE | TMR3IE | CCP2IE | 64 |
| IPR2 | OSCFIP | CM2IP | CM1IP | — | BCL1IP | LVDIP | TMR3IP | CCP2IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| TRISG | — | — | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 | 64 |
| TMR1L(1) | Timer1 Register Low Byte | | | | | | | | 62 |
| TMR1H(1) | Timer1 Register High Byte | | | | | | | | 62 |
| ODCON1(2) | — | — | — | CCP5OD | CCP4OD | ECCP3OD | ECCP2OD | ECCP1OD | 62 |
| T1CON(1) | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{T1SYNC}$ | TMR1CS | TMR1ON | 62 |
| TMR3H | Timer3 Register High Byte | | | | | | | | 65 |
| TMR3L | Timer3 Register Low Byte | | | | | | | | 65 |
| T3CON | RD16 | T3CCP2 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{T3SYNC}$ | TMR3CS | TMR3ON | 65 |
| CCPR4L | Capture/Compare/PWM Register 4 Low Byte | | | | | | | | 65 |
| CCPR4H | Capture/Compare/PWM Register 4 High Byte | | | | | | | | 65 |
| CCPR5L | Capture/Compare/PWM Register 5 Low Byte | | | | | | | | 65 |
| CCPR5H | Capture/Compare/PWM Register 5 High Byte | | | | | | | | 65 |
| CCP4CON | — | — | DC4B1 | DC4B0 | CCP4M3 | CCP4M2 | CCP4M1 | CCP4M0 | 65 |
| CCP5CON | — | — | DC5B1 | DC5B0 | CCP5M3 | CCP5M2 | CCP5M1 | CCP5M0 | 65 |

**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.

**2:** Configuration SFR, overlaps with the default SFR at this address; available only when WDTCON<4> = 1.

## 18.4 PWM Mode

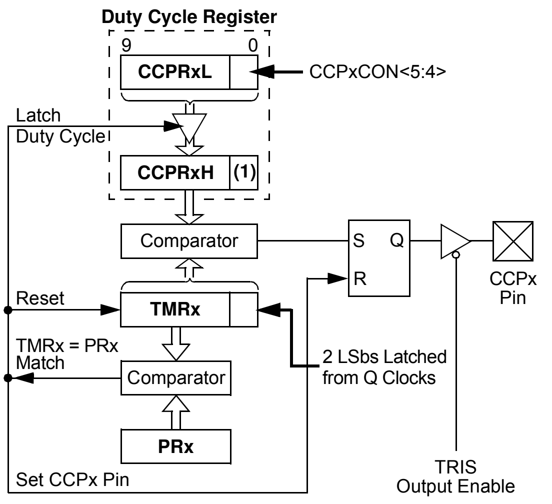
In Pulse-Width Modulation (PWM) mode, the CCP pin produces up to a 10-bit resolution PWM output. Since the CCP4 and CCP5 pins are multiplexed with a PORTG data latch, the appropriate TRISG bit must be cleared to make the CCP4 or CCP5 pin an output.

> **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 18-4 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 18.4.3 "Setup for PWM Operation"**.

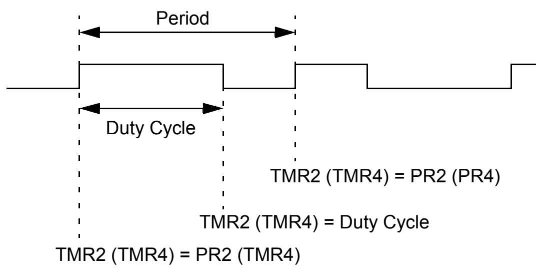
**FIGURE 18-4: SIMPLIFIED PWM BLOCK DIAGRAM**




**Note 1:** The two LSbs of the Duty Cycle register are held by a 2-bit latch that is part of the module's hardware. It is physically separate from the CCPRx registers.

A PWM output (Figure 18-5) 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 18-5: PWM OUTPUT**




### 18.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 (PR4) register. The PWM period can be calculated using Equation 18-1:

**EQUATION 18-1:**

$$\text{PWM Period} = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

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 15.0 "Timer2 Module"** and **Section 17.0 "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.

### 18.4.2 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 18-2 is used to calculate the PWM duty cycle in time.

**EQUATION 18-2:**

$$\text{PWM Duty Cycle} = (\text{CCPRxL:CCPxCON<5:4>}) \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

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 18-3:

**EQUATION 18-3:**

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

| **Note:** | If the PWM duty cycle value is longer than the PWM period, the CCP pin will not be cleared. |
|---|---|

### 18.4.3 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 18-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz**

| PWM Frequency | 2.44 kHz | 9.77 kHz | 39.06 kHz | 156.25 kHz | 312.50 kHz | 416.67 kHz |
|---|---|---|---|---|---|---|
| Timer Prescaler (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | FFh | FFh | FFh | 3Fh | 1Fh | 17h |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.58 |

**TABLE 18-4: REGISTERS ASSOCIATED WITH PWM, TIMER2 AND TIMER4**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| RCON | IPEN | — | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 62 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| TRISG | — | — | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 | 64 |
| TMR2[(1)] | Timer2 Register | | | | | | | | 62 |
| PR2[(1)] | Timer2 Period Register | | | | | | | | 62 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | 62 |
| TMR4 | Timer4 Register | | | | | | | | 65 |
| PR4 | Timer4 Period Register | | | | | | | | 65 |
| T4CON | — | T4OUTPS3 | T4OUTPS2 | T4OUTPS1 | T4OUTPS0 | TMR4ON | T4CKPS1 | T4CKPS0 | 65 |
| CCPR4L | Capture/Compare/PWM Register 4 Low Byte | | | | | | | | 65 |
| CCPR4H | Capture/Compare/PWM Register 4 High Byte | | | | | | | | 65 |
| CCPR5L | Capture/Compare/PWM Register 5 Low Byte | | | | | | | | 65 |
| CCPR5H | Capture/Compare/PWM Register 5 High Byte | | | | | | | | 65 |
| CCP4CON | — | — | DC4B1 | DC4B0 | CCP4M3 | CCP4M2 | CCP4M1 | CCP4M0 | 65 |
| CCP5CON | — | — | DC5B1 | DC5B0 | CCP5M3 | CCP5M2 | CCP5M1 | CCP5M0 | 65 |
| ODCON1[(2)] | — | — | — | CCP5OD | CCP4OD | ECCP3OD | ECCP2OD | ECCP1OD | 62 |

**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.
**2:** Configuration SFR, overlaps with the default SFR at this address; available only when WDTCON<4> = 1.

## 19.0 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 19.4 "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 19-1. 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 19-1: CCPxCON: ECCPx CONTROL REGISTER (ECCP1/ECCP2/ECCP3)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PxM1 | PxM0 | DCxB1 | DCxB0 | CCPxM3 | CCPxM2 | CCPxM1 | CCPxM0 |
| 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 7-6 **PxM<1:0>:** Enhanced PWM Output Configuration bits

If CCPxM<3:2> = `00`, `01`, `10`:
`xx` = PxA is assigned as Capture/Compare input/output; PxB, PxC, PxD are assigned as port pins

If CCPxM<3:2> = `11`:
`00` = Single output: PxA modulated; PxB, PxC, PxD assigned as port pins
`01` = Full-bridge output forward: P1D modulated; P1A active; P1B, P1C inactive
`10` = Half-bridge output: P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins
`11` = Full-bridge output reverse: P1B modulated; P1C active; P1A, P1D inactive

bit 5-4 **DCxB<1:0>:** PWM Duty Cycle Bit 1 and Bit 0 bits

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.

**Note 1:** Implemented only for ECCP1 and ECCP2; the same as '`1010`' for ECCP3.

**REGISTER 19-1: CCPxCON: ECCPx CONTROL REGISTER (ECCP1/ECCP2/ECCP3) (CONTINUED)**

bit 3-0 **CCPxM<3:0>:** Enhanced CCPx Module Mode Select bits
- `0000` = Capture/Compare/PWM off (resets ECCPx module)
- `0001` = Reserved
- `0010` = Compare mode, toggle output on match
- `0011` = Capture mode
- `0100` = Capture mode: Every falling edge
- `0101` = Capture mode: Every rising edge
- `0110` = Capture mode: Every 4th rising edge
- `0111` = Capture mode: Every 16th rising edge
- `1000` = Compare mode: Initialize ECCPx pin low; set output on compare match (set CCPxIF)
- `1001` = Compare mode: Initialize ECCPx pin high; clear output on compare match (set CCPxIF)
- `1010` = Compare mode: Generate software interrupt only; ECCPx pin reverts to I/O state
- `1011` = Compare mode: Trigger special event (ECCPx resets TMR1 or TMR3, sets CCPxIF bit, ECCPx trigger also starts A/D conversion if A/D module is enabled)[(1)]
- `1100` = PWM mode: PxA, PxC active-high; PxB, PxD active-high
- `1101` = PWM mode: PxA, PxC active-high; PxB, PxD active-low
- `1110` = PWM mode: PxA, PxC active-low; PxB, PxD active-high
- `1111` = PWM mode: PxA, PxC active-low; PxB, PxD active-low

**Note 1:** Implemented only for ECCP1 and ECCP2; the same as '`1010`' for ECCP3.

## 19.1 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 EMBx Configuration bits (80-pin devices only)

The pin assignments for the Enhanced CCP modules are summarized in Table 19-1, Table 19-2 and Table 19-3. 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.

### 19.1.1 ECCP1/ECCP3 OUTPUTS AND PROGRAM MEMORY MODE

In 80-pin devices, the use of Extended Microcontroller mode has an indirect effect on the use of ECCP1 and ECCP3 in Enhanced PWM modes. By default, PWM outputs, P1B/P1C and P3B/P3C, are multiplexed to PORTE pins along with the high-order byte of the External Memory Bus. When the bus is active in Extended Microcontroller mode, it overrides the Enhanced CCP outputs and makes them unavailable. Because of this, ECCP1 and ECCP3 can only be used in compatible (single output) PWM modes when the device is in Extended Microcontroller mode and default pin configuration.

An exception to this configuration is when a 12-bit address width is selected for the external bus (EMB<1:0> 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.

### 19.1.2 ECCP2 OUTPUTS AND PROGRAM MEMORY MODES

For 80-pin devices, the program memory mode of the device (**Section 6.1.3 "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.

#### 19.1.3 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 pin for ECCP3 and ECCP1, respectively. The CCP4 and CCP5 modules remain functional but their outputs are overridden.

#### 19.1.4 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 18.1.1 "CCP Modules and Timer Resources"**.

#### 19.1.5 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 11.1.5 "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 WDTCON register (WDTCON<4>).

**TABLE 19-1: PIN CONFIGURATIONS FOR ECCP1**

| ECCP Mode | CCP1CON Configuration | RC2 | RE6 | RE5 | RG4 | RH7 | RH6 |
|---|---|---|---|---|---|---|---|
| **All PIC18F6XJ1X Devices:** | | | | | | | |
| Compatible CCP | 00xx 11xx | ECCP1 | RE6 | RE5 | RG4/CCP5 | N/A | N/A |
| Dual PWM | 10xx 11xx | P1A | P1B | RE5 | RG4/CCP5 | N/A | N/A |
| Quad PWM[(1)] | x1xx 11xx | P1A | P1B | P1C | P1D | N/A | N/A |
| **PIC18F8XJ1X Devices, ECCPMX = 0, Microcontroller mode:** | | | | | | | |
| Compatible CCP | 00xx 11xx | ECCP1 | RE6/AD14 | RE5/AD13 | RG4/CCP5 | RH7/AN15 | RH6/AN14 |
| Dual PWM | 10xx 11xx | P1A | RE6/AD14 | RE5/AD13 | RG4/CCP5 | P1B | RH6/AN14 |
| Quad PWM[(1)] | x1xx 11xx | P1A | RE6/AD14 | RE5/AD13 | P1D | P1B | P1C |
| **PIC18F8XJ1X Devices, ECCPMX = 1, Extended Microcontroller mode, 16-Bit or 20-Bit Address Width:** | | | | | | | |
| Compatible CCP | 00xx 11xx | ECCP1 | RE6/AD14 | RE5/AD13 | RG4/CCP5 | RH7/AN15 | RH6/AN14 |
| **PIC18F8XJ1X Devices, ECCPMX = 1, Microcontroller mode or Extended Microcontroller mode, 12-Bit Address Width:** | | | | | | | |
| Compatible CCP | 00xx 11xx | ECCP1 | RE6/AD14 | RE5/AD13 | RG4/CCP5 | RH7/AN15 | RH6/AN14 |
| Dual PWM | 10xx 11xx | P1A | P1B | RE5/AD13 | RG4/CCP5 | RH7/AN15 | RH6/AN14 |
| Quad PWM[(1)] | x1xx 11xx | P1A | P1B | P1C | P1D | RH7/AN15 | RH6/AN14 |

**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, the CCP5 module's output is overridden by P1D; otherwise, CCP5 is fully operational.

**TABLE 19-2: PIN CONFIGURATIONS FOR ECCP2**

| ECCP Mode | CCP2CON Configuration | RB3 | RC1 | RE7 | RE2 | RE1 | RE0 |
|---|---|---|---|---|---|---|---|
| **All Devices, CCP2MX = 1, Either Operating mode:** | | | | | | | |
| Compatible CCP | `00xx 11xx` | RB3/INT3 | ECCP2 | RE7 | RE2 | RE1 | RE0 |
| Dual PWM | `10xx 11xx` | RB3/INT3 | P2A | RE7 | P2B | RE1 | RE0 |
| Quad PWM | `x1xx 11xx` | RB3/INT3 | P2A | RE7 | P2B | P2C | P2D |
| **All Devices, CCP2MX = 0, Microcontroller mode:** | | | | | | | |
| Compatible CCP | `00xx 11xx` | RB3/INT3 | RC1/T1OS1 | ECCP2 | RE2 | RE1 | RE0 |
| Dual PWM | `10xx 11xx` | RB3/INT3 | RC1/T1OS1 | P2A | P2B | RE1 | RE0 |
| Quad PWM | `x1xx 11xx` | RB3/INT3 | RC1/T1OS1 | P2A | P2B | P2C | P2D |
| **PIC18F8XJ1X Devices, CCP2MX = 0, Extended Microcontroller mode:** | | | | | | | |
| Compatible CCP | `00xx 11xx` | ECCP2 | RC1/T1OS1 | RE7/AD15 | RE2/$\overline{CS}$ | RE1/$\overline{WR}$ | RE0/$\overline{RD}$ |
| Dual PWM | `10xx 11xx` | P2A | RC1/T1OS1 | RE7/AD15 | P2B | RE1/$\overline{WR}$ | RE0/$\overline{RD}$ |
| Quad PWM | `x1xx 11xx` | P2A | RC1/T1OS1 | RE7/AD15 | P2B | P2C | P2D |

**Legend:** x = Don't care. Shaded cells indicate pin assignments not used by ECCP2 in a given mode.

**TABLE 19-3: PIN CONFIGURATIONS FOR ECCP3**

| ECCP Mode | CCP3CON Configuration | RG0 | RE4 | RE3 | RG3 | RH5 | RH4 |
|---|---|---|---|---|---|---|---|
| **PIC18F6XJ1X Devices:** | | | | | | | |
| Compatible CCP | `00xx 11xx` | ECCP3 | RE4 | RE3 | RG3/CCP4 | N/A | N/A |
| Dual PWM | `10xx 11xx` | P3A | P3B | RE3 | RG3/CCP4 | N/A | N/A |
| Quad PWM(1) | `x1xx 11xx` | P3A | P3B | P3C | P3D | N/A | N/A |
| **PIC18F8XJ1X Devices, ECCPMX = 0, Microcontroller mode:** | | | | | | | |
| Compatible CCP | `00xx 11xx` | ECCP3 | RE6/AD14 | RE5/AD13 | RG3/CCP4 | RH7/AN15 | RH6/AN14 |
| Dual PWM | `10xx 11xx` | P3A | RE6/AD14 | RE5/AD13 | RG3/CCP4 | P3B | RH6/AN14 |
| Quad PWM(1) | `x1xx 11xx` | P3A | RE6/AD14 | RE5/AD13 | P3D | P3B | P3C |
| **PIC18F8XJ1X Devices, ECCPMX = 1, Extended Microcontroller mode, 16-Bit or 20-Bit Address Width:** | | | | | | | |
| Compatible CCP | `00xx 11xx` | ECCP3 | RE6/AD14 | RE5/AD13 | RG3/CCP4 | RH7/AN15 | RH6/AN14 |
| **PIC18F8XJ1X Devices, ECCPMX = 1, Microcontroller mode or Extended Microcontroller mode, 12-Bit Address Width:** | | | | | | | |
| Compatible CCP | `00xx 11xx` | ECCP3 | RE4/AD12 | RE3/AD11 | RG3/CCP4 | RH5/AN13 | RH4/AN12 |
| Dual PWM | `10xx 11xx` | P3A | P3B | RE3/AD11 | RG3/CCP4 | RH5/AN13 | RH4/AN12 |
| Quad PWM(1) | `x1xx 11xx` | P3A | P3B | P3C | P3D | RH5/AN13 | RH4/AN12 |

**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, the CCP4 module's output is overridden by P1D; otherwise, CCP4 is fully operational.

## 19.2 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 18.2 "Capture Mode"** and **Section 18.3 "Compare Mode"**.

### 19.2.1 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 (CCPxM<3:0> = 1010).

| **Note:** | The Special Event Trigger from ECCP2 will not set the Timer1 or Timer3 interrupt flag bits. |
|---|---|

## 19.3 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 18.4 "PWM Mode"**. This is also sometimes referred to as "Compatible CCP" mode as in Tables 19-1 through 19-3.

| **Note:** | When setting up single output PWM operations, users are free to use either of the processes described in **Section 18.4.3 "Setup for PWM Operation"** or **Section 19.4.9 "Setup for PWM Operation"**. The latter is more generic but will work for either single or multi-output PWM. |
|---|---|

## 19.4 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 PxM<1:0> and CCPxM<3:0> 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 19-1 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.

### 19.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the equation:

**EQUATION 19-1:**

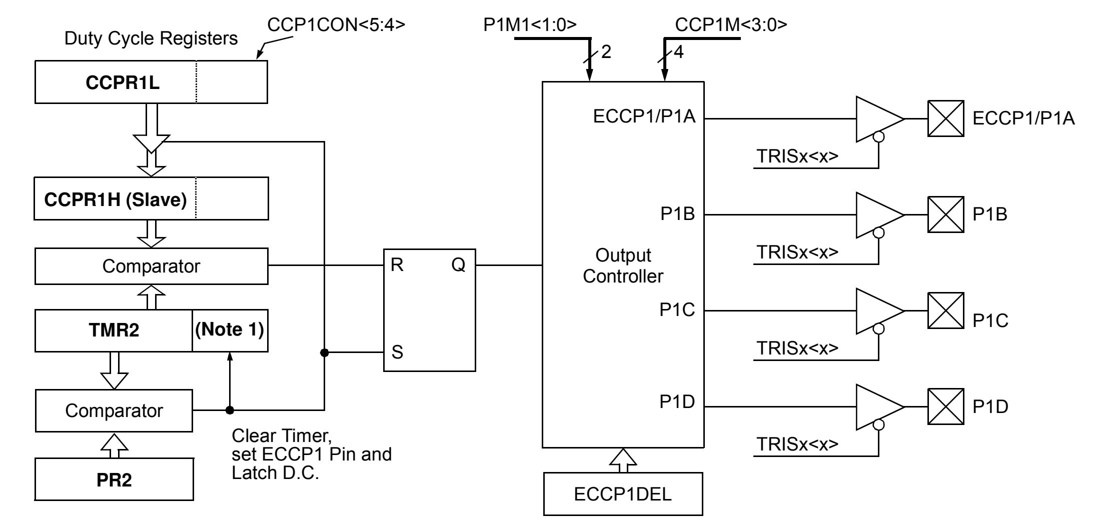
$$\text{PWM Period} = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[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 15.0 "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 19-1: 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.

### 19.4.2 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 19-2:**

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

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 Equation 19-3.

**EQUATION 19-3:**

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

**Note:** If the PWM duty cycle value is longer than the PWM period, the ECCP1 pin will not be cleared.

### 19.4.3 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 19.4 "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 19-2.

**TABLE 19-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz**

| PWM Frequency | 2.44 kHz | 9.77 kHz | 39.06 kHz | 156.25 kHz | 312.50 kHz | 416.67 kHz |
|---|---|---|---|---|---|---|
| Timer Prescaler (1, 4, 16) | 16 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | FFh | FFh | FFh | 3Fh | 1Fh | 17h |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.58 |

**FIGURE 19-2: PWM OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)**

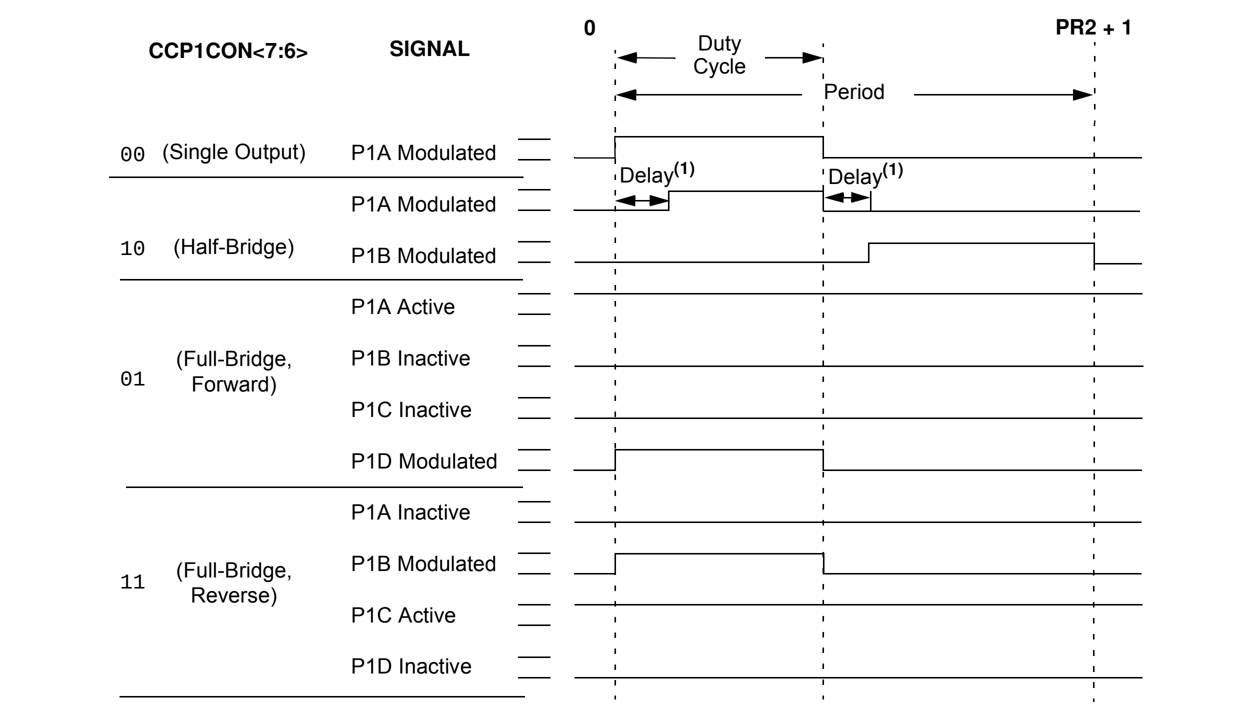
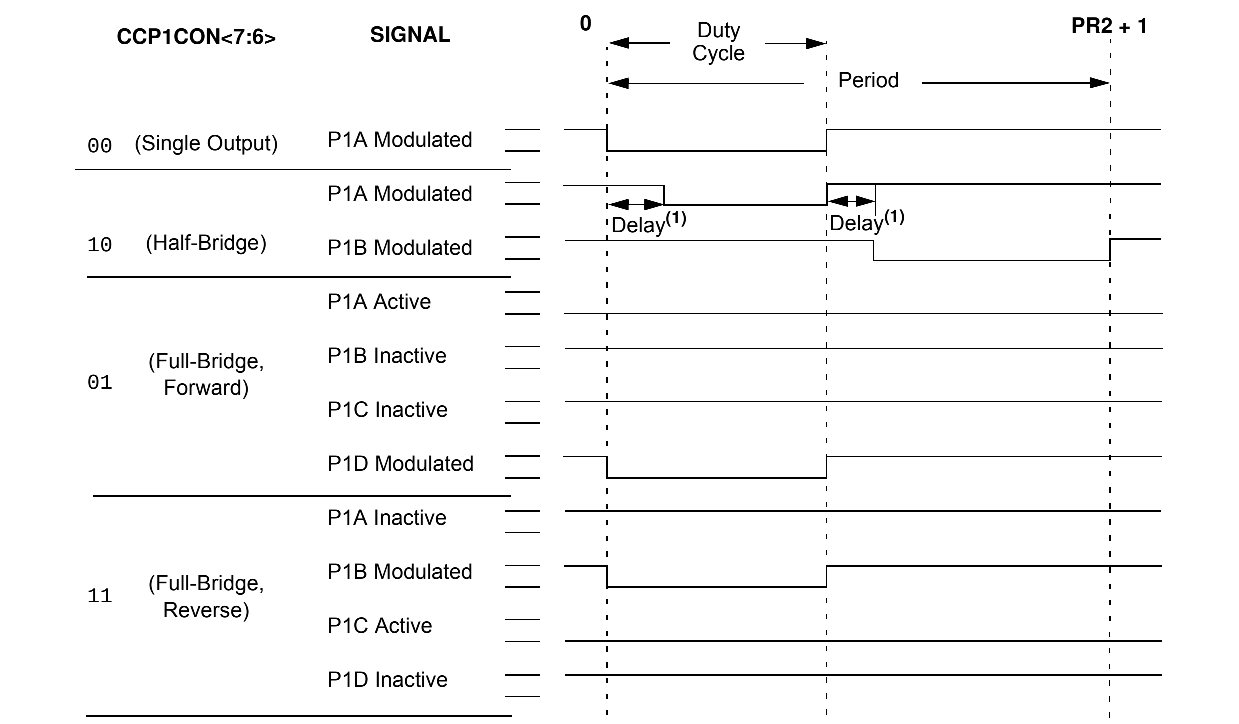



**FIGURE 19-3: PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)**




**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 19.4.6 “Programmable Dead-Band Delay”**).

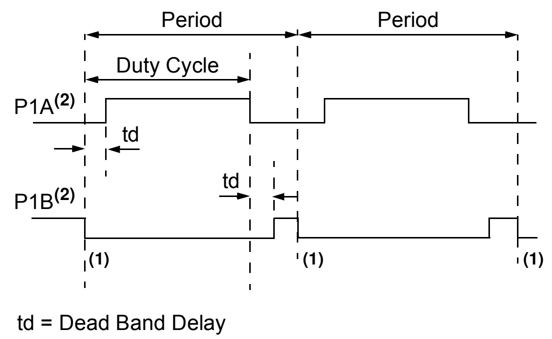
### 19.4.4 HALF-BRIDGE MODE

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 19-4). This mode can be used for half-bridge applications, as shown in Figure 19-5, 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 19.4.6 "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.

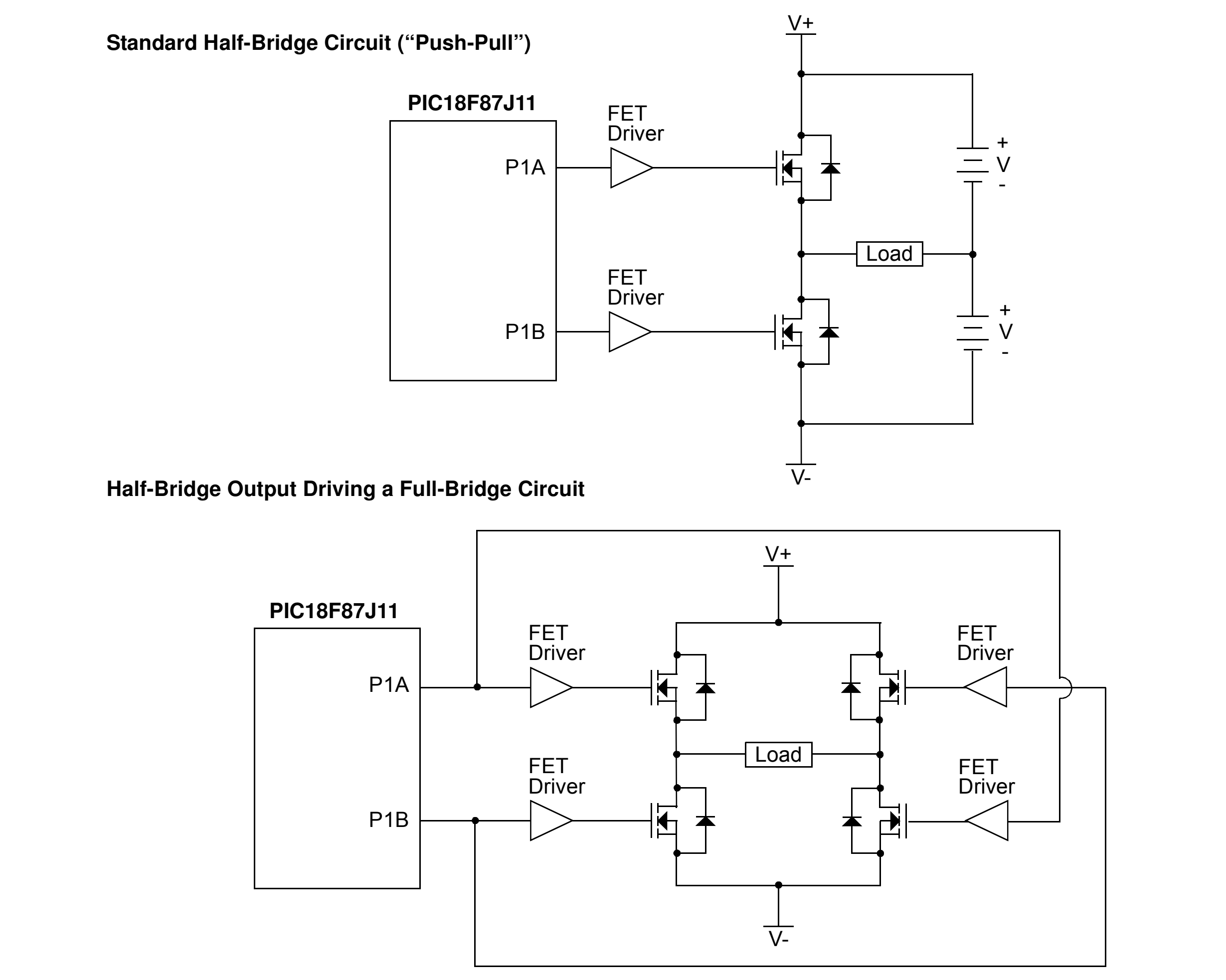
**FIGURE 19-4: HALF-BRIDGE PWM OUTPUT**




**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

**2:** Output signals are shown as active-high.

**FIGURE 19-5: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS**

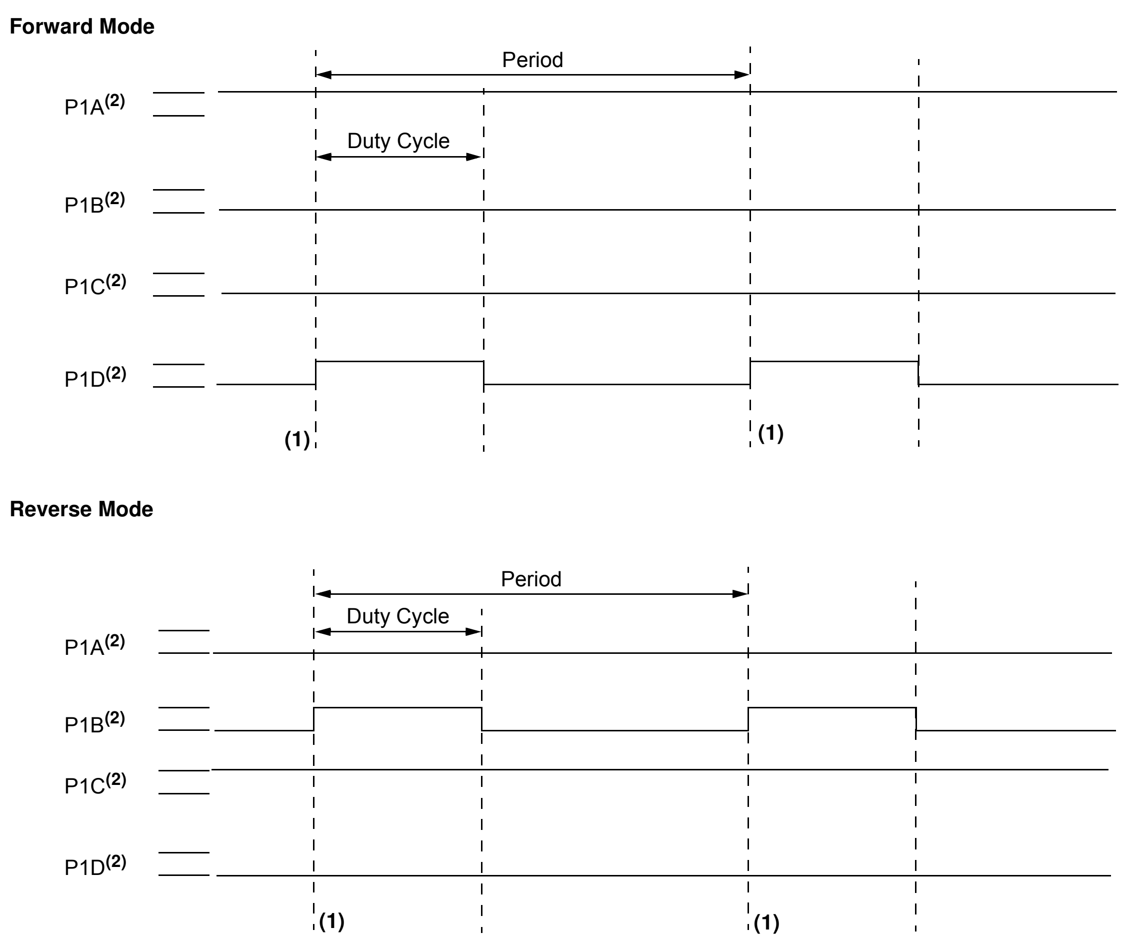


### 19.4.5 FULL-BRIDGE MODE

In Full-Bridge Output mode, four pins are used as outputs; however, only two outputs are active at a time. In the Forward mode, the P1A pin is continuously active and the P1D pin is modulated. In the Reverse mode, the P1C pin is continuously active and the P1B pin is modulated. These are illustrated in Figure 19-6.

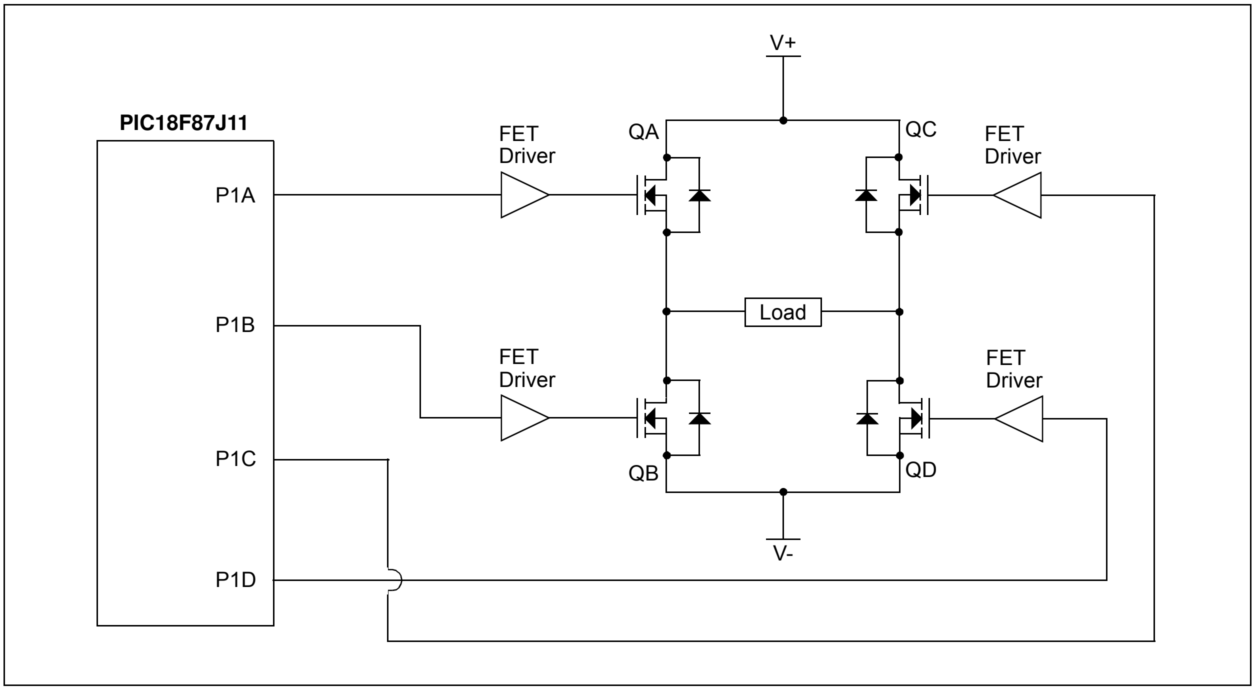
P1A, P1B, P1C and P1D outputs are multiplexed with the port pins, as described in Table 19-1, Table 19-2 and Table 19-3. The corresponding TRIS bits must be cleared to make the P1A, P1B, P1C and P1D pins outputs.

**FIGURE 19-6: FULL-BRIDGE PWM OUTPUT**




**Note 1:** At this time, the TMR2 register is equal to the PR2 register.
**2:** Output signal is shown as active-high.

**FIGURE 19-7: EXAMPLE OF FULL-BRIDGE OUTPUT APPLICATION**




### 19.4.5.1 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 TOSC * (Timer2 Prescale Value) before the next PWM period begins. The Timer2 prescaler will be either 1, 4 or 16, depending on the value of the T2CKPSx 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 19-8.

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.

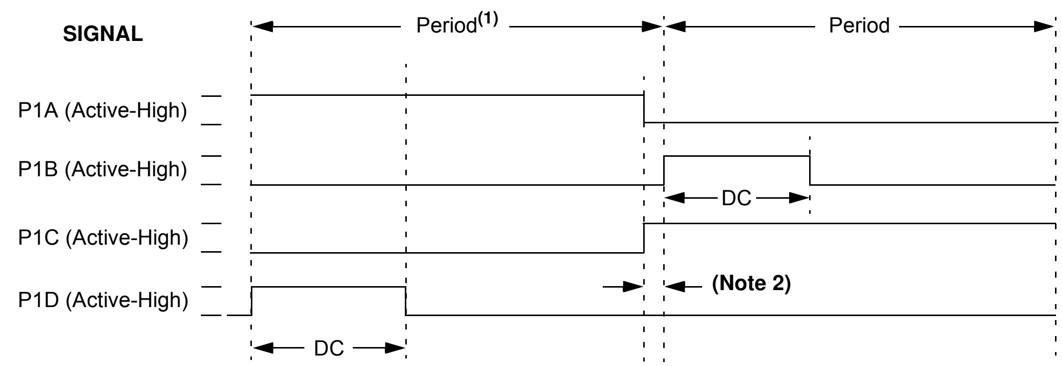
Figure 19-9 shows an example where the PWM direction changes from forward to reverse at a near 100% duty cycle. At time, t1, the outputs, P1A and P1D, become inactive, while output, P1C, becomes active. In this example, since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current may flow through power devices, QC and QD (see Figure 19-7), 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.

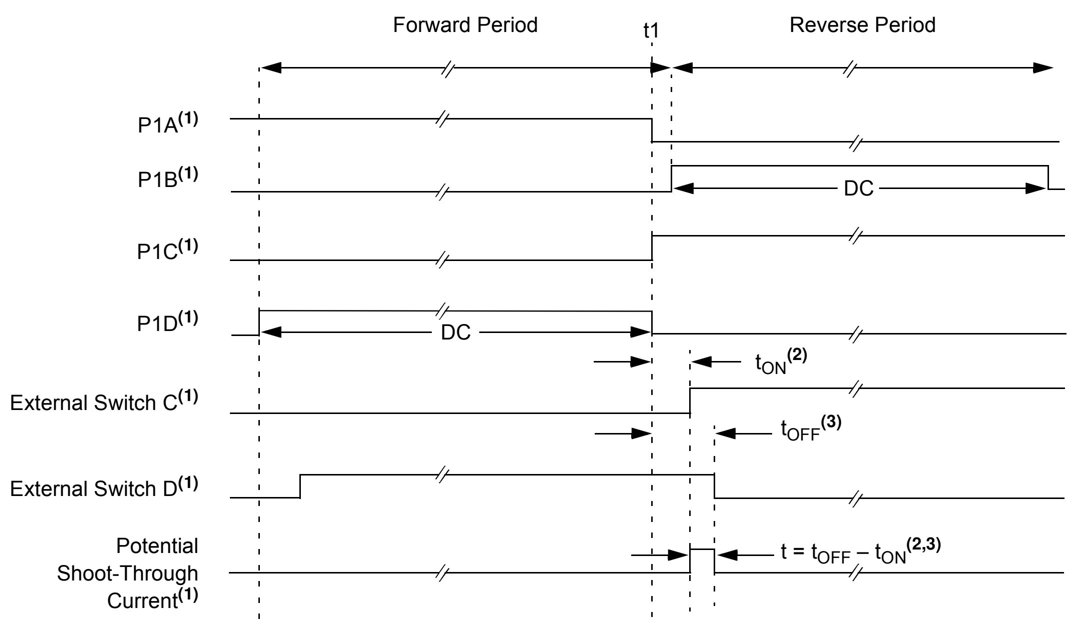
**FIGURE 19-8: PWM DIRECTION CHANGE**




**Note 1:** The direction bit in the ECCP1 Control register (CCP1CON<7>) is written at any time during the PWM cycle.

**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 19-9: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE**




**Note 1:** All signals are shown as active-high.

**2:** $t_{ON}$ is the turn-on delay of power switch QC and its driver.

**3:** $t_{OFF}$ is the turn-off delay of power switch QD and its driver.

### 19.4.6 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 19-4 for illustration). The lower seven bits of the ECCPxDEL register (Register 19-2) set the delay period in terms of microcontroller instruction cycles ($T_{CY}$ or 4 $T_{OSC}$).

### 19.4.7 ENHANCED PWM AUTO-SHUTDOWN

When the ECCP1 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 ECCP1AS<2:0> bits (ECCP1AS<6:4>).

When a shutdown occurs, the output pins are asynchronously placed in their shutdown states, specified by the PSS1AC<1:0> and PSS1BD<1:0> bits (ECCP1AS<3:0>). 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 19-2: ECCPxDEL: ECCPx PWM DELAY REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PxRSEN | PxDC6 | PxDC5 | PxDC4 | PxDC3 | PxDC2 | PxDC1 | PxDC0 |
| 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 **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 **PxDC<6:0>:** PWM Delay Count bits

Delay time, in number of FOSC/4 (4 * TOSC) cycles, between the scheduled and actual time for a PWM signal to transition to active.

### REGISTER 19-3: ECCPxAS: ECCPx AUTO-SHUTDOWN CONTROL REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ECCPxASE | ECCPxAS2 | ECCPxAS1 | ECCPxAS0 | PSSxAC1 | PSSxAC0 | PSSxBD1 | PSSxBD0 |
| 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 **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 **ECCPxAS<2:0>:** 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 **PSSxAC<1:0>:** 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 **PSSxBD<1:0>:** 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

#### 19.4.7.1 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 19-10), 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 19-11), 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.

### 19.4.8 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 CCP1M<1:0> 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.

**FIGURE 19-10: PWM AUTO-SHUTDOWN (P1RSEN = `1`, AUTO-RESTART ENABLED)**

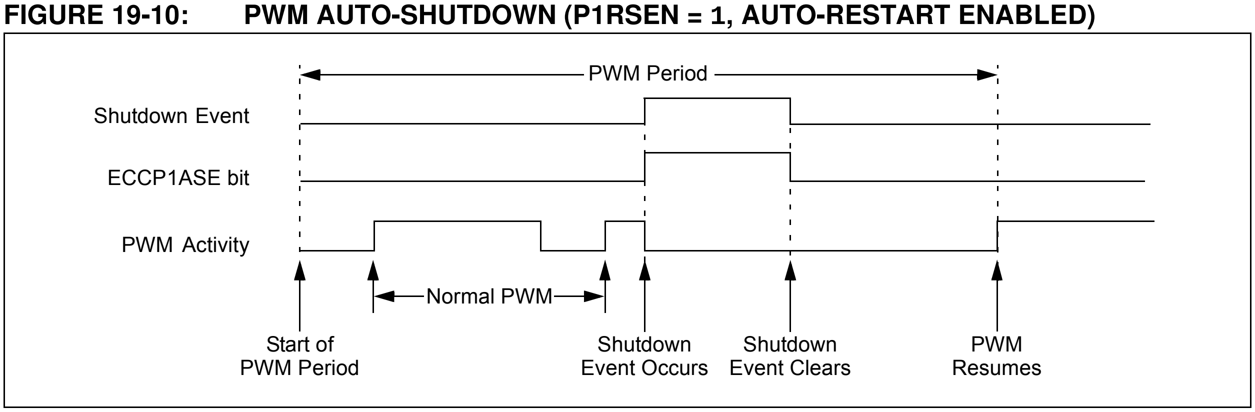
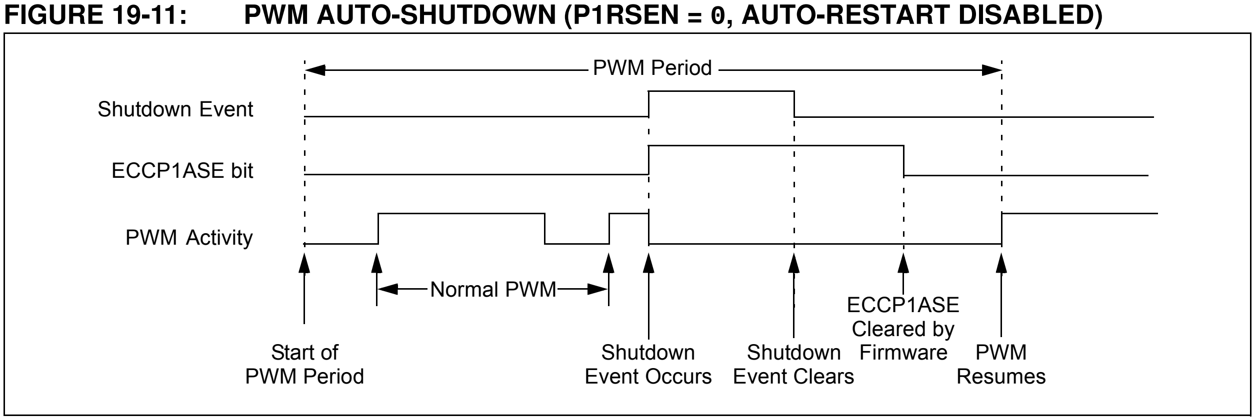



**FIGURE 19-11: PWM AUTO-SHUTDOWN (P1RSEN = `0`, AUTO-RESTART DISABLED)**



### 19.4.9 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 PxM<1:0> bits.
   - Select the polarities of the PWM output signals with the CCPxM<3:0> 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 ECCPxAS<2:0> bits.
   - Select the shutdown states of the PWM output pins using the PSSxAC<1:0> and PSSxBD<1:0> 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 TnCKPSx 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>).

### 19.4.10 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 19-5: REGISTERS ASSOCIATED WITH ECCP MODULES AND TIMER1 TO TIMER4

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| RCON | IPEN | — | $\overline{\text{CM}}$ | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 62 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR2 | OSCFIF | CM2IF | CM1IF | — | BCL1IF | LVDIF | TMR3IF | CCP2IF | 64 |
| PIE2 | OSCFIE | CM2IE | CM1IE | — | BCL1IE | LVDIE | TMR3IE | CCP2IE | 64 |
| IPR2 | OSCFIP | CM2IP | CM1IP | — | BCL1IP | LVDIP | TMR3IP | CCP2IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 64 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 64 |
| TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 | 64 |
| TRISG | — | — | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 | 64 |
| TRISH[1] | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 | 64 |
| TMR1L[3] | Timer1 Register Low Byte | | | | | | | | 62 |
| TMR1H[3] | Timer1 Register High Byte | | | | | | | | 62 |
| ODCON1[4] | — | — | — | CCP5OD | CCP4OD | ECCP3OD | ECCP2OD | ECCP1OD | 62 |
| T1CON[3] | RD16 | T1RUN | T1CKPS1 | T1CKPS0 | T1OSCEN | $\overline{\text{T1SYNC}}$ | TMR1CS | TMR1ON | 62 |
| TMR2[3] | Timer2 Register | | | | | | | | 62 |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | 62 |
| PR2[3] | Timer2 Period Register | | | | | | | | 62 |
| TMR3L | Timer3 Register Low Byte | | | | | | | | 65 |
| TMR3H | Timer3 Register High Byte | | | | | | | | 65 |
| T3CON | RD16 | T3CCP2 | T3CKPS1 | T3CKPS0 | T3CCP1 | $\overline{\text{T3SYNC}}$ | TMR3CS | TMR3ON | 65 |
| TMR4 | Timer4 Register | | | | | | | | 65 |
| T4CON | — | T4OUTPS3 | T4OUTPS2 | T4OUTPS1 | T4OUTPS0 | TMR4ON | T4CKPS1 | T4CKPS0 | 65 |
| PR4[3] | Timer4 Period Register | | | | | | | | 65 |
| CCPRxL[2] | Capture/Compare/PWM Register x Low Byte | | | | | | | | 63 |
| CCPRxH[2] | Capture/Compare/PWM Register x High Byte | | | | | | | | 63, |
| CCPxCON[2] | PxM1 | PxM0 | DCxB1 | DCxB0 | CCPxM3 | CCPxM2 | CCPxM1 | CCPxM0 | 63 |
| ECCPxAS[2] | ECCPxASE | ECCPxAS2 | ECCPxAS1 | ECCPxAS0 | PSSxAC1 | PSSxAC0 | PSSxBD1 | PSSxBD0 | 63 |
| ECCPxDEL[2] | PxRSEN | PxDC6 | PxDC5 | PxDC4 | PxDC3 | PxDC2 | PxDC1 | PxDC0 | 63 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during ECCP operation.

**Note 1:** This register is 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 the default SFR at this address; available only when WDTCON<4> = 1.

**NOTES:**

# 20.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

## 20.1 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 (I$^2$C™)
  - Full Master mode
  - Slave mode (with general address call)

The I$^2$C 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. |
|---|---|

## 20.2 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 I$^2$C 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. |
|---|---|

## 20.3 SPI Mode

| **Note:** | Disabling the MSSPx module by clearing the SSPEN (SSPxCON1<5>) bit may not reset the module. It is recommended to clear the SSPxSTAT, SSPxCON1 and SSPxCON2 registers, and select the mode prior to setting the SSPEN bit to enable the MSSPx module. |
|---|---|

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:

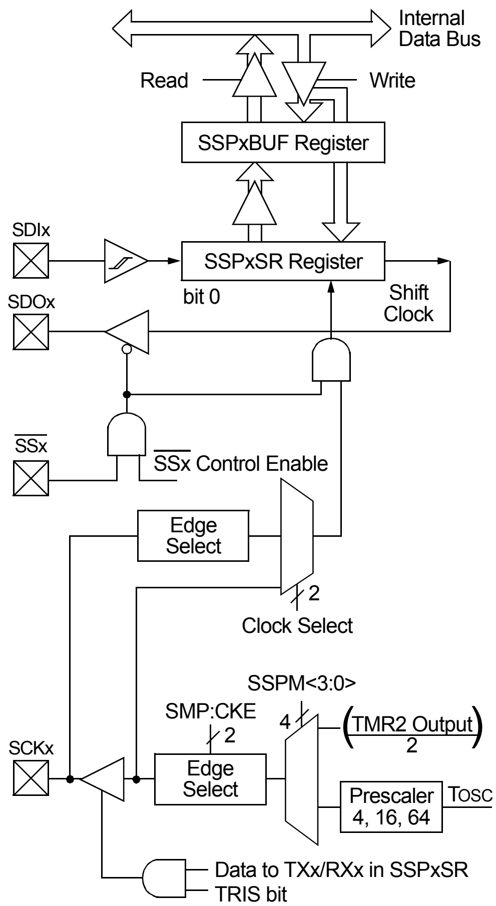
- Serial Data Out (SDOx) – RC5/SDO1 or RD4/PMD4/SDO2
- Serial Data In (SDIx) – RC4/SDI1/SDA1 or RD5/PMD5/SDI2/SDA2
- Serial Clock (SCKx) – RC3/SCK1/SCL1 or RD6/PMD6/SCK2/SCL2

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{\text{SSx}}$) – RF7/$\overline{\text{SS1}}$ or RD7/PMD7/$\overline{\text{SS2}}$

Figure 20-1 shows the block diagram of the MSSPx module when operating in SPI mode.

**FIGURE 20-1: MSSPx BLOCK DIAGRAM (SPI MODE)**




**Note:** Only port I/O names are used in this diagram for the sake of brevity. Refer to the text for a full list of multiplexed functions.

### 20.3.1 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.

> **Note:** Because the SSPxBUF register is double-buffered, using read-modify-write instructions, such as `BCF`, `COMF`, etc., will not work.
>
> Similarly, when debugging under an in-circuit debugger, performing actions that cause reads of SSPxBUF (mouse hovering, watch, etc.) can consume data that the application code was expecting to receive.

### REGISTER 20-1: SSPxSTAT: MSSPx STATUS REGISTER (SPI MODE)

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SMP | CKE(1) | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -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 is sampled at the end of data output time
0 = Input data is sampled at the 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 the active to Idle clock state
0 = Transmit occurs on transition from the Idle to active clock state

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit
Used in I$^2$C™ mode only.

bit 4 **P:** Stop bit
Used in I$^2$C mode only. This bit is cleared when the MSSPx module is disabled and SSPEN is cleared.

bit 3 **S:** Start bit
Used in I$^2$C mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{\text{Write}}$ Information bit
Used in I$^2$C mode only.

bit 1 **UA:** Update Address bit
Used in I$^2$C mode only.

bit 0 **BF:** Buffer Full Status bit (Receive mode only)
1 = Receive is complete, SSPxBUF is full
0 = Receive is not complete, SSPxBUF is empty

**Note 1:** The polarity of the clock state is set by the CKP bit (SSPxCON1<4>).

### REGISTER 20-2: SSPxCON1: MSSPx CONTROL REGISTER 1 (SPI MODE)

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| WCOL | SSPOV(1) | SSPEN(2) | CKP | SSPM3(3) | SSPM2(3) | SSPM1(3) | SSPM0(3) |
| 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 **WCOL:** Write Collision Detect bit
`1` = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
`0` = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit(1)
SPI Slave mode:
`1` = A new byte is received while the SSPxBUF register is still holding the previous data. In case of overflow, the data in SSPxSR is lost. Overflow can only occur in Slave mode. The user must read the SSPxBUF, even if only transmitting data, to avoid setting an overflow (must be cleared in software).
`0` = No overflow

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit(2)
`1` = Enables serial port and configures SCKx, SDOx, SDIx and $\overline{SSx}$ as serial port pins
`0` = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit
`1` = Idle state for clock is a high level
`0` = Idle state for clock is a low level

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits(3)
`0101` = SPI Slave mode; Clock = SCKx pin, $\overline{SSx}$ pin control is disabled, $\overline{SSx}$ can be used as an I/O pin
`0100` = SPI Slave mode; Clock = SCKx pin, $\overline{SSx}$ pin control is enabled
`0011` = SPI Master mode; Clock = TMR2 Output/2
`0010` = SPI Master mode; Clock = FOSC/64
`0001` = SPI Master mode; Clock = FOSC/16
`0000` = SPI Master mode; Clock = FOSC/4

**Note 1:** In Master mode, the overflow bit is not set, since each new reception (and transmission) is initiated by writing to the SSPxBUF register.
**2:** When enabled, these pins must be properly configured as inputs or outputs.
**3:** Bit combinations not specifically listed here are either reserved or implemented in $I^2C$™ mode only.

### 20.3.2 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, SSPxIF, are set. This double-buffering of the received data (SSPxBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPxBUF register during transmission/reception of data will be ignored and the Write Collision Detect bit, WCOL (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 20-1 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.

### 20.3.3 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 11.1.5 "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 20-1: 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
```

### 20.3.4 ENABLING SPI I/O

To enable the serial port, MSSPx Enable bit, SSPEN (SSPxCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPxCON registers and then set the SSPEN bit. This configures the SDIx, SDOx, SCKx and $\overline{\text{SSx}}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDIx must have the TRISC<4> or TRISD<5> bit set
- 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
- $\overline{\text{SSx}}$ must have the TRISF<7> or TRISD<7> bit set

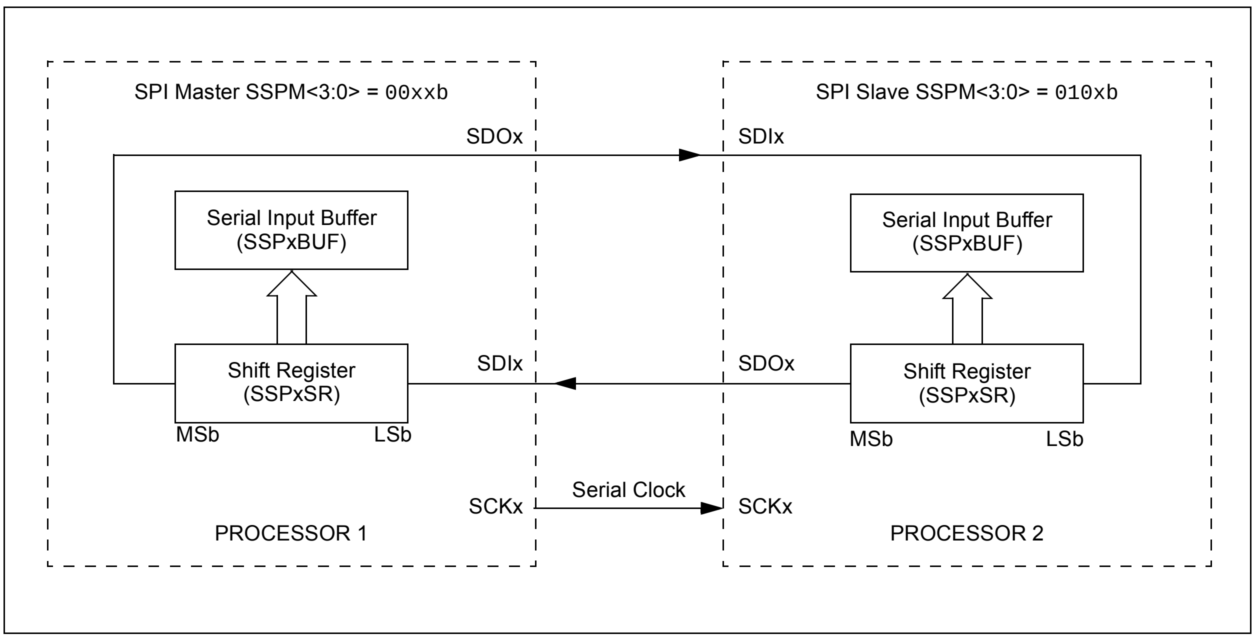
Any serial port function that is not desired may be overridden by programming the corresponding Data Direction (TRIS) register to the opposite value.

### 20.3.5 TYPICAL CONNECTION

Figure 20-2 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 20-2: SPI MASTER/SLAVE CONNECTION**



### 20.3.6 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 20-2) 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.

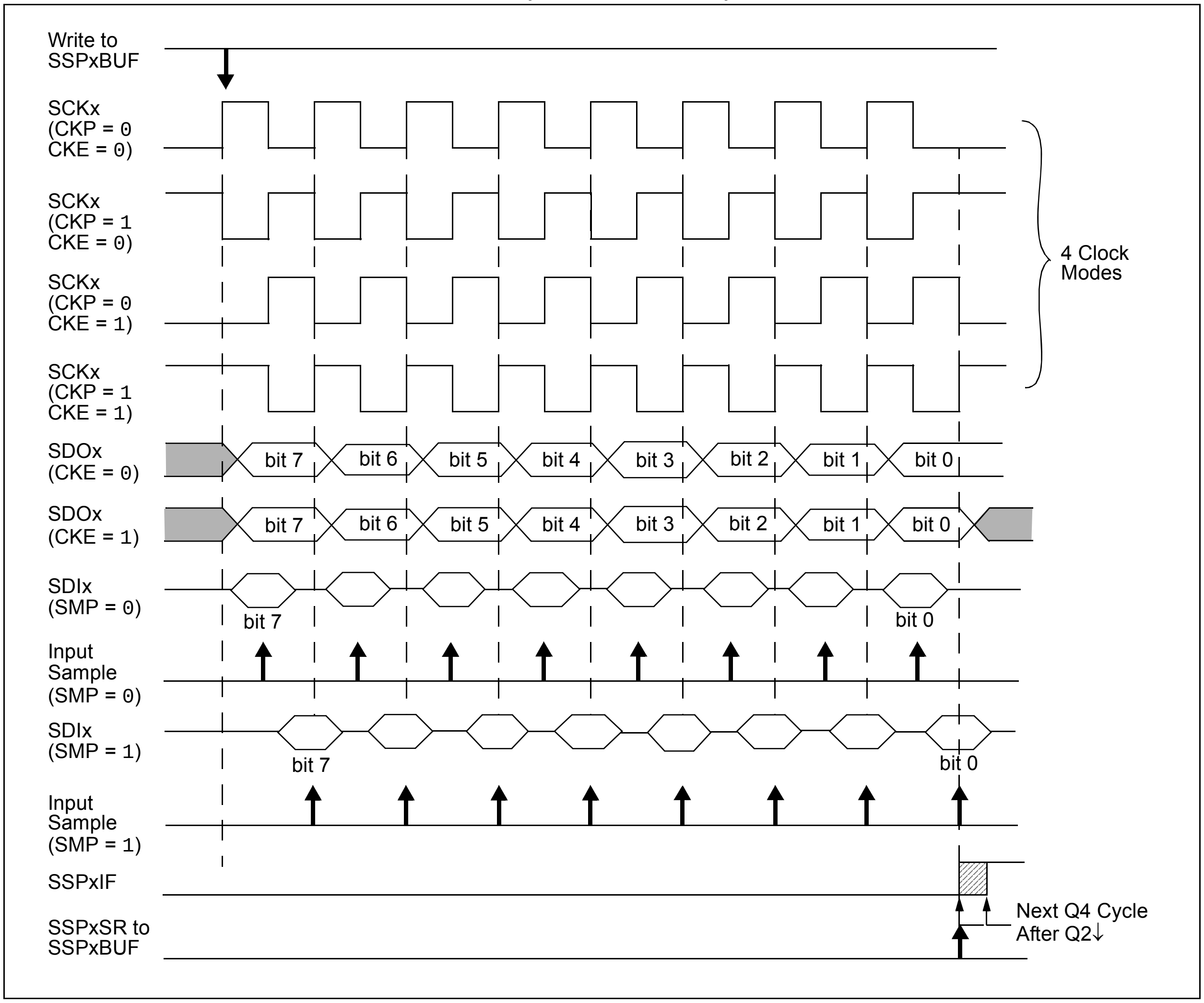
The clock polarity is selected by appropriately programming the CKP bit (SSPxCON1<4>). This then, would give waveforms for SPI communication as shown in Figure 20-3, Figure 20-5 and Figure 20-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to 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 20-3 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 20-3: SPI MODE WAVEFORM (MASTER MODE)**



### 20.3.7 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.

### 20.3.8 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 the SSx pin control enabled, (SSPxCON1<3:0> = 0100), the SPI module will reset if the SSx pin is set to VDD.
>
> **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 20-4: SLAVE SYNCHRONIZATION WAVEFORM**

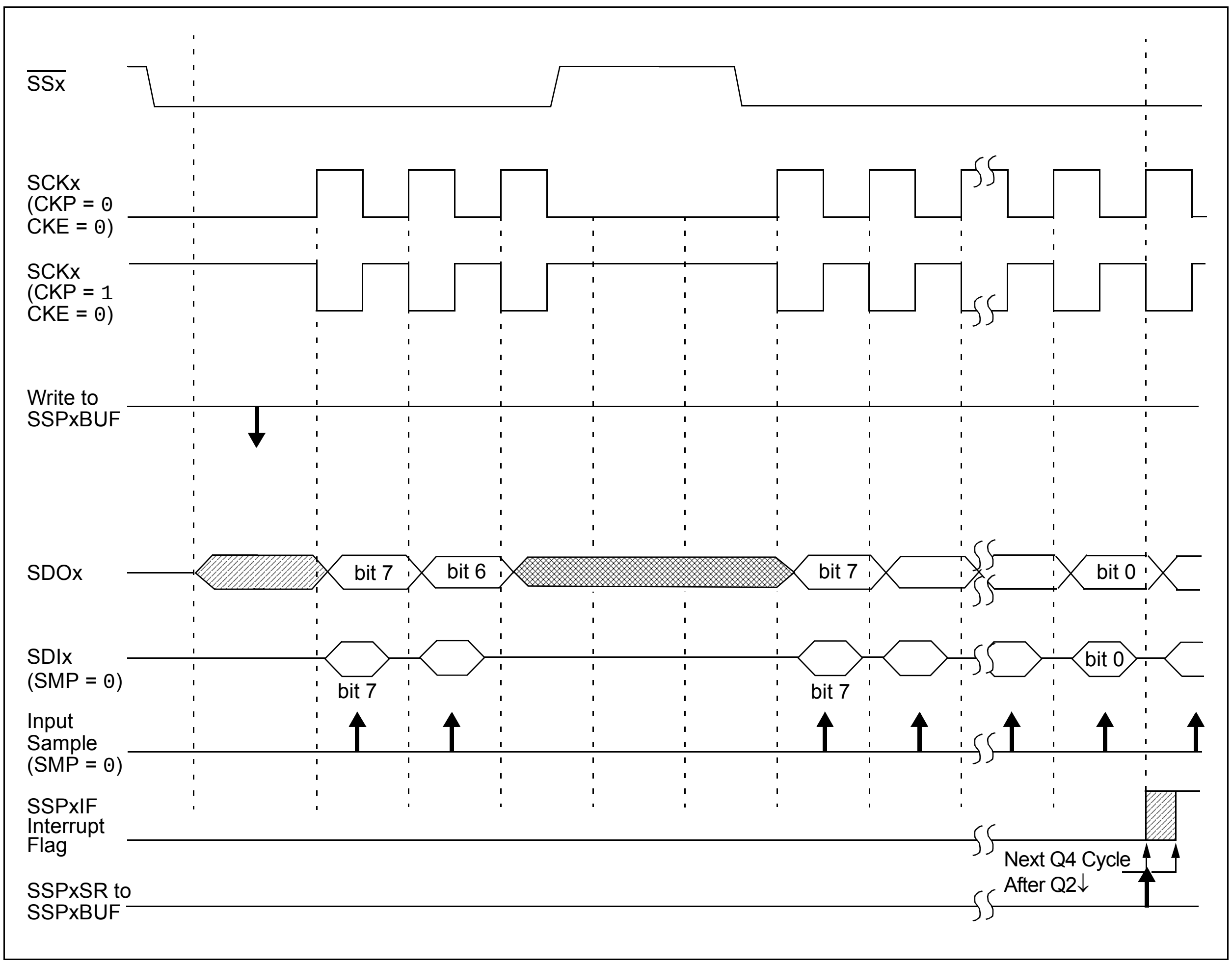


**FIGURE 20-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**

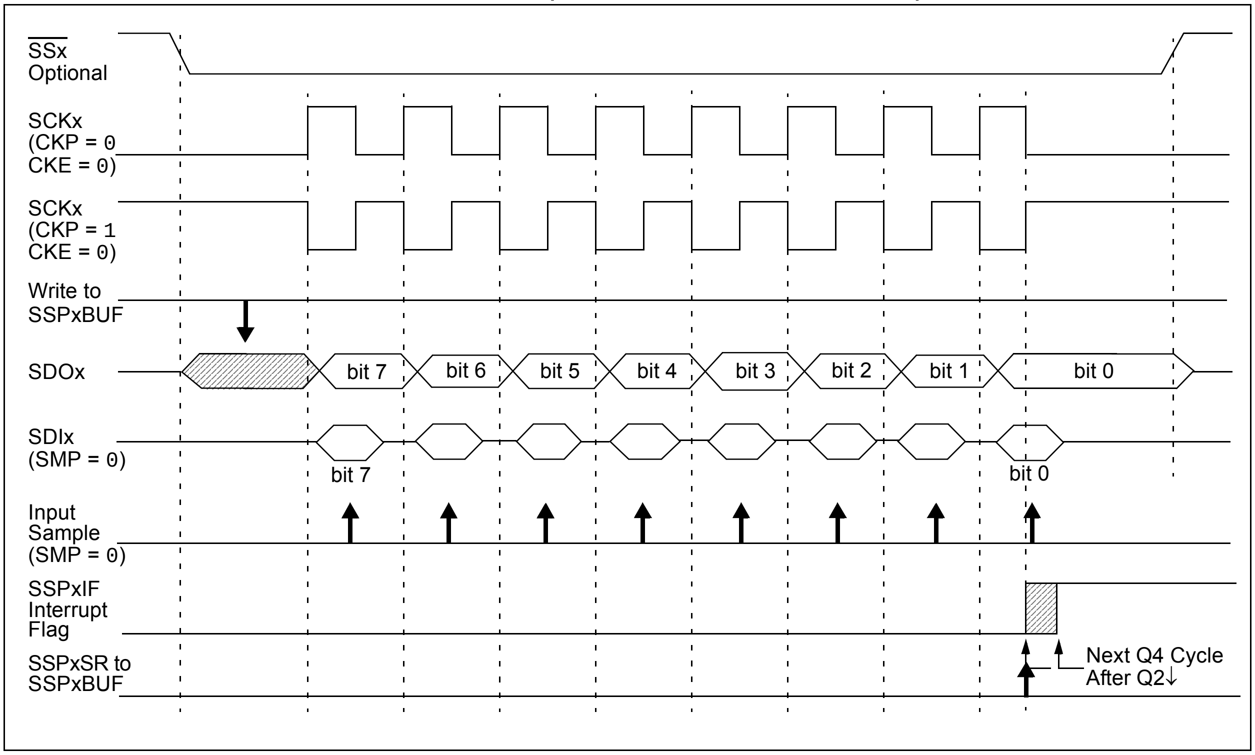
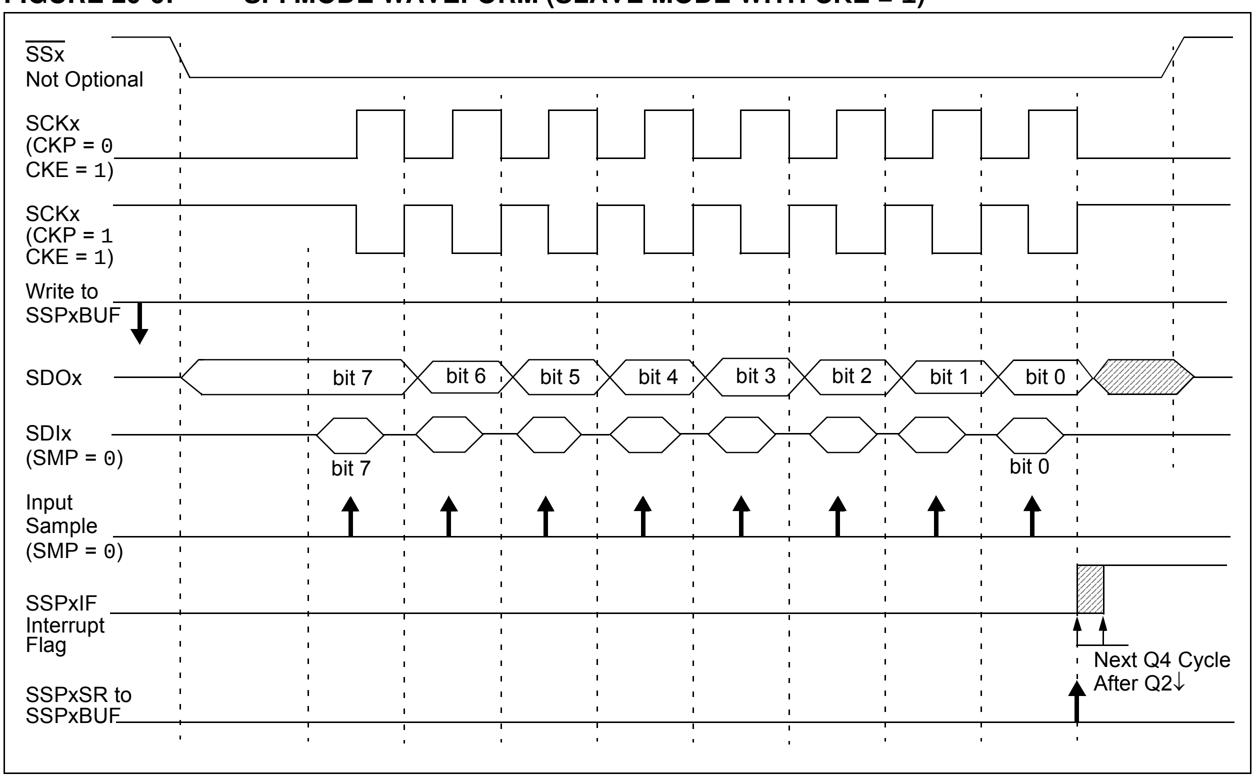



**FIGURE 20-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**



### 20.3.9 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 3.3 "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 MSSPx Interrupt Flag bit, SSPxIF, will be set and if enabled, will wake the device.

### 20.3.10 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

### 20.3.11 BUS MODE COMPATIBILITY

Table 20-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

**TABLE 20-1: SPI BUS MODES**

| Standard SPI Mode Terminology | Control Bits State | |
|---|---|---|
| | CKP | CKE |
| 0, 0 | 0 | 1 |
| 0, 1 | 0 | 0 |
| 1, 0 | 1 | 1 |
| 1, 1 | 1 | 0 |

There is also an SMP bit which controls when the data is sampled.

### 20.3.12 SPI CLOCK SPEED AND MODULE INTERACTIONS

Because MSSP1 and MSSP2 are independent modules, they can operate simultaneously at different data rates. Setting the SSPM<3:0> 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 20-2: REGISTERS ASSOCIATED WITH SPI OPERATION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 64 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 64 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | — | — | 64 |
| SSP1BUF | MSSP1 Receive Buffer/Transmit Register | | | | | | | | 62 |
| SSPxCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 62, 65 |
| SSPxSTAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 62, 65 |
| SSP2BUF | MSSP2 Receive Buffer/Transmit Register | | | | | | | | 65 |
| ODCON3[1] | — | — | — | — | — | — | SPI2OD | SPI1OD | 62 |

**Legend:** Shaded cells are not used by the MSSPx module in SPI mode.

**Note 1:** Configuration SFR, overlaps with the default SFR at this address; available only when WDTCON<4> = 1.

## 20.4 I²C Mode

| **Note:** | Disabling the MSSPx module by clearing the SSPEN (SSPxCON1<5>) bit may not reset the module. It is recommended to clear the SSPxSTAT, SSPxCON1 and SSPxCON2 registers, and select the mode prior to setting the SSPEN bit to enable the MSSPx module. |
|---|---|

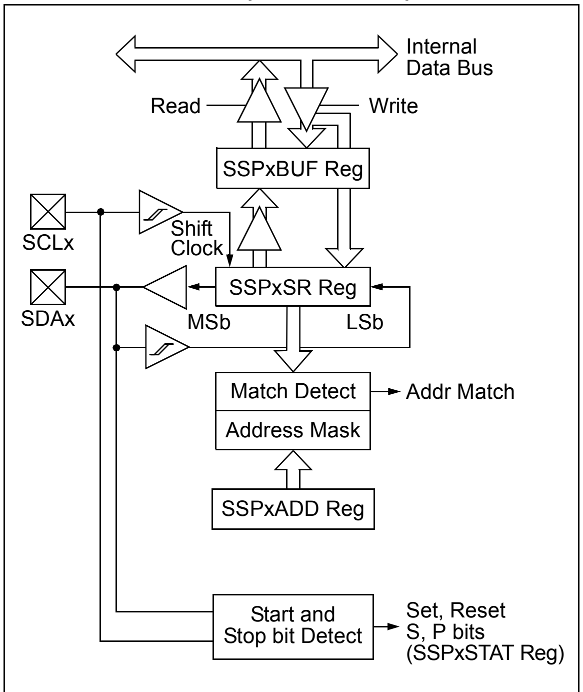
The MSSP module in I²C mode fully implements all master and slave functions (including general call support), and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit 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 20-7: MSSPx BLOCK DIAGRAM (I²C™ MODE)**




**Note:** Only port I/O names are used in this diagram for the sake of brevity. Refer to the text for a full list of multiplexed functions.

### 20.4.1 REGISTERS

The MSSPx module has six registers for I²C 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)
- I²C Slave Address Mask Register (SSPxMSK)

SSPxCON1, SSPxCON2 and SSPxSTAT are the control and status registers in I²C mode operation. The SSPxCON1 and SSPxCON2 registers are 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.

SSPxADD contains the slave device address when the MSSPx is configured in I²C Slave mode. When the MSSPx is configured in Master mode, SSPxADD acts 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 SSPM<3:0> bits are specifically set to permit access. Additional details are provided in **Section 20.4.3.4 "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 SSPxIF interrupt is set.

During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

### REGISTER 20-3: SSPxSTAT: MSSPx STATUS REGISTER (I²C™ MODE)

| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
|---|---|---|---|---|---|---|---|
| SMP | CKE | D/$\overline{\text{A}}$ | P[(1)] | S[(1)] | R/$\overline{\text{W}}$[(2,3)] | UA | BF |
| 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 **SMP:** Slew Rate Control bit
In Master or Slave mode:
`1` = Slew rate control is disabled for Standard Speed mode (100 kHz and 1 MHz)
`0` = Slew rate control is enabled for High-Speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit
In Master or Slave mode:
`1` = Enables SMBus-specific inputs
`0` = Disables SMBus-specific inputs

bit 5 **D/$\overline{\text{A}}$:** Data/$\overline{\text{Address}}$ bit
In Master mode:
Reserved.
In Slave mode:
`1` = Indicates that the last byte received or transmitted was data
`0` = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit[(1)]
`1` = Indicates that a Stop bit has been detected last
`0` = Stop bit was not detected last

bit 3 **S:** Start bit[(1)]
`1` = Indicates that a Start bit has been detected last
`0` = Start bit was not detected last

bit 2 **R/$\overline{\text{W}}$:** Read/$\overline{\text{Write}}$ Information bit[(2,3)]
In Slave mode:
`1` = Read
`0` = Write
In Master mode:
`1` = Transmit is in progress
`0` = Transmit is not in progress

bit 1 **UA:** Update Address bit (10-Bit Slave mode only)
`1` = Indicates that the user needs to update the address in the SSPxADD register
`0` = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit
In Transmit mode:
`1` = SSPxBUF is full
`0` = SSPxBUF is empty
In Receive mode:
`1` = SSPxBUF is full (does not include the $\overline{\text{ACK}}$ and Stop bits)
`0` = SSPxBUF is empty (does not include the $\overline{\text{ACK}}$ and Stop bits)

**Note 1:** This bit is cleared on Reset and when SSPEN is cleared.
**2:** This bit holds the R/$\overline{\text{W}}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{\text{ACK}}$ bit.
**3:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSPx is in Active mode.

**REGISTER 20-4: SSPxCON1: MSSPx CONTROL REGISTER 1 (I²C™ MODE)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| WCOL | SSPOV | SSPEN[(1)] | CKP | SSPM3[(2)] | SSPM2[(2)] | SSPM1[(2)] | SSPM0[(2)] |
| 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 **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.

bit 6 **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.

bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit[(1)]

1 = Enables the serial port and configures the SDAx and SCLx pins as the serial port pins

0 = Disables the serial port and configures these pins as I/O port pins

bit 4 **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.

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits[(2)]

1111 = I²C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

1110 = I²C Slave mode, 7-bit address with Start and Stop bit interrupts enabled

1011 = I²C Firmware Controlled Master mode (Slave is Idle)

1001 = Loads the SSPxMSK register at the SSPxADD SFR address[(3,4)]

1000 = I²C Master mode, Clock = FOSC/(4 * (SSPxADD + 1))

0111 = I²C Slave mode, 10-bit address

0110 = I²C Slave mode, 7-bit address

**Note 1:** When enabled, the SDAx and SCLx pins must be configured as inputs.

**2:** Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

**3:** When SSPM<3:0> = 1001, any reads or writes to the SSPxADD SFR address actually accesses the SSPxMSK register.

**4:** This mode is only available when 7-Bit Address Masking mode is selected (MSSPMSK Configuration bit is '1').

## REGISTER 20-5: SSPxCON2: MSSPx CONTROL REGISTER 2 ($I^2C$™ MASTER MODE)

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| GCEN | ACKSTAT | ACKDT(1) | ACKEN(2) | RCEN(2) | PEN(2) | RSEN(2) | SEN(2) |
| 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 **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)
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit(2)
1 = Initiates Acknowledge sequence on SDAx and SCLx pins and transmits ACKDT data bit; automatically cleared by hardware
0 = Acknowledge sequence is Idle

bit 3 **RCEN:** Receive Enable bit (Master Receive mode only)(2)
1 = Enables Receive mode for $I^2C$
0 = Receive is Idle

bit 2 **PEN:** Stop Condition Enable bit(2)
1 = Initiates a Stop condition on the SDAx and SCLx pins; automatically cleared by hardware
0 = Stop condition is Idle

bit 1 **RSEN:** Repeated Start Condition Enable bit(2)
1 = Initiates Repeated Start condition on the SDAx and SCLx pins; automatically cleared by hardware
0 = Repeated Start condition is Idle

bit 0 **SEN:** Start Condition Enable bit(2)
1 = Initiates Start condition on the SDAx and SCLx pins; automatically cleared by hardware
0 = Start condition is Idle

**Note 1:** The value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.
**2:** If the $I^2C$ module is active, these bits may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).

**REGISTER 20-6: SSPxCON2: MSSPx CONTROL REGISTER 2 ($I^2C$™ SLAVE MODE)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| GCEN | ACKSTAT | ADMSK5 | ADMSK4 | ADMSK3 | ADMSK2 | ADMSK1 | SEN(1) |
| 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 **GCEN:** General Call Enable bit
1 = Enables interrupt when a general call address (0000h) is received in the SSPxSR
0 = General call address is disabled

bit 6 **ACKSTAT:** Acknowledge Status bit
Unused in Slave mode.

bit 5-2 **ADMSK5:ADMSK2:** Slave Address Mask Select bits (5-Bit Address Masking mode)
1 = Masking of the corresponding bits of SSPxADD is enabled
0 = Masking of the corresponding bits of SSPxADD is disabled

bit 1 **ADMSK1:** Slave Address Least Significant bit(s) Mask Select bit
In 7-Bit Addressing mode:
1 = Masking of SSPxADD<1> only is enabled
0 = Masking of SSPxADD<1> only is disabled
In 10-Bit Addressing mode:
1 = Masking of SSPxADD<1:0> is enabled
0 = Masking of SSPxADD<1:0> is disabled

bit 0 **SEN:** Stretch Enable bit(1)
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is disabled

**Note 1:** If the $I^2C$ module is active, this bit may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).

**REGISTER 20-7: SSPxMSK: MSSPx $I^2C$™ SLAVE ADDRESS MASK REGISTER (7-BIT MASKING MODE)(1)**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0(2) |
| 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 **MSK<7:0>:** Slave Address Mask Select bit(2)
1 = Masking of the corresponding bit of SSPxADD is enabled
0 = Masking of the corresponding bit of SSPxADD is disabled

**Note 1:** This register shares the same SFR address as SSPxADD and is only addressable in select MSSPx operating modes. See **Section 20.4.3.4 “7-Bit Address Masking Mode”** for more details.
**2:** MSK0 is not used as a mask bit in 7-bit addressing.

### 20.4.2 OPERATION

The MSSP module functions are enabled by setting the MSSPx Enable bit, SSPEN (SSPxCON1<5>).

The SSPxCON1 register allows control of the I$^2$C operation. Four mode selection bits (SSPxCON1<3:0>) allow one of the following I$^2$C modes to be selected:

- I$^2$C Master mode, clock
- I$^2$C Slave mode (7-bit address)
- I$^2$C Slave mode (10-bit address)
- I$^2$C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I$^2$C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I$^2$C Firmware Controlled Master mode, slave is Idle

Selection of any I$^2$C 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.

### 20.4.3 SLAVE MODE

In Slave mode, the SCLx and SDAx pins must be configured as inputs (TRISC<4:3> set). The MSSPx module will override the input state with the output data when required (slave-transmitter).

The I$^2$C 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 ($\overline{ACK}$) pulse and load the SSPxBUF register with the received value currently in the SSPxSR register.

Any combination of the following conditions will cause the MSSPx module not to give this $\overline{ACK}$ 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 SSPxIF 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 I$^2$C specification, as well as the requirement of the MSSPx module, are shown in timing Parameter 100 and Parameter 101.

#### 20.4.3.1 Addressing

Once the MSSPx 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 $\overline{ACK}$ pulse is generated.
4. The MSSPx 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/$\overline{W}$ (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, SSPxIF, BF and UA, are set on an address match).
2. Update the SSPxADD register with the second (low) byte of the address (clears bit, UA, and releases the SCLx line).
3. Read the SSPxBUF register (clears bit, BF) and clear flag bit, SSPxIF.
4. Receive second (low) byte of address (bits, SSPxIF, BF and UA, are set).
5. Update the SSPxADD register with the first (high) byte of the address. If the match releases the SCLx line, this will clear bit, UA.
6. Read the SSPxBUF register (clears bit, BF) and clear flag bit, SSPxIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of address (bits, SSPxIF and BF, are set).
9. Read the SSPxBUF register (clears bit, BF) and clear flag bit, SSPxIF.

#### 20.4.3.2 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 I$^2$C Slave behaves the same way whether address masking is used or not. However, when address masking is used, the I$^2$C 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 I$^2$C 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.

#### 20.4.3.3 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 20-2). 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 I$^2$C Slave mode (Register 20-6). 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 20-2: ADDRESS MASKING EXAMPLES IN 5-BIT MASKING MODE**

**7-Bit Addressing:**

SSPxADD<7:1> = A0h (1010000) (SSPxADD<0> is assumed to be '0')

ADMSK<5:1> = 00111

Addresses Acknowledged: A0h, A2h, A4h, A6h, A8h, AAh, ACh, AEh

**10-Bit Addressing:**

SSPxADD<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

#### 20.4.3.4 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 20-3). 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' (SSPxCON1<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 $I^2C$ 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 SSPxADD register address (FC8h for MSSP1, F6Eh for MSSP2).
3. Set the appropriate $I^2C$ 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<n> = 0), the corresponding SSPxADD 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.

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<n> = x).

> **Note:** The two Most Significant bits of the address are not affected by address masking.

**EXAMPLE 20-3: ADDRESS MASKING EXAMPLES IN 7-BIT MASKING MODE**

**7-Bit Addressing:**

SSPxADD<7:1> = 1010 000

SSPxMSK<7:1> = 1111 001

Addresses Acknowledged: ACh, A8h, A4h, A0h

**10-Bit Addressing:**

SSPxADD<7:0> = 1010 0000 (The two MSb are ignored in this example since they are not affected)

SSPxMSK<7:0> = 1111 0011

Addresses Acknowledged: ACh, A8h, A4h, A0h

#### 20.4.3.5 Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPxSTAT register is cleared. The received address is loaded into the SSPxBUF register and the SDAx line is held low ($\overline{ACK}$).

When the address byte overflow condition exists, then the no Acknowledge ($\overline{ACK}$) 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 20.4.4 "Clock Stretching"** for more details.

#### 20.4.3.6 Transmission

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPxSTAT register is set. The received address is loaded into the SSPxBUF register. The $\overline{ACK}$ pulse will be sent on the ninth bit and pin, SCLx, is held low regardless of SEN (see **Section 20.4.4 "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, the SCLx pin 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 20-10).

The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCLx input pulse. If the SDAx line is high (not $\overline{ACK}$), then the data transfer is complete. In this case, when the $\overline{ACK}$ 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 ($\overline{ACK}$), 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 20-8: I²C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)**

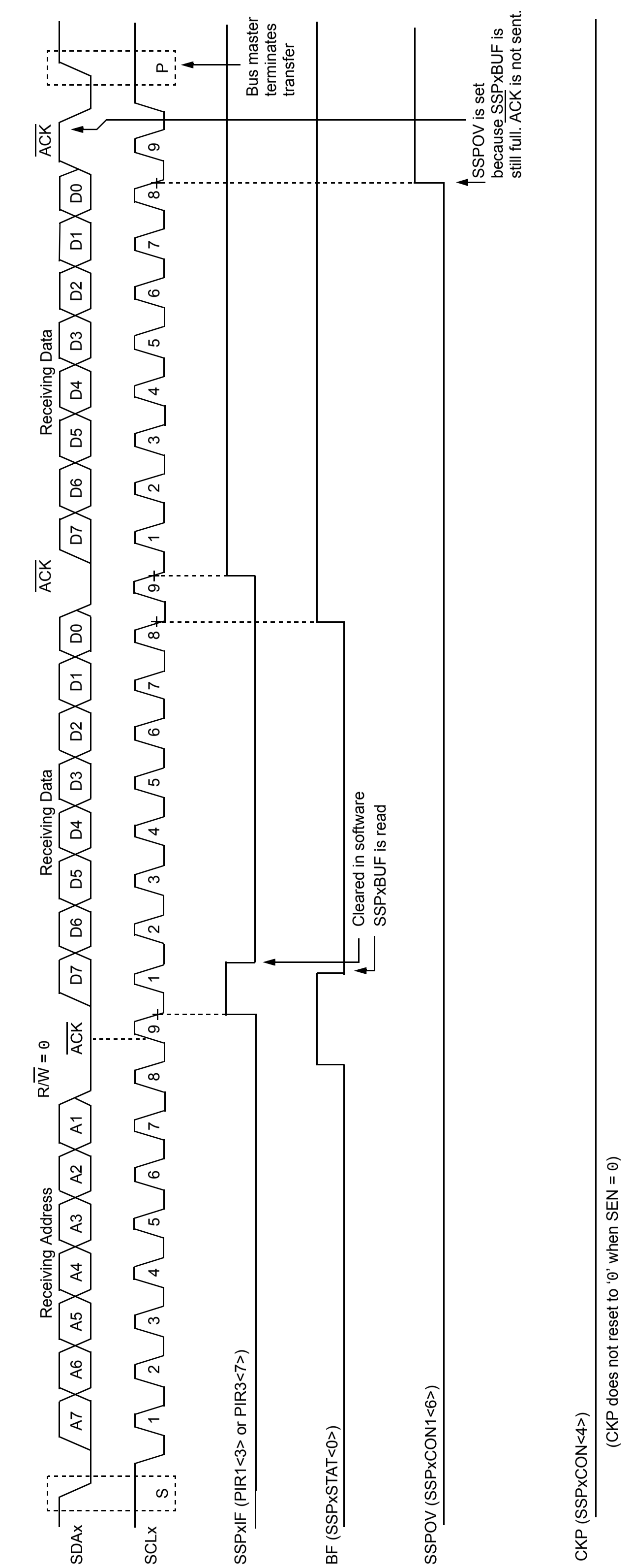
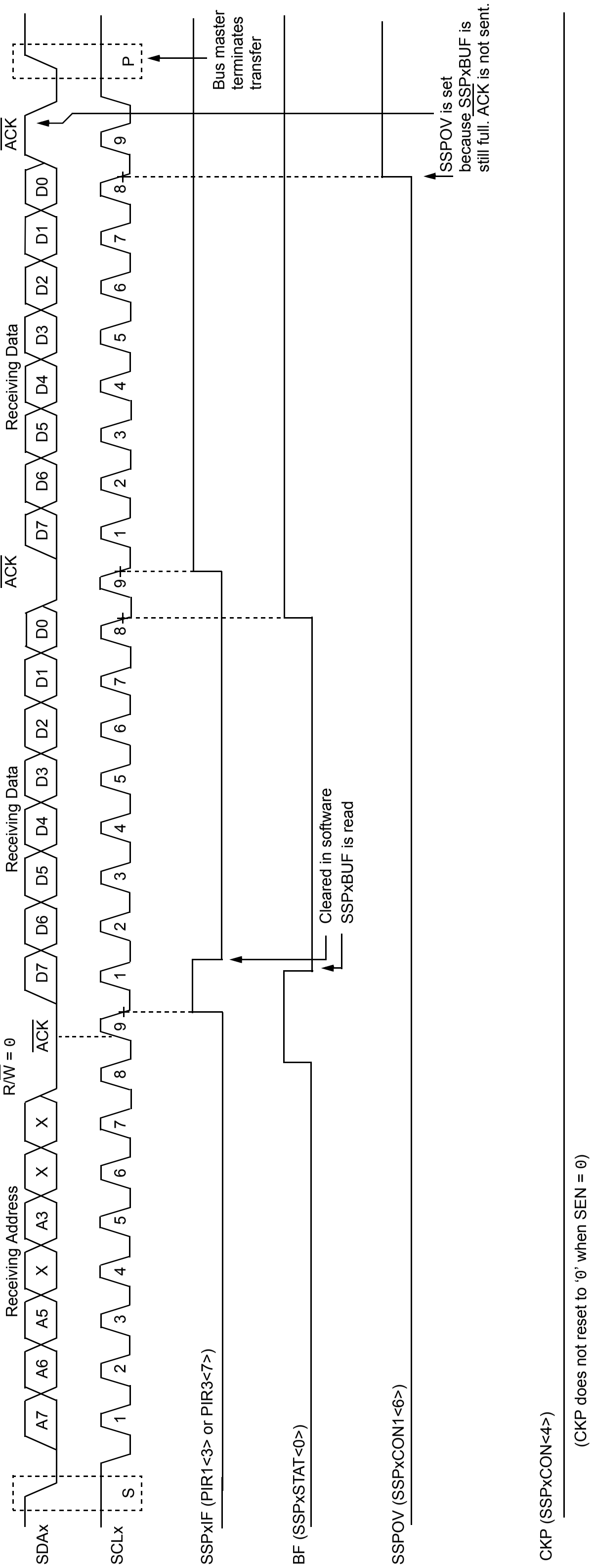


**FIGURE 20-9: I²C™ SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01011 (RECEPTION, 7-BIT ADDRESS)**




**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 20-10: I²C™ SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)**

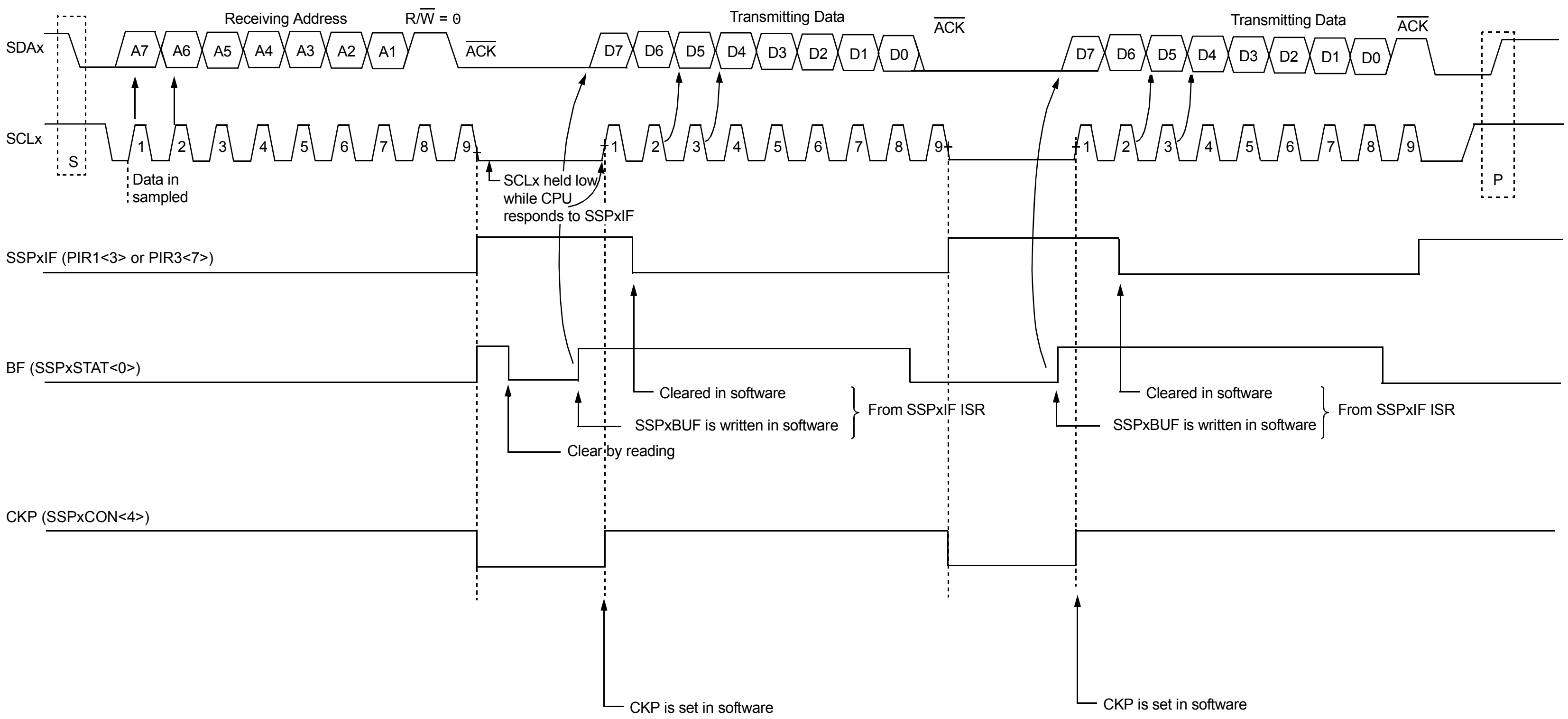
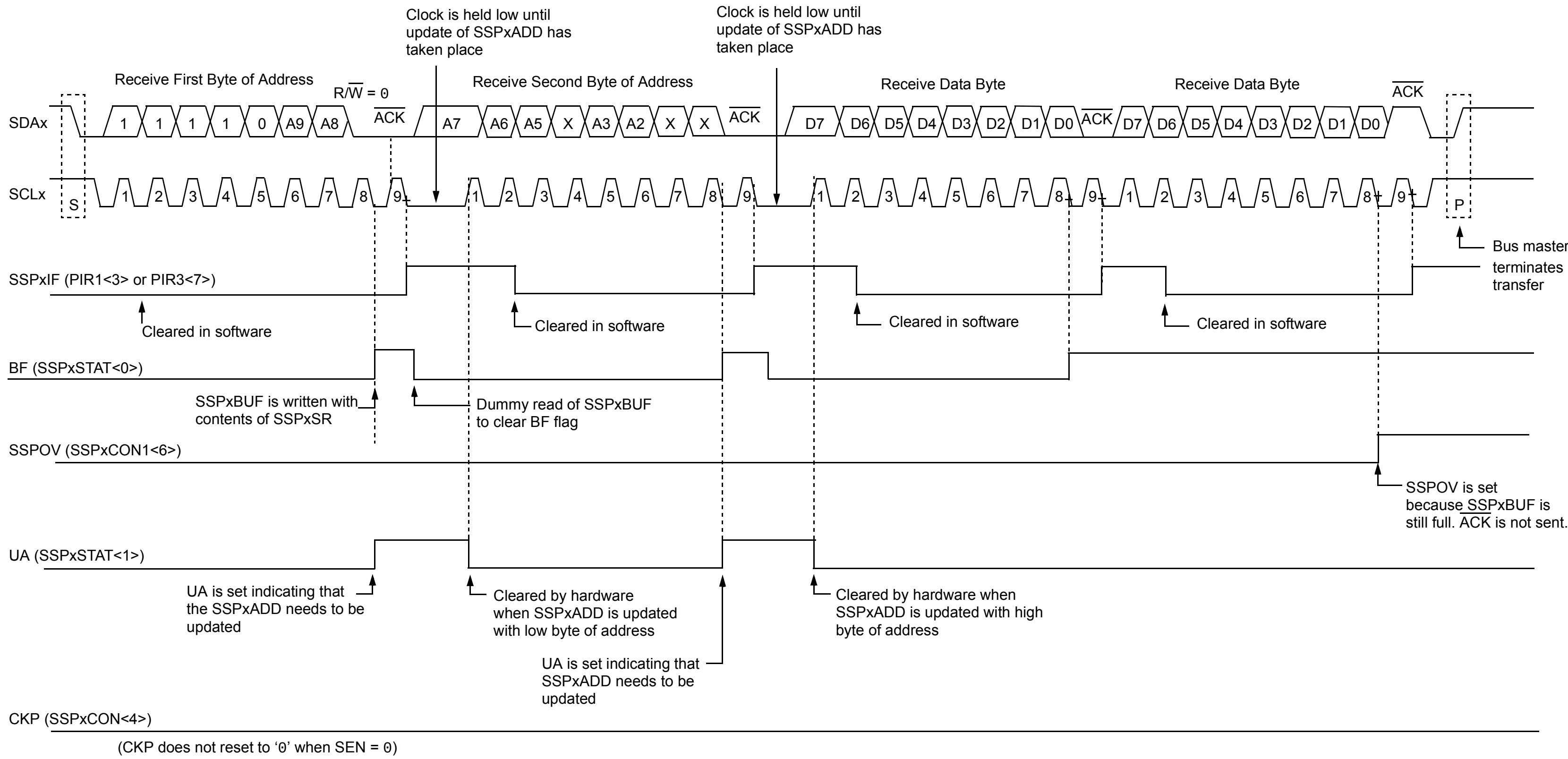


**FIGURE 20-11: I²C™ SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01001 (RECEPTION, 10-BIT ADDRESS)**




**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 A9.A8.A7.A6.A5.X.A3.A2.X.X will be Acknowledged and cause an interrupt.

**3:** Note that the Most Significant bits of the address are not affected by the bit masking.

FIGURE 20-12: I²C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)

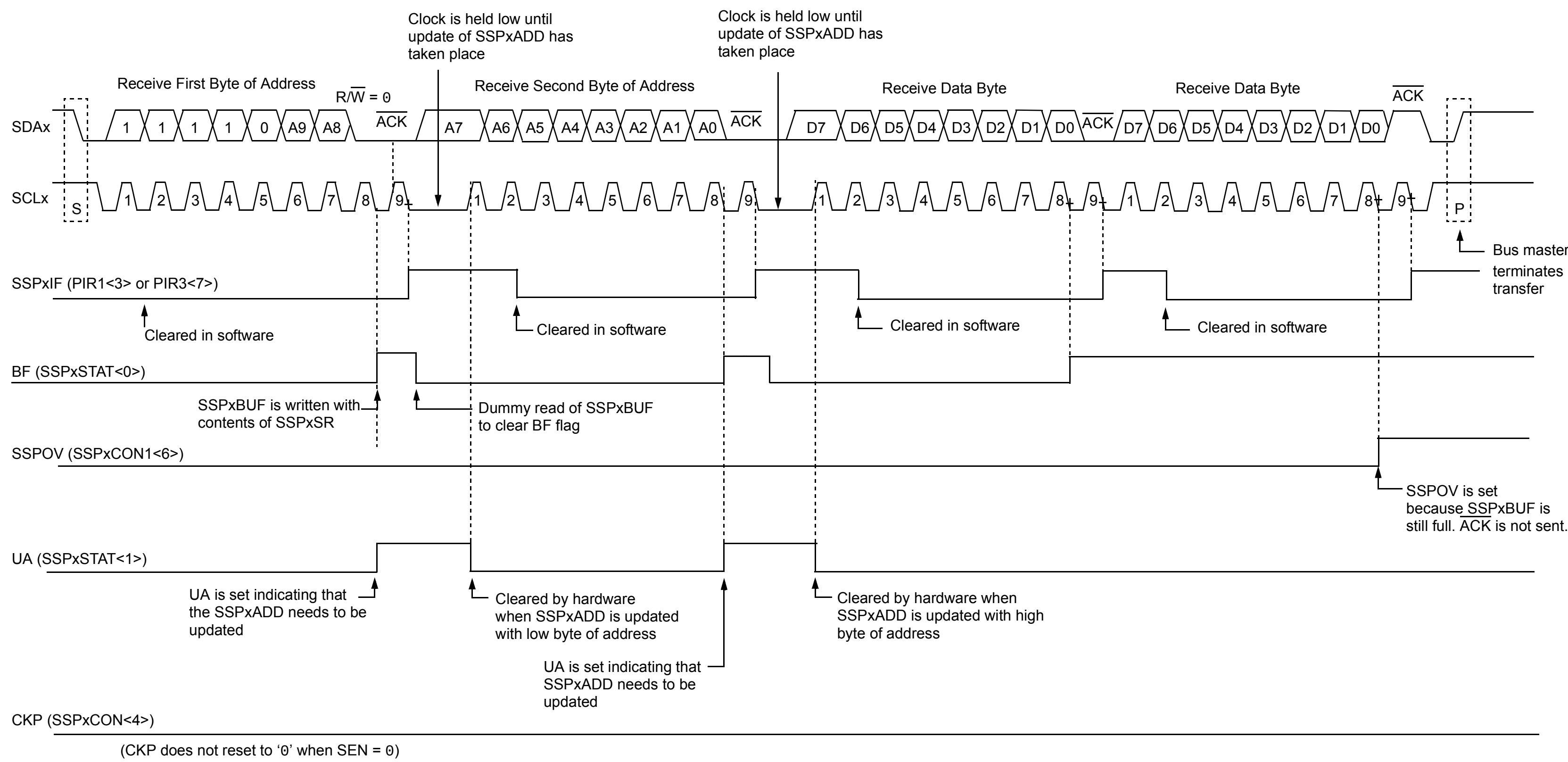
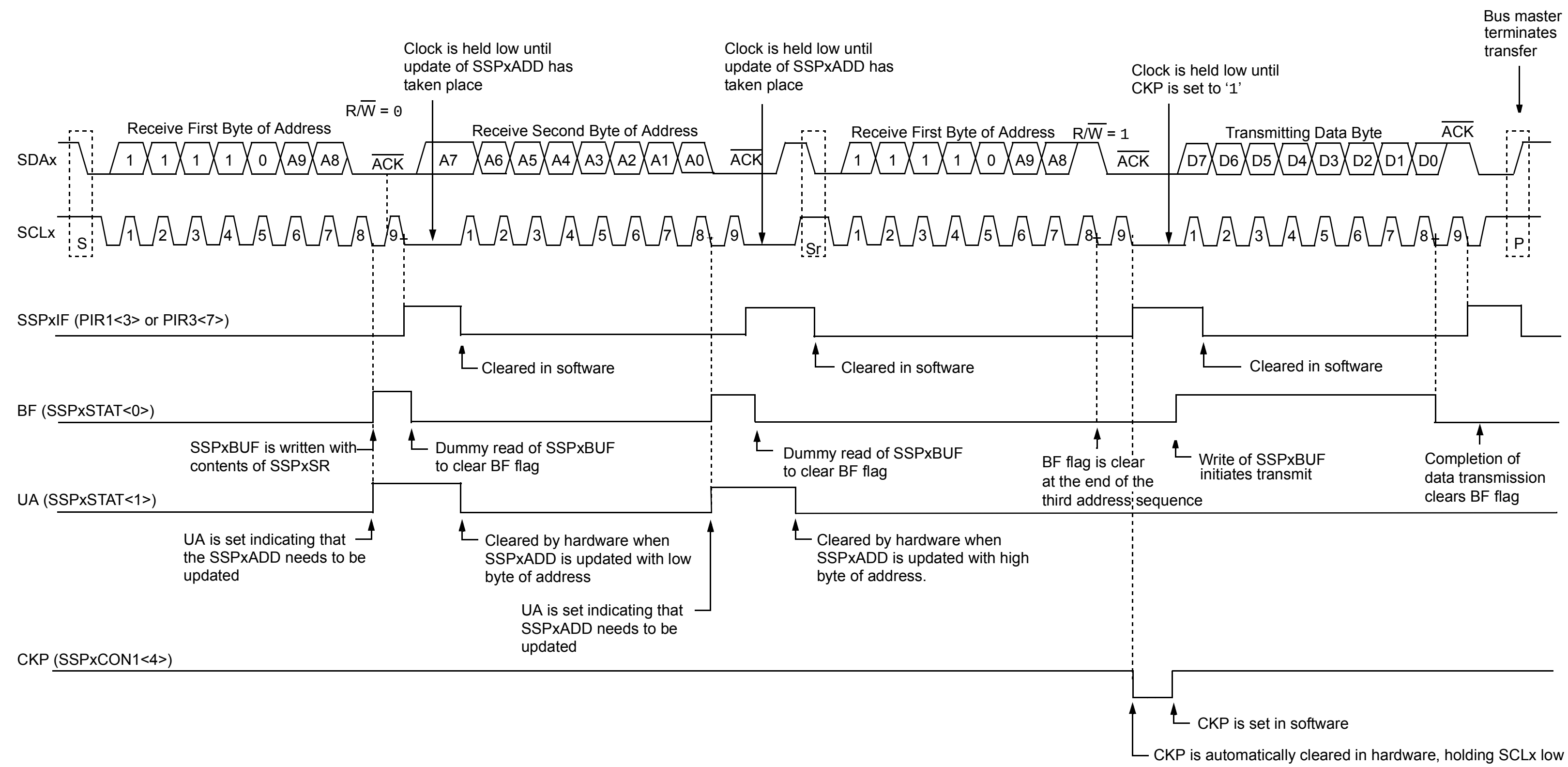


**FIGURE 20-13: I$^2$C™ SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)**



### 20.4.4 CLOCK STRETCHING

Both 7-Bit and 10-Bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPxCON2<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.

#### 20.4.4.1 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 $\overline{\text{ACK}}$ 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 20-15).

> **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.

#### 20.4.4.2 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/$\overline{\text{W}}$ 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.

#### 20.4.4.3 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 20-10).

> **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.

#### 20.4.4.4 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/$\overline{\text{W}}$ 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 20-13).

#### 20.4.4.5 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 $I^2C$ master device has already asserted the SCLx line. The SCLx output will remain low until the CKP bit is set and all other devices on the $I^2C$ bus have deasserted SCLx. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCLx (see Figure 20-14).

**FIGURE 20-14: CLOCK SYNCHRONIZATION TIMING**

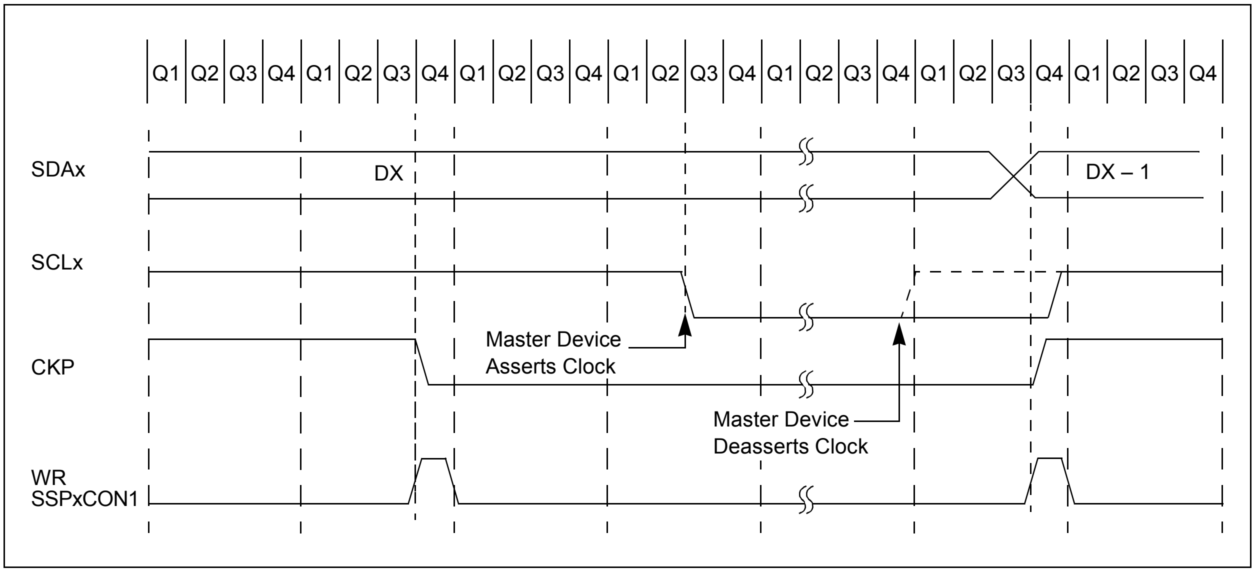


**FIGURE 20-15: I$^2$C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)**

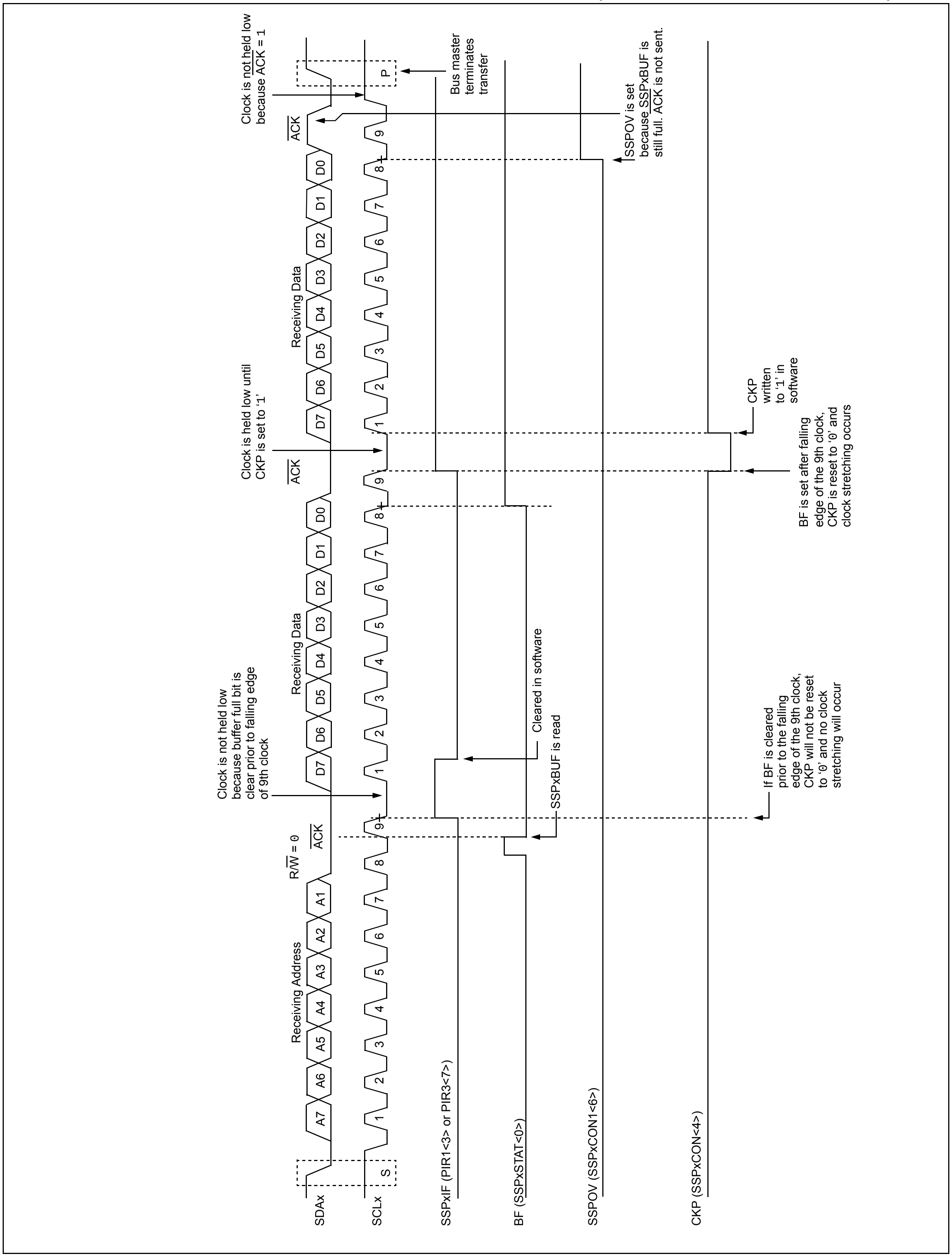
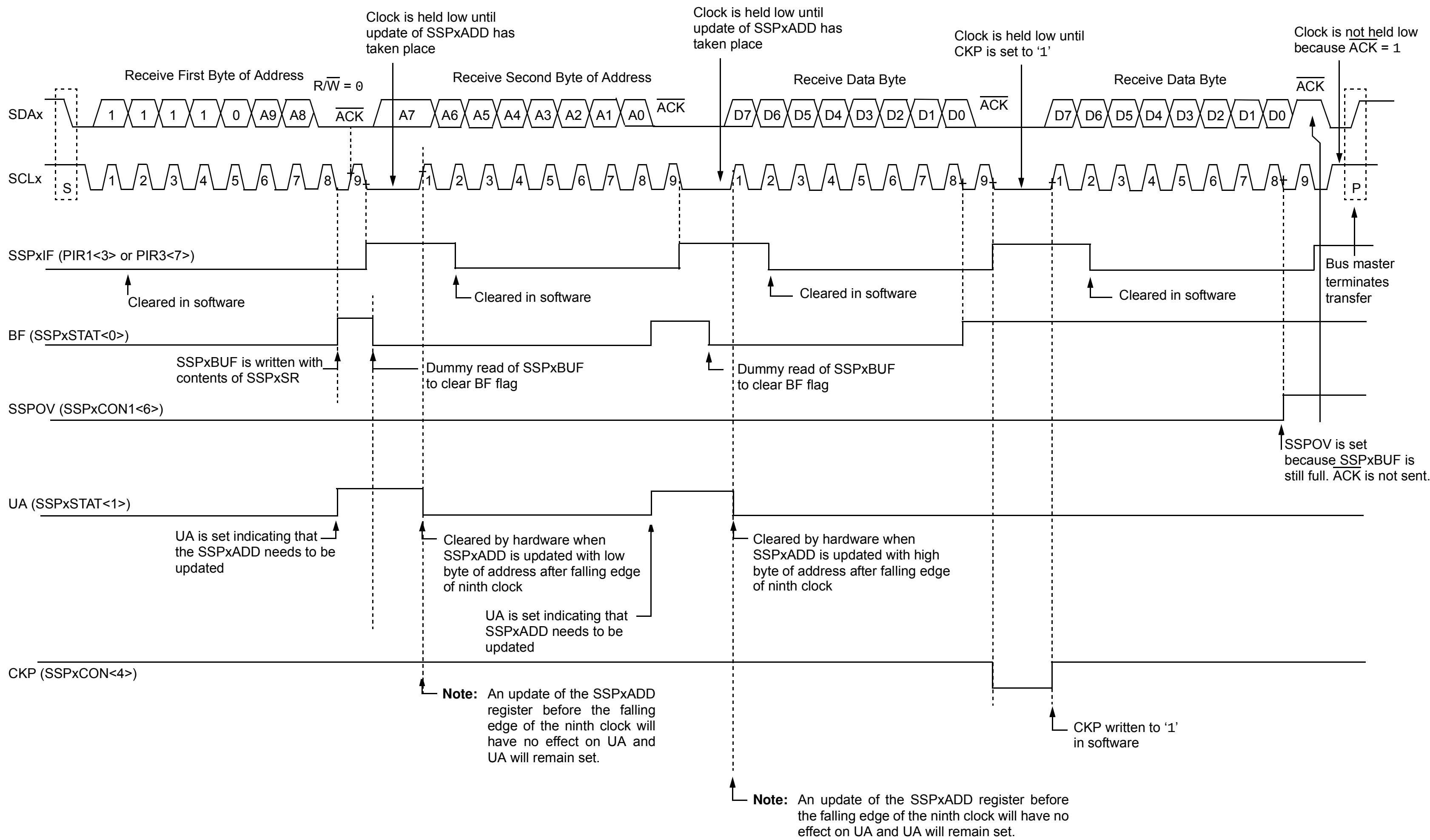


**FIGURE 20-16: I²C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESS)**



### 20.4.5 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/$\overline{W}$ = 0.

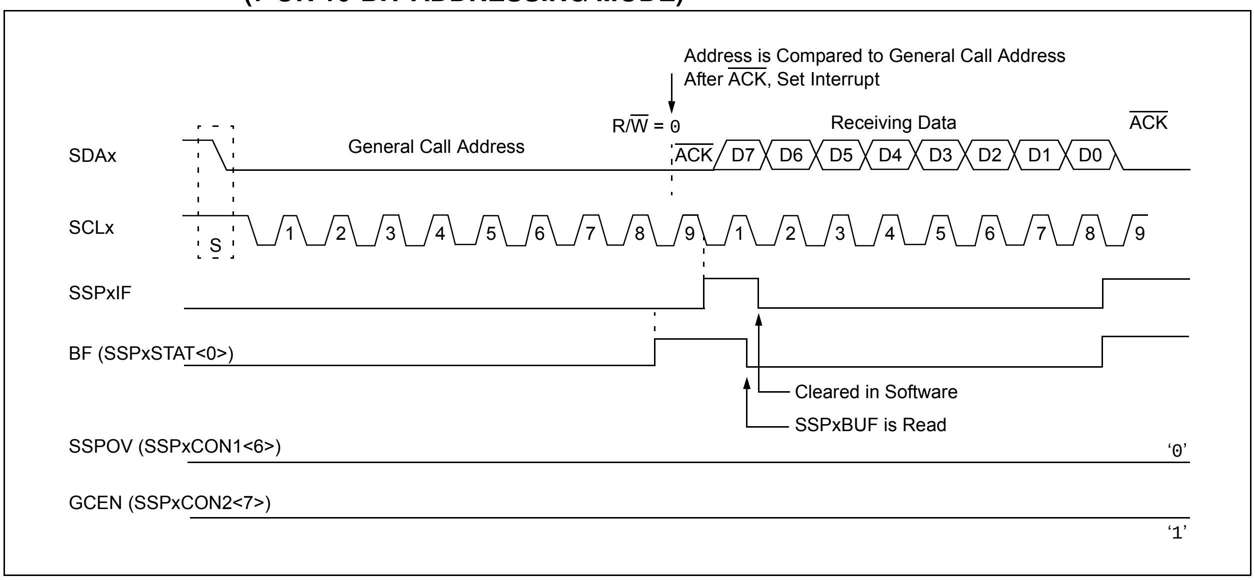
The general call address is recognized when the General Call Enable bit, GCEN, is enabled (SSPxCON2<7> set). Following a Start bit detect, 8 bits are shifted into the SSPxSR and the address is compared against the SSPxADD. It is also compared to the general call address and fixed in hardware.

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 ($\overline{ACK}$ 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 20-17).

**FIGURE 20-17: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESSING MODE)**



### 20.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPMx 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 I$^2$C 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 I$^2$C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

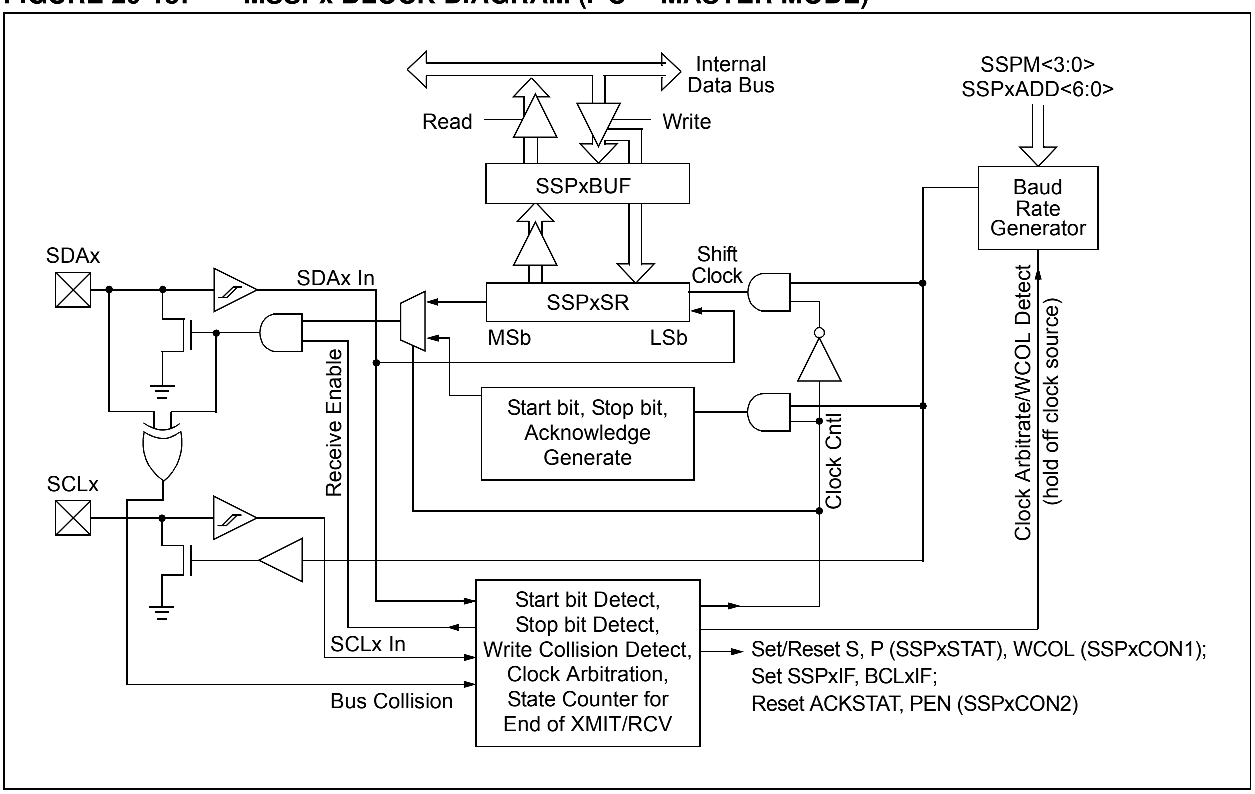
1. Assert a Start condition on SDAx and SCLx.
2. Assert a Repeated Start condition on SDAx and SCLx.
3. Write to the SSPxBUF register, initiating transmission of data/address.
4. Configure the I$^2$C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDAx and SCLx.

| **Note:** | The MSSPx module, when configured in I$^2$C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur. |
|---|---|

The following events will cause the MSSPx 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 20-18: MSSPx BLOCK DIAGRAM (I$^2$C™ MASTER MODE)**



#### 20.4.6.1 I²C Master Mode Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDAx while SCLx outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/$\overline{\text{Write}}$ ($R/\overline{W}$) bit. In this case, the $R/\overline{W}$ bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the $R/\overline{W}$ bit. In this case, the $R/\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address, followed by a '1' to indicate the receive bit. Serial data is received via SDAx, while SCLx outputs the serial clock. Serial data is received 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 I²C operation. See **Section 20.4.7 "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 MSSPx 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 of the SDAx pin until all 8 bits are transmitted.
5. The MSSPx module shifts in the ACK bit from the slave device and writes its value into the SSPxCON2 register (SSPxCON2<6>).
6. The MSSPx 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 of the SDAx pin until all 8 bits are transmitted.
9. The MSSPx module shifts in the ACK bit from the slave device and writes its value into the SSPxCON2 register (SSPxCON2<6>).
10. The MSSPx 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.

### 20.4.7 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 20-19). 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 20-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD.

#### 20.4.7.1 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 20-19: BAUD RATE GENERATOR BLOCK DIAGRAM**

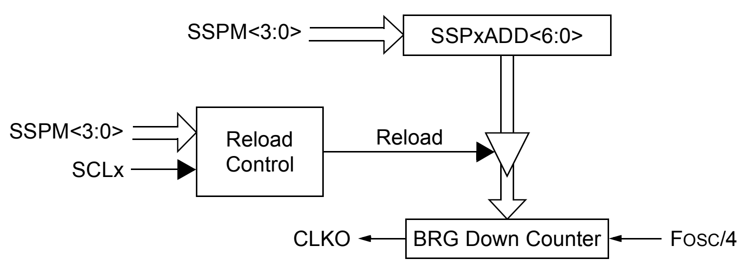



**TABLE 20-3: I²C™ CLOCK RATE w/BRG**

| FOSC | FCY | FCY * 2 | BRG Value | FSCL (2 Rollovers of BRG) |
|---|---|---|---|---|
| 40 MHz | 10 MHz | 20 MHz | 18h | 400 kHz(1) |
| 40 MHz | 10 MHz | 20 MHz | 1Fh | 312.5 kHz |
| 40 MHz | 10 MHz | 20 MHz | 63h | 100 kHz |
| 16 MHz | 4 MHz | 8 MHz | 09h | 400 kHz(1) |
| 16 MHz | 4 MHz | 8 MHz | 0Ch | 308 kHz |
| 16 MHz | 4 MHz | 8 MHz | 27h | 100 kHz |
| 4 MHz | 1 MHz | 2 MHz | 02h | 333 kHz(1) |
| 4 MHz | 1 MHz | 2 MHz | 09h | 100 kHz |
| 16 MHz | 4 MHz | 8 MHz | 03h | 1 MHz(1,2) |

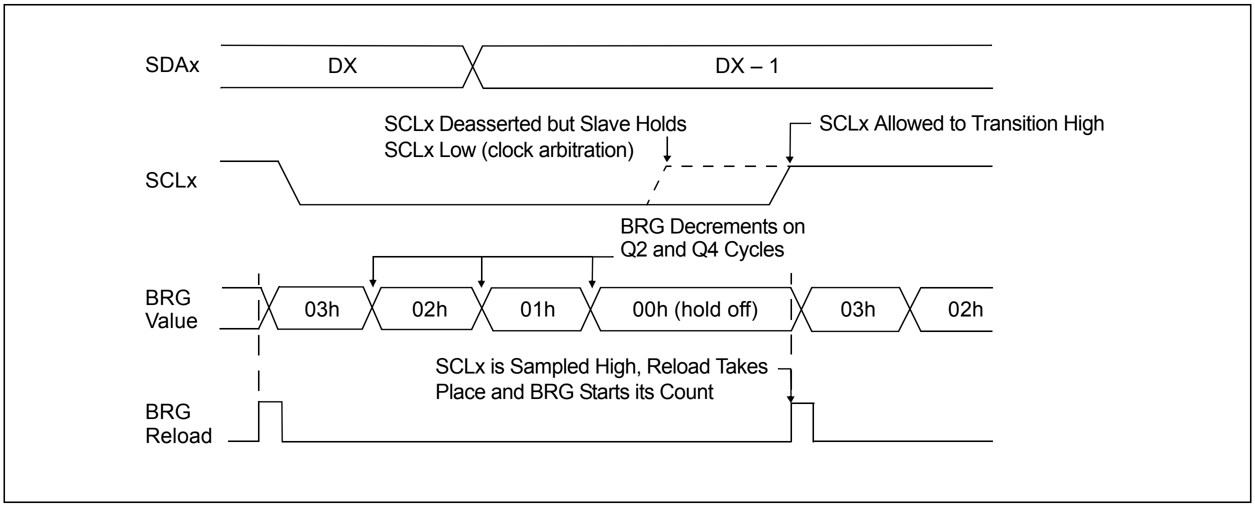
**Note 1:** The I²C interface does not conform to the 400 kHz I²C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.

**2:** A minimum 16 MHz FOSC is required for the 1 MHz I²C.

### 20.4.7.2 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts the SCLx pin (SCLx is 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 20-20).

**FIGURE 20-20: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**



### 20.4.8 I$^2$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.

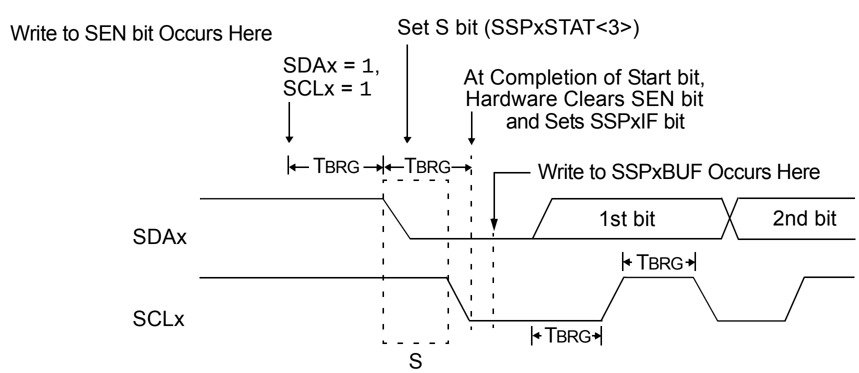
| **Note:** | If, at the beginning of the Start condition, the SDAx and SCLx pins are already sampled low or if during the Start condition, the SCLx line is sampled low before the SDAx line is driven low, a bus collision occurs; the Bus Collision Interrupt Flag, BCLxIF, is set, the Start condition is aborted and the I$^2$C module is reset into its Idle state. |
|---|---|

#### 20.4.8.1 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 20-21: FIRST START BIT TIMING**



### 20.4.9 I$^2$C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPxCON2<1>) is programmed high and the I$^2$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<6:0> 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:<br>• SDAx is sampled low when SCLx goes from low-to-high.<br>• SCLx goes low before SDAx is asserted low. This may indicate that another master is attempting to transmit a data ‘1’. |

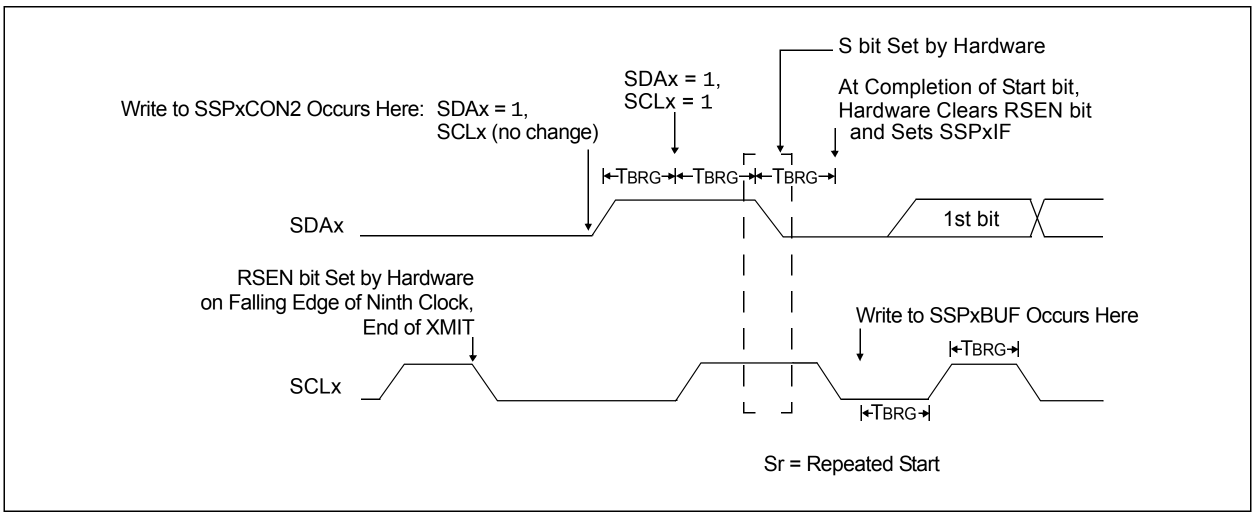
Immediately following the SSPxIF bit getting set, the user may write the SSPxBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

#### 20.4.9.1 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 20-22: REPEATED START CONDITION WAVEFORM**



### 20.4.10 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address, is accomplished by simply writing a value to the SSPxBUF register. This action will set the Buffer Full flag bit, BF, and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDAx pin after the falling edge of SCLx is asserted (see data hold time specification Parameter 106). SCLx is held low for one Baud Rate Generator rollover count (TBRG). 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 $\overline{ACK}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{ACK}$ is written into the 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 20-23).

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/$\overline{W}$ 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 $\overline{ACK}$ 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.

#### 20.4.10.1 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.

#### 20.4.10.2 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.

#### 20.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPxCON2<6>) is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the slave does not Acknowledge ($\overline{ACK}$ = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

### 20.4.11 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPxCON2<3>).

| **Note:** | The MSSPx module must be in an inactive state before the RCEN bit is set or the RCEN bit will be disregarded. |
|---|---|

The Baud Rate Generator begins counting and on each rollover, the state of the SCLx pin changes (high-to-low/low-to-high) and data is shifted into the SSPxSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPxSR are loaded into the SSPxBUF, the BF flag bit is set, the SSPxIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCLx low. The MSSPx is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN (SSPxCON2<4>).

#### 20.4.11.1 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.

#### 20.4.11.2 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.

#### 20.4.11.3 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 20-23: I²C™ MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)**

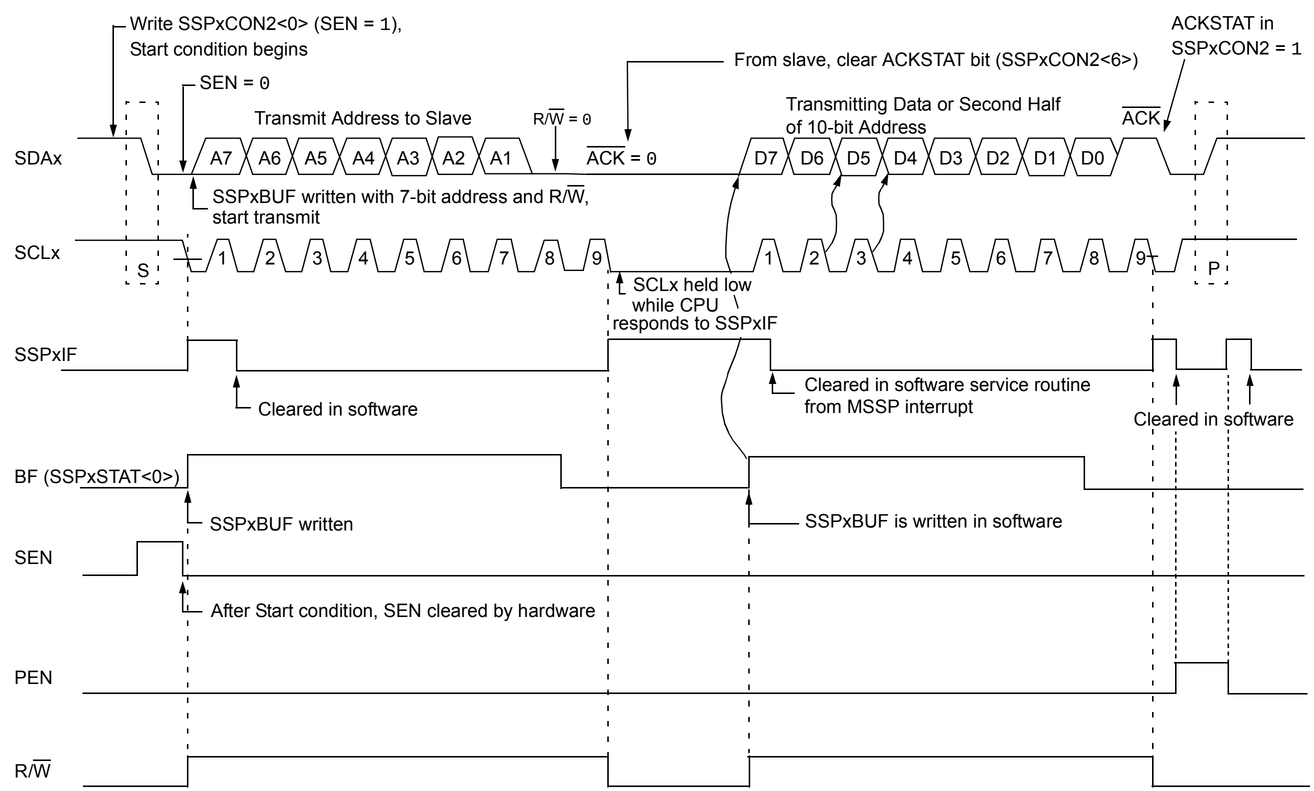
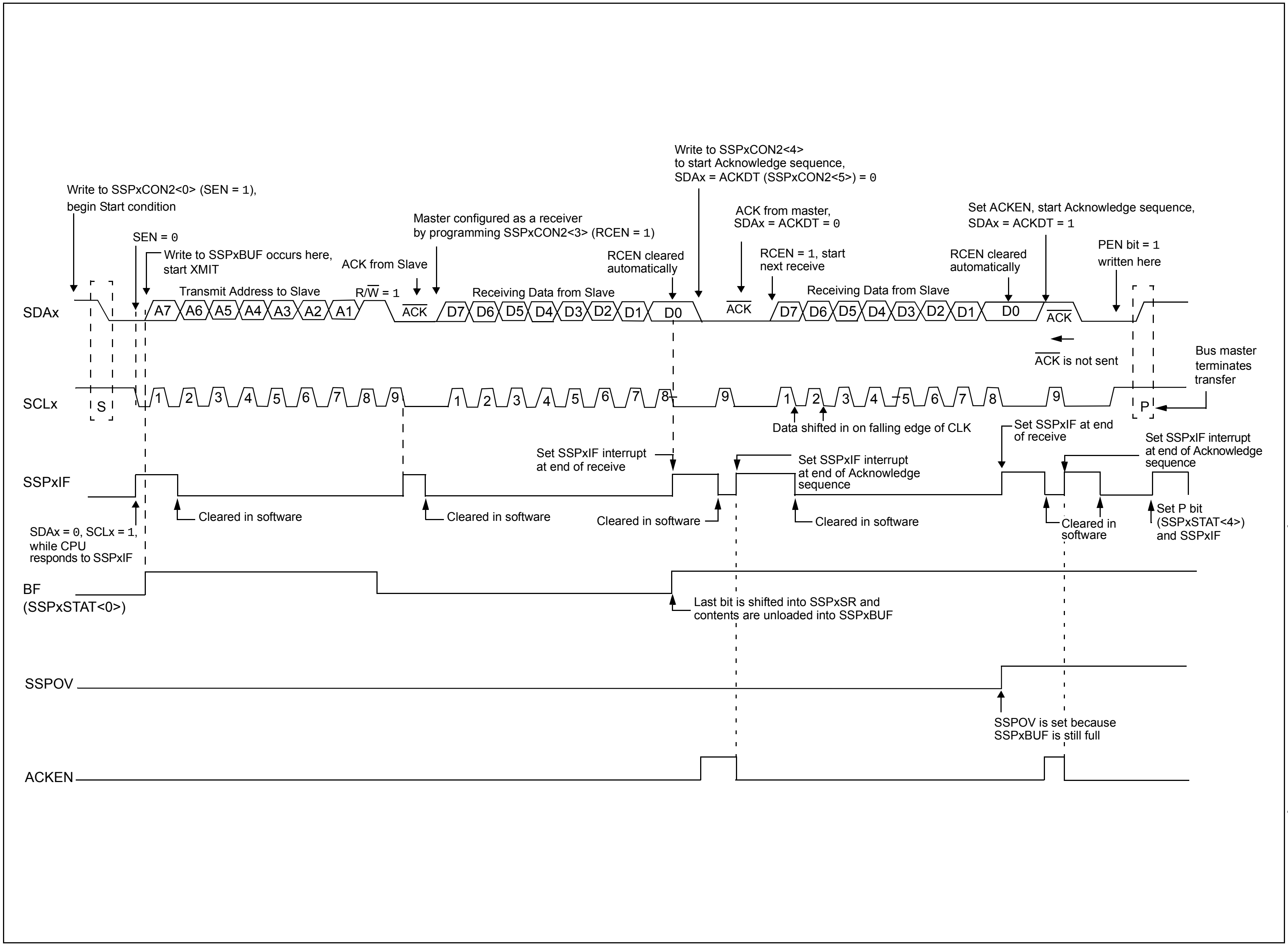


FIGURE 20-24: I²C™ MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



### 20.4.12 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 MSSPx module then goes into an inactive state (Figure 20-25).

#### 20.4.12.1 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).

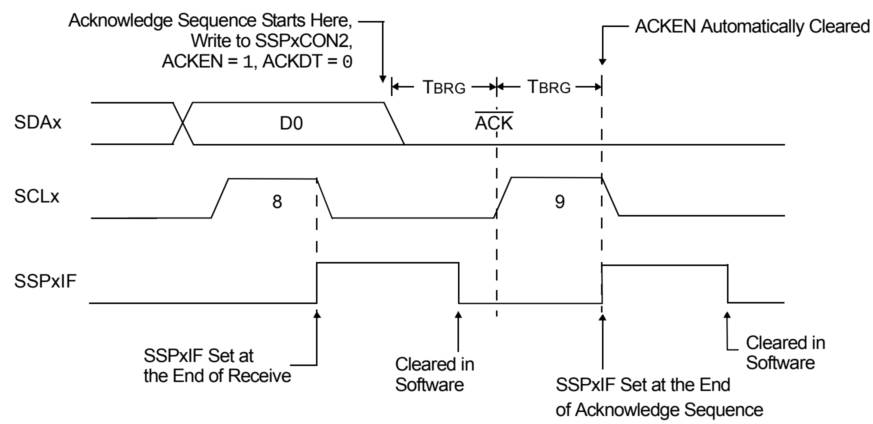
### 20.4.13 STOP CONDITION TIMING

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 SSPxIF bit is set (Figure 20-26).

#### 20.4.13.1 WCOL Status Flag

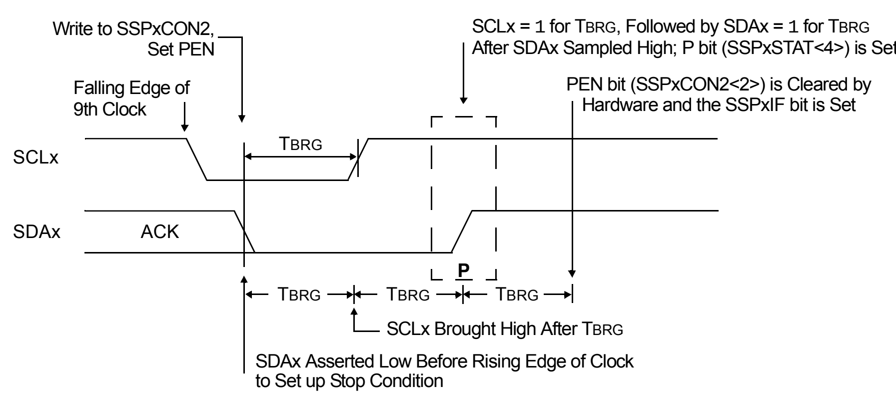
If the user writes the SSPxBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 20-25: ACKNOWLEDGE SEQUENCE WAVEFORM**




**Note:** TBRG = one Baud Rate Generator period.

**FIGURE 20-26: STOP CONDITION RECEIVE OR TRANSMIT MODE**




**Note:** TBRG = one Baud Rate Generator period.

### 20.4.14 SLEEP OPERATION

While in Sleep mode, the I$^2$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).

### 20.4.15 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

### 20.4.16 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 I$^2$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

### 20.4.17 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 I$^2$C port to its Idle state (Figure 20-27).

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 I$^2$C bus is free, the user can resume communication by asserting a Start condition.

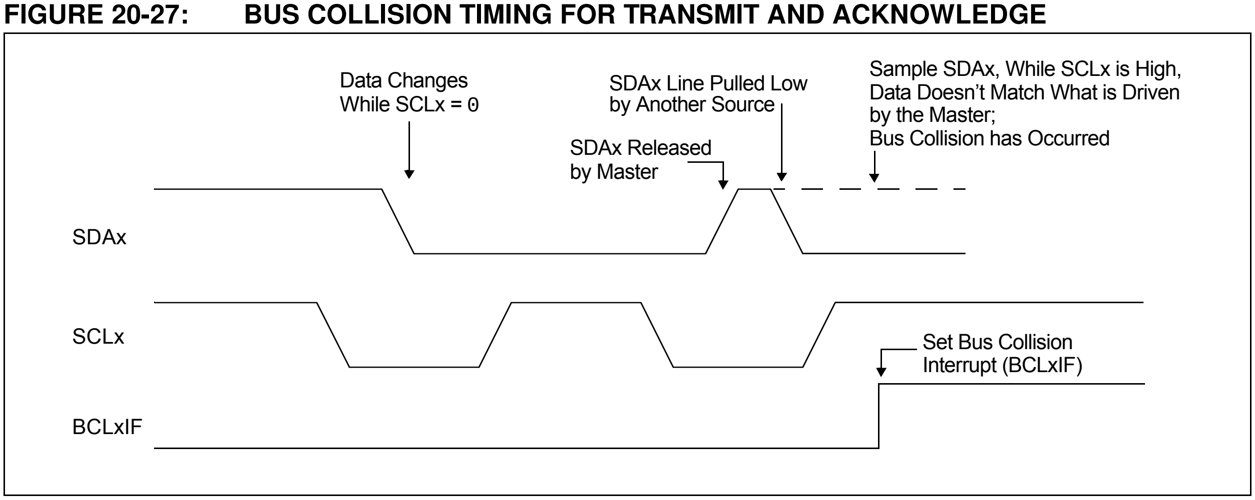
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDAx and SCLx lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine, and if the I$^2$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 I$^2$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 20-27: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**



#### 20.4.17.1 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 20-28).
b) SCLx is sampled low before SDAx is asserted low (Figure 20-29).

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 20-28)

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 20-30). 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 20-28: BUS COLLISION DURING START CONDITION (SDAx ONLY)**

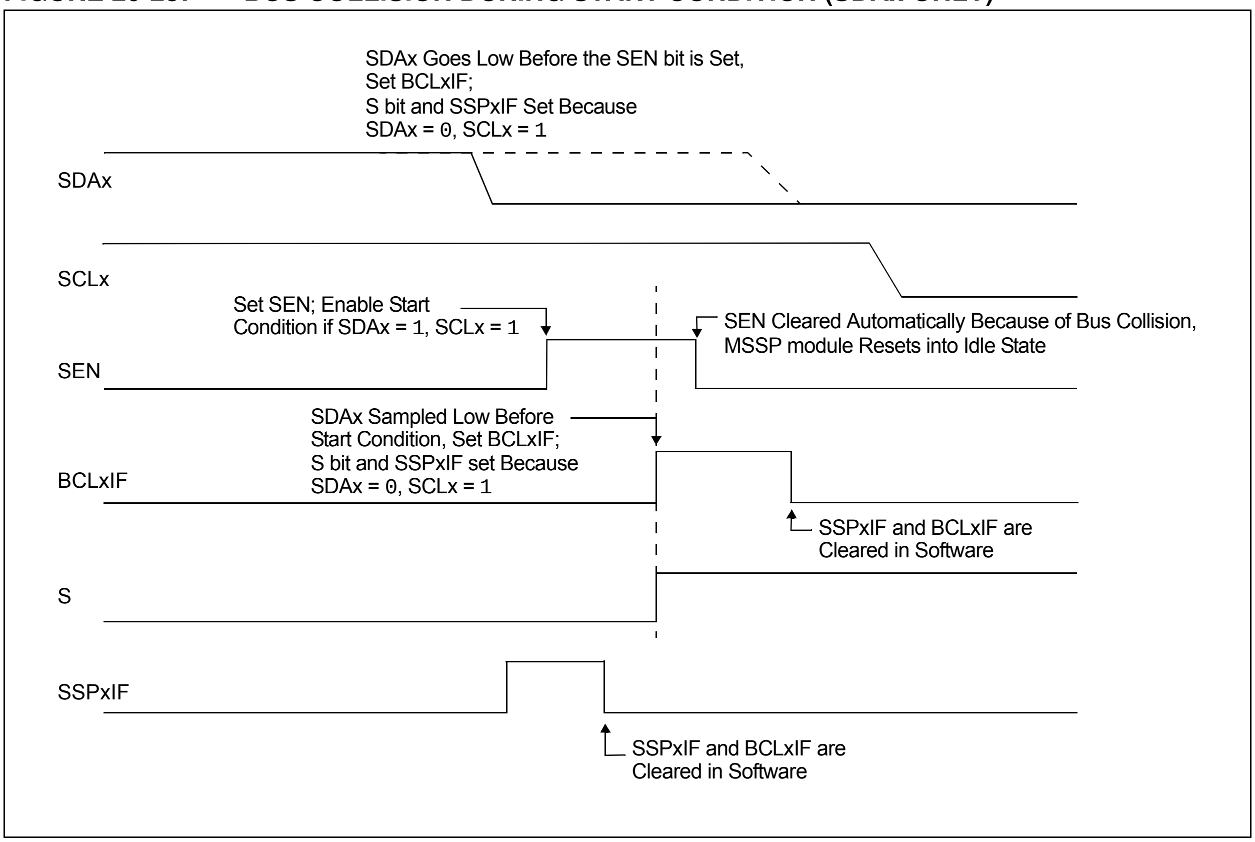


**FIGURE 20-29: BUS COLLISION DURING START CONDITION (SCLx = 0)**

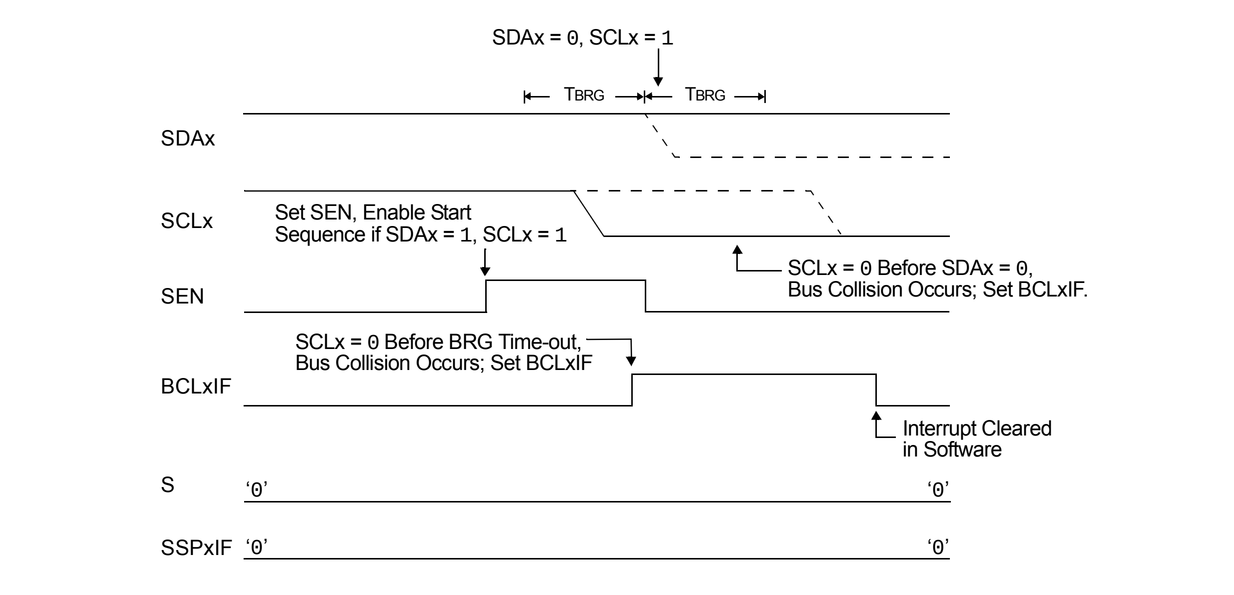
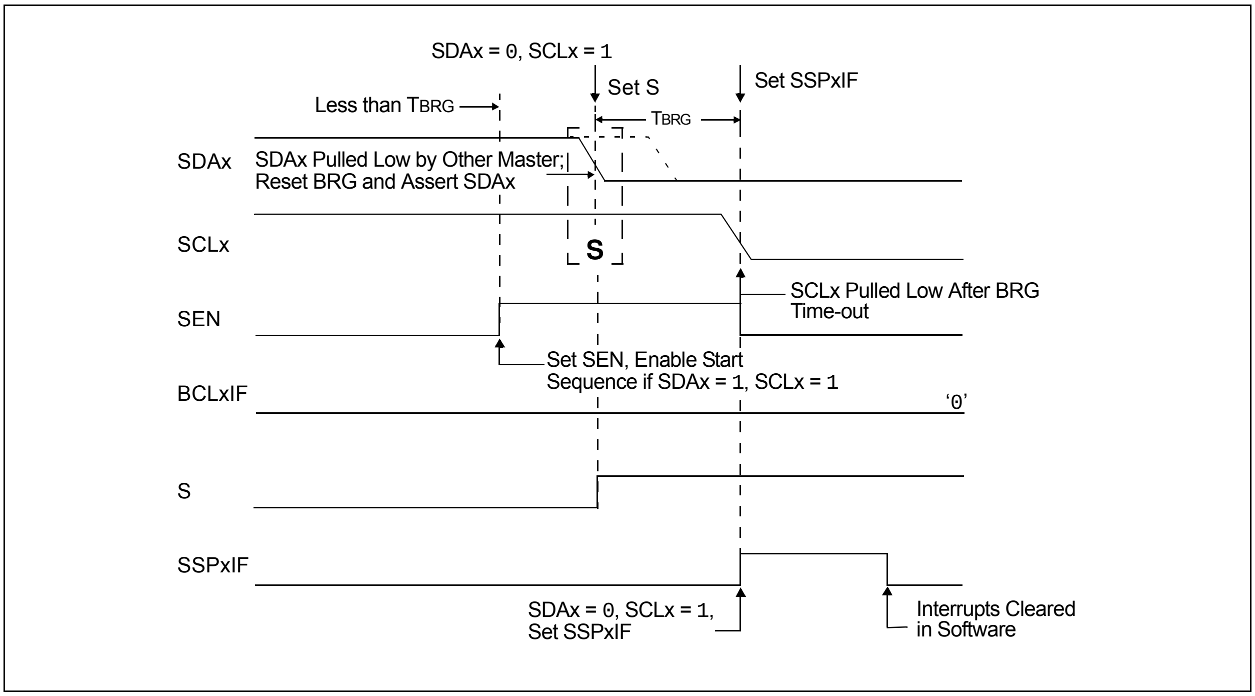



**FIGURE 20-30: BRG RESET DUE TO SDAx ARBITRATION DURING START CONDITION**



### 20.4.17.2 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 20-31). 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 20-32).

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 20-31: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**

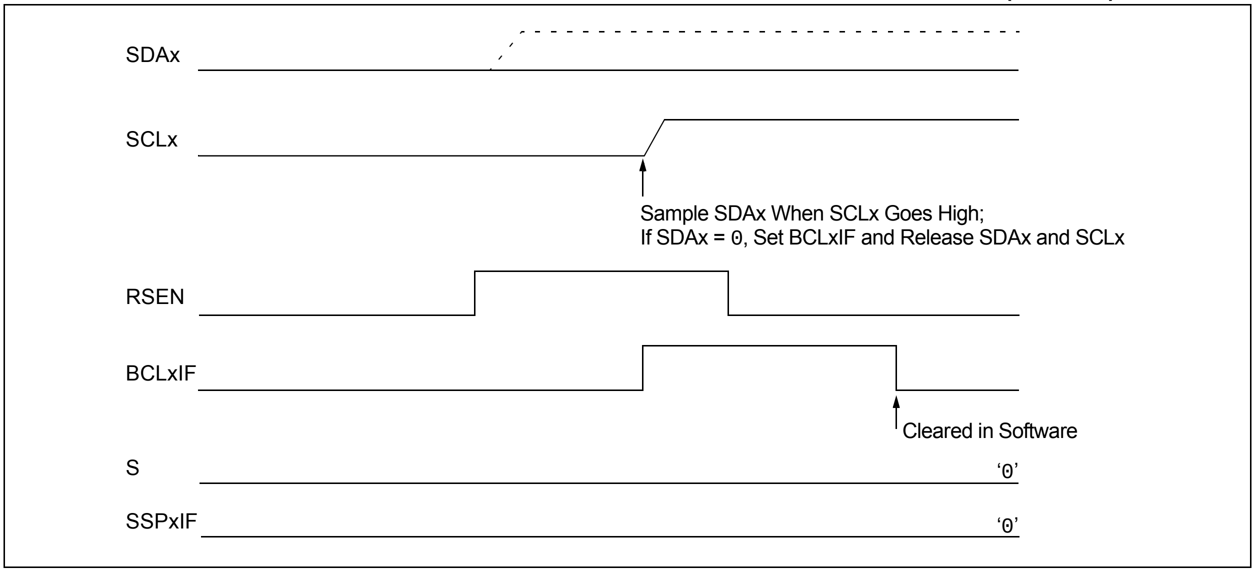
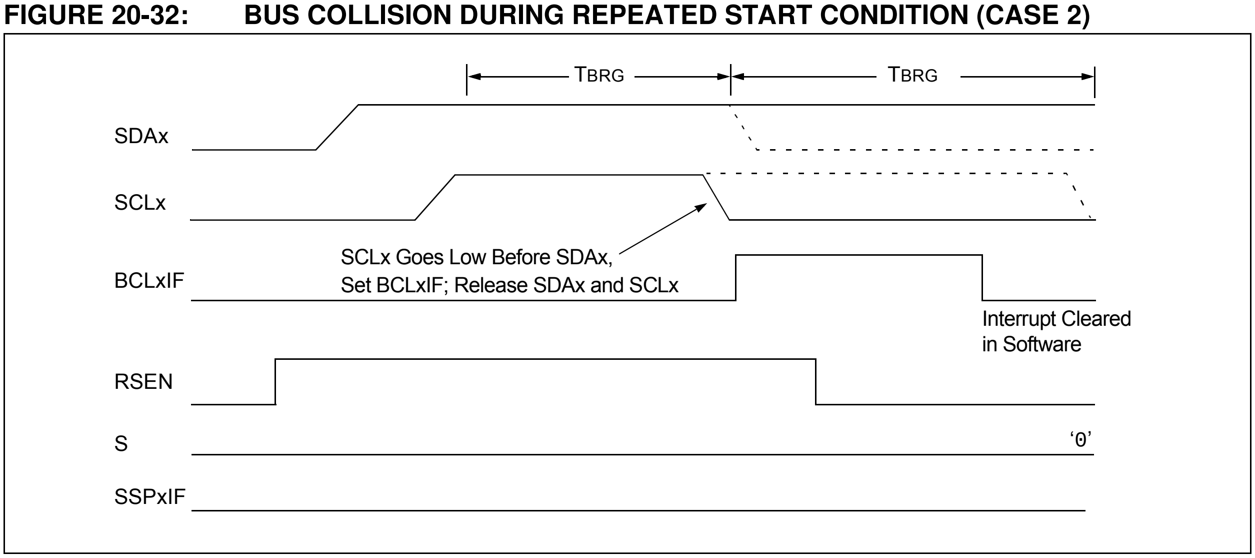



**FIGURE 20-32: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)**



#### 20.4.17.3 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 20-33). 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 20-34).

**FIGURE 20-33: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

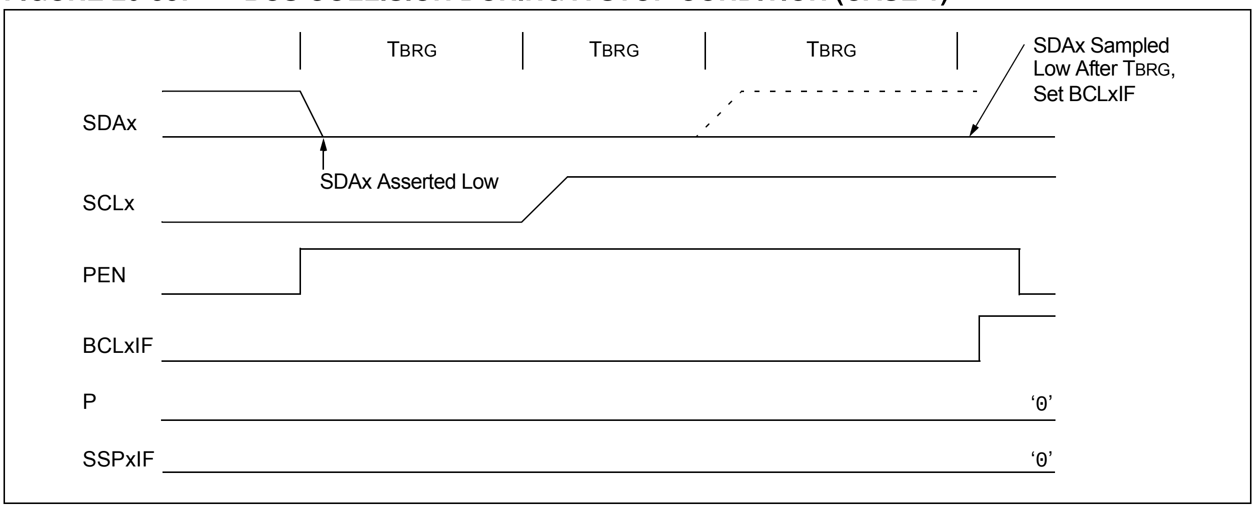



**FIGURE 20-34: BUS COLLISION DURING A STOP CONDITION (CASE 2)**

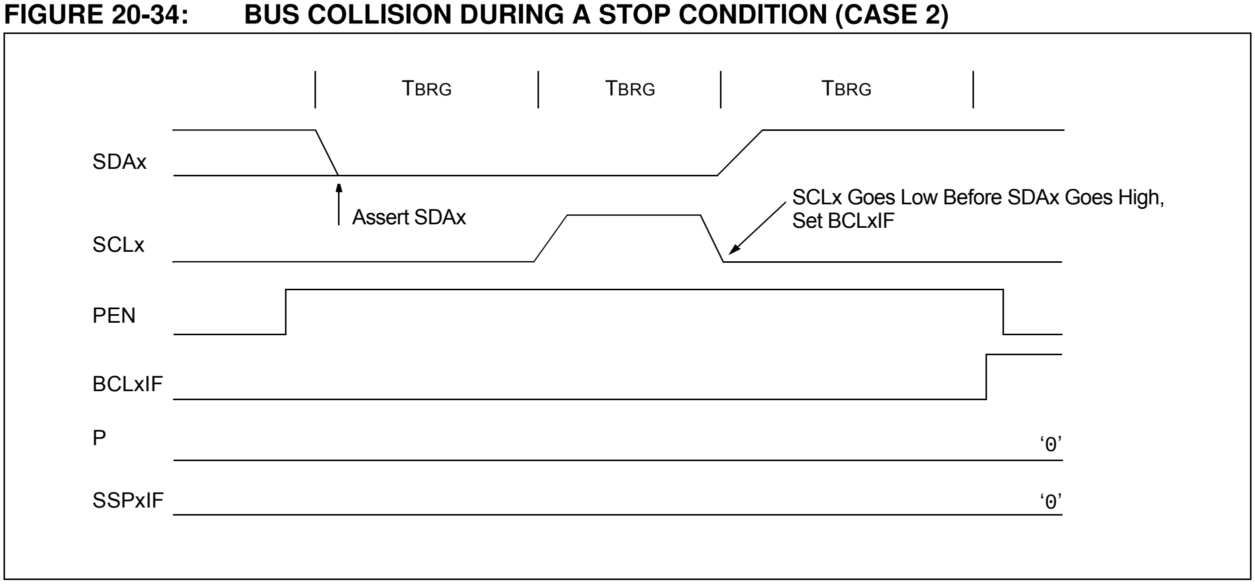


**TABLE 20-4: REGISTERS ASSOCIATED WITH I²C™ OPERATION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR2 | OSCFIF | CM2IF | CM1IF | — | BCL1IF | LVDIF | TMR3IF | CCP2IF | 64 |
| PIE2 | OSCFIE | CM2IE | CM1IE | — | BCL1IE | LVDIE | TMR3IE | CCP2IE | 64 |
| IPR2 | OSCFIP | CM2IP | CM1IP | — | BCL1IP | LVDIP | TMR3IP | CCP2IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 64 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 64 |
| SSP1BUF | MSSP1 Receive Buffer/Transmit Register | | | | | | | | 62 |
| SSP1ADD | MSSP1 Address Register (I²C™ Slave mode),<br>MSSP1 Baud Rate Reload Register (I²C Master mode) | | | | | | | | 62 |
| SSP1MSK[(1)] | MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0 | 62 |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 62 |
| SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 62 |
| | GCEN | ACKSTAT | ADMSK5[(2)] | ADMSK4[(2)] | ADMSK3[(2)] | ADMSK2[(2)] | ADMSK1[(2)] | SEN | |
| SSP1STAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 62 |
| SSP2BUF | MSSP2 Receive Buffer/Transmit Register | | | | | | | | 65 |
| SSP2ADD | MSSP2 Address Register (I²C Slave mode),<br>MSSP2 Baud Rate Reload Register (I²C Master mode) | | | | | | | | 65 |
| SSP2MSK[(1)] | MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0 | 65 |
| SSP2CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 65 |
| SSP2CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 65 |
| | GCEN | ACKSTAT | ADMSK5[(2)] | ADMSK4[(2)] | ADMSK3[(2)] | ADMSK2[(2)] | ADMSK1[(2)] | SEN | |
| SSP2STAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 65 |

**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 20.4.3.4 "7-Bit Address Masking Mode"** for more details.

**2:** Alternate bit definitions for use in I²C Slave mode operations only.

**NOTES:**

## 21.0 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/J2602 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 EUSARTx:

- For EUSART1:
  - SPEN bit (RCSTA1<7>) must be set (= `1`)
  - TRISC<7> bit must be set (= `1`)
  - TRISC<6> bit must be cleared (= `0`) for Asynchronous and Synchronous Master modes
  - TRISC<6> bit must be set (= `1`) for Synchronous Slave mode
- For EUSART2:
  - SPEN bit (RCSTA2<7>) must be set (= `1`)
  - TRISG<2> bit must be set (= `1`)
  - TRISG<1> bit must be cleared (= `0`) for Asynchronous and Synchronous Master modes
  - TRISC<6> bit must be set (= `1`) for Synchronous Slave mode

> **Note:** The EUSARTx 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 21-1, Register 21-2 and Register 21-3, 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 21-1: TXSTAx: EUSARTx TRANSMIT STATUS AND CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-1 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CSRC | TX9 | TXEN[(1)] | SYNC | SENDB | BRGH | TRMT | TX9D |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| -n = 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 is enabled
0 = Transmit is disabled

bit 4 **SYNC:** EUSARTx Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Sends Sync Break on the next transmission (cleared by hardware upon completion)
0 = Sync Break transmission has 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 is empty
0 = TSR is full

bit 0 **TX9D:** 9th bit of Transmit Data
This can be an address/data bit or a parity bit.

**Note 1:** SREN/CREN overrides TXEN in Sync mode.

**REGISTER 21-2: RCSTAx: EUSARTx RECEIVE STATUS AND CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-0 | R-x |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | ADDEN | FERR[(1)] | OERR[(1)] | RX9D |
| 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 **SPEN:** Serial Port Enable bit
1 = Serial port is enabled
0 = Serial port is 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 the 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 a parity bit
Asynchronous mode 8-Bit (RX9 = 0):
Don’t care.

bit 2 **FERR:** Framing Error bit[(1)]
1 = Framing Error (can be cleared by reading the RCREGx register and receiving the next valid byte)
0 = No Framing Error

bit 1 **OERR:** Overrun Error bit[(1)]
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 an address/data bit or a parity bit and must be calculated by user firmware.

**Note 1:** Once a Framing or Overrun Error occurs, further EUSART reception will be discarded until cleared.

**REGISTER 21-3: BAUDCONx: BAUD RATE CONTROL REGISTER**

| R/W-0 | R-1 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| 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 **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 is ignored

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

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = EUSARTx will continue to sample the RXx pin – interrupt is generated on the falling edge; bit is cleared in hardware on the following rising edge
0 = RXx pin is not monitored or rising edge detected
Synchronous mode:
Unused in this mode.

bit 0 **ABDEN:** Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Enables baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
0 = Baud rate measurement is disabled or has completed
Synchronous mode:
Unused in this mode.

## 21.1 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 EUSARTx. 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 21-1 shows the formula for computation of the baud rate for different EUSARTx 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 21-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 21-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 21-2. 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. When operated in Synchronous mode, SPBRGHx:SPBRGx values of 0000h and 0001h are not supported. In the Asynchronous mode, all BRG values may be used.

### 21.1.1 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.

### 21.1.2 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 21-1: BAUD RATE FORMULAS**

| Configuration Bits | | | BRG/EUSARTx Mode | Baud Rate Formula |
|---|---|---|---|---|
| **SYNC** | **BRG16** | **BRGH** | | |
| 0 | 0 | 0 | 8-bit/Asynchronous | FOSC/[64 (n + 1)] |
| 0 | 0 | 1 | 8-bit/Asynchronous | FOSC/[16 (n + 1)] |
| 0 | 1 | 0 | 16-bit/Asynchronous | |
| 0 | 1 | 1 | 16-bit/Asynchronous | FOSC/[4 (n + 1)] |
| 1 | 0 | x | 8-bit/Synchronous | |
| 1 | 1 | x | 16-bit/Synchronous | |

**Legend:** x = Don’t care; n = value of SPBRGHx:SPBRGx register pair

### EXAMPLE 21-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, and 8-bit BRG:

Desired Baud Rate = FOSC/(64 ([SPBRGHx:SPBRGx] + 1))

Solving for SPBRGHx:SPBRGx:

X = ((FOSC/Desired Baud Rate)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 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 21-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| TXSTAx | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 63 |
| RCSTAx | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 63 |
| BAUDCONx | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 65 |
| SPBRGHx | EUSARTx Baud Rate Generator Register High Byte | | | | | | | | 65 |
| SPBRGx | EUSARTx Baud Rate Generator Register Low Byte | | | | | | | | 65 |

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

**TABLE 21-3: BAUD RATES FOR ASYNCHRONOUS MODES**

| Baud Rate (K) | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 40.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 10.000 MHz | | | FOSC = 8.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) |
| 0.3 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1.2 | — | — | — | 1.221 | 1.73 | 255 | 1.202 | 0.16 | 129 | 1.201 | -0.16 | 103 |
| 2.4 | 2.441 | 1.73 | 255 | 2.404 | 0.16 | 129 | 2.404 | 0.16 | 64 | 2.403 | -0.16 | 51 |
| 9.6 | 9.615 | 0.16 | 64 | 9.766 | 1.73 | 31 | 9.766 | 1.73 | 15 | 9.615 | -0.16 | 12 |
| 19.2 | 19.531 | 1.73 | 31 | 19.531 | 1.73 | 15 | 19.531 | 1.73 | 7 | — | — | — |
| 57.6 | 56.818 | -1.36 | 10 | 62.500 | 8.51 | 4 | 52.083 | -9.58 | 2 | — | — | — |
| 115.2 | 125.000 | 8.51 | 4 | 104.167 | -9.58 | 2 | 78.125 | -32.18 | 1 | — | — | — |

| Baud Rate (K) | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 4.000 MHz | | | FOSC = 2.000 MHz | | | FOSC = 1.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) |
| 0.3 | 0.300 | 0.16 | 207 | 0.300 | -0.16 | 103 | 0.300 | -0.16 | 51 |
| 1.2 | 1.202 | 0.16 | 51 | 1.201 | -0.16 | 25 | 1.201 | -0.16 | 12 |
| 2.4 | 2.404 | 0.16 | 25 | 2.403 | -0.16 | 12 | — | — | — |
| 9.6 | 8.929 | -6.99 | 6 | — | — | — | — | — | — |
| 19.2 | 20.833 | 8.51 | 2 | — | — | — | — | — | — |
| 57.6 | 62.500 | 8.51 | 0 | — | — | — | — | — | — |
| 115.2 | 62.500 | -45.75 | 0 | — | — | — | — | — | — |

| Baud Rate (K) | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 40.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 10.000 MHz | | | FOSC = 8.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) |
| 0.3 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1.2 | — | — | — | — | — | — | — | — | — | — | — | — |
| 2.4 | — | — | — | — | — | — | 2.441 | 1.73 | 255 | 2.403 | -0.16 | 207 |
| 9.6 | 9.766 | 1.73 | 255 | 9.615 | 0.16 | 129 | 9.615 | 0.16 | 64 | 9.615 | -0.16 | 51 |
| 19.2 | 19.231 | 0.16 | 129 | 19.231 | 0.16 | 64 | 19.531 | 1.73 | 31 | 19.230 | -0.16 | 25 |
| 57.6 | 58.140 | 0.94 | 42 | 56.818 | -1.36 | 21 | 56.818 | -1.36 | 10 | 55.555 | 3.55 | 8 |
| 115.2 | 113.636 | -1.36 | 21 | 113.636 | -1.36 | 10 | 125.000 | 8.51 | 4 | — | — | — |

| Baud Rate (K) | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 4.000 MHz | | | FOSC = 2.000 MHz | | | FOSC = 1.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) |
| 0.3 | — | — | — | — | — | — | 0.300 | -0.16 | 207 |
| 1.2 | 1.202 | 0.16 | 207 | 1.201 | -0.16 | 103 | 1.201 | -0.16 | 51 |
| 2.4 | 2.404 | 0.16 | 103 | 2.403 | -0.16 | 51 | 2.403 | -0.16 | 25 |
| 9.6 | 9.615 | 0.16 | 25 | 9.615 | -0.16 | 12 | — | — | — |
| 19.2 | 19.231 | 0.16 | 12 | — | — | — | — | — | — |
| 57.6 | 62.500 | 8.51 | 3 | — | — | — | — | — | — |
| 115.2 | 125.000 | 8.51 | 1 | — | — | — | — | — | — |

**TABLE 21-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)**

| Baud Rate (K) | SYNC = 0, BRGH = 0, BRG16 = 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 40.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 10.000 MHz | | | FOSC = 8.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) |
| 0.3 | 0.300 | 0.00 | 8332 | 0.300 | 0.02 | 4165 | 0.300 | 0.02 | 2082 | 0.300 | -0.04 | 1665 |
| 1.2 | 1.200 | 0.02 | 2082 | 1.200 | -0.03 | 1041 | 1.200 | -0.03 | 520 | 1.201 | -0.16 | 415 |
| 2.4 | 2.402 | 0.06 | 1040 | 2.399 | -0.03 | 520 | 2.404 | 0.16 | 259 | 2.403 | -0.16 | 207 |
| 9.6 | 9.615 | 0.16 | 259 | 9.615 | 0.16 | 129 | 9.615 | 0.16 | 64 | 9.615 | -0.16 | 51 |
| 19.2 | 19.231 | 0.16 | 129 | 19.231 | 0.16 | 64 | 19.531 | 1.73 | 31 | 19.230 | -0.16 | 25 |
| 57.6 | 58.140 | 0.94 | 42 | 56.818 | -1.36 | 21 | 56.818 | -1.36 | 10 | 55.555 | 3.55 | 8 |
| 115.2 | 113.636 | -1.36 | 21 | 113.636 | -1.36 | 10 | 125.000 | 8.51 | 4 | — | — | — |

| Baud Rate (K) | SYNC = 0, BRGH = 0, BRG16 = 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 4.000 MHz | | | FOSC = 2.000 MHz | | | FOSC = 1.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) |
| 0.3 | 0.300 | 0.04 | 832 | 0.300 | -0.16 | 415 | 0.300 | -0.16 | 207 |
| 1.2 | 1.202 | 0.16 | 207 | 1.201 | -0.16 | 103 | 1.201 | -0.16 | 51 |
| 2.4 | 2.404 | 0.16 | 103 | 2.403 | -0.16 | 51 | 2.403 | -0.16 | 25 |
| 9.6 | 9.615 | 0.16 | 25 | 9.615 | -0.16 | 12 | — | — | — |
| 19.2 | 19.231 | 0.16 | 12 | — | — | — | — | — | — |
| 57.6 | 62.500 | 8.51 | 3 | — | — | — | — | — | — |
| 115.2 | 125.000 | 8.51 | 1 | — | — | — | — | — | — |

| Baud Rate (K) | SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 40.000 MHz | | | FOSC = 20.000 MHz | | | FOSC = 10.000 MHz | | | FOSC = 8.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) |
| 0.3 | 0.300 | 0.00 | 33332 | 0.300 | 0.00 | 16665 | 0.300 | 0.00 | 8332 | 0.300 | -0.01 | 6665 |
| 1.2 | 1.200 | 0.00 | 8332 | 1.200 | 0.02 | 4165 | 1.200 | 0.02 | 2082 | 1.200 | -0.04 | 1665 |
| 2.4 | 2.400 | 0.02 | 4165 | 2.400 | 0.02 | 2082 | 2.402 | 0.06 | 1040 | 2.400 | -0.04 | 832 |
| 9.6 | 9.606 | 0.06 | 1040 | 9.596 | -0.03 | 520 | 9.615 | 0.16 | 259 | 9.615 | -0.16 | 207 |
| 19.2 | 19.193 | -0.03 | 520 | 19.231 | 0.16 | 259 | 19.231 | 0.16 | 129 | 19.230 | -0.16 | 103 |
| 57.6 | 57.803 | 0.35 | 172 | 57.471 | -0.22 | 86 | 58.140 | 0.94 | 42 | 57.142 | 0.79 | 34 |
| 115.2 | 114.943 | -0.22 | 86 | 116.279 | 0.94 | 42 | 113.636 | -1.36 | 21 | 117.647 | -2.12 | 16 |

| Baud Rate (K) | SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | FOSC = 4.000 MHz | | | FOSC = 2.000 MHz | | | FOSC = 1.000 MHz | | |
| | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) | Actual Rate (K) | % Error | SPBRG Value (decimal) |
| 0.3 | 0.300 | 0.01 | 3332 | 0.300 | -0.04 | 1665 | 0.300 | -0.04 | 832 |
| 1.2 | 1.200 | 0.04 | 832 | 1.201 | -0.16 | 415 | 1.201 | -0.16 | 207 |
| 2.4 | 2.404 | 0.16 | 415 | 2.403 | -0.16 | 207 | 2.403 | -0.16 | 103 |
| 9.6 | 9.615 | 0.16 | 103 | 9.615 | -0.16 | 51 | 9.615 | -0.16 | 25 |
| 19.2 | 19.231 | 0.16 | 51 | 19.230 | -0.16 | 25 | 19.230 | -0.16 | 12 |
| 57.6 | 58.824 | 2.12 | 16 | 55.555 | 3.55 | 8 | — | — | — |
| 115.2 | 111.111 | -3.55 | 8 | — | — | — | — | — | — |

### 21.1.3 AUTO-BAUD RATE DETECT

The Enhanced USART modules support 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 21-1) 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/J2602 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 rollovers 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 21-2).

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. The BRG16 bit must be set to use both SPBRGx and SPBRGHx as a 16-bit counter. 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 21-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the EUSARTx 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 EUSARTx 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 (BAUDCONx<3>) is set to enable the auto-baud feature.

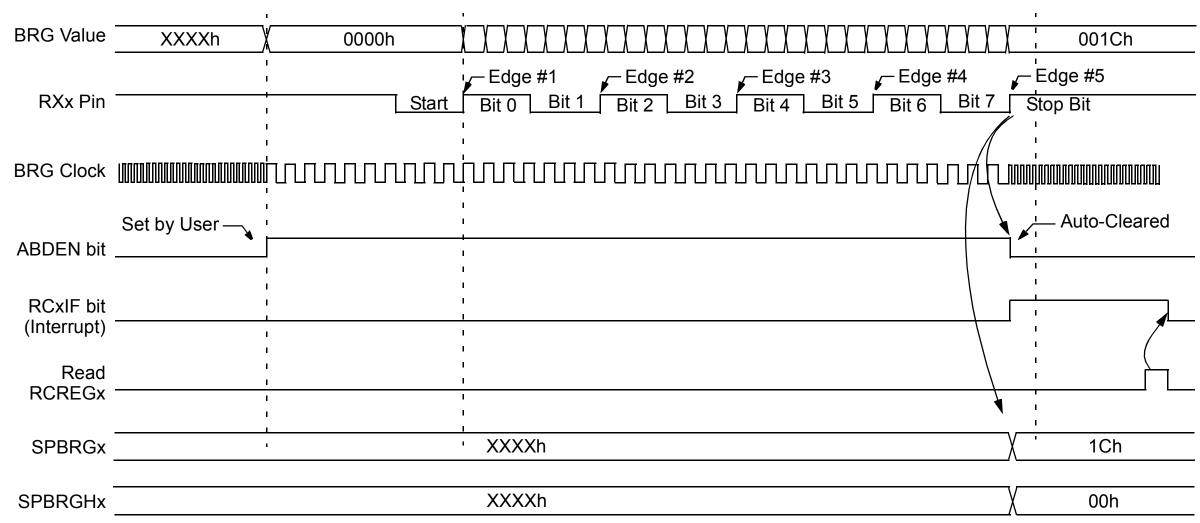
**TABLE 21-4: BRG COUNTER CLOCK RATES**

| BRG16 | BRGH | BRG Counter Clock |
|---|---|---|
| 0 | 0 | FOSC/512 |
| 0 | 1 | FOSC/128 |
| 1 | 0 | FOSC/128 |
| 1 | 1 | FOSC/32 |

#### 21.1.3.1 ABD and EUSARTx Transmission

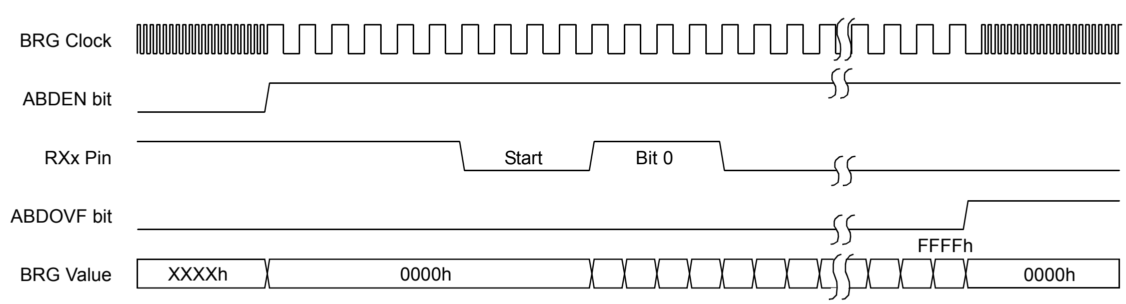
Since the BRG clock is reversed during ABD acquisition, the EUSARTx transmitter cannot be used during ABD. This means that whenever the ABDEN bit is set, TXREGx cannot be written to. Users should also ensure that ABDEN does not become set during a transmit sequence. Failing to do this may result in unpredictable EUSART operation.

**FIGURE 21-1: AUTOMATIC BAUD RATE CALCULATION**




**Note:** The ABD sequence requires the EUSARTx module to be configured in Asynchronous mode and WUE = 0.

**FIGURE 21-2: BRG OVERFLOW SEQUENCE**



## 21.2 EUSARTx Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTAx<4>). In this mode, the EUSARTx 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 EUSARTx transmits and receives the LSb first. The EUSARTx'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 EUSARTx 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

### 21.2.1 EUSARTx ASYNCHRONOUS TRANSMITTER

The EUSARTx transmitter block diagram is shown in Figure 21-3. 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 $T_{CY}$), the TXREGx register is empty and the TXxIF flag bit is set. This interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxIE. TXxIF will be set regardless of the state of TXxIE; it cannot be cleared in software. TXxIF is also not cleared immediately upon loading TXREGx, but becomes valid in the second instruction cycle following the load instruction. Polling TXxIF immediately following a load of TXREGx will return invalid results.

While TXxIF 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.
>
> **2:** Flag bit, TXxIF, is set when enable bit, TXEN, 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, TXxIE.
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, TXxIF.
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 21-3: EUSARTx TRANSMIT BLOCK DIAGRAM**

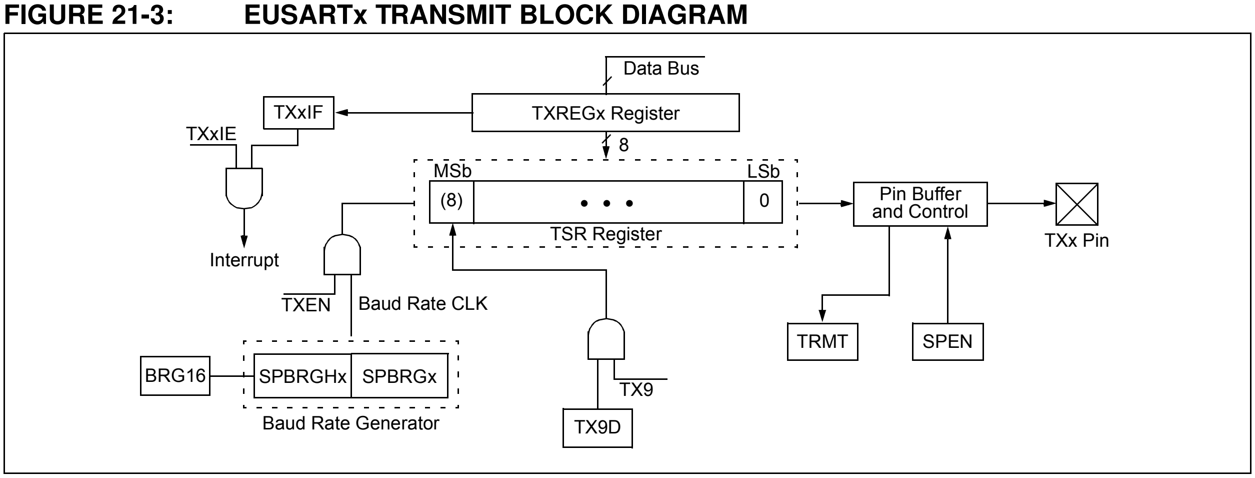


**FIGURE 21-4: ASYNCHRONOUS TRANSMISSION**

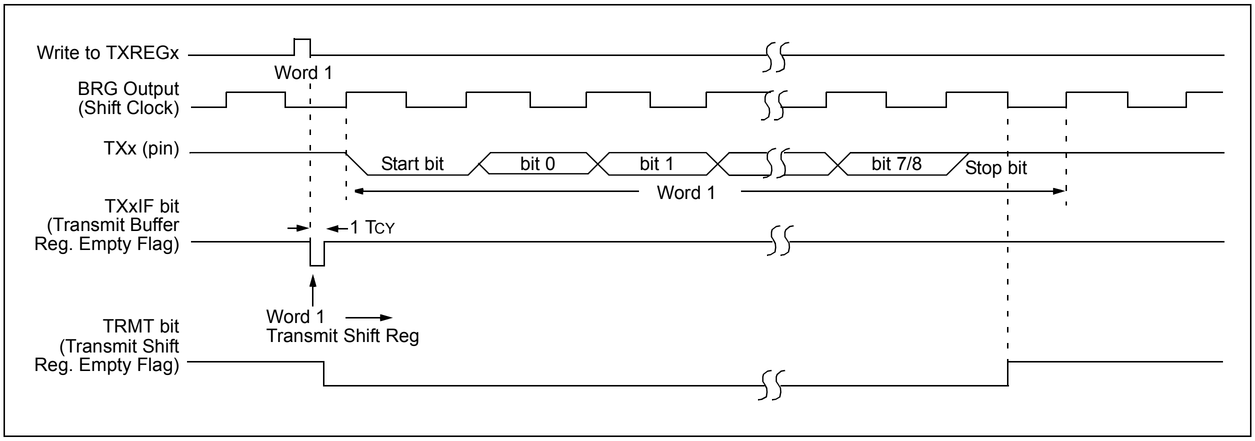
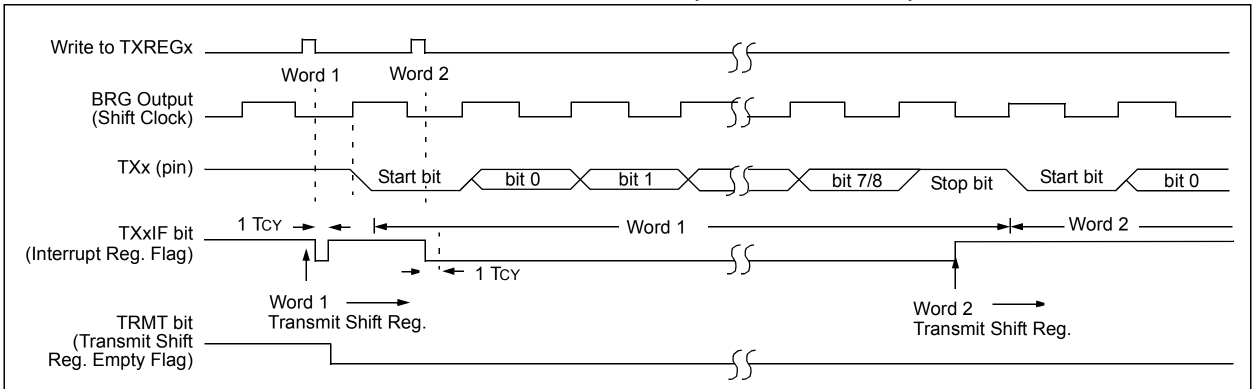



**FIGURE 21-5: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**




**Note:** This timing diagram shows two consecutive transmissions.

**TABLE 21-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| RCSTAx | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 63 |
| TXREGx | EUSARTx Transmit Register | | | | | | | | 63 |
| TXSTAx | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 63 |
| BAUDCONx | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 65 |
| SPBRGHx | EUSARTx Baud Rate Generator Register High Byte | | | | | | | | 65 |
| SPBRGx | EUSARTx Baud Rate Generator Register Low Byte | | | | | | | | 65 |

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

### 21.2.2 EUSARTx ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 21-6. 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.

#### 21.2.2.1 Receiving Data

The receiver data recovery circuit initiates character reception on the falling edge of the first bit. The first bit, also known as the Start bit, is always a zero (after accounting for the RXDTP setting). Following the Start bit will be the Least Significant bit of the data character being received. As each bit is received, the value will be sampled and shifted into the Receive Shift Register (RSR). After all 8 or 9 data bits (user-selectable option) of the character have been shifted in, one final bit time is measured and the level is sampled. This is the Stop bit, which should always be a '1' (after accounting for the RXDTP setting). If the data recovery circuit samples a '0' in the Stop bit position, then a Framing Error (FERR) is set for this character; otherwise, the Framing Error is cleared for this character.

Once all data bits of the character and the Stop bit have been received, the data bits in the RSR will immediately be transferred to a two-character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters before software is required to service the EUSARTx receiver. The RSR register is not directly accessible by software. Firmware can read data from the FIFO by reading the RCREGx register. Each firmware initiated read from the RCREGx register will advance the FIFO by one character and will clear the EUSARTx Receive Interrupt Flag (RCxIF) if no additional data exists in the FIFO.

#### 21.2.2.2 Receive Overrun Error

If the user firmware allows the FIFO to become full, and a third character is received before the firmware reads from RCREGx, a buffer Overrun Error (OERR) condition will occur. In this case, the hardware will block the RSR contents (the third byte received) from being copied into the receive FIFO, the character will be lost and the OERR status bit in the RCSTAx register will become set. If an OERR condition is allowed to occur, firmware must clear the condition by clearing, and then resetting CREN, before additional characters can be successfully received.

#### 21.2.2.3 Setting Up Asynchronous Receive

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, RCxIF, will be set when reception is complete and an interrupt will be generated if enable 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.

### 21.2.3 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 RCxIF 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 21-6: EUSARTx RECEIVE BLOCK DIAGRAM**

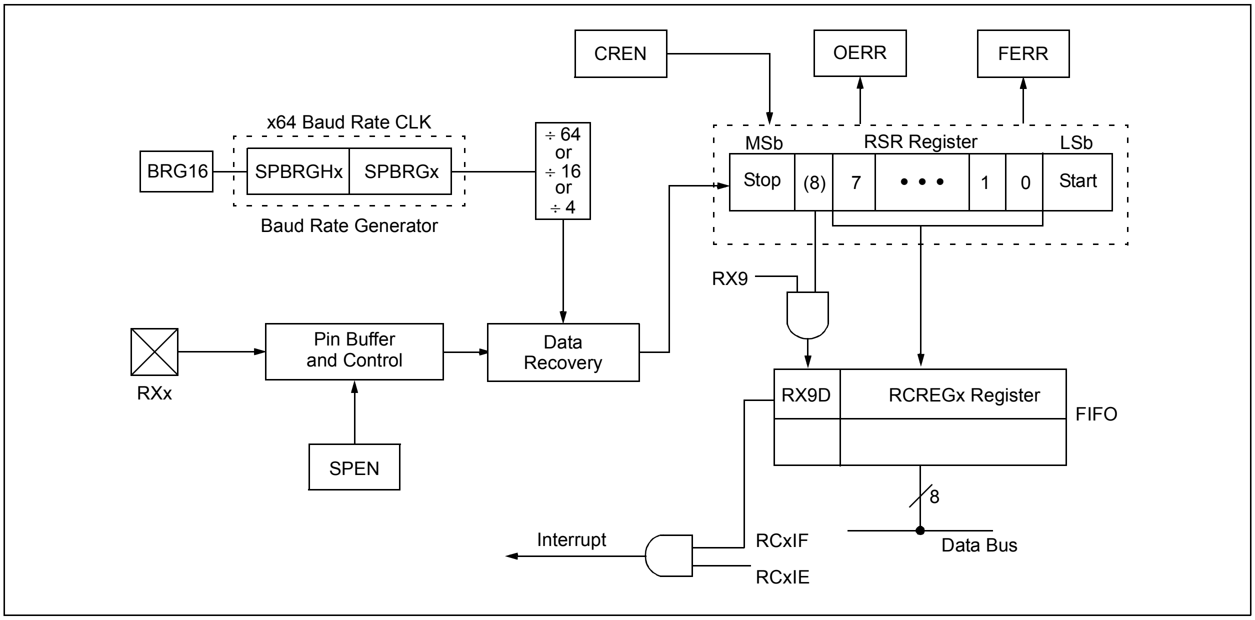
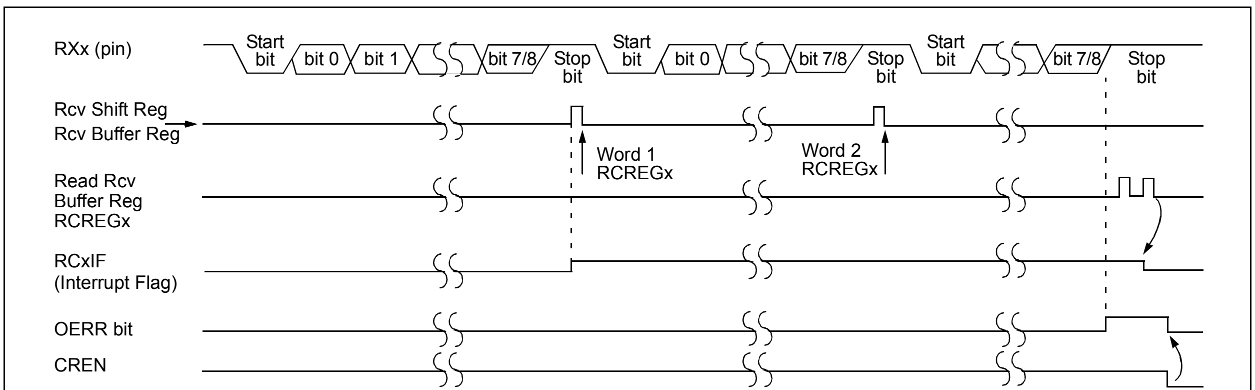



**FIGURE 21-7: ASYNCHRONOUS RECEPTION**




**Note:** This timing diagram shows three words appearing on the RXx input. The RCREGx (Receive Buffer) is read after the third word, causing the OERR (Overrun Error) bit to be set.

**TABLE 21-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| RCSTAx | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 63 |
| RCREGx | EUSARTx Receive Register | | | | | | | | 63 |
| TXSTAx | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 63 |
| BAUDCONx | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 65 |
| SPBRGHx | EUSARTx Baud Rate Generator Register High Byte | | | | | | | | 65 |
| SPBRGx | EUSARTx Baud Rate Generator Register Low Byte | | | | | | | | 65 |

**Legend:** — = unimplemented locations read as ‘0’. Shaded cells are not used for asynchronous reception.

### 21.2.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the EUSARTx 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 EUSARTx 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 EUSARTx 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/J2602 protocol.)

Following a wake-up event, the module generates an RCxIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 21-8) and asynchronously if the device is in Sleep mode (Figure 21-9). 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 EUSARTx module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.

#### 21.2.4.1 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 bits) for standard RS-232 devices or 000h (12 bits) for the LIN/J2602 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 EUSARTx.

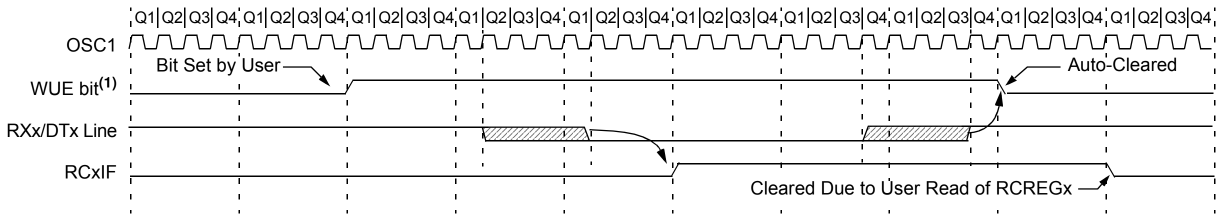
#### 21.2.4.2 Special Considerations Using the WUE Bit

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 EUSARTx 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.

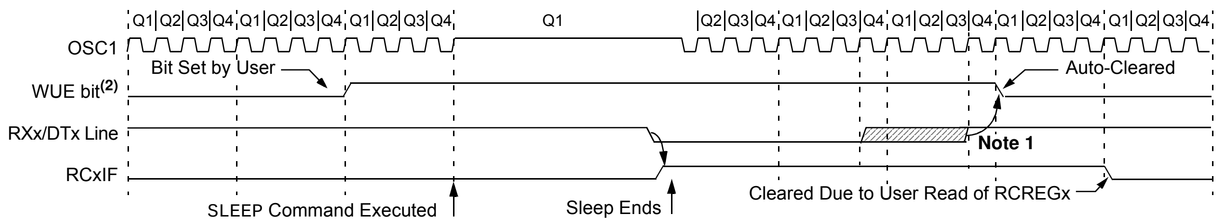
To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

**FIGURE 21-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION**




**Note 1:** The EUSARTx remains in Idle while the WUE bit is set.

**FIGURE 21-9: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP**




**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.

**2:** The EUSARTx remains in Idle while the WUE bit is set.

### 21.2.5 BREAK CHARACTER SEQUENCE

The EUSARTx module has the capability of sending the special Break character sequences that are required by the LIN/J2602 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 (TXSTAx<3> and TXSTAx<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/J2602 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 21-10 for the timing of the Break character sequence.

#### 21.2.5.1 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/J2602 bus master.

1. Configure the EUSARTx 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.

### 21.2.6 RECEIVING A BREAK CHARACTER

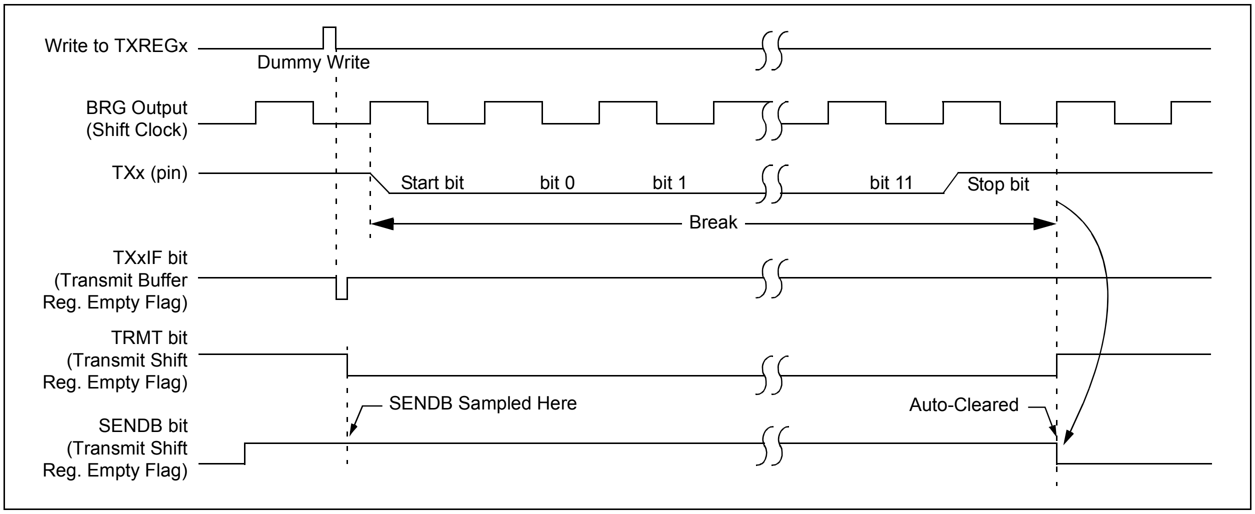
The Enhanced USARTx modules can receive a Break character in two ways.

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 21.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the EUSARTx 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 21-10: SEND BREAK CHARACTER SEQUENCE**



## 21.3 EUSARTx 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.

### 21.3.1 EUSARTx SYNCHRONOUS MASTER TRANSMISSION

The EUSARTx transmitter block diagram is shown in Figure 21-3. 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).

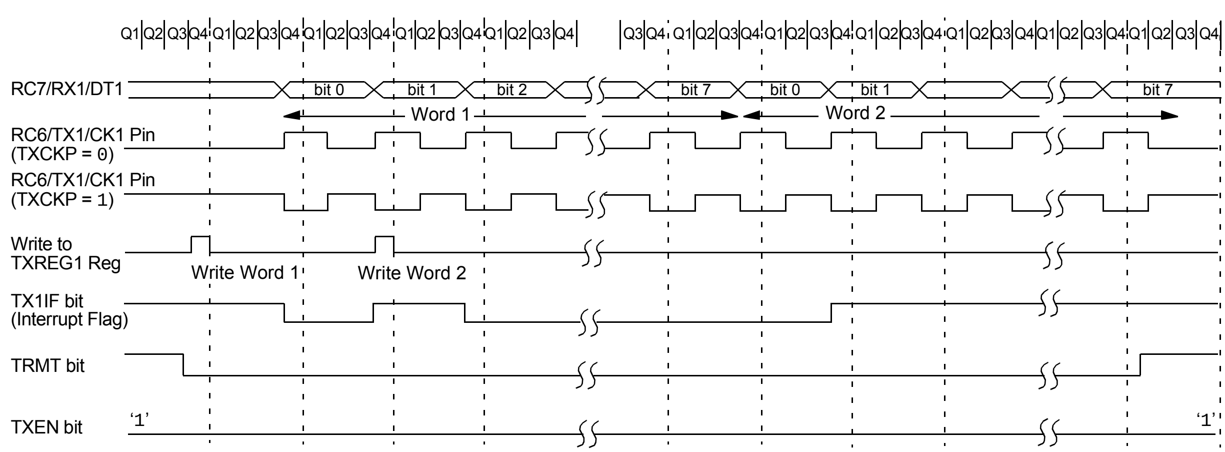
Once the TXREGx register transfers the data to the TSR register (occurs in one TCY), the TXREGx is empty and the TXxIF flag bit is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxIE. TXxIF is set regardless of the state of enable bit, TXxIE; it cannot be cleared in software. It will reset only when new data is loaded into the TXREGx register.

While flag bit, TXxIF, 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:

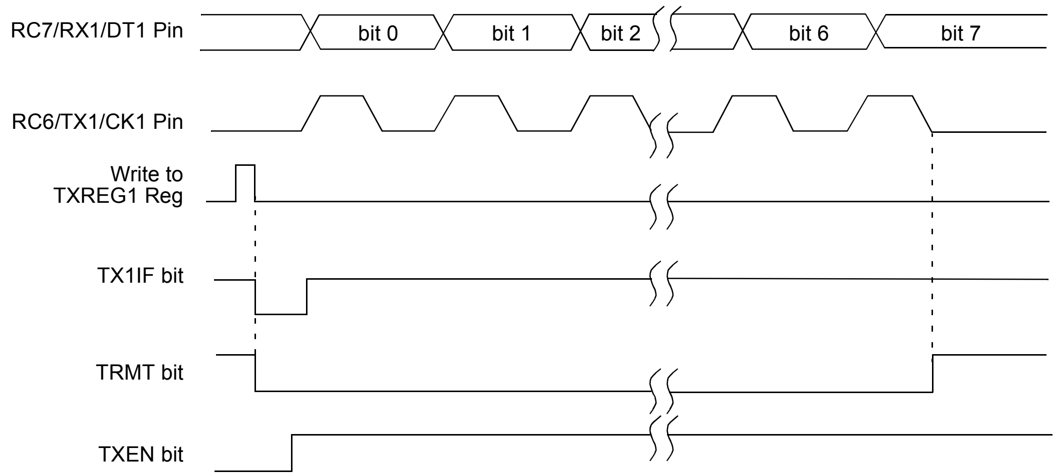
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit, TXxIE.
4. If 9-bit transmission 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.

**FIGURE 21-11: SYNCHRONOUS TRANSMISSION**




**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 21-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**




**Note:** This example is equally applicable to EUSART2 (RG1/TX2/CK2 and RG2/RX2/DT2).

**TABLE 21-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| RCSTAx | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 63 |
| TXREGx | EUSARTx Transmit Register | | | | | | | | 63 |
| TXSTAx | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 63 |
| BAUDCONx | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 65 |
| SPBRGHx | EUSARTx Baud Rate Generator Register High Byte | | | | | | | | 65 |
| SPBRGx | EUSARTx Baud Rate Generator Register Low Byte | | | | | | | | 65 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

### 21.3.2 EUSARTx SYNCHRONOUS MASTER RECEPTION

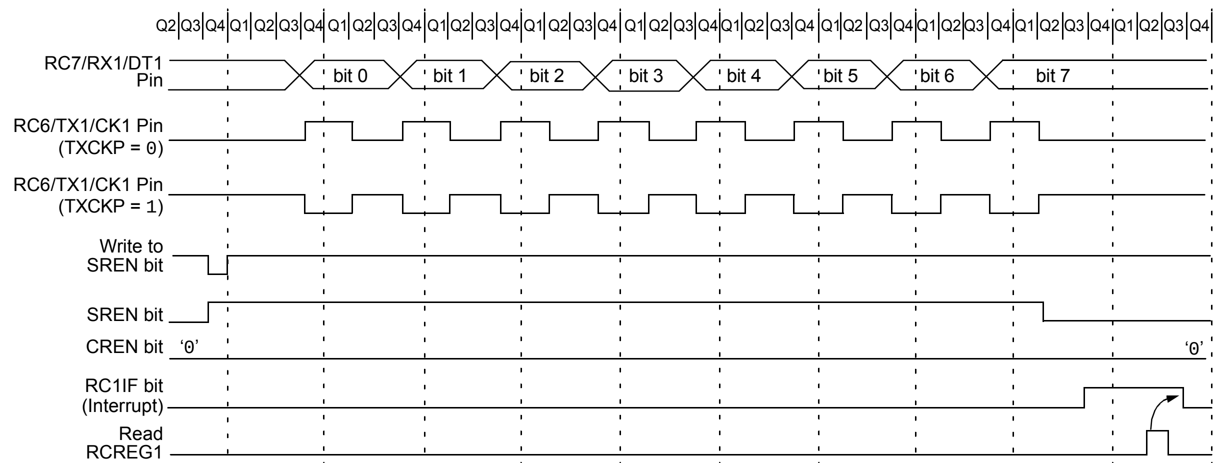
Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTAx<5>) or the Continuous Receive Enable bit, CREN (RCSTAx<4>). Data is sampled on the RXx pin on the falling edge of the clock.

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 21-13: 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).

**TABLE 21-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| RCSTAx | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 63 |
| RCREGx | EUSARTx Receive Register | | | | | | | | 63 |
| TXSTAx | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 63 |
| BAUDCONx | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 65 |
| SPBRGHx | EUSARTx Baud Rate Generator Register High Byte | | | | | | | | 65 |
| SPBRGx | EUSARTx Baud Rate Generator Register Low Byte | | | | | | | | 65 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.

## 21.4 EUSARTx 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.

### 21.4.1 EUSARTx 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, TXxIF, 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, TXxIF, will now be set.

e) If enable bit, TXxIE, 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. 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.

**TABLE 21-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| RCSTAx | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 63 |
| TXREGx | EUSARTx Transmit Register | | | | | | | | 63 |
| TXSTAx | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 63 |
| BAUDCONx | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 65 |
| SPBRGHx | EUSARTx Baud Rate Generator Register High Byte | | | | | | | | 65 |
| SPBRGx | EUSARTx Baud Rate Generator Register Low Byte | | | | | | | | 65 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

### 21.4.2 EUSARTx 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 21-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR3 | SSP2IF | BCL2IF | RC2IF | TX2IF | TMR4IF | CCP5IF | CCP4IF | CCP3IF | 64 |
| PIE3 | SSP2IE | BCL2IE | RC2IE | TX2IE | TMR4IE | CCP5IE | CCP4IE | CCP3IE | 64 |
| IPR3 | SSP2IP | BCL2IP | RC2IP | TX2IP | TMR4IP | CCP5IP | CCP4IP | CCP3IP | 64 |
| RCSTAx | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 63 |
| RCREGx | EUSARTx Receive Register | | | | | | | | 63 |
| TXSTAx | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 63 |
| BAUDCONx | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 65 |
| SPBRGHx | EUSARTx Baud Rate Generator Register High Byte | | | | | | | | 65 |
| SPBRGx | EUSARTx Baud Rate Generator Register Low Byte | | | | | | | | 65 |

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.

NOTES:

## 22.0 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 22-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 22-2, configures the A/D clock source, programmed acquisition time and justification.

### REGISTER 22-1: ADCON0: A/D CONTROL REGISTER 0[1]

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| VCFG1 | VCFG0 | CHS3 | CHS2 | CHS1 | CHS0 | GO/DONE | ADON |
| 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 **VCFG1:** Voltage Reference Configuration bit (VREF- source)
1 = VREF- (AN2)
0 = AVSS

bit 6 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)
1 = VREF+ (AN3)
0 = AVDD

bit 5-2 **CHS<3:0>:** 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 is in progress
0 = A/D is 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 22-2: ADCON1: A/D CONTROL REGISTER 1[(1)]

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADFM | ADCAL | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 |
| 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 **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 the next A/D conversion
0 = Normal A/D Converter operation (no conversion is performed)

bit 5-3 **ACQT<2:0>:** 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

bit 2-0 **ADCS<2:0>:** 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 PCFGx 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 6.3.4.1 "Shared Address SFRs"** for more information.

### REGISTER 22-3: ANCON0: A/D PORT CONFIGURATION REGISTER 0

| R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PCFG7 | PCFG6 | — | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 |
| 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 7-6 **PCFG<7:6>:** Analog Port Configuration bits (AN7 and AN6)
1 = Pin is configured as a digital port
0 = Pin is configured as an analog channel; digital input is disabled and reads '0'

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

bit 4-0 **PCFG<4:0>:** Analog Port Configuration bits (AN4 through AN0)
1 = Pin is configured as a digital port
0 = Pin is configured as an analog channel; digital input is disabled and reads '0'

### REGISTER 22-4: ANCON1: A/D PORT CONFIGURATION REGISTER 1

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| PCFG15[1] | PCFG14[1] | PCFG13[1] | PCFG12[1] | PCFG11 | PCFG10 | PCFG9 | PCFG8 |
| 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 **PCFG<15:8>:** Analog Port Configuration bits (AN15 through AN8)[1]
1 = Pin is configured as a digital port
0 = Pin is configured as an analog channel; digital input is 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.

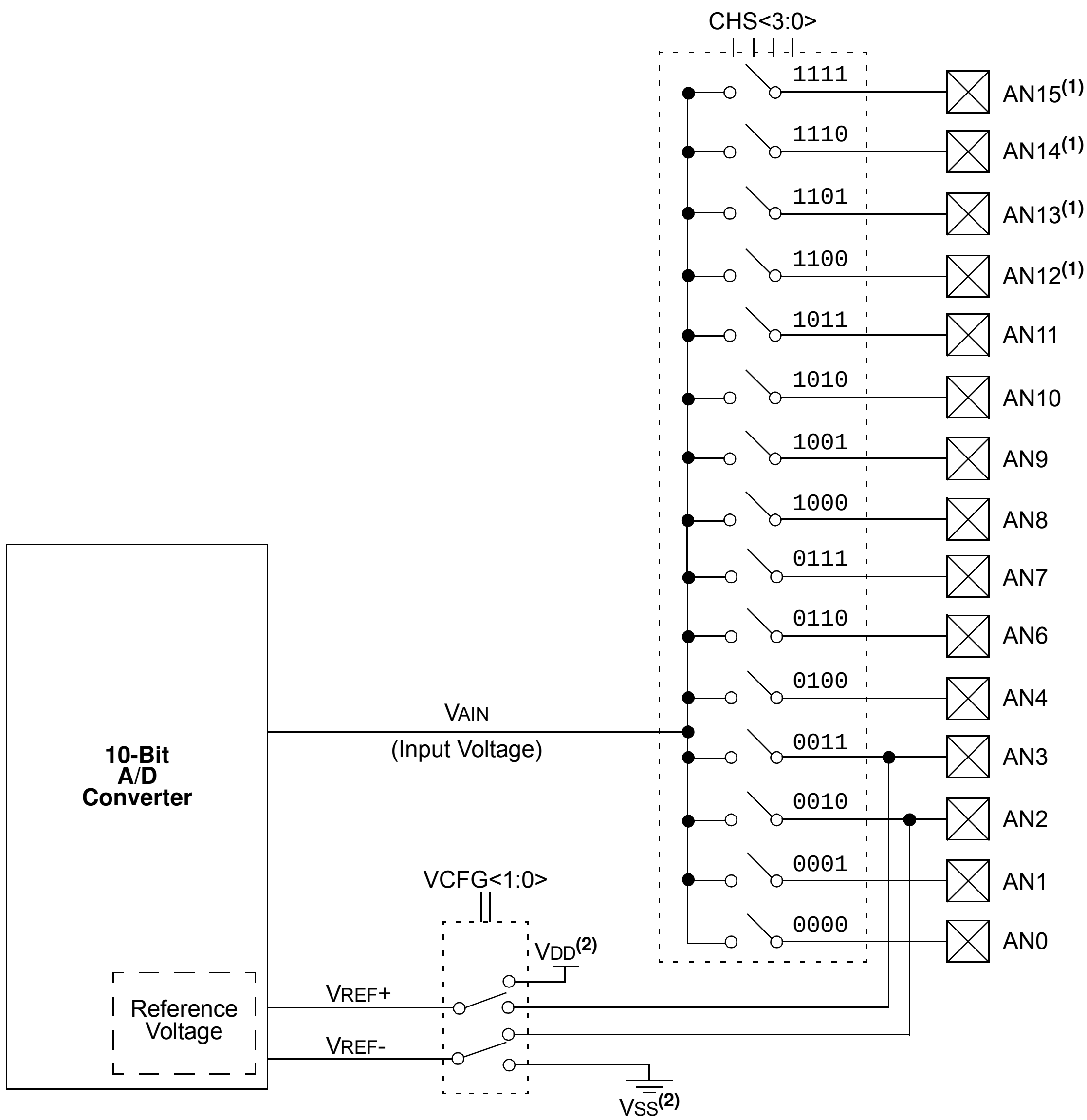
The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

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/$\overline{DONE}$ 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 22-1.

**FIGURE 22-1: A/D BLOCK DIAGRAM**




**Note 1:** Channels AN15 through AN12 are not available on 64-pin devices.
**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 22.1 "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/$\overline{\text{DONE}}$ bit and the actual start of the conversion.

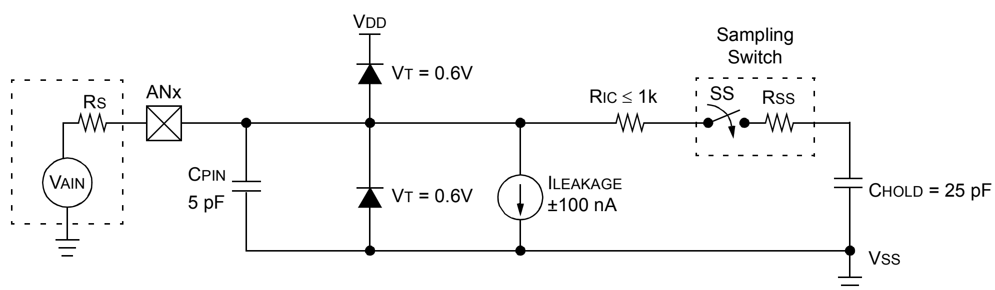
The following steps should be followed to do an A/D conversion:

1. Configure the A/D module:
   - Configure the required A/D 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/$\overline{\text{DONE}}$ bit (ADCON0<1>)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

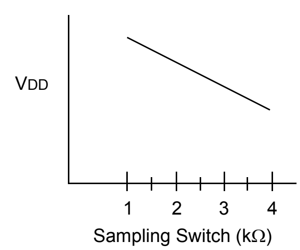
   OR

   - 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 22-2: ANALOG INPUT MODEL**




| | | |
|---|---|---|
| **Legend:** | CPIN | = Input Capacitance |
| | VT | = Threshold Voltage |
| | ILEAKAGE | = Leakage Current at the pin due to various junctions |
| | RIC | = Interconnect Resistance |
| | SS | = Sampling Switch |
| | CHOLD | = Sample/Hold Capacitance (from DAC) |
| | RSS | = Sampling Switch Resistance |



## 22.1 A/D Acquisition Requirements

For the A/D Converter to meet its specified accuracy, the charge holding capacitor ($C_{HOLD}$) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 22-2. The source impedance ($R_S$) and the internal sampling switch ($R_{SS}$) impedance directly affect the time required to charge the capacitor $C_{HOLD}$. The sampling switch ($R_{SS}$) impedance varies over the device voltage ($V_{DD}$). The 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 22-1 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 22-3 shows the calculation of the minimum required acquisition time, $T_{ACQ}$. This calculation is based on the following application system assumptions:

| | | |
|---|---|---|
| $C_{HOLD}$ | = | 25 pF |
| Rs | = | 2.5 kΩ |
| Conversion Error | ≤ | 1/2 LSb |
| $V_{DD}$ | = | 3V → Rss = 2 kΩ |
| Temperature | = | 85°C (system max.) |

**EQUATION 22-1: ACQUISITION TIME**

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$

**EQUATION 22-2: A/D MINIMUM CHARGING TIME**

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

or

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2048)$$

**EQUATION 22-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME**

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{AMP} = 0.2\ \mu s$$

$$T_{COFF} = (\text{Temp} - 25°C)(0.02\ \mu s/°C)$$
$$(85°C - 25°C)(0.02\ \mu s/°C)$$
$$1.2\ \mu s$$

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, $T_{COFF}$ = 0 ms.

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2048)\ \mu s$$
$$-(25\text{ pF})(1\text{ k}\Omega + 2\text{ k}\Omega + 2.5\text{ k}\Omega)\ln(0.0004883)\ \mu s$$
$$1.05\ \mu s$$

$$T_{ACQ} = 0.2\ \mu s + 1.05\ \mu s + 1.2\ \mu s$$
$$2.45\ \mu s$$

## 22.2 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 ACQT<2:0> 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 ACQTx 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.

## 22.3 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 28-31 for more information).

Table 22-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

**TABLE 22-1: TAD vs. DEVICE OPERATING FREQUENCIES**

| AD Clock Source (TAD) | | Maximum Device Frequency |
|---|---|---|
| **Operation** | **ADCS<2:0>** | |
| 2 TOSC | 000 | 2.86 MHz |
| 4 TOSC | 100 | 5.71 MHz |
| 8 TOSC | 001 | 11.43 MHz |
| 16 TOSC | 101 | 22.86 MHz |
| 32 TOSC | 010 | 40.00 MHz |
| 64 TOSC | 110 | 40.00 MHz |
| RC[2] | x11 | 1.00 MHz[1] |

**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.

## 22.4 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 CHS<3:0> 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.

## 22.5 A/D Conversions

Figure 22-3 shows the operation of the A/D Converter after the GO/DONE bit has been set and the ACQT<2:0> bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 22-4 shows the operation of the A/D Converter after the GO/DONE bit has been set, the ACQT<2:0> 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. |
|---|---|

## 22.6 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 CCP2M<3:0> 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 22-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**

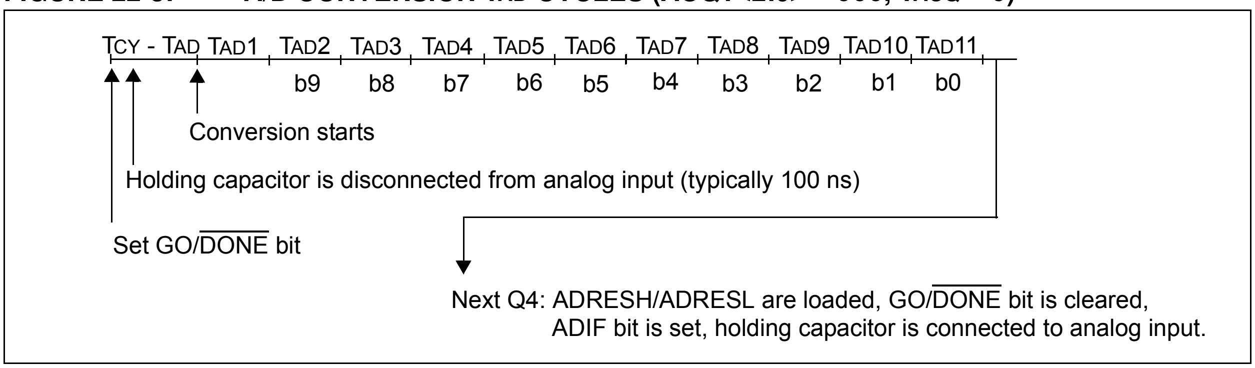
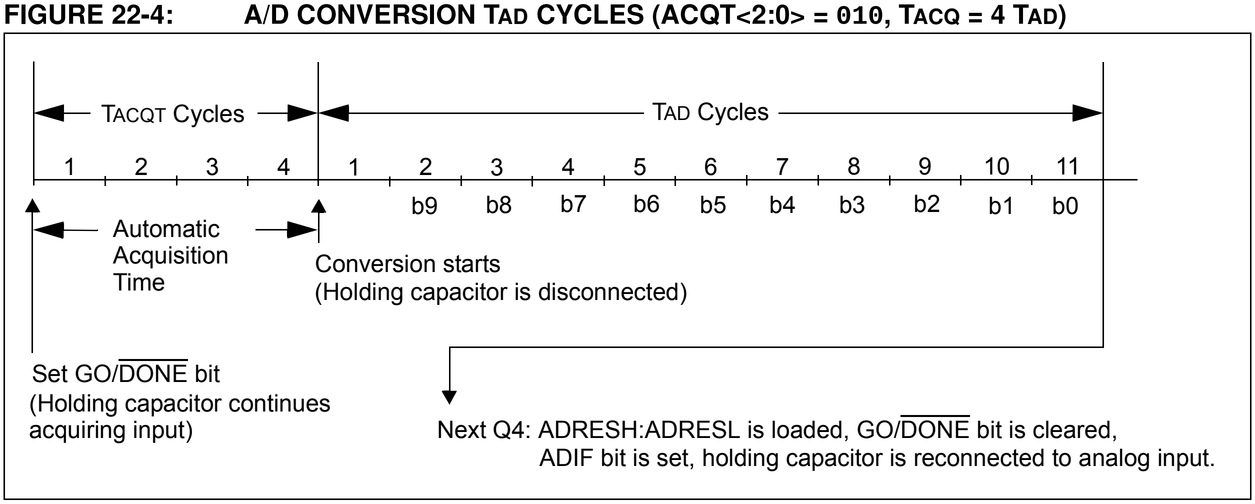



**FIGURE 22-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**



## 22.7 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/$\overline{\text{DONE}}$ 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 22-1.

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.

## 22.8 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 ACQT<2:0> and ADCS<2:0> 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 Sleep mode requires the A/D RC clock to be selected. If bits, ACQT<2:0>, 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 SCSx bits in the OSCCON register must have already been cleared prior to starting the conversion.

**EXAMPLE 22-1: 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,ADCAL        ;Enable Calibration
        BSF     ADCON0,GO           ;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 22-2: SUMMARY OF A/D REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR1 | PMPIF | ADIF | RC1IF | TX1IF | SSP1IF | CCP1IF | TMR2IF | TMR1IF | 64 |
| PIE1 | PMPIE | ADIE | RC1IE | TX1IE | SSP1IE | CCP1IE | TMR2IE | TMR1IE | 64 |
| IPR1 | PMPIP | ADIP | RC1IP | TX1IP | SSP1IP | CCP1IP | TMR2IP | TMR1IP | 64 |
| PIR2 | OSCFIF | CM2IF | CM1IF | — | BCL1IF | LVDIF | TMR3IF | CCP2IF | 64 |
| PIE2 | OSCFIE | CM2IE | CM1IE | — | BCL1IE | LVDIE | TMR3IE | CCP2IE | 64 |
| IPR2 | OSCFIP | CM2IP | CM1IP | — | BCL1IP | LVDIP | TMR3IP | CCP2IP | 64 |
| ADRESH | A/D Result Register High Byte | | | | | | | | 63 |
| ADRESL | A/D Result Register Low Byte | | | | | | | | 63 |
| ADCON0(2) | VCFG1 | VCFG0 | CHS3 | CHS3 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON | 63 |
| ANCON0(3) | PCFG7 | PCFG6 | — | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 63 |
| ADCON1(2) | ADFM | ADCAL | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 63 |
| ANCON1(3) | PCFG15 | PCFG14 | PCFG13 | PCFG12 | PCFG11 | PCFG10 | PCFG9 | PCFG8 | 63 |
| CCP2CON | P2M1 | P2M0 | DC2B1 | DC2B0 | CCP2M3 | CCP2M2 | CCP2M1 | CCP2M0 | 63 |
| PORTA | RA7(4) | RA6(4) | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | 65 |
| TRISA | TRISA7(4) | TRISA6(4) | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 64 |
| PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | — | 65 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — | 64 |
| PORTH(1) | RH7 | RH6 | RH5 | RH4 | RH3 | RH2 | RH1 | RH0 | 65 |
| TRISH(1) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 | 64 |

**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.

**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:** These bits are only available in select oscillator modes (FOSC2 Configuration bit = 0); otherwise, they are unimplemented.

## 23.0 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 23-1.

Key features of the module includes:

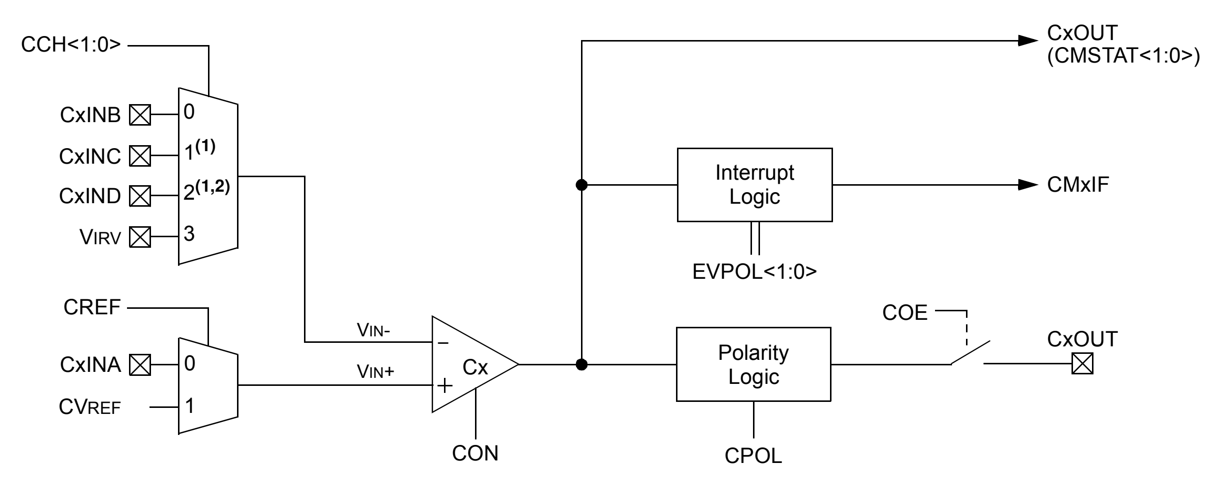
- Independent comparator control
- Programmable input configuration
- Output to both pin and register levels
- Programmable output polarity
- Independent interrupt generation for each comparator with configurable interrupt-on-change

## 23.1 Registers

The CMxCON registers (Register 23-1) select the input and output configuration for each comparator, as well as the settings for interrupt generation.

The CMSTAT register (Register 23-2) provides the output results of the comparators. The bits in this register are read-only.

**FIGURE 23-1: COMPARATOR SIMPLIFIED BLOCK DIAGRAM**




**Note 1:** Available in 80-pin devices only.

**2:** Implemented in Comparator 2 only.

## REGISTER 23-1: CMxCON: COMPARATORx CONTROL REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 |
| 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 **CON:** Comparator Enable bit
1 = Comparator is enabled
0 = Comparator is disabled

bit 6 **COE:** Comparator Output Enable bit
1 = Comparator output is present on the CxOUT pin
0 = Comparator output is internal only

bit 5 **CPOL:** Comparator Output Polarity Select bit
1 = Comparator output is inverted
0 = Comparator output is not inverted

bit 4-3 **EVPOL<1:0>:** Interrupt Polarity Select bits
11 = Interrupt generation on any change of the output(1)
10 = Interrupt generation only on high-to-low transition of the output
01 = Interrupt generation only on low-to-high transition of the output
00 = Interrupt generation is disabled

bit 2 **CREF:** Comparator Reference Select bit (non-inverting input)
1 = Non-inverting input connects to internal CVREF voltage
0 = Non-inverting input connects to CxINA pin

bit 1-0 **CCH<1:0>:** Comparator Channel Select bits
11 = Inverting input of comparator connects to VIRV
10 = Inverting input of comparator connects to CxIND pin(2)
01 = Inverting input of comparator connects to CxINC pin(2)
00 = Inverting input of comparator connects to CxINB pin

**Note 1:** The CMxIF bit 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 23-2: CMSTAT: COMPARATOR OUTPUT STATUS REGISTER

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R-1 | R-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | COUT2 | COUT1 |
| 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 |

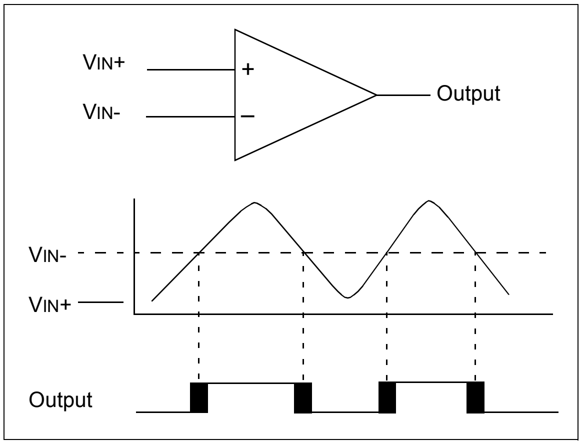
bit 7-2 **Unimplemented:** Read as ‘0’

bit 1-0 **COUT<2:1>:** Comparator x Status bits
If CPOL = 0 (non-inverted polarity):
1 = Comparator’s $V_{IN}+$ > $V_{IN}-$
0 = Comparator’s $V_{IN}+$ < $V_{IN}-$
If CPOL = 1 (inverted polarity):
1 = Comparator $V_{IN}+$ < $V_{IN}-$
0 = Comparator $V_{IN}+$ > $V_{IN}-$

## 23.2 Comparator Operation

A single comparator is shown in Figure 23-2, 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 23-2 represent the uncertainty due to input offsets and response time.

**FIGURE 23-2: SINGLE COMPARATOR**



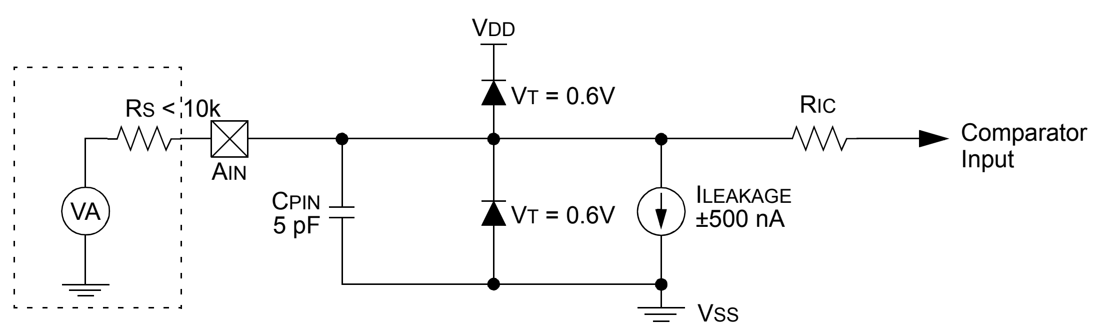

## 23.3 Comparator Response Time

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 28.0 "Electrical Characteristics"**).

## 23.4 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 23-3. 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 23-3: 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 |

## 23.5 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 23-1. The available configurations and their corresponding bit settings are shown in Figure 23-1.

**TABLE 23-1: COMPARATOR INPUTS AND OUTPUTS**

| Comparator | Input or Output | I/O Pin |
|---|---|---|
| 1 | C1INA (VIN+) | RF6 |
| | C1INB (VIN-) | RF5 |
| | C1INC (VIN-)(1) | RH6(1) |
| | C1OUT | RF2 |
| 2 | C2INA(VIN+) | RF4 |
| | C2INB(VIN-) | RF3 |
| | C2INC(VIN-)(1) | RH4(1) |
| | C2IND(VIN-)(1) | RH5(1) |
| | C2OUT | RF1 |

**Note 1:** Available in 80-pin devices only.

### 23.5.1 COMPARATOR ENABLE AND INPUT SELECTION

Setting the CON bit of the CMxCON register (CMxCON<7>) enables the comparator for operation. Clearing the CON bit disables the comparator resulting in minimum current consumption.

The CCH<1:0> bits in the CMxCON register (CMxCON<1:0>) direct either one of three analog input pins, or the Internal Reference Voltage (VIRV), to the comparator VIN-. Depending on the comparator operating mode, either an external or internal voltage reference may be used. The analog signal present at VIN- is compared to the signal at VIN+ and the digital output of the comparator is adjusted accordingly.

The external reference is used when CREF (CMxCON<2>) = `0` and VIN+ is connected to the CxINA pin. When external voltage references are used, the comparator module can be configured to have the reference sources externally. The reference signal must be between VSS and VDD, and can be applied to either pin of the comparator.

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 24.0 "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 CCH<1:0>, must be configured as an input by setting both the corresponding TRISF or TRISH bit, and the corresponding PCFGx bit in the ANCON1 register. |
|---|---|

#### 23.5.1.1 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 23-1 are footnoted to indicate where they are not available.

### 23.5.2 COMPARATOR ENABLE AND OUTPUT SELECTION

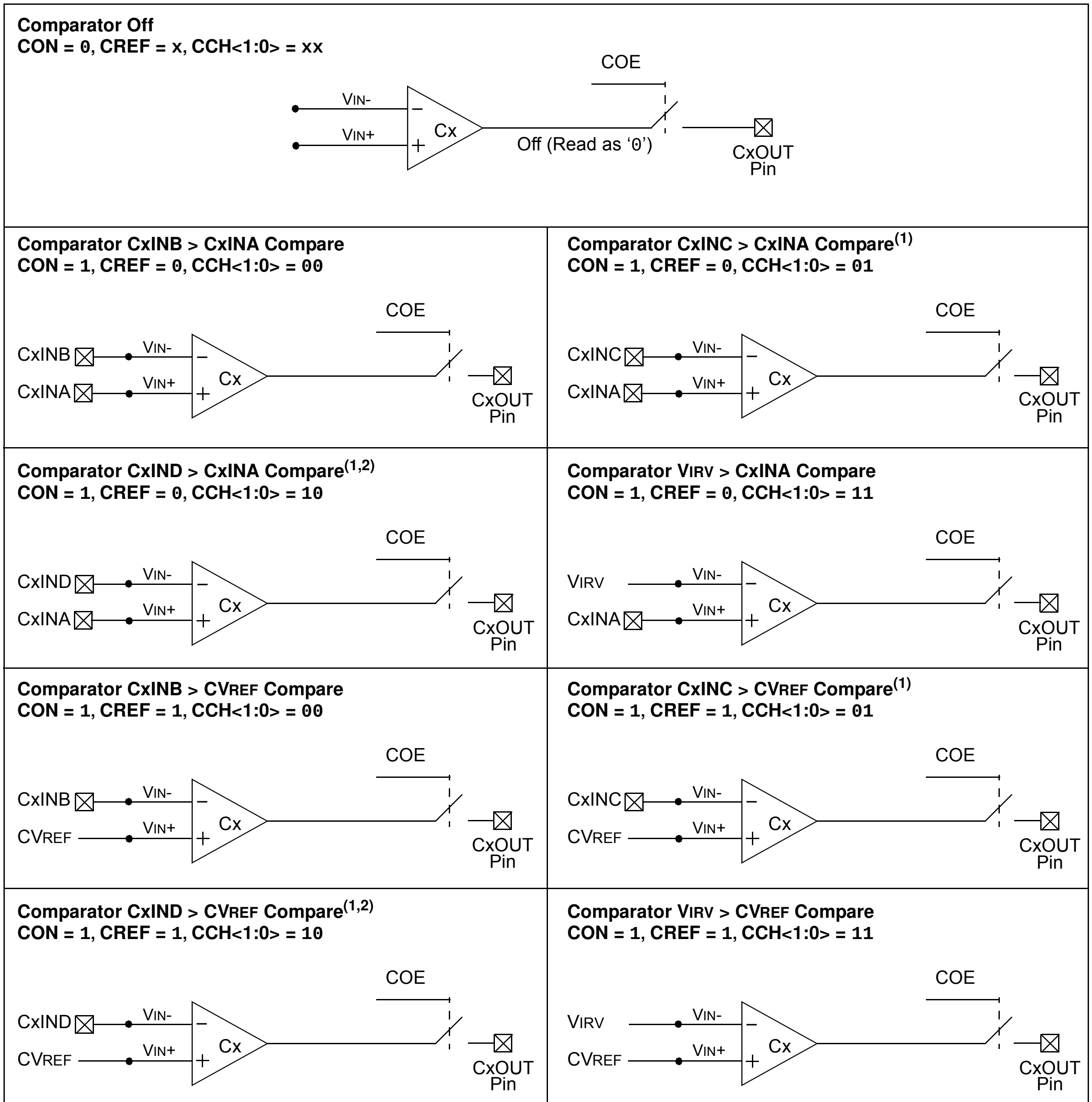
The comparator outputs are read through the CMSTAT register. The CMSTAT<0> reads the Comparator 1 output and CMSTAT<1> reads the Comparator 2 output. These bits are read-only.

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<2:1> bits still function as the digital output enable bits 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 23.2 "Comparator Operation"**.

**FIGURE 23-4: COMPARATOR I/O CONFIGURATIONS**




**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.

**2:** Configuration is available in Comparator 2 only (80-pin devices).

## 23.6 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 EVPOL<1:0> 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 23-2 shows the interrupt generation with respect to comparator input voltages and EVPOLx 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 23-2: COMPARATOR INTERRUPT GENERATION**

| CPOL | EVPOL<1:0> | Comparator Input Change | CxOUT Transition | Interrupt Generated |
|---|---|---|---|---|
| 0 | 00 | VIN+ > VIN- | Low-to-High | No |
| | | VIN+ < VIN- | High-to-Low | No |
| | 01 | VIN+ > VIN- | Low-to-High | Yes |
| | | VIN+ < VIN- | High-to-Low | No |
| | 10 | VIN+ > VIN- | Low-to-High | No |
| | | VIN+ < VIN- | High-to-Low | Yes |
| | 11 | VIN+ > VIN- | Low-to-High | Yes |
| | | VIN+ < VIN- | High-to-Low | Yes |
| 1 | 00 | VIN+ > VIN- | High-to-Low | No |
| | | VIN+ < VIN- | Low-to-High | No |
| | 01 | VIN+ > VIN- | High-to-Low | No |
| | | VIN+ < VIN- | Low-to-High | Yes |
| | 10 | VIN+ > VIN- | High-to-Low | Yes |
| | | VIN+ < VIN- | Low-to-High | No |
| | 11 | VIN+ > VIN- | High-to-Low | Yes |
| | | VIN+ < VIN- | Low-to-High | Yes |

## 23.7 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.

## 23.8 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 23-3: REGISTERS ASSOCIATED WITH COMPARATOR MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 61 |
| PIR2 | OSCFIF | CM2IF | CM1IF | — | BCL1IF | LVDIF | TMR3IF | CCP2IF | 64 |
| PIE2 | OSCFIE | CM2IE | CM1IE | — | BCL1IE | LVDIE | TMR3IE | CCP2IE | 64 |
| IPR2 | OSCFIP | CM2IP | CM1IP | — | BCL1IP | LVDIP | TMR3IP | CCP2IP | 64 |
| CM1CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 | 62 |
| CM2CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 | 62 |
| CMSTAT | — | — | — | — | — | — | COUT2 | COUT1 | 62 |
| CVRCON(2) | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 65 |
| ANCON1(2) | PCFG15 | PCFG14 | PCFG13 | PCFG12 | PCFG11 | PCFG10 | PCFG9 | PCFG8 | 63 |
| ANCON0(2) | PCFG7 | PCFG6 | — | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 63 |
| PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | — | 65 |
| LATF | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | — | 64 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — | 64 |
| PORTH(1) | RH7 | RH6 | RH5 | RH4 | RH3 | RH2 | RH1 | RH0 | 65 |
| TRISH(1) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 | 64 |

**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.

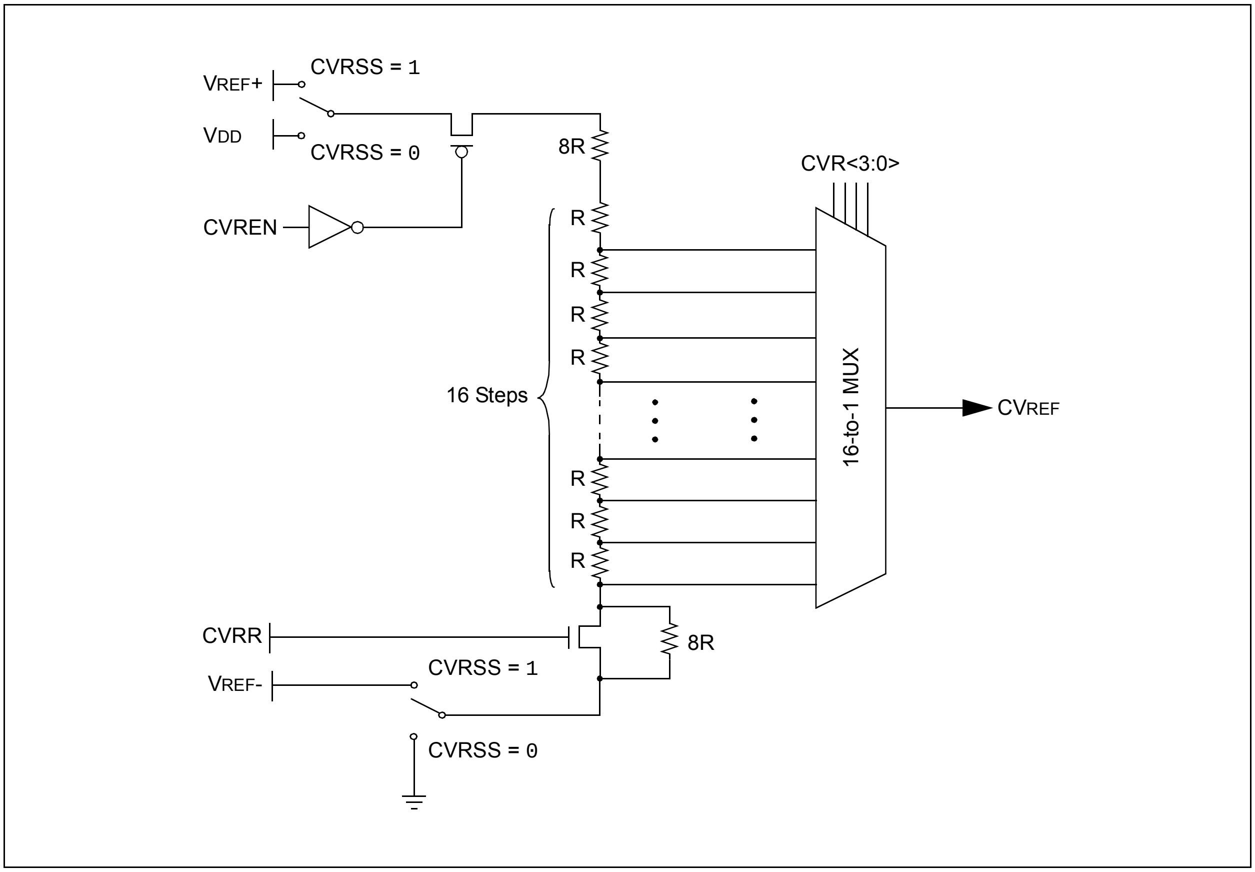
**2:** Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = `1`.

## 24.0 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 24-1. 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.

**FIGURE 24-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM**



## 24.1 Configuring the Comparator Voltage Reference

The comparator voltage reference module is controlled through the CVRCON register (Register 24-1). 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 (CVR<3:0>), 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 = ((CVR<3:0>)/24) x (CVRSRC)

If CVRR = 0:
CVREF = (CVRSRC/4) + ((CVR<3:0>)/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 28-3 in **Section 28.0 "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 24-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL REGISTER

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CVREN | CVROE[1] | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 |
| 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 **CVREN:** Comparator Voltage Reference Enable bit
1 = CVREF circuit is powered on
0 = CVREF circuit is 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 **CVR<3:0>:** Comparator VREF Value Selection bits (0 ≤ (CVR3:CVR0) ≤ 15)
When CVRR = 1:
CVREF = ((CVR<3:0>)/24) • (CVRSRC)
When CVRR = 0:
CVREF = (CVRSRC/4) + ((CVR<3:0>)/32) • (CVRSRC)

**Note 1:** CVROE overrides the TRISF<5> bit setting.

## 24.2 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 24-1) 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 28.0 "Electrical Characteristics"**.

## 24.3 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 24-2 shows an example buffering technique.

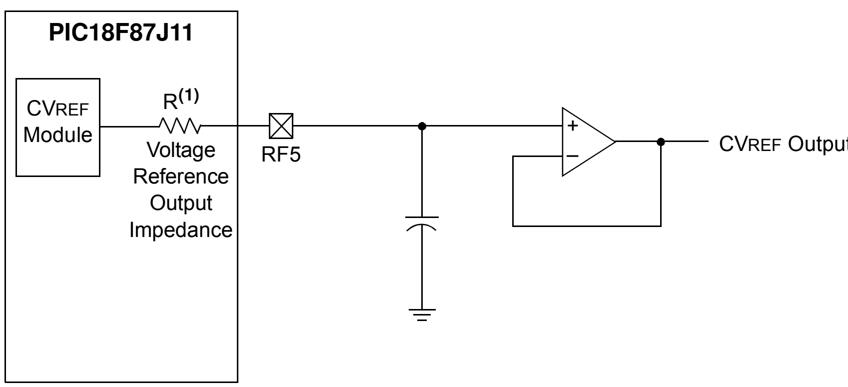
## 24.4 Operation During Sleep

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.

## 24.5 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 CVRx value select bits are also cleared.

**FIGURE 24-2: COMPARATOR VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE**




**Note 1:** R is dependent upon the comparator voltage reference configuration bits, CVRCON<5> and CVRCON<3:0>.

**TABLE 24-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| CVRCON(2) | CVREN | CVROE | CVRR | CVRSS | CVR3 | CVR2 | CVR1 | CVR0 | 65 |
| CM1CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 | 62 |
| CM2CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 | 62 |
| TRISA | TRISA7(1) | TRISA6(1) | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 64 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — | 64 |
| ANCON0(2) | PCFG7 | PCFG6 | — | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 63 |
| ANCON1(2) | PCFG15 | PCFG14 | PCFG13 | PCFG12 | PCFG11 | PCFG10 | PCFG9 | PCFG8 | 63 |

**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.

**2:** Configuration SFR, overlaps with default SFR at this address; available only when WDTCON<4> = 1.

NOTES:

# 25.0 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 3.0 "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.

## 25.1 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 25-2. A detailed explanation of the various bit functions is provided in Register 25-1 through Register 25-6.

### 25.1.1 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 25-2, 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 or after any device Reset.

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 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 25-1: MAPPING OF THE FLASH CONFIGURATION WORDS TO THE CONFIGURATION REGISTERS**

| Configuration Byte | Code Space Address | Configuration Register Address |
|---|---|---|
| CONFIG1L | XXXF8h | 300000h |
| CONFIG1H | XXXF9h | 300001h |
| CONFIG2L | XXXFAh | 300002h |
| CONFIG2H | XXXFBh | 300003h |
| CONFIG3L | XXXFCh | 300004h |
| CONFIG3H | XXXFDh | 300005h |

**TABLE 25-2: CONFIGURATION BITS AND DEVICE IDs**

| File Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Default/ Unprogrammed Value(1) |
|---|---|---|---|---|---|---|---|---|---|---|
| 300000h | CONFIG1L | $\overline{\text{DEBUG}}$ | XINST | STVREN | — | — | — | — | WDTEN | 111- ---1 |
| 300001h | CONFIG1H | —(2) | —(2) | —(2) | —(2) | — | CP0 | — | — | 1111 -111 |
| 300002h | CONFIG2L | IESO | FCMEN | — | — | — | FOSC2 | FOSC1 | FOSC0 | 11-- -111 |
| 300003h | CONFIG2H | —(2) | —(2) | —(2) | —(2) | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | 1111 1111 |
| 300004h | CONFIG3L | WAIT(3) | BW(3) | EMB1(3) | EMB0(3) | EASHFT(3) | — | — | — | 1111 1--- |
| 300005h | CONFIG3H | —(2) | —(2) | —(2) | —(2) | MSSPMSK | PMPMX(3) | ECCPMX(3) | CCP2MX | 1111 1111 |
| 3FFFFEh | DEVID1 | DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 | xxx0 0000(4) |
| 3FFFFFh | DEVID2 | DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 | 0100 00xx(4) |

**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:** These bits are implemented in 80-pin devices only.

**4:** See Register 25-7 and Register 25-8 for DEVID values. These registers are read-only and cannot be programmed by the user.

**REGISTER 25-1: CONFIG1L: CONFIGURATION REGISTER 1 LOW (BYTE ADDRESS 300000h)**

| R/WO-1 | R/WO-1 | R/WO-1 | U-0 | U-0 | U-0 | U-0 | R/WO-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{DEBUG}}$ | XINST | STVREN | — | — | — | — | WDTEN |
| bit 7 | | | | | | | bit 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 |

bit 7 **$\overline{\text{DEBUG}}$:** Background Debugger Enable bit
1 = Background debugger is disabled; RB6 and RB7 are configured as general purpose I/O pins
0 = Background debugger is enabled; RB6 and RB7 are dedicated to In-Circuit Debug

bit 6 **XINST:** Extended Instruction Set Enable bit
1 = Instruction set extension and Indexed Addressing mode are enabled
0 = Instruction set extension and Indexed Addressing mode are disabled (Legacy mode)

bit 5 **STVREN**: Stack Overflow/Underflow Reset Enable bit
1 = Reset on stack overflow/underflow is enabled
0 = Reset on stack overflow/underflow is disabled

bit 4-1 **Unimplemented:** Read as ‘0’

bit 0 **WDTEN:** Watchdog Timer Enable bit
1 = WDT is enabled
0 = WDT is disabled (control is placed on the SWDTEN bit)

**REGISTER 25-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)**

| U-1 | U-1 | U-1 | U-1 | U-0 | R/WO-1 | U-1 | U-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | CP0 | — | — |
| bit 7 | | | | | | | bit 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 |

bit 7-3 **Unimplemented:** Maintain as ‘11110’

bit 2 **CP0:** Code Protection bit
1 = Program memory is not code-protected
0 = Program memory is code-protected

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

**REGISTER 25-3: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)**

| R/WO-1 | R/WO-1 | U-0 | U-0 | U-0 | R/WO-1 | R/WO-1 | R/WO-1 |
|---|---|---|---|---|---|---|---|
| IESO | FCMEN | — | — | — | FOSC2 | FOSC1 | FOSC0 |
| bit 7 | | | | | | | bit 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 |

bit 7 **IESO:** Two-Speed Start-up (Internal/External Oscillator Switchover) Control bit
1 = Two-Speed Start-up is enabled
0 = Two-Speed Start-up is disabled

bit 6 **FCMEN:** Fail-Safe Clock Monitor Enable bit
1 = Fail-Safe Clock Monitor is enabled
0 = Fail-Safe Clock Monitor is disabled

bit 5-3 **Unimplemented:** Read as ‘0’

bit 2-0 **FOSC<2:0>:** Oscillator Selection bits
111 = EC oscillator with PLL enabled; CLKO on RA6 (ECPLL)
110 = EC oscillator; CLKO on RA6 (EC)
101 = HS oscillator with PLL enabled (HSPLL)
100 = HS oscillator (HS)
011 = Internal oscillator with PLL enabled; CLKO on RA6, port function on RA7 (INTPLL1)
010 = Internal oscillator with PLL enabled; port function on RA6 and RA7 (INTPLL2)
001 = Internal oscillator block; CLKO on RA6, port function on RA7 (INTIO1)
000 = Internal oscillator block; port function on RA6 and RA7 (INTIO2)

**REGISTER 25-4: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)**

| U-1 | U-1 | U-1 | U-1 | R/WO-1 | R/WO-1 | R/WO-1 | R/WO-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 |
| bit 7 | | | | | | | bit 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 |

bit 7-4 **Unimplemented:** Maintain as ‘1’

bit 3-0 **WDTPS<3:0>:** 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 25-5: CONFIG3L: CONFIGURATION REGISTER 3 LOW (BYTE ADDRESS 300004h)**

| R/WO-1 | R/WO-1 | R/WO-1 | R/WO-1 | R/WO-1 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| WAIT(1) | BW(1) | EMB1(1) | EMB0(1) | EASHFT(1) | — | — | — |
| bit 7 | | | | | | | bit 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 |

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 **EMB<1:0>:** External Memory Bus Configuration bits(1)
11 = Microcontroller mode, external bus is 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 is enabled – external address bus is shifted to start at 000000h
0 = Address shifting is disabled – external address bus reflects the PC value

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

**Note 1:** These bits are implemented on 80-pin devices only.

**REGISTER 25-6: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)**

| U-1 | U-1 | U-1 | U-1 | R/WO-1 | R/WO-1 | R/WO-1 | R/WO-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | MSSPMSK | PMPMX[1] | ECCPMX[1] | CCP2MX |
| bit 7 | | | | | | | bit 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 |

bit 7-4 **Unimplemented:** Maintain as '1'

bit 3 **MSSPMSK:** MSSP Address Masking Mode Select bit
1 = 7-Bit Address Masking mode is enabled
0 = 5-Bit Address Masking mode is enable

bit 2 **PMPMX:** PMP Pin Multiplex bit[1]
1 = PMP data and control are multiplexed to the same pins as the External Memory Bus (PORTD and PORTE)
0 = PMP data and control are multiplexed to alternate pin assignments (PORTA, PORTF and PORTH)

bit 1 **ECCPMX:** ECCPx 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:** These bits are implemented on 80-pin devices only.

### REGISTER 25-7: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F87J11 FAMILY DEVICES

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 |
| 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 **DEV<2:0>:** Device ID bits
See Register 25-8 for a complete listing.

bit 4-0 **REV<4:0>:** Revision ID bits
These bits are used to indicate the device revision.

### REGISTER 25-8: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F87J11 FAMILY DEVICES

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 |
| 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 **DEV<10:3>:** Device ID bits:

| DEV<10:3> (DEVID2<7:0>) | DEV<2:0> (DEVID1<7:5>) | Device |
|---|---|---|
| 0100 0100 | 010 | PIC18F66J11 |
| 0100 0100 | 011 | PIC18F66J16 |
| 0100 0100 | 100 | PIC18F67J11 |
| 0100 0100 | 111 | PIC18F86J11 |
| 0100 0101 | 000 | PIC18F86J16 |
| 0100 0101 | 001 | PIC18F87J11 |

## 25.2 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 WDTPSx 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.

> **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.

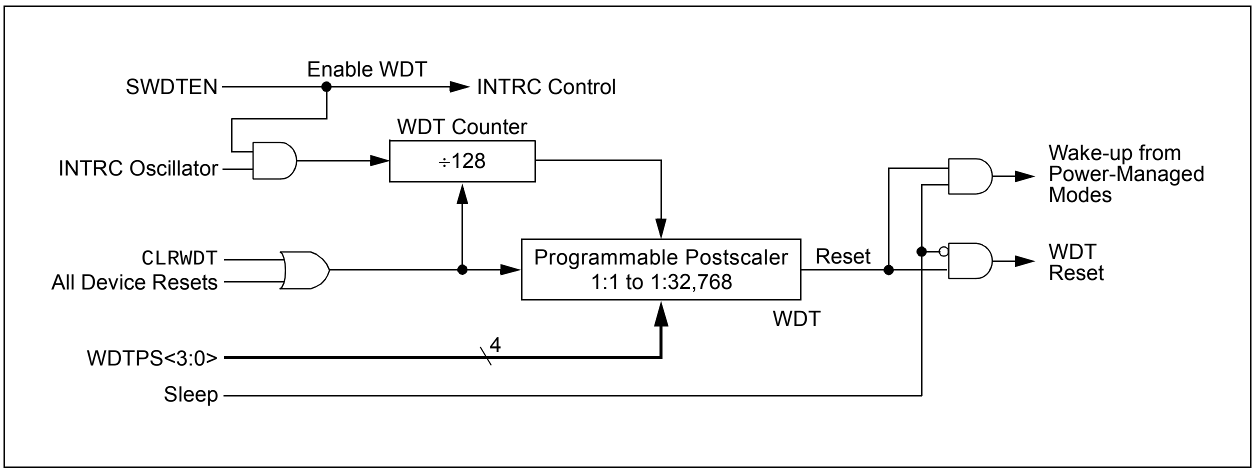
### 25.2.1 CONTROL REGISTER

The WDTCON register (Register 25-9) is a readable and writable register. The SWDTEN bit enables or disables WDT operation. This allows software to override the WDTEN 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 6.3.4.1 "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 $V_{DDCORE}$. This bit is only valid when the on-chip voltage regulator is enabled.

**FIGURE 25-1: WDT BLOCK DIAGRAM**



### REGISTER 25-9: WDTCON: WATCHDOG TIMER CONTROL REGISTER

| R/W-0 | R-x | U-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| REGSLP | LVDSTAT | — | ADSHR | — | — | — | SWDTEN[1] |
| 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 **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 = $V_{DDCORE}$ > 2.45V
0 = $V_{DDCORE}$ < 2.45V

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

bit 4 **ADSHR:** Shared Address SFR Select bit
For details of bit operation, see Register 6-3.

bit 3-1 **Unimplemented**: Read as '0'

bit 0 **SWDTEN:** Software Controlled Watchdog Timer Enable bit[1]
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 25-3: SUMMARY OF WATCHDOG TIMER REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page: |
|---|---|---|---|---|---|---|---|---|---|
| RCON | IPEN | — | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 62 |
| WDTCON | REGSLP | LVDSTAT | — | ADSHR | — | — | — | SWDTEN | 63 |

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

## 25.3 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 25-2). This helps to maintain the stability of the regulator. The recommended value for the filter capacitor is provided in **Section 28.3 "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 25-2 for possible configurations.

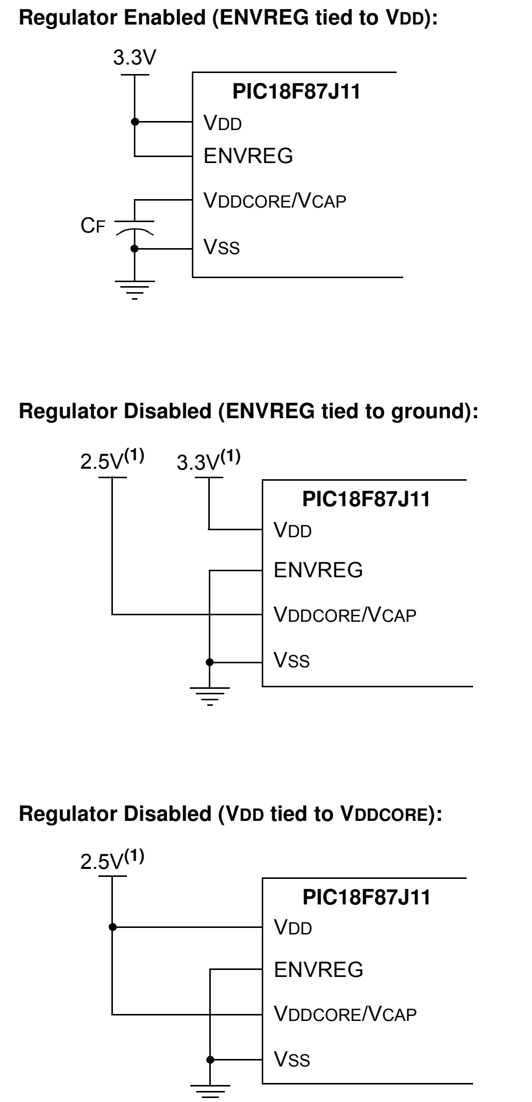
### 25.3.1 VOLTAGE REGULATOR TRACKING MODE AND LOW-VOLTAGE DETECTION

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.

**FIGURE 25-2: CONNECTIONS FOR THE ON-CHIP REGULATOR**




**Note 1:** These are typical operating voltages. Refer to **Section 28.1 "DC Characteristics: Supply Voltage"** for the full operating ranges of VDD and VDDCORE.

### 25.3.2 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 $\overline{BOR}$ flag bit (RCON<0>).

The operation of the Brown-out Reset is described in more detail in **Section 5.4 "Brown-out Reset (BOR)"** and **Section 5.4.1 "Detecting BOR"**. The brown-out voltage levels are specific in **Section 28.1 "DC Characteristics: Supply Voltage PIC18F87J11 Family (Industrial)"**.

### 25.3.3 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, $V_{DDCORE}$ must never exceed $V_{DD}$ by 0.3 volts.

### 25.3.4 OPERATION IN SLEEP MODE

When enabled, the on-chip regulator always consumes a small incremental amount of current over $I_{DD}$. 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 25-9). 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.

## 25.4 Two-Speed Start-up

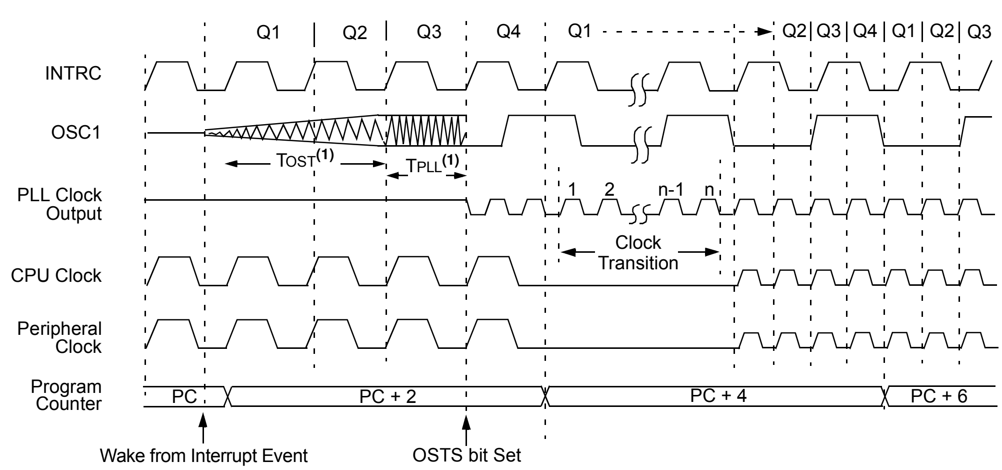
The Two-Speed Start-up feature helps to minimize the latency period, from oscillator start-up to code execution, by allowing the microcontroller to use the INTRC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO Configuration bit.

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 25-3: TIMING TRANSITION FOR TWO-SPEED START-UP (INTRC TO HSPLL)**




**Note 1:** $T_{OST}$ = 1024 $T_{OSC}$; $T_{PLL}$ = 2 ms (approx). These intervals are not shown to scale.

### 25.4.1 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 4.1.4 "Multiple Sleep Commands"**). In practice, this means that user code can change the SCS<1:0> 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.

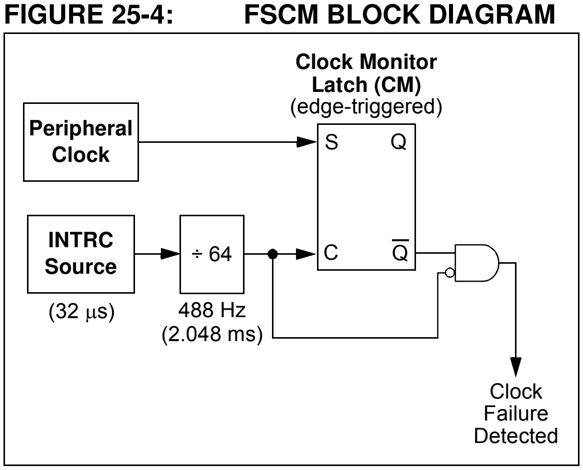
User code can also check if the primary clock source is currently providing the device clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

## 25.5 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 25-4) 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.

**FIGURE 25-4: FSCM BLOCK DIAGRAM**




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 25-5). 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)
- 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 4.1.4 "Multiple Sleep Commands"** and **Section 25.4.1 "Special Considerations for Using Two-Speed Start-up"** for more details.

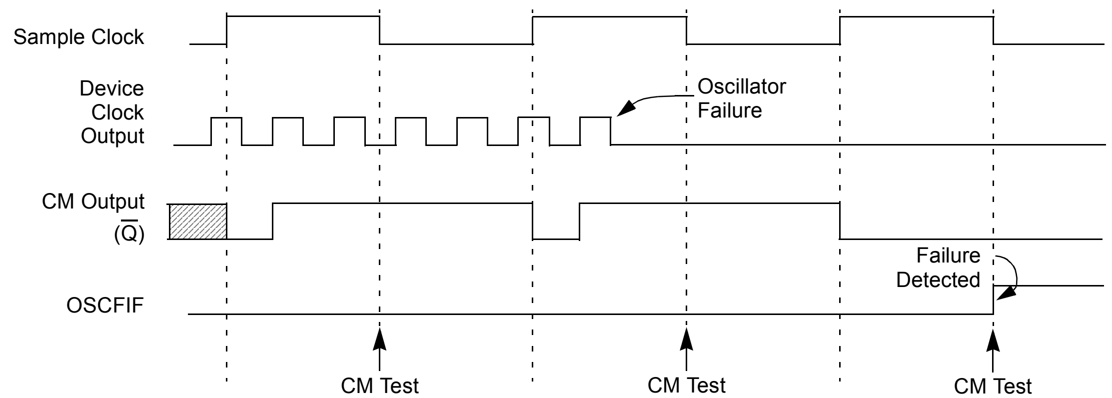
The FSCM will detect failures of the primary or secondary clock sources only. If the internal oscillator block fails, no failure would be detected, nor would any action be possible.

### 25.5.1 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.

**FIGURE 25-5: FSCM TIMING DIAGRAM**




**Note:** The device clock is normally at a much higher frequency than the sample clock. The relative frequencies in this example have been chosen for clarity.

### 25.5.2 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.

### 25.5.3 FSCM INTERRUPTS IN POWER-MANAGED MODES

By entering a power-managed mode, the clock multiplexor 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 multiplexor. 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.

### 25.5.4 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 25.4.1 "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.

## 25.6 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.

### 25.6.1 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 is set, the source data for device configuration is also protected as a consequence.

## 25.7 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.

## 25.8 In-Circuit Debugger

When the $\overline{\text{DEBUG}}$ 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 25-4 shows which resources are required by the background debugger.

**TABLE 25-4: DEBUGGER RESOURCES**

| | |
|---|---|
| I/O Pins: | RB6, RB7 |
| Stack: | 2 Levels |
| Program Memory: | < 1 Kbyte |
| Data Memory: | < 16 Bytes |

**NOTES:**

## 26.0 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.

### 26.1 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 26-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 26-1 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 26-1 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 26-2, lists the standard instructions recognized by the Microchip MPASM™ Assembler.

**Section 26.1.1 "Standard Instruction Set"** provides a description of each instruction.

### TABLE 26-1: OPCODE FIELD DESCRIPTIONS

| Field | Description |
|---|---|
| a | RAM access bit:<br>a = 0: RAM location in Access RAM (BSR register is ignored)<br>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: **C**arry, **D**igit **C**arry, **Z**ero, **Ov**erflow, **N**egative. |
| d | Destination select bit:<br>d = 0: store result in WREG<br>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.<br>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. |
| $\overline{PD}$ | Power-Down bit. |
| PRODH | Product of Multiply High Byte. |
| PRODL | Product of Multiply Low Byte. |
| s | Fast Call/Return mode select bit:<br>s = 0: do not update into/from shadow registers<br>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. |
| $\overline{TO}$ | 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). |

**FIGURE 26-1: GENERAL FORMAT FOR INSTRUCTIONS**

**Byte-oriented** file register operations — **Example Instruction**

| 15 10 | 9 | 8 | 7 0 |
|---|---|---|---|
| OPCODE | d | a | f (FILE #) |

Example: `ADDWF MYREG, W, B`

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)

| 15 12 | 11 0 |
|---|---|
| OPCODE | f (Source FILE #) |

| 15 12 | 11 0 |
|---|---|
| `1111` | f (Destination FILE #) |

Example: `MOVFF MYREG1, MYREG2`

f = 12-bit file register address

**Bit-oriented** file register operations

| 15 12 | 11 9 | 8 | 7 0 |
|---|---|---|---|
| OPCODE | b (BIT #) | a | f (FILE #) |

Example: `BSF MYREG, bit, B`

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

| 15 8 | 7 0 |
|---|---|
| OPCODE | k (literal) |

Example: `MOVLW 7Fh`

k = 8-bit immediate value

**Control** operations

**CALL, GOTO and Branch** operations

| 15 8 | 7 0 |
|---|---|
| OPCODE | n<7:0> (literal) |

| 15 12 | 11 0 |
|---|---|
| `1111` | n<19:8> (literal) |

Example: `GOTO Label`

n = 20-bit immediate value

| 15 8 | | 7 0 |
|---|---|---|
| OPCODE | S | n<7:0> (literal) |

| 15 12 | 11 0 |
|---|---|
| `1111` | n<19:8> (literal) |

Example: `CALL MYFUNC`

S = Fast bit

| 15 11 | 10 0 |
|---|---|
| OPCODE | n<10:0> (literal) |

Example: `BRA MYFUNC`

| 15 8 | 7 0 |
|---|---|
| OPCODE | n<7:0> (literal) |

Example: `BC MYFUNC`

**TABLE 26-2: PIC18F87J11 FAMILY INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word MSb | | | LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| **BYTE-ORIENTED OPERATIONS** | | | | | | | | | |
| ADDWF | f, d, a | Add WREG and f | 1 | 0010 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ADDWFC | f, d, a | Add WREG and Carry bit to f | 1 | 0010 | 00da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ANDWF | f, d, a | AND WREG with f | 1 | 0001 | 01da | ffff | ffff | Z, N | 1,2 |
| CLRF | f, a | Clear f | 1 | 0110 | 101a | ffff | ffff | Z | 2 |
| COMF | f, d, a | Complement f | 1 | 0001 | 11da | ffff | ffff | Z, N | 1, 2 |
| CPFSEQ | f, a | Compare f with WREG, Skip = | 1 (2 or 3) | 0110 | 001a | ffff | ffff | None | 4 |
| CPFSGT | f, a | Compare f with WREG, Skip > | 1 (2 or 3) | 0110 | 010a | ffff | ffff | None | 4 |
| CPFSLT | f, a | Compare f with WREG, Skip < | 1 (2 or 3) | 0110 | 000a | ffff | ffff | None | 1, 2 |
| DECF | f, d, a | Decrement f | 1 | 0000 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| DECFSZ | f, d, a | Decrement f, Skip if 0 | 1 (2 or 3) | 0010 | 11da | ffff | ffff | None | 1, 2, 3, 4 |
| DCFSNZ | f, d, a | Decrement f, Skip if Not 0 | 1 (2 or 3) | 0100 | 11da | ffff | ffff | None | 1, 2 |
| INCF | f, d, a | Increment f | 1 | 0010 | 10da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| INCFSZ | f, d, a | Increment f, Skip if 0 | 1 (2 or 3) | 0011 | 11da | ffff | ffff | None | 4 |
| INFSNZ | f, d, a | Increment f, Skip if Not 0 | 1 (2 or 3) | 0100 | 10da | ffff | ffff | None | 1, 2 |
| IORWF | f, d, a | Inclusive OR WREG with f | 1 | 0001 | 00da | ffff | ffff | Z, N | 1, 2 |
| MOVF | f, d, a | Move f | 1 | 0101 | 00da | ffff | ffff | Z, N | 1 |
| MOVFF | $f_s$, $f_d$ | Move $f_s$ (source) to 1st word | 2 | 1100 | ffff | ffff | ffff | None | |
| | | $f_d$ (destination) 2nd word | | 1111 | ffff | ffff | ffff | | |
| MOVWF | f, a | Move WREG to f | 1 | 0110 | 111a | ffff | ffff | None | |
| MULWF | f, a | Multiply WREG with f | 1 | 0000 | 001a | ffff | ffff | None | 1, 2 |
| NEGF | f, a | Negate f | 1 | 0110 | 110a | ffff | ffff | C, DC, Z, OV, N | |
| RLCF | f, d, a | Rotate Left f through Carry | 1 | 0011 | 01da | ffff | ffff | C, Z, N | 1, 2 |
| RLNCF | f, d, a | Rotate Left f (No Carry) | 1 | 0100 | 01da | ffff | ffff | Z, N | |
| RRCF | f, d, a | Rotate Right f through Carry | 1 | 0011 | 00da | ffff | ffff | C, Z, N | |
| RRNCF | f, d, a | Rotate Right f (No Carry) | 1 | 0100 | 00da | ffff | ffff | Z, N | |
| SETF | f, a | Set f | 1 | 0110 | 100a | ffff | ffff | None | 1, 2 |
| SUBFWB | f, d, a | Subtract f from WREG with Borrow | 1 | 0101 | 01da | ffff | ffff | C, DC, Z, OV, N | |
| SUBWF | f, d, a | Subtract WREG from f | 1 | 0101 | 11da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| SUBWFB | f, d, a | Subtract WREG from f with Borrow | 1 | 0101 | 10da | ffff | ffff | C, DC, Z, OV, N | |
| SWAPF | f, d, a | Swap Nibbles in f | 1 | 0011 | 10da | ffff | ffff | None | 4 |
| TSTFSZ | f, a | Test f, Skip if 0 | 1 (2 or 3) | 0110 | 011a | ffff | ffff | None | 1, 2 |
| XORWF | f, d, a | Exclusive OR WREG with f | 1 | 0001 | 10da | ffff | ffff | Z, N | |

**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.

**TABLE 26-2: PIC18F87J11 FAMILY INSTRUCTION SET (CONTINUED)**

| Mnemonic, | Operands | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **BIT-ORIENTED OPERATIONS** | | | | | | | | | |
| BCF | f, b, a | Bit Clear f | 1 | 1001 | bbba | ffff | ffff | None | 1, 2 |
| BSF | f, b, a | Bit Set f | 1 | 1000 | bbba | ffff | ffff | None | 1, 2 |
| BTFSC | f, b, a | Bit Test f, Skip if Clear | 1 (2 or 3) | 1011 | bbba | ffff | ffff | None | 3, 4 |
| BTFSS | f, b, a | Bit Test f, Skip if Set | 1 (2 or 3) | 1010 | bbba | ffff | ffff | None | 3, 4 |
| BTG | f, b, a | Bit Toggle f | 1 | 0111 | bbba | ffff | ffff | None | 1, 2 |
| **CONTROL OPERATIONS** | | | | | | | | | |
| BC | n | Branch if Carry | 1 (2) | 1110 | 0010 | nnnn | nnnn | None | |
| BN | n | Branch if Negative | 1 (2) | 1110 | 0110 | nnnn | nnnn | None | |
| BNC | n | Branch if Not Carry | 1 (2) | 1110 | 0011 | nnnn | nnnn | None | |
| BNN | n | Branch if Not Negative | 1 (2) | 1110 | 0111 | nnnn | nnnn | None | |
| BNOV | n | Branch if Not Overflow | 1 (2) | 1110 | 0101 | nnnn | nnnn | None | |
| BNZ | n | Branch if Not Zero | 1 (2) | 1110 | 0001 | nnnn | nnnn | None | |
| BOV | n | Branch if Overflow | 1 (2) | 1110 | 0100 | nnnn | nnnn | None | |
| BRA | n | Branch Unconditionally | 2 | 1101 | 0nnn | nnnn | nnnn | None | |
| BZ | n | Branch if Zero | 1 (2) | 1110 | 0000 | nnnn | nnnn | None | |
| CALL | n, s | Call Subroutine 1st word | 2 | 1110 | 110s | kkkk | kkkk | None | |
| | | 2nd word | | 1111 | kkkk | kkkk | kkkk | | |
| CLRWDT | — | Clear Watchdog Timer | 1 | 0000 | 0000 | 0000 | 0100 | $\overline{TO}$, $\overline{PD}$ | |
| DAW | — | Decimal Adjust WREG | 1 | 0000 | 0000 | 0000 | 0111 | C | |
| GOTO | n | Go to Address 1st word | 2 | 1110 | 1111 | kkkk | kkkk | None | |
| | | 2nd word | | 1111 | kkkk | kkkk | kkkk | | |
| NOP | — | No Operation | 1 | 0000 | 0000 | 0000 | 0000 | None | |
| NOP | — | No Operation | 1 | 1111 | xxxx | xxxx | xxxx | None | 4 |
| POP | — | Pop Top of Return Stack (TOS) | 1 | 0000 | 0000 | 0000 | 0110 | None | |
| PUSH | — | Push Top of Return Stack (TOS) | 1 | 0000 | 0000 | 0000 | 0101 | None | |
| RCALL | n | Relative Call | 2 | 1101 | 1nnn | nnnn | nnnn | None | |
| RESET | | Software Device Reset | 1 | 0000 | 0000 | 1111 | 1111 | All | |
| RETFIE | s | Return from Interrupt Enable | 2 | 0000 | 0000 | 0001 | 000s | GIE/GIEH, PEIE/GIEL | |
| RETLW | k | Return with Literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| RETURN | s | Return from Subroutine | 2 | 0000 | 0000 | 0001 | 001s | None | |
| SLEEP | — | Go into Standby mode | 1 | 0000 | 0000 | 0000 | 0011 | $\overline{TO}$, $\overline{PD}$ | |

**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.

**TABLE 26-2: PIC18F87J11 FAMILY INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **LITERAL OPERATIONS** | | | | | | | | | |
| ADDLW | k | Add Literal and WREG | 1 | 0000 | 1111 | kkkk | kkkk | C, DC, Z, OV, N | |
| ANDLW | k | AND Literal with WREG | 1 | 0000 | 1011 | kkkk | kkkk | Z, N | |
| IORLW | k | Inclusive OR Literal with WREG | 1 | 0000 | 1001 | kkkk | kkkk | Z, N | |
| LFSR | f, k | Move Literal (12-bit) 2nd word | 2 | 1110 | 1110 | 00ff | kkkk | None | |
| | | to FSR (f) 1st word | | 1111 | 0000 | kkkk | kkkk | | |
| MOVLB | k | Move Literal to BSR<3:0> | 1 | 0000 | 0001 | 0000 | kkkk | None | |
| MOVLW | k | Move Literal to WREG | 1 | 0000 | 1110 | kkkk | kkkk | None | |
| MULLW | k | Multiply Literal with WREG | 1 | 0000 | 1101 | kkkk | kkkk | None | |
| RETLW | k | Return with Literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| SUBLW | k | Subtract WREG from Literal | 1 | 0000 | 1000 | kkkk | kkkk | C, DC, Z, OV, N | |
| XORLW | k | Exclusive OR Literal with WREG | 1 | 0000 | 1010 | kkkk | kkkk | Z, N | |
| **DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS** | | | | | | | | | |
| TBLRD* | | Table Read | 2 | 0000 | 0000 | 0000 | 1000 | None | |
| TBLRD*+ | | Table Read with Post-Increment | | 0000 | 0000 | 0000 | 1001 | None | |
| TBLRD*- | | Table Read with Post-Decrement | | 0000 | 0000 | 0000 | 1010 | None | |
| TBLRD+* | | Table Read with Pre-Increment | | 0000 | 0000 | 0000 | 1011 | None | |
| TBLWT* | | Table Write | 2 | 0000 | 0000 | 0000 | 1100 | None | |
| TBLWT*+ | | Table Write with Post-Increment | | 0000 | 0000 | 0000 | 1101 | None | |
| TBLWT*- | | Table Write with Post-Decrement | | 0000 | 0000 | 0000 | 1110 | None | |
| TBLWT+* | | Table Write with Pre-Increment | | 0000 | 0000 | 0000 | 1111 | None | |

**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.

### 26.1.1 STANDARD INSTRUCTION SET

**ADDLW** **ADD Literal to W**

Syntax: ADDLW k

Operands: $0 \leq k \leq 255$

Operation: $(W) + k \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| `0000` | `1111` | `kkkk` | `kkkk` |

Description: The contents of W are added to the 8-bit literal 'k' and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: `ADDLW 15h`

Before Instruction
W = 10h
After Instruction
W = 25h

**ADDWF** **ADD W to f**

Syntax: ADDWF f {,d {,a}}

Operands: $0 \leq f \leq 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(W) + (f) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| `0010` | `01da` | `ffff` | `ffff` |

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'.

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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

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<br>d ∈ [0,1]<br>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'.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: `ANDLW 05Fh`

Before Instruction
- W = A3h

After Instruction
- W = 03h

**ANDWF** **AND W with f**

| | |
|---|---|
| Syntax: | ANDWF f {,d {,a}} |
| Operands: | 0 ≤ f ≤ 255<br>d ∈ [0,1]<br>a ∈ [0,1] |
| Operation: | (W) .AND. (f) → 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’.<br>If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.<br>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 26.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example: `ANDWF REG, 0, 0`

Before Instruction
W = 17h
REG = C2h
After Instruction
W = 02h
REG = C2h

**BC** **Branch if Carry**

| | |
|---|---|
| Syntax: | BC n |
| Operands: | -128 ≤ n ≤ 127 |
| Operation: | if Carry bit is ‘1’,<br>(PC) + 2 + 2n → PC |
| Status Affected: | None |
| Encoding: | `1110` `0010` `nnnn` `nnnn` |
| Description: | If the Carry bit is ’1’, then the program will branch.<br>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 BC 5`

Before Instruction
PC = address (HERE)
After Instruction
If Carry = 1;
PC = address (HERE + 12)
If Carry = 0;
PC = address (HERE + 2)

## BCF — Bit Clear f

Syntax: BCF f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

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 $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

## BN — Branch if Negative

Syntax: BN n

Operands: $-128 \le n \le 127$

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 ≤ n ≤ 127 |
| Operation: | if Carry bit is '0',<br>(PC) + 2 + 2n → PC |
| Status Affected: | None |
| Encoding: | `1110` `0011` `nnnn` `nnnn` |
| Description: | If the Carry bit is '0', then the program will branch.<br>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 BNC Jump`

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 ≤ n ≤ 127 |
| Operation: | if Negative bit is '0',<br>(PC) + 2 + 2n → PC |
| Status Affected: | None |
| Encoding: | `1110` `0111` `nnnn` `nnnn` |
| Description: | If the Negative bit is '0', then the program will branch.<br>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 BNN Jump`

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',<br>(PC) + 2 + 2n → PC |
| Status Affected: | None |
| Encoding: | `1110` `0101` `nnnn` `nnnn` |
| Description: | If the Overflow bit is '0', then the program will branch.<br>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 BNOV Jump`

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',<br>(PC) + 2 + 2n → PC |
| Status Affected: | None |
| Encoding: | `1110` `0001` `nnnn` `nnnn` |
| Description: | If the Zero bit is '0', then the program will branch.<br>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 BNZ Jump`

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: | $-1024 \leq n \leq 1023$ |
| 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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: `HERE BRA Jump`

Before Instruction
PC = address (`HERE`)
After Instruction
PC = address (`Jump`)

**BSF** **Bit Set f**

| | |
|---|---|
| Syntax: | BSF f, b {,a} |
| Operands: | $0 \leq f \leq 255$<br>$0 \leq b \leq 7$<br>$a \in [0,1]$ |
| Operation: | 1 → f<b> |
| Status Affected: | None |
| Encoding: | `1000` `bbba` `ffff` `ffff` |
| Description: | Bit 'b' in register 'f' is set.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `BSF FLAG_REG, 7, 1`

Before Instruction
FLAG_REG = 0Ah
After Instruction
FLAG_REG = 8Ah

**BTFSC** **Bit Test File, Skip if Clear**

Syntax: 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:

| 1011 | bbba | ffff | ffff |
|---|---|---|---|

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 26.2.3 "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 register 'f' | Process Data | No 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 |
| No operation | No operation | No operation | No operation |

Example:

```
HERE    BTFSC   FLAG, 1, 0
FALSE   :
TRUE    :
```

Before Instruction
PC = address (`HERE`)
After Instruction
If FLAG<1> = 0;
PC = address (`TRUE`)
If FLAG<1> = 1;
PC = address (`FALSE`)

**BTFSS** **Bit Test File, Skip if Set**

Syntax: BTFSS f, b {,a}

Operands: $0 \leq f \leq 255$
$0 \leq b < 7$
$a \in [0,1]$

Operation: skip if (f<b>) = `1`

Status Affected: None

Encoding:

| 1010 | bbba | ffff | ffff |
|---|---|---|---|

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 $f \leq 95$ (5Fh). See **Section 26.2.3 "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 register 'f' | Process Data | No 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 |
| No operation | No operation | No operation | No operation |

Example:

```
HERE    BTFSS   FLAG, 1, 0
FALSE   :
TRUE    :
```

Before Instruction
PC = address (`HERE`)
After Instruction
If FLAG<1> = 0;
PC = address (`FALSE`)
If FLAG<1> = 1;
PC = address (`TRUE`)

**BTG** **Bit Toggle f**

| | |
|---|---|
| Syntax: | BTG f, b {,a} |
| Operands: | $0 \le f \le 255$<br>$0 \le b < 7$<br>$a \in [0,1]$ |
| Operation: | $(\overline{f<b>}) \rightarrow f<b>$ |
| Status Affected: | None |
| Encoding: | `0111` `bbba` `ffff` `ffff` |
| Description: | Bit ‘b’ in data memory location ‘f’ is inverted.<br>If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write register ‘f’ |

Example: `BTG PORTC, 4, 0`

Before Instruction:
PORTC = `0111 0101` [75h]
After Instruction:
PORTC = `0110 0101` [65h]

**BOV** **Branch if Overflow**

| | |
|---|---|
| Syntax: | BOV n |
| Operands: | $-128 \le n \le 127$ |
| Operation: | if Overflow bit is ‘1’,<br>$(PC) + 2 + 2n \rightarrow PC$ |
| Status Affected: | None |
| Encoding: | `1110` `0100` `nnnn` `nnnn` |
| Description: | If the Overflow bit is ‘1’, then the program will branch.<br>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 BOV Jump`

Before Instruction
PC = address (`HERE`)
After Instruction
If Overflow = 1;
PC = address (`Jump`)
If Overflow = 0;
PC = address (`HERE + 2`)

**BZ** **Branch if Zero**

Syntax: BZ n

Operands: $-128 \leq n \leq 127$

Operation: if Zero bit is '1',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0000 | nnnn | nnnn |

Description: If the Zero 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 BZ Jump`

Before Instruction
PC = address (HERE)
After Instruction
If Zero = 1;
PC = address (Jump)
If Zero = 0;
PC = address (HERE + 2)

**CALL** **Subroutine Call**

Syntax: CALL k {,s}

Operands: $0 \leq k \leq 1048575$
$s \in [0,1]$

Operation: (PC) + 4 → TOS,
k → PC<20:1>;
if s = 1,
(W) → WS,
(STATUS) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (k<7:0>) | 1110 | 110s | $k_7$kkk | kkkk$_0$ |
| 2nd word(k<19:8>) | 1111 | $k_{19}$kkk | kkkk | kkkk$_8$ |

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. 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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0>, | Push PC to stack | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation | No operation | No operation |

Example: `HERE CALL THERE,1`

Before Instruction
PC = address (HERE)
After Instruction
PC = address (THERE)
TOS = address (HERE + 4)
WS = W
BSRS = BSR
STATUSS = STATUS

**CLRF** **Clear f**

| | |
|---|---|
| Syntax: | CLRF f {,a} |
| Operands: | 0 ≤ f ≤ 255<br>a ∈ [0,1] |
| Operation: | 000h → f,<br>1 → Z |
| Status Affected: | Z |
| Encoding: | `0110` `101a` `ffff` `ffff` |
| Description: | Clears the contents of the specified register.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `CLRF FLAG_REG,1`

Before Instruction
FLAG_REG = 5Ah
After Instruction
FLAG_REG = 00h

**CLRWDT** **Clear Watchdog Timer**

| | |
|---|---|
| Syntax: | CLRWDT |
| Operands: | None |
| Operation: | 000h → WDT,<br>000h → WDT postscaler,<br>1 → $\overline{TO}$,<br>1 → $\overline{PD}$ |
| Status Affected: | $\overline{TO}$, $\overline{PD}$ |
| Encoding: | `0000` `0000` `0000` `0100` |
| Description: | `CLRWDT` instruction resets the Watchdog Timer. It also resets the post-scaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | No operation |

Example: `CLRWDT`

Before Instruction
WDT Counter = ?
After Instruction
WDT Counter = 00h
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

## COMF — Complement f

| | |
|---|---|
| Syntax: | COMF f {,d {,a}} |
| Operands: | 0 ≤ f ≤ 255<br>d ∈ [0,1]<br>a ∈ [0,1] |
| Operation: | $\bar{f}$ → dest |
| Status Affected: | N, Z |
| Encoding: | `0001` `11da` `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'.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `COMF REG, 0, 0`

Before Instruction
REG = 13h
After Instruction
REG = 13h
W = ECh

## CPFSEQ — Compare f with W, Skip if f = W

| | |
|---|---|
| Syntax: | CPFSEQ f {,a} |
| Operands: | 0 ≤ f ≤ 255<br>a ∈ [0,1] |
| Operation: | (f) – (W),<br>skip if (f) = (W)<br>(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.<br>If 'f' = W, then the fetched instruction is discarded and a `NOP` is executed instead, making this a two-cycle instruction.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1(2)<br>**Note:** 3 cycles if skip and followed by a 2-word instruction. |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No 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 |
| No operation | No operation | No operation | No operation |

Example:

```
HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

**CPFSGT** — **Compare f with W, Skip if f > W**

| | |
|---|---|
| Syntax: | CPFSGT f {,a} |
| Operands: | $0 \le f \le 255$<br>$a \in [0,1]$ |
| Operation: | (f) – (W),<br>skip if (f) > (W)<br>(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.<br>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.<br>If 'a' is '`0`', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1(2)<br>**Note:** 3 cycles if skip and followed by a 2-word instruction. |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No 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 |
| 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: | $0 \le f \le 255$<br>$a \in [0,1]$ |
| Operation: | (f) – (W),<br>skip if (f) < (W)<br>(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.<br>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.<br>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)<br>**Note:** 3 cycles if skip and followed by a 2-word instruction. |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No 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 |
| 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,<br>(W<3:0>) + 6 → W<3:0>;<br>else,<br>(W<3:0>) → W<3:0><br><br>If [W<7:4> > 9] or [C = 1] then,<br>(W<7:4>) + 6 → W<7:4>,<br>C = 1;<br>else,<br>(W<7:4>) → 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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register W | Process Data | Write W |

Example 1: `DAW`

Before Instruction
- W = A5h
- C = 0
- DC = 0

After Instruction
- W = 05h
- C = 1
- DC = 0

Example 2:

Before Instruction
- W = CEh
- C = 0
- DC = 0

After Instruction
- W = 34h
- C = 1
- DC = 0

### DECF — Decrement f

| | |
|---|---|
| Syntax: | DECF f {,d {,a}} |
| Operands: | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$ |
| Operation: | (f) – 1 → 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'.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

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: | $0 \le f \le 255$<br>$d \in [0,1]$<br>$a \in [0,1]$ |
| Operation: | (f) – 1 → dest,<br>skip if result = `0` |
| Status Affected: | None |
| Encoding: | `0010` `11da` `ffff` `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'.<br>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.<br>If 'a' is '`0`', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1(2)<br>**Note:** 3 cycles if skip and followed by a 2-word instruction. |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

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 |
| No operation | No operation | No operation | No operation |

Example:

```
HERE      DECFSZ   CNT, 1, 1
          GOTO     LOOP
CONTINUE
```

Before Instruction
PC = Address (`HERE`)
After Instruction
CNT = CNT – 1
If CNT = 0;
PC = Address (`CONTINUE`)
If CNT ≠ 0;
PC = Address (`HERE + 2`)

| DCFSNZ | Decrement f, Skip if not 0 |
|---|---|
| Syntax: | DCFSNZ f {,d {,a}} |
| Operands: | $0 \le f \le 255$<br>$d \in [0,1]$<br>$a \in [0,1]$ |
| Operation: | (f) – 1 → dest,<br>skip if result ≠ `0` |
| Status Affected: | None |
| Encoding: | `0100` `11da` `ffff` `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'.<br>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.<br>If 'a' is '`0`', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1(2)<br>**Note:** 3 cycles if skip and followed by a 2-word instruction. |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

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 |
| No operation | No operation | No operation | No operation |

Example:

```
HERE    DCFSNZ  TEMP, 1, 0
ZERO    :
NZERO   :
```

Before Instruction
TEMP = ?
After Instruction
TEMP = TEMP – 1,
If TEMP = 0;
PC = Address (`ZERO`)
If TEMP ≠ 0;
PC = Address (`NZERO`)

**GOTO** **Unconditional Branch**

Syntax: GOTO k

Operands: $0 \le k \le 1048575$

Operation: k → PC<20:1>

Status Affected: None

| Encoding: | | | | |
|---|---|---|---|---|
| 1st word (k<7:0>) | 1110 | 1111 | $k_7$kkk | kkkk$_0$ |
| 2nd word(k<19:8>) | 1111 | $k_{19}$kkk | kkkk | kkkk$_8$ |

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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0>, | No operation | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation | No operation | No operation |

Example: `GOTO THERE`

After Instruction
PC = Address (THERE)

**INCF** **Increment f**

Syntax: INCF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

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'.

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 $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

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: INCFSZ f {,d {,a}}

Operands: $0 \le f \le 255$
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) + 1 → dest,
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'.

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 $f \le 95$ (5Fh). See **Section 26.2.3 "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 register 'f' | Process Data | Write to destination |

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 |
| No operation | No operation | No operation | No operation |

Example:

```
HERE    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

**INFSNZ** **Increment f, Skip if not 0**

Syntax: INFSNZ f {,d {,a}}

Operands: $0 \le f \le 255$
d ∈ [0,1]
a ∈ [0,1]

Operation: (f) + 1 → dest,
skip if result ≠ 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'.

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 $f \le 95$ (5Fh). See **Section 26.2.3 "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 register 'f' | Process Data | Write to destination |

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 |
| No operation | No operation | No operation | No operation |

Example:

```
HERE    INFSNZ  REG, 1, 0
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

**IORLW** **Inclusive OR Literal with W**

| | |
|---|---|
| Syntax: | IORLW k |
| Operands: | $0 \leq k \leq 255$ |
| 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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | 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: | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$ |
| 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'.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | 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: $0 \leq f \leq 2$
$0 \leq k \leq 4095$

Operation: k → FSRf

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 1110 | 00ff | $k_{11}$kkk |
| 1111 | 0000 | $k_7$kkk | kkkk |

Description: The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' MSB | Process Data | Write literal 'k' MSB to FSRfH |
| Decode | Read literal 'k' LSB | Process Data | Write literal 'k' to FSRfL |

Example: `LFSR 2, 3ABh`

After Instruction
FSR2H = 03h
FSR2L = ABh

### MOVF — Move f

Syntax: MOVF f {,d {,a}}

Operands: $0 \leq f \leq 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: f → dest

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 00da | ffff | ffff |

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'. 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 $f \leq 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write W |

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: | $0 \le f_s \le 4095$<br>$0 \le f_d \le 4095$ |
| Operation: | $(f_s) \rightarrow f_d$ |
| Status Affected: | None |

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (source) | `1100` | `ffff` | `ffff` | `ffff`$_s$ |
| 2nd word (destin.) | `1111` | `ffff` | `ffff` | `ffff`$_d$ |

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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' (src) | Process Data | No operation |
| Decode | No operation<br>No dummy read | No operation | Write register 'f' (dest) |

Example: `MOVFF   REG1, REG2`

Before Instruction
REG1 = 33h
REG2 = 11h
After Instruction
REG1 = 33h
REG2 = 33h

| **MOVLB** | **Move Literal to Low Nibble in BSR** |
|---|---|
| Syntax: | MOVLB k |
| Operands: | $0 \le k \le 255$ |
| Operation: | $k \rightarrow$ BSR |
| Status Affected: | None |

Encoding:

| | | | |
|---|---|---|---|
| `0000` | `0001` | `kkkk` | `kkkk` |

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 $k_7$:$k_4$.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write literal 'k' to BSR |

Example: `MOVLB   5`

Before Instruction
BSR Register = 02h
After Instruction
BSR Register = 05h

**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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: `MOVLW 5Ah`

After Instruction
W = 5Ah

**MOVWF** **Move W to f**

| | |
|---|---|
| Syntax: | MOVWF f {,a} |
| Operands: | 0 ≤ f ≤ 255<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

| | |
|---|---|
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

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: $0 \leq k \leq 255$

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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write registers PRODH: PRODL |

Example: `MULLW 0C4h`

Before Instruction
- W = E2h
- PRODH = ?
- PRODL = ?

After Instruction
- W = E2h
- PRODH = ADh
- PRODL = 08h

### MULWF — Multiply W with f

Syntax: MULWF f {,a}

Operands: $0 \leq f \leq 255$
$a \in [0,1]$

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 the 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 $f \leq 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write registers PRODH: PRODL |

Example: `MULWF REG, 1`

Before Instruction
- W = C4h
- REG = B5h
- PRODH = ?
- PRODL = ?

After Instruction
- W = C4h
- REG = B5h
- PRODH = 8Ah
- PRODL = 94h

**NEGF** **Negate f**

| | |
|---|---|
| Syntax: | NEGF f {,a} |
| Operands: | $0 \le f \le 255$<br>$a \in [0,1]$ |
| Operation: | $(\bar{f}) + 1 \rightarrow f$ |
| 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'.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

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`<br>`1111` `xxxx` `xxxx` `xxxx` |
| Description: | No operation. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |

Example:

None.

**POP** **Pop Top of Return Stack**

| | |
|---|---|
| Syntax: | POP |
| Operands: | None |
| Operation: | (TOS) → bit bucket |
| Status Affected: | None |
| Encoding: | `0000` `0000` `0000` `0110` |
| Description: | 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. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | POP TOS value | No operation |

Example:

```
POP
GOTO    NEW
```

Before Instruction
TOS = 0031A2h
Stack (1 level down) = 014332h

After Instruction
TOS = 014332h
PC = NEW

**PUSH** **Push Top of Return Stack**

| | |
|---|---|
| Syntax: | PUSH |
| Operands: | None |
| Operation: | (PC + 2) → TOS |
| Status Affected: | None |
| Encoding: | `0000` `0000` `0000` `0101` |
| Description: | 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. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | PUSH PC + 2 onto return stack | No operation | No operation |

Example:

```
PUSH
```

Before Instruction
TOS = 345Ah
PC = 0124h

After Instruction
PC = 0126h
TOS = 0126h
Stack (1 level down) = 345Ah

### RCALL — Relative Call

| | |
|---|---|
| Syntax: | RCALL n |
| Operands: | $-1024 \leq n \leq 1023$ |
| Operation: | (PC) + 2 → TOS,<br>(PC) + 2 + 2n → PC |
| Status Affected: | None |
| Encoding: | `1101` `1nnn` `nnnn` `nnnn` |
| Description: | 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. |
| Words: | 1 |
| Cycles: | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n'<br>PUSH PC to stack | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: `HERE RCALL Jump`

Before Instruction
PC = Address (`HERE`)
After Instruction
PC = Address (`Jump`)
TOS = Address (`HERE + 2`)

### RESET — Reset

| | |
|---|---|
| Syntax: | RESET |
| Operands: | None |
| Operation: | Reset all registers and flags that are affected by a $\overline{MCLR}$ Reset. |
| Status Affected: | All |
| Encoding: | `0000` `0000` `1111` `1111` |
| Description: | This instruction provides a way to execute a $\overline{MCLR}$ Reset in software. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Start reset | No operation | No operation |

Example: `RESET`

After Instruction
Registers = Reset Value
Flags* = Reset Value

| RETFIE | Return from Interrupt |
|---|---|
| Syntax: | RETFIE {s} |
| Operands: | s ∈ [0,1] |
| Operation: | (TOS) → PC,<br>1 → GIE/GIEH or PEIE/GIEL;<br>if s = 1,<br>(WS) → W,<br>(STATUSS) → STATUS,<br>(BSRS) → BSR,<br>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. |
| Words: | 1 |
| Cycles: | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | POP PC from stack<br>Set GIEH or GIEL |
| No operation | No operation | No operation | No operation |

Example: RETFIE 1

After Interrupt

| | | |
|---|---|---|
| PC | = | TOS |
| W | = | WS |
| BSR | = | BSRS |
| STATUS | = | STATUSS |
| GIE/GIEH, PEIE/GIEL | = | 1 |

| RETLW | Return Literal to W |
|---|---|
| Syntax: | RETLW k |
| Operands: | 0 ≤ k ≤ 255 |
| Operation: | k → W,<br>(TOS) → PC,<br>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: RETURN {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC;
if s = 1,
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, 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, STATUSS and BSRS are loaded into their corresponding registers W, STATUS and BSR. If ‘s’ = 0, no update of these registers occurs.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | POP PC from stack |
| No operation | No operation | No operation | No operation |

Example: `RETURN`

After Instruction:
PC = TOS

**RLCF — Rotate Left f through Carry**

Syntax: RLCF f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → 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’.

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 26.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.


Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example: `RLCF REG, 0, 0`

Before Instruction
REG = 1110 0110
C = 0

After Instruction
REG = 1110 0110
W = 1100 1100
C = 1

**RLNCF** **Rotate Left f (No Carry)**

Syntax: RLNCF f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f<n>) → dest<n + 1>,
(f<7>) → dest<0>

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'.

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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.


Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RLNCF REG, 1, 0

Before Instruction
REG = 1010 1011
After Instruction
REG = 0101 0111

**RRCF** **Rotate Right f through Carry**

Syntax: RRCF f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: (f<n>) → dest<n – 1>,
(f<0>) → C,
(C) → dest<7>

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'.

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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.


Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RRCF REG, 0, 0

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 0111 0011
C = 0

**RRNCF** **Rotate Right f (No Carry)**

Syntax: RRNCF f {,d {,a}}

Operands: $0 \leq f \leq 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<n>) → dest<n – 1>,
(f<0>) → 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’.

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.

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 26.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.


Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

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: $0 \leq f \leq 255$
$a \in [0,1]$

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 $f \leq 95$ (5Fh). See **Section 26.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write register ‘f’ |

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 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

| 0000 | 0000 | 0000 | 0011 |
|---|---|---|---|

Description: The Power-Down status bit ($\overline{PD}$) is cleared. The Time-out status bit ($\overline{TO}$) 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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | Go to Sleep |

Example: `SLEEP`

Before Instruction
$\overline{TO}$ = ?
$\overline{PD}$ = ?
After Instruction
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† If WDT causes wake-up, this bit is cleared.

**SUBFWB** — **Subtract f from W with Borrow**

Syntax: SUBFWB f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) – (f) – ($\overline{C}$) → 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'.

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 $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: `SUBFWB REG, 1, 0`

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: `SUBFWB REG, 0, 0`

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: `SUBFWB REG, 1, 0`

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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

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: | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$ |
| 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'.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(f) – (W) – (\overline{C}) \rightarrow$ 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'.

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 $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: `SUBWFB REG, 1, 0`

Before Instruction
REG = 19h (0001 1001)
W = 0Dh (0000 1101)
C = 1
After Instruction
REG = 0Ch (0000 1011)
W = 0Dh (0000 1101)
C = 1
Z = 0
N = 0 ; result is positive

Example 2: `SUBWFB REG, 0, 0`

Before Instruction
REG = 1Bh (0001 1011)
W = 1Ah (0001 1010)
C = 0
After Instruction
REG = 1Bh (0001 1011)
W = 00h
C = 1
Z = 1 ; result is zero
N = 0

Example 3: `SUBWFB REG, 1, 0`

Before Instruction
REG = 03h (0000 0011)
W = 0Eh (0000 1101)
C = 1
After Instruction
REG = F5h (1111 0100) ; [2's comp]
W = 0Eh (0000 1101)
C = 0
Z = 0
N = 1 ; result is negative

**SWAPF** **Swap f**

Syntax: SWAPF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<3:0>) $\rightarrow$ dest<7:4>,
(f<7:4>) $\rightarrow$ 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'.

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 $f \le 95$ (5Fh). See **Section 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `SWAPF REG, 1, 0`

Before Instruction
REG = 53h
After Instruction
REG = 35h

**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:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 10nn<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |

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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation (Read Program Memory) | No operation | No operation (Write TABLAT) |

**TBLRD** **Table Read (Continued)**

Example 1: `TBLRD *+ ;`

Before Instruction

| | | |
|---|---|---|
| TABLAT | = | 55h |
| TBLPTR | = | 00A356h |
| MEMORY(00A356h) | = | 34h |

After Instruction

| | | |
|---|---|---|
| TABLAT | = | 34h |
| TBLPTR | = | 00A357h |

Example 2: `TBLRD +* ;`

Before Instruction

| | | |
|---|---|---|
| TABLAT | = | AAh |
| TBLPTR | = | 01A357h |
| MEMORY(01A357h) | = | 12h |
| MEMORY(01A358h) | = | 34h |

After Instruction

| | | |
|---|---|---|
| TABLAT | = | 34h |
| TBLPTR | = | 01A358h |

**TBLWT** **Table Write**

| | |
|---|---|
| Syntax: | TBLWT ( *; *+; *-; +*) |
| Operands: | None |
| Operation: | if TBLWT*,<br>(TABLAT) → Holding Register,<br>TBLPTR – No Change;<br>if TBLWT*+,<br>(TABLAT) → Holding Register,<br>(TBLPTR) + 1 → TBLPTR;<br>if TBLWT*-,<br>(TABLAT) → Holding Register,<br>(TBLPTR) – 1 → TBLPTR;<br>if TBLWT+*,<br>(TBLPTR) + 1 → TBLPTR,<br>(TABLAT) → Holding Register |
| Status Affected: | None |

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 11nn<br>nn=0 *<br>=1 *+<br>=2 *-<br>=3 +* |

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 6.0 "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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation (Read TABLAT) | No operation | No operation (Write to Holding Register) |

**TBLWT** **Table Write (Continued)**

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) | = | 55h |

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: | $0 \leq f \leq 255$<br>$a \in [0,1]$ |
| 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.<br>If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1(2)<br>**Note:** 3 cycles if skip and followed by a 2-word instruction. |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No 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 |
| No operation | No operation | No operation | No operation |

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: | $0 \leq k \leq 255$ |
| Operation: | (W) .XOR. k $\rightarrow$ 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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example:

```
XORLW   0AFh
```

Before Instruction
W = B5h
After Instruction
W = 1Ah

| **XORWF** | **Exclusive OR W with f** |
|---|---|
| Syntax: | XORWF f {,d {,a}} |
| Operands: | $0 \leq f \leq 255$<br>$d \in [0,1]$<br>$a \in [0,1]$ |
| Operation: | (W) .XOR. (f) $\rightarrow$ 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'.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>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 26.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `XORWF REG, 1, 0`

Before Instruction
- REG = AFh
- W = B5h

After Instruction
- REG = 1Ah
- W = B5h

## 26.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, the PIC18F87J11 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 (`ADDULNK` and `SUBULNK`) 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 26-3. Detailed descriptions are provided in **Section 26.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 26-1 (page 348) apply to both the standard and extended PIC18 instruction sets.

| **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. |
|---|---|

### 26.2.1 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 26.2.3.1 "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 ("{ }"). |
|---|---|

**TABLE 26-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word MSb | | | LSb | Status Affected |
|---|---|---|---|---|---|---|---|---|
| ADDFSR | f, k | Add Literal to FSR | 1 | 1110 | 1000 | ffkk | kkkk | None |
| ADDULNK | k | Add Literal to FSR2 and Return | 2 | 1110 | 1000 | 11kk | kkkk | None |
| CALLW | | Call Subroutine using WREG | 2 | 0000 | 0000 | 0001 | 0100 | None |
| MOVSF | $z_s$, $f_d$ | Move $z_s$ (source) to 1st word | 2 | 1110 | 1011 | 0zzz | zzzz | None |
| | | $f_d$ (destination) 2nd word | | 1111 | ffff | ffff | ffff | |
| MOVSS | $z_s$, $z_d$ | Move $z_s$ (source) to 1st word | 2 | 1110 | 1011 | 1zzz | zzzz | None |
| | | $z_d$ (destination) 2nd word | | 1111 | xxxx | xzzz | zzzz | |
| PUSHL | k | Store Literal at FSR2, Decrement FSR2 | 1 | 1110 | 1010 | kkkk | kkkk | None |
| SUBFSR | f, k | Subtract Literal from FSR | 1 | 1110 | 1001 | ffkk | kkkk | None |
| SUBULNK | k | Subtract Literal from FSR2 and Return | 2 | 1110 | 1001 | 11kk | kkkk | None |

### 26.2.2 EXTENDED INSTRUCTION SET

**ADDFSR** **Add Literal to FSR**

| | |
|---|---|
| Syntax: | ADDFSR f, k |
| Operands: | 0 ≤ k ≤ 63<br>f ∈ [ 0, 1, 2 ] |
| Operation: | FSR(f) + k → FSR(f) |
| Status Affected: | None |
| Encoding: | `1110` `1000` `ffkk` `kkkk` |
| Description: | The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'. |
| Words: | 1 |
| Cycles: | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to FSR |

Example: `ADDFSR 2, 23h`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

**ADDULNK** **Add Literal to FSR2 and Return**

| | |
|---|---|
| Syntax: | ADDULNK k |
| Operands: | 0 ≤ k ≤ 63 |
| Operation: | FSR2 + k → FSR2,<br>(TOS) → PC |
| Status Affected: | None |
| Encoding: | `1110` `1000` `11kk` `kkkk` |
| Description: | The 6-bit literal 'k' is added to the contents of FSR2. A `RETURN` is then executed by loading the PC with the TOS.<br>The instruction takes two cycles to execute; a `NOP` is performed during the second cycle.<br>This may be thought of as a special case of the `ADDFSR` instruction, where f = 3 (binary '`11`'); it operates only on FSR2. |
| Words: | 1 |
| Cycles: | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to FSR |
| No Operation | No Operation | No Operation | No Operation |

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,<br>(W) $\rightarrow$ PCL,<br>(PCLATH) $\rightarrow$ PCH,<br>(PCLATU) $\rightarrow$ PCU |
| Status Affected: | None |
| Encoding: | `0000 0000 0001 0100` |
| 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.<br>Unlike `CALL`, there is no option to update W, STATUS or BSR. |
| Words: | 1 |
| Cycles: | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read WREG | Push PC to stack | No operation |
| No operation | No operation | No operation | No operation |

Example: `HERE CALLW`

Before Instruction
```
PC     = address (HERE)
PCLATH = 10h
PCLATU = 00h
W      = 06h
```
After Instruction
```
PC     = 001006h
TOS    = address (HERE + 2)
PCLATH = 10h
PCLATU = 00h
W      = 06h
```

**MOVSF** **Move Indexed to f**

| | |
|---|---|
| Syntax: | MOVSF $[z_s]$, $f_d$ |
| Operands: | $0 \le z_s \le 127$<br>$0 \le f_d \le 4095$ |
| Operation: | $((FSR2) + z_s) \rightarrow f_d$ |
| Status Affected: | None |
| Encoding:<br>1st word (source)<br>2nd word (destin.) | `1110 1011 0zzz` $zzzz_s$<br>`1111 ffff ffff` $ffff_d$ |
| 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).<br>The `MOVSF` instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.<br>If the resultant source address points to an Indirect Addressing register, the value returned will be 00h. |
| Words: | 2 |
| Cycles: | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | No operation<br>No dummy read | No operation | Write register 'f' (dest) |

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 — Move Indexed to Indexed

Syntax: MOVSS [$z_s$], [$z_d$]

Operands: $0 \le z_s \le 127$
$0 \le z_d \le 127$

Operation: $((FSR2) + z_s) \rightarrow ((FSR2) + z_d)$

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (source) | 1110 | 1011 | 1zzz | $zzzz_s$ |
| 2nd word (dest.) | 1111 | xxxx | xzzz | $zzzz_d$ |

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 '$z_s$' or '$z_d$', 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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | Determine dest addr | Determine dest addr | Write to dest reg |

Example: `MOVSS [05h], [06h]`

Before Instruction
- FSR2 = 80h
- Contents of 85h = 33h
- Contents of 86h = 11h

After Instruction
- FSR2 = 80h
- Contents of 85h = 33h
- Contents of 86h = 33h

### PUSHL — Store Literal at FSR2, Decrement FSR2

Syntax: PUSHL k

Operands: 0 £ k £ 255

Operation: k → (FSR2),
FSR2 – 1 → FSR2

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 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:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process data | Write to destination |

Example: `PUSHL 08h`

Before Instruction
- FSR2H:FSR2L = 01ECh
- Memory (01ECh) = 00h

After Instruction
- FSR2H:FSR2L = 01EBh
- Memory (01ECh) = 08h

## SUBFSR — Subtract Literal from FSR

Syntax: SUBFSR f, k

Operands: 0 £ k £ 63
f Î [ 0, 1, 2 ]

Operation: FSRf – k ® FSRf

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| `1110` | `1001` | `ffkk` | `kkkk` |

Description: The 6-bit literal ‘k’ is subtracted from the contents of the FSR specified by ‘f’.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example: `SUBFSR 2, 23h`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 03DCh

## SUBULNK — Subtract Literal from FSR2 and Return

Syntax: SUBULNK k

Operands: 0 £ k £ 63

Operation: FSR2 – k ® FSR2,
(TOS) → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| `1110` | `1001` | `11kk` | `kkkk` |

Description: 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.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |
| No Operation | No Operation | No Operation | No Operation |

Example: SUBULNK 23h

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 03DCh
PC = (TOS)

### 26.2.3 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 6.6.1 "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 26.2.3.1 "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.

#### 26.2.3.1 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.

### 26.2.4 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, 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)**

Syntax: ADDWF [k] {,d}

Operands: $0 \leq k \leq 95$
$d \in [0,1]$

Operation: (W) + ((FSR2) + k) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 01d0 | kkkk | kkkk |

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'.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process Data | Write to destination |

Example: `ADDWF [OFST] ,0`

Before Instruction
W = 17h
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 20h
After Instruction
W = 37h
Contents of 0A2Ch = 20h

**BSF** — **Bit Set Indexed (Indexed Literal Offset mode)**

Syntax: BSF [k], b

Operands: $0 \leq f \leq 95$
$0 \leq b \leq 7$

Operation: 1 → ((FSR2) + k)<b>

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1000 | bbb0 | kkkk | kkkk |

Description: Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `BSF [FLAG_OFST], 7`

Before Instruction
FLAG_OFST = 0Ah
FSR2 = 0A00h
Contents of 0A0Ah = 55h
After Instruction
Contents of 0A0Ah = D5h

**SETF** — **Set Indexed (Indexed Literal Offset mode)**

Syntax: SETF [k]

Operands: $0 \leq k \leq 95$

Operation: FFh → ((FSR2) + k)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 1000 | kkkk | kkkk |

Description: The contents of the register indicated by FSR2, offset by 'k', are set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process Data | Write register |

Example: `SETF [OFST]`

Before Instruction
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 00h
After Instruction
Contents of 0A2Ch = FFh

#### 26.2.5 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. 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.

## 27.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers and dsPIC® digital signal controllers are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB C Compiler for Various Device Families
  - HI-TECH C® for Various Device Families
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- Device Programmers
  - PICkit™ 2 Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits, and Starter Kits

### 27.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16/32-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)
  - In-Circuit 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
- 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 IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either C or assembly)
- One-touch compile or assemble, and download to emulator and simulator tools (automatically updates all project information)
- Debug using:
  - Source files (C or assembly)
  - Mixed C and assembly
  - 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.

## 27.2 MPLAB C Compilers for Various Device Families

The MPLAB C Compiler code development systems are complete ANSI C compilers for Microchip's PIC18, PIC24 and PIC32 families of microcontrollers and the dsPIC30 and dsPIC33 families of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

## 27.3 HI-TECH C for Various Device Families

The HI-TECH C Compiler code development systems are complete ANSI C compilers for Microchip's PIC family of microcontrollers and the dsPIC family of digital signal controllers. These compilers provide powerful integration capabilities, omniscient code generation and ease of use.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

The compilers include a macro assembler, linker, preprocessor, and one-step driver, and can run on multiple platforms.

## 27.4 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB 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

## 27.5 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

## 27.6 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC devices. MPLAB 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 device instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

## 27.7 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 C Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 27.8 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 emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB IDE. In upcoming releases of MPLAB IDE, new devices will be supported, and new features will be added. MPLAB REAL ICE offers significant advantages over competitive emulators including low-cost, full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 27.9 MPLAB ICD 3 In-Circuit Debugger System

MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost effective high-speed hardware debugger/programmer for Microchip Flash Digital Signal Controller (DSC) and microcontroller (MCU) devices. It debugs and programs PIC® Flash microcontrollers and dsPIC® DSCs with the powerful, yet easy-to-use graphical user interface of MPLAB Integrated Development Environment (IDE).

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 27.10 PICkit 3 In-Circuit Debugger/ Programmer and PICkit 3 Debug Express

The MPLAB PICkit 3 allows debugging and programming of PIC® and dsPIC® Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB Integrated Development Environment (IDE). The MPLAB PICkit 3 is connected to the design engineer's PC using a full speed USB interface and can be connected to the target via an Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the reset line to implement in-circuit debugging and In-Circuit Serial Programming™.

The PICkit 3 Debug Express include the PICkit 3, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

## 27.11 PICkit 2 Development Programmer/Debugger and PICkit 2 Debug Express

The PICkit™ 2 Development Programmer/Debugger is a low-cost development tool with an easy to use interface for programming and debugging Microchip's Flash families of microcontrollers. The full featured Windows® programming interface supports baseline (PIC10F, PIC12F5xx, PIC16F5xx), midrange (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30, dsPIC33, and PIC32 families of 8-bit, 16-bit, and 32-bit microcontrollers, and many Microchip Serial EEPROM products. With Microchip's powerful MPLAB Integrated Development Environment (IDE) the PICkit™ 2 enables in-circuit debugging on most PIC® microcontrollers. In-Circuit-Debugging runs, halts and single steps the program while the PIC microcontroller is embedded in the application. When halted at a breakpoint, the file registers can be examined and modified.

The PICkit 2 Debug Express include the PICkit 2, demo board and microcontroller, hookup cables and CDROM with user's guide, lessons, tutorial, compiler and MPLAB IDE software.

## 27.12 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 MMC card for file storage and data applications.

## 27.13 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta A/D, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

# 28.0 ELECTRICAL CHARACTERISTICS

## Absolute Maximum Ratings(†)

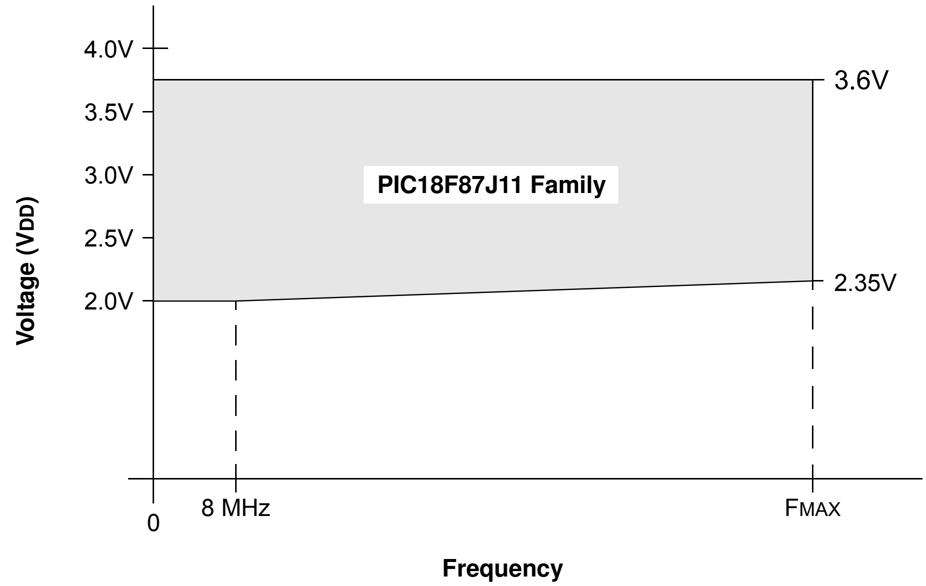
Ambient temperature under bias ............ -40°C to +100°C
Storage temperature ............ -65°C to +150°C
Voltage on any digital only input pin or $\overline{\text{MCLR}}$ with respect to $V_{SS}$ (except $V_{DD}$) ............ -0.3V to 6.0V
Voltage on any combined digital and analog pin with respect to $V_{SS}$ ............ -0.3V to ($V_{DD}$ + 0.3V)
Voltage on $V_{DDCORE}$ with respect to $V_{SS}$ ............ -0.3V to 2.75V
Voltage on $V_{DD}$ with respect to $V_{SS}$ ............ -0.3V to 4.0V
Total power dissipation **(Note 1)** ............ 1.0W
Maximum current out of $V_{SS}$ pin ............ 300 mA
Maximum current into $V_{DD}$ pin ............ 250 mA
Input clamp current, $I_{IK}$ ($V_I$ < 0 or $V_I$ > $V_{DD}$) **(Note 2)** ............ ±0 mA
Output clamp current, $I_{OK}$ ($V_O$ < 0 or $V_O$ > $V_{DD}$) **(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:
$P_{dis} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL}) + \sum (V_{TPOUT} \times I_{TPOUT})$

**2:** No clamping diodes are present.

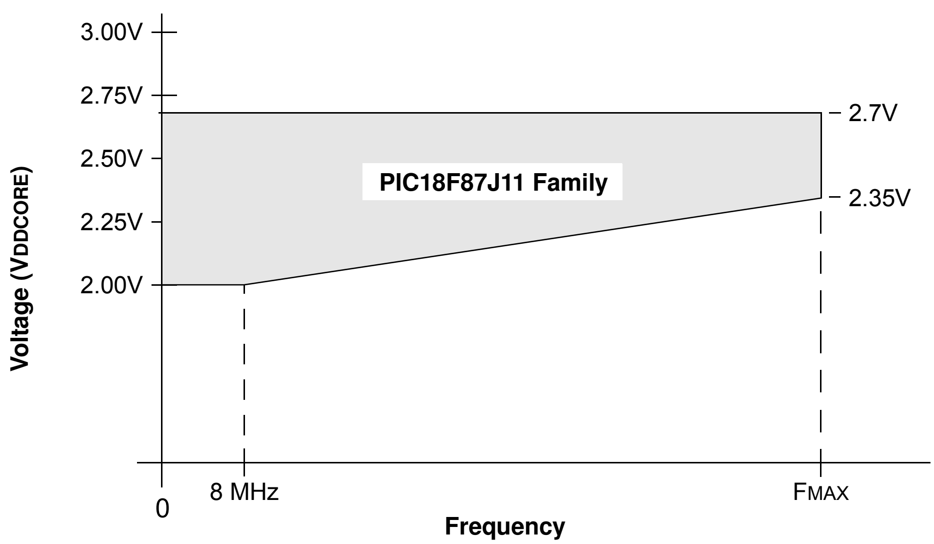
† **NOTICE:** Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

**FIGURE 28-1: PIC18F87J11 FAMILY VOLTAGE-FREQUENCY GRAPH, REGULATOR ENABLED (INDUSTRIAL)**




FMAX = 25 MHz in 8-Bit External Memory mode
FMAX = 48 MHz in all other modes

**FIGURE 28-2: PIC18F87J11 FAMILY VOLTAGE-FREQUENCY GRAPH, REGULATOR DISABLED (INDUSTRIAL)()**




FMAX = 25 MHz in 8-Bit External Memory mode
FMAX = 48 MHz in all other modes

**Note 1:** When the on-chip voltage regulator is disabled, VDD and VDDCORE must be maintained so that VDDCORE ≤ VDD ≤ 3.6V.

## 28.1 DC Characteristics: Supply Voltage PIC18F87J11 Family (Industrial)

| **PIC18F87J11 Family** (Industrial) | | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| D001 | VDD | **Supply Voltage** | VDDCORE<br>2.0 | —<br>— | 3.6<br>3.6 | V<br>V | ENVREG tied to VSS<br>ENVREG tied to VDD |
| D001B | VDDCORE | **External Supply for Microcontroller Core** | 2.0 | — | 2.7 | V | ENVREG tied to VSS |
| D001C | AVDD | **Analog Supply Voltage** | VDD – 0.3 | — | VDD + 0.3 | V | |
| D001D | AVSS | **Analog Ground Potential** | VSS – 0.3 | — | VSS + 0.3 | V | |
| D002 | VDR | **RAM Data Retention Voltage**(1) | 1.5 | — | — | V | |
| D003 | VPOR | **VDD Power-on Reset Voltage** | — | — | 0.7 | V | See **Section 5.3 “Power-on Reset (POR)”** for details |
| D004 | SVDD | **VDD Rise Rate** to Ensure Internal Power-on Reset Signal | 0.05 | — | — | V/ms | See **Section 5.3 “Power-on Reset (POR)”** for details |
| D005 | VBOR | **Brown-out Reset Voltage** | 1.75(2) | 2.0 | 2.4 | V | |

**Note 1:** This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

**2:** When the Brown-out Reset is enabled, the part will continue to operate until the BOR occurs. This is valid, although VDD may be below the minimum VDD voltage.

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J11 Family (Industrial)

| PIC18F87J11 Family (Industrial) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | |
| | **Power-Down Current (IPD)**(1) | | | | | |
| | All devices | 0.5 | 1.4 | μA | -40°C | VDD = 2.0V(4)<br>(**Sleep** mode) |
| | | 0.5 | 1.4 | μA | +25°C | |
| | | 5.5 | 10.2 | μA | +85°C | |
| | All devices | 0.6 | 1.5 | μA | -40°C | VDD = 2.5V(4)<br>(**Sleep** mode) |
| | | 0.6 | 1.5 | μA | +25°C | |
| | | 6.8 | 12.6 | μA | +85°C | |
| | All devices | 2.9 | 7 | μA | -40°C | VDD = 3.3V(5)<br>(**Sleep** mode) |
| | | 3.6 | 7 | μA | +25°C | |
| | | 9.6 | 19 | μA | +85°C | |

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**2:** The supply current is mainly a function of the 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 is 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 is disabled (ENVREG = 0, tied to VSS).

**5:** Voltage regulator is enabled (ENVREG = 1, tied to VDD, REGSLP = 1).

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J11 Family (Industrial) (Continued)

| PIC18F87J11 Family (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)**(2,3) | | | | | | |
| | All devices | 5 | 14.2 | µA | -40°C | VDD = 2.0V, VDDCORE = 2.0V(4) | FOSC = 31 kHz (**RC_RUN** mode, internal oscillator source) |
| | | 5.5 | 14.2 | µA | +25°C | | |
| | | 10 | 19.0 | µA | +85°C | | |
| | All devices | 6.8 | 16.5 | µA | -40°C | VDD = 2.5V, VDDCORE = 2.5V(4) | |
| | | 7.6 | 16.5 | µA | +25°C | | |
| | | 14 | 22.4 | µA | +85°C | | |
| | All devices | 37 | 84 | µA | -40°C | VDD = 3.3V(5) | |
| | | 51 | 84 | µA | +25°C | | |
| | | 72 | 108 | µA | +85°C | | |
| | All devices | 0.43 | 0.82 | mA | -40°C | VDD = 2.0V, VDDCORE = 2.0V(4) | FOSC = 1 MHz (**RC_RUN** mode, internal oscillator source) |
| | | 0.47 | 0.82 | mA | +25°C | | |
| | | 0.52 | 0.95 | mA | +85°C | | |
| | All devices | 0.52 | 0.98 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V(4) | |
| | | 0.57 | 0.98 | mA | +25°C | | |
| | | 0.63 | 1.10 | mA | +85°C | | |
| | All devices | 0.59 | 0.96 | mA | -40°C | VDD = 3.3V(5) | |
| | | 0.65 | 0.96 | mA | +25°C | | |
| | | 0.72 | 1.18 | mA | +85°C | | |
| | All devices | 0.88 | 1.45 | mA | -40°C | VDD = 2.0V, VDDCORE = 2.0V(4) | FOSC = 4 MHz (**RC_RUN** mode, internal oscillator source) |
| | | 1 | 1.45 | mA | +25°C | | |
| | | 1.1 | 1.58 | mA | +85°C | | |
| | All devices | 1.2 | 1.72 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V(4) | |
| | | 1.3 | 1.72 | mA | +25°C | | |
| | | 1.4 | 1.85 | mA | +85°C | | |
| | All devices | 1.3 | 2.87 | mA | -40°C | VDD = 3.3V(5) | |
| | | 1.4 | 2.87 | mA | +25°C | | |
| | | 1.5 | 2.96 | mA | +85°C | | |

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**2:** The supply current is mainly a function of the 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 is 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 is disabled (ENVREG = 0, tied to VSS).

**5:** Voltage regulator is enabled (ENVREG = 1, tied to VDD, REGSLP = 1).

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J11 Family (Industrial) (Continued)

| PIC18F87J11 Family (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.**(2,3) | | | | | | |
| | All devices | 3 | 9.4 | μA | -40°C | VDD = 2.0V, VDDCORE = 2.0V(4) | FOSC = 31 kHz (**RC_IDLE** mode, internal oscillator source) |
| | | 3.3 | 9.4 | μA | +25°C | | |
| | | 8.5 | 17.2 | μA | +85°C | | |
| | All devices | 4 | 10.5 | μA | -40°C | VDD = 2.5V, VDDCORE = 2.5V(4) | |
| | | 4.3 | 10.5 | μA | +25°C | | |
| | | 10.3 | 19.5 | μA | +85°C | | |
| | All devices | 34 | 82 | μA | -40°C | VDD = 3.3V(5) | |
| | | 48 | 82 | μA | +25°C | | |
| | | 69 | 105 | μA | +85°C | | |
| | All devices | 0.33 | 0.75 | mA | -40°C | VDD = 2.0V, VDDCORE = 2.0V(4) | FOSC = 1 MHz (**RC_IDLE** mode, internal oscillator source) |
| | | 0.37 | 0.75 | mA | +25°C | | |
| | | 0.41 | 0.84 | mA | +85°C | | |
| | All devices | 0.39 | 0.78 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V(4) | |
| | | 0.42 | 0.78 | mA | +25°C | | |
| | | 0.47 | 0.91 | mA | +85°C | | |
| | All devices | 0.43 | 0.82 | mA | -40°C | VDD = 3.3V(5) | |
| | | 0.48 | 0.82 | mA | +25°C | | |
| | | 0.54 | 0.95 | mA | +85°C | | |
| | All devices | 0.53 | 0.98 | mA | -40°C | VDD = 2.0V, VDDCORE = 2.0V(4) | FOSC = 4 MHz (**RC_IDLE** mode, internal oscillator source) |
| | | 0.57 | 0.98 | mA | +25°C | | |
| | | 0.61 | 1.12 | mA | +85°C | | |
| | All devices | 0.63 | 1.14 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V(4) | |
| | | 0.67 | 1.14 | mA | +25°C | | |
| | | 0.72 | 1.25 | mA | +85°C | | |
| | All devices | 0.7 | 1.27 | mA | -40°C | VDD = 3.3V(5) | |
| | | 0.76 | 1.27 | mA | +25°C | | |
| | | 0.82 | 1.45 | mA | +85°C | | |

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**2:** The supply current is mainly a function of the 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;
$\overline{\text{MCLR}}$ = VDD; WDT is 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 is disabled (ENVREG = 0, tied to VSS).

**5:** Voltage regulator is enabled (ENVREG = 1, tied to VDD, REGSLP = 1).

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J11 Family (Industrial) (Continued)

| PIC18F87J11 Family (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.**[2,3] | | | | | | |
| | All devices | 0.17 | 0.35 | mA | -40°C | VDD = 2.0V, VDDCORE = 2.0V[4] | FOSC = 1 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 0.18 | 0.35 | mA | +25°C | | |
| | | 0.20 | 0.42 | mA | +85°C | | |
| | All devices | 0.29 | 0.52 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | |
| | | 0.31 | 0.52 | mA | +25°C | | |
| | | 0.34 | 0.61 | mA | +85°C | | |
| | All devices | 0.59 | 1.1 | mA | -40°C | VDD = 3.3V[5] | |
| | | 0.44 | 0.85 | mA | +25°C | | |
| | | 0.42 | 0.85 | mA | +85°C | | |
| | All devices | 0.70 | 1.25 | mA | -40°C | VDD = 2.0V, VDDCORE = 2.0V[4] | FOSC = 4 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 0.75 | 1.25 | mA | +25°C | | |
| | | 0.79 | 1.36 | mA | +85°C | | |
| | All devices | 1.10 | 1.7 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | |
| | | 1.10 | 1.7 | mA | +25°C | | |
| | | 1.12 | 1.82 | mA | +85°C | | |
| | All devices | 1.55 | 1.95 | mA | -40°C | VDD = 3.3V[5] | |
| | | 1.47 | 1.89 | mA | +25°C | | |
| | | 1.54 | 1.92 | mA | +85°C | | |
| | All devices | 9.9 | 14.8 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | FOSC = 48 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 9.5 | 14.8 | mA | +25°C | | |
| | | 10.1 | 15.2 | mA | +85°C | | |
| | All devices | 13.3 | 23.2 | mA | -40°C | VDD = 3.3V[5] | |
| | | 12.2 | 22.7 | mA | +25°C | | |
| | | 12.1 | 22.7 | mA | +85°C | | |

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**2:** The supply current is mainly a function of the 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;
$\overline{MCLR}$ = VDD; WDT is 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 is disabled (ENVREG = `0`, tied to VSS).

**5:** Voltage regulator is enabled (ENVREG = `1`, tied to VDD, REGSLP = `1`).

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J11 Family (Industrial) (Continued)

| PIC18F87J11 Family (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.[2,3]** | | | | | | |
| | All devices | 4.5 | 5.2 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | FOSC = 4 MHz, 16 MHz internal (**PRI_RUN HSPLL** mode) |
| | | 4.4 | 5.2 | mA | +25°C | | |
| | | 4.5 | 5.2 | mA | +85°C | | |
| | All devices | 5.7 | 6.7 | mA | -40°C | VDD = 3.3V[5] | |
| | | 5.5 | 6.3 | mA | +25°C | | |
| | | 5.3 | 6.3 | mA | +85°C | | |
| | All devices | 10.8 | 13.5 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | FOSC = 10 MHz, 40 MHz internal (**PRI_RUN HSPLL** mode) |
| | | 10.8 | 13.5 | mA | +25°C | | |
| | | 9.9 | 13.0 | mA | +85°C | | |
| | All devices | 13.4 | 24.1 | mA | -40°C | VDD = 3.3V[5] | |
| | | 12.3 | 20.2 | mA | +25°C | | |
| | | 11.2 | 19.5 | mA | +85°C | | |

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**2:** The supply current is mainly a function of the 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;
$\overline{\text{MCLR}}$ = VDD; WDT is 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 is disabled (ENVREG = `0`, tied to VSS).

**5:** Voltage regulator is enabled (ENVREG = `1`, tied to VDD, REGSLP = `1`).

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J11 Family (Industrial) (Continued)

| PIC18F87J11 Family (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.**[2,3] | | | | | | |
| | All devices | 0.10 | 0.26 | mA | -40°C | VDD = 2.0V, VDDCORE = 2.0V[4] | FOSC = 1 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 0.07 | 0.18 | mA | +25°C | | |
| | | 0.09 | 0.22 | mA | +85°C | | |
| | All devices | 0.25 | 0.48 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | |
| | | 0.13 | 0.30 | mA | +25°C | | |
| | | 0.10 | 0.26 | mA | +85°C | | |
| | All devices | 0.45 | 0.68 | mA | -40°C | VDD = 3.3V[5] | |
| | | 0.26 | 0.45 | mA | +25°C | | |
| | | 0.30 | 0.54 | mA | +85°C | | |
| | All devices | 0.36 | 0.60 | mA | -40°C | VDD = 2.0V, VDDCORE = 2.0V[4] | FOSC = 4 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 0.33 | 0.56 | mA | +25°C | | |
| | | 0.35 | 0.56 | mA | +85°C | | |
| | All devices | 0.52 | 0.81 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | |
| | | 0.45 | 0.70 | mA | +25°C | | |
| | | 0.46 | 0.70 | mA | +85°C | | |
| | All devices | 0.80 | 1.15 | mA | -40°C | VDD = 3.3V[5] | |
| | | 0.66 | 0.98 | mA | +25°C | | |
| | | 0.65 | 0.98 | mA | +85°C | | |
| | All devices | 5.2 | 6.5 | mA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | FOSC = 48 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 4.9 | 5.9 | mA | +25°C | | |
| | | 3.4 | 4.5 | mA | +85°C | | |
| | All devices | 6.2 | 12.4 | mA | -40°C | VDD = 3.3V[5] | |
| | | 5.9 | 11.5 | mA | +25°C | | |
| | | 5.8 | 11.5 | mA | +85°C | | |

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**2:** The supply current is mainly a function of the 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;
$\overline{\text{MCLR}}$ = VDD; WDT is 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 is disabled (ENVREG = `0`, tied to VSS).

**5:** Voltage regulator is enabled (ENVREG = `1`, tied to VDD, REGSLP = `1`).

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J11 Family (Industrial) (Continued)

| PIC18F87J11 Family (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.**(2,3) | | | | | | |
| | All devices | 18 | 35 | μA | -40°C | VDD = 2.0V, VDDCORE = 2.0V(4) | FOSC = 32 kHz(3) (**SEC_RUN** mode, Timer1 as clock) |
| | | 19 | 35 | μA | +25°C | | |
| | | 28 | 49 | μA | +85°C | | |
| | All devices | 20 | 45 | μA | -40°C | VDD = 2.5V, VDDCORE = 2.5V(4) | |
| | | 21 | 45 | μA | +25°C | | |
| | | 32 | 61 | μA | +85°C | | |
| | All devices | 0.06 | 0.11 | mA | -40°C | VDD = 3.3V(5) | |
| | | 0.07 | 0.11 | mA | +25°C | | |
| | | 0.09 | 0.15 | mA | +85°C | | |
| | All devices | 14 | 28 | μA | -40°C | VDD = 2.0V, VDDCORE = 2.0V(4) | FOSC = 32 kHz(3) (**SEC_IDLE** mode, Timer1 as clock) |
| | | 15 | 28 | μA | +25°C | | |
| | | 24 | 43 | μA | +85°C | | |
| | All devices | 15 | 31 | μA | -40°C | VDD = 2.5V, VDDCORE = 2.5V(4) | |
| | | 16 | 31 | μA | +25°C | | |
| | | 27 | 50 | μA | +85°C | | |
| | All devices | 0.05 | 0.10 | mA | -40°C | VDD = 3.3V(5) | |
| | | 0.06 | 0.10 | mA | +25°C | | |
| | | 0.08 | 0.14 | mA | +85°C | | |

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**2:** The supply current is mainly a function of the 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 is 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 is disabled (ENVREG = 0, tied to VSS).

**5:** Voltage regulator is enabled (ENVREG = 1, tied to VDD, REGSLP = 1).

## 28.2 DC Characteristics: Power-Down and Supply Current PIC18F87J11 Family (Industrial) (Continued)

| PIC18F87J11 Family (Industrial) | | Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| D022 (ΔIWDT) | **Module Differential Currents (ΔIWDT, ΔIOSCB, ΔIAD)** | | | | | | |
| | **Watchdog Timer** | 2.1 | 7.0 | μA | -40°C | VDD = 2.0V, VDDCORE = 2.0V[4] | |
| | | 2.2 | 7.0 | μA | +25°C | | |
| | | 4.3 | 9.5 | μA | +85°C | | |
| | | 3.0 | 8.0 | μA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | |
| | | 3.1 | 8.0 | μA | +25°C | | |
| | | 5.5 | 10.4 | μA | +85°C | | |
| | | 5.9 | 12.1 | μA | -40°C | VDD = 3.3V | |
| | | 6.2 | 12.1 | μA | +25°C | | |
| | | 6.9 | 13.6 | μA | +85°C | | |
| D025 (ΔIOSCB) | **Timer1 Oscillator** | 14 | 24 | μA | -40°C | VDD = 2.0V, VDDCORE = 2.0V[4] | 32 kHz on Timer1[3] |
| | | 15 | 24 | μA | +25°C | | |
| | | 23 | 36 | μA | +85°C | | |
| | | 17 | 26 | μA | -40°C | VDD = 2.5V, VDDCORE = 2.5V[4] | 32 kHz on Timer1[3] |
| | | 18 | 26 | μA | +25°C | | |
| | | 25 | 38 | μA | +85°C | | |
| | | 19 | 35 | μA | -40°C | VDD = 3.3V | 32 kHz on Timer1[3] |
| | | 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 | VDD = 2.0V, VDDCORE = 2.0V[4] | A/D on, not converting |
| | | 3.0 | 10.0 | μA | -40°C to +85°C | VDD = 2.5V, VDDCORE = 2.5V[4] | |
| | | 3.2 | 11.0 | μA | -40°C to +85°C | VDD = 3.3V | |

**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 a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

**2:** The supply current is mainly a function of the 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 is 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 is disabled (ENVREG = 0, tied to VSS).

**5:** Voltage regulator is enabled (ENVREG = 1, tied to VDD, REGSLP = 1).

## 28.3 DC Characteristics:PIC18F87J11 Family (Industrial)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min** | **Max** | **Units** | **Conditions** |
| | VIL | **Input Low Voltage** | | | | |
| | | All I/O Ports: | | | | |
| D030 | | with TTL Buffer | VSS | 0.15 VDD | V | |
| D031 | | with Schmitt Trigger Buffer | VSS | 0.2 VDD | V | |
| D032 | | $\overline{MCLR}$ | VSS | 0.2 VDD | V | |
| D033 | | OSC1 | VSS | 0.3 VDD | V | HS, HSPLL modes |
| D033A | | OSC1 | VSS | 0.2 VDD | V | EC, ECPLL modes |
| D034 | | T13CKI | VSS | 0.3 | V | |
| | VIH | **Input High Voltage** | | | | |
| | | I/O Ports with Non 5.5V Tolerance:(2) | | | | |
| D040 | | with TTL Buffer | 0.25 VDD + 0.8V | VDD | V | VDD < 3.3V |
| D040A | | | 2.0 | VDD | V | 3.3V ≤ VDD ≤ 3.6V |
| D041 | | with Schmitt Trigger Buffer | 0.8 VDD | VDD | V | |
| D041A | | RC3 and RC4 | 0.7 VDD | VDD | V | I²C™ enabled |
| D041B | | | 2.1 | VDD | V | SMBus enabled |
| | | I/O Ports with 5.5V Tolerance:(2) | | | | |
| | | with TTL Buffer | 0.25 VDD + 0.8V | 5.5 | V | VDD < 3.3V |
| | | | 2.0 | 5.5 | V | 3.3V ≤ VDD ≤ 3.6V |
| | | with Schmitt Trigger Buffer | 0.8 VDD | 5.5 | V | |
| D042 | | $\overline{MCLR}$ | 0.8 VDD | VDD | V | |
| D043 | | OSC1 | 0.7 VDD | VDD | V | HS, HSPLL modes |
| D043A | | OSC1 | 0.8 VDD | VDD | V | EC, ECPLL modes |
| D044 | | T13CKI | 1.6 | VDD | V | |
| | IIL | **Input Leakage Current(1)** | | | | |
| D060 | | I/O Ports with Non 5.5V Tolerance(2) | — | ±1 | μA | VSS ≤ VPIN ≤ VDD,<br>Pin at high-impedance |
| D060A | | I/O Ports with 5.5V Tolerance(2) | — | ±1 | μA | VSS ≤ VPIN ≤ 5.5V,<br>Pin at high-impedance |
| D061 | | $\overline{MCLR}$ | — | ±1 | μA | Vss ≤ VPIN ≤ VDD |
| D063 | | OSC1 | — | ±5 | μA | Vss ≤ VPIN ≤ VDD |
| | IPU | **Weak Pull-up Current** | | | | |
| D070 | IPURB | PORTB Weak Pull-up Current | 80 | 400 | μA | VDD = 3.3V, VPIN = VSS |

**Note 1:** Negative current is defined as current sourced by the pin.
**2:** Refer to Table 11-1 for the pins that have corresponding tolerance limits.

## 28.3 DC Characteristics:PIC18F87J11 Family (Industrial) (Continued)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ $T_A$ ≤ +85°C for industrial | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min** | **Max** | **Units** | **Conditions** |
| | $V_{OL}$ | **Output Low Voltage** | | | | |
| D080 | | I/O Ports: | | | | |
| | | PORTA, PORTF, PORTG, PORTH | — | 0.4 | V | $I_{OL}$ = 2 mA, $V_{DD}$ = 3.3V, -40°C to +85°C |
| | | PORTD, PORTE, PORTJ | — | 0.4 | V | $I_{OL}$ = 4 mA, $V_{DD}$ = 3.3V, -40°C to +85°C |
| | | PORTB, PORTC | — | 0.4 | V | $I_{OL}$ = 8.5 mA, $V_{DD}$ = 3.3V, -40°C to +85°C |
| D083 | | OSC2/CLKO (EC, ECPLL modes) | — | 0.4 | V | $I_{OL}$ = 1.6 mA, $V_{DD}$ = 3.3V, -40°C to +85°C |
| | $V_{OH}$ | **Output High Voltage**(1) | | | | |
| D090 | | I/O Ports: | | | V | |
| | | PORTA, PORTF, PORTG, PORTH | 2.4 | — | V | $I_{OH}$ = -2 mA, $V_{DD}$ = 3.3V, -40°C to +85°C |
| | | PORTD, PORTE, PORTJ | 2.4 | — | V | $I_{OH}$ = -3 mA, $V_{DD}$ = 3.3V, -40°C to +85°C |
| | | PORTB, PORTC | 2.4 | — | V | $I_{OH}$ = -6 mA, $V_{DD}$ = 3.3V, -40°C to +85°C |
| D092 | | OSC2/CLKO (INTOSC, EC, ECPLL modes) | 2.4 | — | V | $I_{OH}$ = -1 mA, $V_{DD}$ = 3.3V, -40°C to +85°C |
| | | **Capacitive Loading Specs on Output Pins** | | | | |
| D100 | COSC2 | OSC2 Pin | — | 15 | pF | In HS mode when external clock is used to drive OSC1 |
| D101 | $C_{IO}$ | All I/O Pins and OSC2 | — | 50 | pF | To meet the AC Timing Specifications |
| D102 | $C_B$ | SCLx, SDAx | — | 400 | pF | I²C™ Specification |

**Note 1:** Negative current is defined as current sourced by the pin.

**2:** Refer to Table 11-1 for the pins that have corresponding tolerance limits.

**TABLE 28-1: MEMORY PROGRAMMING REQUIREMENTS**

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ $T_A$ ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| | | **Program Flash Memory** | | | | | |
| D130 | $E_P$ | Cell Endurance | 10K | — | — | E/W | -40°C to +85°C |
| D131 | $V_{PR}$ | $V_{DD}$ for Read | $V_{MIN}$ | — | 3.6 | V | $V_{MIN}$ = Minimum operating voltage |
| D132B | $V_{PEW}$ | Voltage for Self-Timed Erase or Write: | | | | | |
| | | $V_{DD}$ | 2.35 | — | 3.6 | V | ENVREG tied to $V_{DD}$ |
| | | $V_{DDCORE}$ | 2.25 | — | 2.7 | V | ENVREG tied to $V_{SS}$ |
| D133A | $T_{IW}$ | Self-Timed Write Cycle Time | — | 2.8 | — | ms | |
| | | Self-Timed Page Erase Cycle Time | — | 33.0 | — | ms | |
| D134 | $T_{RETD}$ | Characteristic Retention | 20 | — | — | Year | Provided no other specifications are violated |
| D135 | $I_{DDP}$ | Supply Current During Programming | — | 3 | 14 | mA | |
| D1xxx | $T_{WE}$ | Writes per Erase Cycle | — | — | 1 | | For each physical address |

† 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 28-2: COMPARATOR SPECIFICATIONS**

| Operating Conditions: 3.0V < VDD < 3.6V, -40°C < TA < +85°C (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| D300 | VIOFF | Input Offset Voltage | — | ±5.0 | ±25 | mV | |
| D301 | VICM | Input Common-Mode Voltage | 0 | — | AVDD – 1.5 | V | |
| D302 | CMRR | Common-Mode Rejection Ratio | 55 | — | — | dB | |
| D303 | TRESP | Response Time[1] | — | 150 | 400 | ns | |
| D304 | TMC2OV | Comparator Mode Change to Output Valid | — | — | 10 | µs | |
| D305 | VIRV | Internal Reference Voltage | — | 1.2[2] | — | V | ±1.2% |

**Note 1:** Response time is measured with one comparator input at (VDD – 1.5)/2, while the other input transitions from VSS to VDD.

**2:** The tolerance is ±1.2%.

**TABLE 28-3: VOLTAGE REFERENCE SPECIFICATIONS**

| Operating Conditions: 3.0V < VDD < 3.6V, -40°C < TA < +85°C (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| D310 | VRES | Resolution | VDD/24 | — | VDD/32 | LSb | |
| D311 | VRAA | Absolute Accuracy | — | — | 1/2 | LSb | |
| D312 | VRUR | Unit Resistor Value (R) | — | 2k | — | Ω | |
| D313 | TSET | Settling Time[1] | — | — | 10 | µs | |

**Note 1:** Settling time is measured while CVRR = 1 and the CVR<3:0> bits transition from '0000' to '1111'.

**TABLE 28-4: INTERNAL VOLTAGE REGULATOR SPECIFICATIONS**

| Operating Conditions: -40°C < TA < +85°C (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| | VRGOUT | Regulator Output Voltage | — | 2.5 | — | V | |
| | CF | External Filter Capacitor Value | 4.7 | 10 | — | µF | Capacitor must be low series resistance (<5 Ohms) |

## 28.4 AC (Timing) Characteristics

### 28.4.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

| | | | | | |
|---|---|---|---|---|---|
| 1. TppS2ppS | | | 3. TCC:ST | (I²C specifications only) | |
| 2. TppS | | | 4. Ts | (I²C specifications only) | |
| T | | | | | |
| | F | Frequency | | T | Time |
| Lowercase letters (pp) and their meanings: | | | | | |
| pp | | | | | |
| | cc | CCP1 | | osc | OSC1 |
| | ck | CLKO | | rd | $\overline{\text{RD}}$ |
| | cs | $\overline{\text{CS}}$ | | rw | $\overline{\text{RD}}$ or $\overline{\text{WR}}$ |
| | di | SDI | | sc | SCK |
| | do | SDO | | ss | $\overline{\text{SS}}$ |
| | dt | Data in | | t0 | T0CKI |
| | io | I/O port | | t1 | T13CKI |
| | mc | $\overline{\text{MCLR}}$ | | wr | $\overline{\text{WR}}$ |
| Uppercase letters and their meanings: | | | | | |
| S | | | | | |
| | F | Fall | | P | Period |
| | H | High | | R | Rise |
| | I | Invalid (High-impedance) | | V | Valid |
| | L | Low | | Z | High-impedance |
| I²C only | | | | | |
| | AA | output access | | High | High |
| | BUF | Bus free | | Low | Low |
| TCC:ST (I²C specifications only) | | | | | |
| CC | | | | | |
| | HD | Hold | | SU | Setup |
| ST | | | | | |
| | DAT | DATA input hold | | STO | Stop condition |
| | STA | Start condition | | | |

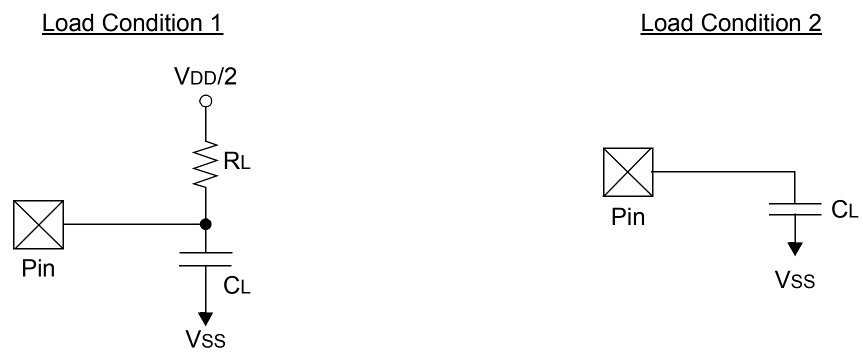
### 28.4.2 TIMING CONDITIONS

The temperature and voltages specified in Table 28-5 apply to all timing specifications unless otherwise noted. Figure 28-3 specifies the load conditions for the timing specifications.

**TABLE 28-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

| | |
|---|---|
| **AC CHARACTERISTICS** | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>Operating voltage VDD range as described in **Section 28.1** and **Section 28.3**. |

**FIGURE 28-3: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS**




RL = 464Ω

CL = 50 pF for all pins except OSC2/CLKO/RA6 and including D and E outputs as ports

CL = 15 pF for OSC2/CLKO/RA6

### 28.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 28-4: EXTERNAL CLOCK TIMING**

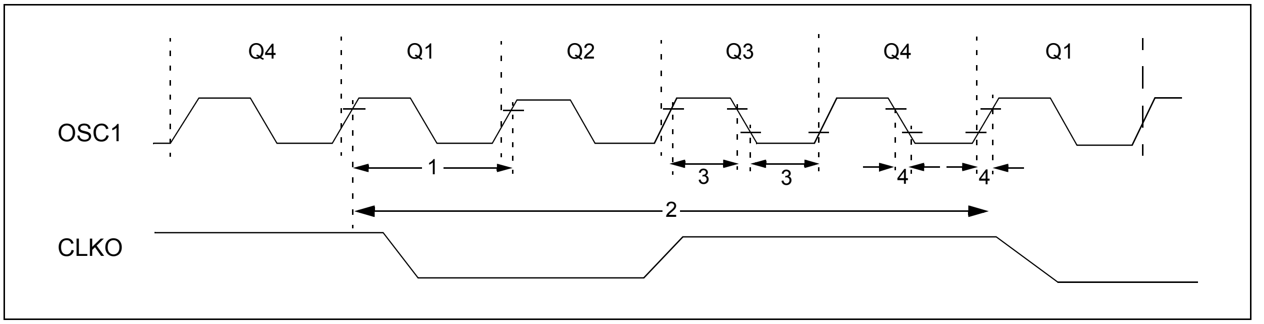



**TABLE 28-6: EXTERNAL CLOCK TIMING REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 1A | FOSC | External CLKI Frequency[1] | DC | 48 | MHz | EC Oscillator mode |
| | | | DC | 10 | | ECPLL Oscillator mode |
| | | Oscillator Frequency[1] | 4 | 25 | MHz | HS Oscillator mode |
| | | | 4 | 10 | | HSPLL Oscillator mode |
| 1 | TOSC | External CLKI Period[1] | 20.8 | — | ns | EC Oscillator mode |
| | | | 100 | — | | ECPLL Oscillator mode |
| | | Oscillator Period[1] | 40.0 | 250 | ns | HS Oscillator mode |
| | | | 100 | 250 | | HSPLL Oscillator mode |
| 2 | TCY | Instruction Cycle Time[1] | 83.3 | — | ns | TCY = 4/FOSC, Industrial |
| 3 | TOSL, TOSH | External Clock in (OSC1) High or Low Time | 10 | — | ns | HS Oscillator mode |
| 4 | TOSR, TOSF | External Clock in (OSC1) Rise or Fall Time | — | 7.5 | ns | HS Oscillator mode |

**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 28-7: PLL CLOCK TIMING SPECIFICATIONS (VDD = 2.15V TO 3.6V)**

| Param No. | Sym | Characteristic | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| F10 | FOSC | Oscillator Frequency Range | 4 | — | 10 | MHz | |
| F11 | FSYS | On-Chip VCO System Frequency | 16 | — | 40 | MHz | |
| F12 | $t_{rc}$ | PLL Start-up Time (lock time) | — | — | 2 | ms | |
| F13 | ΔCLK | CLKO Stability (jitter) | -2 | — | +2 | % | |

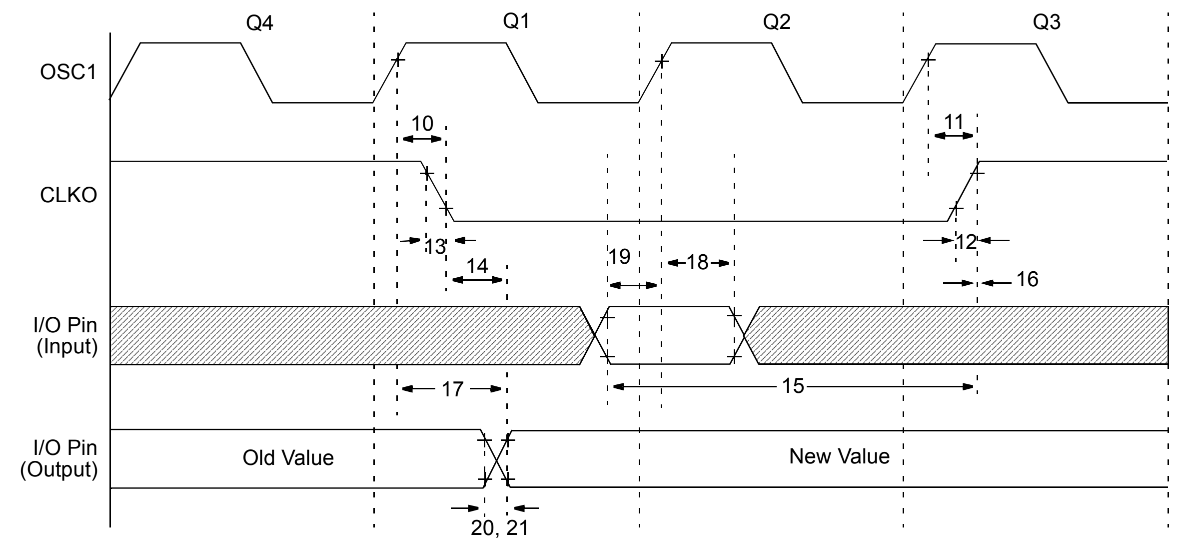
† Data in "Typ" column is at 3.3V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

**TABLE 28-8: INTERNAL RC ACCURACY (INTOSC AND INTRC SOURCES)**

| Param No. | Device | Min | Typ | Max | Units | Conditions | |
|---|---|---|---|---|---|---|---|
| | INTOSC Accuracy @ Freq = 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz, 125 kHz, 31 kHz(1) | | | | | | |
| | All Devices | -2 | +/-1 | 2 | % | +25°C | VDD = 2.7-3.3V |
| | | -5 | — | 5 | % | -10°C to +85°C | VDD = 2.0-3.3V |
| | | -10 | +/-1 | 10 | % | -40°C to +85°C | VDD = 2.0-3.3V |
| | INTRC Accuracy @ Freq = 31 kHz(1) | | | | | | |
| | All Devices | 21.7 | — | 40.3 | kHz | | |

**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 28-5: CLKO AND I/O TIMING**




**Note:** Refer to Figure 28-3 for load conditions.

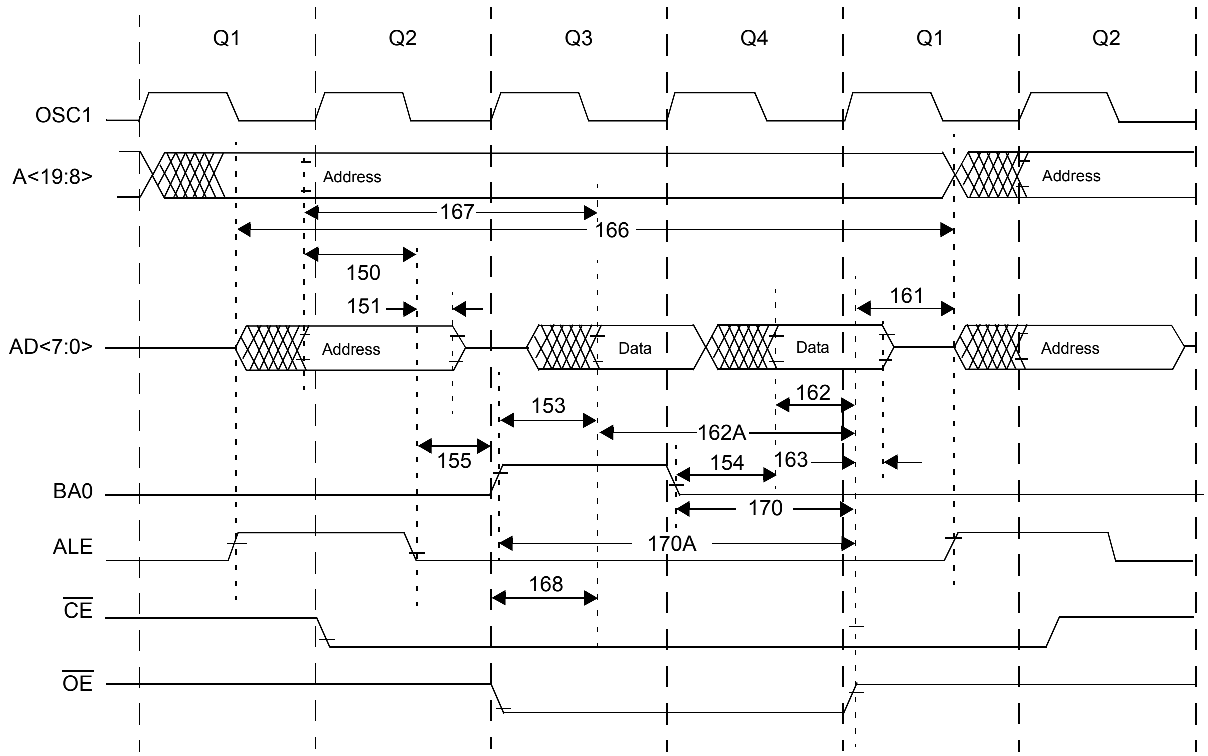
**TABLE 28-9: CLKO AND I/O TIMING REQUIREMENTS**

| Param No. | Symbol | Characteristic | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 10 | TOSH2CKL | OSC1 ↑ to CLKO ↓ | — | 75 | 200 | ns | **(Note 1)** |
| 11 | TOSH2CKH | OSC1 ↑ to CLKO ↑ | — | 75 | 200 | ns | **(Note 1)** |
| 12 | TCKR | CLKO Rise Time | — | 15 | 30 | ns | **(Note 1)** |
| 13 | TCKF | CLKO Fall Time | — | 15 | 30 | ns | **(Note 1)** |
| 14 | TCKL2IOV | CLKO ↓ to Port Out Valid | — | — | 0.5 TCY + 20 | ns | |
| 15 | TIOV2CKH | Port In Valid Before CLKO ↑ | 0.25 TCY + 25 | — | — | ns | |
| 16 | TCKH2IOI | Port In Hold After CLKO ↑ | 0 | — | — | ns | |
| 17 | TOSH2IOV | OSC1 ↑ (Q1 cycle) to Port Out Valid | — | 50 | 150 | ns | |
| 18 | TOSH2IOI | OSC1 ↑ (Q2 cycle) to Port Input Invalid (I/O in hold time) | 100 | — | — | ns | |
| 19 | TIOV2OSH | Port Input Valid to OSC1 ↑ (I/O in setup time) | 0 | — | — | ns | |
| 20 | TIOR | Port Output Rise Time | — | — | 6 | ns | |
| 21 | TIOF | Port Output Fall Time | — | — | 5 | ns | |
| 22† | TINP | INTx Pin High or Low Time | TCY | — | — | ns | |
| 23† | TRBP | RB<7:4> Change INTx High or Low Time | TCY | — | — | ns | |

† 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.

**FIGURE 28-6: PROGRAM MEMORY FETCH TIMING DIAGRAM (8-BIT)**



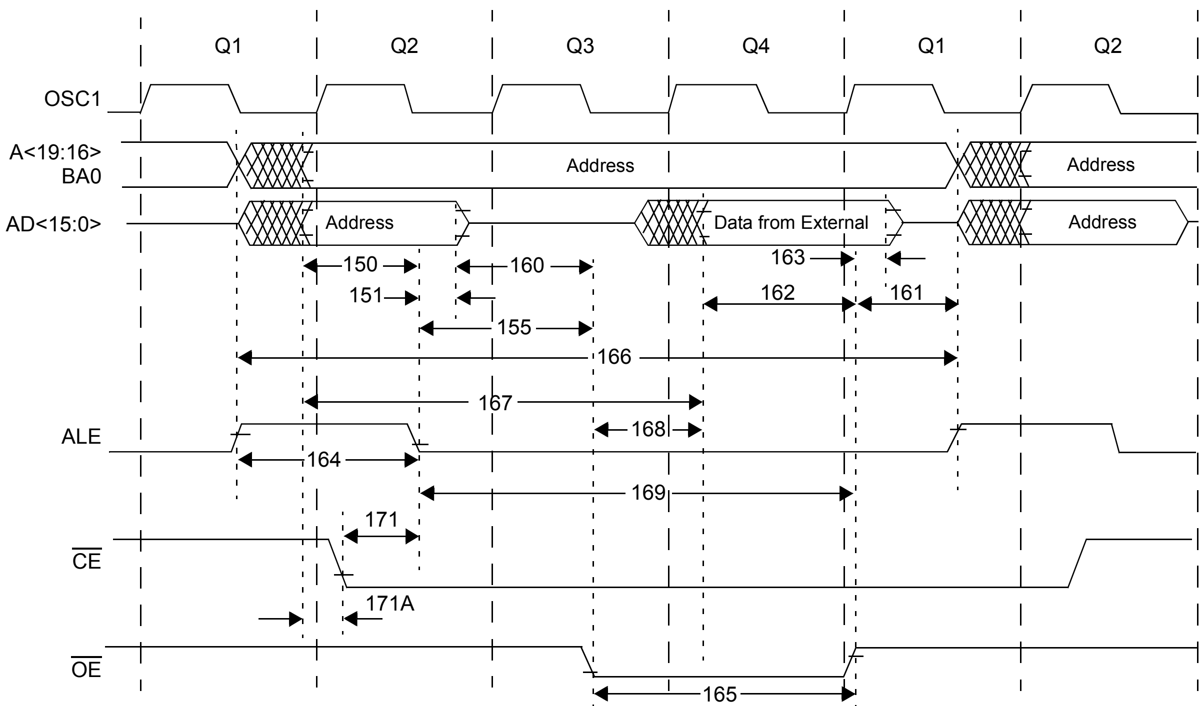

**Note:** FMAX = 25 MHz in 8-Bit External Memory mode.

**TABLE 28-10: PROGRAM MEMORY FETCH TIMING REQUIREMENTS (8-BIT)**

| Param No | Symbol | Characteristics | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| 150 | TadV2aIL | Address Out Valid to ALE ↓ (address setup time) | 0.25 TCY – 10 | — | — | ns |
| 151 | TaIL2adl | ALE ↓ to Address Out Invalid (address hold time) | 5 | — | — | ns |
| 153 | BA01 | BA0 ↑ to Most Significant Data Valid | 0.125 TCY | — | — | ns |
| 154 | BA02 | BA0 ↓ to Least Significant Data Valid | 0.125 TCY | — | — | ns |
| 155 | TaIL2oeL | ALE ↓ to $\overline{\text{OE}}$ ↓ | 0.125 TCY | — | — | ns |
| 161 | ToeH2adD | $\overline{\text{OE}}$ ↑ to A/D Driven | 0.125 TCY – 5 | — | — | ns |
| 162 | TadV2oeH | Least Significant Data Valid Before $\overline{\text{OE}}$ ↑ (data setup time) | 20 | — | — | ns |
| 162A | TadV2oeH | Most Significant Data Valid Before $\overline{\text{OE}}$ ↑ (data setup time) | 0.25 TCY + 20 | — | — | ns |
| 163 | ToeH2adl | $\overline{\text{OE}}$ ↑ to Data in Invalid (data hold time) | 0 | — | — | ns |
| 166 | TaIH2aIH | ALE ↑ to ALE ↑ (cycle time) | — | TCY | — | ns |
| 167 | TACC | Address Valid to Data Valid | 0.5 TCY – 10 | — | — | ns |
| 168 | Toe | $\overline{\text{OE}}$ ↓ to Data Valid | — | — | 0.125 TCY + 5 | ns |
| 170 | TubH2oeH | BA0 = 0 Valid Before $\overline{\text{OE}}$ ↑ | 0.25 TCY | — | — | ns |
| 170A | TubL2oeH | BA0 = 1 Valid Before $\overline{\text{OE}}$ ↑ | 0.5 TCY | — | — | ns |

**FIGURE 28-7: PROGRAM MEMORY READ TIMING DIAGRAM**

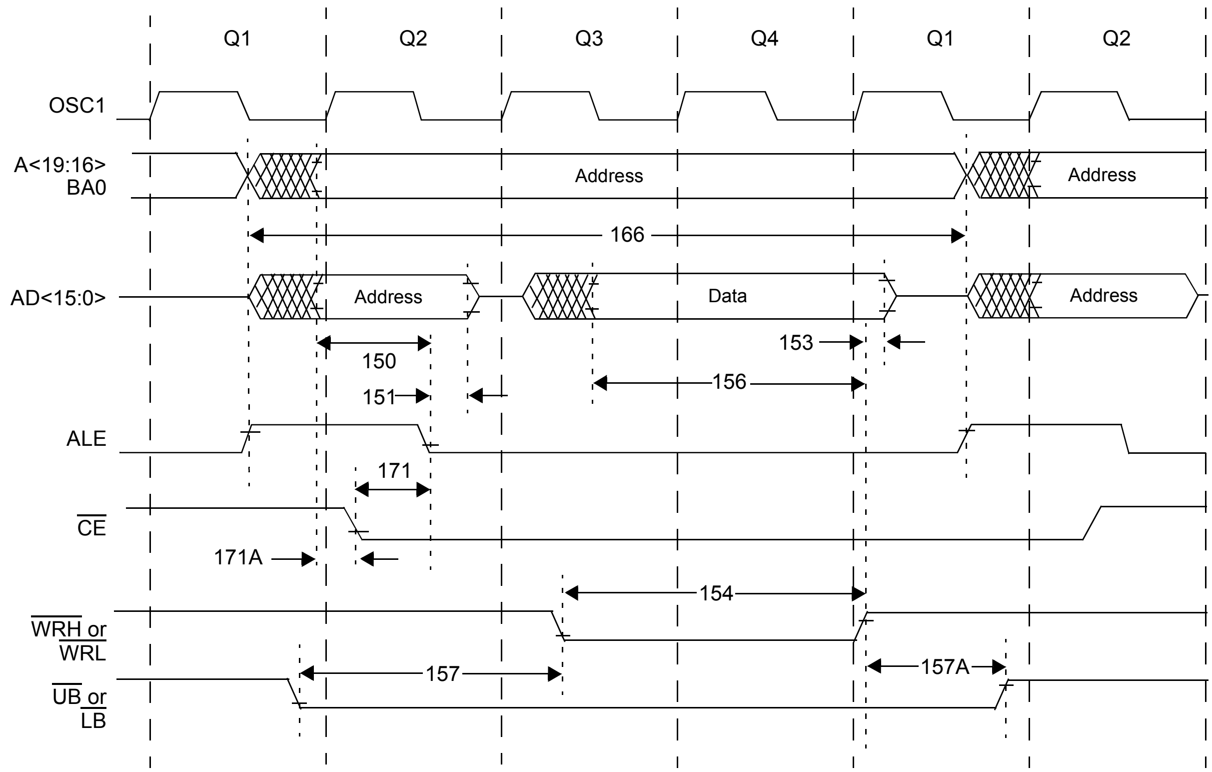



Operating Conditions: 2.0V < VCC < 3.6V, -40°C < TA < +125°C unless otherwise stated.

**TABLE 28-11: PROGRAM MEMORY READ TIMING REQUIREMENTS**

| Param. No | Symbol | Characteristics | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| 150 | TadV2alL | Address Out Valid to ALE ↓ (address setup time) | 0.25 TCY – 10 | — | — | ns |
| 151 | TalL2adl | ALE ↓ to Address Out Invalid (address hold time) | 5 | — | — | ns |
| 155 | TalL2oeL | ALE ↓ to OE ↓ | 10 | 0.125 TCY | — | ns |
| 160 | TadZ2oeL | AD high-Z to OE ↓ (bus release to OE) | 0 | — | — | ns |
| 161 | ToeH2adD | OE ↑ to AD Driven | 0.125 TCY – 5 | — | — | ns |
| 162 | TadV2oeH | Least Significant Data Valid Before OE ↑ (data setup time) | 20 | — | — | ns |
| 163 | ToeH2adl | OE ↑ to Data In Invalid (data hold time) | 0 | — | — | ns |
| 164 | TalH2alL | ALE Pulse Width | — | 0.25 TCY | — | ns |
| 165 | ToeL2oeH | OE Pulse Width | 0.5 TCY – 5 | 0.5 TCY | — | ns |
| 166 | TalH2alH | ALE ↑ to ALE ↑ (cycle time) | — | TCY | — | ns |
| 167 | Tacc | Address Valid to Data Valid | 0.75 TCY – 25 | — | — | ns |
| 168 | Toe | OE ↓ to Data Valid | | — | 0.5 TCY – 25 | ns |
| 169 | TalL2oeH | ALE ↓ to OE ↑ | 0.625 TCY – 10 | — | 0.625 TCY + 10 | ns |
| 171 | TalH2csL | Chip Enable Active to ALE ↓ | 0.25 TCY – 20 | — | — | ns |
| 171A | TubL2oeH | AD Valid to Chip Enable Active | — | — | 10 | ns |

**FIGURE 28-8: PROGRAM MEMORY WRITE TIMING DIAGRAM**

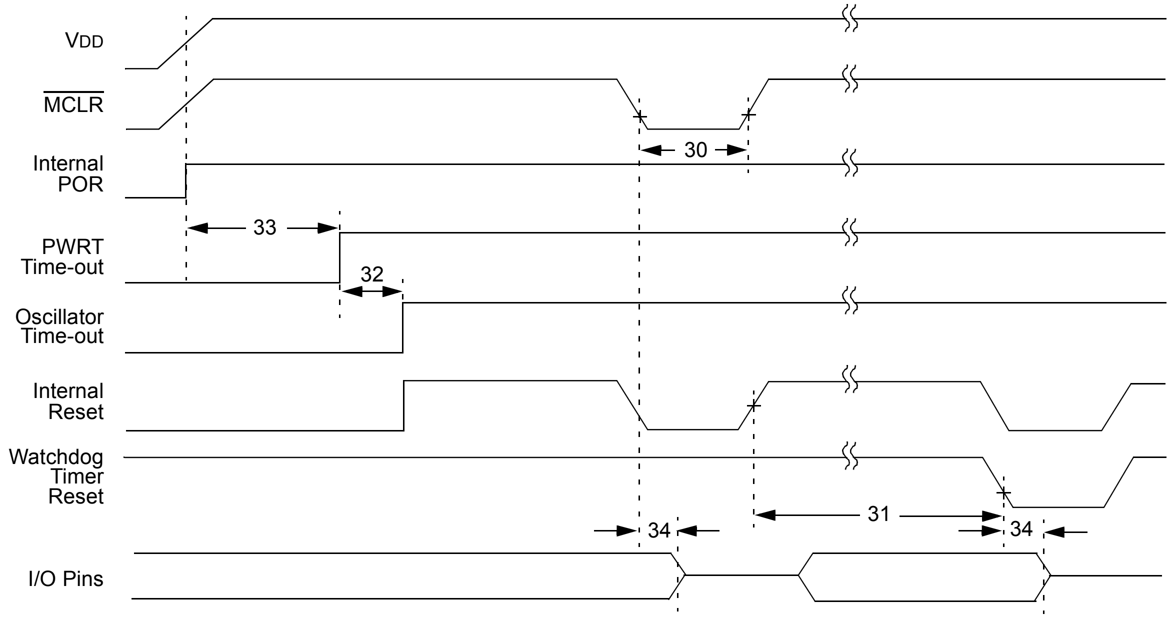



Operating Conditions: 2.0V < VCC < 3.6V, -40°C < TA < +125°C unless otherwise stated.

**TABLE 28-12: PROGRAM MEMORY WRITE TIMING REQUIREMENTS**

| Param. No | Symbol | Characteristics | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| 150 | TadV2alL | Address Out Valid to ALE ↓ (address setup time) | 0.25 TCY – 10 | — | — | ns |
| 151 | TalL2adl | ALE ↓ to Address Out Invalid (address hold time) | 5 | — | — | ns |
| 153 | TwrH2adl | $\overline{\text{WRn}}$ ↑ to Data Out Invalid (data hold time) | 5 | — | — | ns |
| 154 | TwrL | $\overline{\text{WRn}}$ Pulse Width | 0.5 TCY – 5 | 0.5 TCY | — | ns |
| 156 | TadV2wrH | Data Valid Before $\overline{\text{WRn}}$ ↑ (data setup time) | 0.5 TCY – 10 | — | — | ns |
| 157 | TbsV2wrL | Byte Select Valid Before $\overline{\text{WRn}}$ ↓ (byte select setup time) | 0.25 TCY | — | — | ns |
| 157A | TwrH2bsI | $\overline{\text{WRn}}$ ↑ to Byte Select Invalid (byte select hold time) | 0.125 TCY – 5 | — | — | ns |
| 166 | TalH2alH | ALE ↑ to ALE ↑ (cycle time) | — | TCY | — | ns |
| 171 | TalH2csL | Chip Enable Active to ALE ↓ | 0.25 TCY – 20 | — | — | ns |
| 171A | TubL2oeH | AD Valid to Chip Enable Active | — | — | 10 | ns |

**FIGURE 28-9: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**




**Note:** Refer to Figure 28-3 for load conditions.

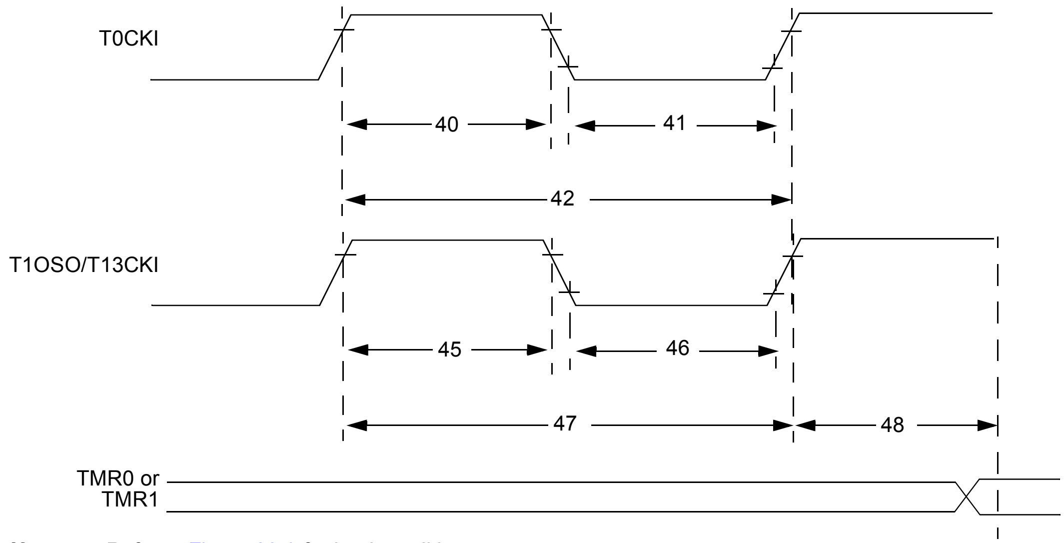
**TABLE 28-13: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 30 | TMCL | MCLR Pulse Width (low) | 2 | — | — | TCY | **(Note 1)** |
| 31 | TWDT | Watchdog Timer Time-out Period (no postscaler) | 3.4 | 4.0 | 4.6 | ms | |
| 32 | TOST | Oscillator Start-up Timer Period | 1024 TOSC | — | 1024 TOSC | — | TOSC = OSC1 period |
| 33 | TPWRT | Power-up Timer Period | 45.8 | 65.5 | 85.2 | ms | |
| 34 | TIOZ | I/O High-Impedance from MCLR Low or Watchdog Timer Reset | — | 2 | — | μs | |
| 38 | TCSD | CPU Start-up Time | — | 200 | — | μs | |

**Note 1:** To ensure a device Reset, MCLR must be low for at least 2 TCY or 400 μs, whichever is lower.

**TABLE 28-14: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**

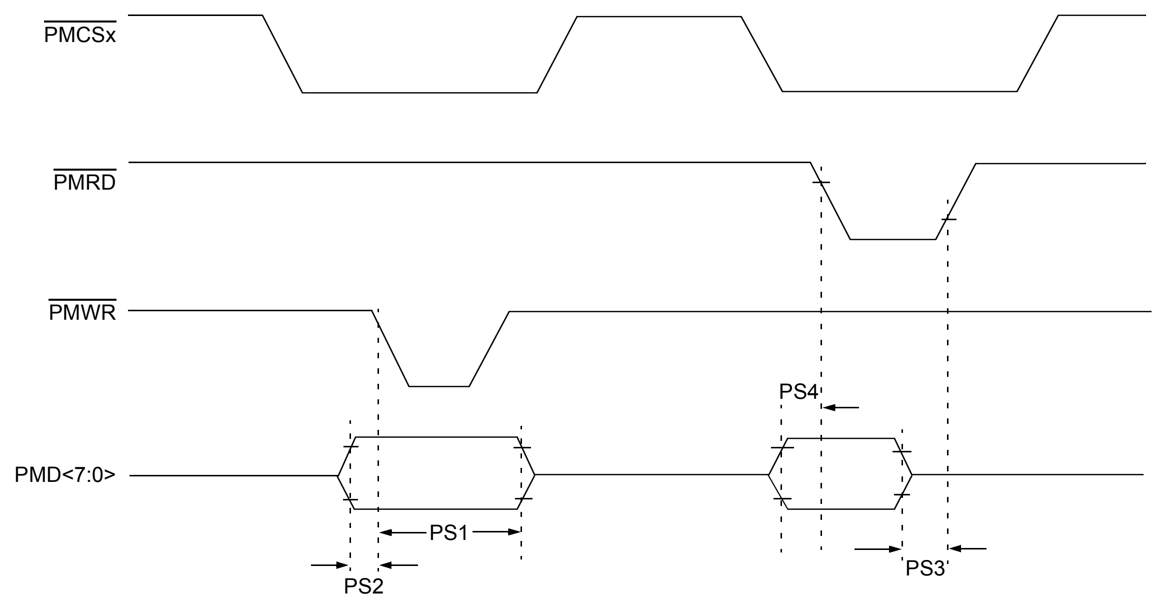



**Note:** Refer to Figure 28-3 for load conditions.

**TABLE 28-15: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 40 | TT0H | T0CKI High Pulse Width | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | 10 | — | ns | |
| 41 | TT0L | T0CKI Low Pulse Width | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | 10 | — | ns | |
| 42 | TT0P | T0CKI Period | No prescaler | TCY + 10 | — | ns | |
| | | | With prescaler | Greater of: 20 ns or (TCY + 40)/N | — | ns | N = prescale value (1, 2, 4,..., 256) |
| 45 | TT1H | T13CKI High Time | Synchronous, no prescaler | 0.5 TCY + 20 | — | ns | |
| | | | Synchronous, with prescaler | 10 | — | ns | |
| | | | Asynchronous | 30 | — | ns | |
| 46 | TT1L | T13CKI Low Time | Synchronous, no prescaler | 0.5 TCY + 5 | — | ns | |
| | | | Synchronous, with prescaler | 10 | — | ns | |
| | | | Asynchronous | 30 | — | ns | |
| 47 | TT1P | T13CKI Input Period | Synchronous | Greater of: 20 ns or (TCY + 40)/N | — | ns | N = prescale value (1, 2, 4, 8) |
| | | | Asynchronous | 60 | — | ns | |
| | FT1 | T13CKI Oscillator Input Frequency Range | | DC | 50 | kHz | |
| 48 | TCKE2TMRI | Delay from External T13CKI Clock Edge to Timer Increment | | 2 TOSC | 7 TOSC | — | |

**FIGURE 28-10: PARALLEL SLAVE PORT TIMING**

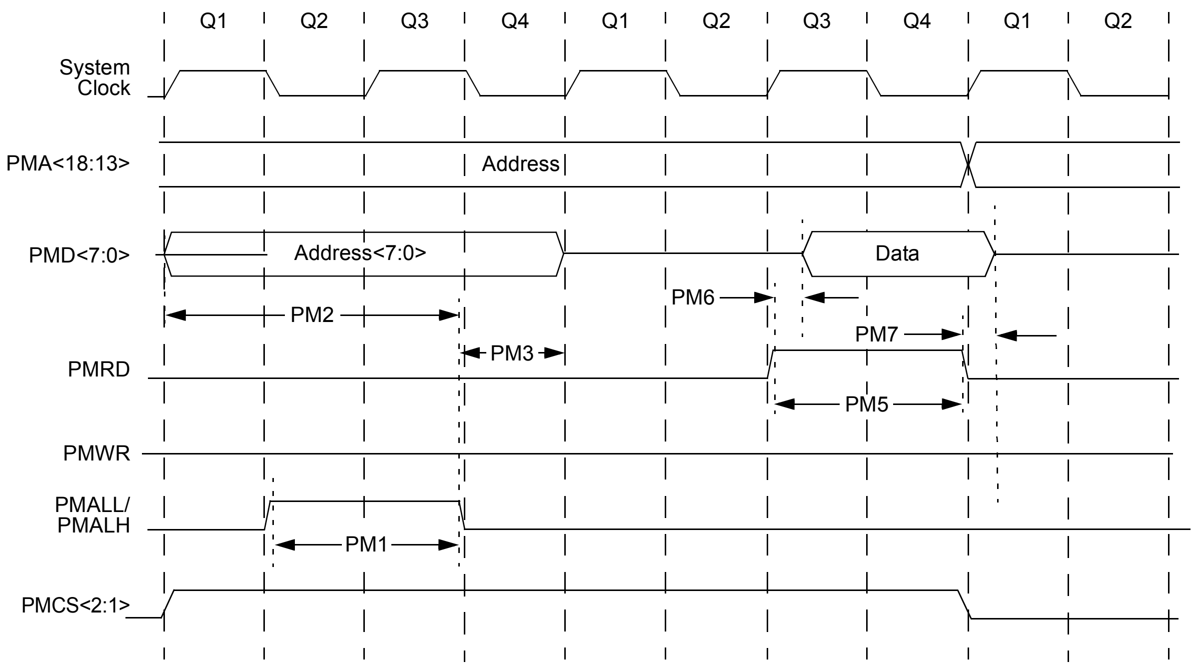



**Note:** Refer to Figure 28-3 for load conditions.

**TABLE 28-16: PARALLEL SLAVE PORT REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| PS1 | TdtV2wrH | Data In Valid Before $\overline{PMWR}$ or $\overline{PMCSx}$ Inactive (setup time) | 20 | — | ns | |
| PS2 | TwrH2dtI | $\overline{PMWR}$ or $\overline{PMCSx}$ Inactive to Data–In Invalid (hold time) | 20 | — | ns | |
| PS3 | TrdL2dtV | $\overline{PMRD}$ and $\overline{PMCSx}$ Active to Data–Out Valid | — | 80 | ns | |
| PS4 | TrdH2dtI | $\overline{PMRD}$ Active or $\overline{PMCSx}$ Inactive to Data–Out Invalid | 10 | 30 | ns | |

**FIGURE 28-11: PARALLEL MASTER PORT READ TIMING DIAGRAM**

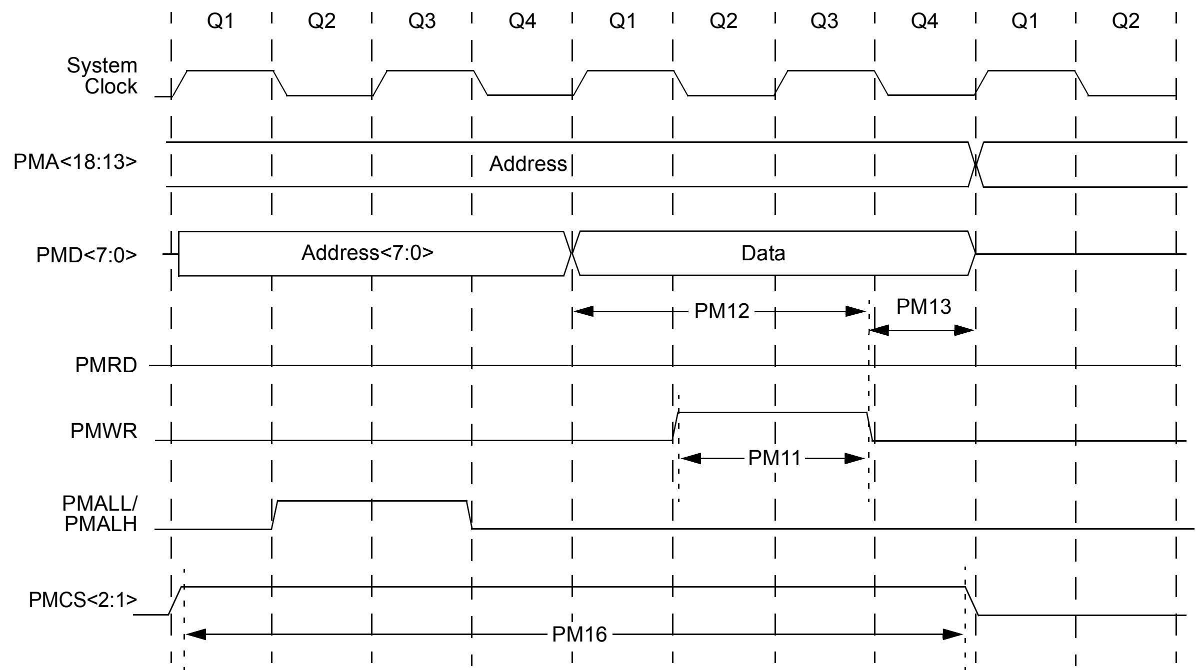



Operating Conditions: 2.0V < VCC < 3.6V, -40°C < TA < +85°C unless otherwise stated.

**TABLE 28-17: PARALLEL MASTER PORT READ TIMING REQUIREMENTS**

| Param. No | Symbol | Characteristics | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| PM1 | | PMALL/PMALH Pulse Width | — | 0.5 TCY | — | ns |
| PM2 | | Address Out Valid to PMALL/PMALH Invalid (address setup time) | — | 0.75 TCY | — | ns |
| PM3 | | PMALL/PMALH Invalid to Address Out Invalid (address hold time) | — | 0.25 TCY | — | ns |
| PM5 | | PMRD Pulse Width | — | 0.5 TCY | — | ns |
| PM6 | | PMRD or PMENB Active to Data In Valid (data setup time) | — | — | — | ns |
| PM7 | | PMRD or PMENB Inactive to Data In Invalid (data hold time) | — | — | — | ns |

**FIGURE 28-12: PARALLEL MASTER PORT WRITE TIMING DIAGRAM**



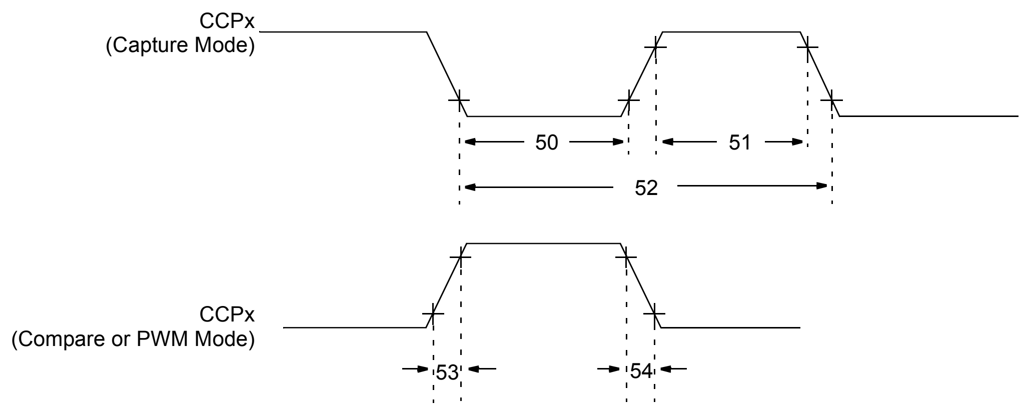

Operating Conditions: 2.0V < VCC < 3.6V, -40°C < TA < +85°C unless otherwise stated.

**TABLE 28-18: PARALLEL MASTER PORT WRITE TIMING REQUIREMENTS**

| Param. No | Symbol | Characteristics | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| PM11 | | PMWR Pulse Width | — | 0.5 TCY | — | ns |
| PM12 | | Data Out Valid before PMWR or PMENB Goes Inactive (data setup time) | — | — | — | ns |
| PM13 | | PMWR or PMEMB Invalid to Data Out Invalid (data hold time) | — | — | — | ns |
| PM16 | | PMCSx Pulse Width | TCY – 5 | — | — | ns |

**FIGURE 28-13: CAPTURE/COMPARE/PWM TIMINGS (INCLUDING ECCP MODULES)**

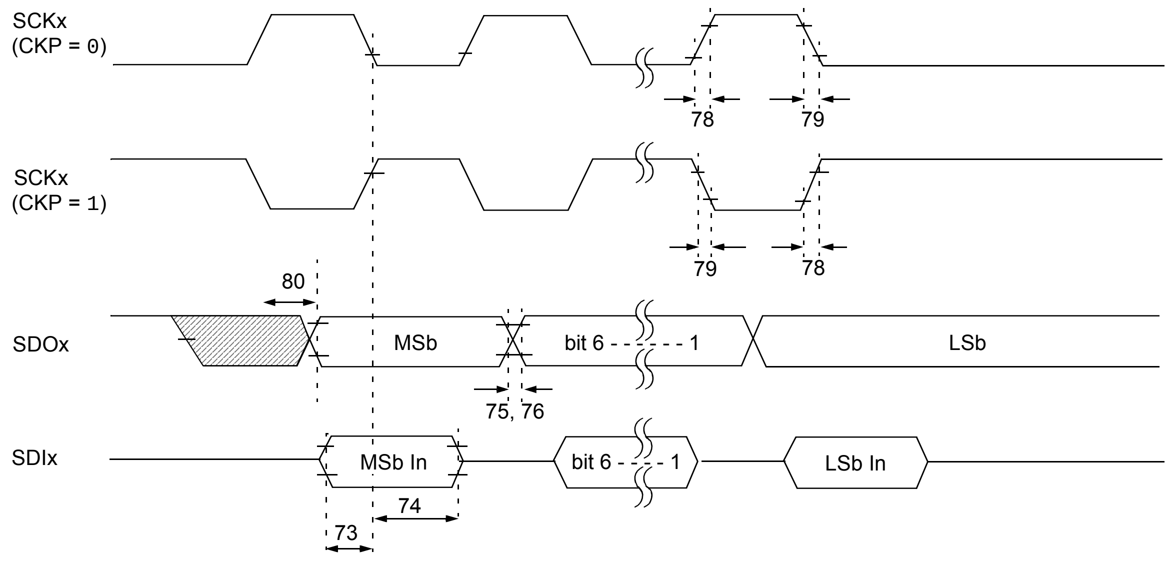



**Note:** Refer to Figure 28-3 for load conditions.

**TABLE 28-19: CAPTURE/COMPARE/PWM REQUIREMENTS (INCLUDING ECCP MODULES)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 50 | TCCL | CCPx Input Low Time | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | 10 | — | ns | |
| 51 | TCCH | CCPx Input High Time | No prescaler | 0.5 TCY + 20 | — | ns | |
| | | | With prescaler | 10 | — | ns | |
| 52 | TCCP | CCPx Input Period | | $\frac{3\ T_{CY} + 40}{N}$ | — | ns | N = prescale value (1, 4 or 16) |
| 53 | TCCR | CCPx Output Fall Time | | — | 25 | ns | |
| 54 | TCCF | CCPx Output Fall Time | | — | 25 | ns | |

**FIGURE 28-14: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)**

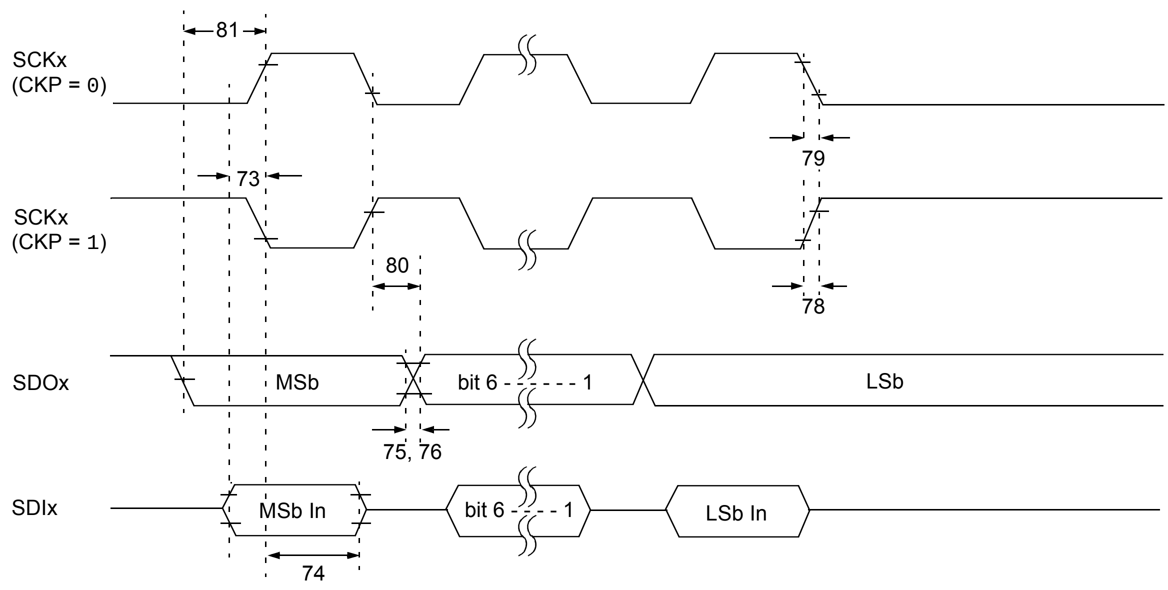



**Note:** Refer to Figure 28-3 for load conditions.

**TABLE 28-20: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)**

| Param No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 73 | TDIV2SCH, TDIV2SCL | Setup Time of SDIx Data Input to SCKx Edge | 100 | — | ns | |
| 75 | TDOR | SDOx Data Output Rise Time | — | 25 | ns | |
| 76 | TDOF | SDOx Data Output Fall Time | — | 25 | ns | |
| 78 | TSCR | SCKx Output Rise Time | — | 25 | ns | |
| 79 | TSCF | SCKx Output Fall Time | — | 25 | ns | |
| 80 | TSCH2DOV, TSCL2DOV | SDOx Data Output Valid after SCKx Edge | — | 50 | ns | |

**FIGURE 28-15: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)**

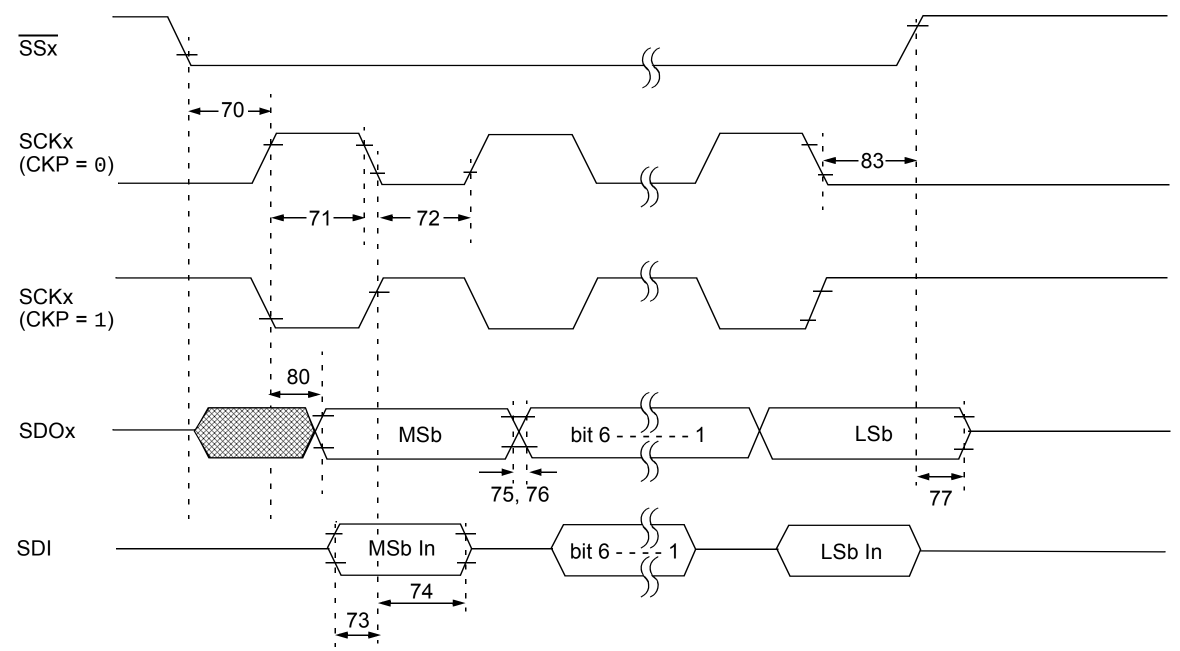



**Note:** Refer to Figure 28-3 for load conditions.

**TABLE 28-21: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)**

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 73 | TDIV2SCH, TDIV2SCL | Setup Time of SDIx Data Input to SCKx Edge | 100 | — | ns | |
| 74 | TSCH2DIL, TSCL2DIL | Hold Time of SDIx Data Input to SCKx Edge | 100 | — | ns | |
| 75 | TDOR | SDOx Data Output Rise Time | — | 25 | ns | |
| 76 | TDOF | SDOx Data Output Fall Time | — | 25 | ns | |
| 78 | TSCR | SCKx Output Rise Time | — | 25 | ns | |
| 79 | TSCF | SCKx Output Fall Time | — | 25 | ns | |
| 80 | TSCH2DOV, TSCL2DOV | SDOx Data Output Valid after SCKx Edge | — | 50 | ns | |
| 81 | TDOV2SCH, TDOV2SCL | SDOx Data Output Setup to SCKx Edge | TCY | — | ns | |

**FIGURE 28-16: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)**




**Note:** Refer to Figure 28-3 for load conditions.

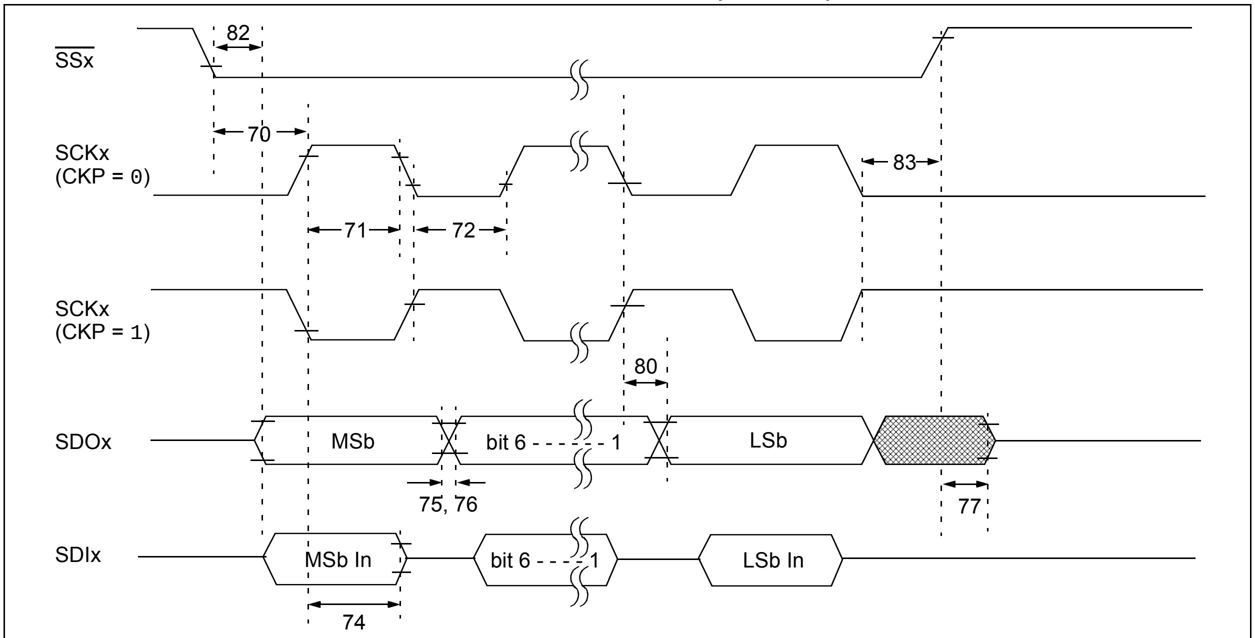
**TABLE 28-22: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 70 | TSSL2SCH, TSSL2SCL | $\overline{SSx}$ ↓ to SCKx ↓ or SCKx ↑ Input | | 3 TCY | — | ns | |
| 70A | TSSL2WB | $\overline{SSx}$ ↓ to write to SSPxBUF | | 3 TCY | — | ns | |
| 71 | TSCH | SCKx Input High Time | Continuous | 1.25 TCY + 30 | — | ns | |
| 71A | | | Single byte | 40 | — | ns | **(Note 1)** |
| 72 | TSCL | SCKx Input Low Time | Continuous | 1.25 TCY + 30 | — | ns | |
| 72A | | | Single byte | 40 | — | ns | **(Note 1)** |
| 73 | TDIV2SCH, TDIV2SCL | Setup Time of SDIx Data Input to SCKx Edge | | 25 | — | ns | |
| 73A | TB2B | Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2 | | 1.5 TCY + 40 | — | ns | **(Note 2)** |
| 74 | TSCH2DIL, TSCL2DIL | Hold Time of SDIx Data Input to SCKx Edge | | 35 | — | ns | VDD = 3.3V, VDDCORE = 2.5V |
| | | | | 100 | — | ns | VDD = 2.15V |
| 75 | TDOR | SDOx Data Output Rise Time | | — | 25 | ns | |
| 76 | TDOF | SDOx Data Output Fall Time | | — | 25 | ns | |
| 77 | TSSH2DOZ | $\overline{SSx}$ ↑ to SDOx Output High-Impedance | | 10 | 50 | ns | |
| 80 | TSCH2DOV, TSCL2DOV | SDOx Data Output Valid after SCKx Edge | | — | 50 | ns | |
| 83 | TSCH2SSH, TSCL2SSH | $\overline{SSx}$ ↑ after SCKx Edge | | 1.5 TCY + 40 | — | ns | |

**Note 1:** Requires the use of Parameter #73A.
**2:** Only if Parameter #71A and #72A are used.

**FIGURE 28-17: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)**




**Note:** Refer to Figure 28-3 for load conditions.

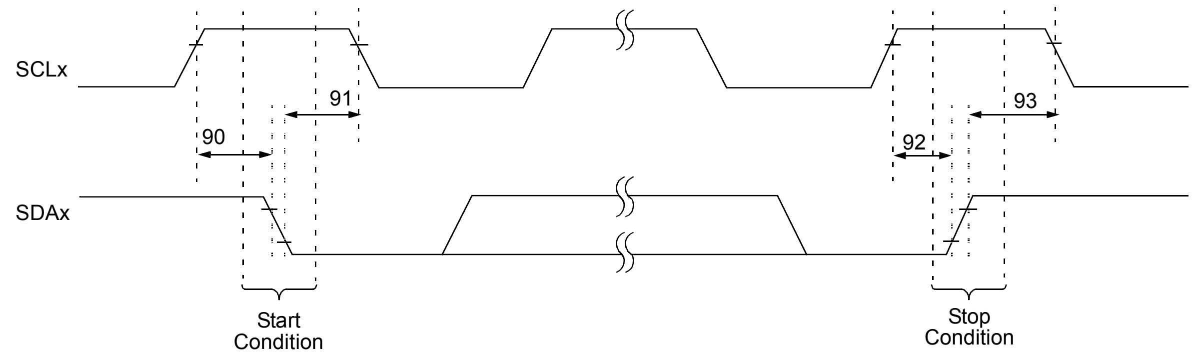
**TABLE 28-23: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)**

| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 70 | TSSL2SCH, TSSL2SCL | $\overline{SSx}$ ↓ to SCKx ↓ or SCKx ↑ Input | | 3 TCY | — | ns | |
| 70A | TSSL2WB | $\overline{SSx}$ ↓ to Write to SSPxBUF | | 3 TCY | — | ns | |
| 71 | TSCH | SCKx Input High Time | Continuous | 1.25 TCY + 30 | — | ns | |
| 71A | | | Single byte | 40 | — | ns | **(Note 1)** |
| 72 | TSCL | SCKx Input Low Time | Continuous | 1.25 TCY + 30 | — | ns | |
| 72A | | | Single byte | 40 | — | ns | **(Note 1)** |
| 73 | TDIV2SCH, TDIV2SCL | Setup Time of SDIx Data Input to SCKx Edge | | 25 | — | ns | |
| 73A | TB2B | Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2 | | 1.5 TCY + 40 | — | ns | **(Note 2)** |
| 74 | TSCH2DIL, TSCL2DIL | Hold Time of SDIx Data Input to SCKx Edge | | 35 | — | ns | VDD = 3.3V, VDDCORE = 2.5V |
| | | | | 100 | — | ns | VDD = 2.15V |
| 75 | TDOR | SDOx Data Output Rise Time | | — | 25 | ns | |
| 76 | TDOF | SDOx Data Output Fall Time | | — | 25 | ns | |
| 77 | TSSH2DOZ | $\overline{SSx}$ ↑ to SDOx Output High-Impedance | | 10 | 50 | ns | |
| 80 | TSCH2DOV, TSCL2DOV | SDOx Data Output Valid After SCKx Edge | | — | 50 | ns | |
| 82 | TSSL2DOV | SDOx Data Output Valid After $\overline{SSx}$ ↓ Edge | | — | 50 | ns | |
| 83 | TSCH2SSH, TSCL2SSH | $\overline{SSx}$ ↑ After SCKx Edge | | 1.5 TCY + 40 | — | ns | |

**Note 1:** Requires the use of Parameter #73A.

**2:** Only if Parameter #71A and #72A are used.

**FIGURE 28-18: I²C™ BUS START/STOP BITS TIMING**

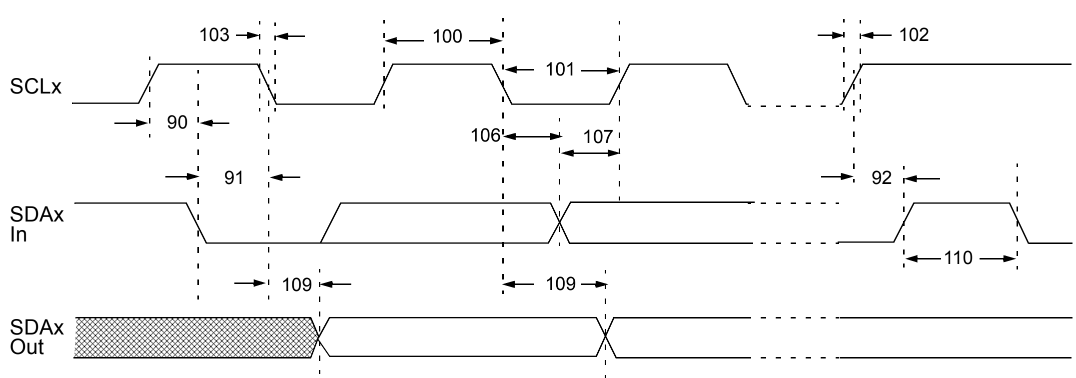



**Note:** Refer to Figure 28-3 for load conditions.

**TABLE 28-24: I²C™ BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 4700 | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 600 | — | | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 4000 | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 600 | — | | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 4700 | — | ns | |
| | | | 400 kHz mode | 600 | — | | |
| 93 | THD:STO | Stop Condition Hold Time | 100 kHz mode | 4000 | — | ns | |
| | | | 400 kHz mode | 600 | — | | |

**FIGURE 28-19: I²C™ BUS DATA TIMING**




**Note:** Refer to Figure 28-3 for load conditions.

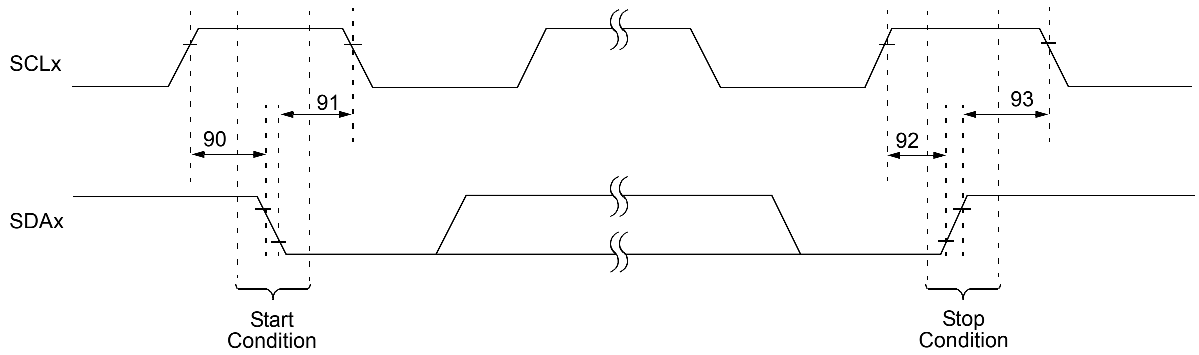
## TABLE 28-25: I²C™ BUS DATA REQUIREMENTS (SLAVE MODE)

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 4.0 | — | μs | |
| | | | 400 kHz mode | 0.6 | — | μs | |
| | | | MSSP modules | 1.5 TCY | — | | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 4.7 | — | μs | |
| | | | 400 kHz mode | 1.3 | — | μs | |
| | | | MSSP modules | 1.5 TCY | — | | |
| 102 | TR | SDAx and SCLx Rise Time | 100 kHz mode | — | 1000 | ns | |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | CB is specified to be from 10 to 400 pF |
| 103 | TF | SDAx and SCLx Fall Time | 100 kHz mode | — | 300 | ns | |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | CB is specified to be from 10 to 400 pF |
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 4.7 | — | μs | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 0.6 | — | μs | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 4.0 | — | μs | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 0.6 | — | μs | |
| 106 | THD:DAT | Data Input Hold Time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | μs | |
| 107 | TSU:DAT | Data Input Setup Time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 4.7 | — | μs | |
| | | | 400 kHz mode | 0.6 | — | μs | |
| 109 | TAA | Output Valid from Clock | 100 kHz mode | — | 3500 | ns | **(Note 1)** |
| | | | 400 kHz mode | — | — | ns | |
| 110 | TBUF | Bus Free Time | 100 kHz mode | 4.7 | — | μs | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | μs | |
| D102 | CB | Bus Capacitive Loading | | — | 400 | pF | |

**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.

**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 28-20: MSSPx I²C™ BUS START/STOP BITS TIMING WAVEFORMS**



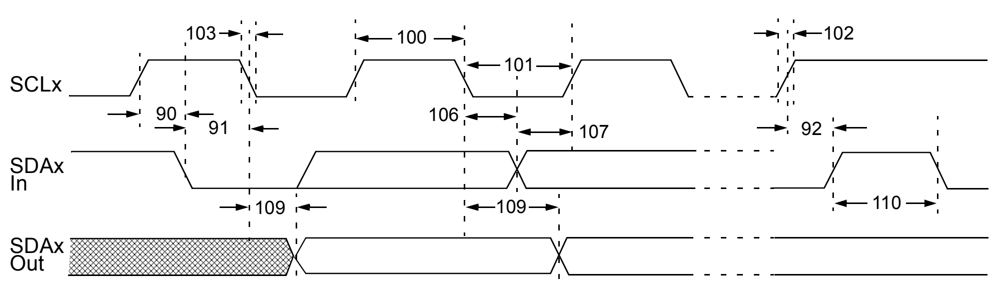

**Note:** Refer to Figure 28-3 for load conditions.

**TABLE 28-26: MSSPx I²C™ BUS START/STOP BITS REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |
| 93 | THD:STO | Stop Condition Hold Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ns | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | | |

**Note 1:** Maximum pin capacitance = 10 pF for all I²C™ pins.

**FIGURE 28-21: MSSPx I²C™ BUS DATA TIMING**




**Note:** Refer to Figure 28-3 for load conditions.

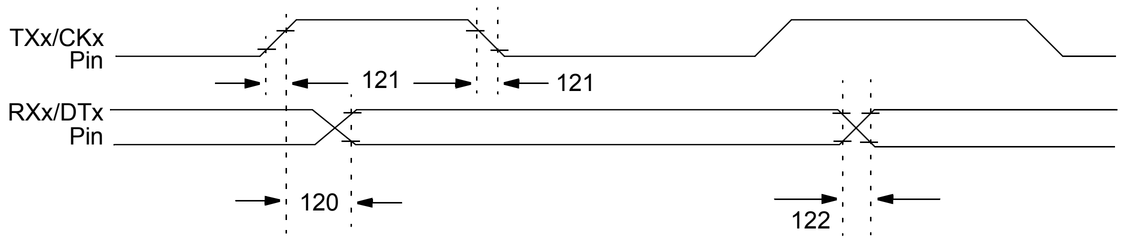
**TABLE 28-27: MSSPx I²C™ BUS DATA REQUIREMENTS**

| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 102 | TR | SDAx and SCLx Rise Time | 100 kHz mode | — | 1000 | ns | CB is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | |
| | | | 1 MHz mode(1) | — | 300 | ns | |
| 103 | TF | SDAx and SCLx Fall Time | 100 kHz mode | — | 300 | ns | CB is specified to be from 10 to 400 pF |
| | | | 400 kHz mode | 20 + 0.1 CB | 300 | ns | |
| | | | 1 MHz mode(1) | — | 100 | ns | |
| 90 | TSU:STA | Start Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 91 | THD:STA | Start Condition Hold Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 106 | THD:DAT | Data Input Hold Time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | ms | |
| | | | 1 MHz mode(1) | — | — | ns | |
| 107 | TSU:DAT | Data Input Setup Time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| | | | 1 MHz mode(1) | — | — | ns | |
| 92 | TSU:STO | Stop Condition Setup Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | ms | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | ms | |
| 109 | TAA | Output Valid from Clock | 100 kHz mode | — | 3500 | ns | |
| | | | 400 kHz mode | — | 1000 | ns | |
| | | | 1 MHz mode(1) | — | — | ns | |
| 110 | TBUF | Bus Free Time | 100 kHz mode | 4.7 | — | ms | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | ms | |
| | | | 1 MHz mode(1) | — | — | ms | |
| D102 | CB | Bus Capacitive Loading | | — | 400 | pF | |

**Note 1:** Maximum pin capacitance = 10 pF for all I²C™ pins.

**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 28-22: EUSARTx SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

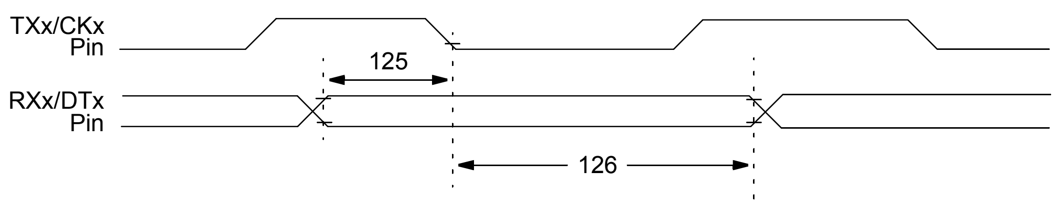



**Note:** Refer to Figure 28-3 for load conditions.

**TABLE 28-28: EUSARTx SYNCHRONOUS TRANSMISSION REQUIREMENTS**

| Param No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 120 | TCKH2DTV | SYNC XMIT (MASTER and SLAVE)<br>Clock High to Data Out Valid | — | 40 | ns | |
| 121 | TCKRF | Clock Out Rise Time and Fall Time (Master mode) | — | 20 | ns | |
| 122 | TDTRF | Data Out Rise Time and Fall Time | — | 20 | ns | |

**FIGURE 28-23: EUSARTx SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**




**Note:** Refer to Figure 28-3 for load conditions.

**TABLE 28-29: EUSARTx SYNCHRONOUS RECEIVE REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 125 | TDTV2CKL | SYNC RCV (MASTER and SLAVE)<br>Data Hold Before CKx ↓ (DTx hold time) | 10 | — | ns | |
| 126 | TCKL2DTL | Data Hold After CKx ↓ (DTx hold time) | 15 | — | ns | |

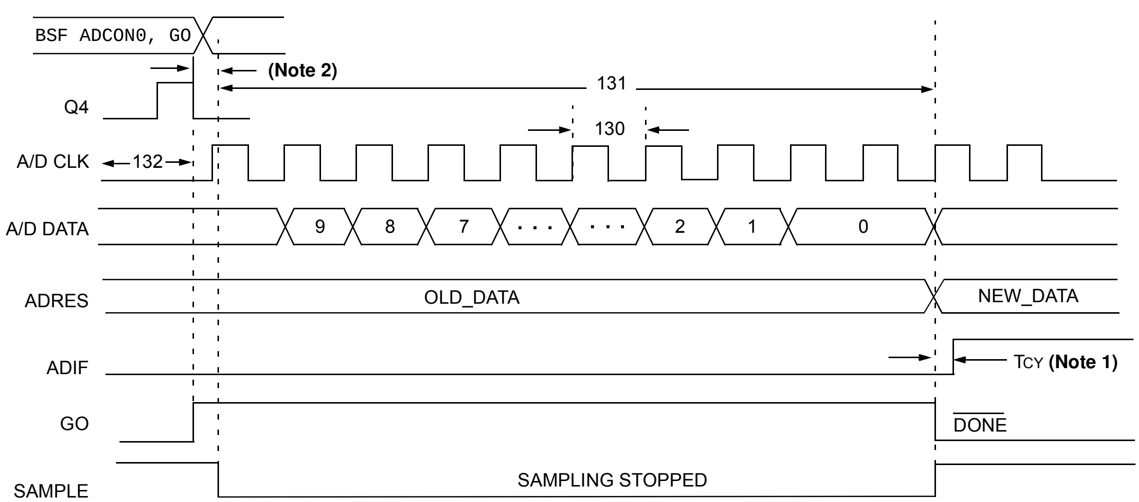
**TABLE 28-30: A/D CONVERTER CHARACTERISTICS: PIC18F87J11 FAMILY (INDUSTRIAL)**

| Param No. | Symbol | Characteristic | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| A01 | NR | Resolution | — | — | 10 | bit | $\Delta$VREF $\geq$ 3.0V |
| A03 | EIL | Integral Linearity Error | — | — | <±1 | LSb | $\Delta$VREF $\geq$ 3.0V |
| A04 | EDL | Differential Linearity Error | — | — | <±1 | LSb | $\Delta$VREF $\geq$ 3.0V |
| A06 | EOFF | Offset Error | — | — | <±3 | LSb | $\Delta$VREF $\geq$ 3.0V |
| A07 | EGN | Gain Error | — | — | <±3 | LSb | $\Delta$VREF $\geq$ 3.0V |
| A10 | — | Monotonicity | Guaranteed(1) | | | — | VSS $\leq$ VAIN $\leq$ VREF |
| A20 | $\Delta$VREF | Reference Voltage Range (VREFH – VREFL) | 2.0<br>3 | —<br>— | —<br>— | V<br>V | VDD < 3.0V<br>VDD $\geq$ 3.0V |
| A21 | VREFH | Reference Voltage High | VSS + $\Delta$VREF | — | VDD | V | |
| A22 | VREFL | Reference Voltage Low | VSS – 0.3V | — | VDD – 3.0V | V | |
| A25 | VAIN | Analog Input Voltage | VREFL | — | VREFH | V | |
| A30 | ZAIN | Recommended Impedance of Analog Voltage Source | — | — | 2.5 | k$\Omega$ | |
| A50 | IREF | VREF Input Current(2) | —<br>— | —<br>— | 5<br>150 | $\mu$A<br>$\mu$A | During VAIN acquisition.<br>During A/D conversion cycle. |

**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 28-24: 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 28-31: A/D CONVERSION REQUIREMENTS**

| Param No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 130 | TAD | A/D Clock Period | 0.7 | 25.0[1] | μs | TOSC based, VREF ≥ 3.0V |
| | | | — | 1 | μs | A/D RC mode |
| 131 | TCNV | Conversion Time<br>(not including acquisition time) **(Note 2)** | 11 | 12 | TAD | |
| 132 | TACQ | Acquisition Time **(Note 3)** | 1.4 | — | μs | -40°C to +85°C |
| 135 | TSWC | Switching Time from Convert → Sample | — | **(Note 4)** | | |
| 136 | TDIS | Discharge Time | 0.2 | — | μs | |

**Note 1:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

**2:** The ADRES registers may be read on the following TCY cycle.

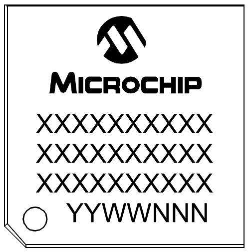
**3:** The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD). The source impedance (RS) on the input channels is 50Ω.

**4:** On the following cycle of the device clock.

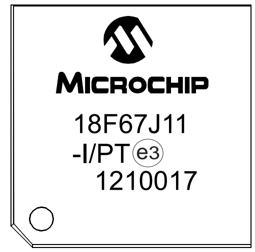
# 29.0 PACKAGING INFORMATION

## 29.1 Package Marking Information

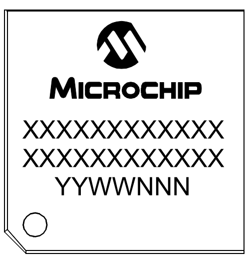
64-Lead TQFP



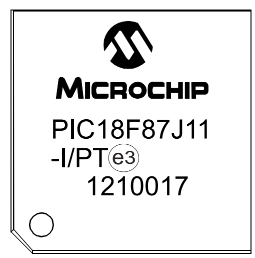

Example




80-Lead TQFP




Example




| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week '01') |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC designator for Matte Tin (Sn) |
| | * | This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package. |

**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

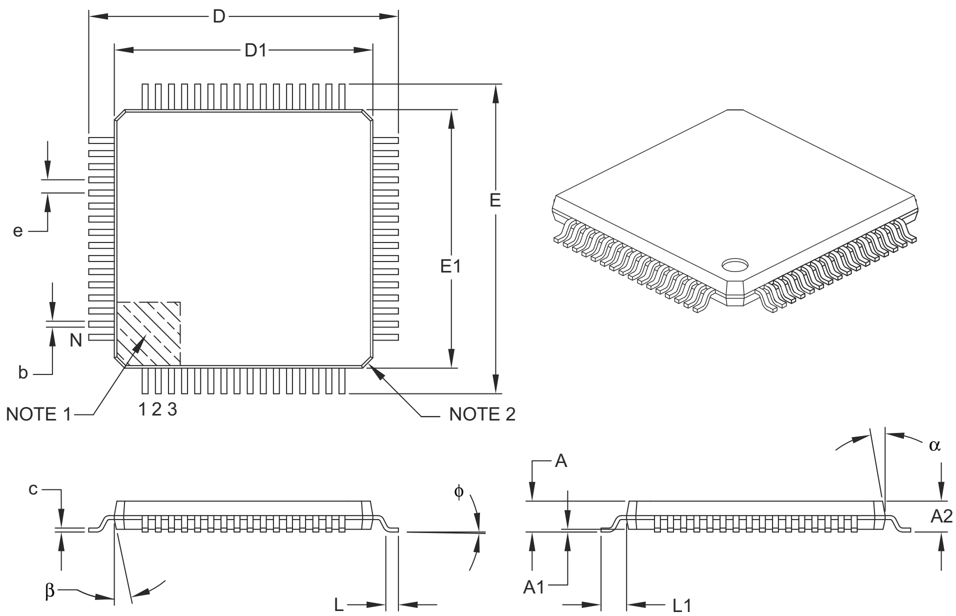
 

## 29.2 Package Details

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 |
|---|---|




| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Leads | N | 64 | | |
| Lead Pitch | e | 0.50 BSC | | |
| Overall Height | A | – | – | 1.20 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Standoff | A1 | 0.05 | – | 0.15 |
| Foot Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | 1.00 REF | | |
| Foot Angle | φ | 0° | 3.5° | 7° |
| Overall Width | E | 12.00 BSC | | |
| Overall Length | D | 12.00 BSC | | |
| Molded Package Width | E1 | 10.00 BSC | | |
| Molded Package Length | D1 | 10.00 BSC | | |
| Lead Thickness | c | 0.09 | – | 0.20 |
| Lead Width | b | 0.17 | 0.22 | 0.27 |
| Mold Draft Angle Top | α | 11° | 12° | 13° |
| Mold Draft Angle Bottom | β | 11° | 12° | 13° |

**Notes:**

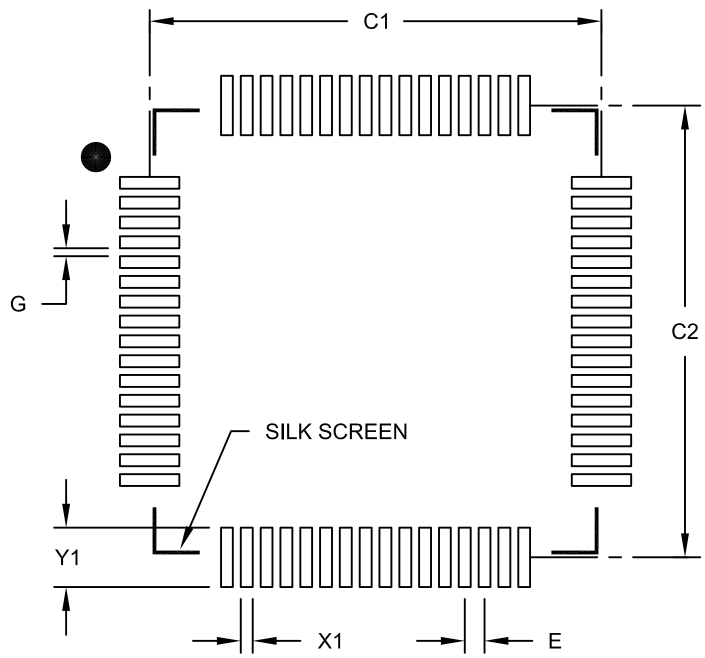
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME 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 Footprint [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.50 BSC | | |
| Contact Pad Spacing | C1 | | 11.40 | |
| Contact Pad Spacing | C2 | | 11.40 | |
| Contact Pad Width (X64) | X1 | | | 0.30 |
| Contact Pad Length (X64) | Y1 | | | 1.50 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

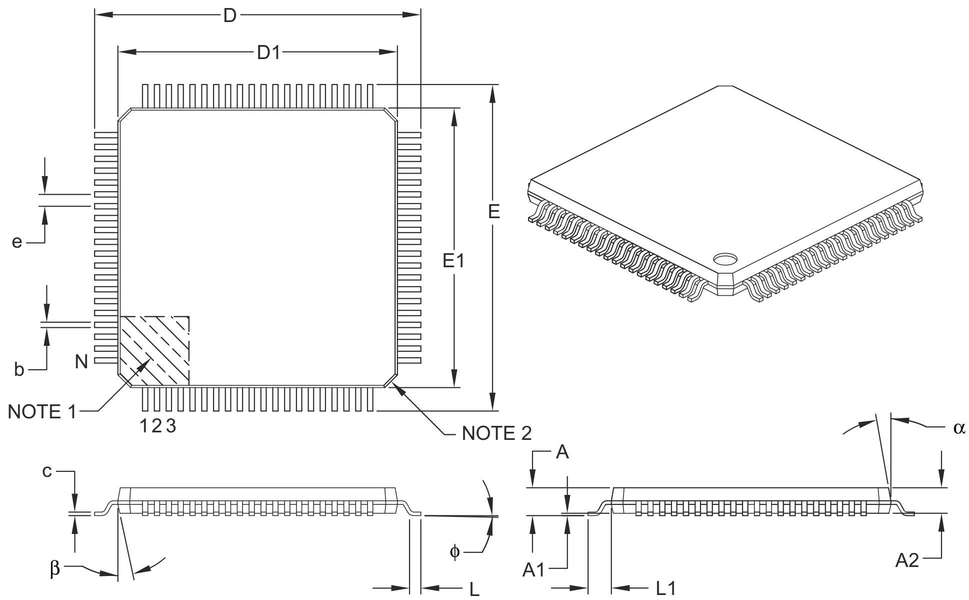
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2085B

## 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 |
|---|---|




| | | | | |
|---|---|---|---|---|
| Units | | MILLIMETERS | | |
| Dimension Limits | | MIN | NOM | MAX |
| Number of Leads | N | 80 | | |
| Lead Pitch | e | 0.50 BSC | | |
| Overall Height | A | – | – | 1.20 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Standoff | A1 | 0.05 | – | 0.15 |
| Foot Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | 1.00 REF | | |
| Foot Angle | φ | 0° | 3.5° | 7° |
| Overall Width | E | 14.00 BSC | | |
| Overall Length | D | 14.00 BSC | | |
| Molded Package Width | E1 | 12.00 BSC | | |
| Molded Package Length | D1 | 12.00 BSC | | |
| Lead Thickness | c | 0.09 | – | 0.20 |
| Lead Width | b | 0.17 | 0.22 | 0.27 |
| Mold Draft Angle Top | α | 11° | 12° | 13° |
| Mold Draft Angle Bottom | β | 11° | 12° | 13° |

**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.

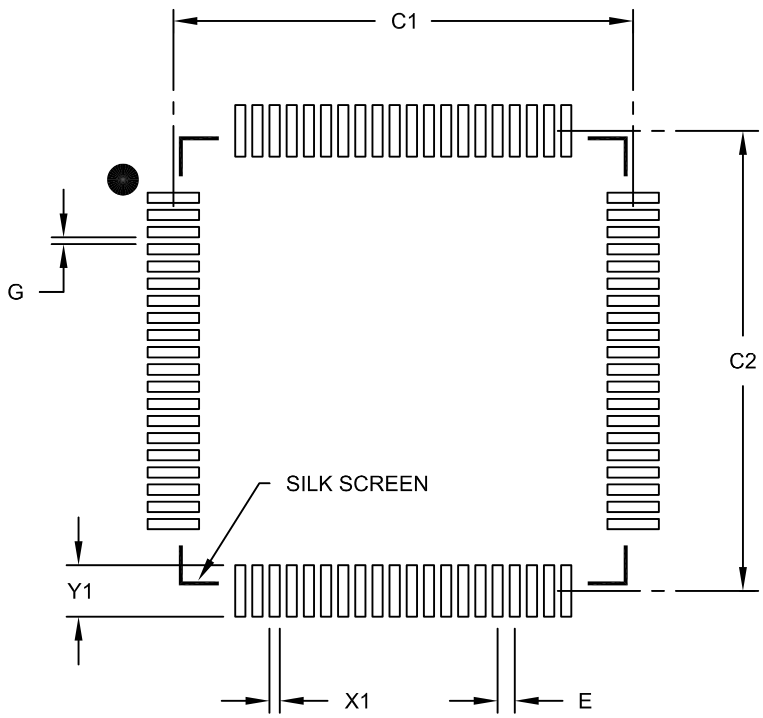
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-092B

80-Lead Plastic Thin Quad Flatpack (PT) - 12x12x1mm Body, 2.00 mm Footprint [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| | | Units | MILLIMETERS | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.50 BSC | | |
| Contact Pad Spacing | C1 | | 13.40 | |
| Contact Pad Spacing | C2 | | 13.40 | |
| Contact Pad Width (X80) | X1 | | | 0.30 |
| Contact Pad Length (X80) | Y1 | | | 1.50 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2092B

**NOTES:**

## 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.

### Revision D (October 2009)

Removed "Preliminary" marking.

### Revision E (June 2012)

Added **Section 2.0 "Guidelines for Getting Started with PIC18FJ Microcontrollers"**. Added all Data Sheet errata. Updated values in **Section 28.0 "Electrical Characteristics"**, and added Figure 28-6 and Table 28-10 for 8-bit EMB. Updated package drawings in **Section 29.0 "Packaging Information"**. Minor edits to text throughout the document.

## 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**

| Features | PIC18F66J11 | PIC18F66J16 | PIC18F67J11 | PIC18F86J11 | PIC18F86J16 | PIC18F87J11 |
|---|---|---|---|---|---|---|
| Program memory | 64K | 96K | 128K | 64K | 96K | 128K |
| Program Memory (Instructions) | 32764 | 49148 | 65532 | 32764 | 49148 | 65532 |
| I/O Ports | Ports A, B, C, D, E, F, G | | | Ports A, B, C, D, E, F, G, H, J | | |
| EMB | No | | | Yes | | |
| 10-Bit A/D module | 11 Input Channels | | | 15 Input Channels | | |
| Packages | 64-Pin TQFP | | | 80-Pin TQFP | | |

**NOTES:**

## 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. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support
- 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://microchip.com/support**

## 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

Total Pages Sent ________

From: Name ____________________
Company ____________________
Address ____________________
City / State / ZIP / Country ____________________
Telephone: (_______) _________ - _________
FAX: (______) _________ - _________

Application (optional):

Would you like a reply? ___Y ___N

Device: PIC18F87J11 Family

Literature Number: DS39778E

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?

____________________

____________________

# INDEX

## A



## B

## S



## T



## V



## W



## X

NOTES:

## PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| PART NO. | X | /XX | XXX |
|---|---|---|---|
| Device | Temperature Range | Package | Pattern |

| | |
|---|---|
| Device | PIC18F66J11/66J16/67J11[(1)], PIC18F86J11/86J16/87J11[(1)], PIC18F66J11/66J16/67J11T[(2)], PIC18F86J11/86J16/87J11T[(2)] |
| 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
**2:** T = in tape and reel

NOTES:

ISBN: 978-1-62076-346-9

# Worldwide Sales and Service

## AMERICAS

**Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

**Atlanta**
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Cleveland**
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

**Indianapolis**
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

**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**
Suites 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: 86-10-8569-7000
Fax: 86-10-8528-2104

**China - Chengdu**
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

**China - Chongqing**
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

**China - Hangzhou**
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

**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 - Xian**
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

**China - Xiamen**
Tel: 86-592-2388138
Fax: 86-592-2388130

**China - Zhuhai**
Tel: 86-756-3210040
Fax: 86-756-3210049

## ASIA/PACIFIC

**India - Bangalore**
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

**India - New Delhi**
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

**India - Pune**
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

**Japan - Osaka**
Tel: 81-66-152-7160
Fax: 81-66-152-9310

**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 or
82-2-558-5934

**Malaysia - Kuala Lumpur**
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

**Malaysia - Penang**
Tel: 60-4-227-8870
Fax: 60-4-227-4068

**Philippines - Manila**
Tel: 63-2-634-9065
Fax: 63-2-634-9069

**Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

**Taiwan - Hsin Chu**
Tel: 886-3-5778-366
Fax: 886-3-5770-955

**Taiwan - Kaohsiung**
Tel: 886-7-536-4818
Fax: 886-7-330-9305

**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

11/29/11